University of Alberta

Coding Techniques to Reduce Material Saturation in Holographic Data Storage

by

Seth William Phillips

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Communications

Department of Electrical and Computer Engineering

Spring 2014
Edmonton, Alberta

To Marissa

Abstract

Holographic data storage (HDS) is an emerging data storage technology that has received attention due to a high theoretical data capacity, fast readout times, and a potentially long lifetime of the recording materials. The work presented in this thesis was undertaken to solve one of the technical impediments preventing the widespread use of HDS, the occurrence of large concentrations of power in recorded holograms. Such peak values of optical power cause the medium to saturate during the recording process. As a result, the most significant portions of the hologram are not recorded accurately, and on readout, saturated recordings are not reconstructed correctly.

In the implementation of HDS considered in this thesis, data is organized into an array of pixels using hybrid ternary modulation that contains an OFF-pixel and two different ON-pixels that are differentiated by their phase terms. The Fourier transform of this data array is created optically and the image of the Fourier transform is recorded holographically.

This thesis presents a two-step coding technique that decreases the likelihood and severity of peaks in encoded holograms. In the first step, sparsity, the proportion of OFF-pixels in the array, is increased, which decreases the total power in the encoded array. In the second step, phase masks are used

to alter the phase of ON-pixels to decrease periodic content in the data array. This reduces the likelihood of an encoded array containing large peak values at any point in the Fourier domain.

Analysis is presented for the sparsity encoding which demonstrates the worst-case sparsity for certain system parameters. The performance of both the sparsity encoding and phase masking procedure are tested with numerical simulations. The results of these simulations indicate that these encoding techniques effectively inhibit the occurrence of large intensity peaks the holograms of encoded arrays.

Contents

List of Tables

List of Figures

Table of Acronyms

CCD	Charge Coupled Device
CDF	Cumulative Distribution Function
DC	Direct Current (refers to the zero frequency term)
DFT	Discrete Fourier Transform
DTFT	Discrete-Time Fourier Transform
FFT	Fast Fourier Transform
FT	Fourier Transform
GF(N)	Denotes the Galois Field of N elements
HDS	Holographic Data Storage
HTM	Hybrid Ternary Modulation
LCD	Liquid Crystal Display
PDF	Probability Density Function
SLM	Spatial Light Modulator

Chapter 1

Introduction

Within the field of information storage, holographic data storage (HDS) is an emerging technology that promises high storage density, fast data readout, and long material lifetimes [2, 3]. A few technical problems exist that inhibit the practicality of HDS systems. The work presented in this thesis seeks to resolve one of these problems: the occurrence of spectral peaks in recorded data arrays.

In a holographic data storage system, information is recorded as a hologram in an appropriate medium. Chapter 2 begins with a brief overview of the history of holography. The fundamental concepts underlying holographic data storage are then presented. Additionally, the modulation methods and system configurations are detailed.

A problem currently facing holographic storage systems is material saturation. This occurs if the optical intensity of any point in the recorded image exceeds the saturation threshold value of the medium. On readout, saturated recordings are reconstructed improperly, which leads to errors in the data array at the decoder.

One method that reduces the optical intensity of recorded holograms is sparsity encoding. Chapter 3 first demonstrates an algorithm that converts binary data into ternary and follows with guided scrambling as a method that can be used to introduce sparsity to ternary data. A ternary alphabet is used because it allows more information to be stored per array compared to

binary modulation, and hybrid amplitude/phase modulation provides inherent benefits to the Fourier domain properties of recorded arrays, as detailed in Chapter 2. Analytical results determine the worst-case sparsity using guided scrambling, and the number of worst cases is demonstrated.

Selective phase masking, a process for mitigating material saturation, is detailed in Chapter 4. Previous studies have shown that phase masking reduces the Fourier domain peak at DC. In those studies, the phase mask was implemented as a static optical device which introduced new difficulties such as diffraction effects and stringent requirements for optical alignment. In this thesis, phase masking is studied at the encoding level for data arrays using ternary modulation. Since a phase mask implemented during encoding is not a physical device in the optical path, diffraction effects and alignment problems are avoided. Furthermore, multiple phase masks can be used, which affords the encoder a choice among a set of masked variants (including the unmasked array) to represent a data array. This choice decreases the likelihood of an encoded data array containing Fourier domain peaks.

Simulation results are presented for systems using selective phase masking. An evolutionary algorithm was designed to generate data arrays with Fourier domain peaks in all masked and unmasked cases. This algorithm was then used to determine the effect of using various numbers of phase masks. A decreasing trend in the magnitude of Fourier domain peaks was observed as the number of phase masks available to the encoder increases. This algorithm was also used to test selective phase masking for arrays of various size. It was found that as the size of the array increases, the relative magnitude of Fourier transform peaks decreases, which indicates that the number of masks required to keep the occurrence of large Fourier domain peaks acceptably low does not increase when using arrays that are larger than the ones studied in this thesis.

Additionally in Chapter 4, the selective use of interleavers is detailed and studied. It was determined that adding interleavers to a selective phase masking system is less beneficial than adding additional masks.

In Chapters 3 and 4, simulation results are presented that indicate a decrease to the average and maximum peak values in the Fourier domain of data arrays. These results are not translated into a final bit error rate improvement

because doing so would require the selection of several system parameters to be made. The results presented here are intended to serve to as evidence of the utility of these techniques in general, so computing a resulting bit error rate for a specific system runs counter to this goal.

Chapter 5 presents an example that incorporates the techniques presented in this thesis: conversion from binary data to ternary form, then sparse guided scrambling and selective phase masking. The example begins with binary data that is converted into its ternary representation. The ternary data is made sparse using guided scrambling, and this sparse data is mapped to an array. Next, selective phase masking is used to generate a masked array that contains no significant peak values in its Fourier domain representation. At each step, the data and its Fourier domain representation are plotted to illustrate the benefit of each step in the coding procedure. This example demonstrates that arrays can be encoded using the techniques developed in this thesis to record ternary data without large Fourier domain peaks.

Finally, in Chapter 6, the thesis concludes with a summary of the techniques presented herein, and a discussion of future work. With the analytical and simulated results, it can be concluded that guided scrambling works well to introduce sparsity to the data array, and that selective phase masking effectively reduces the peak values in the Fourier domain of the encoded data array. These techniques result in data arrays that are less likely to saturate a recording medium, which can yield higher reliability for a given material as well as allow the use of new materials that have relatively low saturation thresholds.

In this thesis, several novel contributions are presented. For encoding, the binary to ternary conversion with subsequent guided scrambling for sparsity is a new technique. Also, guided scrambling with a set of ternary symbols has not previously been investigated, and the proofs that describe the worst-case sparsity are novel. Although phase masking has been studied in the past, the implementation of phase masks at the coding step is a new technique for this application. Additionally, the use of multiple phase masks, and the evolutionary algorithm simulations that indicate the benefit of using multiple masks are original contributions.

Chapter 2

Background

2.1 Holography

Holography is a method of recording images within the volume of a recording medium. The technique was invented in 1948 by Dennis Gabor[1]. This discovery would earn him the 1971 Nobel Prize in Physics.

The general principle governing holography is the ability for a holographic recording medium to record an interference pattern within its chemical structure. The recorded interference pattern is created by superimposing two coherent light waves upon one another. Once the recording is made, either of the waves can be imposed on the holographic material to reconstruct its counterpart.

Gabor's original system was devised as a method of microscopy. In his setup, he sought to produce a beam of electrons brought into focus at a point. From this point the beam would expand as a coherent, point source wave, which he called the primary wave. He would then place an object at some distance from the focus such that the object was fully illuminated by the primary wave but did not completely block the primary wave. The wave that is transmitted by the object he called the secondary wave. A photographic plate would be placed some distance from the object, which would record the interference pattern between the primary and secondary waves.

He found that applying the primary wave to the recording did, in fact,

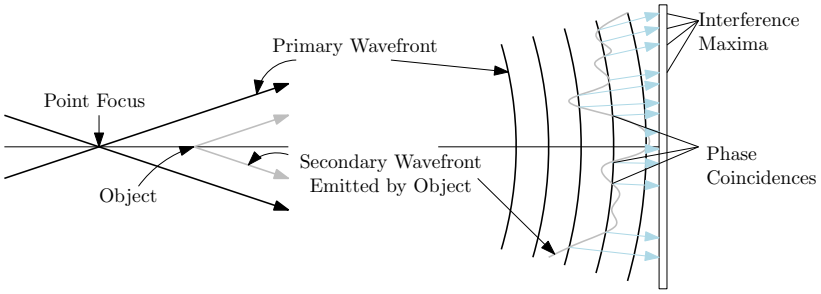


Figure 2.1: System diagram of Gabor's original holographic process [1].

replicate the secondary wave, complete with depth, which he noted:

> It is a striking property of these diagrams that they constitute records of three-dimensional as well as of plane objects. One plane after another of extended objects can be observed in the microscope, just as if the object were really in position.[1]

Gabor later replicated this experiment with monochromatic light sources[4].

As research progressed into improving the holographic recording process, new system configurations were developed. The idea of placing the primary and second waves on different optical paths was first introduced by Emmett Leith and Juris Upatnieks [5]. Further work by Leith and Upatnieks [6, 7] expanded upon this insight by designing new geometries for the optical system.

In [7] the authors illuminated three-dimensional objects with laser light. The reflected portion of the light was then used as the secondary wavefront (now called the object beam). Additionally, they placed a mirror in the scene with the three-dimensional objects. The laser light reflected by the mirror provided the primary wavefront (now known as the reference beam). With the optical system oriented in such a way that the holographic material is struck by the reference beam, a hologram is recorded with the depiction of the three-dimensional object as seen from the perspective of the holographic material.

The reference beam needs to interfere with the object beam in order to record the hologram. Since the object beam is now light reflected from a three-dimensional structure, the holographic material will be struck by the object beam if it is in a location that receives any of the reflected light. Since

the reference beam is a plane wave reflected from a mirror, all that is necessary for interference to occur is for mirror to be oriented in such a way that the reference beam impinges upon the holographic material. This will ensure that an interference pattern is formed within the holographic material.

Once the hologram is recorded, reconstruction of the object beam is possible by illuminating the holographic material with a duplicate of the reference beam. Since the reconstructed object beam is an accurate replica of the original object beam, the hologram will have all the properties of the original wavefront. This means that every ray that impinged upon the recording medium will be emulated with its amplitude and direction intact. This effectively treats the holographic material itself as a window through which the entire recorded, three-dimension scene can be observed. A moving observer will notice parallax between distant and near field objects in the recording.

2.2 Holographic Materials

The holographic medium itself is obviously very important to any holographic system. Several varieties of material have been utilized in systems described in the literature. The two main material classes used are inorganic crystals (particularly lithium niobate [8]) and organic materials [9]. The fundamental requirement for a material to be suitable for holography is that it must be responsive to light. Both organic and inorganic materials are capable of providing refractive index modulation due to exposure to light. The physical processes a specific material undergoes in the formation of the refractive index modulation differs depending on its chemical composition. For instance, an inorganic crystal may store the interference pattern as changes in its electron distribution [10], whereas an organic monomer/polymer material can store the hologram through polymerization of the monomers [11, 12] or alignment of polymer chains [13].

There also exist additional material requirements that any substance should meet if it is to be used not just for holography but for holographic data storage. Firstly, the material should have high sensitivity, where sensitivity describes the change in refractive index due to some level of exposure [2]. A high sen-

sitivity material will be able to record holograms more quickly or record with lower overall power, which would in turn allow more holograms to be recorded within the same medium. Secondly, the material should exhibit high stability, which is to say that the recording should not degrade over time. Generally there is a trade off in materials between stability and sensitivity, in that a highly sensitive material is usually not as stable as a less sensitive one, and vice versa. Finally, the material should be available in thick samples, on the mm-cm range. This is due to holography being a volumetric process, so there are inherent advantages to using a thick material as opposed to a thin film, however thin film holography is an active area of research [14, 15, 16].

Historically, the most often used material for holographic storage is lithium niobate ($LiNbO_3$), an inorganic crystal. One property of lithium niobate is that it undergoes a change in its refractive index when exposed to light [17]. In some optical applications this property is disadvantageous, however, it is precisely what is required for a material to serve as a holographic medium. Shortly after the discovery of this property, lithium niobate was demonstrated as a viable medium for storing holograms [8]. From that point onward, it served as the backbone material for nearly every demonstration of holographic storage. However, $LiNbO_3$ has a few drawbacks, namely its low sensitivity [18]. This low sensitivity results in long write times, and a low overall capacity when compared to the theoretical limits of the technology. These drawbacks have led to a search for more viable materials.

The alternate class of materials, organic polymers and monomers, is currently the focus of many research efforts [13, 19, 20]. The major advantage organic materials have over inorganic crystals is that organic compounds are both easier and less expensive to produce. Also, organic materials are capable of high sensitivities [2]. However, these materials introduce new issues that are not found with inorganic crystals. For one, the physical changes which record the hologram can so severely alter the material that warping can be introduced to the holographic medium. For a system relying on angle multiplexing of the reference beam, losing control over the precise angle of the medium is a serious problem. Also, organic films are easy to produce with spin-coating, but creating a thick material, with constant optical properties throughout its

bulk, is more challenging.

The lack of a truly ideal material that satisfies all the requirements for holographic storage is one of the main hindrances to the commercialization of the technology. There are research efforts underway which will hopefully yield a viable candidate material, from either glass-like polymers [21] or composite materials [18]. The work presented in this thesis is independent of the recording material, and is valid for current holographic materials as well as more ideal materials that may be developed in the future. Additionally, the coding techniques presented in this thesis reduce the likelihood of material saturation which enables less ideal recording materials to be reliably used as holographic storage media.

2.3 Holographic Data Storage

The premise underlying holographic data storage is the utilization of a holographic medium to record images that contain an organized data structure, rather than a three dimensional image of a scene. This is done by forming the data structure with the object beam, then recording its interference with a plane wave reference beam. Retrieval of this data structure is then possible by once again applying the reference beam used to create the holographic recording to the material [22]. Readout of the stored data is then possible through some method which is dependent on the nature of the data structure chosen for the storage system.

The simplest type of data structure is an array of ON and OFF data pixels. Each element in the array can correspond to either a one or a zero depending on whether or not the object beam is ON or OFF at that location. This modulation scheme can be implemented with something as simple as a transparency that contains a printed image of the data array, to a liquid crystal display that produces the data array by turning some pixels ON or OFF. Once the data has been encoded into the object beam wavefront, it is impinged upon the holographic medium and interfered with by a chosen reference beam, causing the medium to record a hologram that represents the data.

After the recording is completed the object beam is reproduced by applying

the reference beam to the medium. The object wavefront is then decoded by some method such as projecting the reconstructed wavefront onto an image sensor. Signal and image processing techniques can then be used to extract the data represented by the object beam. For instance, if data was encoded serially into the array, recovery of the data would consist of scrolling serially through the image and interpreting light and dark pixels as ones or zeros with respect to the modulation scheme chosen.

In most systems described in the recent literature, data is represented, optically, through the use of a spatial light modulator (SLM). An SLM is, in the simplest case, an LCD without a light source. In the holographic data storage system a laser will serve as the light source, and the SLM will be used either to pass or to block some of the light, thereby depicting the data as a wavefront. Readout is usually performed by a charge coupled device (CCD). The CCD is similar to the component in a digital camera that converts the image passed through the lens and shutter into an electrical signal to be stored as a digital image. It is used in exactly the same manner in holographic storage systems. The reconstructed data wave front is displayed onto the CCD, which converts the pattern of incoming photons into an electrical signal, which is then analyzed to recover the data.

2.4 Recording in the Fourier Domain

Despite its simplicity, directly recording the data array introduces a few problems. Most notably, any material defect in the holographic medium will result in corruption of the reconstructed wave front. When the data array is recorded directly the corrupted wave front will likely cause errors in the data itself. Fortunately, alternatives to direct recording of data arrays exist. The most common form discussed in the literature is known as the 4-f configuration [2]. This system consists of using two convex lenses, with one placed between the SLM and the holographic medium and the other placed between the medium and the CCD. The spacings between the SLM and first lens, first lens and holographic medium, holographic medium and second lens, second lens and CCD are all equal to the focal lengths of the lenses used. For convenience, it

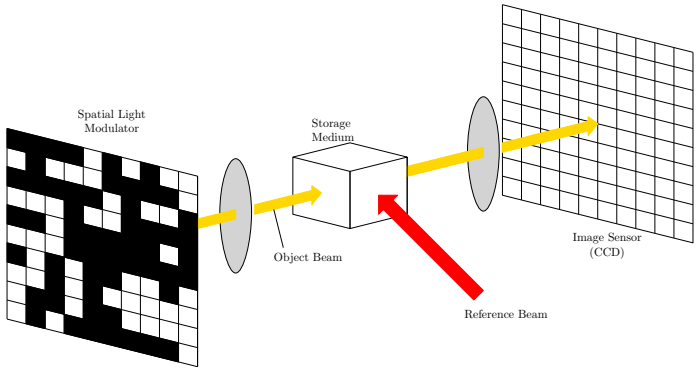


Figure 2.2: A simple diagram of a 4-f system configuration [2]

is generally assumed that the lenses are symmetrical and identical, meaning all focal lengths in the optical system are equal to one another. With these assumptions the distance between the SLM and CCD will be four focal lengths, hence the 4-f designation of the system. A diagram of this type of system is given in Figure 2.2.

In this system, the first lens optically converts the data array image into its two-dimensional spatial frequency representation, or its Fourier transform [23]. This is a well-known property of lenses operating on an image generated by coherent light, which will be further discussed in section 2.6. By the same physical principle, the second lens converts the reconstructed hologram into its Fourier transform. Since the recorded hologram and the original data array are Fourier pairs, this is in effect the inverse Fourier transform, which recovers the data array from the stored hologram. The advantage here is that any material defects in the medium will now cause a relatively mild degradation of the whole reconstructed image rather than more severe noise localized to a few data pixels.

Recording the Fourier transform presents some drawbacks. The most serious of these is the fact that the Fourier transform of a data array can focus a significant fraction of the optical power in the data array to small areas. For instance, the center point in the spatial frequency representation encodes the DC, or zero frequency, component of the data array. In the case of a data array consisting of ON and OFF pixels the DC component will be equivalent to the total power in the array. A very high intensity peak in the hologram will

result in very poor performance of the holographic data storage system due to saturation of the medium. This is explained in greater detail in section 2.7.

Peaks in the spatial frequency image can also be the result of periodic structures within the data. The two-dimensional spatial Fourier transform of a plane wave image has two peaks that are equidistant from and fall on a line that includes the central point of the image. The distance from the DC location to either peak is proportional to the spatial frequency of the plane wave, and the orientation of the peaks about the center represents the direction of the wave in the image plane. Because of this property of the Fourier transform, any periodicity in the data array will also generate significant peaks in the hologram.

Developing methods for reduction of these peaks is one of the main goals of this research. In chapter 4, selective phase masking is introduced and detailed. This method effectively reduces the DC component to zero in most cases, and also disrupts periodic structures within an array that has large Fourier-domain peaks at higher frequencies. To this end, in Chapter 3, sparse guided scrambling is detailed. This type of encoding reduces the severity of Fourier-domain peaks by limiting the number of ON-pixels in a data array, thereby providing a limit to the total optical power.

2.5 Modulation Methods

As with any communication system, the precise manner in which information is represented in the holographic storage device is dependant on the modulation scheme that is employed. The modulation schemes available depend on the physics of the channel as well as the modulating and demodulating apparatuses. Data must be represented such that symbols with differing values are discernable upon readout. In the case of holographic storage by means of an SLM and CCD, modulation is most often limited to the amplitude and phase of the object wave at different physical locations.

The following discussion of modulation schemes describes the manner in which the SLM forms a set of symbols to represent the data in the storage system. The multiplexing methods and specific nature of the reference beams

are not considered so that more attention can be given to the alterations performed on the object beam in each system.

2.5.1 Amplitude Modulation

In traditional communications, amplitude modulation refers to the act of transmitting a signal as a change in amplitude of some carrier wave. In holographic storage scenarios the term essentially has the same meaning. The laser light source provides the carrier wave, and the SLM performs the modulation. In the typical case the carrier wave is multiplied by the signal to produce the modulation. For the holographic storage case, the multiplication is carried out as well, but the signal is now the SLM that can have, for instance, binary values. That is to say, for a binary one the SLM passes the light unperturbed, and for a binary zero, the SLM blocks the light completely. Thus, the multiplication is carried out with the SLM representing an array of ones and zeros.

It is also possible to store more than one bit per pixel by using gray-scale amplitude-modulated arrays [24]. In this scheme, the data page contains more than two amplitude levels. This provides multiple symbols for encoding the data, which allows each symbol to carry more than one bit of data.

Recording a hologram with multiple brightness levels is not overly complicated. Some SLMs can provide multiple brightness levels intrinsically. With these, the task of recording a hologram with multiple amplitude levels is straightforward. If a binary SLM is used then it is still possible to record a gray-scale array. To do this the exposure time is altered for individual pixels such that some are recorded more strongly than others [25]. For example, if a four-level array is desired, three levels of ON pixel brightness are required, with the fourth level being the OFF state. The exposure time is then subdivided into 3 time slots. Pixels which are to be the least bright ON state are recorded for the first time slot, the intermediate state are recorded for the first two time slots, and the brightest state are recorded during all three slots. With this procedure, the recorded array contains ON pixels of three brightness levels as well as the OFF state.

Demodulating a gray-scale array is much more difficult than the simple

scheme required for binary amplitude modulation. Typically, signal dependent effects will make simple threshold detection unreliable for determining the differing gray levels. As shown in [26], the page diffraction efficiency[1] is dependent on the number of holograms multiplexed within a given material, but is independent of the proportion of ON-pixels in the array. However, since the reference beam ideally diffracts only to ON-pixel locations the diffraction efficiency per pixel is dependent on the number of ON-pixels in the array. An array with fewer ON pixels will reconstruct each pixel more brightly than one with a greater number of ON pixels. This variation in reconstructed pixel intensities causes serious problems if detection thresholds are set for a system prior to reading out the hologram. To circumvent this, demodulating makes use of intensity variations within an array itself to determine the thresholds for detection.

In [24], the authors used several demodulating methods to reproduce data represented in a gray-scale modulated array. For smaller block sizes it is possible to sort pixels by brightness and demodulate accordingly, if a constraint is enforced such that a known proportion of each gray-level is used. In [24], an array was recorded with three total amplitude levels (0, 1 and 2 in order of increasing intensity). A 15:12 code was used to map 15 user bits to 12 pixels, with the constraint that four of the pixels would be level-0, four would be level-1, and the final four would be level-2 for each data block. At the demodulator, each block is sorted by brightness, so that the four brightest are decoded as the level-2 pixels, the four dimmest are decoded as the level-0 pixels, and the remaining four are the level-1 pixels.

Amplitude modulation provides the simplest detection requirements of all modulation schemes used in holographic storage. Amplitude modulation benefits from simple modulating procedures as well. On the other hand, both binary and gray-scale amplitude-modulated arrays suffer from very large peaks at DC in their Fourier transforms. This causes the holographic material to become saturated and reduces the overall storage capacity. Alternative modulation schemes, described in the following sections, do not have such problematic

[1]The total proportion of reference beam power that is diffracted to form the hologram on readout

DC components, but introduce further complexity. Techniques for mitigating the DC peak are described in chapter 4.

2.5.2 Phase Modulation

Although amplitude modulation is perhaps the simplest modulation scheme, it is not the only modulation method that is used. Advances in liquid crystal display technologies have made it possible to alter the phase of pixels independently in spatial light modulators. This has led to researchers using phase modulation as a method for encoding data into arrays for holographic storage [27, 28, 29, 30].

Phase modulation has several advantages over the amplitude modulation techniques described above. As described in section 2.4, data images are typically recorded as their Fourier transforms, and amplitude-modulated data pages have a strong DC component. With phase modulation, it is possible to reduce the DC peak through destructive interference of the out-of-phase pixels.

In phase modulation, zeros and ones are represented as 0 and π phase shifts respectively. Since the zero frequency component in the center of the Fourier image is the DC value of the data image, it is proportional to the sum of the data array. In this case, the 0 and π phase shifted pixels cancel each other out, so that for a perfectly balanced array (i.e. an array in which the number of 0 pixels exactly equals the number of π pixels) the DC component is zero.

In amplitude modulation, the DC component is the most significant peak in the hologram. However, the total power contained in the Fourier image is equivalent to the power of the laser source scaled by the number of ON pixels that are in the data array. In phase modulated data arrays, however, all pixels are ON pixels, since only phase shifts used to encode the data. Because of this, the total power in a phase-modulated array is higher than that of an amplitude modulated array for all possible input data with the exception of the all-ON array, in which case both modulation schemes will result in equal power in the array.

The fact that a phase-modulated array will contain more power in its holo-

gram than an amplitude-modulated array carrying identical data is important to consider while attempting to reduce Fourier-domain intensity peaks of the recording. Since this power is not concentrated at DC, in relatively balanced arrays, it must be located elsewhere in the Fourier image. Because of this, it is possible for a phase-modulated array to produce very large peaks elsewhere in its Fourier image if the data array contains highly periodic pixel patterns.

Reading data from the reconstructed array is more challenging for phase modulated arrays than it is for amplitude modulated arrays. This is due to the fact that the CCDs used to read the data are unable to detect the phase of an incoming light wave, but can only detect intensity. To recover the data from a phase modulated array it is necessary to impinge a plane wave upon the reconstructed data image. This plane wave should be in phase with one of the data symbols. This will cause constructive interference at the points which are in phase, and destructive interference at the points which are out of phase with the plane wave. In this way, the CCD is able to discriminate a binary zero from a binary one via the variation of intensity. If the intensity of the plane wave closely matches that of the reconstructed wave, then the out of phase pixels reduce nearly to zero and the phase matched pixels are made brighter.

Several methods for producing the interference plane wave have been proposed. One solution, known as the real time holographic interferometric method [28], involves using the SLM to generate the interference wave. This method essentially creates an object beam of constant phase to be displayed concurrently with the reconstructed hologram, which is read out with the appropriate reference beam. The constant phase interference wave will traverse the holographic system to reach the CCD. Since this method involves displaying both an object and reference beam on the holographic material, it is not suitable for re-writable systems. In such systems the read out process would be identical to the write process, and as a result, each read out would record the constant phase wave on top of the data hologram. This method would work well for systems with permanently recorded holograms or systems employing gated recording.

2.5.3 Hybrid Ternary Modulation

Another modulation method for holographic storage is hybrid ternary modulation (HTM) [31, 32]. The premise for HTM is to combine both amplitude and phase modulation into one modulation scheme. The motivation for this is that it should be possible to achieve the advantageous Fourier properties of a phase-modulated array while maintaining most of the positive attributes of an amplitude-modulated array, all while increasing the total capacity of the data storage system.

In its initial presentation, HTM was intended to be a method for reducing the previously mentioned DC peaks in the Fourier image [31]. In that publication, the authors constructed a system that would essentially use amplitude modulation to represent data, but the ON pixels would not all have the same phase state. They proposed introducing a π phase shift to roughly half of the ON pixels. With this approach, the magnitude of the DC peak should be reduced, given that it is determined by the sum of the values of the pixels in the data array. Through the Fourier transform, the π shifted ON pixels destructively interfere with the unshifted ON pixels, resulting in the reduced DC value. When the number of phase-shifted pixels is equivalent to the number of unshifted pixels, the DC value is at its minimum.

As noted previously, the phase shifts for each ON pixel has no effect in read-out because CCDs can only detect optical intensity and not the phase of the light wave. So, for data read-out, the detector can function exactly as one designed for an amplitude-modulated data array.

Since HTM provides three different states for the pixels in the data array, a natural extensions is to use all of those states to represent data[33]. The premise is that a π phase shifted ON pixel and a zero phase shifted ON pixel will represent different information content. Doing so provides a ternary data array rather than a binary one. The ternary array is capable of encoding more than one data bit per pixel, so this results in an overall capacity increase.

The main drawback to using HTM is that it results in a significant increase in the complexity of the overall system. In the original HTM system the phase did not carry any information, so it did not need to be detected. If,

however, data is encoded in the phase of the ON pixels then the detector must be able to recover the phase if the input information is to be reconstructed correctly. Phase detection is possible with interferometric techniques, but this does require a high level of precision.

The general readout method, as described in [33], is a two step process. In the first step the detector reads the recorded hologram itself. This allows the detector to differentiate the ON pixels from the OFF pixels. The detector then reads the recorded array and a constant-phase reference wave simultaneously. This produces an interference pattern on the CCD in which the ON pixels that are in-phase with the reference wave will constructively interfere, causing doubly bright pixels. The OFF pixels will be illuminated by the reference wave, and appear as less bright pixels to the detector. Finally, the ON pixels that are out of phase with the reference wave will interfere destructively, and appear as minimally bright pixels. This assumes the reference wave is of an intensity equal to the intensity of the recorded hologram's pixels, and that the reference wave is exactly in phase with one set of ON pixels and exactly out of phase with another. Maintaining these conditions is likely to be the primary challenge in the physical detection of an HTM data array.

The origin of the reference wave is another matter that will add complexity to the HTM system. In [33], the authors describe a method of recording a constant phase data array in the holographic medium. This constant phase array is then reconstructed at the same time as the data array that is being read during the second step of the detection process. Generating the reference wave in this manner would seem to be a good method for ensuring that it has equal amplitude and is phase-matched with one set of the ON pixels in the data array because both arrays are generated by the same laser source, and both have phase alterations set by the same process (the modulating SLM). Reconstructing two holograms simultaneously, however, requires generating two separate reference waves simultaneously, which results in increased complexity in the system.

For the purposes of this thesis, error-free hybrid ternary modulation and demodulation will be assumed because the focus of this work is encoding HTM arrays in a manner so as to improve the characteristics of their frequency

domain representations.

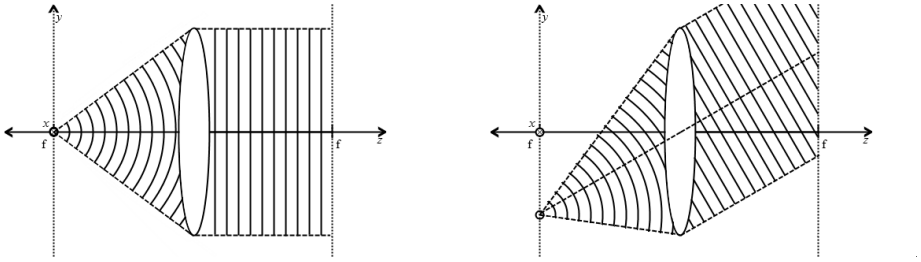
2.6 Optical Fourier Transform

In section 2.4 it was noted that convex lenses are used to create the Fourier transform of an image. This property requires that the input light is coherent, which is satisfied in the holographic storage case by the laser light source. With coherent light, the optical rays exhibit predictable interference effects which are necessary for the Fourier transform to be carried out. It can be shown mathematically that the output focal plane of a lens will be proportional to the Fourier transform of the input focal plane image under the conditions described above (see chapter 4 - section 4.2 of [23]). However, a more intuitively accessible, geometric approach will be explained below. This description follows the treatment provided in [34].

2.6.1 Conceptual Analysis of Lenses and Point Sources

The Fourier property of lenses is evident given the fact that a point-source, spherical light wave at the input focal plane is transformed into a plane wave after passing through the lens. This also works in reverse, in that a plane wave passing through the lens is focused to a point at the focal plane. In the case of a point source producing a plane wave, the plane wave is directed along the line from the point source through the center of the lens. For a point source lying on the optical axis, the plane wave is directed along the optical axis. For a point source lying in the focal plane, but some distance off the optical axis, the plane wave is directed askew to the optical axis.

This principle is illustrated in Figure 2.3. The y-axis is directed vertically, the z-axis is horizontal (and also represents the optical axis for these systems), and the x-axis is directed into the page. A point source is shown for two different orientations: the on-axis case, and the off-axis case (the axis being the optical axis). In both cases, the point source is depicted as a circle, and the resulting wavefront is shown as a series of dark lines. On the left side of the lens the point source produces a spherical wave front. In this and subsequent



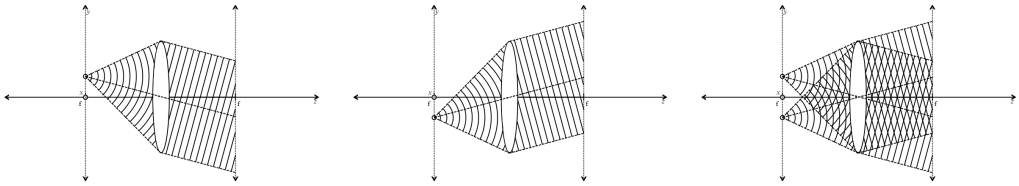
(a) on-axis (b) off-axis

Figure 2.3: Lens Focusing for two points, one on-axis, and one off-axis

figures, the portion of the spherical wavefront not impinging upon the lens is omitted for clarity. In both cases, the lens shapes the spherical wavefront into a plane wave. The plane wave is depicted as a series of dark, parallel lines which indicate points of zero phase. The two vertical, dotted lines on either side of the lens represent the focal planes.

An on-axis point source is shown in 2.3a. This point source produces a spherical wave front, a small portion of which is incident on the lens. The lens then focuses this spherical wave into a plane wave. Since the point source lies on the optical axis, the plane wave is directed along the optical axis. When the edge effects of the lens are neglected, the result is an area of uniform amplitude and uniform phase at the back focal plane. The uniform phase of the back focal plane image is indicated by the wavefront being parallel to the focal plane. This shows that, at any two points on the back focal plane, the light wave is in phase.

In Figure 2.3b, however, the point source has been moved some distance from the optical axis. Because of this positional translation, the resulting plane wave is now directed along a line that intersects the optical axis in the center of the lens. The result of this, in the back focal plane, is that the focused image is still of uniform amplitude, but there is a phase shift along the vertical axis as shown in the figure. The phase shift is evident because the wavefronts are not parallel to the focal plane. Since the dark lines represent the zero-phase points of the plane wave, the points at which the dark lines intersect the focal plane represent zero-phase points in the focal plane. The phase gradation is evident by inspecting the back focal plane along the vertical



(a) Point $+a$ (b) Point $-a$ (c) Both Points

Figure 2.4: Lens with two point sources

axis. Beginning at one zero-phase point, it is clear that moving vertically will show that the phase of the light wave progresses linearly from 0 to 2π, which is the phase at the subsequent zero-phase point, and then repeats this phase change periodically. This is due to the fact that the phase has a period of 2π, and therefore any phase value ϕ can be equivalently expressed as ϕ $(\mathbf{mod} 2\pi)$.

Figure 2.4 shows a lens with two point sources that are equidistant from the optical axis and both lie on the y-axis. Figure 2.4a shows one of the sources with a positive shift in the vertical direction, and Figure 2.4b shows the other source which has a negative shift. The plane wave in figure 2.4b is similar in form to the one from Figure 2.3b. As before, scrolling vertically in the back focal plane, in this case, will reveal that the phase of the image cycles linearly through 0 to 2π. However, scrolling vertically in the back focal plane of Figure 2.4a will reveal that its image cycles linearly from 2π to 0. What is important about this is that the phase values of the two back-focal-plane images, originating from the two point sources, are exact reversals of one another. As a result, the image at the back focal plane of Figure 2.4c will be a plane wave with high-intensity maxima where the two phase components are equal (when they are both 0 or when they are both π) and low-intensity minima when the two phase components are π radians out of phase (when one is $\pi/2$ and the other is $3\pi/2$).

Figure 2.4c shows the two waves overlapped in the back focal plane. The two plane waves are in phase at the points where the wavefronts meet; these points indicate a 0 phase shift relative to the original point source. The waves are also in phase at the midway points between each of these 0 phase points; at

these points both wavefronts have a π phase shift. Also, note that Figure 2.4 is equally accurate when the propagation direction of the light is right to left. A plane wave in the front focal plane will be converted to two points in the back focal plane.

The above discussion provides hints of the Fourier property of a lens. It is illustrated that a plane wave on one side of the lens will be focused to two points on the other. This is analogous to the Fourier transform of a one dimensional sinusoid, which is two Dirac delta functions spaced equally from the zero frequency point. The essence of the Fourier transform is the representation of a signal by a sum of weighted, complex-exponential basis functions.

In the two-dimensional case, the basis function is a complex plane wave. One way to conceptualize the lens performing the two-dimensional Fourier transform is to realize that every point in an image is converted into a plane wave with its direction determined by the direction to the optical axis from the point in question. Additionally, each of these complex plane waves will be scaled by the value of the point in the front focal plane which produced them. Because of this, the output at the back focal plane is the sum of each complex plane wave generated by each point in the image. This sum of complex plane waves is actually the Fourier transform of the input image.

2.6.2 Mathematical Analysis of Lenses and Point Sources

Mathematically the image in the back focal plane arising from a point source in the front focal plane can be expressed as a complex exponential function. Following the notation of [34], the image of the front focal plane is referenced as $f(x, y)$ and the image in the back focal plane is referred to as $F(d_u, d_v)$. The arguments of F are (d_u, d_v), respectively, to indicate that F is a function of distances. In section 2.7, F is re-introduced as a function of spatial frequencies, and a relation will be drawn between the function of spatial frequencies and the function of distances in the back focal plane. Since the Fourier transform is reversible it is also possible to express $f(x, y)$ as a function of spatial frequencies. However, for this presentation of the optical Fourier transform,

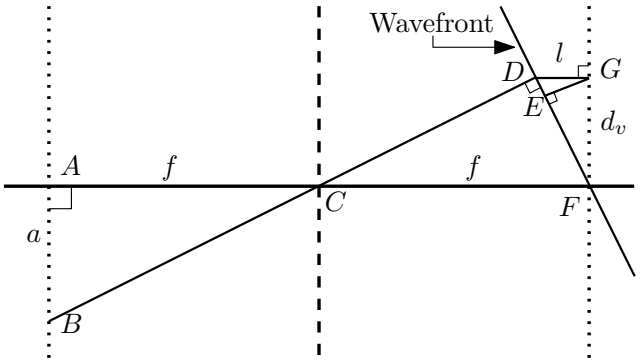


Figure 2.5: A geometric representation of the Fourier lens system.

the image in the front focal plane will be referred to exclusively in terms of distances.

On-Axis Point Sources

For a single point source in the front focal plane $f(x, y) = \delta(x - d_x, y - d_y)$, where d_x, d_y are the distances of the point source in the x and y axes, respectively. Also, $F(d_u, d_v) = e^{i\phi(d_u, d_v)}$, where $\phi(d_u, d_v)$ is the phase factor in the u, v plane. Also, the amplitude of $F(d_u, d_v)$ must equal $\iint f(x, y)$. By the definition of the Dirac delta function, $\iint f(x, y) = 1$ and, therefore, $|F(d_u, d_v)| = 1$. In the above discussion, the phase was discussed intuitively, however, it is necessary to have a mathematical description of it to show that $F(d_u, d_v)$ is indeed the Fourier transform of $f(x, y)$.

Following the work presented in [34], a mathematically simple and intuitively satisfying derivation of the phase factor $\phi(d_u, d_v)$ can be attained through geometry. Figure 2.5 depicts a geometric system representing the Fourier lens system in discussion. As in Figures 2.3 and 2.4, the vertical axis represents the y-axis, the horizontal axis is the z-axis (also the optical axis), and the x-axis is directed into the page.

To determine an expression for the phase of the Fourier image it is necessary to determine the optical path difference between any two points in the Fourier image. The two points chosen in Figure 2.5 are the center point of the back focal plane (F) and a point located a distance d_v from the optical axis

(G). This distance can be approximated as the length, l, of segment DG in Figure 2.5. This approximation is based on the assumption that the rays in the optical system are paraxial, or close enough to the optical axis for the small angle approximation to hold (i.e. $\sin(\theta) = \theta$). The paraxial approximation is valid in this case, and is already employed with discussion of the focal plane of the lens; Non-paraxial rays do not converge at the focal plane, in an effect known as spherical abberation (see [23], Chapter 1, Section 1.2-C). The actual optical path difference would be determined by the length of segment EG; the approximation is that $\overline{DG} \approx \overline{EG}$.

An expression for l is found by examining the three triangles $\triangle ABC$, $\triangle CDF$, and $\triangle DFG$. These three triangles are all similar. $\triangle ABC \sim \triangle CDF$ because $\angle ACB$ and $\angle DCF$ are vertical angles, and therefore congruent. Since both triangles are right, it follows that their third angles must also be congruent, and the triangles are similar. $\triangle CDF \sim \triangle DFG$ because they are both right, and DG and CF are parallel, so the alternate interior angles $\angle GDF$ and $\angle CFD$ are congruent. Therefore $\triangle CDF$ and $\triangle DFG$ have two congruent angles, so their third angles must also be congruent, which makes the triangles similar.

Since the three triangles are similar, ratios of their similar sides are equal, and $l/d_v = a/f$. So the expression for l as a function of a, f, and d_v is:

$$l = \frac{ad_v}{f} \tag{2.1}$$

With the optical path difference known, the phase difference is deduced through the use of properties of the wave nature of light. To do this, the optical path difference is divided by the wavelength of the light, λ, which gives a value in dimensionless units equivalent to the optical path difference in wavelengths. This value is then multiplied by 2π to convert the wavelengths value to radians. This conversion is possible because one wavelength represents 2π radians in terms of phase. The result of this operation is the expression for the phase difference in terms of a, d_v, f, and λ:

$$\phi(d_u, d_v) = \frac{2\pi a d_v}{f\lambda} \tag{2.2}$$

Using Equation 2.2 with the discussion above concerning Figure 2.4, the result of the superposition of the image of two point sources spaced equidistant from the optical axis is mathematically expressible. As mentioned previously, the back focal plane image resulting from an off-axis point source, as in Figure 2.3b, is:

$$F(d_u, d_v) = e^{i\phi(d_u, d_v)} \tag{2.3}$$

Inserting the value for $\phi(d_u, d_v)$ found in Equation 2.2, yields:

$$F(d_u, d_v) = e^{\left(\frac{2\pi a d_v i}{f\lambda}\right)} \tag{2.4}$$

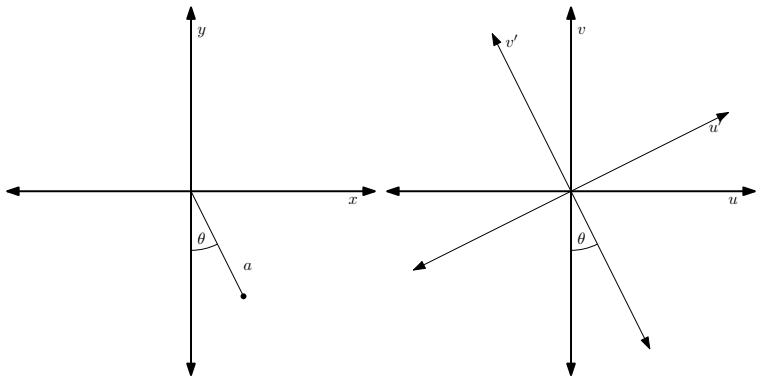
From Equation 2.4, it is clear that the phase gradient in the back focal plane is linear along the v direction. It is also evident that a point source located at $-a$ will produce an image in the back focal plane with a phase gradient exactly opposite the one produced by a point source located at $+a$ (i.e. using a value of $-a$ for a in Equation 2.4 gives the function an equal value but negative exponent, which translates to a phasor with equal but opposite frequency). The superposition of the back focal plane images of a point source a distance a and another a distance $-a$ can be written mathematically as:

$$F(d_u, d_v) = e^{\left(\frac{2\pi a d_v i}{f\lambda}\right)} + e^{\left(\frac{-2\pi a d_v i}{f\lambda}\right)}$$

since the complex exponentials have opposite exponents, their sum will be:

$$F(d_u, d_v) = 2\cos\left(2\pi\frac{a}{f\lambda}d_v\right) \tag{2.5}$$

which is a standard plane-wave. In Section 2.6.1, it was stated that the image at the back focal plane of two point sources in the front focal plane has intensity maxima at locations in which both point sources produce 0 and π phase shifts. Equation 2.5 indicates that $F(d_u, d_v)$ has values of 2 and -2 for these respective phase shifts. The intensity, however, is determined by $|F(d_u, d_v)|^2$, which does have maxima at 0 and π.



(a) x, y plane (b) u, v plane

Figure 2.6: Diagram of axis rotation

Off-Axis Point Sources

Conceptually, it is apparent that rotating the image in the front focal plane should simply produce a corresponding rotation of the image in the back focal plane. That is to say, if the point source discussed above is not located on the y-axis, but instead is located at an arbitrary point in the x, y plane the output will be a phase gradient as described in Equation 2.4, but with a direction dictated by the rotation in the front focal plane.

Describing this mathematically can be done by considering a rotation of the axes, as presented in [34]. A single point source located a distance a from the optical axis, but now at some angle from the y-axis, θ, is shown in Figure 2.6a. Equation 2.4 accurately describes the image in the back focal plane if an arbitrary axis is created with a rotation equal to θ as shown in Figure 2.6b. Using Equation 2.4 with the arbitrary axis gives the following:

$$F(d_u, d_v) = e^{\left(\frac{2\pi a d'_v i}{f\lambda}\right)} \tag{2.6}$$

where d'_v is the distance along the rotated axis v'. This distance is a function of d_u and d_v. Also, due to the rotation, the point source needs to be expressed in terms of its extent in both the x and y directions. These distances are d_x and d_y respectively. From Figure 2.6, the following relations between a, d_x,

and d_y as well as d_u, d_v, and d'_v can be determined:

$$d_x = a\sin(\theta) \tag{2.7}$$

$$d_y = -a\cos(\theta) \tag{2.8}$$

$$d'_v = d_u\sin(\theta) - d_v\cos(\theta) \tag{2.9}$$

Substituting Equation 2.9 into Equation 2.6 gives:

$$F(d_u, d_v) = e^{\left(\frac{2\pi i a(d_u\sin(\theta) - d_v\cos(\theta))}{f\lambda}\right)} \tag{2.10}$$

Rewriting this expression yields:

$$F(d_u, d_v) = e^{\left(\frac{2\pi i[d_u(a\sin(\theta)) + d_v(-a\cos(\theta))]}{f\lambda}\right)} \tag{2.11}$$

Finally, substituting Equations 2.7 and 2.8 into Equation 2.11 gives:

$$F(d_u, d_v) = e^{\left(\frac{2\pi i[d_u d_x + d_v d_y]}{f\lambda}\right)} \tag{2.12}$$

Arbitrary Image as Composite Point Sources

To progress, mathematically, from a single point to an image, it is necessary to note that the lens apparatus is a linear system. This means that the output of several points will be equivalent to the sum of the output of each point separately. This property was discussed in Section 2.6.1, in which the output of two point sources was shown to be the superposition of the output of each point source. Mathematically, this superposition is described by a summation, which is the definition of the linearity property.

Describing an input image as a sum of point sources, and then determining the output of that summation demonstrates the Fourier property of the single lens system. This is the approach taken by the authors in [34], and will be detailed here as well.

Firstly, the input image is described as a double integral of point sources

of varying amplitude:

$$f(x,y) = \iint_{-\infty}^{+\infty} f(d_x, d_y)\delta(x - d_x, y - d_y)\mathrm{d}d_x\mathrm{d}d_y \tag{2.13}$$

The image in the back focal plane is governed by the relation described in Equation 2.12. Each point source in the input image produces a plane wave, with direction and frequency dictated by the orientation and distance of the point source with respect to the optical axis, and amplitude dictated by the amplitude of the point source.

$$F(d_u, d_v) = \iint_{-\infty}^{+\infty} f(d_x, d_y)e^{\left(\frac{2\pi i[d_u d_x + d_v d_y]}{f\lambda}\right)}\mathrm{d}d_x\mathrm{d}d_y \tag{2.14}$$

Lastly, as a matter of notational clarity, d_x, d_y, the distance of each point in the x, y axes, are re-written as x, y, likewise d_u, d_v are written as u, v.

$$F(u,v) = \iint_{-\infty}^{+\infty} f(x,y)e^{\left(\frac{2\pi i[ux+vy]}{f\lambda}\right)}\mathrm{d}x\mathrm{d}y \tag{2.15}$$

This gives the definition of the inverse Fourier transform, due to the exponent being positive. This is not a problem, however, due to the Fourier transform being reversible.

Reexamining the derivation of the exponent reveals that the positive factor is due to the point G in Figure 2.5 lagging point F in phase. If the analysis were done in reverse, but keeping the same convention that point G lags point F in phase, it would reveal that point B leads point A. Thus, the phase term for the front focal plane would be Equation 2.2 multiplied by -1. Following the remainder of the derivation with the negative value of ϕ yields:

$$f(x,y) = \iint_{-\infty}^{+\infty} F(u,v)e^{-\left(\frac{2\pi i[ux+vy]}{f\lambda}\right)}\mathrm{d}u\mathrm{d}v \tag{2.16}$$

which is the definition of the forward Fourier transform.

In reality, the notion of forward and reverse Fourier transform is only arbitrarily connected to the lens system. The matter of the back focal plane being the reverse Fourier transform of the image in the front focal plane is simply due to the axis conventions used in this derivation. The important concept is that the two images are mathematically related as a Fourier transform.

2.6.3 Fourier Optics

The Fourier property of a lens can also be described with standard Fourier optics. Through conventional Fourier theory, an image can be represented by the sum of complex-amplitude harmonic functions of different spatial frequencies. For this analysis, the front focal plane of the lens is referred to as the xy-plane or the $z = 0$ plane.

A harmonic function with spatial frequency ν_x and ν_y in the x and y directions respectively is equivalent to a plane wave directed with angles θ_x and θ_y to the optical axis in the x and y dimensions respectively. This plane wave has the wave-vector $\mathbf{k} = (k_x, k_y, k_z)$ and wavenumber $k = \sqrt{k_x^2 + k_y^2 + k_z^2} = \frac{2\pi}{\lambda}$ where λ is the wavelength of the light.

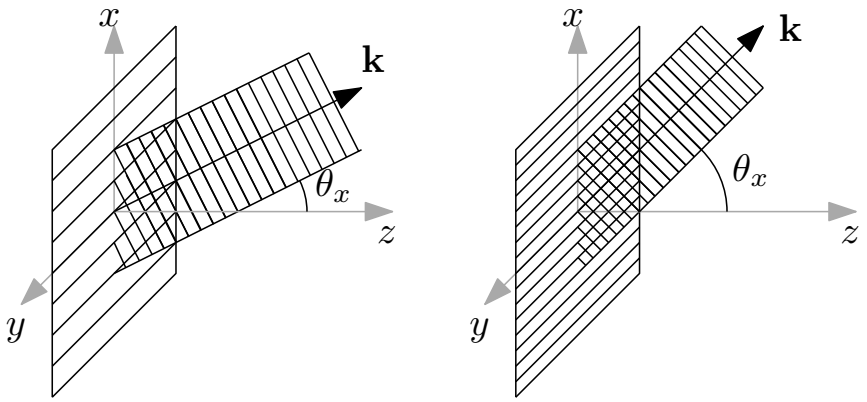
The components of $\mathbf{k}$ are related to the spatial frequencies of the harmonic function in the $z = 0$ plane by:

$$\nu_x = \frac{k_x}{2\pi}$$
$$\nu_y = \frac{k_y}{2\pi}$$

The angles θ_x and θ_y are determined by the x and y components of the wave-vector as:

$$\theta_x = \sin^{-1}\left(\frac{k_x}{k}\right) = \sin^{-1}\left(\lambda\nu_x\right)$$
$$\theta_y = \sin^{-1}\left(\frac{k_y}{k}\right) = \sin^{-1}\left(\lambda\nu_y\right)$$

The relationship between an harmonic function with spatial frequency ν_x in



(a) Plane wave resulting from low ν_x.

(b) Plane wave resulting from high ν_x.

Figure 2.7: Relation between plane waves and spatial frequency in the x, y plane.

the xy-plane and a plane wave with wave vector $\mathbf{k}$ is depicted in Figure 2.7. The harmonic function with a relatively low spatial frequency shown in Figure 2.7a is related to a plane wave that creates a relatively small angle θ_x with the optical axis. Conversely, the harmonic function with a relatively high spatial frequency shown in Figure 2.7b is related to a plane wave that makes a much larger angle with the optical axis.

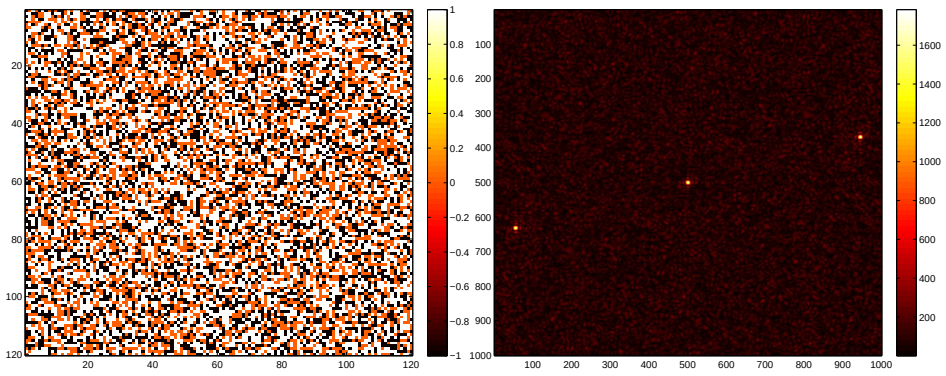
When the plane wave encounters the lens, the wave front is focused to a point, (x, y), in the back focal plane. This point is determined by the angles θ_x and θ_y. For paraxial waves, the coordinates x, y are given by the following relation:

$$x = \theta_x f$$

$$y = \theta_y f$$

where f is the focal length of the lens.

Accordingly, every point in the back focal plane of the lens represents the contribution of its corresponding harmonic function in the image in the front focal plane. Therefore, the image in the back focal plane of the lens is proportional to the two-dimensional Fourier transform of the image in the front focal plane.



(a) Data Array

(b) Fourier Transform of Data Array

Figure 2.8: A Data Array and its Fourier Transform

2.7 Physical Properties of Fourier Images

In a hybrid ternary modulation scheme, as described in section 2.5.3, data is represented with the symbols 1, 0, or -1. Both the 1 and -1 represent ON pixels, however, the -1 indicates an optical phase shift of π radians. When analyzing data arrays it is more convenient to use these logical symbol values, rather than using a value that takes into account the actual laser power of the system. From a coding perspective, using logical units allows isolation of the coding method from the other parameters in the holographic storage system. This isolation allows several coding-based approaches to be directly compared with each other. Additionally, in the analysis of frequency domain properties of data arrays, the data that is represented by the different logical values is not important; the physical transformation processes are carried out by the lens without regard to the data represented by the pixels. An HTM array is shown in Figure 2.8a.

However, to accurately analyze the physical properties of the Fourier transform of a data array, it is necessary to convert the logical units (1, 0, -1) to physical units. This is due to the fact that a Fourier transform for an array with logical units contains values that are dependent upon the logical units rather than physical units. The resulting values in the Fourier transform are, therefore, unrelated to the actual system in review, and peak values in the

transform cannot give an indication as to whether or not the data array will cause the material to saturate. For instance, Figure 2.8b shows the magnitude of the Fourier transform of the data array in Figure 2.8a. The magnitude of the transform has three peaks each of approximately 1700. Without additional information about the physical parameters of the holographic system, this value is meaningless.

However, the logical units of an array are not physically defined (See [23]: Section 2.1). Fortunately, the physical units can be derived from the fact that the total magnitude squared value of $f[m,n]$ represents the power of the data array (where $f[m,n]$ denotes logical value in the m^{th} row and n^{th} column of the data array). To convert the logical-valued data array into a properly scaled array of physical values for any given system, the total magnitude squared value of the data array must equal the power of the object laser beam multiplied by the proportion of ON pixels. With this scaling, it is possible to compare the peak values of the Fourier image to a threshold value to determine if a given data array will saturate the medium. The specific threshold value is a property of the material used for the holographic recording medium.

The scaling factor is constant for all pixel values if it is assumed that the laser power is distributed evenly across the SLM. This is a common assumption that will be applied here. With this assumption, the scaling factor (s_c) is $\sqrt{P_o/M}$, where P_o is the object beam power, and M is the total number of pixels in the array. This scaling factor has units $\sqrt{\text{W}}$. The SLM will simply absorb the power impinging on pixels that are OFF. Multiplying the scaling factor by the data array gives the result $\sum_m \sum_n |s_c \cdot f[m,n]|^2 = P_o \cdot M_o/M$, where M_o is the number of ON pixels. This is the result that is required to properly scale the data array.

With this scaling factor in place, the square of the data array values represents the amount of optical power contained in each pixel. However, holographic material saturation is determined by the amount of energy delivered to a specific area of the medium. It is therefore necessary to derive an expression for the area of each pixel. The area is affected by system parameters, namely the lens focal length, the wavelength of the laser used, and the size of the SLM. Also, to obtain simulation results, the discrete Fourier transform

will be used. The physical area represented by a single point in the transform array is dependent upon the sample spacing chosen for each dimension of both the spatial and transform domains.

2.7.1 Discrete Time Fourier Transform

A continuous Fourier transform of a discrete function is known as the discrete-time Fourier transform (DTFT). The term DTFT will be used to describe the two-dimensional transform of a two dimensional data array. Even though the discrete domain for the data array is spatial, rather than temporal, the DTFT designation is used due to convention. The definition of the two-dimensional DTFT is:

$$F(\nu_u, \nu_v) = \sum_m \sum_n f[m,n] e^{-2\pi i(\nu_u m x_o + \nu_v n y_o)} \tag{2.17}$$

where x_o and y_o are the sample spacings in the x and y directions of the data array respectively, and ν_u, ν_v are spatial frequencies in the u and v directions. Here, ν is used as the argument variable to distinguish it from the expression for F provided in section 2.6, which was a function of distances.

Examination of the units in the DTFT definition reveals that $F(\nu_u, \nu_v)$ has the same units as $f[m,n]$. In the exponent, ν_u and ν_v have units of 1/meters, m and n are unit-less indices, and x_o and y_o have units of meters. Therefore, the argument of the exponential is radians, given by the 2π term, and the exponential term does not alter the units in the equation. So $F(\nu_u, \nu_v)$ has the same $\sqrt{W}$ units as $f[m,n]$.

Because $f[m,n]$ is discrete, $F(\nu_u, \nu_v)$ will be periodic. The periodic nature of $F(\nu_u, \nu_v)$ can be shown by examining the complex exponential summand. Using Euler's formula[2], Equation 2.17 can be re-written as:

$$F(\nu_u, \nu_v) = \sum_m \sum_n f[m,n] \left[\cos(\theta) + i \sin(\theta)\right]$$

where:

$$\theta = -2\pi\nu_u m x_o - 2\pi\nu_v n y_o$$

[2]Euler's Formula: $e^{i\theta} = \cos(\theta) + i\sin(\theta)$

Since cosine and sine are periodic about 2π, values for $F(\nu_u, \nu_v)$ will be identical when the arguments of the cosine and sine functions are incremented or decremented by 2π. Therefore, $F(\nu_u + p, \nu_v + q) = F(\nu_u, \nu_v)$ for some values of p and q. Determining the value for p which results in periodicity with ν_u is accomplished by examining the arguments of the cosine and sine functions:

$$\cos(-2\pi(\nu_u + p)mx_o - 2\pi\nu_v ny_o) + i\sin(-2\pi(\nu_u + p)mx_o - 2\pi\nu_v ny_o)$$
$$\cos(-2\pi\nu_u mx_o - 2\pi pmx_o - 2\pi\nu_v ny_o) + i\sin(-2\pi\nu_u mx_o - 2\pi pmx_o - 2\pi\nu_v ny_o)$$

The change resulting from the addition of p to ν_u is the term $-2\pi pmx_o$ in each argument. When this term is an integer multiple of 2π, $F(\nu_u + p, \nu_v) = F(\nu_u, \nu_v)$. This condition can be written:

$$-2\pi pmx_o = 2\pi k \tag{2.18}$$

where k is some integer. Since the variable m is an arbitrary integer index, its value is irrelevant in this expression. Likewise, the negative sign is not important, as either a positive or negative value for k will satisfy the periodicity requirement. This reduces the condition for periodicity in Equation 2.18 to:

$$px_o = k'$$

where k' is an integer accounting for the removal of m and the negation. This gives a value of p:

$$p = \frac{k'}{x_o} \tag{2.19}$$

This expression indicates that $F\left(\nu_u + \frac{k'}{x_o}, \nu_v\right) = F(\nu_u, \nu_v)$. In other words, $F(\nu_u, \nu_v)$ is periodic in ν_u, with a period of $1/x_o$. A similar derivation can be followed for ν_v which gives $q = {}^{l'}/_{y_o}$, where l' is some integer. This demonstrates that the period of $F(\nu_u, \nu_v)$ in ν_v is $1/y_o$.

As a result, only values of $F(\nu_u, \nu_v)$ for $\nu_u \in \left(\frac{-1}{2x_o}, \frac{1}{2x_o}\right)$ and $\nu_v \in \left(\frac{-1}{2y_o}, \frac{1}{2y_o}\right)$ accurately reflect the actual Fourier transform as carried out by the lens. Any values outside these ranges will be the values for the next period, and those will

not accurately represent the values produced by the lens because the physical Fourier transform of the spatial data array is a non-periodic image.

2.7.2 Area Represented by the DTFT

Given an understanding of the frequency range calculated by the DTFT, it is important to develop the relationship between the spatial frequencies of the DTFT and the physical distances in the Fourier image. Discussion thus far has focused on the DTFT operating in the frequency domain. However, in the holographic system, the material will be saturated by an excess amount of energy per area, so the size of the Fourier image, in meters, is crucially important. System parameters such as light wavelength and the lens focal length affect the size of the image that is projected upon the holographic medium. With variation of these parameters, it is possible for a single data array to saturate a medium in one system, but not saturate the same material in a different system.

Equation 2.4 is the basis for determining a relation between distance in the image and frequency in the DTFT. Reprinted here for clarity, the Fourier transform of a point source is:

$$F(d_u, d_v) = e^{\left(\frac{2\pi a d_v i}{f\lambda}\right)} \tag{2.4}$$

where a is the distance of a point source from the optical axis in the front focal plane, d_v is a distance from the optical axis in the back focal plane, f is the focal length of the lens, and λ is the wavelength of the laser. Based on this result, it is shown in the following that the optical Fourier transform of a sinusoidal plane-wave in one domain is two points in the other domain. These two points will be some distance from the optical axis, and this distance will represent a certain spatial frequency in the DTFT. That frequency is the spatial frequency of the plane-wave which is the Fourier transform pair of the two points.

Beginning with two point sources, one located at $+a$ and the other at $-a$ from the optical axis, the resulting plane-wave can be determined using

Equation 2.4. The plane-wave is the sum of the output of each point:

$$F(d_u, d_v) = e^{\left(\frac{2\pi a d_v i}{f\lambda}\right)} + e^{\left(\frac{-2\pi a d_v i}{f\lambda}\right)}.$$

Since the exponents of the complex exponentials have opposite polarity, their sum is:

$$F(d_u, d_v) = 2\cos\left(2\pi\frac{a}{f\lambda}d_v\right)$$

From this expression, it is evident that the frequency of the plane wave, due to two point sources, is:

$$\begin{aligned} \nu &= \frac{a}{f\lambda} \\ a &= f\lambda\nu \end{aligned} \tag{2.20}$$

The area represented by the DTFT can be calculated using this relation. Equation 2.20 was derived for a point source located a distance a from the optical axis in the image plane causing a plane wave of spatial frequency ν in the Fourier plane. However, the distinction between the image plane and Fourier plane is mathematically arbitrary because the two are Fourier pairs. So it is also evident from Equation 2.20 that a plane wave of spatial frequency ν in the image plane creates a point source located a distance a from the optical axis in the Fourier plane. It is thereby possible to use Equation 2.20 to convert the ranges of spatial frequencies for which the DTFT is accurate into a range of distances for which the DTFT is accurate.

Because the extent of the DTFT in the spatial frequency domain is $\nu_u \in \left(\frac{-1}{2x_o}, \frac{1}{2x_o}\right)$ and $\nu_v \in \left(\frac{-1}{2y_o}, \frac{1}{2y_o}\right)$, the physical extent of the Fourier image can be determined to be $d_u \in \left(\frac{-f\lambda}{2x_o}, \frac{f\lambda}{2x_o}\right)$ and $d_v \in \left(\frac{-f\lambda}{2y_o}, \frac{f\lambda}{2y_o}\right)$. The total extent in each direction is thereby $\frac{\lambda f}{x_o}$ meters in the u direction and $\frac{\lambda f}{y_o}$ meters in the v direction. This gives a total area calculated by the DTFT of:

$$A = \frac{(\lambda f)^2}{x_o y_o} \tag{2.21}$$

Since a closed-form solution to Equation 2.17 isn't possible, evaluating

this Fourier transform at discrete values for (ν_u, ν_v) is useful. This results in a two dimensional array of sampled values from a continuous two dimensional function. If the spacing between the samples of ν_u and ν_v is constant in each direction (though not necessarily the same for each) and both cover the entire frequency range (i.e. $\nu_u \in (-1/2x_o, 1/2x_o)$ and $\nu_v \in (-1/2y_o, 1/2y_o)$), the resulting array is effectively equivalent to the Discrete Fourier Transform (DFT) of the original image. This DFT can be expressed as:

$$F[k,l] = \sum_m \sum_n f[m,n] e^{-2\pi i \left(\frac{mk}{N_k} + \frac{nl}{N_l}\right)} \tag{2.22}$$

where k and l are unit-less indices of the Fourier transform array, and N_k and N_l are the number of elements in each dimension of the array. The inverse DFT is given by:

$$f[m,n] = \frac{1}{N_k N_l} \sum_k \sum_l F[k,l] e^{2\pi i \left(\frac{mk}{N_k} + \frac{nl}{N_l}\right)} \tag{2.23}$$

Further insight into these definitions is given in Section 2.7.5.

Equation 2.22 is equivalent to Equation 2.17 when $k/N_k = \nu_u x_o$ and $l/N_l = \nu_v y_o$, and therefore Equation 2.22 is a DTFT when evaluated at the spatial frequencies $\nu_u = k/N_k x_o$ and $\nu_v = l/N_l y_o$.

Since k and l are integers, the preceding equations describe ν_u and ν_v as one dimensional arrays of N_k and N_l elements, respectively, which cover the range $\nu_u \in (0, 1/x_o)$ and $\nu_v \in (0, 1/y_o)$. This range is not identical to the range derived above for the DTFT, but it does contain the correct values due to the periodic nature of the DTFT. The DTFT does not accurately depict the true, continuous optical Fourier transform in the region beyond $\nu_u \in (-1/2x_o, 1/2x_o)$ and $\nu_v \in (-1/2y_o, 1/2y_o)$. However, the values outside this range represent a tiling of the range in which the DTFT is accurate. So, the range of spatial frequencies over which the DFT calculates the Fourier transform of the data array contains every value within the range of accuracy of the DTFT.

Using the periodicity of the DFT indicated by Equation 2.19, the results of the DFT can be rearranged to be in the same order as those of the DTFT by adding the period to each argument of F. The range that is not directly

calculated by the DFT is $\nu_u \in (-1/2x_o, 0)$ and $\nu_v \in (-1/2y_o, 0)$. However, due to the periodicity of the DFT:

$$F\left[-1/2x_o, -1/2y_o\right] = F\left[-1/2x_o + 1/x_o, -1/2y_o + 1/y_o\right]$$
$$F\left[-1/2x_o, -1/2y_o\right] = F\left[1/2x_o, 1/2y_o\right]$$

Likewise:

$$F\left[0, 0\right] = F\left[1/x_o, 1/y_o\right]$$

So the ranges $\nu_u \in (-1/2x_o, 0)$ and $\nu_v \in (-1/2y_o, 0)$ are evaluated by the DFT along the ranges $\nu_u \in (1/2x_o, 1/x_o)$ and $\nu_v \in (1/2y_o, 1/y_o)$. Because of this, the DFT contains the DC value of the Fourier transform ($F[0, 0]$) in the upper left corner of the array, as per signal processing convention. In this derivation, the DFT is shifted so that the zero frequency point $F[0, 0]$ is located in the center of the array rather than the corner to provide an accurate image of the Fourier transform of $f[m, n]$. This convention is used for all figures of Fourier-domain images.

2.7.3 Energy Represented by the Fourier Transform

It is necessary to derive an expression for the energy contained in each element of $F[k, l]$ to determine whether or not a given holographic medium will be saturated by a given data array. This expression can be derived from Parseval's Theorem, which describes conservation of energy through the Fourier lens system. Parseval's Theorem for the discrete Fourier transform is:

$$\sum_m \sum_n |f[m, n]|^2 = \frac{1}{N^2} \sum_k \sum_l |F[k, l]|^2 \tag{2.24}$$

Here, it has been assumed that $N_k = N_l = N$, for notational convenience as well as for practicality. It is generally the case that both indices in $F[k, l]$ will consist of an equal number of elements. Additionally, the squaring operations are element-wise, rather than matrix squaring. This is because $f[m, n]$ and $F[k, l]$ are two-dimensional discrete functions, rather than systems of equations represented by matrices.

A proof of Equation 2.24 follows from the relationship:

$$\sum_m \sum_n |f[m,n]|^2 = \sum_m \sum_n f[m,n] \cdot f^*[m,n]$$

with the dot operator indicating the element-wise multiplication, and $f^*[m,n]$ representing the conjugate of $f[m,n]$. Using Equation 2.23 to represent $f^*[m,n]$ as the inverse of its transform gives:

$$\sum_m \sum_n |f[m,n]|^2 = \sum_m \sum_n f[m,n] \cdot \frac{1}{N^2} \sum_k \sum_l F^*[k,l] e^{-2\pi i \left(\frac{mk}{N} + \frac{nl}{N}\right)}$$

where both $F[k,l]$ and the exponential term are represented by their conjugates in the inverse DFT expansion of $f^*[m,n]$. Rearranging the summations gives:

$$\begin{aligned}
\sum_m \sum_n |f[m,n]|^2 &= \frac{1}{N^2} \sum_k \sum_l F^*[k,l] \cdot \sum_m \sum_n f[m,n] e^{-2\pi i \left(\frac{mk}{N} + \frac{nl}{N}\right)} \\
&= \frac{1}{N^2} \sum_k \sum_l F^*[k,l] \cdot F[k,l] \\
&= \frac{1}{N^2} \sum_k \sum_l |F[k,l]|^2
\end{aligned}$$

which concludes the proof of Parseval's Theorem for the DFT.

The left-hand side of Equation 2.24 was used above to determine the scaling of $f[m,n]$ required when representing the distribution of laser power over the data array. To evaluate the scaling factor it was stated that the scaled summation must be equivalent to the power contained in the data array. Because of this, the right-hand side is also equivalent to the power contained in the data array.

However, the equality in Equation 2.24 assumes that all light from the SLM will be contained within the area represented by the DFT. In practice, this assumption causes this derivation to yield an upper bound on the actual value of the energy delivered to the holographic material from the laser source because, in the physical system, some light that passes through the lens might not fall on the recording medium; the power of any light that, in reality,

falls outside the DFT boundary is considered to have been supplied to the holographic medium. Therefore, the actual energy within a prescribed area in the hologram may be lower than the value calculated by the formulation resulting from Parseval's Theorem. The calculated upper bound is expected to closely approximate the actual energy, because only negligible high frequency content is outside the range of the transform.

Returning to Equation 2.24, the $1/N^2$ term can be written inside the double summation to give the power at each frequency domain sample:

$$P_d = \frac{1}{N^2}|F[k,l]|^2$$

Summing this across the k and l indices gives the total power contained in the data array. The area expression given in Equation 2.21 can be divided by N^2 to give the area covered by each sample in the DFT. This is possible because the DFT evenly divides the area of the frequency into N^2 equal-sized sample spacings. The expression for the area of a single frequency domain sample is:

$$A_d = \frac{(\lambda f)^2}{x_o y_o N^2} \tag{2.25}$$

Dividing P_d by A_d gives an expression for the intensity ($^W/_{m^2}$) at each data sample in the $F[k,l]$ array:

$$I_d = |F[k,l]|^2 \frac{x_o y_o}{(\lambda f)^2}$$

This expression is true only if $f[m,n]$ is already properly scaled. If the data array still contains logical units then $F[k,l]$ must be multiplied by the scaling factor s_c to give an accurate result. This multiplication is valid because the DFT is a linear operation. For simplicity, $|s_c|^2 = {}^{P_o}/_M$ can be multiplied with I_d to give the properly scaled version of I_d for each point in the transform domain when logical units are used in the original data array:

$$I_{d,s_c} = |F[k,l]|^2 \frac{P_o x_o y_o}{M(\lambda f)^2} \tag{2.26}$$

The units of I_{d,s_c} are $^{W}/_{m^2}$ because $|F[k,l]|^2$ is composed of the unit-less logical values in this expression.

Holographic materials are saturated by an amount of energy per meter squared, not power per meter squared. To convert I_{d,s_c} to an energy per meter squared expression it is necessary to multiply by the exposure time. This gives units of Ws/m^2, which are equivalent to units of J/m^2:

$$F_{d,s_c} = |F[k,l]|^2 \frac{P_o x_o y_o t}{M(\lambda f)^2} \tag{2.27}$$

where t is the exposure time.

If a saturation limit (F_o) is known for a given holographic recording material, an expression can be derived for the lowest value of $|F[k,l]|$ that will result in a saturated medium. This is done by placing the value for saturation into the left-hand side of equation 2.27, then solving for $|F[k,l]|$.

$$|F[k,l]| = \sqrt{\frac{F_o M(\lambda f)^2}{P_o x_o y_o t}}$$

This expression gives a value for the amplitude of the Fourier transform of a data array that will result in saturation. This is useful when working with data arrays in logical units as it allows an upper limit for the peaks in the Fourier domain to be derived, given several system parameters.

Finally, it is important to note that this expression does not take into account the reference beam of the holographic storage system. The treatment of the reference beam in this analysis assumes that the reference beam has a constant intensity profile in the Fourier plane. Because of the constant profile, the intensity at each pixel due to the reference beam is constant. A variable representing the intensity of the reference beam can be added to Equation 2.26 to account for its effects:

$$I_T = |F[k,l]|^2 \frac{P_o x_o y_o}{M\left(\lambda f\right)^2} + I_r$$

where I_r represents the intensity of the reference beam.

Following the same reasoning as before, multiplying this value by the exposure time gives the total amount of energy per area delivered to the holographic storage medium.

$$F_T = |F[k,l]|^2 \frac{P_o x_o y_o t}{M\left(\lambda f\right)^2} + I_r t$$

And finally, solving for $|F[k,l]|$ with F_T set to F_o gives:

$$|F[k,l]| = \sqrt{\frac{(F_o - I_r t)\, M(\lambda f)^2}{P_o x_o y_o t}} \tag{2.28}$$

This expression gives the amplitude limit for saturation in the Fourier plane. If any point in $F[k,l]$ exceeds the value determined by Equation 2.28, the holographic material will become saturated at that point when the data array, $f[m,n]$, is recorded.

2.7.4 Over-Sampling and Zero-Padding

When using the DFT, it is required that $f[m,n]$ and $F[k,l]$ have the same dimensions. In the case of optical Fourier transforms, it is usually more informative for $F[k,l]$ to have more elements than $f[m,n]$. This is due to the fact that the Fourier image tends to have more variation within a given area than does the data array, with its fixed-size data elements.

There are two methods for increasing the size of $F[k,l]$. The first is to over-sample the data array, so that each pixel in the SLM is represented by more than one sample in $f[m,n]$. Over-sampling decreases the values of x_o and y_o if the size of the pixels in the SLM is kept constant. This increases the area covered by the DFT, as shown in Equation 2.21. However, the resolution in the frequency domain remains constant, as shown with Equation 2.25. Increasing N by a factor of 2 will decrease both x_o and y_o by a factor of ½. This relation between N and the sampling spacing variables causes the denominator of Equation 2.25 to remain constant, and thus the area per element of $F[k,l]$ does not change.

The second method is to zero-pad $f[m,n]$. Zero-padding involves simply appending zeros after the data samples in $f[m,n]$ to create a larger array.

This technique increases N without decreasing x_o and y_o. Because of this, the resolution of the frequency domain image is increased. This is useful because the DFT can only determine the total amount of power per each frequency domain sample, not the distribution of power within the sample itself. Higher resolution, thereby, allows a more accurate depiction of the Fourier image. If zero-padding is used, care must be taken when scaling the data array. The value M refers only to the number of elements in the data array which represent the SLM. The zero-padded elements will represent the empty space beyond the SLM, and should not be counted when scaling the data array values.

2.7.5 Analysis of DFT Definitions

The definitions for the DFT and its inverse, Equations 2.22 and 2.23 respectively, can be explained in greater detail through the use of a matrix representation of the DFT operator. This can be more readily understood in the case of a one-dimensional DFT. The definition for the one-dimensional DFT is:

$$F[k] = \sum_{m=0}^{N-1} f[m] e^{-2\pi i \left(\frac{mk}{N}\right)}$$

In this case, since $F[k]$ and $f[m]$ are one-dimensional arrays, the Fourier operator can be expressed as an $N \times N$ matrix. With both $F[k]$ and $f[m]$ represented as $N \times 1$ vectors, the exponential terms can be written in the following $\mathbf{\Phi}$ matrix, where the $(k,m)^{\text{th}}$ element in the matrix is given by the corresponding exponential term in the DFT definition:

$$\mathbf{\Phi} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-2\pi i \frac{1}{N}} & e^{-2\pi i \frac{2}{N}} & \cdots & e^{-2\pi i \frac{N-1}{N}} \\ 1 & e^{-2\pi i \frac{2}{N}} & e^{-2\pi i \frac{4}{N}} & \cdots & e^{-2\pi i \frac{2(N-1)}{N}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-2\pi i \frac{N-1}{N}} & e^{-2\pi i \frac{2(N-1)}{N}} & \cdots & e^{-2\pi i \frac{(N-1)^2}{N}} \end{bmatrix}$$

It is important to note here that the indices k and m have values between 0 and $N-1$, inclusively. To simplify notation in this section, also let the rows

and columns of the matrices be numbered 0 through $N-1$.

With the matrix $\mathbf{\Phi}$ defined, the DFT definition can be re-written as:

$$\mathbf{F} = \mathbf{\Phi f}$$

where the discrete functions $f[m]$ and $F[k]$ are written as $N \times 1$ vectors $\mathbf{f}$ and $\mathbf{F}$, respectively.

Considering $N = 4$, for example, gives the following:

$$\begin{bmatrix} F[0] \\ F[1] \\ F[2] \\ F[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-2\pi i \frac{1}{4}} & e^{-2\pi i \frac{2}{4}} & e^{-2\pi i \frac{3}{4}} \\ 1 & e^{-2\pi i \frac{2}{4}} & e^{-2\pi i \frac{4}{4}} & e^{-2\pi i \frac{6}{4}} \\ 1 & e^{-2\pi i \frac{3}{4}} & e^{-2\pi i \frac{6}{4}} & e^{-2\pi i \frac{9}{4}} \end{bmatrix} \begin{bmatrix} f[0] \\ f[1] \\ f[2] \\ f[3] \end{bmatrix}$$

This specific example shows the matrix operations more explicitly than the previous equations. It is clear that $F[k]$ is the sum of each element of $f[m]$ multiplied by a complex exponential term which has a frequency of $^{mk}/_{N}$. This also makes the indexing of $\mathbf{\Phi}$ clear. The vertical index must be k so that the product of $\mathbf{f}$ with each row corresponds to the k^{th} index of $\mathbf{F}$. Likewise, the horizontal index must be m so that the multiplication of $\mathbf{f}$ and $\mathbf{\Phi}$ proceeds along the m index.

Writing the DFT operator as a matrix has the advantage of providing an intuitive approach for deriving the inverse DFT operator. The inverse of matrix $\mathbf{\Phi}$ provides the inverse DFT operator as a matrix. This can be shown through matrix mathematics as:

$$\begin{aligned} \mathbf{\Phi}^{-1}\mathbf{F} &= \mathbf{\Phi}^{-1}\mathbf{\Phi f} \\ &= \mathbf{If} \\ &= \mathbf{f} \end{aligned}$$

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

The inverse of $\mathbf{\Phi}$ can be verified as follows. First, assume it will be the conjugate of $\mathbf{\Phi}$ multiplied by some scaling factor. If this is the case, the product of $\mathbf{\Phi}$ and its conjugate, $\mathbf{\Phi}^*$, should be a diagonal matrix with a constant value

along the diagonal.

Consider:

$$\mathbf{D} = \mathbf{\Phi}\mathbf{\Phi}^*$$

$$\mathbf{D} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-2\pi i\frac{1}{N}} & \cdots & e^{-2\pi i\frac{N-1}{N}} \\ 1 & e^{-2\pi i\frac{2}{N}} & \cdots & e^{-2\pi i\frac{2(N-1)}{N}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-2\pi i\frac{N-1}{N}} & \cdots & e^{-2\pi i\frac{(N-1)^2}{N}} \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{2\pi i\frac{1}{N}} & \cdots & e^{2\pi i\frac{N-1}{N}} \\ 1 & e^{2\pi i\frac{2}{N}} & \cdots & e^{2\pi i\frac{2(N-1)}{N}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{2\pi i\frac{N-1}{N}} & \cdots & e^{2\pi i\frac{(N-1)^2}{N}} \end{bmatrix}$$

From this it can be seen that the $(k,m)^{\text{th}}$ element in $\mathbf{D}$ is the product of the k^{th} row of $\mathbf{\Phi}$ and the m^{th} column of $\mathbf{\Phi}^*$. Using the definitions of these rows and columns, this product can be expressed as:

$$\mathbf{D}_{k,m} = \sum_{z=0}^{N-1} e^{-2\pi i\left(\frac{zk}{N}\right)} e^{2\pi i\left(\frac{mz}{N}\right)}$$

where z is an arbitrary variable of summation that represents m in the k^{th} row of $\mathbf{\Phi}$ and k in the m^{th} column of $\mathbf{\Phi}^*$. This can be re-written as:

$$\mathbf{D}_{k,m} = \sum_{z=0}^{N-1} e^{\left(-2\pi i\frac{k-m}{N}\right)z} \tag{2.29}$$

which is a standard geometric series. For the diagonal elements of $\mathbf{D}$, $k = m$. Using the previous expression it is clear that for $k = m$, the sum is:

$$\begin{aligned} \mathbf{D}_{k=m} &= \sum_{z=0}^{N-1} e^{\left(-2\pi i\frac{0}{N}\right)z} \\ &= \sum_{z=0}^{N-1} 1 \\ &= N \end{aligned}$$

Thus it is shown that that diagonal elements of $\mathbf{D}$ are constant, and have value N. For $\mathbf{\Phi}^{-1}$ to be a scaled version of $\mathbf{\Phi}^*$, it must also be demonstrated that the

non-diagonal elements of $\mathbf{D}$ are zero. This is done by evaluating Equation 2.29 for $k \neq m$:

$$\begin{aligned}\mathbf{D}_{k\neq m} &= \sum_{z=0}^{N-1} e^{\left(-2\pi i \frac{k-m}{N}\right)z} \\ &= \frac{1 - e^{\left(-2\pi i \frac{k-m}{N}\right)N}}{1 - e^{\left(-2\pi i \frac{k-m}{N}\right)}}\end{aligned} \tag{2.30}$$

For this expression to be zero, it is sufficient that the numerator be equal to zero and sufficient that the denominator have a non-zero value. Evaluating the numerator gives:

$$1 - e^{\left(-2\pi i \frac{k-m}{N}\right)N} = 1 - e^{-2\pi i(k-m)}$$

This expression is always equal to zero because k and m are integers, so $k-m$ must also be an integer, and $1-e^{2\pi in} = 0$, for any integer n. The denominator of Equation 2.30 is non-zero for $k - m \neq 0$ and $k - m$ is not a multiple of N. It is not possible for $k - m = 0$ or for $k - m$ to be a multiple of N in this expression. The first case would require $k = m$, and Equation 2.30 is derived for $k \neq m$. The second case is impossible because k and m only take values between 0 and $N - 1$, so the difference between k and m can never be N.

The above derivation demonstrates that $\mathbf{D} = N\mathbf{I}$. Therefore:

$$\begin{aligned}\mathbf{\Phi}\mathbf{\Phi}^* &= \mathbf{D} \\ &= N\mathbf{I} \\ \mathbf{\Phi}\left(\frac{1}{N}\mathbf{\Phi}^*\right) &= \mathbf{I} \\ \frac{1}{N}\mathbf{\Phi}^* &= \mathbf{\Phi}^{-1}\end{aligned} \tag{2.31}$$

The equation for the inverse DFT can be derived given knowledge of $\mathbf{\Phi}^{-1}$:

$$\mathbf{f} = \mathbf{\Phi}^{-1}\mathbf{F}$$

$$\mathbf{f} = \frac{1}{N}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{2\pi i\frac{1}{N}} & e^{2\pi i\frac{2}{N}} & \cdots & e^{2\pi i\frac{N-1}{N}} \\ 1 & e^{2\pi i\frac{2}{N}} & e^{2\pi i\frac{4}{N}} & \cdots & e^{2\pi i\frac{2(N-1)}{N}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{2\pi i\frac{N-1}{N}} & e^{2\pi i\frac{2(N-1)}{N}} & \cdots & e^{2\pi i\frac{(N-1)^2}{N}} \end{bmatrix}\mathbf{F}$$

The matrix multiplication of $\mathbf{\Phi}^{-1}$ and $\mathbf{F}$ can be re-written in summation notation to give the standard definition of the inverse DFT:

$$f[m] = \frac{1}{N}\sum_{k=0}^{N-1} F[k]e^{2\pi i\left(\frac{mk}{N}\right)}$$

This matrix approach to the DFT is also amenable to two dimensional transforms. The DFT of a two-dimensional array can be determined by computing the DFT of the rows of the array, and then computing the DFT of the columns of the resulting array. This process is easily carried out with the matrix formulation. If the dimensions of the data array are not equal, two separate $\mathbf{\Phi}$ matrices will be needed to compute the DFT. Each $\mathbf{\Phi}$ matrix is computed using N_k and N_l in place of N, respectively.

The process of computing the two-dimensional DFT using matrices can be written:

$$\mathbf{F} = \left[\mathbf{\Phi}_l\left(\mathbf{\Phi}_k\mathbf{f}\right)^{\mathsf{T}}\right]^{\mathsf{T}} \tag{2.32}$$

where $\mathbf{f}$ is a $k \times l$ array, $\mathbf{\Phi}_k$ and $\mathbf{\Phi}_l$ are the DFT matrices computed with N_k and N_l, respectively, and $^{\mathsf{T}}$ indicates the non-conjugate matrix transpose. This expression evaluates the DFT of the rows of $\mathbf{f}$ then evaluates the DFT of the columns of the resulting matrix.

Equation 2.32 can be re-written using matrix identities:

$$\begin{aligned}\mathbf{F} &= \left[\mathbf{\Phi}_l \left(\mathbf{\Phi}_k \mathbf{f}\right)^{\mathsf{T}}\right]^{\mathsf{T}} \\ &= \left(\left(\mathbf{\Phi}_k \mathbf{f}\right)^{\mathsf{T}}\right)^{\mathsf{T}} \left(\mathbf{\Phi}_l\right)^{\mathsf{T}} \\ &= \mathbf{\Phi}_k \mathbf{f} \mathbf{\Phi}_l\end{aligned}$$

Here, $\mathbf{\Phi}_l^{\mathsf{T}} = \mathbf{\Phi}_l$ because it is a symmetric matrix.

For $N_l = N_k = N$ the two-dimensional matrix DFT reduces to:

$$\mathbf{F} = \mathbf{\Phi} \mathbf{f} \mathbf{\Phi} \tag{2.33}$$

Since this expression introduces $\mathbf{\Phi}$ twice, it follows that the inverse DFT equation for two-dimensional arrays will require a $1/N^2$ term.

Chapter 3

Binary-to-Ternary Sparse Guided Scrambling

3.1 Introduction

In this chapter, a method for converting binary data to sparse ternary data is presented. The first step in this encoding involves representing binary data with unconstrained ternary sequences. Binary to ternary conversion is necessary for a practical implementation of a ternary holographic storage system because the overwhelming majority of data is currently represented in binary form. Thus any system that makes use of the data stored by a ternary HDS system will need to input and read out binary data.

It is also important to ensure that the recorded data arrays are sparse, which is the objective of the second step of encoding. Sparsity is defined as the proportion of OFF-pixels to the total number of pixels in a given block of data. Higher sparsity is beneficial to an HDS system because sparse data pages contain less power than data pages with a high number of ON-pixels, and therefore consume a smaller proportion of the dynamic range of the recording medium. This increases the storage capacity because more data pages can be recorded[35, 36, 37]. As a point of reference, uniformly distributed ternary symbols would have an average sparsity of $33.\bar{3}\%$.

3.2 Binary to Ternary Encoding

The initial conversion of data from binary to ternary symbols can be implemented with either a fixed length block code or a variable-length code. Since the final HTM data must be mapped to fixed size arrays, a fixed-size block code in which n_b bits are mapped to n_t ternary symbols is used. There are 2^{n_b} possible binary blocks and 3^{n_t} possible ternary blocks. The main requirement for encoding is that the number of ternary blocks must be greater than or equal to the number of binary blocks so that every input block can be mapped to a unique ternary block.

The code rate, R_b is defined as the ratio of n_b to n_t: $R_b = {}^{n_b}/_{n_t}$. The rate quantifies the number of bits stored per ternary symbol. Given that there are 3^{n_t} ternary blocks and 2^{n_b} binary blocks, the requirement that there be at least as many ternary blocks as binary blocks gives $3^{n_t} \geq 2^{n_b}$; with this constraint the maximum code rate is $\log_2(3)$. However, it is impossible for 2^{n_b} and 3^{n_t} to be equal for integers n_b and n_t because that would require some number to be both a power of 2 and a power of 3. In such a case, that number would have two distinct prime factorizations, which violates the fundamental theorem of arithmetic. Therefore, the rate of any binary-to-ternary block code must be strictly less than $\log_2(3)$.

Since sparsity is not a concern for this first step in binary-to-ternary conversion, values for n_b and n_t are chosen to maximize R_b and an efficiency metric $\eta = {}^{R_b}/_{\log_2(3)}$. The values chosen here are $n_b = 19$ and $n_t = 12$, which results in $R_b = 1.5833$ and $\eta = 0.99897$. The next n_b, n_t pair that achieves a higher rate is $84 : 53$ which provides $\eta = 0.99996$.

The binary-to-ternary conversion can be done with a lookup table for small block sizes, but this will be impossible with larger block sizes. For instance, a lookup table for the 19:12 code would need to store $2^{19} = 524288$ entries of length-12 ternary sequences for encoding, and 3^{12} length-19 binary entries for decoding where "illegal" ternary sequences are matched to "best guesses." A lookup table of this size would be manageable, but would require the encoding and decoding hardware to use more memory than is required for an algorithmic conversion. However, the 84:53 code would require a lookup table with $2^{83} \approx$

10^{25} length-53 ternary entries for encoding and 3^{53} length-83 binary entries for decoding, which is infeasible due to the memory requirement alone. The following subsections therefore describe algorithms to perform the conversion when large values of n_b and n_t are used.

3.2.1 Direct Base Conversion

One method that can be used to convert a binary data sequence to a ternary valued codeword is to treat the binary sequence as an n_b bit binary number and convert this number to its n_t symbol base-3 representation. This base conversion can be performed directly using a successive division algorithm with operations carried out in base-2. The first step of the algorithm is to divide the binary number by 11_2, where the subscript denotes the base of the number. The remainder, converted to base-3, is the least significant digit of the ternary word. For example, the remainder could be 10_2 which is equivalent to 2_3. Continuing with the algorithm, the quotient is again divided by 11_2, and the remainder of this second division operation is the second-least significant digit. The successive divisions continue until the quotient is zero; the last remainder is the most significant non-zero digit of the base-3 number.

An example of this process is shown in Table 3.1a. The binary input $\{111011\}$ is converted to $\{2012\}$, its ternary representation, by repeatedly dividing successive quotients by 11_2.

It is possible for the running quotient to become zero before n_t ternary symbols have been found, which means that the base-3 representation of the binary number has leading zeros. Enforcing the constraint that 3^{n_t} be greater than 2^{n_b} ensures that this conversion will never produce a ternary sequence containing more than n_t symbols. To verify this, the greatest possible number represented by n_b bits is $2^{n_b} - 1$, and the greatest possible number than can be represented by n_t ternary symbols is $3^{n_t} - 1$, so with the constraint that $3^{n_t} > 2^{n_b}$, it is clear that any sequence of n_b bits will represent a number that is less than the largest number that can be represented by n_t ternary symbols.

Decoding the ternary sequence back to binary is performed in a manner similar to the encoding procedure. The successive divisions can directly con-

Table 3.1: Direct Base Conversions

(a) Base-2 to Base-3 Encoding

		1	0	0	1	1	r10 = 2_3
11	1	1	1	0	1	1	
			1	1	0		r1 = 1_3
11	1	0	0	1	1		
		1	0				r0 = 0_3
11	1	1	0				
		0					r10 = 2_3
11	1	0					

(b) Base-3 to Base-2 Decoding

	1	0	0	2	r1
2	2	0	1	2	
		1	1	2	r1
2	1	0	0	2	
		2	1		r0
2	1	1	2		
	1	0			r1
2	2	1			
		1			r1
2	1	0			
	0				r1
2	1				

vert from base-3 to base-2 with all operations performed in base-3. As an example, the ternary sequence {2012} is decoded in Table 3.1b. The remainders, read bottom to top, give the correct binary sequence {111011}.

3.2.2 Conversion through Base-10

In practice, it may be simpler to first convert the binary number to base-10, given that commonly available processors are designed for base-10 operations. Using only base-10 operations, the conversion can be performed by finding the inner product of the binary number, treated as an n_b length vector, and another vector consisting of $2^{n_b-1}, 2^{n_b-2}, \cdots, 2^0$. The base-10 representation of the binary number can then be converted to a base-3 number using the previously described successive division algorithm, with the advantage that all arithmetical operations will be in base-10.

As an example, the binary input shown in Table 3.1a is first converted to its decimal representation through matrix multiplication of the vectors [1 1 1 0 1 1] and $[2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0]^\intercal$:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2^5 \\ 2^4 \\ 2^3 \\ 2^2 \\ 2^1 \\ 2^0 \end{bmatrix} = 2^5 + 2^4 + 2^3 + 2 + 1 = 59$$

Then, using base-10 arithmetic the number 59 is converted to its base-3 representation by successive divisions. This process is similar to the examples of Table 3.1, with 59 as the dividend and 3 as the divisor.

To decode, the ternary sequence can first be converted to base-10 by finding the inner product of the base-3 number and a vector composed of $3^{n_t-1}, 3^{n_t-2}, \cdots, 3^0$. The base-10 number can then be converted to binary using successive divisions with operations in base-10.

3.3 Guided Scrambling

Sparsity is introduced into the ternary data by way of a guided scrambling coding step. Guided scrambling is a multi-mode coding technique that produces a set of candidate codewords for a given source word[38]. The candidate codeword with the greatest number of zeros is selected from the set of candidates.

To determine each candidate codeword, the length-W source word is augmented by appending a prefix of A ternary symbols at the start of the block. The augmented sequence contains $S = W + A$ terms and is scrambled to produce one of the candidate codewords that may represent the source word. Since each possible augmenting pattern is used, the set of candidate codewords, called the selection set, includes 3^A candidate codewords.

All sequences are composed of elements from the set $\{0, 1, 2\}$. These values are used to represent the HTM symbols; 0 corresponds to the OFF-pixel, 1 to the ON-pixel with zero phase shift, and 2 to the ON-pixel with π phase shift.

The guided scrambling step is distinct from the initial binary-to-ternary encoding. It is implemented as a block code, but the size of the input ternary

Table 3.2: GF(3) Arithmetic

(a) Addition

+	**0**	**1**	**2**
0	0	1	2
1	1	2	0
2	2	0	1

(b) Multiplication

×	**0**	**1**	**2**
0	0	0	0
1	0	1	2
2	0	2	1

blocks is unrelated to n_t.

3.3.1 Mathematical Nomenclature

Throughout this analysis several representations are used for the ternary data. This subsection outlines those representations and the relationships between them. All operations are performed using arithmetic from the Galois field of three elements, GF(3); these operations are shown in Table 3.2. For the addition operation, each element has an inverse that is defined as the element that when added to the original element gives a sum of zero. The elements 1 and 2 are mutual additive inverses, and zero is its own additive inverse. Additive inverses allow subtraction in GF(3). The additive inverse of 1 can be written as -1, which implies that in GF(3), $2 = -1$ and similarly $1 = -2$.

The most basic form for the ternary data is a sequence of consecutive ternary symbols. Sequences are described in this chapter using the notation $\{t_0 t_1 t_2 ...\}$. An example sequence is $\{01220211\}$. With this representation of sequences, the left-most symbol is the first symbol in time i.e. t_0 occurs before t_1.

Sequences are translated into a polynomial form that is useful to describe the guided scrambling algorithm. For this translation, the sequence values are used as the coefficients of a polynomial, and the coefficient of the highest order term in the polynomial represents the first symbol in the sequence. The example sequence $\{01220211\}$ translates to the polynomial $x^6 + 2x^5 + 2x^4 + 2x^2 + x + 1$. Polynomials are referenced using the notation $p(x)$.

Because the guided scrambling process generates a set of candidate codewords for each source word, this analysis often needs to refer to several related

polynomials at once. To do this, a matrix notation is used to represent the set of polynomials. Each row of the matrix is a vector consisting of the coefficients of the polynomial that the row represents. The row vectors are referred to with bold-face, lower-case notation such as $\mathbf{m}_i$ where i is the index of the row. The matrix is referred to with bold-face, capital notation such as $\mathbf{M}$. As an example, the polynomials $p_0(x) = x^3 + 2x + 2$, $p_1(x) = x^3 + x^2 + 1$, $p_2(x) = 2x^3 + 2x^2 + 1$, and $p_3(x) = x^3 + x + 2$ are represented by rows in the following matrix:

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_0 \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \mathbf{m}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2 & 2 \\ 1 & 1 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix} \tag{3.1}$$

The polynomials and vectors/matrix are related by:

$$\begin{bmatrix} p_0(x) \\ p_1(x) \\ p_2(x) \\ p_3(x) \end{bmatrix} = \mathbf{M} \begin{bmatrix} x^3 \\ x^2 \\ x^1 \\ x^0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2 & 2 \\ 1 & 1 & 0 & 1 \\ 2 & 2 & 0 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x^3 \\ x^2 \\ x^1 \\ x^0 \end{bmatrix}$$

Finally, the operation of adding two sequences is used frequently in the following subsections. This operation is defined for the polynomial and vector representations of the sequence, so the addition is always element-wise, where the result of each element-wise addition is always an element from the set {0, 1, 2}.

3.3.2 Scrambling Process

The scrambling process is defined by polynomial division. Each of the sequences is represented by a polynomial in which the first term in the sequence is the coefficient of the highest order term in the polynomial. Let the length-W input word be represented with the polynomial $w(x)$, the i^{th} augmenting sequence be $a_i(x)$, the i^{th} augmented sequence be $s_i(x)$, and the i^{th} candidate codeword be $q_i(x)$.

The polynomials are related according to:

$$w(x) = \sum_{j=0}^{W-1} w_j x^j$$
$$a(x) = \sum_{j=0}^{A-1} a_j x^j$$
$$s(x) = a(x)x^W + w(x)$$

where w_j, $j \in (0 : W-1)$, are the values of the symbols in the source word, and a_j, $j \in (0 : A-1)$, are the symbols in the i^{th} augmenting pattern. Let the division of $s(x)$ by $d(x)$ be expressed with the operator $Q_{d(x)}[s(x)]$, defined as:

$$\begin{aligned} q(x) &= Q_{d(x)}[s(x) \cdot x^D] \\ &\triangleq \frac{s(x)x^D}{d(x)} - \frac{r(x)}{d(x)} \end{aligned}$$

where $r(x)$ is the remainder of the division. Subtracting $r(x)/d(x)$ from the quotient effectively neglects the remainder when constructing $q(x)$. This remainder term is inconsequential because a codeword is decoded by multiplying it with the scrambling polynomial, which gives:

$$q(x)d(x) = s(x)x^D - r(x)$$

The lowest order term in $s(x)$ is the constant term, so the lowest order term in $s(x)x^D$ is the x^D term. Since the degree of $r(x)$ must be less than D (including the degenerate case when $r(x) = 0$) the subtraction of $r(x)$ will not alter any of the terms in $s(x)x^D$, and thus $s(x)$ can be reconstructed by reading the first S terms in $q(x)d(x)$.

An example of this process is shown in Table 3.3. The source sequence $\{111111\}$ is augmented with the sequence $\{21\}$ to produce the augmented sequence $\{21111111\}$. The polynomial form of this sequence is $s(x) = 2x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$. The scrambling polynomial $d(x) = x^2 + x + 2$ divides $s(x)x^D$ as shown in Table 3.3a. This division yields the quotient $q(x) =$

Table 3.3: Guided Scrambling Division Encoding and Multiplication Decoding

(a) Division

$$
\begin{array}{r|llllllll}
 & & & 2x^7 & +2x^6 & +x^5 & +2x^4 & & & +x \\
\hline
x^2+x+2 & 2x^9 & +x^8 & +x^7 & +x^6 & +x^5 & +x^4 & +x^3 & +x^2 & \\
 & -2x^9 & -2x^8 & -x^7 & & & & & & \\
\hline
 & & 2x^8 & & +x^6 & & & & & \\
 & & -2x^8 & -2x^7 & -x^6 & & & & & \\
\hline
 & & & x^7 & & +x^5 & & & & \\
 & & & -x^7 & -x^6 & -2x^5 & & & & \\
\hline
 & & & & 2x^6 & +2x^5 & +x^4 & & & \\
 & & & & -2x^6 & -2x^5 & -x^4 & & & \\
\hline
 & & & & & & & x^3 & +x^2 & \\
 & & & & & & & -x^3 & -x^2 & -2x \\
\hline
 & & & & & & & & & x
\end{array}
$$

(b) Multiplication

$$
\begin{array}{rllllllll}
q(x)d(x) = & (2x^7+2x^6+x^5+2x^4+x)(x^2+x+2) & & & & & & & & \\
= & 2x^9 & +2x^8 & +x^7 & +2x^6 & & & +x^3 & & \\
 & + & +2x^8 & +2x^7 & +x^6 & +2x^5 & & & +x^2 & \\
 & + & & +x^7 & +x^6 & +2x^5 & +x^4 & & & +2x \\
\hline
= & 2x^9 & +x^8 & +x^7 & +x^6 & +x^5 & +x^4 & +x^3 & +x^2 & +2x
\end{array}
$$

$2x^7+2x^6+x^5+2x^4+x$ and the remainder $r(x)=x$. The quotient represents the scrambled sequence $\{22120010\}$, and the remainder is neglected.

Table 3.3b shows that the product of $q(x)$ and $d(x)$ is $2x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+2x$. This is equivalent to $s(x)x^D-r(x)$, and the augmented sequence can be recovered from this product by reading the first eight terms, which gives $\{21111111\}$. The source word is reconstructed by discarding the augmenting sequence.

The rate of the guided scrambling code is defined as the ratio of the number of input symbols to the number of encoded symbols: $R_g = W/S$. Once the choice of D, A, and W has been made, the rate is constant for all scrambling polynomials of degree D. Since all candidate codewords are the same length, the selection of the most sparse candidate codeword does not affect the rate.

3.3.3 Set of Relationship Sequences

The quotients in the selection set for a particular input word are related through addition with a set of relationship sequences. To illustrate this property, consider a specific augmented word $s_i(x)$ which is defined as:

$$s_i(x) = a_i(x) \cdot x^W + w(x)$$

The quotient for $s_i(x)$ is given by:

$$q_i(x) = Q_{d(x)} \left[s_i(x) x^D \right]$$

which can be rewritten as:

$$\begin{aligned} q_i(x) &= Q_{d(x)} \left[\left(a_i(x) x^W + w(x) \right) x^D \right] \\ &= Q_{d(x)} \left[a_i(x) x^{W+D} \right] + Q_{d(x)} \left[w(x) x^D \right] \end{aligned}$$

The term $Q_{d(x)} \left[a_i(x) x^{W+D} \right]$ is equivalent to the quotient produced by scrambling the all-zero source sequence augmented with $a_i(x)$. Let this quotient be $h_i(x)$. Similarly, $Q_{d(x)} \left[w(x) x^D \right]$ is equivalent to the quotient produced by scrambling the input word $w(x)$ augmented with the all-zero augmenting sequence. Let this quotient be $q_0(x)$. With these definitions, $q_i(x)$ can be expressed as:

$$q_i(x) = h_i(x) + q_0(x) \tag{3.2}$$

The selection set is the set of all $q_i(x)$ with $i \in (0 : 3^A - 1)$. Equation 3.2 implies that the entire selection set can be constructed by adding each $h_i(x)$ polynomial to a single quotient $q_0(x)$. All of the $h_i(x)$ sequences are determined solely by the augmenting sequences and the scrambling polynomial, both of which do not depend on the input word. The set of all $h_i(x)$ with $i \in (0 : 3^A - 1)$ is called the set of relationship sequences.

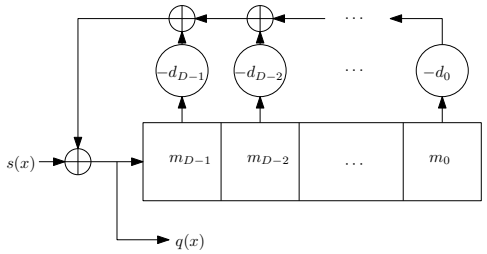


Figure 3.1: General Shift Register Division

3.4 Primitive Scrambling Polynomials

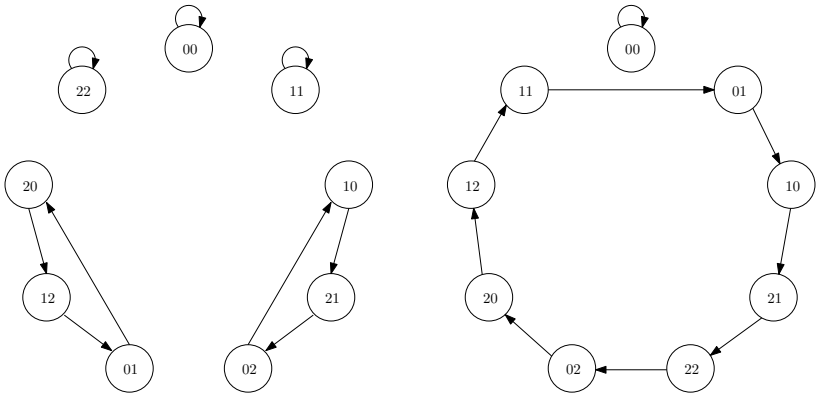
The scrambling polynomial has a significant effect on the overall sparsity of the output data. Extensive simulations, which are summarized in Section 3.5.1, have indicated that the use of primitive polynomials in GF(3) as scrambling polynomials results in the highest average sparsity in the encoded arrays.

In this section it is proven that systems employing primitive scrambling polynomials guarantee a sparsity greater than 1/3, and that the probability of encoded sequences having this worst-case sparsity decreases with increasing D.

3.4.1 m-sequences

The polynomial division algorithm can be performed directly using a linear feedback shift register with D memory elements. Figure 3.1 shows a general design for such a shift register for a monic scrambling polynomial (when the coefficient of the x^D term in $d(x)$ has the value 1). The $\bigoplus$ symbols denote GF(3) addition, and the circled $-d_i$ terms denote GF(3) multiplication by $-d_i$. The coefficients of $s(x)$ are fed into the register serially and the coefficients of $q(x)$ are calculated immediately. The negatives of the order $D-1$ and lower terms of $d(x)$ multiply the value in each corresponding memory element. These values are all summed over $GF(3)$, and this feedback term is added to the next input value to determine each subsequent quotient value.

The division circuitry can be represented by a finite state machine with 3^D possible states. Because the circuitry contains feedback, the finite state machine will cycle through some set of states while the input is zero. With



(a) $d(x) = x^2 + x + 1$ (b) $d(x) = x^2 + x + 2$

Figure 3.2: Finite State Diagrams for two $d(x)$ polynomials.

all-zero input, if the circuit ever reaches the all-zero state, it will remain there perpetually for any $d(x)$. As such, the longest possible cycle will progress through every state except the all-zero state.

Figure 3.2 shows the finite state diagrams for two different degree-2 $d(x)$ polynomials with transitions shown for $s(x) = 0$. The states are labeled by the contents of the memory elements (m_1, m_0) as described in Figure 3.1. As shown in Figure 3.2a, the polynomial $x^2 + x + 1$ contains several cycles. By contrast, the polynomial $x^2 + x + 2$ contains only two cycles, the self-loop on state 00 and a maximum length cycle of length $3^2 - 1 = 8$.

The output of a maximum length cycle is known as a m-sequence[39]. Ternary m-sequences repeat with a period of $3^D - 1$ and contain every combination of D symbols except for the length-D all-zero pattern. Primitive polynomials over GF(P) produce m-sequences[40].

Additionally, a linear recurrence relation can be defined for the output of a given shift register circuit. The general form of this relation is:

$$q_n = s_n - \sum_{i=0}^{D-1} d_i q_{n-D+i} \tag{3.3}$$

where d_i represents the i^{th} coefficient of $d(x)$ with d_0 being the constant term. In Equation 3.3, the q_n and s_n terms are time-indexed.

As an example, the polynomial $d(x) = x^2 + x + 2$, which is primitive

n	q_n	$=$	$-d_0 \cdot q_{n-2} - d_1 \cdot q_{n-1}$	
0	0	$=$	$-2 \cdot 1 - 1 \cdot 1$	$= 1 + 2$
1	1	$=$	$-2 \cdot 1 - 1 \cdot 0$	$= 1 + 0$
2	2	$=$	$-2 \cdot 0 - 1 \cdot 1$	$= 0 + 2$
3	2	$=$	$-2 \cdot 1 - 1 \cdot 2$	$= 1 + 1$
4	0	$=$	$-2 \cdot 2 - 1 \cdot 2$	$= 2 + 1$
5	2	$=$	$-2 \cdot 2 - 1 \cdot 0$	$= 2 + 0$
6	1	$=$	$-2 \cdot 0 - 1 \cdot 2$	$= 0 + 1$
7	1	$=$	$-2 \cdot 2 - 1 \cdot 1$	$= 2 + 2$

Table 3.4: Example m-sequence and recurrence relation

over GF(3), generates the m-sequence $\{01220211\}$. Table 3.4 verifies that this sequence satisfies the recurrence relation of Equation 3.3 using the coefficients of $d(x)$. Since the m-sequence is cyclical, q_6 and q_7 are used to represent q_{-2} and q_{-1} respectively. Additionally, s_n is equal to zero for all n in this example.

3.4.2 Table of Relationship Sequences

When the scrambling polynomial is primitive and the number of augmenting symbols is equal to the degree of the scrambling polynomial, each relationship sequence is either an m-sequence or the all-zero sequence, as shown below.

The system will have 3^D possible states and the 3^A augmenting sequences will consist of all possible sequences of length A. Each of these augmenting sequences will force the shift register to a different initial state. From each non-zero state, the shift register will cycle through every other state, producing the rest of the m-sequence as its output, for the remaining W iterations. With the all-zero input word, the all-zero augmenting sequence will initialize the system to the all-zero state where it will remain and give the all-zero output.

Therefore, the set of relationship sequences for a system that uses a primitive scrambling polynomial of degree $D = A$ will consist of the all-zero sequence as $h_0(x)$, and each of the remaining $3^D - 1$ relationship sequences will be a shifted version of one particular m-sequence. All $h_i(x)$, $i \in (0 : 3^D - 1)$, are the same m-sequence because the state transitions that generate the m-sequence are determined by the scrambling polynomial. They are all shifted versions of

$h_i(x)$	Coefficients							
$h_0(x)$	0	0	0	0	0	0	0	0
$h_1(x)$	0	1	2	2	0	2	1	1
$h_2(x)$	0	2	1	1	0	1	2	2
$h_3(x)$	1	2	2	0	2	1	1	0
$h_4(x)$	1	0	1	2	2	0	2	1
$h_5(x)$	1	1	0	1	2	2	0	2
$h_6(x)$	2	1	1	0	1	2	2	0
$h_7(x)$	2	2	0	2	1	1	0	1
$h_8(x)$	2	0	2	1	1	0	1	2

Table 3.5: An example set of relationship sequences generated by the primitive polynomial $x^2 + x + 2$.

this m-sequence because the shift register is initialized to a different state for each $h_i(x)$. Finally, the set of all $h_i(x)$ sequences will include all shifts of the m-sequence because every possible initial state will be used.

Table 3.5 lists the coefficients of each $h_i(x)$ for the scrambling polynomial $d(x) = x^2 + x + 2$. It can be observed that $h_0(x)$ is the all-zero sequence and each of the remaining sequences are shifted versions of the same m-sequence.

A set of sequences containing the all-zero sequence as well as each shifted version of a particular m-sequence forms an Abelian group with the operation of element-wise addition of the sequences in GF(3). This was proven in [39] for binary valued m-sequences in GF(2), and the group properties hold for ternary m-sequences in GF(3). The only significant difference between the binary and ternary cases is that any sequence is its own additive inverse in the binary case, but this is not true for the ternary case. However it is straightforward to show that the additive inverse of any relationship sequence will be in the set of relationship sequences.

Any particular sequence, $h_i(x)$, is generated by initializing the system with one of the augmenting prefixes, $a_i(x)$. The additive inverse of $h_i(x)$, denoted $\bar{h}_i(x)$, will be generated when the additive inverse of $a_i(x)$, denoted $\bar{a}_i(x)$, is used as the augmenting prefix. This is because $\bar{a}_i(x) = 2a_i(x)$ in GF(3), so $a_i(x)$ and its additive inverse are related linearly. As shown by the linear feedback shift register implementation and the linear recurrence relation,

$Q_{d(x)}\left[s(x)\right]$ is a linear operation. Thus:

$$\begin{aligned} Q_{d(x)}\left[\bar{a}_i(x)x^{W+D}\right] &= Q_{d(x)}\left[2a_i(x)x^{W+D}\right] \\ &= 2Q_{d(x)}\left[a_i(x)x^{W+D}\right] \\ &= 2h_i(x) \\ &= \bar{h}_i(x) \end{aligned}$$

Additionally, $\bar{a}_i(x)$ is certain to be used to generate a sequence in the set of relationship sequences because all possible ternary sequences of length A are used as prefixes to generate the selection set. Therefore, every $h_i(x)$ has its additive inverse in the set of relationship sequences.

It can also be shown that the first half of a ternary m-sequence is the additive inverse of its second half. This is a result of the fact that the set of relationship sequences is composed of shifted versions of an m-sequence, and that the set of relationship sequences forms a group. Because of the group structure, the additive inverse of an m-sequence is in the set of relationship sequences, and therefore the additive inverse can also be formed by shifting the original sequence by k positions. Since k shifts invert the original sequence, k more shifts must invert the inverted sequence to reconstruct the original sequence. Because m-sequences have a period of S, $k = {}^{S}/_{2}$, and the first half of an m-sequence is equal to the additive inverse of its second half.

3.4.3 Selection Set Analysis

It will now be shown that for any quotient in a selection set, the other members of that selection set can be found by adding each $h(x)$ sequence to the quotient. The given quotient, $q_i(x)$, is equal to $q_0(x) + h_i(x)$. Adding a different relationship sequence, $h_j(x)$, to $q_i(x)$ gives:

$$\begin{aligned} q_k(x) &= q_i(x) + h_j(x) \\ q_k(x) &= q_0(x) + h_i(x) + h_j(x) \\ q_k(x) &= q_0(x) + h_k(x) \end{aligned}$$

$q_i(x)$	Coefficients							
$q_0(x)$	0	0	1	0	2	2	1	2
$q_1(x)$	0	1	0	2	2	1	2	0
$q_2(x)$	0	2	2	1	2	0	0	1
$q_3(x)$	1	2	0	0	1	0	2	2
$q_4(x)$	1	0	2	2	1	2	0	0
$q_5(x)$	1	1	1	1	1	1	1	1
$q_6(x)$	2	1	2	0	0	1	0	2
$q_7(x)$	2	2	1	2	0	0	1	0
$q_8(x)$	2	0	0	1	0	2	2	1

Table 3.6: An example selection set constructed by adding $q_0(x)$ to each sequence in the set of relationship sequences generated with the scrambling polynomial $d(x) = x^2 + x + 2$, as shown in Table 3.5.

The third expression follows since, due to the group structure of the set of relationship sequences, the sum of two sequences is a third sequence in the group. Adding each of the 3^D $h(x)$ sequences to the initial quotient is equivalent to adding each sequence to $q_0(x)$. Thus, the entire selection set is determined. This property allows the selection set to be analyzed by considering the sum of a given length-S sequence with each of the length-S m-sequences created by the primitive scrambling polynomial.

Table 3.6 shows an example selection set. In this table, $q_0(x)$ was calculated by scrambling the source word {111111} with the all-zero, length-two augmenting sequence. The other rows were determined by adding $q_0(x)$ to each relationship sequence shown in Table 3.5. These results are identical to those obtained by scrambling the source word with each respective augmenting sequence.

3.4.4 Worst-case Sets

In this application, the worst-case codewords are the words that are selected from the selection set to represent the source word, but that contain the fewest zeros of all codewords. A codeword, the word selected from the selection set, is defined as the quotient with the most zeros among the quotients in the selection set (in the event of ties, the designation of codeword is arbitrary

among the quotients with the highest number of zeros in a selection set). An analysis of the worst-case codewords follows for systems employing primitive scrambling polynomials over GF(3) with $A = D$ and $W = 3^D - 1 - A$.

With the described system characteristics, it will now be shown that a quotient with 3^{D-1} zeros will be the worst-case codeword.

A matrix $\mathbf{H}$ can be formed from the set of relationship sequences. Each row in $\mathbf{H}$ consists of the coefficients of a single $h_i(x)$ polynomial. Thus, $\mathbf{H}$ contains 3^D rows and $3^D - 1$ columns. A general definition of $\mathbf{H}$ is given in Equation 3.4. The $h_{i,j}$ term in the matrix is the j^{th} coefficient of $h_i(x)$, with $j = 3^D - 2$ being the term representing the most significant coefficient in the polynomial $h_i(x)$, and $j = 0$ being the term representing the least significant coefficient.

$$\mathbf{H} = \begin{bmatrix} h_{0,3^D-2} & h_{0,3^D-3} & \cdots & h_{0,0} \\ h_{1,3^D-2} & h_{1,3^D-3} & \cdots & h_{1,0} \\ \vdots & \vdots & \ddots & \vdots \\ h_{3^D-1,3^D-2} & h_{3^D-1,3^D-3} & \cdots & h_{3^D-1,0} \end{bmatrix} \tag{3.4}$$

As an example, the coefficients shown in Table 3.5 form the $\mathbf{H}$ matrix for the system using $d(x) = x^2 + x + 2$, with two augmenting symbols and six symbols per source word. Reprinted here as a matrix this gives:

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 0 & 2 & 1 & 1 \\ 0 & 2 & 1 & 1 & 0 & 1 & 2 & 2 \\ 1 & 2 & 2 & 0 & 2 & 1 & 1 & 0 \\ 1 & 0 & 1 & 2 & 2 & 0 & 2 & 1 \\ 1 & 1 & 0 & 1 & 2 & 2 & 0 & 2 \\ 2 & 1 & 1 & 0 & 1 & 2 & 2 & 0 \\ 2 & 2 & 0 & 2 & 1 & 1 & 0 & 1 \\ 2 & 0 & 2 & 1 & 1 & 0 & 1 & 2 \end{bmatrix} \tag{3.5}$$

Every column in $\mathbf{H}$ has exactly 3^{D-1} zeros, ones, and twos. This is true because each row, except the first, is an m-sequence, and every possible shift

of the m-sequence is present. A ternary m-sequence contains 3^{D-1} ones, 3^{D-1} twos, and $3^{D-1}-1$ zeros. The first row consists of the coefficients of $h_0(x)$ which are all zeros. Thus, the columns of $\mathbf{H}$ all contain 3^{D-1} zeros, ones, and twos.

It follows from Section 3.4.3 that a matrix representing the selection set can be created by adding the coefficients of $q_0(x)$ to each row in $\mathbf{H}$. Let this selection set matrix be $\mathbf{Q}$. The columns of $\mathbf{Q}$ have the same distribution of zeros, ones, and twos as $\mathbf{H}$, because every column in $\mathbf{Q}$ will be the sum of a column from $\mathbf{H}$ and a single $q_{0,j}$ value. If the $q_{0,j}$ value is zero the column of $\mathbf{Q}$ is identical to the column from $\mathbf{H}$; if the value is one the column of $\mathbf{Q}$ has the same structure as the column from $\mathbf{H}$ except the zeros in $\mathbf{H}$ are now ones, the ones are twos, and the twos are zeros; for $q_{0,j}$ being two, the structure is the same, but the zeros are now twos, the ones are zeros, and the twos are ones. Thus, the number of zeros, ones, and twos in the columns of $\mathbf{Q}$ are identical to the numbers of each in $\mathbf{H}$.

Since each column must contain 3^{D-1} zeros and there are 3^D-1 columns in $\mathbf{Q}$, every selection set with the described system characteristics must contain exactly $3^{D-1}\left(3^D-1\right)=3^{2D-1}-3^{D-1}$ zeros.

To determine the number of zeros in a worst-case codeword, assume that a quotient with $3^{D-1}-1$ zeros is selected. Since the system chooses the quotient with the most zeros to be the codeword, this implies that no other quotient in the selection set has more than $3^{D-1}-1$ zeros. Given that there are 3^D quotients in the selection set, this gives a maximum of $3^D\left(3^{D-1}-1\right)=3^{2D-1}-3^D$ zeros in the selection set in total. Since this maximum is lower than the required number of zeros in a selection set, it cannot possibly be a valid $\mathbf{Q}$ matrix, and no selection set with $3^{D-1}-1$ zeros as its best choice can exist. Similarly, no selection set with fewer than $3^{D-1}-1$ zeros as its best choice can exist.

Alternatively, a selection set with 3^{D-1} zeros in its best choice quotient will have a maximum of $3^{D-1}\cdot 3^D=3^{2D-1}$ zeros. Since this is greater than the required value, a selection set with such a quotient can at least conceivably exist. It will now be shown that such selection sets do exist.

3.4.5 Identification and Occurrence of Worst-case Sets

Selection sets that produce worst-case codewords can be identified and enumerated based on a few properties of the table of relationship sequences and the selection set.

Table of Relationship Sequences

To illustrate these properties, consider the following organization of the set of relationship sequences:

$$\mathbf{H}_c = \begin{bmatrix} \mathbf{H}_{c0} \\ \mathbf{H}_{c1} \\ \mathbf{H}_{c2} \end{bmatrix}$$

Here, $\mathbf{H}_c$ is the matrix formed from the coefficients of each $h_i(x)$ relationship sequence sorted along column c. The sub-matrices $\mathbf{H}_{c0}$, $\mathbf{H}_{c1}$, and $\mathbf{H}_{c2}$ are each defined as the set of rows in $\mathbf{H}$ with a 0, 1, or 2 in column c respectively. These three subsets each contain 3^{D-1} rows because, as shown previously, each column in $\mathbf{H}$ has exactly 3^{D-1} occurrences of each symbol.

Continuing with the example of $d(x) = x^2 + x + 2$ with $A = 2$ and $W = 6$, the $\mathbf{H}$ matrix from Equation 3.5 is sorted along column 0 to give the following $\mathbf{H}_0$ matrix:

$$\mathbf{H}_0 = \left[\begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 0 & 2 & 1 & 1 \\ 0 & 2 & 1 & 1 & 0 & 1 & 2 & 2 \\ \hdashline 1 & 0 & 1 & 2 & 2 & 0 & 2 & 1 \\ 1 & 1 & 0 & 1 & 2 & 2 & 0 & 2 \\ 1 & 2 & 2 & 0 & 2 & 1 & 1 & 0 \\ \hdashline 2 & 0 & 2 & 1 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 0 & 1 & 2 & 2 & 0 \\ 2 & 2 & 0 & 2 & 1 & 1 & 0 & 1 \end{array}\right] = \begin{bmatrix} \mathbf{H}_{00} \\ \mathbf{H}_{01} \\ \mathbf{H}_{02} \end{bmatrix} \tag{3.6}$$

In subset $\mathbf{H}_{c0}$, column c contains all-zeros as does column $c + {}^{S}\!/_{2} \pmod S$. Column c is all-zero because of the definition of $\mathbf{H}_{c0}$, and column $c + {}^{S}\!/_{2} \pmod S$ is all-zero because each row in $\mathbf{H}_{c0}$ is either an m-sequence or the all-zero row.

The second half of an m-sequence is the additive inverse of the first half of the m-sequence so any positions in one half of the m-sequence that contain zeros must also contain zeros in those positions in the other half. For example, if the first symbol in the first half of a row is zero, then the first symbol in the second half of the m-sequence must also be zero. For brevity, column $c + {}^{S}/_{2} \pmod{S}$ is referred to as column $\bar{c}$ henceforth because these two columns are additive inverses in $\mathbf{H}$.

Note that $\mathbf{H}_{c0}$ is a subgroup of $\mathbf{H}$, because the rows of $\mathbf{H}_{c0}$ are selected from the rows of $\mathbf{H}$, and $\mathbf{H}_{c0}$ has the four properties of a group:

1. The additive identity is in $\mathbf{H}_{c0}$ because h_0 has zeros in every column, so it will be placed in $\mathbf{H}_{c0}$ for all c.

2. $\mathbf{H}_{c0}$ has closure because all elements in $\mathbf{H}_{c0}$ have a zero in column c and $\mathbf{H}$ has closure. Thus, the sum of any two elements in $\mathbf{H}_{c0}$ will have a zero in column c and the sum is also an element of $\mathbf{H}$. All such sums are therefore also in $\mathbf{H}_{c0}$ by definition.

3. Every element in $\mathbf{H}_{c0}$ also has its additive inverse because $\mathbf{H}$ has this property, and the additive inverse of any row with a zero in column c must also have a zero in column c.

4. Associativity follows from the standard definition of finite field addition.

A column k of $\mathbf{H}_{c0}$ can only have one of two distinct symbol distributions. Either the column will only contain zeros, or it will contain an equal number of each ternary symbol. This follows from a few implications of the group properties of $\mathbf{H}_{c0}$. Since every row in $\mathbf{H}_{c0}$ has an additive inverse in $\mathbf{H}_{c0}$, then any 1s in column k will be balanced by an equal number of 2s in that column. Furthermore, every individual sum of any row with a 1 in column k with all rows containing a 2 in column k must result in a unique row with a zero in column k. So, the number of ones or twos in a column of $\mathbf{H}_{c0}$ cannot be greater than the number of zeros in that column.

Likewise, every sum of any row with a 1 (or 2) in column k with all rows containing a zero in column k must each result in a unique row with a 1 (or

2) in row k. Because of this, the number of rows with a zero in column k cannot be greater than the number of rows with a 1 (or 2), unless column k contains no 1s or 2s. Given these two conditions, the number of 1s or 2s in column k must be either zero or equal to the number of zeros in column k. In the second case, the number of zeros in column k is equal to the number of ones and equal to the number of twos. Therefore columns of this type are balanced, and contain 3^{D-2} occurrences of each symbol.

The total number of ones (or twos) in $\mathbf{H}_{c0}$ is $3^{D-1}\left(3^{D-1}-1\right)$, because each row, except for the additive identity, is an m-sequence and has 3^{D-1} ones or twos and there are $3^{D-1}-1$ such rows. Since ones and twos can only occur in columns with 3^{D-2} of each symbol, then 3^D-3 columns must contain ones and twos. There are 3^D-1 columns in $\mathbf{H}_{c0}$, so there can only be two columns with no ones or twos for all D. These two columns are columns c and $\bar{c}$, and hence every column except for these two must contain the same number of ones, twos, and zeros in $\mathbf{H}_{c0}$.

$\mathbf{H}_{c1}$ and $\mathbf{H}_{c2}$ also have balanced columns in all columns except c and $\bar{c}$, because they are cosets of $\mathbf{H}_{c0}$. $\mathbf{H}_{c1}$ can be generated by adding any row in $\mathbf{H}$ that has a 1 in column c to every row in $\mathbf{H}_{c0}$, and $\mathbf{H}_{c2}$ can be generated by adding any row in $\mathbf{H}$ that has a 2 in column c to every row in $\mathbf{H}_{c0}$. These summations uniquely produce every row in $\mathbf{H}$ that has a one or two, respectively, in column c due to the closure of $\mathbf{H}$. These summations will also simply increment each column by a constant, so the distribution of elements in every column except c and $\bar{c}$ will be unchanged.

Selection Set

Because a selection set $\mathbf{Q}$ can be constructed by summing rows of $\mathbf{H}$ with the coefficients of $q_0(x)$, and therefore that $\mathbf{Q}$ is a coset of the group $\mathbf{H}$, it follows that the selection set can be sorted in a manner similarly to that previously described for the matrix of relationship sequences, $\mathbf{H}_c$. This sorting gives:

$$\mathbf{Q}_c = \begin{bmatrix} \mathbf{Q}_{c0} \\ \mathbf{Q}_{c1} \\ \mathbf{Q}_{c2} \end{bmatrix}$$

where $\mathbf{Q}_c$ is the sorted selection set, with $\mathbf{Q}_{c0}$, $\mathbf{Q}_{c1}$, and $\mathbf{Q}_{c2}$ being the subsets of $\mathbf{Q}_c$ with a 0, 1, or 2 in column c respectively. The example presented in Table 3.6 is continued in Equation 3.7 to illustrate this sorting. The selection set is formed by scrambling the source word {111111} with the scrambling polynomial $q(x) = x^2 + x + 2$. The matrix $\mathbf{Q}$ is comprised of the coefficients of each quotient in the selection set, and this matrix is sorted along column 0, the first column, to produce $\mathbf{Q}_0$. The sub-matrices $\mathbf{Q}_{00}$, $\mathbf{Q}_{01}$, and $\mathbf{Q}_{02}$ are separated to show that each contains a 0, 1, or 2 in column 0, respectively.

$$\mathbf{Q}_0 = \left[\begin{array}{cccccccc} 0 & 0 & 1 & 0 & 2 & 2 & 1 & 2 \\ 0 & 1 & 0 & 2 & 2 & 1 & 2 & 0 \\ 0 & 2 & 2 & 1 & 2 & 0 & 0 & 1 \\ \hdashline 1 & 0 & 2 & 2 & 1 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 & 1 & 0 & 2 & 2 \\ \hdashline 2 & 0 & 0 & 1 & 0 & 2 & 2 & 1 \\ 2 & 1 & 2 & 0 & 0 & 1 & 0 & 2 \\ 2 & 2 & 1 & 2 & 0 & 0 & 1 & 0 \end{array}\right] = \left[\begin{array}{c} \mathbf{Q}_{00} \\ \mathbf{Q}_{01} \\ \mathbf{Q}_{02} \end{array}\right] \tag{3.7}$$

Each of the subsets in the general $\mathbf{Q}_c$ has a balanced distribution of symbols in all columns except for c and $\bar{c}$. This property exists because $\mathbf{Q}_c$ is a coset of $\mathbf{H}_c$. Any row $\mathbf{q}_z$ from $\mathbf{Q}$ with a zero in column c can be added to each subset of $\mathbf{H}_c$ to generate $\mathbf{Q}_c$. Since $\mathbf{Q}_c$ can be formed by a constant addition to each row in $\mathbf{H}_c$, $\mathbf{Q}_c$ has the same column-wise properties as $\mathbf{H}_c$.

Since each subset in $\mathbf{Q}_c$ has an equal number of each symbol in all columns except c and $\bar{c}$, the total number of zeros in each subset can be determined. Column c of $\mathbf{Q}_{c0}$ contains only zeros. The number of zeros in $\mathbf{Q}_{c0}$ is thus determined by the symbol in column $\bar{c}$ of q_0. If this symbol is a zero, then column $\bar{c}$ is also composed entirely of zeros. In such a case, there are two columns with 3^{D-1} zeros and $3^D - 3$ columns with 3^{D-2} zeros, and the total number of zeros in $\mathbf{Q}_{c0}$ is $3^{2D-2} + 3^{D-1}$. Since $\mathbf{Q}_{c0}$ has 3^{D-1} rows, the mean number of zeros in each row is $\left(3^{2D-2} + 3^{D-1}\right) 3^{-D+1} = 3^{D-1} + 1$. Thus, $\mathbf{Q}_{c0}$ must contain at least one row with more than 3^{D-1} zeros, and such a selection set cannot be a worst-case set.

Furthermore, in the case where columns c and $\bar{c}$ of $\mathbf{Q}_{c0}$ are both zero, columns c and $\bar{c}$ in $\mathbf{Q}_{c1}$ must be non-zero and opposite, that is, one of the symbols is a one and the other is a two. This is true because $\mathbf{q}_z$ has zeros in columns c and $\bar{c}$ so its summation with $\mathbf{H}_{c1}$ has the same values in columns c and $\bar{c}$ as $\mathbf{H}_{c1}$ i.e., column c is entirely 1s and column $\bar{c}$ is entirely 2s. This property also holds for $\mathbf{Q}_{c2}$ by similar reasoning, with the values in column c being all 2s and $\bar{c}$ being all 1s.

In a worst-case selection set, $\mathbf{Q}_{c0}$ must have zeros in column c and one of the non-zero symbols in column $\bar{c}$. This arrangement gives $\mathbf{Q}_{c0}$ one column with 3^{D-1} zeros, one column with no zeros, and $3^D - 3$ columns with 3^{D-2} zeros. This gives a total of 3^{2D-2} zeros in $\mathbf{Q}_{c0}$. Since there are 3^{D-1} rows, the mean number of zeros per row must be 3^{D-1}. This implies that all rows of $\mathbf{Q}_{c0}$ must have exactly 3^{D-1} zeros in a worst-case selection set. Any row with fewer than 3^{D-1} zeros will force another row to have more than 3^{D-1} zeros, resulting in a selection set that is not a worst-case.

Any row in $\mathbf{Q}_c$ that contains zeros will appear in $\mathbf{Q}_{c0}$ when c is equal to a column in which the row has a zero. Given that $\mathbf{Q}_{c0}$ of a worst-case selection set has 3^{D-1} zeros in every row, all rows in a worst-case set with any zeros must have exactly 3^{D-1} zeros because each row will sorted into $\mathbf{Q}_{c0}$ for some c.

The selection set $\mathbf{Q}$ has 3^{D-1} zeros per column and 3^D-1 columns for a total of $3^{D-1}\left(3^D-1\right)$ zeros. In a worst-case selection set, these zeros are divided into rows each containing exactly 3^{D-1} zeros. As such, there are exactly 3^D-1 rows containing zeros. Since $\mathbf{Q}$ has a total of 3^D rows there must be exactly one row with no zeros.

Let the row with no zeros be called quotient $\mathbf{q}_\pi$. Since this row contains no zeros, it is not placed in $\mathbf{Q}_{c0}$ for any c. As noted previously, when the values in columns c and $\bar{c}$ of $\mathbf{Q}_{c0}$ are zeros, the values in these columns of $\mathbf{Q}_{c1}$ and $\mathbf{Q}_{c2}$ must be additive inverses. The converse of this is true as well. For instance, column c of $\mathbf{Q}_{c1}$ is composed of ones by definition, so if column $\bar{c}$ is composed of twos the value in column $\bar{c}$ of $\mathbf{q}_z$ is necessarily zero.

This indicates that in a worst-case selection set the values in column $\bar{c}$ of $\mathbf{Q}_{c1}$ must be either ones or zeros, and similarly the values in column $\bar{c}$ of $\mathbf{Q}_{c2}$

must be either twos or zeros. Also, column $\bar{c}$ must have a set of 3^{D-1} zeros exclusively in either $\mathbf{Q}_{c1}$ or $\mathbf{Q}_{c2}$ because the column as a whole must contain 3^{D-1} occurrences of each symbol. So, only one of the thirds will have identical elements in columns c and $\bar{c}$. Since $\mathbf{q}_\pi$ has no zeros it must always be sorted into this dense third. Therefore, the value in column c must be equal to the value in column $\bar{c}$ of $\mathbf{q}_\pi$ for all c, and the second half of $\mathbf{q}_\pi$ is identical to its first half.

The selection set shown in Equation 3.7 is a worst-case selection set. The second quotient in $\mathbf{Q}_{01}$ is $\mathbf{q}_\pi$ for that selection set. Note that it has no zeros, and that its first half is identical to its second half. Since this particular example has the all-one sequence as $\mathbf{q}_\pi$ the equivalence of the two halves may not be readily apparent. It is a special case of the defining characteristic of a $\mathbf{q}_\pi$ sequence: that it consists of the concatenation of two identical subsequences without zeros, a condition satisfied by, for instance, the sequence $\{12111211\}$.

The number of worst-case selection sets can be enumerated by enumerating the number of possible quotients q_π because each of these quotients corresponds to only one selection set and each worst-case selection set contains only one of these quotients. Since the quotient q_π consists of a subsequence of $S/2$ elements repeated twice, and each element must be either one or two, there are $2^{\frac{S}{2}}$ total possible quotients of this form.

The number of different source words is $3^W = 3^{S-D}$. Given the one-to-one mapping of source words to selection sets, there are 3^{S-D} unique selection sets of which $2^{\frac{S}{2}}$ are worst cases. The fraction of source words that will be represented by worst-case codewords is then $2^{\frac{S}{2}}/3^{S-D} < 3^{\frac{S}{2}}/3^{S-D} = 3^{\frac{-S}{2}+D}$, which becomes vanishingly small as sequence length S increases.

3.5 Simulation Results

Simulations have been designed to test the effect that the choice of scrambling polynomial has on the overall sparsity performance of a guided scrambling code. The first set of simulations test the sparsity performance of different polynomials of the same degree. The second set of simulations test the performance of primitive polynomials with different degrees.

Table 3.7: Degree-2 scrambling polynomial cohorts

Cohort	$d(x)$	Average Sparsity
1	x^2	50%
2	$x^2 + x$ $x^2 + 2x$	58.5%
3	$x^2 + 1$ $x^2 + 2$	59.25%
4	$x^2 + x + 1$ $x^2 + 2x + 1$	60%
5	$x^2 + x + 2$ $x^2 + 2x + 2$	60.75%

For every simulation only monic polynomials are considered. Polynomials with a leading coefficient of 2 perform identically to their additive inverse polynomials. This is a result of the linearity of the scrambling process. The selection set $\mathbf{Q}$ arising from any scrambling polynomial $d(x)$ and source word $w(x)$ is identical to $2\bar{\mathbf{Q}}$ where $\bar{\mathbf{Q}}$ is the selection set generated by the polynomial $2d(x)$ and the source word $w(x)$. Since the number and placement of zeros is identical between $\mathbf{Q}$ and $\bar{\mathbf{Q}}$ the sparsity performance of $d(x)$ is identical to the sparsity performance of $2d(x)$.

3.5.1 Primitive Scrambling Polynomials

The scrambling polynomial is important because it defines the correspondence between each input data block and each codeword. A poor choice of scrambling polynomial can result in a system which maps several of the more sparse codewords to the same source word, causing more source words to be represented by codewords with few zeros. Therefore, the choice of scrambling polynomial can affect the sparsity in the scrambled data without affecting the rate.

Table 3.7 lists all possible monic scrambling polynomials of degree 2. For each scrambling polynomial, every length-6 input word was augmented with two symbols and this augmented word was scrambled to produce a set of

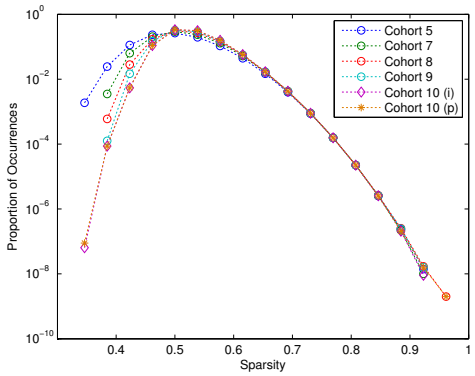


Figure 3.3: Sparsity performance of degree-3 polynomials

nine quotients. The quotient with the highest number of zeros was selected. The right column lists the average sparsity of the selected codewords for each scrambling polynomial.

Some of the polynomials produced identical average sparsity values. These polynomials were grouped together to produce a set of five cohorts of polynomials. The two polynomials in the cohort with the highest values are both primitive polynomials, and the only primitive polynomials, of degree 2 in $GF(3)$.

Simulations were also conducted for higher degree polynomials. In the case of degree-3 polynomials, every monic degree-3 scrambling polynomial was used to encode one billion random source words. The occurrence of each sparsity value was recorded; this data is presented in Figure 3.3. The horizontal axis records sparsity values, and the vertical axis indicates the number of times that a codeword of each sparsity value occurred, divided by the number of codewords that were tested. A log scale is used on the vertical axis to help distinguish the lower valued data points.

In the degree-3 tests, the polynomials are divided into 10 cohorts with equal sparsity performance. Table 3.8 lists the members of each of these cohorts as well as the average sparsity of each cohort. The polynomials of cohort 10 are divided into two subgroups. The polynomials above the dashed line are irreducible, and the polynomials below the dashed line are primitive. This cohort contains all the degree-3 irreducible and primitive monic polynomials.

Table 3.8: Degree-3 scrambling polynomial cohorts

Cohort	Polynomials	Average Sparsity
1	x^3	41.025%
2	$x^3 + x^2 \quad x^3 + 2x^2$	47.824%
3	$x^3 + x \quad x^3 + 2x$	49.512%
4	$x^3 + x^2 + x \quad x^3 + 2x^2 + x$	50.264%
5	$x^3 + 1 \quad x^3 + 2$	50.321%
6	$x^3 + x^2 + 2x \quad x^3 + 2x^2 + 2x$	50.608%
7	$x^3 + x^2 + x + 1$ $x^3 + 2x^2 + x + 2$	51.536%
8	$x^3 + x^2 + 2x + 2$ $x^3 + 2x^2 + 2x + 1$	52.259%
9	$x^3 + x + 1 \quad x^3 + x + 2$ $x^3 + x^2 + 1 \quad x^3 + 2x^2 + 2$	52.589%
10	$x^3 + 2x + 2 \quad x^3 + x^2 + x + 2$ $x^3 + x^2 + 2 \quad x^3 + 2x^2 + 2x + 2$ $x^3 + 2x + 1 \quad x^3 + x^2 + 2x + 1$ $x^3 + 2x^2 + 1 \quad x^3 + 2x^2 + x + 1$	52.982%

For clarity, the results for only one member of each cohort are plotted in Figure 3.3, with the exception that two polynomials from cohort 10 are plotted. Additionally, polynomials from cohorts 1, 2, 3, 4, and 6 are not represented because these polynomials have a zero constant term which means that they are effectively lower degree scrambling polynomials operating with a delay.

Finally, note that with degree-3 scrambling polynomials, irreducible polynomials perform as well as primitive polynomials in terms of average sparsity. In Figure 3.3, results for both an irreducible polynomial and a primitive polynomial from Cohort 10 are represented; the data points of each plot are marked with diamonds and asterisks respectively. These irreducible polynomials produce "half" m-sequences as their respective relationship sequences. These sequences differ from true m-sequences in that their first and second halves are identical rather than additive inverses. A set of relationship sequences composed of half m-sequences performs similarly to a set composed of m-sequences because such a set contains the additive inverse of each half m-sequence due to the linearity of the division operation and the fact that all possible augmenting sequences are used to determine the set of relationship sequences. The analogue of $\mathbf{q}_\pi$ for irreducible scrambling polynomials consists of sequences that contain no zeros and the two halves of which are additive inverses, such as $\{12112122\}$ for example. Further work is required to determine why irreducible polynomials of degree-3 produce half m-sequences but irreducible polynomials of degree-2 and degree-4 do not.

Similar tests were conducted for degree-4 polynomials. Instead of testing all possible monic polynomials, simulations were run for the reducible polynomial x^4+2x+1, the irreducible, non-primitive polynomial $x^4+2x^3+x^2+1$, and the primitive polynomial x^4+x+2. For each polynomial, 100 million pseudo-random length-76 input words were encoded using 4 augmenting symbols. These results are shown in Figure 3.4. The primitive polynomial produced codewords with the highest average sparsity with 46.62%, and in this simulation the irreducible polynomial performed worse than the primitive polynomial with an average sparsity of 46.39%. As expected, the reducible polynomial produced the least sparse encoded words with 46.16% average sparsity.

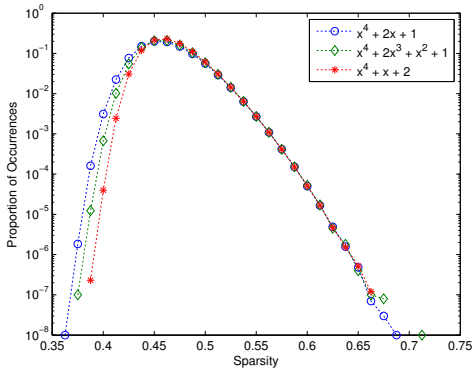


Figure 3.4: Sparsity performance of degree-4 polynomials.

3.5.2 Degree of Scrambling Polynomial

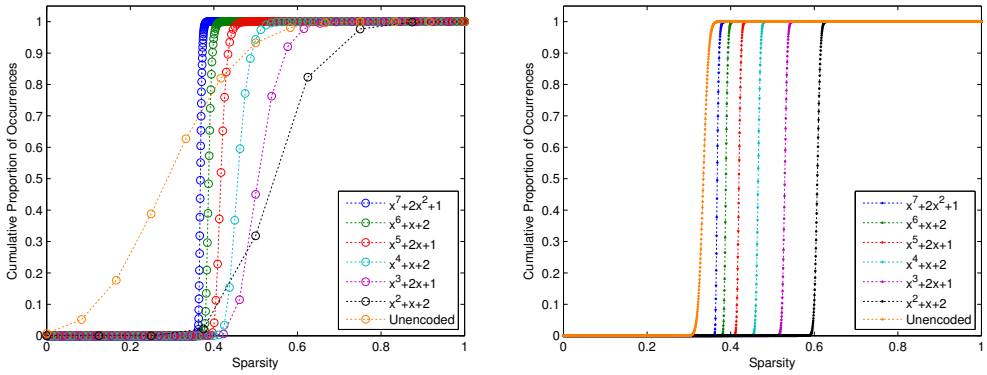
The degree of the scrambling polynomial is another important component of the encoding system. In the analysis presented in Section 3.4, where $D = A$, the degree of the scrambling polynomial affects the number of augmenting symbols, the length of the source words, the length of the codewords, and thereby the coding rate. Additionally, that analysis only holds for primitive scrambling polynomials, because they were found to outperform non-primitive polynomials in terms of average sparsity, as shown in the previous subsection. As a result only primitive polynomials were used in the following simulations.

Simulations were designed to test the sparsity of selected codewords for various degrees of primitive scrambling polynomials. For the degree-2 case, the polynomial $d(x) = x^2 + x + 2$ was chosen. In this case, there are only 729 possible source words, so the simulation considered all possible source words.

In the degree-3 case, the primitive polynomial $d(x) = x^3+2x+1$ was chosen as the scrambling polynomial. Since the number of possible source words is 3^S where $S = 3^D - D - 1 = 23$, encoding all source words is intractable for degree-3 and higher simulations. Instead, a number of pseudo-randomly generated source words are encoded, and a count of the resulting sparsity of each codeword is recorded. The counts are then divided by the number of encoded source words to give a distribution of the sparsity performance of the scrambling polynomial. In the degree-3 case, one billion source words were encoded.

Table 3.9: Average Sparsity for Primitive Scrambling Polynomials

Degree	Scrambling Polynomial	Rate	Average Sparsity
N/A	Unencoded	1	33.50%
2	$x^2 + x + 2$	$6/8$	60.75%
3	$x^3 + 2x + 1$	$23/26$	52.98%
4	$x^4 + x + 2$	$76/80$	46.62%
5	$x^5 + 2x + 1$	$237/242$	42.04%
6	$x^6 + x + 2$	$722/728$	38.91%
7	$x^7 + 2x^2 + 1$	$2179/2186$	36.85%



(a) Single Blocks (b) Concatenated Blocks

Figure 3.5: Experimentally derived sparsity CDFs for single blocks and concatenated blocks using primitive scrambling polynomials of different degree.

In the degree-4 case, the primitive polynomial $d(x) = x^4 + x + 2$ was chosen as the scrambling polynomial, and 100 million pseudo-random source words were scrambled. For the simulations of degree 5, 6, and 7 the primitive polynomials $x^5 + 2x + 1$, $x^6 + x + 2$, and $x^7 + 2x^2 + 1$ were used as scrambling polynomials respectively, and 25 million pseudo-random source words were encoded using each polynomial. Finally, as a point of reference, a set of 25 million pseudo-random binary sequences were converted to ternary form using the 19:12 code described in Section 3.2. The average sparsity of codewords using each of these polynomials as well as the unencoded ternary data is shown in Table 3.9.

Each simulation of a scrambling polynomial resulted in a frequency distribution of the sparsity in a single block of encoded data for that polynomial.

Table 3.10: Number of symbols and concatenated blocks calculated in Figure 3.5b.

Scrambling Polynomial	Block Length	Enc. Blocks	Enc. Symbols
$x^2 + x + 2$	8	273	2184
$x^3 + 2x + 1$	26	84	2184
$x^4 + x + 2$	80	27	2160
$x^5 + 2x + 1$	242	9	2178
$x^6 + x + 2$	728	3	2184
$x^7 + 2x^2 + 1$	2186	1	2186
Unencoded	12	182	2184

The cumulative sum of a frequency distribution yields an experimental cumulative distribution function (CDF) of the sparsity of a single block of encoded data. These CDFs are plotted for each scrambling polynomial in Figure 3.5a.

Since the blocks have different lengths for scrambling polynomials with different degrees, the granularity of each CDF is different. A more uniform comparison is achieved by considering the sparsity of sequences of encoded symbols of approximately the same length. To obtain this result, the single-block frequency distribution is used to calculate a probability density function (PDF) of concatenated blocks for each scrambling polynomial from Figure 3.5a. To determine the PDF of k concatenated blocks, the single-block frequency distribution is convolved with itself k times.

Table 3.10 lists the number of blocks that are concatenated for each polynomial such that encoded sequences of approximately the same length are considered. The CDFs of the concatenated sequences, plotted in Figure 3.5b, have more similar granularity.

From these CDFs, it can be seen that as the degree of the scrambling polynomial is increased the relative sparsity of the encoded words decreases. This result is expected in light of the fact that the rate of the guided scrambling code increases drastically with increases to the degree of the scrambling polynomial. An increased rate means that a higher proportion of all possible length-S words must be used as codewords to represent data, which forces less sparse words to be selected as codewords, resulting in poorer sparsity performance as the rate increases. All coded systems exhibit higher sparsity than

what is expected without sparsity encoding.

3.6 Conclusion

A coding scheme that algorithmically maps binary data to sparse ternary data has been presented. The encoding is performed in a two step process. Firstly, the input binary data is converted into a ternary data stream with a high rate. The base conversion algorithm is relatively simple, and for small block lengths the process can be efficiently implemented with a look-up table.

Secondly, the ternary data is made sparse through the use of guided scrambling. The scrambling step guarantees a minimum worst-case sparsity of ${}^{3^{D-1}}/_{3^D-1}$, and this worst case occurs very rarely as D increases. Simulations demonstrated an average sparsity in excess of 60% with a code rate of 0.75, diminishing to 36.9% with a code rate of 0.997, as compared to an average sparsity of approximately ⅓ that is expected without sparsity encoding.

Chapter 4

Selective Phase Masking

Holographic data storage as described in Chapter 2 is a two-dimensional storage system, in contrast to one-dimensional storage systems such as magnetic storage systems and optical storage systems, where the dimensionality refers to the data structures in which information is contained. Two-dimensional holographic systems store data as pages of pixels, whereas magnetic and optical drives store data as a spiral on a platter or disk, respectively.

Applying constraints to the recorded data allows a level of control over the physical process of storing the data. In the case of holographic storage, as discussed in Section 2.7, some data patterns can cause the holographic material to saturate at certain locations. Upon readout, the data array will likely contain errors, due to the corrupted recording. For a practical holographic storage device to be realized, safeguards must be established to ensure that either no material-saturating data arrays are recorded or the probability of a material-saturating array being recorded is remote enough so as to be negligible. Constrained coding methods can be utilized to provide such safeguards.

The most common types of two-dimensional constraints are run-length limits and the "no isolated bits" constraint [41]. These two constraints chiefly increase the detector's ability to discriminate between different channel symbols and keep track of pixel spacing.

However, these constraints do not address system errors that arise in holographic storage systems from either periodic bit patterns or a significant DC

component within a data array. As described in Section 2.7, periodic data patterns or a large DC component will create Fourier images with large peaks, and thereby cause saturation of the material used to store holographic images leading to readout errors.

Designing an encoder to dynamically map the input data to pixel values while minimizing the DC and periodic content of the data array is a significant combinatoric challenge. This is because, in the two-dimensional array, repetitive pixel patterns in any orientation can result in large peaks in the Fourier image. To constrain these cases the encoder would be required to eliminate or minimize periodic pixel trends in two dimensions. An encoder that serially maps incoming data to an array would be required to ensure that data arrays do not have periodic components among all possible directions in which periodicity could be oriented. Given the difficulty of doing so, more practical page-oriented approaches are considered.

This chapter discusses selective phase masking, a practical method to reduce the occurrence and severity of peaks in the Fourier domain representation of data arrays. Additionally, a similar technique involving interleavers is presented, but it is shown not to perform as well as phase masks. Both of these techniques are similar to techniques used in OFDM systems to address the problem of a high peak to average power ratio [42].

4.1 Introduction

Phase masks are used to reduce the peaks in a Fourier image by altering the phase of a subset of the pixels in the data array. This method was originally proposed using an etched glass slide to alter pixel phase, and conceived as a means to reduce the DC peak in systems which record data as strictly ON or OFF pixels[43, 44, 45]. However, using a physical device to impart these phase shifts introduces additional difficulties, as discussed below.

In this section, phase masks implemented at the coding level are discussed, and this procedure and its effects in the Fourier domain are described mathematically. Finally, a method to test the efficacy of a system selectively employing multiple phase masks at the coding step is introduced. The results of

several tests are presented in the following sections to demonstrate the utility of selectively employing phase masks at the coding step.

4.1.1 Physical-Level Phase Masks

The use of phase masks to reduce the peaks in the Fourier image of a data array for holographic storage was first demonstrated in 1970 by Burckhardt[43]. The original method involved fabricating a mask that shifted the phase of some pixels by π radians. This was accomplished by etching a piece of glass such that the glass corresponding to a random selection of data pixels was thicker than the glass corresponding to the others. The difference in thickness was chosen to introduce a phase delay of one half-wavelength.

This method of using a static phase mask was first proposed for systems using amplitude modulation with ON and OFF pixels representing binary data. For such modulation schemes, the DC component of the Fourier image tends to be the dominant cause of material saturation. The phase mask diminishes the DC peak because ON pixels of differing phase interfere destructively at DC in the Fourier domain.

Physical phase masks have a few weaknesses, however. Firstly, while it may result in improved performance with some data arrays, it does not always result in an improvement over an unmasked array. In the worst case for an unmasked array, the all ON case, a physical random phase mask which alters the phase of half of the pixels would remove the DC component completely. However, it is unlikely that any practical system would allow an all ON array to be recorded in the first place. Most holographic storage systems described in the literature make use of fairly restrictive sparsity codes that force some percentage of the data to be recorded as OFF pixels. A code that enforces an equal probability of ON and OFF pixels would produce arrays that are just as problematic for the phase masked system as for the unmasked system. The two worst-case arrays would be the array in which the ON pixels correspond exactly with the zero phase-shifted pixels in the phase mask, and its complement, the array in which the ON pixels correspond exactly with the π phase-shifted pixels. In these situations, the phase mask would do nothing to diminish the DC peak

in the Fourier image.

Other poor-case arrays are arrays in which the resulting phase-masked array is unbalanced. Here, balance refers to the proportion of π phase-shifted ON pixels to zero phase-shifted ON pixels. An array in which the two are exactly equal is said to be perfectly balanced. The worst-case arrays discussed in the previous paragraph would be described as being completely unbalanced. Also, partially unbalanced arrays can result in DC values above the saturation limit for data arrays in a system that employs a physical phase mask.

The second major drawback for a physical phase mask is that it is an optical element. This requires very precise alignment between the phase mask and SLM, in all three spatial directions[46, 47, 48]. Additionally, the phase mask introduces diffraction effects which lead to cross-talk between adjacent pixels[49, 50].

4.1.2 Selective Phase Masks

An alternative to physical phase masking is to utilize an SLM that can control the phase of the data pixels. It is then possible to use a phase mask during the coding procedure, rather than a physical optical element. A phase mask of this type is an array with size equal to that of the data array, and values of its elements are either $+1$ or -1. The phase mask array and data array are then multiplied element-wise to produce the masked data array that will be recorded. For clarity: in this thesis the term "data array" always refers to the unmasked array of ternary symbols; the term "mask array" refers to the array of ± 1 values that represents the phase mask; and the term "masked array" refers to the array resulting from pixel-wise multiplication of the data array and the mask array.

There are several advantages of this method over use of a physical phase mask, two of which are highlighted here. Firstly, since the mask is not a physical device, there are no diffraction effects or alignment requirements beyond those inherent in the holographic storage system.

Secondly, it is possible to use multiple phase masks within the same system. To introduce multiple physical-level phase masks, some device would be

required to move the various masks into and out of position while maintaining the stringent alignment requirements. In the case of masking at the coding step, however, the only requirement is storing each mask in memory. This allows the storage system to evaluate the performance of several different phase masks for a given data array, and to choose the one with the lowest probability of saturating the medium. Doing this reduces both the number and severity of poor-case data arrays that the system may encounter, assuming good choices have been made in the design of the set of masks to be used.

These two advantages eliminate the main difficulties of using physical phase masks. The primary drawback of selective phase masks is the increased complexity of the overall system. An encoder must determine which of the masked data arrays provides the best Fourier domain properties. This requires calculating the Fourier transform for each possible masked array, recording the peak value of the transform, and choosing the mask which generates the lowest peak value, or at least choosing a mask that ensures all spectral components of the masked array are below a predetermined threshold. This is a significant cost, but if it is able to provide a holographic system that is free from material saturation it may be worthwhile.

With respect to phase masking at the coding step versus using a physical phase mask, this work is focused solely on the coding-level phase masks. From this point onward any use of the term "phase mask" is in reference to a phase mask or phase masks applied at the coding level.

4.1.3 Mathematics of Phase Masks

As described in Section 2.7, data arrays can be represented as two-dimensional discrete functions. The allowed values of the elements in the array are determined by the type of modulation method in use. In general, a single element in the array will have the form $f[m,n] = a_{m,n}e^{i\theta_{m,n}}$, where $a_{m,n}$ represents the amplitude and $\theta_{m,n}$ represents the phase of the pixel at position (m,n). This general format allows for any number of amplitude and phase levels to be utilized in the modulation scheme.

For the purposes of this work, only hybrid-ternary modulation (HTM) is

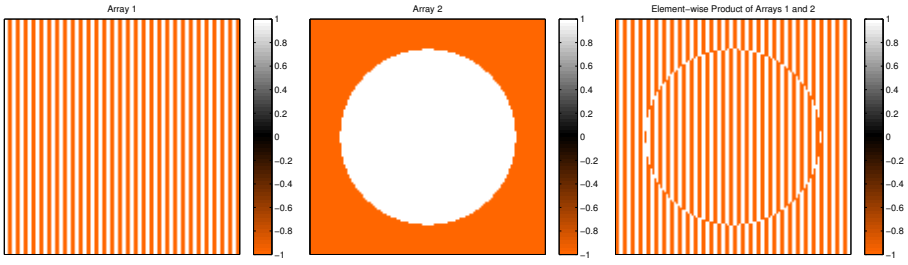
considered. HTM is represented by either $a_{m,n} \in \{-1, 0, 1\}$ and $\theta_{m,n} = 0$, or $a_{m,n} \in \{0, 1\}$ and $\theta_{m,n} \in \{0, \pi\}$, with the understanding that $\{a_{m,n} = 0, \theta_{m,n} = 0\}$ and $\{a_{m,n} = 0, \theta_{m,n} = \pi\}$ represent the same modulation symbol. Since the two representations are mathematically identical, the choice of which notation to use is arbitrary. For simplicity, the first description is used henceforth. This allows dispensing with the phase term altogether, and elements in the data array are represented with just a single value, $a_{m,n}$.

The mask or masks can similarly be represented as a two-dimensional discrete function. This chapter is focused entirely upon the use of binary phase masking, where only two values are used for an individual element in the phase mask: either 1 or -1. The 0 value that is used in HTM is not used in the phase mask array because a non-zero original data value would be impossible to recover after the multiplication with 0. Using the same notation as before, this is described as either $a_{m,n} \in \{1, -1\}$ and $\theta_{m,n} = 0$ or $a_{m,n} = 1$ and $\theta \in \{0, \pi\}$. The choice of which notation to use is arbitrary; as before, the first is chosen for simplicity.

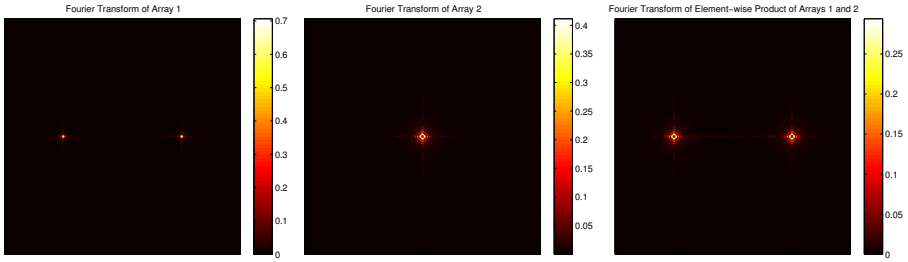
The process of masking a data array is described mathematically as the element-wise multiplication of the data array with the mask array. This two-dimensional multiplication results in a two-dimensional convolution of the Fourier transforms of the data array and phase mask. Because of this, the Fourier transform of the phase mask itself has a critical influence on the performance of the system as a whole. Figure 4.1 illustrates this convolution in the Fourier domain. In this figure and all subsequent figures, only the magnitude of the complex-valued Fourier transforms are depicted; phase plots of the Fourier images are not shown, since it is only the magnitude of spectral peaks that is of concern.

4.1.4 Evolutionary Algorithm Simulations

The ideal metric to determine the efficacy of a peak reduction technique is the peak value of the worst-case data array. However, finding the worst-case data array is no simple task. There are 3^{N^2} possible arrays of size $N \times N$. For values of N on the order of 100, a brute-force search to find the absolute



(a) Element-wise Multiplication of Data Array and Mask



(b) Magnitude of FT of Data Array, Mask, and Element-wise Product

Figure 4.1: Element-wise multiplication in the data array results in a convolution in the Fourier domain.

worst-case is practically impossible.

In order to find an array that produces poor results, an evolutionary algorithm was developed. The algorithm begins with some number of random arrays which represent the primary generation. It then finds the best way to mask each of the primary arrays. The best choice of mask is the mask that results in the lowest peak value in the Fourier transform of the masked array. Then, the array with the highest peak Fourier-domain values in its best choice masked array is selected as the surviving array. In the search for a problematic array, the algorithm selects the data array with the worst best-case peak values to be the surviving array.

The surviving array is then mutated very slightly by adding a random array, called the mutation array, that consists of zeros with 99.95% probability and ones with 0.05% probability. The probability of changing the value of an element in the data array must be very low. Initially it was set higher, but the algorithm did not focus on one particular solution and instead alternated between several differing solutions. Keeping the probability of change low

reduces the scope over which the algorithm searches for data arrays with large peaks in their Fourier transforms, encouraging it to settle on a single solution.

After the mutation array is added to the data array, any elements in the array with a value of 2 are set to -1. The algorithm also keeps track of the proportion of ON and OFF pixels in the data array. This feature was implemented to simulate an enforced sparsity constraint. Also, if the algorithm is allowed to increase the number of ON-pixels without bound, it will do so rather than evolve a single solution. The constraint is enforced as an upper bound on the percentage of the array that can be composed of ON pixels. If a mutated array contains ON pixels in excess of this percentage, the algorithm changes some ON pixels to OFF pixels. The number of ON pixels that are made OFF is chosen so that the mutated array will be within compliance. The specific ON pixels to be made OFF are chosen at random. The balance between ON pixels with 0 and π phase shifts is not controlled by the algorithm. A more in-depth description of this algorithm is found in Appendix A.

This mutation is repeated some number of times until a new generation of arrays is created. The process repeats with the selection of the best choice of mask for each of the new arrays. The simulation runs for some specified number of generations before halting.

The variable parameters for the evolutionary algorithm simulations are the number of arrays in each generation, the number of generations that the algorithm simulates, and the size of the arrays. The mutation probability and sparsity constraint are constant for all simulations in this chapter.

4.2 Phase Mask Design

For a set of masks to be ineffective in generating at least one array with minimal periodic components, every possible masked version of the data array, and the data array itself, must have Fourier transforms with large peaks. This happens through constructive interference in the convolution of the Fourier transforms of the mask and data array. The scaled versions of the Fourier transform of the masks must sum to produce large peaks at some points in the Fourier transform of the resulting masked data array.

An early approach taken to generate a set of phase masks was to design masks based on their Fourier transforms. The goal was to determine characteristics in the Fourier domain that contribute, either positively or negatively, to the performance of either a single mask or a set of masks. The starting point for this task was the convolution relation between the Fourier transform of the phase mask and the Fourier transform of the data array.

The Fourier transform of a data array that saturates the holographic medium will contain one or more large peaks. The convolution of the Fourier transform of the data array with the Fourier transform of the mask is the summation of "copies" of the Fourier transform of the mask centered at the location of each sample value in the Fourier transform of the data array. Also, each "copy" of the Fourier transform of the mask is scaled by the value upon which it is centered. Because of this scaling, the copies of the Fourier transform of the mask which are centered upon the peaks in the Fourier transform of the data array have a much larger scaling factor than the copies that are centered upon non-peak values of the Fourier transform of the data array. It is difficult to determine generally whether the few, larger-scaled copies have a greater effect on the resulting convolution than the more numerous, yet smaller-scaled copies.

Several designs of phase masks that approximate broad Gaussian pulses were investigated. The motivation for masks of this style is the idea that masks exhibiting a large pulse in the Fourier domain would distribute the intensity from peaks in the Fourier transform of a data array across a broader area, and therefore not allow a data array to saturate the medium in every masked or unmasked case.

As a contrast to the designed masks, phase masks with values chosen pseudo-randomly were also investigated. These types of phase masks have low peak values in the Fourier domain.

4.2.1 Gaussian Pulses

The first simulation used a set of four masks, representing four pulses of varying width in the Fourier domain. For this first set of masks, the requirement

that the masks be composed entirely of ± 1 values was removed. This restriction was removed so that the simulation would provide the effect of using Gaussian pulses for masks, which would then serve as a baseline for comparison of the results obtained from masks that emulate Gaussian pulses in the Fourier domain but are composed solely of ± 1 values.

To construct the pulses that would serve as the masks, a script was written to generate a two-dimensional Gaussian pulse. The function first generates an array of size $N \times N$. The function then creates two indexing vectors a and b for the vertical and horizontal indices of the array, respectively. The elements of the mask array are calculated as $e^{-(a^2+b^2)}$. The pulse width can be altered by changing the values of a and b. For instance, if a and b are vectors equal to $[-2:2]$, the pulse in the mask array will be wider than a pulse calculated with a and b equal to $[-5:5]$.

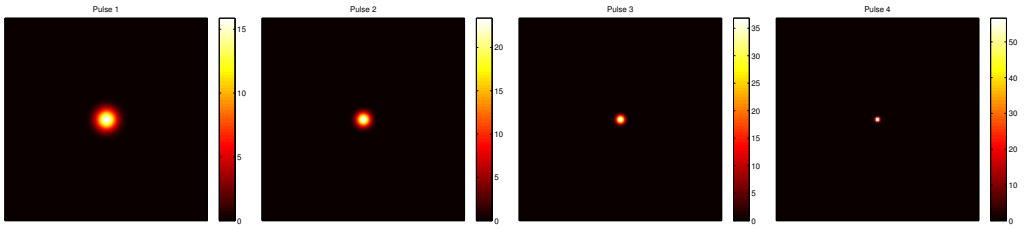
For uniformity between this simulation and others, the total power represented by each mask must be constant, otherwise the masks themselves scale the data array, thereby distorting the peak values of the transforms of the masked data arrays. For a mask comprised entirely of ± 1 values, the total power is $\sum_m \sum_n |\mu[m,n]|^2 = N^2$, where $\mu[m,n]$ represents pixel values in a phase mask of size $N \times N$. This scaling can be written as:

$$\mu[m,n] = \mu_u[m,n] \cdot \sqrt{\frac{N^2}{\sum_m \sum_n |\mu_u[m,n]|^2}}$$

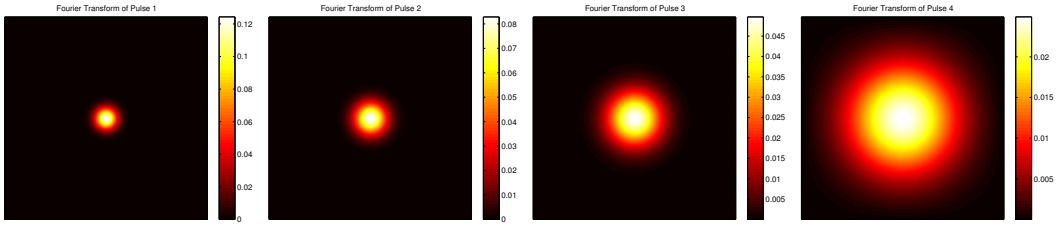
where $\mu_u[m,n]$ is the unscaled mask array.

As Figure 4.2 shows, there exists an inverse relationship between the widths of pulses in the two domains. The narrower pulse masks have broader distributions of power in the Fourier domain. For a pulse mask to be successful in redistributing the peak power of a Fourier image, a relatively broad pulse in the Fourier domain is required.

Because of the inverse relation, however, a mask with a broad Fourier pulse allows only a few pixels in the center of the array to dominate the masked data array. For example, Pulse 4, shown in Figure 4.2, contains only 16 pixels with a value greater than 1. The non-uniform nature of the pixels within Gaussian pulse masks raises several impediments to a practical implementation. The



(a) Gaussian Pulse Masks



(b) Fourier Transforms of Gaussian Pulse Masks

Figure 4.2: Gaussian Pulse Masks and Fourier Transforms

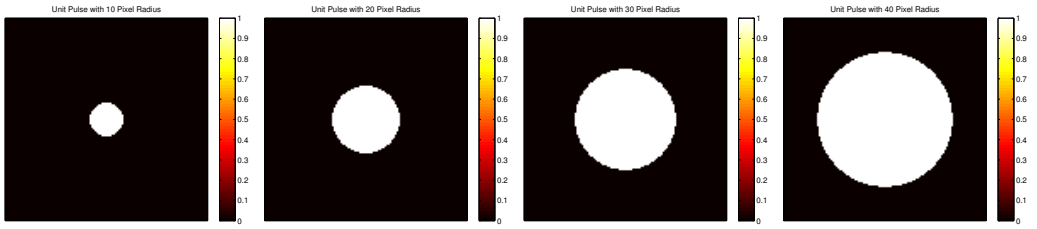
feasibility of these types of masks is discussed in Section 4.2.6.

4.2.2 Unit Pulses

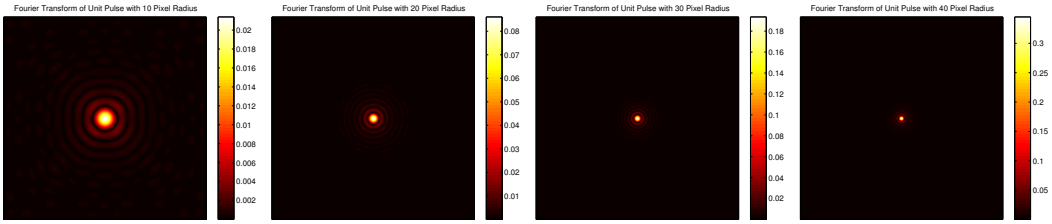
Practical phase masks should only be composed of ± 1 values. With this constraint, the challenge is creating phase masks that emulate the Fourier properties of the Gaussian pulse masks. The sinc[1] function shares some features with the Gaussian pulse, namely a large central distribution of power. Unlike the Gaussian pulse, however, the sinc has a Fourier transform that is more amenable to the constraints required for a practical phase mask.

In the one-dimensional case, a single square pulse has the sinc function as its Fourier transform. In the two-dimensional case, a circular unit-pulse function has a circular sinc as its Fourier transform. The width of the central lobe in the sinc function is inversely proportional to the width of the circular pulse. A circular pulse with a smaller radius will result in a sinc that distributes its power more broadly. Figure 4.3 illustrates the relationship between the radius of the circular pulse and the width of the central lobe of the corresponding

[1] $\text{sinc}(x) = \begin{cases} \frac{\sin(x)}{x} & x \neq 0 \\ 1 & x = 0 \end{cases}$



(a) Circular Pulses



(b) Sinc Functions

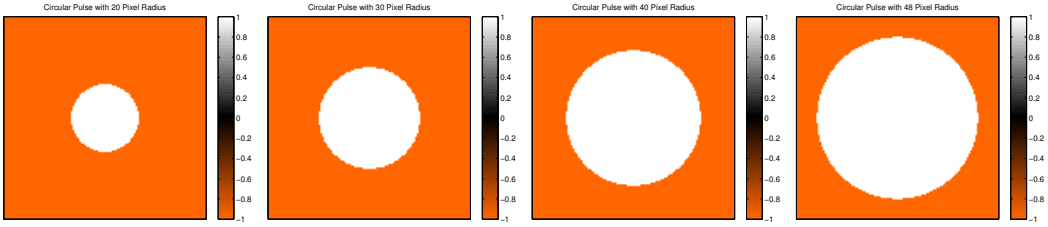
Figure 4.3: Fourier Pairs: Circular Pulses and Sincs

sinc function.

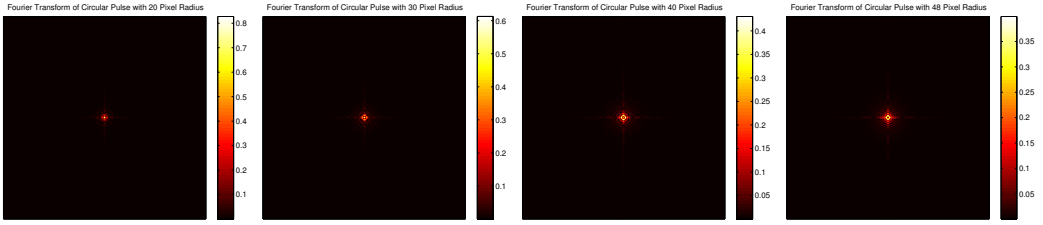
However, the circular unit-pulses shown in Figure 4.3 are not appropriate phase masks because of the zeros contained outside the pulses. To convert the unit-pulses to phase masks, the zero pixels must be translated to -1's. The Fourier transform of a mask of this type is the sum of the Fourier transform of a unit pulse, scaled by a factor of 2, and the Fourier transform of an array consisting entirely of -1 values. The Fourier transform of a uniform array of ON-pixels is simply a large peak at DC. This DC term tends to dominate the Fourier transform of the mask if the contribution from the circular unit-pulse is relatively insignificant. This is the case for circular unit-pulses with small radii, particularly those with the broadest central pulses in the Fourier domain.

As shown in Figure 4.4, the very broad Fourier domain pulses shown in Figure 4.3 are no longer attainable with the ± 1 constraint applied. A balance must be struck between making the circular pulse as narrow as possible, while not allowing the mask to become too unbalanced. An obvious choice is to construct a mask that is as balanced as possible. Finding the appropriate radius to give a balanced mask is a geometric problem.

Figure 4.5a shows a diagram of the problem. The solution is to find values

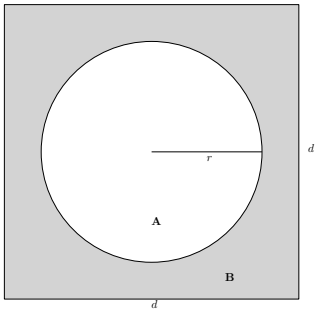


(a) Circular Pulse Masks

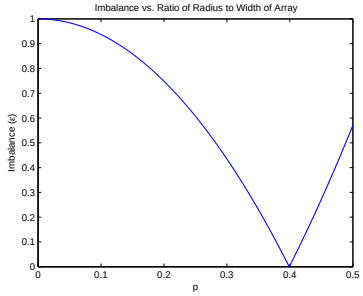


(b) Fourier Transforms of Circular Pulse Masks

Figure 4.4: Circular Pulses as Phase Masks and the corresponding Fourier Transforms



(a) Geometry of the Circular Mask



(b) Plot of ϵ vs. p

Figure 4.5: Geometric Diagram and Plot of ϵ

of r, the radius of the circle, and d, the length of a side of the square, such that the two areas, $\mathbf{A}$ and $\mathbf{B}$ are equal. The expression for $\mathbf{A}$ is the area of a circle of radius r:

$$\mathbf{A} = \pi r^2$$

The area of $\mathbf{B}$ is the area of the square minus the area of section $\mathbf{A}$:

$$\mathbf{B} = d^2 - \mathbf{A}$$

A value for the imbalance of the two areas, proportional to the total area of the square, can be written as:

$$\begin{aligned} \epsilon &= \frac{|\mathbf{A} - \mathbf{B}|}{d^2} \\ &= \frac{|2\pi r^2 - d^2|}{d^2} \end{aligned}$$

Rewriting r as pd, where $p = {}^{r}/_{d}$ gives:

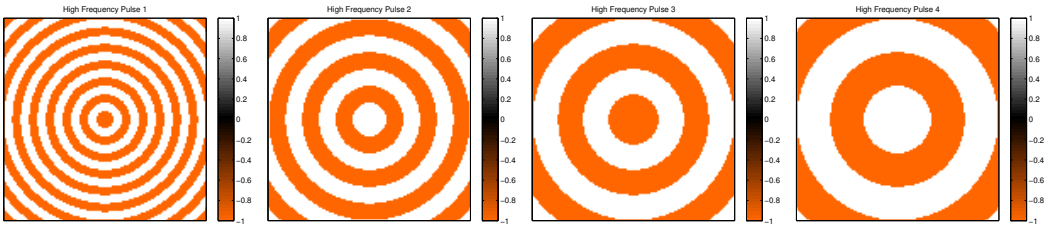
$$\begin{aligned} \epsilon &= \frac{|2\pi (pd)^2 - d^2|}{d^2} \\ &= \left|2\pi p^2 - 1\right| \end{aligned}$$

Solving for $\epsilon = 0$ gives:

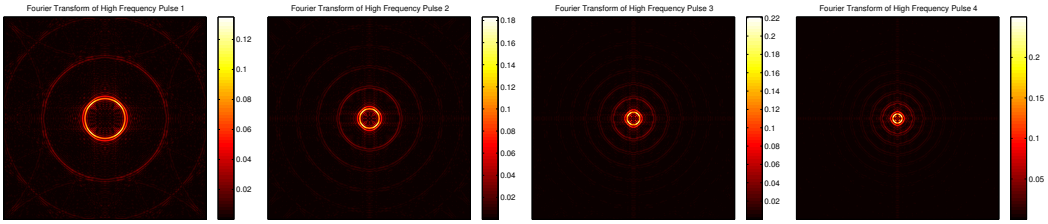
$$p = \sqrt{\frac{1}{2\pi}}$$

Therefore, the ratio of the radius of the circle to the width of the array should be as close to $\sqrt{{}^{1}/_{2\pi}}$ as possible to produce the most balanced mask. An example of this is the fourth mask shown in Figure 4.4. The array itself if 120×120 pixels, and the radius of this circular pulse is 48 pixels. This gives a ratio of 0.4, which is as close to $\sqrt{{}^{1}/_{2\pi}}$ as possible given the discrete nature of the mask array.

The balanced array still has a fairly narrow distribution of power in the Fourier domain. As such, the convolution of this mask's Fourier transform and that of a data array with large Fourier peaks will not be very effective in



(a) High Frequency Unit Pulse Masks



(b) Magnitude Plots of Fourier Transforms of High Frequency Unit Pulse Masks

Figure 4.6: High Frequency Unit Pulse Masks and the corresponding Fourier Transforms

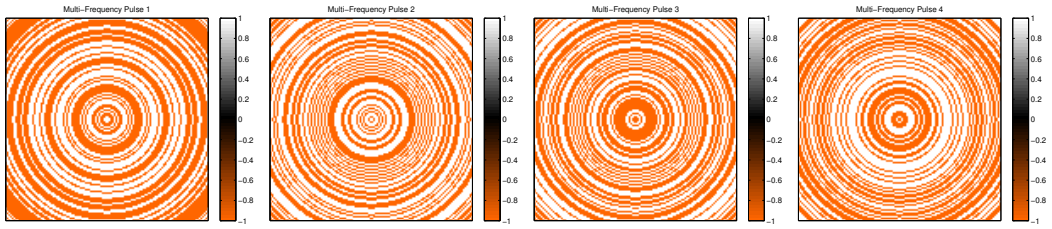
reducing the peaks.

4.2.3 High Frequency Unit Pulses

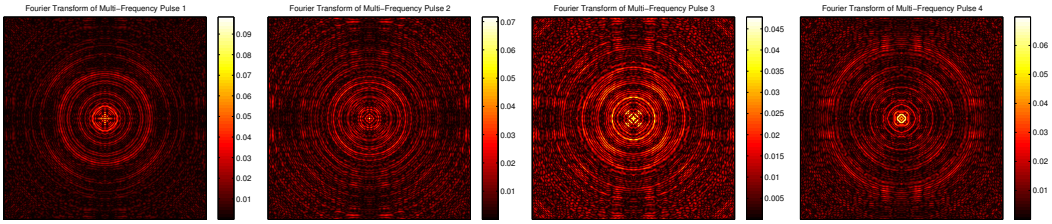
An extension of circular unit pulses is high frequency unit pulses. Rather than consisting of a single circular pulse, high frequency unit pulses are constructed from several concentric pulses. The process for combining the single pulses begins with constructing one pulse with a large radius, then subtracting a pulse with a smaller radius, then adding yet a third pulse with a still smaller radius, and so on.

Since the periodic component of the mask is circular, the power in the Fourier domain is distributed circularly as well. A constant decrease in pulse radius gives a mask with a significant portion of its power distributed at a single frequency. This frequency is the inverse of the period of change in pulse radius. The radius of the circle in the frequency domain represents this frequency.

As shown in Figure 4.6, a smaller change in radius of the circles used to create the masks (that is, a mask with a smaller period) results in a mask



(a) Multi-Frequency Unit Pulse Masks



(b) Fourier Transforms of Multi-Frequency Unit Pulse Masks

Figure 4.7: Multi-Frequency Unit Pulse Masks and the corresponding Fourier Transforms

with more of its power distributed at higher frequencies. This is illustrated as an increase in radius of the circular distribution of power shown Figure 4.6b. Also, the increasing radius of the distribution of power in the Fourier domain entails a decrease in the peak value of the Fourier transform of the mask. The circumference of the power distribution is increased, which implies an increase of the area over which the power is distributed.

4.2.4 Multi-Frequency Unit Pulses

A better approximation of the Gaussian pulse is attained by using several circular frequencies, rather than just one because this allows the power of the data array to be distributed across multiple circular areas in the Fourier image. Spreading the power over a larger area reduces the likelihood of saturation of the holographic material.

One way to generate masks using multiple frequencies is to proceed as described previously, but to alter the radius of the circular pulses by a non-constant value during mask construction. When the rings making up the high frequency unit pulses have variable width, multiple frequencies are introduced

in the Fourier domain. Figure 4.7 shows a set of four multi-frequency unit pulse masks and their Fourier transforms.

Another method is summing several of the high frequency unit pulse masks. The Fourier transform of the sum of multiple high frequency unit pulse masks is equivalent to the sum of the Fourier transforms of each mask, due to linearity of the Fourier transform. By combining several high frequency unit pulse masks, a mask with a broad central distribution of power can be attained.

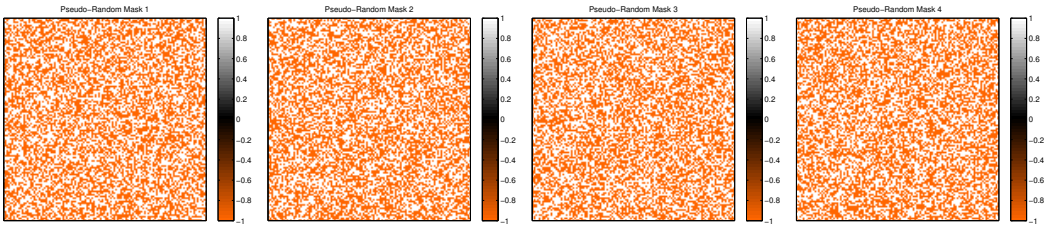
However, this summation results in arrays with values not equal to ± 1. In some cases several of the masks will have a $+1$ in the same location, which results in the summation of the masks containing integer values greater than 1. Likewise, integer values less than -1 appear. Additionally, values of 0 may appear if an even number of high frequency unit pulse masks are used for the summation.

To form an appropriate phase mask, the values higher than 1 are set equal to 1 and the values lower than -1 are set to -1. For the elements of the array that are equal to 0, a value of either $+1$ or -1 is assigned at random. The random values distribute the power of the data array away from the rings of spatial frequencies dictated by the high frequency pulses. This rough quantization tends to distort the Fourier transform of the resulting mask such that this method is no better than generating multi-frequency unit pulse masks with random variations in pulse radii.

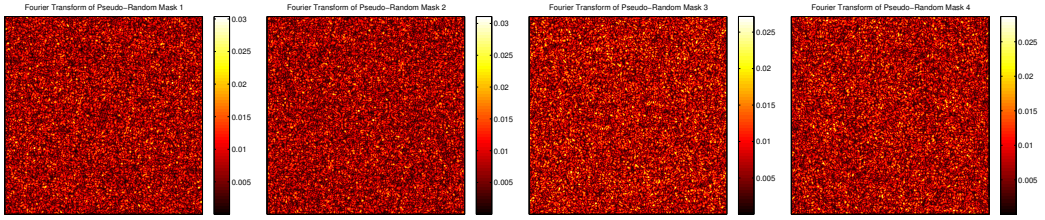
4.2.5 Pseudo-random Masks

Finally, phase masks can be generated by populating the mask array with pseudo-random values. The values are limited to ± 1, but the choice of which value to use is made according to a pseudo-random algorithm. With pseudo-random phase masks, generating the masks is a simple process, and the Fourier transform of a random mask has a relatively even intensity distribution.

An even intensity profile in the Fourier domain is advantageous for the reduction of peaks in a data array because of the convolution relation. Every point in the Fourier transform of the data array is distributed across the Fourier transform of the mask. In a data array with very strong periodic com-



(a) Pseudo-Random Masks



(b) Fourier Transforms of Pseudo-Random Masks

Figure 4.8: Pseudo-Random Masks and the corresponding Fourier Transforms

ponents, the Fourier transform of the masked array is mostly determined by the convolution of the periodic components and the Fourier transform of the mask. The non-periodic portions of the Fourier transform of the data array become more important as the data array becomes less periodic.

The Fourier transform of a pseudo-random mask has low peak values, but the precise location of peaks is also random. Thus, it is difficult to obtain any control over the Fourier properties of the pseudo-random masks. Although this contrasts with the approach taken for the prior mask designs, it does not pose a problem since the peak values of the Fourier transforms are small.

4.2.6 Simulations

The results of the evolutionary algorithm for various phase masks are detailed below. In each case, the proportion of OFF pixels is enforced to be at least 30%.

In these simulations, the data arrays are modeled as 120×120 arrays of HTM data. For the calculation of the Fourier transforms during the evolutionary algorithm used in the simulations of this section, the data arrays are oversampled by representing each data pixel as a block of 4×4 samples.

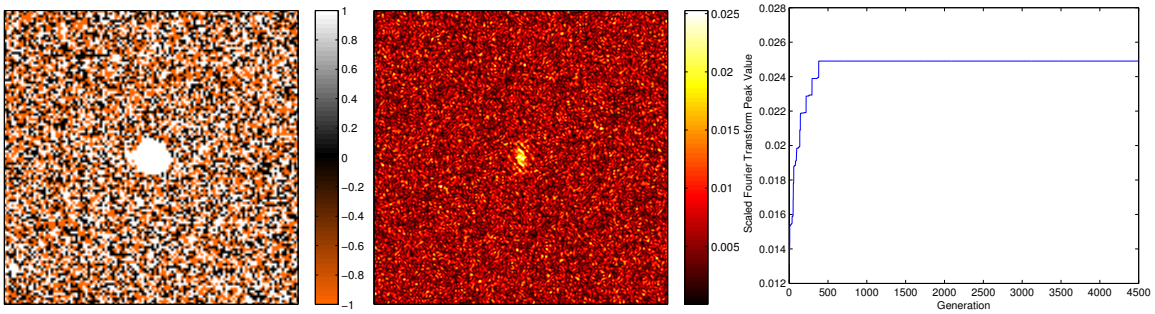
Additionally, the Fourier transforms are calculated by zero padding the oversampled data arrays into 1001×1001 arrays. An odd number was chosen for the Fourier transform arrays so that the array would contain an equal number of samples on both sides of the DC row and column. These sizes were also chosen to increase the amount of data obtained for the frequency domain representation of the data arrays while keeping processing times reasonable. This zero padding may introduce some inaccuracy due to the edge transition between the data array and the padded zeros. Windowing techniques exist to alleviate this problem, but none were utilized in this application because the resulting zero padded Fourier transformed were closely matched to Fourier transforms computed without zero padding in test cases.

Gaussian Pulses

The simulation result using the set of pulse masks shown in Figure 4.2 showed that only a small portion of the data values affected the masked data array. In one simulation, the evolved data array contained a circular portion of +1 pixels. In this case, the resulting Fourier transforms of the masked data arrays were roughly equivalent to the Fourier transforms of the masks themselves. In another, the circular shape of ON pixels was present, but the pixels varied in phase in each column. In this second test, the Fourier transforms of the masked data arrays were equal to the Fourier transforms of the masks, convolved with the Fourier transform of the plane wave representing the variation in phase.

In both cases, the peak value of the Fourier images of both evolved data arrays was close to the peak value of the Fourier transform of the mask labeled "Pulse 4" in Figure 4.2. The evolutionary algorithm in both cases found arrays with ON pixels for the central four pixels, which are the most significant pixels, in that those pixels are multiplied by the greatest factor. Once the smallest pulse was used the evolutionary algorithm could not find an array that had a higher peak FT value than Pulse 4 itself.

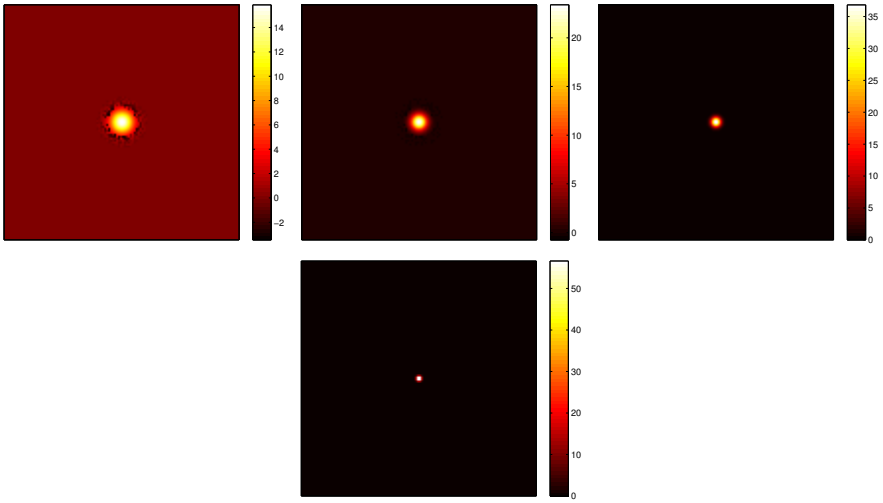
Figure 4.9 shows the first results of the evolutionary algorithm for a set of four Gaussian pulses. Here, the algorithm found that a patch of ON pixels in the center of the array provides the highest peak values in the Fourier



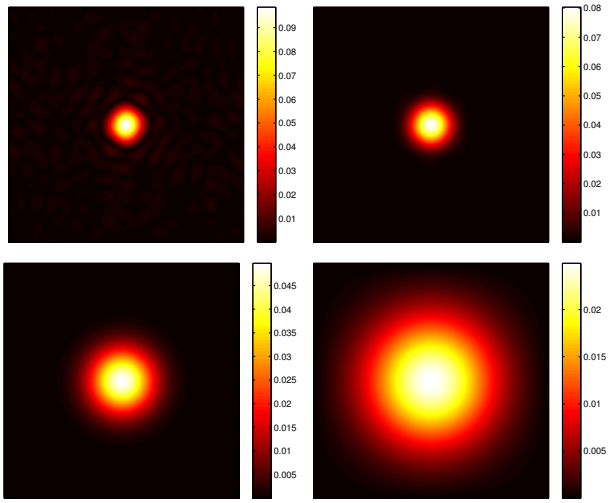
(a) Evolved Data Array

(b) Fourier Transform of Evolved Data Array

(c) Peak Value of Fourier Transform of Evolved Data Array vs. Generation



(d) Masked Versions of Evolved Data Array



(e) Fourier Transforms of Masked Versions of Evolved Data Array

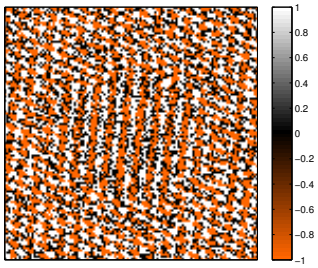
Figure 4.9: Results of one test of a set of four Gaussian Pulses (Set shown in Figure 4.2).

transform. The pixels outside the central region are random which indicates that those pixels did not have a strong impact on the overall result.

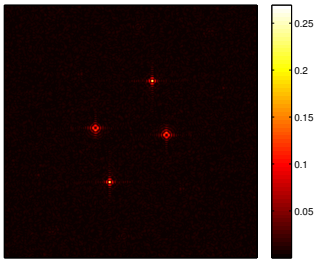
The evolution of the array is presented in the plot in Figure 4.9c. The algorithm records the peak value of the Fourier transform of the best choice for the surviving data array before the mutation step. The graph in Figure 4.9c is a plot of these recorded values each divided by N^2, where N is the number of pixels in each dimension of the data array.

The Gaussian pulse masks cause the evolution of the data array to proceed less smoothly than the evolution of other masks. This is due to the imbalance between the effect that individual pixels in the data array have on the Fourier transforms: changes to the central pixels have a much stronger effect than changes to the peripheral pixels, and the evolutionary algorithm only makes progress as it changes the central pixels. The plot shows that roughly after generation 400 the algorithm made no significant change to the data array. This is likely due to the central pixels being ON at this point, and any change to these pixels resulted in arrays with lower peaks in their Fourier images. Changes to pixels outside the central region would have a negligible affect on the overall peak value of the Fourier transform, so the algorithm was unable to progress by changing those pixel values.

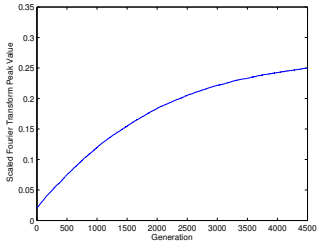
In principle, the results show that a mask with a broad pulse in the Fourier domain will greatly minimize the ability of any data array to saturate the recording material. In practice, however, the pulse masks are unusable. The masks require the data array to be modulated in amplitude in such a way that the central pixels have a much higher magnitude than the pixels toward the edge of the array. This is impractical for two reasons. Firstly, the actual process of varying the amplitude according to the Gaussian pulse mask is either overly complex or impossible. Secondly, and perhaps most importantly, the pixels which are amplitude modulated by a factor less than 1 will become nearly impossible to decode accurately. In the case of Pulse 4, all but the central 16 pixels will be reduced nearly to zero.



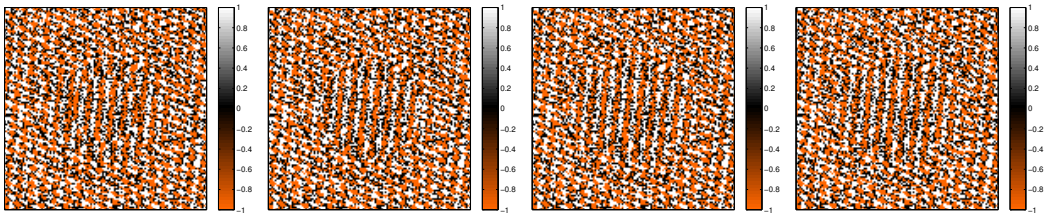
(a) Evolved Data Array



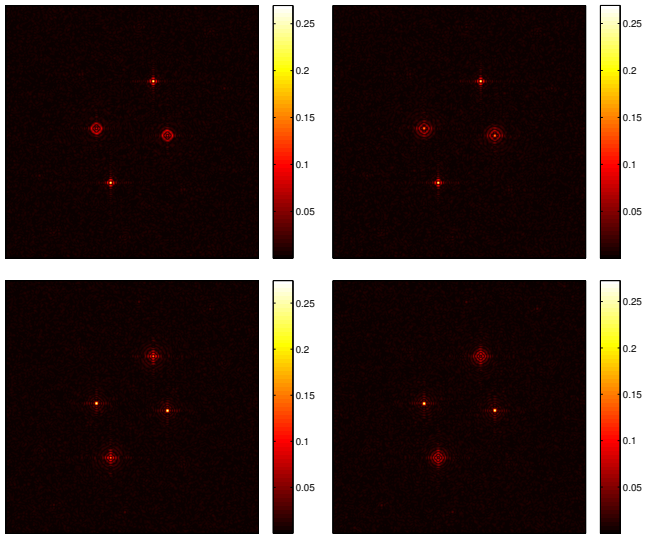
(b) Fourier Transform of Evolved Data Array



(c) Peak Value of Fourier Transform of Evolved Data Array vs. Generation



(d) Masked Versions of Evolved Data Array



(e) Fourier Transforms of Masked Versions of Evolved Data Array

Figure 4.10: Results of one test of a set of four Circular Unit Pulse Masks (Set shown in Figure 4.4).

Unit Pulses

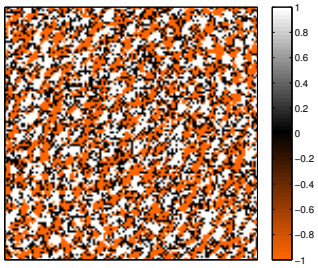
The evolutionary algorithm generated an interesting solution when the mask set consisted of the four unit pulse masks shown in Figure 4.4. The evolved array contained one dominant frequency component as well as a minor, modulated circular component. The main frequency component caused the Fourier transform of the array to have peaks in the unmasked case as well as the data array being masked with the first or second mask. The minor circular components caused the third and fourth masks to generate large peaks in the Fourier image.

The difference between each of the masks is that balance increases from mask 1 to mask 4. Increasing balance indicates a decreasing DC component. The masks with a more significant DC component are unable to reduce inherent periodicity in a data array. This is because the convolution of the Fourier transform of the data array and the Fourier transform of the mask are dominated by the DC component of the mask replicating the inherent frequency component of the data array.

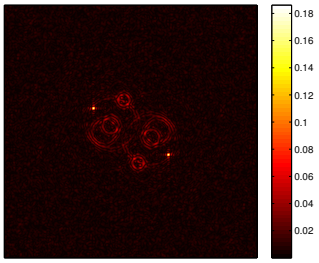
The results of this simulation are shown in Figure 4.10. The unit pulse masks fared much worse than the Gaussian pulse masks, with peaks roughly 10 times larger than those generated by the algorithm when Gaussian pulses were used. This is mainly due to the set of masks lacking one member that virtually blocked out the data array, as was the case with the fourth mask in the Gaussian pulse set. However, the unit pulse masks are practical for use in the holographic storage system because they do not require pixel by pixel amplification or attenuation because all elements in the mask arrays are either ± 1.

High Frequency Unit Pulses

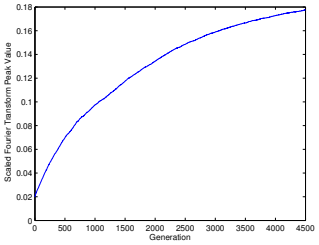
The results from the evolutionary algorithm for the set of four high frequency unit pulses, shown in Figure 4.6, were somewhat similar to the results for the unit pulses. It is clearer in this case that the data array has been evolved to contain within it some similarity to the masks. The Fourier transform of the data array, shown in Figure 4.11b, indicates a strong periodic component in



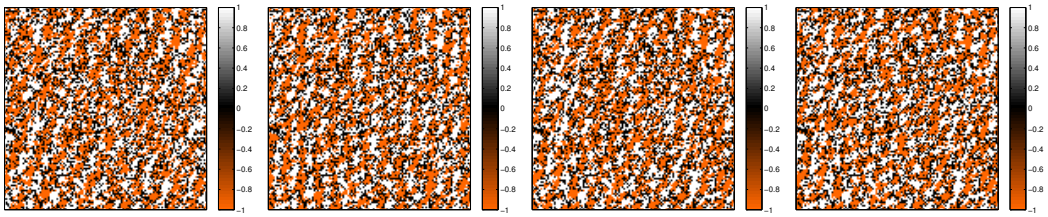
(a) Evolved Data Array



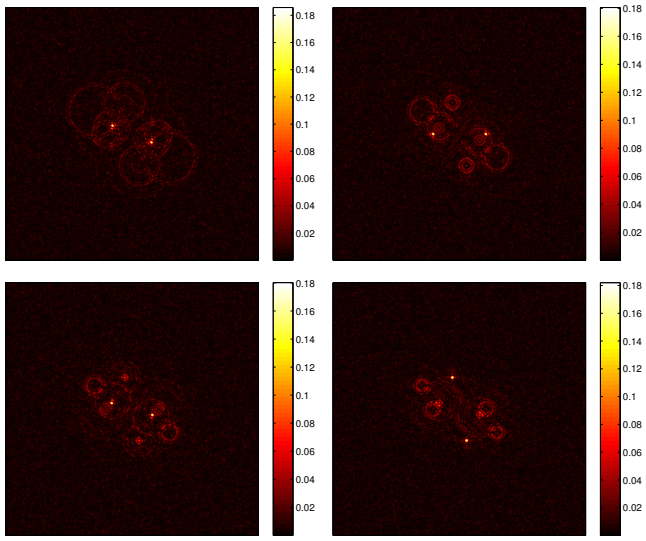
(b) Fourier Transform of Evolved Data Array



(c) Peak Value of Fourier Transform of Evolved Data Array vs. Generation



(d) Masked Versions of Evolved Data Array



(e) Fourier Transforms of Masked Versions of Evolved Data Array

Figure 4.11: Results of one test of a set of four High Frequency Unit Pulse Masks (Set shown in Figure 4.6).

the data array, as well as replications of the Fourier transforms of each mask at some spatial frequency.

The Fourier transforms of the masks are visible as pairs of circles that are equidistant from the origin. These portions are representative of the convolution of each mask with some spatial frequency. This implies that in the data array some pixels create a pattern that is partially equivalent to each mask multiplied by some periodic data pattern. When the data array is multiplied by any of the masks, the intrinsic periodic component in the data array is reduced, but the portion of data equivalent to the masked version of a plane wave becomes "unmasked" and results in a strong periodic component in the Fourier transform of the newly masked array.

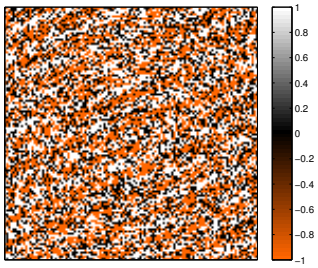
Multi-Frequency Unit Pulses

The results of one evolutionary simulation of the set of four multi-frequency unit pulse masks, shown in Figure 4.7, are displayed in Figure 4.12. The evolved data array contains periodic components that produce large peaks in every masked case. The components that are periodic in each masked case are present in the unmasked data array as masked versions of the periodic components.

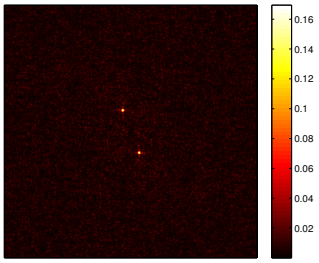
This is much less noticeable with this set of masks than it was for the unit pulse and high-frequency pulse masks. With the two prior types of mask, the Fourier transform of the data array contained features relating to the convolution of each mask and some periodic data pattern. Since the Fourier transforms of the multi-frequency pulse masks have lower peaks and are more uniform than the other types, these convolutional features are more difficult to discern visually.

Pseudo-Random Phase Masks

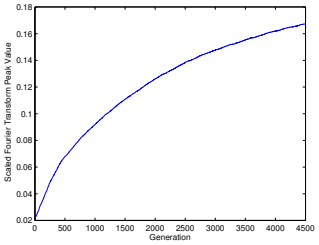
Among the various types of phase mask designs that have been reviewed in this section, pseudo-random phase masks have the most uniform Fourier transforms. That is to say, the Fourier images of pseudo-random masks contain the lowest peak values and have these smaller peaks in more locations. Following



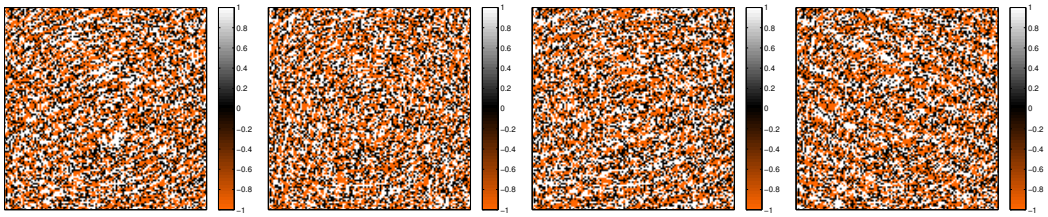
(a) Evolved Data Array



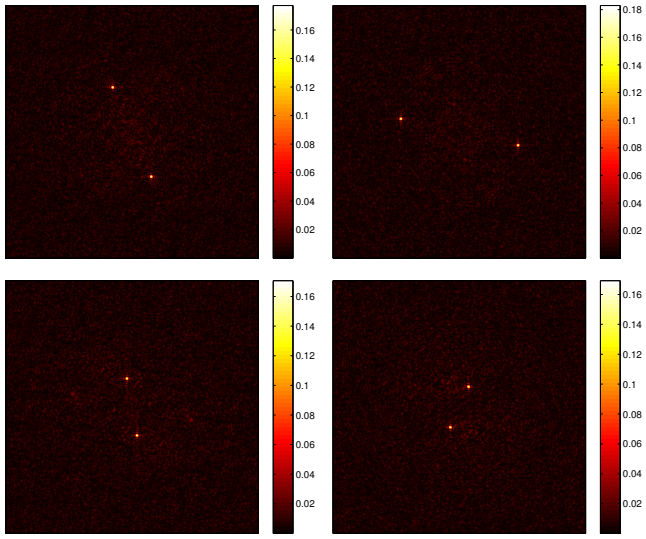
(b) Fourier Transform of Evolved Data Array



(c) Peak Value of Fourier Transform of Evolved Data Array vs. Generation

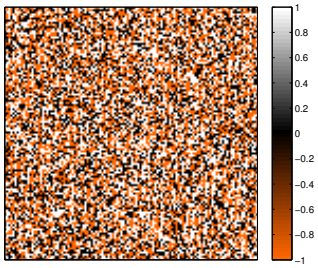


(d) Masked Versions of Evolved Data Array

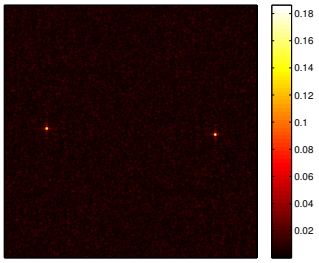


(e) Fourier Transforms of Masked Versions of Evolved Data Array

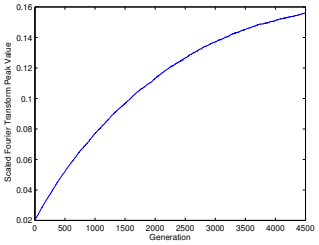
Figure 4.12: Results of one test of a set of four Multi-Frequency Unit Pulse Masks (Set shown in Figure 4.7).



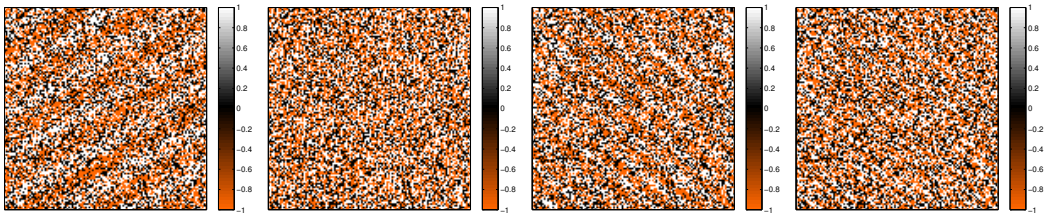
(a) Evolved Data Array



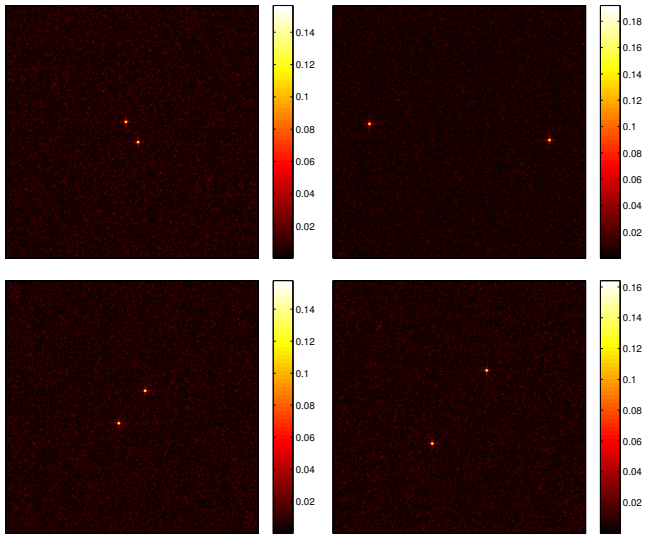
(b) Fourier Transform of Evolved Data Array



(c) Peak Value of Fourier Transform of Evolved Data Array vs. Generation



(d) Masked Versions of Evolved Data Array



(e) Fourier Transforms of Masked Versions of Evolved Data Array

Figure 4.13: Results of one test of a set of four random phase masks (Set shown in Figure 4.8).

the results presented thus far, which indicate that masks with more uniform Fourier transforms produce Fourier images with lower peak values, it is expected that pseudo-random phase masks will work better than masks with some inherent periodic components.

The results of an evolutionary algorithm simulation using a set of four pseudo-random phase masks are shown in Figure 4.13. The first masked array contains the lowest peak Fourier-domain values of any array in this section, with the exception of the arrays that were masked using the impractical Gaussian pulse masks. From these results it is concluded that pseudo-random masks outperform all of the practical designed masks presented in this section.

The Fourier-domain peaks in a data array are diminished when the array is masked with a pseudo-random phase mask because the convolution of the Fourier domain representations of the data array and mask array ensures that the peaks will be distributed relatively uniformly in the Fourier domain of the masked array. Furthermore, since the Fourier-domain peaks in each pseudo-random phase mask occur in sporadic locations a data array is less likely to have large peaks in every masked array. This is evident in the results of the evolutionary algorithm simulations since the set of pseudo-random masks resulted in the lowest peak values in evolved arrays.

Because pseudo-random masks outperformed designed masks in these simulations, the remaining sections consider only pseudo-random phase masks.

4.3 Number of Phase Masks

Since the random phase masks are the best performing design under review, the next step is to determine the effect of adding multiple masks to the system. This is done by simply expanding the number of masks in the trial sample and running the evolutionary algorithm. Results from tests of various numbers of phase masks are shown in Figure 4.14.

In each evolutionary algorithm test, the number of child arrays per generation was kept at 50, and the algorithm was executed for 4500 generations. The data arrays were chosen to contain 128×128 pixels, and the Fourier transforms are calculated without over-sampling to keep the simulation times

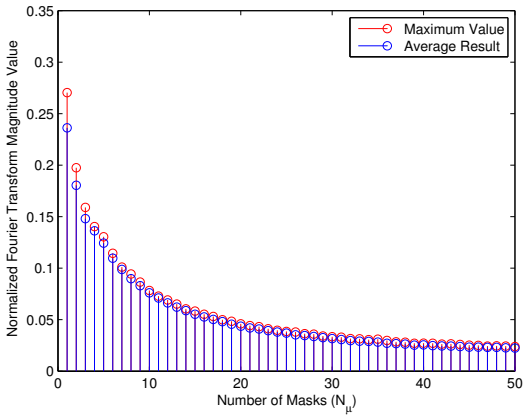


Figure 4.14: Results from several evolutionary tests with various numbers of pseudo-random phase masks.

relatively short as the number of masks is increased. Ten simulations were run for each value of N_μ with a different set of pseudo-random masks used for each simulation.

Since the actual values in the Fourier transforms are dependent on the number of elements in $f[m, n]$ the peak values of the Fourier transform arrays were normalized by dividing by 128^2 in Figure 4.14. An array consisting solely of ON pixels with uniform phase has a normalized value of 1 at DC in the Fourier array.

Since ten simulations were used per each value of N_μ, the Fourier domain peak values of the best-choice masked arrays for the evolved data arrays of each simulation are averaged. Additionally, the overall maximum Fourier domain peak value at any point during all ten simulations is plotted in Figure 4.14.

The results of these simulations show a clear decline in the Fourier-domain peak value as the number of masks is increased. Moreover, the majority of this decrease is achieved with relatively few masks. The average end value for $N_\mu = 6$ is 50% less than this value for $N_\mu = 1$. Likewise, the maximum value with five masks is 50% of the maximum value found in simulations using one mask. This indicates that a practical system will not need to employ more than roughly 10 masks to gain a significant portion of the benefit available through selective phase masking.

4.4 Interleavers

Interleavers are devices that reorder the elements in data arrays. The use of interleavers as a means to reduce peaks in the Fourier domain is possible because shuffling the data array can disrupt periodicity which leads to large peaks. Additionally, phase masks are unable to alter the pattern of ON versus OFF pixels in a data array. Interleavers can alter this pattern, so their potential utility was investigated to determine the effectiveness of their use in diminishing Fourier peaks.

Unlike phase masks, however, interleavers are incapable of actually altering the value of any pixel. Because of this, a system using only interleavers will have not decrease the DC value of an array that is unbalanced. Interleavers are therefore used in conjunction with a set of phase masks in the simulations presented in this section. When combining the two approaches, the number of options available to the system is determined by the expression:

$$N_o = (N_\mu + 1)(N_i + 1) + N_\mu N_i$$

where N_o is the number of options, N_μ is the number of masks, and N_i is the number of interleavers in use in the system. The different options are: the data array as it is, the N_μ masked arrays, the N_i interleaved arrays, the $N_\mu N_i$ masked then interleaved arrays, and finally the $N_i N_\mu$ interleaved then masked arrays. Masking an interleaved array and interleaving a masked array have different results, so the order of the operations is important and provides additional alternatives for representing a data array. To decode an interleaved and masked array some pixels must be reserved to indicate to the decoder which option was used to encode the array. At a minimum 3^{N_o} pixels are required to store this information, but encoding these reserved pixels with an error control code would likely be necessary to ensure that the array is correctly decoded. As in the phase masking case, decoding an array with an incorrect mask or interleaver will result in an error rate roughly equal to decoding each pixel at random.

The peak reduction capabilities of interleavers were tested by running the evolutionary algorithm on systems employing some combination of masks and

interleavers. The results obtained from these tests were compared to similar tests run for systems containing only phase masks. The tests consisted of data arrays of size 120×120 with no oversampling. The arrays are zero padded to size 1001×1001 for calculation of the FFT. The evolutionary algorithm used 35 children per generation and ran for 3000 generations. The normalization factor used in Figure 4.15 is 120^2, because the data arrays contain 120^2 data samples prior to computing the peak values of the Fourier transforms.

Compared to the tests in the prior section, these evolutionary simulations have narrower breadth as a result of having only 35 children per generation as opposed to 50. The interleaver tests also have a shallower depth by searching for only 3000 generations rather than 4500. A narrower breadth decreases the scope over which the algorithm can search in a single generation. Likewise, the shallower depth decreases the level of refinement of the final data array. As a result of these two changes, the peak values of the Fourier transforms of the evolved data arrays, which are plotted in Figure 4.15, are lower than those reported in Section 4.2.6. In this section, the absolute value of the peaks is not as important as the comparative results between masks only and masks with interleavers.

No tests were run for systems consisting solely of interleavers. As mentioned in Section 4.4, interleavers are incapable of reducing a DC value. For this reason, their utility as the only means of peak mitigation is inadequate.

Initially, it was hypothesized that the interleavers would provide results equivalent to those provided by phase masks. The results of the evolutionary algorithm tests, however, indicate that interleaver and mask combinations are inferior to systems that use phase masks exclusively, for the same number of total options.

The trendlines plotted in Figure 4.15 show that the systems which employed interleavers did not reduce peaks in the Fourier image as well as those systems which contained only phase masks. When keeping the number of data array masking options constant, the systems with interleavers are uniformly higher than those without interleavers. The trendline for the interleaved systems is $y = 0.3369x^{-0.583}$. The trendline for the systems containing only phase masks is $y = 0.2795x^{-0.583}$. The exponent of x is the same value for each

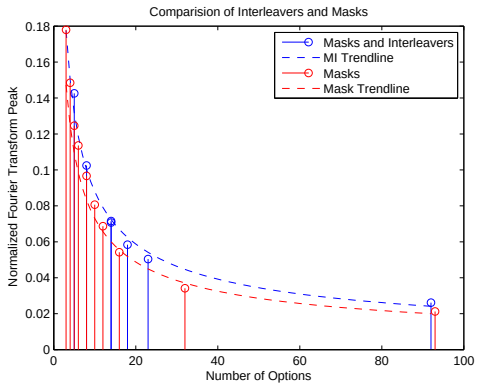


Figure 4.15: Results from tests of Masking systems with and without interleavers.

trendline, but the constant scaling term is greater for the system containing interleavers.

From these results it is concluded that adding an interleaver to a system that employs phase masking is less effective than simply adding an additional mask. If the interleaving operation can be performed faster or more efficiently than masking, interleavers could be used to provide a similar, though less effective, benefit to masking. However, a comparison between the hardware requirements of the two systems is left for future work. Based on these conclusions, only phase masks are considered for the remainder of this thesis.

4.5 Array Size

The results shown in Section 4.3 were demonstrated using data arrays of 128×128 pixels. In practice, the arrays are likely to be composed of a larger number of pixels, therefore the effect of using an array of some other number of pixels must be determined. The work in this section has been conducted to determine how the proportion of arrays that produce large Fourier domain peaks changes with changing array dimensions, from which conclusions can be draw regarding the number of masks required to effectively limit spectral peaks.

4.5.1 Definition of Terms Regarding Array Size

It has been shown in Sections 2.6 and 2.7 that the size of the data array is an important factor in the evaluation of the Fourier transform. Size can refer to the number of pixels in each dimension of the data array as well as the physical measurements of the SLM. For clarity, in this section, the number of pixels in each dimension will be referred to as N with the term "size" used in reference to the physical dimensions of the SLM.

Altering either N or the physical size of the data array can impact the optical power. As N increases, if the total optical power remains constant throughout the entire array, the power per pixel decreases. Alternatively, the power per pixel can be kept constant, in which case the overall power per array increases. In the following analysis, the total maximum possible power in the array is kept constant. This means that an all-ON array of either $+1$ or -1 valued pixels will have the same total power regardless of N. This choice isolates the effect of the change of N by keeping the total optical power in the system constant.

4.5.2 Problem Formulation

With HTM, as N increases, the total number of possible arrays increases as 3^{N^2}. Determining the proportion of all arrays that saturate the medium in such a search space by direct examination of all arrays is intractable for even modest values of N. Instead, work has been carried out to obtain an estimate of the proportion of arrays of size N that produce large peaks as N increases.

With the power kept constant with increasing N, the peak magnitude values in the Fourier domain can range from 0 to N^2 in the worst-case. The magnitude values can thereby be normalized and recorded as simply a proportion of N^2 from 0 to 1.

4.5.3 Fourier Coefficient Analysis

The number of calculations in estimating the likelihood of occurrence of spectral peaks can be reduced by first examining the two-dimensional DFT defini-

$$
\begin{aligned}
C_0 = &\quad f[0,0] + f[1,4] + f[2,3] + f[3,2] + f[4,1] \\
C_1 = &\quad f[0,1] + f[1,0] + f[2,4] + f[3,3] + f[4,2] \\
C_2 = &\quad f[0,2] + f[1,1] + f[2,0] + f[3,4] + f[4,3] \\
C_3 = &\quad f[0,3] + f[1,2] + f[2,1] + f[3,0] + f[4,4] \\
C_4 = &\quad f[0,4] + f[1,3] + f[2,2] + f[3,1] + f[4,0]
\end{aligned}
$$

Table 4.1: Description of each C_r term in $F[1,1]$ for the $N = 5$ case.

tion:

$$F\left[k,l\right] = \sum_{m}\sum_{n} f\left[m,n\right] e^{\frac{-2\pi i}{N}(mk+nl)}$$

This summation results in $F[k,l]$ being composed of N exponential terms of varying phase. In most cases, the amplitudes of these phase terms are determined by the sum of a set of N elements from $f[m,n]$. In the case of HTM data, the $f[m,n]$ array values can take on only three possible values: -1, 0, or 1. Because of this limited range, the sums of the $f[m,n]$ elements can be replaced by coefficients that also have a relatively small range of possible values, from $-N$ to N. Writing the DFT definition with these coefficients gives:

$$F\left[k,l\right] = C_0 + C_1 e^{-1\frac{2\pi i}{N}} + C_2 e^{-2\frac{2\pi i}{N}} + \cdots + C_{N-1} e^{-(N-1)\frac{2\pi i}{N}}$$

where each C_r term is determined by the sum of all $f[m,n]$ elements such that $mk + nl = r$ modulo N.

As an example, the coefficients for $F[1,1]$ when $N = 5$ are detailed in Table 4.1. If this table were calculated for a different $F[k,l]$ value, the specific $f[m,n]$ elements that compose each coefficient would change. However, each of the five coefficients would still be the sum of five $f[m,n]$ elements, with the lone exception of $F[0,0]$ in which all 25 $f[m,n]$ terms are associated with C_0.

There are cases in which each coefficient is not associated with N terms from the $f[m,n]$ array. As mentioned previously, the $F[0,0]$ value for all N is equivalent to the sum of all $f[m,n]$ values. In this case the coefficient C_0 is associated with every element in $f[m,n]$ while the coefficients C_1 through C_{N-1} are not associated with any $f[m,n]$ elements.

In addition to $F[0,0]$ for all N, there are $F[k,l]$ values with non-prime N

values that have an uneven distribution of $f[m,n]$ elements to the coefficients. For instance, $F[2,2]$ for $N = 6$ has twelve $f[m,n]$ elements associated with each of C_0, C_2, and C_4, and zero elements associated with the remaining coefficients C_1, C_3, and C_5. In this case, the three coefficients that are associated with twelve $f[m,n]$ elements range from -12 to 12 rather than -6 to 6. This discrepancy causes the simplifications outlined below not to hold for $F[k,l]$ values with an uneven distribution of $f[m,n]$ elements.

Uneven coefficient distributions are not present when N is prime with the exception of the DC value, and they are in the minority when N is composite. These cases are not considered in the computations presented in the remainder of this section. The computations detailed henceforth compute the distribution of magnitude values for all evenly distributed $F[k,l]$ points for a given N as described in the following.

For small N, the magnitude of $F[k,l]$ reduces to:

$$|F[k,l]| = \sqrt{\sum_{r=0}^{N-1}\sum_{s=0}^{N-1} C_r C_s \cos\left(\phi_{|r-s|}\right)}$$

with:

$$\phi_{|r-s|} = |r-s|\,\frac{2\pi}{N}$$

This equation was manually derived by determining the expression for the magnitude value when $N = 3, 4, 5,$ and 6. It is hypothesized that this expression extends to larger values of N also.

Introducing the coefficients allows the $F[k,l]$ values to be calculated using only the set of all coefficients rather than the set of all $f[m,n]$ arrays. This set of coefficients has N elements, each with $2N+1$ possible values. This results in $(2N+1)^N$ computations rather than 3^{N^2}, which is a significant reduction in complexity even though it still has greater than factorial growth with N.

Since several different arrays produce identical sets of coefficients, it is important to know how many arrays produce a given set of coefficients. The distribution of the coefficient values can be described by a simple recursive relation.

In the $N = 1$ case, the single coefficient can have a value of either -1, 0 or

v:	-4	-3	-2	-1	0	1	2	3	4
$N = 0$:					1				
$N = 1$:				1	1	1			
$N = 2$:			1	2	3	2	1		
$N = 3$:		1	3	6	7	6	3	1	
$N = 4$:	1	4	10	16	19	16	10	4	1

Table 4.2: Listing of $c_N(v)$ values for N equal to zero through four.

1. There are three possible arrays, and each coefficient value is produced by exactly one array.

In the $N = 2$ case, the two coefficients can have values of either -2, -1, 0, 1, or 2. For this case, there are 81 possible arrays, and each coefficient value is not equally represented. For each coefficient, one subset of N $f[m, n]$ elements produces the ± 2 values, two subsets produce the ± 1 values, and three subsets produce the 0 value. Note that the coefficients are determined by wholly separate sub-sets of N pixels in the array. This gives a total array count of $3^2 \cdot 3^2 = 3^{2^2} = 81$, or more generally $(3^N)^N = 3^{N^2}$, which equals the total number of arrays.

As N increases, the distribution of the coefficients follows a pattern similar to Pascal's Triangle. The coefficient distribution, $c_N(v)$, is described by the recursive relation:

$$c_N(v) = c_{N-1}(v) + c_{N-1}(v-1) + c_{N-1}(v+1) \tag{4.1}$$

where v is a value of the coefficient and $c_N(v)$ is the number of N pixel subsets that generate the coefficient value v. Table 4.2 depicts the $c_N(v)$ values for N ranging from 0 to 4. The recursive definition can be demonstrated by inspection of each $c_N(v)$ value in the table. Each element is the sum of the elements in the previous row directly above as well as one column to the left and one column to the right of its position.

The values shown in Table 4.2 and the definition given in Equation 4.1 are useful for determining the range of values the coefficients can take in logical units. However, v increases with N which means that the total power within the array also increases with N. This is a direct contradiction of the

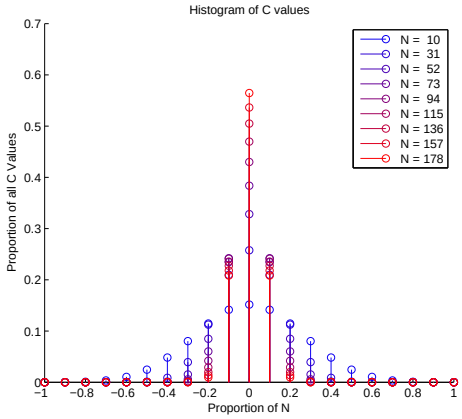


Figure 4.16: Histogram of coefficient distribution as N increases.

assumption stated in Section 4.5.1 that the power contained in the array is kept constant as N increases. To compensate for this, the v values need to be scaled so that the maximum value of v does not increase with N. This can be accomplished by dividing v by N so that v ranges from -1 to 1.

Figure 4.16 shows a histogram of coefficient values as N increases, with scaling so that v ranges from -1 to 1. Additionally, the c_N values have been scaled by 3^N. This is done so that each scaled c_N value is the proportion of all coefficient values that have value v. A histogram is more informative than a direct plot, because with the scaling each point in the plot approaches zero as N increases. This tendency can be observed in Table 4.2 by examining the $c_N(0)$ values. The raw number increases with N, however the number of all possible arrays increases at a faster rate. In the $N = 2$ case, 3 out of 9 pixel subsets produce the value 0. In the $N = 4$ case, this proportion has reduced to 19 out of 81.

The histogram indicates that as N increases the proportion of all c_N values shifts heavily toward 0. With each coefficient value tending toward zero, it follows that the magnitude values of the DFT should also tend to zero due to the fact that the coefficients are the magnitude values for each complex exponential term in the DFT. Therefore, it can be expected that as N increases, the proportion of arrays with low-level peaks increases, while the proportion of arrays with large peak values decreases.

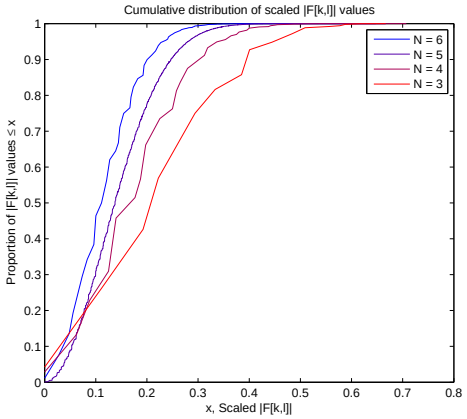


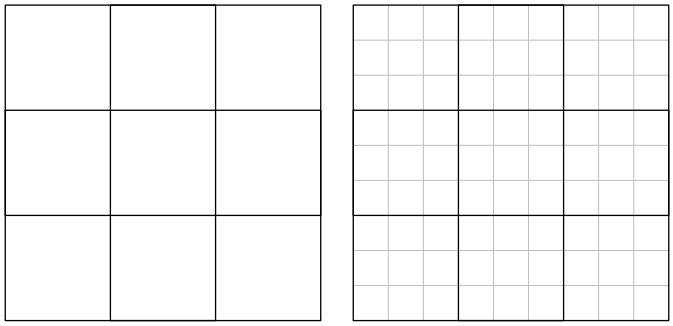
Figure 4.17: Plot of the cumulative proportion of arrays with scaled magnitude values less than or equal to the value on the x-axis for varying N.

4.5.4 Computed Results

Numerical calculations of $|F[k,l]|$ were computed for several values of N. The magnitude values were computed by iteration through all sets of coefficient values for a given N. For each set, the number of arrays producing that set was computed combinatorially. Several sets of coefficients produced identical magnitude values. To mitigate the effect of finite precision, if a calculated magnitude value was within $\pm 5 \cdot 10^{-7}$ of a previously calculated value, the two values were considered identical.

The results of four calculations are plotted in Figure 4.17. In these plots, the magnitude values are scaled by $1/N^2$ to place the range of the magnitude values for each N between 0 and 1. Additionally, the number of arrays at each magnitude value is scaled by $1/3^{N^2}$, the reciprocal of the total number of arrays. This converts the total number of arrays value to the proportion of all arrays for a given magnitude value. Finally, the plot shown is the cumulative sum of the proportion of all arrays. This converts the scattering of points into a cumulative distribution of the magnitude values.

From these results, it is clear that as N increases, arrays with low peak magnitude values form a larger proportion of all arrays. This is in agreement with the results presented in Section 4.5.3 of the distribution of coefficient values as N increases.



(a) $N = 3$ Data Array (b) $N = 9$ Data Array

Figure 4.18: Diagrams of two arrays with different N values.

Conversely, as N increases, the number of arrays that produce large peaks relative to the number of all arrays decreases. An intuitive explanation of this result is as follows.

An $N = 3$ array is shown in Figure 4.18a. Next to that array is an $N = 9$ array in Figure 4.18b. Since the total power in the array is kept constant with increasing N, the $N = 3$ array can be viewed as an $N = 9$ array in which each of the pixels in the 3×3 array are composed of a set of 3×3 sub-pixels, where all sub-pixels in each 3×3 set have the same value. Because of this, the sub-pixel arrays are forced to be either of the worst-case 3×3 arrays (the all 1 and all -1 arrays), or the best-case array (the all-OFF array). Here, the worst and best cases are in reference to the peak magnitude values that arise in the Fourier transforms of the 3×3 pixel patterns.

The two constant-phase, all-ON arrays are the worst-case in the sense that they have the highest possible Fourier domain magnitude values. In each case, the DC component is equivalent to the total power in the array, which is the greatest possible magnitude value, since it is impossible for any portion of the Fourier domain representation to have more power than is contained in the data array. Likewise, the all-OFF array is the best case in that it has peak values of 0 in the Fourier domain due to no optical power being transmitted through the SLM. So, of all sub-arrays available for use as pixel values in the larger $N = 3$ array, ⅔ are worst-case sub-arrays and ⅓ are best-case sub-arrays.

In the $N = 9$ case, however, there is no requirement for any of the pixels to have common values. Keeping the grouping of pixels from the $N = 3$ case, the 3×3 sub-pixel arrays are no longer forced to be either the best-case or the worst case. The smaller 3×3 sub-arrays will have the worst-case value in only $2/3^9$ cases. The best-case sub-array constitutes only one of the 3^9 sub-arrays, but the remaining sub-arrays are more closely distributed to the best-case sub-array. This causes the proportion of all arrays that results in large Fourier-domain peaks to be lower with $N = 9$ than it is with $N = 3$.

4.5.5 Simulations

Simulations using the evolutionary algorithm were run with arrays of 32×32, 64×64, 128×128, and 256×256 pixels. For each array size, simulations were run for systems using 4, 8, 12, 16, and 20 masks. The results of these simulations are plotted in Figure 4.19. The Fourier-domain peak values are normalized by dividing the Fourier transform values by the number of pixels in the array. With this normalization, the total power of the array is constant for the various array sizes since an all-ON array has a Fourier-domain peak value of 1 at DC for all array sizes.

For each array size, the Fourier-domain peaks of evolved arrays decreases as the number of masks increases, similar to the results shown in Section 4.3. Furthermore, it is clear from these results that the evolutionary algorithm produced arrays with greater normalized Fourier domain peaks for the smaller sized arrays.

4.6 Conclusion

Selectively using one of a set of phase masks at the coding step has been shown to provide a significant reduction in the severity of Fourier domain peaks in evolved data arrays. Multiple mask designs were tested, and pseudo-random phase masks have been shown to outperform any of the designed masks that emulated the Fourier-domain characteristics of a Gaussian pulse. This is due to the Fourier transforms of pseudo-random masks having low peak values.

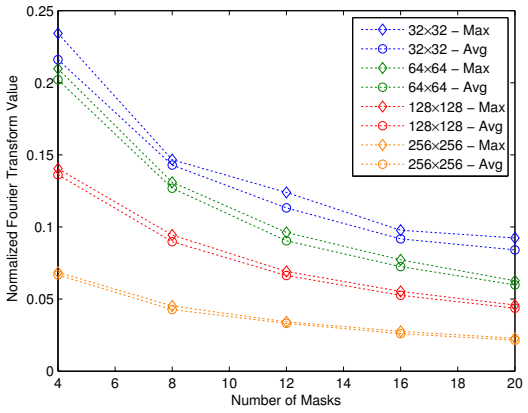


Figure 4.19: Global maximum Fourier Transform peak value per simulation and average resulting Fourier transform peak value for various number of masks using 32×32, 64×64, 128×128, and 256×256 sized arrays.

Selective phase masking provides increased peak reduction ability as more masks are used. A diminishing marginal utility of each additional mask ensures that most of the benefits of this technique can be realized without using an impractically large set of masks. In the simulations presented here, with 128×128 sized data arrays, 10 masks provide more than 50% of the peak reduction capability provided by 50 masks.

Finally, analysis suggests that large Fourier-domain peaks are less likely to occur as the dimensions of the data array are increased. This indicates that arrays larger than the ones used in the simulations presented in this chapter will not require a larger number of masks to maintain the benefits of selective phase masking. These expectations have been confirmed by simulations.

Chapter 5

Example

Having demonstrated the manner in which sparse guided scrambling and selective phase masking function, this chapter includes a brief example to demonstrate the entire coding process from binary source data to encoded array.

5.1 Example

For the example, the final HTM array size is 120×120 pixels, so that a maximum of 14400 HTM symbols can be recorded. The 19:12 binary-to-ternary code is used for the initial binary-to-ternary conversion, and this is followed with guided scrambling using the degree-2 scrambling polynomial $d(x) = x^2 + x + 2$. Following the discussion of Chapter 3 that considers the case when $A = D$ and $S = 3^D - 1$, this degree-2 encoding maps 6 source HTM symbols to 8 encoded HTM symbols. Finally, the sparse array is masked which requires some pixels to be reserved to store the index of the mask.

It is straightforward to determine the number of data symbols to encode at each stage by working backwards. The final array will contain 14400 HTM symbols. The number of HTM symbols before the sparsity encoding must be equal to $14400 \times {}^{6}\!/_{8} = 10800$, which equates to 900 blocks encoded by the 19:12 binary-to-sparsity code, or 17100 bits. However, the final HTM array must have some pixels reserved to store the mask index. Instead of encoding 900 blocks of binary data, 899 blocks are used in this example. This results

Table 5.1: Number of data symbols at each step in the encoding process.

Binary Data	17081 bits
Uncoded HTM	10788 HTM symbols
Encoded Sparse HTM	14384 HTM symbols
Masked HTM Array	14400 HTM symbols

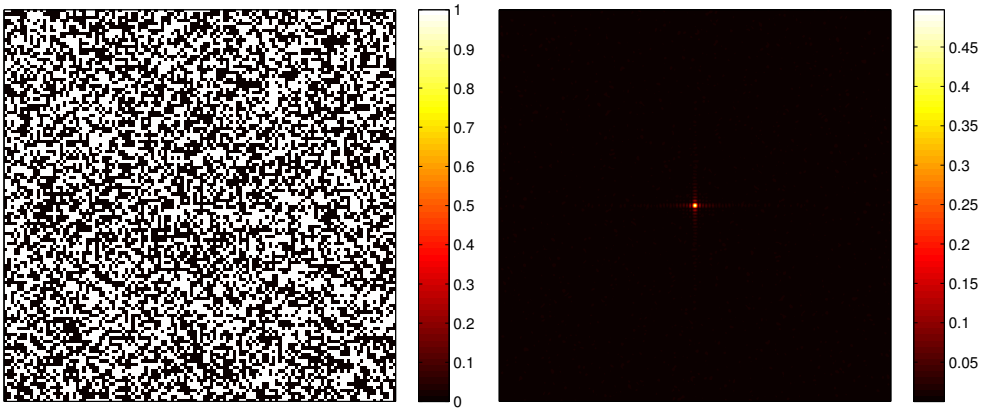
in 17081 bits being represented by 14400 sparse HTM symbols (including the mask index) for an overall code rate of 1.1862 bits per pixel in the encoded array. The number of symbols at each step is outlined in Table 5.1.

Since the final array has 14384 data-bearing HTM symbols, 16 pixels are available to store the mask index. This allows an error control code to be used to minimize the risk of incorrectly decoding the index, assuming fewer than 3^{16} masks are used.

5.1.1 Binary Data

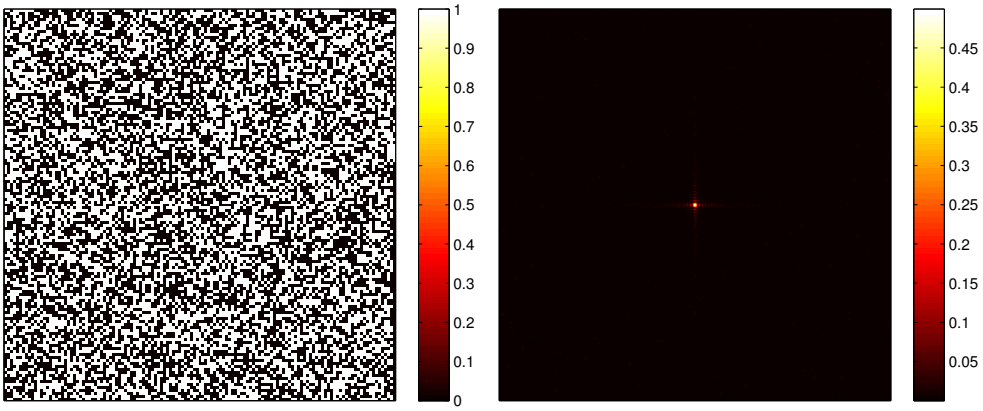
For this example, 17081 bits are generated to represent the data that will be stored. To demonstrate the effectiveness of the coding techniques outlined in this thesis, two different arrays containing this binary data are considered and are shown in Figure 5.1. The first maps the first 14400 of the 17081 bits to a 120×120 array in order to provide an unencoded reference with a constant sized array for each step in the encoding process. To better characterize the entire data set, a second array maps all 17081 bits to a 131×131 array with 80 zeros padded to fill out the array (note the vertical sequence of OFF-pixels in the lower right).

The Fourier transforms of both arrays are dominated by the DC peak. Since the arrays do not use a phase shift of any kind, the magnitude of the DC peak is proportional to the percentage of ON-pixels in the array. The 120×120 array has 49.74% ON-pixels, and the 131×131 array has 49.97% ON-pixels. The scaled DC peaks shown in Figure 5.1b and Figure 5.1d are equal to these values respectively, which indicates that both reference representations of the binary data are approximately equal in terms of the Fourier domain peak value. The zeros padded to the 131×131 array have minimal affect the



(a) 120×120 Binary Array

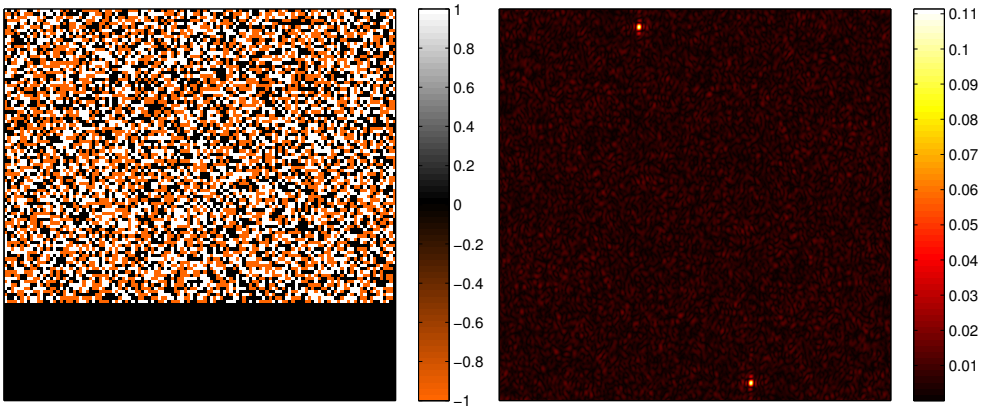
(b) Fourier transform of 120×120 Binary Array



(c) 131×131 Binary Array

(d) Fourier transform of 131×131 Binary Array

Figure 5.1: Two binary arrays and their Fourier Transforms



(a) Uncoded HTM Array

(b) Fourier transform of Uncoded HTM Array

Figure 5.2: HTM Array after the 19:12 binary-to-ternary code.

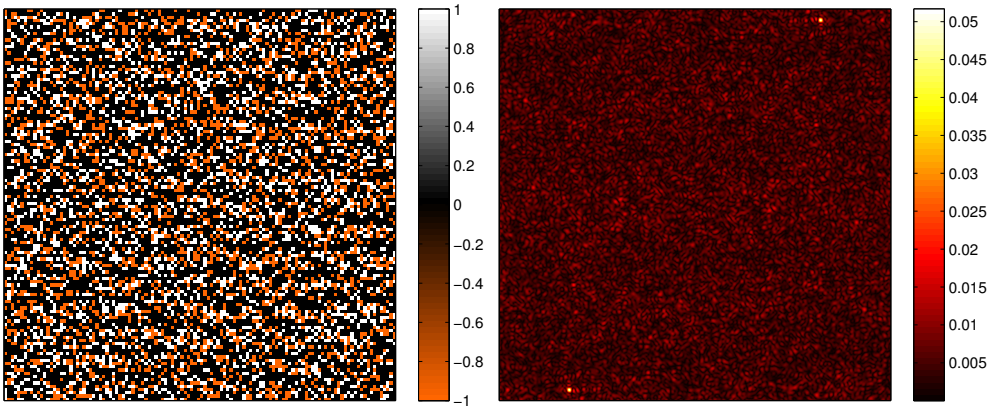
DC peak; instead, the data that was truncated to make the 120×120 array is slightly less sparse than the first 14400 bits.

5.1.2 Binary-to-Ternary Encoding

The binary data is next converted to HTM using the 19:12 code outlined in Chapter 3. This encoding produces a sequence of 10788 ternary symbols containing 66.9% ON-pixels, which indicates a sparsity value of 33.1%.

The uncoded HTM data is mapped directly to a 120×120 array, leaving 3612 pixels unused. In Figure 5.2, these unused pixels are OFF-pixels. Alternatively, the unused pixels could store an additional 3612 HTM symbols, in which case the uncoded array would carry more data than the final output array. However, the goal of the subsequent sparsity encoding is to reduce Fourier domain peaks, and it is instructive to compare the effect of sparsity coding against an uncoded HTM array. Since leaving the unused pixels OFF decreases the magnitude of Fourier domain peaks in general, the array in Figure 5.2 provides the most sparse presentation of the uncoded HTM data, and is therefore a best-case array for this uncoded data. The sparsity of this array, including the padded OFF-pixels, is 49.83%.

The uncoded HTM data is made sparse via guided scrambling with the



(a) Sparse HTM Array

(b) Fourier transform of Sparse HTM Array

Figure 5.3: HTM Array after guided scrambling using $d(x) = x^2 + x + 2$.

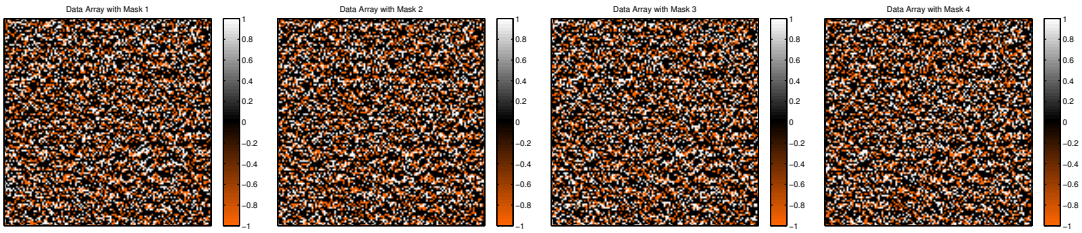
scrambling polynomial $d(x) = x^2 + x + 2$. As described in Chapter 3, this degree-2 primitive polynomial produces m-sequences of length 8, so it is well utilized when $S = 8$ and $A = D = 2$ which implies that $W = 6$. This forms a sparsity code with a rate of $R = 6/8$.

After sparse guided scrambling, the 10788 HTM symbols from the uncoded array produce 14384 sparse HTM symbols, which are mapped into the 120×120 array shown in Figure 5.3. The encoded data array has a sparsity of 60.36%, which is significantly greater than the sparsity of the uncoded array even when the padded zeros are counted.

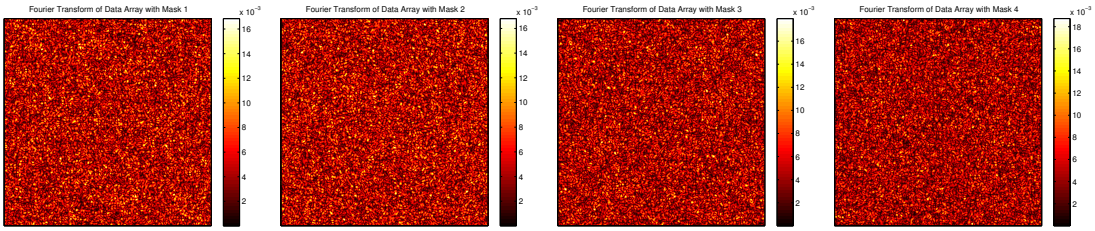
5.1.3 Selective Phase Masking

As is evident from Figure 5.3b, a sparse HTM array can still contain significant peaks in the Fourier domain. These peaks can be reduced using selective phase masking, as described in Chapter 4. The Fourier transforms for each masked variant of the sparse array are calculated, and the masked array with the lowest peak magnitude value is selected to be recorded.

The sparse array from Figure 5.3a is masked using a set of four pseudo-random phase masks. The resulting masked arrays are shown in Figure 5.4a. As shown in Figure 5.4b, the Fourier images of these masked arrays have



(a) Masked arrays



(b) Fourier transforms of masked arrays

Figure 5.4: Masked HTM Array

no significant peaks. The second masked array has the lowest Fourier-domain peak value of the four masked arrays, so the encoder chooses the second masked array for recording. In this example, the peak value of the Fourier image of the masked array is less than Fourier-domain peak value of the sparse array by a factor greater than 25.

As mentioned in Chapter 4, the encoder can alternatively be configured to select the first masked array with a peak value below a threshold value. If this threshold was set at 2% of the total optical power in the array, the encoder would instead calculate only the Fourier transform of the first masked array and select that masked array for recording because it has a peak value only slightly greater than 1.6% of the total optical power in the array. In this instance, the second masked array has Fourier-domain peak values that are lower than those of the first masked array by only 0.01% of the total optical power in the array. This difference can be greater in general, depending on the threshold value. This method results in fewer Fourier transform calculations, thereby increasing the speed of the encoding process, at the cost of slightly higher peak values in recorded holograms.

5.1.4 Summary of Example

In this example, 17081 bits are mapped into a 120×120 HTM array with no significant peaks in the Fourier domain. The binary data is first converted to ternary form using the high-rate 19:12 binary to ternary code described in Chapter 3. The binary form of the data will contain a large peak value at DC, and the ternary form of the data may contain a DC peak if there is an imbalance between the proportion of ON-pixels with zero phase shift and the ON-pixels with π phase shift. Additionally, large peak values can exist if there is some periodicity in the data array.

To reduce the occurrence and severity these Fourier-domain peaks, the HTM data is made sparse with guided scrambling as described in Chapter 3. The primitive degree-2 polynomial $d(x) = x^2 + x + 2$ is used as the scrambling polynomial. Two ternary symbols are used to augment the input ternary sequence of six symbols, resulting in encoded blocks of eight symbols. The resulting code has a rate of $^6/_8$, and increases the sparsity of encoded data from 33.1% to 60.36%.

Finally, the sparse HTM array is masked to remove the peaks that are present in the Fourier image of the array. Four masks are used to create a set of masked arrays through element-wise multiplication of each mask array and the sparse HTM data array. The Fourier transforms of each of these masked arrays are calculated, and the masked array with the lowest peak Fourier domain values is selected for recording.

Although this example exhibits incremental decreases in the Fourier domain peak values with each step of encoding, this is not necessarily the case for all data sequences. It is possible that an uncoded HTM array could have no significant peaks but the guided-scrambling-encoded sparse array does. This holds likewise for every step in the encoding process, with the exception of the phase masking step (which can never be detrimental because the unmasked array can be selected for recording). However, as has been demonstrated in the previous chapters, each step of the encoding process provides an improved likelihood of producing an array without significant Fourier domain peaks.

Chapter 6

Conclusion

This chapter concludes the thesis with a discussion of the main contributions presented herein as well as a brief overview of several potential avenues along which future work can be investigated.

6.1 Main Contributions

The main contributions presented in this thesis include the following:

A binary-to-ternary base conversion code that operates at a high rate and converts binary blocks to ternary blocks by way of algorithms that are easy to implement efficiently was developed. Since this is a block code, the rate is defined by the ratio of the number of bits in the input block to the number of symbols in the encoded block. Using block sizes of 19 and 12 respectively gives a rate of $1.58\bar{3}$ which is very close to the capacity of 1.58496.

Guided scrambling as a means to produce sparse ternary-valued data was evaluated analytically. Guided scrambling has been used to enforce other constraints on data, but the investigation of its use to increase the number of OFF-pixels in an HDS system is novel. Guided scrambling produces significantly more sparse data than an unencoded system, while being relatively simple to perform.

The worst-case for guided scrambling sparsity encoding was proven. It was shown that when using a number of augmenting symbols equal to the degree

of the primitive scrambling polynomial and using encoded blocks of length $S = 3^D - 1$, the worst- case sparsity is equal to ${}^{3^{D-1}}/_{3^D - 1}$, and that this worst case occurs in $2^{(S/2)}$ encoded words.

Simulations indicating that primitive polynomials yield the most sparse data in comparison to non-primitive polynomials were presented. This result prompted the analysis that focused on primitive scrambling polynomials.

Simulations indicating the tradeoff between increasing code rate and decreasing resultant sparsity with increasing scrambling polynomial degree were presented. As the degree of the scrambling polynomial increases the rate of the code increases drastically because the encoded block size is the sum of the number of augmenting symbols and the number of input symbols, and the number of augmenting symbols is equivalent to the degree of the scrambling polynomial while the total number of encoded symbols increases with the cube of the scrambling polynomial degree. Because of the drastically increasing code rate, the degree of control over the encoded distribution is reduced.

The process of selective phase masking was developed. The use of multiple phase masks at the coding step is introduced in this thesis. Simulations indicate that it is an effective method for reducing concentrations of optical power at all locations in the Fourier domain.

Detailed analysis was presented for various mask designs. Masks with structured designs were investigated, and while these designs did provide some degree of reduction of Fourier domain peak values, it was demonstrated that masks composed of pseudo-randomly distributed elements resulted in lower peak values in data arrays produced by the evolutionary algorithm.

The use of interleavers in addition to phase masks was investigated. Interleavers rearrange the elements within a data array, and can thereby reduce Fourier domain peaks away from DC. This method of peak reduction was found to be inferior to the reduction granted by use of additional phase masks.

6.2 Future Work

The results of this thesis can be extended in future work. The following items list a few possible areas in which this work can begin.

The performance of irreducible polynomials of degree 3 bears further investigation. In the case of degree 3 polynomials, irreducible polynomials performed as well as primitive polynomials. This finding was unexpected, given that irreducible polynomials were clearly inferior to primitive polynomials in the degree 2 and 4 cases. As presented in Chapter 3, the degree 3 irreducible polynomials produce sequences that are very similar to m-sequences, but this result was not fully investigated. It is likely that the similar performance of irreducible and primitive polynomials occurs again at some degree higher than 3; the most likely candidates being degree 5, 7, 9 etc.

Guided scrambling for sparsity encoding can be extended beyond ternary alphabets. Several of the modulation schemes outlined in Chapter 2 have been extended to create more than 3 channel symbols. These are useful additions because they allow more data to be stored per array than in the binary or ternary cases. The analysis of guided scrambling detailed in Chapter 3 is applicable only to the ternary case and must be re-evaluated for inclusion in a system operating with a higher order signalling alphabet. Likewise, selective phase masking can be extended for systems using larger alphabets.

The results presented in the thesis indicate the performance of these techniques in general, as discussed in Chapter 1. An applied demonstration of the techniques can be performed to determine effectiveness for a particular system. Such experiments should be designed to evaluate an overall decrease in BER due to the techniques developed for this thesis.

Finally, an interesting analytical view of the phase masking process was suggested during the review of this thesis [51]. It is possible to view an additional masked data array as the multiplication of a previous masked data array by an arbitrary phase mask. In this way, the distribution of peak values in the Fourier domain of a set of masked arrays can be determined by the convolution of the distribution of peak values of several masked arrays. Through the law of large numbers, this distribution approximates a Gaussian distribution. This theory is supported by the fact that the resulting Fourier domain peak values from the set of Gaussian masks approximately equals the value attained through the use of 50 pseudo-random phase masks. This approach has the potential to reveal the upper limit of Fourier domain peak value that is

possible with an arbitrarily large set of masks, and bears further investigation.

Bibliography

[1] D. Gabor, "A new microscopic principle," *Nature*, vol. 161, no. 4098, pp. 777–778, May 1948.

[2] J. Ashley, M.-P. Bernal, G. W. Burr, H. Coufal, H. Guenther, J. A. Hoffnagle, C. M. Jefferson, B. Marcus, R. M. Macfarlane, R. M. Shelby, and G. T. Sincerbox, "Holographic data storage," *IBM J. Res. Develop.*, vol. 44, no. 3, pp. 341–368, May 2000.

[3] L. Hesselink, S. S. Orlov, and M. C. Bashaw, "Holographic data storage systems," *Proc. IEEE*, vol. 92, no. 8, pp. 1231–1280, 2004.

[4] D. Gabor, "Microscopy by reconstructed wave-fronts," *Proc. of the Royal Soc. of London. Series A. Mathematical and Physical Sciences*, vol. 197, no. 1051, pp. 454–487, July 1949.

[5] E. N. Leith and J. Upatnieks, "Reconstructed wavefronts and communication theory," *Journal of the Optical Society of America*, vol. 52, no. 10, pp. 1123–1130, October 1962.

[6] ——, "Wavefront reconstruction with continuous-tone objects," *Journal of the Optical Society of America*, vol. 53, no. 12, pp. 1377–1381, December 1963.

[7] ——, "Wavefront reconstruction with diffused illumination and three-dimensional objects," *Journal of the Optical Society of America*, vol. 54, no. 11, pp. 1295–1301, November 1964.

[8] F. S. Chen, J. T. LaMacchia, and D. B. Fraser, "Holographic storage in lithium niobate," *Applied physics letters*, vol. 13, no. 7, pp. 223–225, October 1968.

[9] D. Close, A. Jacobson, J. Margerum, R. Brault, and F. McClung, "Hologram recording on photopolymer materials," *Applied Physics Letters*, vol. 14, no. 5, pp. 159–160, March 1969.

[10] L. Hesselink, S. S. Orlov, A. Liu, A. Akella, D. Lande, and R. R. Neurgaonkar, "Photorefractive materials for nonvolatile volume holographic data storage," *Science, New Series*, vol. 252, no. 5391, pp. 1089–1094, November 1998.

[11] P. Cheben and M. L. Calvo, "A photopolymerizable glass with diffraction efficiency near 100% for holographic storage," *Applied Physics Letters*, vol. 78, no. 11, pp. 1490–1492, March 2001.

[12] M. R. Gleeson, S. Liu, and J. T. Sheridan, "Improvement of photopolymer materials for holographic data storage," *Journal of Materials Science*, vol. 44, no. 22, pp. 6090–6099, July 2009.

[13] H. Audorff, K. Kreger, R. Walker, D. Haarer, L. Kador, and H.-W. Schmidt, "Holographic gratings and data storage in azobenzene-containing block copolymers and molecular glasses," *Advances in Polymer Science*, vol. 228, no. 1, pp. 59–121, 2010.

[14] P. Lundquist, C. Poga, R. DeVoe, Y. Jia, W. Moerner, M.-P. Bernal, H. Coufal, R. Grygier, J. Hoffnagle, C. Jefferson, R. Macfarlane, R. Shelby, and G. Sincerbox, "Holographic digital data storage in a photorefractive polymer," *Optics Letters*, vol. 21, no. 12, pp. 890–892, June 1996.

[15] E. Zarins, V. Kokars, A. Ozols, and P. Augustovs, "Synthesis and properties of 1,3-dioxo-1h-inden-2(3h)-ylidene fragment and (3-(dicyanomethylene)-5,5-dimethylcyclohex-1-enyl)vinyl fragment containing derivatives of azobenzene for holographic recording materials," *Proc. SPIE*, vol. 8074, pp. 80 740E:1–6, 2011.

[16] Q. Mohammed Ali, P. K. Palanisamy, S. Manickasundaram, and P. Kannan, "Sudan iv dye based poly(alkyloxymethacrylate) films for optical data storage," *Optics Communications*, vol. 267, no. 1, pp. 236–243, 2006.

[17] A. Ashkin, G. D. Boyd, J. M. Dziedzic, R. G. Smith, A. A. Ballman, J. Levinstein, and K. Nassau, "Optically-induced refractive index inhomogeneities in $LiNbO_3$ and $LiTaO_3$," *Applied Physics Letters*, vol. 9, no. 1, pp. 72–74, July 1966.

[18] R. Castagna, F. Vita, D. E. Lucchetta, L. Criante, and F. Simoni, "Superior-performance polymeric composite materials for high-density optical data storage," *Advanced Materials*, vol. 21, no. 5, pp. 589–592, February 2009.

[19] L. K. Roland Walker, Hubert Audorff and H.-W. Schmidt, "Blends of azobenzene-containing polymers and molecular glasses as stable rewritable holographic storage materials," *Proc. SPIE*, vol. 7619, pp. 76 190H:1–9, 2010.

[20] T. Sabel, S. Orlic, K. Pfeiffer, U. Ostrzinski, and G. Grtzner, "Free-surface photopolymerizable recording material for volume holography," *Optical Materials Express*, vol. 3, no. 3, pp. 329–338, March 2013.

[21] J.-H. Chen, C.-T. Yang, C.-H. Huang, M.-F. Hsu, and T.-R. Jeng, "Study of optical properties of glass-like polymer material for blue laser holographic optic data storage recording," *IEEE Transactions on Magnetics*, vol. 45, no. 5, pp. 2256–2259, May 2009.

[22] J. F. Heanue, M. C. Bashaw, and L. Hesselink, "Volume holographic storage and retrieval of digital data," *Science*, vol. 265, no. 5173, pp. 749–752, August 1994.

[23] B. E. A. Saleh and M. C. Teich, *Fundamentals of Photonics*, J. W. Goodman, Ed. Wiley-Interscience, 1991.

[24] G. W. Burr, G. Barking, H. Coufal, J. A. Hoffnagle, and C. M. Jefferson, "Gray-scale data pages for digital holographic data storage," *Optics Letters*, vol. 23, no. 15, pp. 1218–1220, August 1998.

[25] G. W. Burr, H. Coufal, R. K. Grygier, J. A. Hoffnagle, and C. M. Jefferson, "Noise reduction of page-oriented data storage by inverse filtering during recording," *Optics Letters*, vol. 23, no. 4, pp. 289–291, February 1998.

[26] B. M. King and M. A. Neifeld, "Unequal a-priori probabilities for holographic storage," *Proc. SPIE*, vol. 3802, pp. 40–45, 1999.

[27] R. John, J. Joseph, and K. Singh, "Holographic digital data storage using phase-modulated pixels," *Optics and Lasers in Engineering*, vol. 43, no. 2, pp. 183–194, 2005.

[28] J. Joseph and D. A. Waldman, "Homogenized Fourier transform holographic data storage using phase spatial light modulators and methods for recovery of data from the phase image," *Applied Optics*, vol. 45, no. 25, pp. 6374–6380, September 2006.

[29] T. Sarkadi, P. Koppa, F. Ujhelyi, J. Reményi, G. Erdei, and E. oke Lõrincz, "Holographic data storage using phase-only data pages," *Proc. SPIE*, vol. 7000, pp. 700 004:1–11, 2008.

[30] B. Das, S. Vyas, J. Joseph, P. Senthilkumaran, and K. Singh, "Transmission type twisted nematic liquid crystal display for three gray-level phase-modulated holographic data storage systems," *Optics and Lasers Engineering*, vol. 47, pp. 1150–1159, 2009.

[31] J.-S. Jang and D.-H. Shin, "Optical representation of binary data based on both intensity and phase modulation with a twisted-nematic liquid-crystal display for holographic digital data storage," *Optics Letters*, vol. 26, no. 22, pp. 1797–1799, November 2001.

[32] J. Reményi, P. Várhegyi, L. Domján, P. Koppa, and E. Lõrincz, "Amplitude, phase, and hybrid ternary modulation modes of a twisted-nematic

liquid-crystal display at $\sim$ 400 nm," *Applied Optics*, vol. 42, no. 17, pp. 3428–3434, 2003.

[33] G. Berger, M. Dietz, and C. Denz, "Hybrid multinary modulation codes for page-oriented holographic data storage," *Journal of Optics A: Pure and Applied Optics*, vol. 10, no. 11, November 2008.

[34] S. Jutamulia and T. Asakura, "Optical Fourier-transform theory based on geometrical optics," *Optical Engineering*, vol. 41, no. 1, pp. 13–16, January 2002.

[35] B. M. King and M. A. Neifeld, "Sparse modulation coding for increased capacity in volume holographic storage," *Applied Optics*, vol. 39, no. 35, pp. 6681–6688, December 2000.

[36] B. M. King, G. W. Burr, and M. A. Neifeld, "Experimental demonstration of gray-scale sparse modulation codes in volume holographic storage," *Applied Optics*, vol. 42, no. 14, pp. 2546–2559, May 2003.

[37] A. Sütő and E. Lőrincz, "Optimisation of data density in fourier holographic system using spatial filtering and sparse modulation coding," *Optik*, vol. 115, no. 12, pp. 541–546, 2004.

[38] I. J. Fair, W. D. Grover, W. A. Krzymien, and R. I. MacDonald, "Guided scrambling: A new line coding technique for high bit rate fiber optic transmission systems," *IEEE Transactions on Communications*, vol. 39, no. 2, pp. 289–297, 1991.

[39] S. W. Golomb, *Shift Register Sequences*, rev. ed. Walnut Creek, California: Aegean Park Press, 1982.

[40] S. R. Blackburn, "A note on sequences with the shift and add property," *Designs, Codes, and Cryptography*, vol. 9, no. 3, pp. 251–256, 1996.

[41] S. Halevy, J. Chen, R. M. Roth, P. H. Seigel, and J. K. Wolf, "Improved bit-stuffing bound on two-dimensional constraints," *IEEE Transactions on Information Theory*, vol. 50, no. 5, pp. 824–838, May 2004.

[42] J. Tellado, *Multicarrier Modulation with Low PAR.* Kluwer Academic Publishers, 2000.

[43] C. B. Burckhardt, "Use of a random phase mask for the recording of Fourier transform holograms of data masks," *Applied Optics*, vol. 9, no. 3, pp. 695–700, March 1970.

[44] Q. Gao and R. Kostuk, "Improvement to holographic digital data-storage systems with random and pseudorandom phase masks," *Applied Optics*, vol. 36, no. 20, pp. 4853–4861, 1997.

[45] M. J. O'Callaghan, "Sorting through the lore of phase mask options - performance measures and practical commercial designs," *Proc. SPIE*, vol. 5362, pp. 150–159, 2004.

[46] M. J. O'Callaghan, J. R. McNeil, C. Walker, and M. A. Handschy, "Spatial light modulators with integrated phase masks for holographic data storage," *Proc. SPIE*, vol. 6282, pp. 628 208:1–9, 2006.

[47] W. Ren, Q. Ma, L. Cao, Q. He, and G. Jin, "Applications of phase masks in volume holographic data storage system and correlators," *Proc. SPIE*, vol. 6827, pp. 682 710:1–8, 2007.

[48] F. Przygodda, J. Knittel, O. Malki, H. Trautner, and H. Richter, "Special phase mask and related data format for page-based holographic data storage systems," *Optical Review*, vol. 16, no. 6, pp. 583–586, November 2009.

[49] J. Hong, I. McMichael, and J. Ma, "Influence of phase masks on cross talk in holographic memory," *Optics Letters*, vol. 21, no. 20, pp. 1694–1696, October 1996.

[50] M.-P. Bernal, G. Burr, H. Coufal, R. Grygier, J. A. Hoffnagle, C. M. Jefferson, E. Oesterschulze, R. M. Shelby, G. T. Sincerbox, and M. Quintanilla, "Effects of multilevel phase masks on interpixel cross talk in digital holographic storage," *Applied Optics*, vol. 36, no. 14, pp. 3107–3115, May 1997.

[51] J. Ilow, private communication, 2013.

Appendix A

Evolutionary Algorithm

In this appendix, the function of the evolutionary algorithm is described in detail. The main phases of the algorithm are divided into separate sections for organizational clarity.

The algorithm was first developed in Matlab, and then was recreated in C++ to allow for longer simulations due to the inherently faster performance of the compiled C++ program. The results presented in the thesis for phase masking with a different number of pseudo-random masks used the C++ implementation. The results for the different designs of mask as well as the interleaver results were produced with the Matlab implementation. Conceptually the two implementations are identical, the only major difference is that the C++ version does not utilize over-sampling or zero-padding while calculating Fourier transforms. This appendix is focused on the algorithm itself, but refers specifically to functions used in the C++ implementation.

There are several variables that are hard-coded for each run of the algorithm. The values of these variables do not change over the course of a single run, but may be changed between runs as described in Chapter 4. The variables that are changed include the x and y dimensions of the arrays, and the number of masks.

The following variables were not changed between runs and have the value shown in parentheses: the number of arrays per generation (50), the number of generations per simulation (50), the number of iterations of the simulation per

program execution (10), the minimum sparsity value (0.3), and the mutation probability (0.0005).

A.1 Initialization

The first phase of the algorithm is the initialization phase. The first step of the program is to draw a seed for the pseudo-random number generator (RNG) that is used throughout the program. The first seed is the current system time represented as an `unsigned long`. The RNG used is the MTRand package in the C++ implementation (the Matlab implementation uses the native RNG in Matlab).

After the RNG is seeded, the actual algorithm begins by allocating the appropriate number and size of masks, represented by a three dimensional array of `int` values. Each mask is generated by randomly drawing a $+1$ or -1 for each element of the array.

Next, the initial generation of data arrays is randomly generated. The set of arrays is represented by a three dimensional array of `int` values. The data arrays are first generated by drawing a $+1$, 0, or -1 value at random for each element in the array.

A.2 Selection

With the set of masks and the first generation initialized, the algorithm proceeds to the selection phase for the first time. In the selection phase, two new variables are initialized to store the value of the largest Fourier domain peak yet encountered and the index of the data array that produced the peak.

To begin the selection phase, the first data array is copied from the set of data arrays into a temporary array. A score is calculated for this array. This score is found by calculating the Fourier transform of the data array and the Fourier transforms of each array that results from the element-wise multiplication of the data array and each mask array. Then, the maximum magnitude value of each of the Fourier transforms is determined (resulting in

a total number of values equal to one more than the number of masks in the simulation). The minimum of these values is the score for that data array. Likewise, a score is computed for the remaining data arrays in the current generation.

After each score is computed, the algorithm checks this score against the current maximum score. If the new result is greater than the current maximum, then that array becomes the new selected array; its score is stored to be compared against all forthcoming scores, and its index is stored so that the algorithm can quickly find the array that produced the maximum score.

After all arrays in the current generation are scored, the highest score is recorded into the array that keeps track of the score per generation. Then, the algorithm proceeds to the mutation phase.

A.3 Mutation

In the mutation phase, the algorithm loads the data array with the highest score into a temporary array. This temporary array is then mutated to create each data array of the subsequent generation. The mutation is carried out by drawing a random value between 0 and 1 for each element in the array[1]. If this value is less than the mutation probability, the element is incremented by 1. In the event that the value was previously a 1, the new value is set equal to -1 instead of 2. If the value is greater than the mutation probability, the element is not changed for the new array.

After a mutated array is generated, the number of 1 and -1 values is counted. If this value is greater than the maximum number of ON-pixels allowed, some of the non-zero pixels are set to zero. This is accomplished by recording the number of non-zero pixels that must be set to zero. While that value is greater than zero, an element in the array is selected at random. If that element is non-zero, it is set to zero and the counter is decremented by 1. If the element is zero it is ignored and the counter is not changed. This continues until the counter reaches zero. This function can theoretically

[1]This is implemented in C++ via the MTRand function `drand()`, which provides a uniform distribution of numbers on the interval [0,1].

run indefinitely if the random pixels are always zero, but in practice non-zero pixels are selected frequently enough that its processing time is negligible in comparison to the rest of the evolutionary algorithm.

After the last new array is generated by mutation, and made sufficiently sparse the algorithm has a new generation with which to begin the selection process again. As such, it returns to that phase, and iterates until the defined number of generations have been simulated.

A.4 Finalization

After the algorithm as run for the specified number of generations, the data must be exported. This is done in C++ with libraries provided by Matlab that export data to a file type that can be loaded by Matlab. The last surviving array and set of masks are exported as Matlab two-dimensional `int` arrays. The running score is exported as a one-dimensional `double` vector. Additionally, the x and y dimensions, and seed value are exported as well. These values are all contained in a single `.mat` file with the file name generated by the number of masks used and a counter to prevent previously existing files from being overwritten. For instance, if the algorithm finds a file with the currently generated file name in its folder it will append a 1 to the end of the file name, and if the new file name exists as well it removes the 1 and appends a 2 and so on until a file name is generated that does not already exist.

After the data has been saved, the program can terminate. Typically it is configured to run multiple times, and it begins from the beginning until the specified number of iterations have been processed.