ON THE COMPRESSION OF DIGITAL HOLOGRAMS
(Spine title: On the Compression of Digital Holograms)
(Thesis format: Monograph)

by

Andrew Chan

Graduate Program in Computer Science

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

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Supervisor:

.....
Stephen M. Watt

Examiners:

.....
Mahmoud El-Sakka

.....
David J. Jeffrey

.....
Roberto Solis-Oba

The thesis by

Andrew Chan

entitled:

On the Compression of Digital Holograms

is accepted in partial fulfillment of the
requirements for the degree of
Masters of Science

.....
Date

.....
Chair of the Thesis Examination Board

Abstract

This thesis investigates the compression of computer-generated transmission holograms through lossless schemes such as the Burrows-Wheeler compression scheme (BWCS). Ever since Gabor's discovery of holography, much research have been done to improve the recording and viewing of holograms into more convenient uses such as video viewing. However, the compression of holograms where recording is performed from virtual scenes has not received much attention. Phase-shift digital holograms, on the other hand, have received more attention due to their practical application in object recognition, imaging, and video sequencing of physical objects. This study is performed for virtually recorded computer-generated holograms in order to understand compression factors in virtually recorded holograms. We also investigate application of lossless compression schemes to holograms with reduced precision for the intensity and phase values. The overall objective is to explore the factors that affect effective compression of virtual holograms. As a result, this work can be used to assist in the designing of better compression algorithms for applications such as virtual object simulations, video gaming application, and holographic video viewing.

Keywords: computer-generated holograms, digital holography, hologram compression, Burrows-Wheeler

Acknowledgements

I want to take this chance to express my deepest gratitude to my supervisor, Professor Stephen M. Watt, for the patience he had with my struggle in a topic in which I have not much knowledge but the greatest interest. Also, I would like to thank him for his advice and ideas which gave me new insight in ways on how I view the world and approach problems.

To my colleagues in the Ontario Research Centre for Computer Algebra, I would like to thank them for their help and advice helped me get through a lot of problems.

For all the professors and staff members in the Department of Computer Science from whom I have learned or worked with, thank you for teaching me all I know in Computer Science and life. Your lessons will be well-cherished.

For all other friends along my university career, especially those in the Computer Science Undergrad Society, I thank you all for being there for me and making life as a Computer Science student enjoyable.

Last but not least, I wish to thank my beloved family, without their support and love, it would not have been possible to write this thesis.

Contents

List of Figures

List of Tables

List of Appendices

Chapter 1

Introduction

The primary objective of this thesis is to explore the factors that affect compression for interference-based computer-generated holograms using various compression schemes. Different factors of compression are investigated to determine which most affect compression of computer-generated holograms. Holograms where a computer models the holographic film is the main object of investigation in this thesis, and not on the viewing capabilities of the reconstructed hologram. Emphasis is placed on the factors that increase the compression rate while not losing in compression speed. This aims for better hologram storage and for possible implementation of a new compression method in the future for holographic applications.

The discovery of holography started with research in improving electron microscopes [14]. Hungarian-British physicist Dennis Gabor, discovered what is known in the present as "holography" in an attempt to improve electron microscopes. He realized that when a coherent reference wave interferes with the wavefront diffracted by sending a similiar coherent wave by an object, the resulting interference pattern can be recorded. However, back then the application was more focused on electron microscopy. It was not until the 1950s when the understanding of holograms expanded from the rising popularity of lasers, which allowed coherent beams of light to be easily generated. Through decades of research, the understanding of holography now allows us to model and calculate a virtual scene on a computer and create holographic video sequences of the virtual scene using a spatial light modulator (SLM) [32]. However, the amount of space required to store the holograms on a computer is still too great for conventional use, the dimensions of viewable video holographic displays are still in the size of centimetres at best and the scenes are mainly one or two modeled objects. Thus, there is still more room for improvement for holograms in terms of recording, storage, and displaying.

Holography is the only technology today where the display of an image can contain all four depth cues— accommodation, binocular disparity, occlusion, and moving parallax [32]. As such, interest in this field becomes more apparent as the need for realism in image viewing increases. However, the amount of space required to store a typical still hologram on a computer is enormous for modern computers and limits rendering speed. Traditionally, a interference-based computer-generated hologram of viewable size ($10mm^3$ for example) requires storage size of hundreds of megabytes. This size requirement is due to the storage of each object point separately spaced in micrometres or nanometres. Larger holograms may need gigabytes. Given the increase in storage size available in the present day, storing holograms with all depth-cues would be a smaller issue in decades with conventional hard drives growing

from gigabyte to terabyte storage in the last decade. However, currently there is not a way to display interference-based holograms the size of hundreds of megabytes in real time. It is not possible for conventional computers to quickly generate or reconstruct a hologram to a display, and it is even less likely for multiple holograms (even if they are simple ones) to be stored in a conventional computer that can be displayed quickly. Therefore, an effective solution for how the data should be compressed and represented becomes an issue of importance.

1.1 The Basics of Holography

To understand the basics of holography, the nature of light must be understood. This section is mainly based on physical holography [15, 22]. Computer-generated holography will be discussed in another section. Traditionally, holograms are recorded under a coherent beam of light and saved on a physical recording medium (usually some form of photosensitive film). A typical setup includes the recording medium, the object(s) of interest to record (which can be highly transparent or reflective depending on the type of hologram), a laser (the coherent beam of light), a beam-splitter, and a lens if the type of hologram recorded requires it. The laser is then split into two separate beams to create an object beam and a reference beam. The object beam will then be used to illuminate the object, and in turn illuminate the recording medium. When this happens, some light will be absorbed while some light will be diffracted out and comes in contact with the recording medium. The reference beam will be used to illuminate the recording medium in which interference is created, and the pattern is recorded onto the medium.

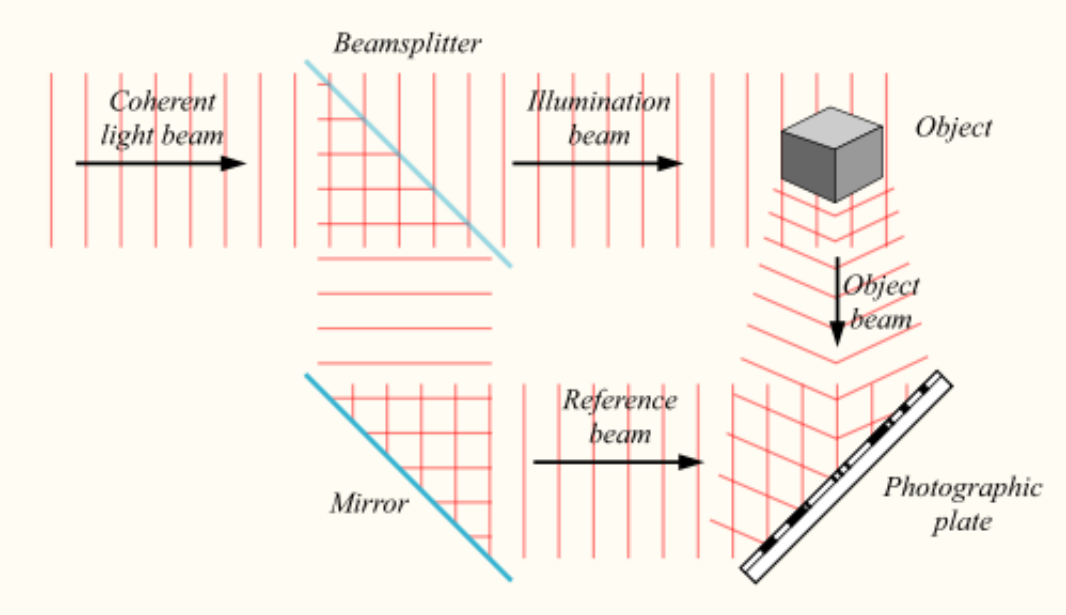


Figure 1.1: Traditional Hologram Recording Setup from [38].

After the object has been recorded on the recording medium, a reference wave will be shot from the same or similiar angle as the recording interference wave onto the medium. A three-

dimensional image similiar to the object will be shown, and the image is the hologram. At each point on the recording medium is a contribution from point sources of light shone at it during the recording. Assuming the wavefront distribution on the recording medium plane (the hologram plane) is:

$$U(x, y) = |A(x, y)|exp[j\phi(x, y)]$$

where A(x, y) is the amplitude at a given point, $\phi(x, y)$ is the phase, $j = \sqrt{-1}$, and

$$U'(x, y) = |a(x, y)|exp[j\eta(x, y)]$$

is the reference beam's wavefront, a(x, y) is the amplitude of the reference beam, $\eta(x, y)$ is its phase, the intensity of the combined wavefront becomes:

$$I(x, y) = |A(x, y)|^2 + |a(x, y)|^2 + 2|A(x, y)||a(x, y)|cos[\phi(x, y) - \eta(x, y)].$$

Expression $|A|^2$ and $|a|^2$ are known as the zero-order terms. They contain useless information that creates extra noise in the final illuminated hologram. The remaining term contains all the information required to recreate the final image. However, it includes information on creating two images: a virtual image and a real image. The real image is the "conjugate" of the virtual image which allows the viewer to see two hologram images when illuminated.

1.1.1 Items of Concern in Physical Holography

Traditionally, the material of the object and recording medium affect the hologram's quality greatly. Objects used in the recording must be transparent or reflective enough for light to diffract or reflect from the object and illuminate the recording medium. Outside of that, the recording medium becomes the next issue. A hologram is recorded onto a photographic film or plate. Other examples of recording medium would be dichromated gelatin, photoresists, or photopolymers. For such media, they must be sensitive enough to light to record the pattern in a short enough time period as to avoid any physical disruptions during recording such as physical jitter or over-exposing. The recording medium then must be converted into something that can modify the phase or amplitude due to the difficulty in modifying both phase and amplitude at the same time — classified as phase or amplitude holograms [15]. For a amplitude hologram, the intensity is affected by the absorption of light by the hologram, which is more or less absorptive depending on the intensity of light. Phase holograms are affected by the optical distance in the material, which changes the hologram's effectiveness by the refractive index or thickness.

The process of hologram creation and viewing is a complicated one. Another problem to worry about when creating or viewing holograms is the recording environment. Since, the recording medium is very sensitive to light, holograms must be recorded and viewed in dimly lit areas. Any environment with strong lighting could cause the film to change (the recording medium might record white light which is a mix of many different waves) and degrade the hologram's quality. Therefore, many different ways of developing the medium and substitutions for the recording medium have been proposed. Different improvements to holographic recording and viewing will be further discussed in later sections.

Some Methods to Producing Holograms

The following are brief descriptions of different ideas to hologram production and viewing. Most of the descriptions are based on [15].

Gabor Hologram

The most original type of hologram is the Gabor Hologram [14]. It is based on Gabor's original attempt to improve on electron microscopes. For this type of hologram, the objects used in hologram recording must be highly transmissive for the light to be shone out of the object and onto the recording medium. In order for the whole object to be recorded, a lens is placed in between the object and the light source so the light can be directly focused onto the object and allow light to penetrate the object fully. With the object being highly transmissive, the reference wave and object wave are both supplied by the object. Therefore, this setup does not require an uninterrupted reference wave.

Fourier Holography

Fourier holograms [15, 16] are completely based on transparent objects and the modeling of the equations used on it (as the name suggests). The recording plane for this setup is in a plane that will create the Fourier transform of the object amplitude transmittance. The object is again focused with a lens as with Gabor holograms with the exception that the positions of the object and recording medium forms a Fourier transform of the object amplitude transmittance. In Fourier transform holograms, the front focal plane of the lens is placed with the object while the rear focal plane is placed with the recording medium. The relation of the two plane fields is then of a Fourier transform. If the reference beam is planar, the light from each point on the object will interfere with the reference beam, creating a sinusoidal fringe with a vector spatial frequency that is to unique to that object point. For viewing, one only needs to place the recorded hologram in front of a positive lens, with an incident plane wave and look for the resulting image in the rear focal plane.

There is also a type of hologram called the lensless Fourier transform hologram. This type of hologram is different in terms that the reference beam doesn't penetrate the object. Rather, it is put into focus at the object plane and interferes with light that passes through the object on the recording plane. A lens is still required in this setup but the difference is the reference only needs to create interference at the recording plane. This is basis of most other physical holographic setups that follow.

Fresnel and Fraunhofer Holography

Fresnel and Fraunhofer holograms are based on the Fresnel or Fraunhofer diffraction equations [15, 21]. This type of hologram is similiar to a Fourier or Gabor hologram, since they also rely on transparent objects. However, the imaging and diffraction conditions are a bit different. As stated in the Fourier Holography section, the relation between the rear and front focal plane of the lens is a Fourier transform. The key difference between these two types of holograms is the diffraction and imaging conditions are best described by the Fresnel or Fraunhofer diffraction

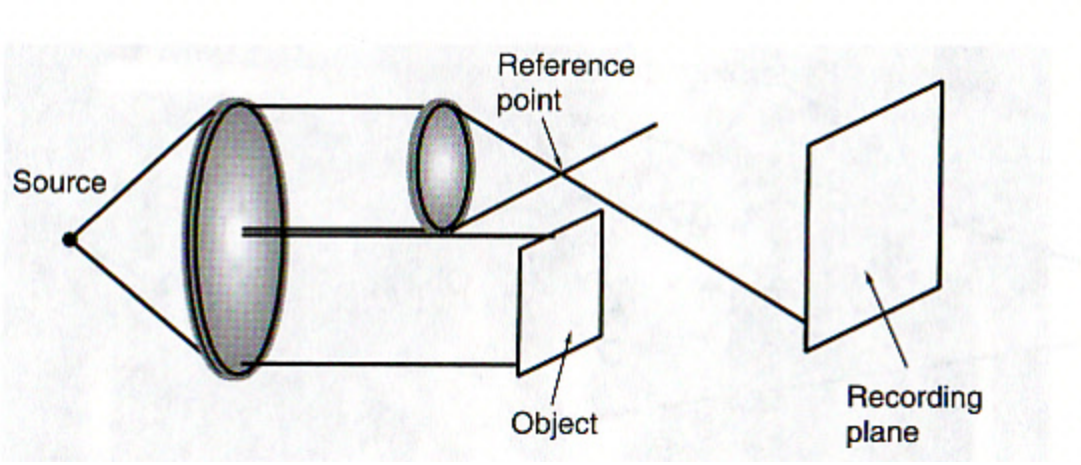


Figure 1.2: Lensless Fourier hologram setup from [15]

equation. Thus, the physical setup for it will yield the best description if it is based on the Fresnel and Fraunhofer diffraction equation.

Reflection Holograms

The previously mentioned holograms are typically called transmission holograms where the image is viewed through the transmission of light through the hologram. There is another type of hologram that is not based on viewing light through a hologram but rather light reflecting off a hologram. This type is called a reflection hologram [11]. The basic setup is to place the recording medium between the object and light source. There is only one beam that will illuminate through the hologram, which acts as both the object and reference beam. When the beam illuminates the recording material, the light will pass through the hologram and onto the object. The object will then reflect light back to the holographic plate, and the plate will record the information from the reflected light. A person wanting to view this hologram can then use an incident beam to illuminate the holographic film and the viewer can see the hologram on the same side as the light. This type of hologram can be viewed even with white light. However, colour differences during recording could occur during photographic emulsion treatment, since the drying step of the film can cause differences in colour. This is something that can be easily fixed with proper chemical treatment during the process.

Stereo Holograms

A type of hologram that makes use of the stereo effect of the eyes are stereo holograms. There are different methods to creating such holograms, most involving the use of overlapping images. One of the ways used is the use of multiple photographs at different perspectives [10]. Each photo will then be projected with an object beam and a reference beam onto a photographic plate through a movable slit. The photos are put in sequence from their perspective and the slit is moved as each photo is recorded. The final recorded hologram will contain a

side-by-side recording of different photographs through different perspectives, due to the slit blocking redirected light at different perspectives. When the final product is chemically treated and illuminated by a reference beam, the result is an image that looks three-dimensional due to each point being a combination of images recorded from different slits and perspectives. Another method would be to overlap holograms of different perspectives on a photographic plate. The Nintendo 3DS, a new portable video gaming console, utilizes the slit idea to create three-dimensional images.

Rainbow Holograms

The name rainbow holograms originates from the use of white light for hologram illumination [3]. For white light to be usable as the illumination method, it must minimize blur in the hologram due to colour dispersion in transmission holograms, by removing one side of parallax. The hologram is created in two steps. The first step entails creating a first hologram using the conventional method with a monochromatic or near monochromatic light. The reference and object wave will be processed in the same way as traditional holograms. The newly created hologram will then be illuminated with the reference beam in the opposite direction of the previous recording's reference beam. This results in the real image being generated and a narrow horizontal slit is introduced in front of the real image and a second hologram is recorded in front of the slitted image. The second hologram then has information of both the slitted hologram and the original hologram, created from the "anti-reference" wave and resulting in a diverging spherical wave. When a viewer looks at the hologram from the slit's image plane, the light viewable is only for a narrow band of colour. Thus the image will appear free from colour blur - resulting in no colour blurring from dispersion. In different positions, the hologram will contain different colours due to the removal of one side of parallax and the slit's colour band acceptance.

Embossed Holograms

Perhaps the best known type of holograms used today would be the embossed hologram [15]. Even though the name is not widely known, its applications on security cards, credit cards, magazines, books, and sometimes money is very wide-spread. This type of hologram is reflective, as is apparent when someone observes the image from a credit card. The general process to create those is rather involved. In general, the objective is to create a metal plate that contains the recording material which can be embossed in polyester material. The first step is to record the object of interest in a photoresist. The type of photoresist selected will be adequate for the resolution. Usually a strong laser such as an argon-ion laser will be used to create the hologram. The next step involves creating a metal master hologram where the recorded hologram is sprayed in silver and immersed in a plating tank using electric current usually with nickel so that a thin layer of nickel can be on top of the photoresist. This layer of nickel is then separated from the photoresist for the metal master. Then, the metal master is submerged in a plating tank again for another electroforming process, creating metal submasters for stamping the actual embossed holograms. The metal submasters can then be used to create the actual hologram using methods such as flat-bed embossing, roll embossing, or hot stamping. In all

cases, the submaster is heated to a high temperature and stamped to a polyester material, which then becomes metalized to create a reflection hologram.

Computer-Generated Holography

Following the advent of the computer age, the generation of holograms on computers became an increasingly popular topic of research. Different researchers have started to look into the possibility of either transferring the recording or displaying of holograms to computers [5, 20, 22, 32, 35, 36]. Thus, the field of computer-generated holography became another field of major interest.

Computer-generated holography has now transformed into a very broad field that incorporates holographic interference pattern creation, recording object simulation, or display of such generated holograms. Whether it be the simulation of interference patterns between the object and reference beam [5, 36], printing of a generated interference pattern on a physical medium, or the display of the hologram using a 3D display that works on the principle of interference of coherent light [22], these are all coined under the umbrella of computer-generated holography. As of the time of this writing, computer generated holography has improved to the level where the calculation required to generate the interference patterns are given to specially designed hardware using FPGAs [19] and/or modern graphics cards with fast GPUs [27] to quickly generate holograms.

In any case, the virtual object wavefront from a virtual object and a virtual reference beam is calculated, and then the resulting wavefront on the image plane is stored either on a physical piece of recording material or on a computer. If it is a physical film, then the film will be illuminated by an actual reference beam to produce the hologram. If it is stored on a computer, it will require the appropriate spatial-light modulator to display the hologram. Seeing the final result as a critical step as to how the encoding of the wavefront should be, the method used to produce this scattered wavefront also becomes an important point of consideration.

Currently, there are two main families of computer-generated holograms [32]: interference-based and diffraction based. The two of them are based on totally different concepts from formulation to viewing. However, the two types are still conceptually based on the same steps to create holograms: computation of the scattered wavefront, encoding the wavefront to a viewable form, reconstructing the wavefront for viewing. Interference-based holograms employ the use of classical interferometry and use one of the previously mentioned methods of holographic recording as a basis of modelling. It recreates a virtual scene with similiar conditions where all objects are not in physical reality. This is advantageous to computer-generated hologram generation since there is no need to have physical objects involved, and all objects in the virtual scenes can have its colour and size adjusted to a desired size.

A prominent type of holography as of the time of this writing is diffraction-based holography. Originally proposed by Mark Lucente at MIT [22], this type of holograms has become popular in its simplicity in representation. In diffraction-based holography, the printing of the interference pattern on an actual printed physical medium is removed, replacing it with a spatial-light modulator (SLM). Classifying a hologram as small parts that diffract light into different direction, the machine used to display the hologram splits the hologram into hogels — holographic elements — and in return changes the problem from how to save the wavefront of the interfered object beam and reference beam to how light should be diffracted when the

viewer sees the hologram. The procedure on how the spatial light modulator will handle hogels is mainly based on the implementation of the SLM. However, since the physical setup of the hardware is now separated from the software calculations — and even can be split even further if the recording of the hologram is replaced by virtual scene calculations — the creation of holograms can be merged with modern GPU architectures for parallel calculations.

The biggest issue with computer-generated holograms is a vast field of research for present day researchers: how to calculate wavefront information when the plane contains information of the size of wavelengths quickly, how to reduce artifacts when generating holograms, and how to view generated holograms efficiently. The field of computer-generated holography will be further discussed in the next chapter.

Digital Holography

Another form of holography is known as digital holography [29]. The main goal for this type of holography is to determine three-dimensional surface data, such as the thickness or shape of an object. After determining the shape of the surface, imaging of the object can be done through the use of a computer screen or spatial light modulator. One of the more notable types are the phase-shift holograms. In general, a phase-shift hologram takes four different phase-shifted wavefronts from shifting the reference beam's phase, each shifted by $\pi/2$, and saved into a CCD camera or similiar device. A laser beam is shot into a beam splitter, where the beam separates into the reference beam that is shone onto the camera and the object beam that intersects with the object and bounces light back into the camera. Before the reference beam reaches the camera plane, the beam hits a retardation plate that modifies its phase. The camera then saves the phase-shifted combined wavefront individually. This results in four different wavefronts that can be later combined to the image desired.

This process causes not one, but four different copies of the original wavefront to be saved and then converted into one viewable form. The amount of time required, and the space required, is now quadrupled. Therefore, research into processing speed and space saving are deeply considered fields. As well, saving wavefronts into a camera means that the continuous stream of data has become discontinuous. Therefore, problems such as speckle effects and blurs still exist and must be handled appropriately.

1.2 Lossless Image Compression

The goal of compression is to reduce file size through redundancy reduction. On holograms, this becomes difficult because holography, by its nature, records information in a distributed, non-local form. Since the data is distributed over the hologram plane, it becomes difficult for most compression algorithms — which are mostly based on local context redundancy — to compress holograms successfully. However, a few compression schemes have been tested on holograms as a first test of effectiveness. As a result, lossy techniques were chosen for further research since lossy techniques can quickly reduce the compression speed given a certain loss in reconstruction quality. This means that the field of lossless hologram compression has not been fully investigated. Since this thesis is interested in furthering the understanding of lossless compression schemes in the application of interference-based holograms, the rest of this

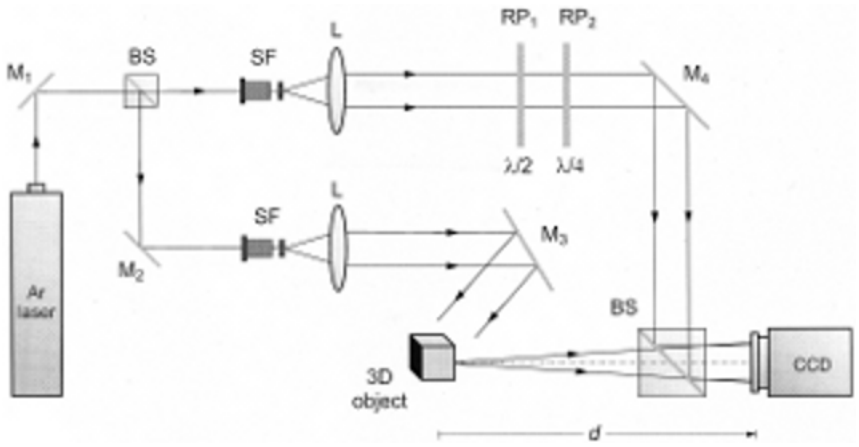


Figure 1.3: Experimental PSI Hologram Setup from [24]. M, mirror; BS, beam splitter; SF, spatial filter; L, lens; RP, retardation plate

section is a general overview of lossless schemes.
Note: for the purpose of clarity, each input symbol in the input file will be considered a character.

1.2.1 Huffman Encoding

The objective of Huffman Encoding is to reduce the number of bits used to represent the most frequently occurring characters from the given input [18]. The input is analyzed to determine the frequency of each character, then the most occurring frequency is assigned a number. These numbers are assigned in frequency order, so the most frequently occurring symbols are associated to the non-negative integers of shortest length (in their binary encoding) [18]. In this fashion, the most frequent characters can obtain the shortest binary representation in which the new representation outputs. The stream of characters will then be encoded with the new mapping. The resulting output will then be smaller since the compressed stream will require less bits to represent as opposed to a regular raw file. However, Huffman encoding is not without its limitations: it not possible to design a code with average bit rate equaling the entropy of evenly distributed input for large alphabet sizes, and it requires the storage of the bit representation to symbols mapping (if the mapping is not standard), which can take a lot of space and time to store and decompress. For images with evenly-distributed symbols, there is no guarantee to achieve any compression.

1.2.2 Arithmetic Encoding

Arithmetic encoding reduces the need for representing individual input symbols by representing the output as one numerically-coded output. Similiar to Huffman coding, the objective is to reduce the number of bits used to represent the most frequently occurring characters. This type of encoding requires the use of a probability distribution model to determine the frequent occurrence of characters. After the probabilities are determined, the probabilities are

used as a base for probability encoding for the input. This heavily requires the skewedness of the distribution of input characters. Each character in the alphabet of the input would have a probability indicating its probability of occurrence. During encoding, these probabilities will contribute to a cumulative density function where a certain interval of numbers will be split into subintervals by the function. Each character is classified by a subinterval based on the cumulative density function. The encoding of the input begins with looking for the first character's subinterval. After the subinterval is known, the output would indicate what the first character should be based on which subinterval the number belongs. To continue encoding, the previously encoding subinterval is split further into subintervals using the same cumulative density distribution. The next character's output value would be created from the subinterval of the previous character's subinterval based on the new cumulative density distribution. This value interval determination continues until the input file is totally read, where the value is a number between the final subinterval. This code in the end will result in the output, where the decoder follows the same process to reproduce the uncompressed file. By the end of encoding, the output is a number that represents the input. In arithmetic encoding, the symbols are split into intervals of a range using their respective cumulative densities and the first input symbol is marked as a number in that interval. That interval is then split into subintervals using the same cumulative densities, where the next input symbol's subinterval is located using the same process. This persists until the end of the input where the output is a number that represents the input. This type of encoding works best for a small range of input symbols of highly-skewed probability.

1.2.3 Limpel-Ziv

The Limpel-Ziv compression scheme is based on a dictionary approach. The general idea is to use part of the previously coded input to add to a dictionary in an attempt to reduce the amount of redundant information sent. How it works is as follows: create a fixed-size viewing window for the input. The left side would be called the search buffer and the right side would be the look-ahead buffer. The input is first contained in the look-ahead buffer, with the first character in the look-ahead buffer being the first entry added to a dictionary. The characters are then left-shifted by one character to the search buffer and then the actual algorithm starts. The rest of the input is processed as follows: each character at the front of the look-ahead buffer is then matched with characters in the search buffer. If there is a match, it searches backward in the search buffer for a match. If a match is found, it tries to match following characters of the matched character. If a mismatch is found after a match, it continues backward searching to find other matches. After the matching, the encoder will generate a token using the longest matched string. The token will be generated for the dictionary: The offset of when the previous match started, the length of the match, and the code of the next symbol. Items in the look-ahead then slides into the search buffer, bumping the left end of the search buffer out if the buffer capacity is at maximum. Decompression is simply following the dictionary and rebuilding right away which can be done in $O(n)$ time.

There are two main types of schemes created from this idea: LZ77 and LZ78 [37]. LZ77 follows the sliding window methodology as above. LZ78, on the other hand, starts off with an empty dictionary with a null character as the first entry. From then on, the dictionary is built with each entry containing two elements: A pointer to an entry in the dictionary and the

next character's code. Using this method, the length component is removed from the tokens due to implicit nature of the pointers to refer to previously matched characters. The character that caused the failure is then added to the dictionary with the pointer referring to the match entry. LZ77 contains an improvement known as LZSS. The main feature of this improvement is to reduce the need to use 3 parts to encode a symbol and use 2 parts when it is possible. The idea is to use two types of messages: a 3 part pointer message (a flag, a match offset, and a match length) if a sufficiently long match is found, and a character message (another flag and the failed character) otherwise. This ensures that new unmatched items are not being described with a long token made up of a variable length offset or match length. LZW is the most famous improvement for LZ78 [34]. In LZW, the new character token is totally removed. The dictionary starts up as every possible input character, and the dictionary is built the same way as LZ78.

The main weakness for both types of dictionary encoding is it does not work too well for highly-variable and large-sized input. In both cases, having a large amount of input symbols results in a dictionary that might take longer to decompress and a larger file size than the original file. For LZ77 schemes, the sliding window and search buffer size becomes a high concern for compression. There is no perfect metric to determine what size is optimal for which type of input and thus makes it hard to work with. For LZ78 schemes, the dictionary size is of main concern. If the data is highly variant, scanning the dictionary for matches becomes a concern since searching would take longer. Compression is also not-likely achieved in this sense from the point of view that the pointers may generate a longer amount of symbols than the original file. In terms of storage issue, a small dictionary size for LZ78 schemes may cause overflow, due to the amount of data that needs to be included in the dictionary.

1.3 Prediction by Partial Matching

The prediction by partial matching scheme (PPM) was designed to work with arithmetic encoding [2, 6, 23, 31]. It is a statistical modelling method simlliar to Dynamic Markov Models. In PPM, input symbols are not only individual characters: They are also considered by the characters that go before them. This is called a context. The order of the context is determined by the number of characters that is considered before the currently processed character. The idea is based on using different probabilities of the contexts. In PPM, the symbols are first built using the zero-order context (no previous symbols are considered). As the algorithm runs, the higher order contexts are built from what it sees similiar to the Limpel-Zev algorithms. The algorithm keeps scanning for previously entered symbols starting from the higher order contexts and goes one level down to each lower order contexts if a match to previously similiar input is not found. If by the zero-order context the input symbol is still not found, the character is added to the zero-order context with an initial count of 1 as well as the higher-order contexts with this new context. However, if a match is found before the zero-context, then the match's count is added and all corresponding lower order match's count are also added. For each match (or non-match), the symbol is marked through an arithmetic encoder using the cumulative density function of the corresponding context. This algorithm suffers from the same weaknesses of LZ algorithms and Arithmetic Encoding, where highly variable local input will cause problems in dictionary size and compression speed.

1.4 Burrows-Wheeler Compression Scheme

The Burrows-Wheeler Compression Scheme (BWCS) is a full compression scheme based on the reordering of data [4, 12, 33]. In simple terms, it sorts the data pseudo-lexographically in an attempt to increase redundancy of nearby characters. This sorted data then goes through extra processing such as run-length encoding to reduce its size. The theory behind decompression is a mathematical one, in which a bijection is used to reorder the data in the right position. In general, it is one transform, one setup for the next stage, and two encoding stages. To explain the whole scheme, the four separate stages of BWCS must be explained.

1.4.1 The Burrows-Wheeler Transform

The Burrows-Wheeler Transform stage is the main focus of the whole scheme. It starts with a file of N characters, where the file can then be split into blocks or the file can be considered as a whole. Either way, a matrix of of either N-by-N characters or B-by-B blocks for a block of B characters is created using the first row as the characters of N or B and each successive row as a left cyclic shift of the previous row. The matrix is then lexographically row-sorted, and the last column is extracted as part of the output. The extracted column and the position of the first and last characters are sent to the move-to-front encoding stage for compression.

1.4.2 Move-To-Front Encoding

Once the last column is extracted, it will be transformed into a different form. The last column is coded as follows: Assign the alphabet to an array the size of the alphabet. Read the characters from the extracted last column. Locate the character's position in the array and code the character as its current position in the array. Promote the character's position as the first in the array and demote the position of the other characters before the promoted character down by one position (the original first character is now at position 1 instead of 0). For each repetition of the same character, encode it as a zero. Once the next character is not the same as the previous character, encode the different character as its current position in the array, promote it to position 0, and encode repetitions as zeros again. This is repeated until all the bytes given from the previous stage are encoded. The result is then passed to the run-length encoding stage.

1.4.3 Run-length Encoding

The next stage involves changing all the zeros from the previous stage into shortened versions. To prevent cases in which there are many zeros dominating and the probability estimation during entropy encoding becomes difficult, the Zero Run Transform [12] was proposed to shorten the input even further. The Zero Run Transform, if done in binary, can also be the entropy encoder for the result. Otherwise, the result is passed down to the entropy encoder.

1.4.4 Entropy Encoding

The last stage involves encoding the data into binary. Either Huffman Encoding or Arithmetic Encoding are usual chosen entropy encoders. The bzip2 compression tool currently uses Huff-

man Encoding.

1.4.5 Decompressing the BWT

Decompression for the BWT file requires the inverse of the previous three steps to return the sorted last column to be recovered. After that, with the knowledge of the first character and last character, the two can be used as a base mapping of first and last character, where the next character's location can be discovered through the permutation of the first and last character. This continues until the end of the data is fully recovered.

1.5 Motivation

The main motivation for this thesis is based on the need to generate holograms on a computer with virtual objects instead of recording from the physical world. In our growing world, the computational time and storage size required are decreasing due to better hardware and the improvement of parallel algorithms. In time, generation of holograms on a computer can have many applications such as using graphics engines to create moving holographic objects on spatial-light modulators. Thus, this thesis is done to investigate the factors of hologram compression such that future work can be done for improving compression methods and possible insight into the design of fast computer-generation of holograms that are effective for storage.

Within the last two decades, many different improvements came about to improve many aspects of computer holography. From fine-tuning creation parameters that reduce the generation load to using various methods to viewing, the problems with creating and viewing still holograms quickly are slowly eroding into a historic issue. A notable improvement would be the use of computers to simulate an actual hologram recording environment instead of a physical setup would remove the need for specific locations and tools to record a hologram. This method complements the viewing of holograms using specific hardware instead of traditional film since the recording and viewing parameters are directly related to each other. One of the most notable contributions to this improvement for holography would be Mark Lucente's diffraction-specific holography [22], where generation of holograms is focused on how the reconstructed light should be diffused instead of just the basic interference from the object and reference beam. This method allows a frame rate of 10 frames per second for an image of 100,000 points using a special-purpose clustered computing system at a size of 1920×1080 [19].

While hologram creation and viewing have been targets of extensive research within modern times, due to factors such as the size of the holograms and the limitation of current hardware and algorithms, there hasn't been a lot of research that looks into the compression of holograms for storage. Therefore, there has not been a lot of focus on how to best store holograms for later viewing.

The field of phase-shifted digital holography was one of the first methods that investigates compression on holograms. In phase-shifted digital holography, a physical environment is still used for recording — the viewing part is the side that requires computer-generation. A CCD camera is used to record the scene's intensity distribution on the camera plane at four different shifted-phases (typically 0, $\pi/2$, $3\pi/2$, and 2π). Then, the actual phase and amplitude informa-

tion is calculated using the four different intensity patterns. The phase and amplitude information is saved, and later the hologram can be reconstructed and displayed on a computer screen. This method of producing holograms takes extra storage and time during the recording phase since it requires the recording of four phase-shifted intensity patterns. However, the space of the final hologram is fixed since the data size is based on the number of pixels in the camera instead of sample points. Using an in-line (reference and object beams are co-linear) setup, hologram quality is improved over other types of physical holograms through the removal of the zero-order and conjugate-image terms that causes blurring. However, such a setup suffers from speckle noise which is natural of a camera's discretized sampling of the the continuous physical world.

1.6 Thesis Organization

The rest of this thesis is organized as follows. The second chapter consists of related works contributing to the thesis. The third chapter gives the proposed setup for the experiments. The fourth chapter is an analysis of hologram images (real, amplitude, phase) and factors that should be noted during the experiments, while the fifth and sixth chapter are the experiments done and the analysis of the results. The last chapter lays out the conclusion and possible future work.

Chapter 2

Related Work

This thesis uses previous work from several areas, which we summarize here.

2.1 Graphical Raytracing

At the time of writing, one of the most popular computer graphics methods in creating three-dimensional images is graphical raytracing. This form of computer graphics, based on a physics origin, employs a backwards approach as to how an image should look like at a certain perspective. From [17], the general idea is as follows: Each pixel on an image is assumed to be a point source that can shoot light outward into space. This light then may or may not go into contact with an object, but if the light comes into contact with an object, the colour and intensity of that point will be saved.

After the point of contact, the point affected will check for other objects or actual light sources that can be affected. One of the ways to check if the observed colour or intensity should be different is by separately identifying actual light sources in the virtual scene such as a lamp or light bulb. By simply checking for intersections of the object point and light vector from source to point, the observed intensity can be altered. If there is blockage between the light source and the object point in question, the intensity will be darkened by the shadow in question.

Another way of checking for the actual colour and intensity of this intersected point would be to look for neighbouring objects that might affect the shading of the object. The best way to check for this would be to see if the normal vector of each object point intersects with any objects including itself (the object point of a donut may face the other side of the donut). If it does intersect, the intersecting point may be close enough to affect the intensity or colour of this point. If it is physically close enough to affect the colour or intensity, then some kind of interpolation must be done to balance out the colours in the final image. This process is repeated per pixel until all possible pixels are accounted.

The modelling becomes even further refined if we model the points with more physical effects such as light absorption, reflection, refraction, and fluorescence. Graphical effects such as ambience, diffusion, specular, and emission effects can be added to add more realism to these physical effects. The result is extra realism in terms of image quality, with a huge overhead in the time it takes to produce the actual image. However, modern GPU architectures, different

modelling improvements, and optical tricks have improved the calculation time greatly. A ray-traced scene can be create and viewed from seconds to minutes on an Intel Duo Core computer with a NVIDIA GeForce 8 series video card, for example, depending on the complexity of the virtual scene. The overall quality of video game graphics is a proof to the practicality of this graphical technique.

Diffusion is a particular problem in graphical raytracing. Since all objects in the world diffuses light one way or another and at wavelength sizes, the modeling of such a term causes a lot of speed problems. Therefore, experiments were done to determine how much light would be diffused using different shading models such as Blinn-Phong, Lambert, Oren-Nayar, and Minnaert. In these shading models, diffusion is usually a fixed term that describes how much intensity contributes to diffused light.

2.2 Computer-Generated Holography

As mentioned in the previous chapter, computer-generated holography is now an umbrella term that incorporates anything from just simple generation of the scattered wavefront of holograms to the encoding and display of the recorded or calculated hologram on a computer. Perhaps the biggest advantage of using computers to generate holograms is the removal of the physical setup to record a hologram. Using a virtual setup, the virtual scene can be calibrated to conditions that can be suitable for viewing using known physics equations. Since there is no perfect equation that can fully describe how light reacts, setting the scene in favour of the restrictions of the used equations allows holograms to be easily generated in the physical world.

The difference between a simple photograph and a hologram must be explained to better understand holograms. A simple photograph is a two-dimensional representation of what a user may view in the physical world. A hologram, on the contrary, differs from a photograph in the fact that it contains depth cues that make it appear three-dimensional. As explained in [32], four types of depth cues are of concern: Accommodation, parallax, binocular disparity, and occlusion. Accommodation refers to the ability to focus on an image. Parallax is an effect where during movement the object of interest (in this case a hologram) appears differently in the physical sense (the parts of a certain side of an object that was not revealed becomes revealed). Binocular disparity is the based on the fact that humans have two eyes - what the left and right eyes see are two different images since the eyes are separated and thus by triangulation humans can perceive the object's depth. Occlusion is the ability for a certain object to appear in front or behind another object. These are important visual information that a human must have before the perception of depth is available. An interference-based hologram can produce all depth cues. Diffraction-based methods, originally developed by Mark Lucent [22], sometimes ignore either horizontal or vertical parallax in exchange for generation speed [22]. The field is so vast that each of the three steps - virtual wavefront generation, wavefront encoding, and hologram display - requires a special section of its own to be explained in further detail.

2.2.1 Virtual Wavefront Generation

In a physical hologram, a physical recording material records the phase and amplitude information of the combined object and reference wavefront. However, in the virtual sense, recording all possible amplitude and phase information in a computer is impossible since data in the physical world is continuous, while a computer can only contain discrete information of the combined wavefront due to space limitations. Also, the proper equations and scene must be used to model the wavefronts and its combination, usually a discrete Fresnel or Fourier transform equation. Therefore, a computer-generated hologram of good quality must: 1) contain enough sampling points for the object and hologram; and 2) be modelled with the correct equations and setup.

The two conditions for a good virtual wavefront are intertwined. To know a good sampling-size for the points, the modeling equation must first be considered. The rest of this subsection will be based on the book "Introduction to Fourier Optics" by Joseph Goodman [15]. Consider a Fourier hologram. The setup for this type of hologram is a lens of infinite size and a certain focal length f. The lens will exist between the object and hologram field. The object field will then be in the back focal plane of the lens while the hologram will be in front of the focal plane. This means the back focal field (object field U_o (x, y) and the front focal field $U_h(\epsilon, \eta)$ are related by a Fourier transform:

$$U_h(x,y) = \frac{1}{\lambda f}\iint_{-\infty}^{\infty} U_o(\epsilon,\eta) exp\left[-j\frac{2\pi}{\lambda f}(\epsilon x + \eta y)\right] \mathrm{d}\epsilon\, \mathrm{d}\eta$$

where x and y are the location of the hologram point, and ϵ and η are the location of the object points.

To find the number of samples required for the hologram field, the spatial bandwidth (the amount of spacing per point) must be determined. This is determined by the Whittaker-Shannon sampling theorem [15] - the number of samples must be twice as much as the highest frequency. In the case of Fourier holograms, the hologram's bandwidth is determined by the object's size. Suppose that the object is $L\epsilon$ by $L\eta$. The hologram plane is also assumed to be a rectangular shape of dimensions $2B_X$ by $2B_Y$. Therefore,

$$2B_X = \frac{L_\epsilon}{\lambda f}$$
$$2B_Y = \frac{L_\eta}{\lambda f}.$$

Therefore, the spacings should be

$$\Delta x = \frac{1}{2B_X} = \frac{\lambda f}{L_\epsilon}$$
$$\Delta y = \frac{1}{2B_Y} = \frac{\lambda f}{L_\eta}.$$

Given that the dimensions of the hologram is L_X by L_Y, the number of samples should be:

$$N_X = \frac{L_X}{\Delta_x} = \frac{L_X L_\epsilon}{\lambda f}$$
$$N_Y = \frac{L_Y}{\Delta_y} = \frac{L_Y L_\eta}{\lambda f}.$$

Since the object and hologram planes are directly related to each other due to the Fourier nature of the transform, the number of samples required are the same.

The Fourier transform equation is defined by integrals. This is a continuous description of the hologram plane which can not be used on a computer. Therefore, the equations must be discretized into a usable form. The sampling space of the hologram plane being (Δx, Δy) and the object plane sampling space being ($\Delta\epsilon$, $\Delta\eta$), the discretized version of the previous equation becomes:

$$U_h(p\Delta x, q\Delta y) = \sum_{m=0}^{N_X-1} \sum_{n=0}^{N_Y-1} U_o(m\Delta\epsilon, n\Delta\eta) exp\left[j2\pi(\frac{pm}{N_X} + \frac{qn}{N_Y}) \right].$$

This discretized version can be then calculated using a discrete Fourier transform algorithm that can rapidly calculate the wavefront. However, after careful inspection of the bandwidth, the dimensions that constitute the number of samples depend on L, which is normally a length of viewable size (at least in the millimetres) divided by λ, which is the wavelength of light in nanometres. This means that a significant amount of samples ranging anywhere from the hundred thousands to billions is needed before a full wavefront can be generated.

2.2.2 Wavefront Encoding

From the previous section, it is apparent that storage space is an issue that requires a lot of attention due to the amount of information involved in holograms. After knowing under what conditions to model the virtual scene, the choice of representation for the combined wavefront must be decided. In order for a hologram to be of good quality and timely creation, knowledge of what the ultimate form of display must be determined. This means the display type gives insight to how the hologram should be represented in storage. The resulting wavefront is a discrete but complex set of amplitudes and phases at each point. However, it is not easy to create machinery that can directly control both amplitude and phase of the amplitude transmittance given the relation between amplitude and phase, so there must be a way to represent this information to the output. This section will use a technique known as the Kinoform to illustrate how the wavefront should be encoded using interference-based approaches. The first assumption that must be done for this type of hologram representation is that the phases carry the majority of information about an object, such that amplitude information can be removed. This is an assumption that is effective with diffuse objects (all object points are assigned random and independent phases).

Placing this in a Fourier holographic setup, the hologram can be divided into N_X by N_Y cells, each one represents a Fourier coefficient of the object [15]. The amplitudes for each cell will be assigned the same value, and the phase is encoded via linearly mapping the phase range (which ranges from 0 to 2π) into a continuum of gray levels displayed by an output device such

as a photographic plotter on a transparency. The resulting transparency is photographically bleached, which solidifies the mapping of phase to grayscale values and makes it ready for display. If the bleaching process is managed well enough such that the mapping is accurate, the image obtained will be of good quality. If the phase match fails from the bleaching, then only a single bright spot will be produced on the mismatched points for the desired image. The transparency can then be illuminated by a plane wave to obtain the image. The type of output for the hologram used in this thesis is similiar to this form except there is no need to photographically bleach the transparency.

2.2.3 Hologram Display

The most traditional mediums for displaying holograms include photopolymer materials and transparencies. However, electronic mediums are becoming more prominent in recent times. Electronic mediums known as spatial-light modulator can manipulate the intensity or phase of light being shone out of it to create the desired image. For example, a digital-micromirror device (DMD) - an electronically-addressed spatial-light modulator - was used in the research for Xu et al [35]. The overall shape of the image is produced using the ultra fast frame rate of a DMD caused from quickly flicking light shone onto the mirrors on and off. The user only needs to look directly into the DMD to be able to view the hologram. In this sense, a DMD can use the turning of mirrors on and off relatively quickly (binary pulse-width modulation) to create an illusion of grayscale. Contemporary DMDs can produce up to 1024 shades of gray. Some holographic displays use LCD screens to directly produce the illusion of 3D images through the phase modulating properties of liquid-crystals. However, since this is out of the scope of this thesis, further discussion is not performed.

2.3 Phase-shift Digital Holography

The origins of phase-shift holography stems from object and pattern recognition, and from the need to not rely on the chemical processing and mechanical focusing of recording materials. In a sense, it is a field of computer-generated holography in which only the reconstruction and display of the combined wavefront is performed on a computer. However, the biggest difference comes from the fact that wavefront generation is still done in the physical world with an actual CCD or similiar device.

In a typical setup such as the one in [36], a two-beam interferometer is used where a laser beam is divided into two paths: One which contains the object beam while the other contains the reference beam. The object would then be directed to a transmitting or reflecting object, whose light will ultimately be the object's contribution to the wavefront on the camera's pixels. The reference wavefront, on the the other hand, will be blocked by a piezoelectric transducer mirror, which has the ability to phase shift the reference beam. The interference pattern of the object and reference wavefront are taken four times by the CCD camera, each time with a different phase. The four interference patterns taken by the camera will vary successively by $\pi/2$, where the first pattern will not contain a phase shift (phase at 0). The following are equations as defined by Yamaguchi and Zhang in [36]. Assuming the object point is located at

x_o, y_o, z_o, the object wave can then be described by:

$$\begin{aligned} U(x,y) =& A exp(i\phi) \\ =& \frac{A_o}{z_o} exp\left[i\phi_o + ikz_o + ik\frac{(x-x_o)^2 + (y-y_o)^2}{2z_o} \right], \end{aligned}$$

where $A_o \exp(i\phi_o)$ is the complex of the point object and k is the wave number. A reference wave $U_R(\phi_R) = A_R \exp(i\phi_R)$ becomes superimposed on the object wave. The camera will only record the intensity of the superimposition, in which the camera's recorded intensity becomes:

$$\begin{aligned} I(x,y;\phi_R) =& |U_R(\phi_R) + U(x,y)|^2 \\ =& A_R^2 + A^2 + 2A_R A\cos(\phi_R - \phi). \end{aligned}$$

The object's phase can then be determined through knowledge of the intensity:

$$\phi(x,y) = \tan^{-1}\frac{I(x,y;3\pi/2) - I(x,y;\pi/2)}{I(x,y;0) - I(x,y;\pi)})$$

The reconstruction of the hologram uses the Fresnel transform equation:

$$U_1(X,Y,Z) = \iint U(x,y) exp\left[ik\frac{(X-x)^2 + (Y-y)^2}{2Z} \right] \mathrm{d}x\,\mathrm{d}y.$$

If we substitute the object wavefront equation back in, we get:

$$\begin{aligned} U_1(X,Y,Z) =& \frac{A_o}{z_o} \\ & \times exp\left[i\phi_o + ikz_o + \frac{ik}{2}\left(\frac{X^2+Y^2}{Z} + \frac{x_o^2 + y_o^2}{z_o}\right) \right] \\ & \times \iint exp\{-ik\left[x\left(\frac{x_o}{z_o} + \frac{X}{Z}\right) + y\left(\frac{y_o}{z_o} + \frac{Y}{Z}\right) - \frac{x^2+y^2}{2}\left(\frac{1}{z_o} + \frac{1}{Z}\right) \right]\} \mathrm{d}x\,\mathrm{d}y. \end{aligned}$$

This equation can then be used to calculate the actual hologram image from a certain angle.

2.4 Compression of Phase-Shift Holograms

2.4.1 Conventional Methods

Phase-shift digital holograms are based on phase-shift interferometry (PSI). As such, these types of digital holograms are usually called PSI holograms. Early studies on compression of PSI holograms by Naughton et al [24] involve using only the most popular conventional compression schemes: Huffman Coding, LZ77, LZW, Burrows-Wheeler, and JPEG. Due to the highly entropic nature of hologram point values, simple tests on these algorithms did not provide a good understanding of compression on holograms. The following is a description of their results. The tests were performed using a 4 megapixel camera with dimensions 2028 ×

2044.

The first experiments were with the compression of the amplitude and phase at the camera plane. Out of 5 different holograms, the highest compression rate would be the Burrows-Wheeler Transform at an average compression rate of 1.95 if the phase and amplitude are treated as one single stream of information and 4.66 if they are treated separately. The next best would be LZ77 at 1.33 for single stream and LZW at 3.20. For Huffman encoding, the average compression ratio is barely near 2.0 so it was not considered at all. The reason behind the varying compression ratios of LZ77 and LZW is due to the addition of a lookup table, in which the data varies too much across the hologram to keep a small lookup table. That is the biggest advantage of having a compression algorithm such as the Burrows-Wheeler algorithm, which does not rely on a look up but rather a reorganization of information. However, the Burrows-Wheeler Transform itself does not rely on a lookup table and instead reorganizes all the data in a more compressible form.

Another method would be to re-sample the hologram's complex wavefront. By re-sampling the hologram using a filtering operation and/or resizing, the discontinuity of the pixels and speckle effect may be reduced. However, using a filtering operation causes certain details to be removed and may cause stronger quality loss in the reconstructed hologram. After filtering, the re-sampled hologram can be compressed using a conventional compression scheme such as BWCS. Through re-sampling, filtering, and discarding the reconstructed object phase information, the average compression rate becomes 8.29 for one of the holograms but may cause a normalized root mean square error with the original image of up to 0.38 using a Burrows-Wheeler scheme. As the results of [24] suggests, holographic microfringes are usually of sizes comparable to camera pixels. Therefore, downsampling would easily cause undersampling and cause significant quality degradations.

Another technique that is comparable to re-sampling would be to use quantization and reduce the number of bits that can be used to store spatial domain information (real and imaginary format). The quantized information would be median filtered to balance out the spread of data across the plane. After experimenting with different bits, it was found that 4 bits used to represent real and imaginary information would give it a normalized root mean squared error of less than 0.1 with a compression rate of 16 from the 5 holograms experimented.

The last test performed by Naughton et al in their paper [24] is to take a Discrete Fourier Transform (DFT) of each 8×8 pixels in the real-imaginary format. The algorithm followed is the same as JPEG, except the DFT is used instead of the Discrete Cosine Transform. This means that quantization is performed on the hologram. For one of the holograms, using 11×11 median filtering, 92% of the DFT coefficients are removed with a 0.22 normalized root-mean-square error. The compression rate was improved to 12.8. Naughton et al discovered that to reduce the loss percentage to less than 0.1%, no more than 78% of the coefficients can be removed but results in a compression rate of 4.6. The remaining non-zero DFT coefficients are stored with 8 bytes, so further compression techniques such as Burrows-Wheeler compression can be used to further improve the compression rate.

2.4.2 MPEG-4 Encoding on PSI Hologram Sequences

In a recent paper by Darakis and Naughton [9], the investigation of encoding holographic sequences are performed. In their paper, the camera is slowly rotated to record holograms

in sequence. The resulting sequence is encoded using MPEG4 part 2 and compared to the uncompressed hologram sequence using a normalized root mean square error metric. After applying MPEG4 part 2 compression on the sequence, the zero-order image is removed using a high-pass filter. The result is a compression rate of around 20 but exchanged with an average normalized root mean square error of 0.645 for four different hologram sequences. The results of this paper suggest that speckle noise is increased and the edge of the objects are darker due to compression being mainly a low-pass filter operation, where high frequency areas are reduced but edges are of higher frequencies in a hologram due to the object's edge having the highest incident angle with the reference beam. Such are artifacts that can happen with lossy compression methods.

2.4.3 Other Compression Tests

Following Naughton et al in [24], Darakis and Soraghan [8] also did some compression on interference patterns and complex wavefronts using lossy techniques such as quantization or JPEG compression. They confirmed that high compression rates (anything beyond 2 on interference patterns and 5 on complex wavefront at the CCD plane) results in normalized root-mean-square errors of over 0.3, which means strong degradations will start occurring after this point.

2.5 Summary

Currently, digital holography has been a popular method for compression research due to its usefulness in 3D object recognition and its application to 3D imaging. Since commercial use of holograms that fully contains all four depth cues are not present, the focus on compressing holographic data were not looked at in great detail outside of imaging. However, with the advent of commercial three-dimensional display hardware being sold (e.g.: 3DTVs, Nintendo 3DS), the time when compression becomes a focused field of research will not be far. As of the time of this writing, Naughton et al in [25] have already started researching internet transmission of holographic data and creating video sequences of real world objects using similiar methods as MPEG-4 encoding. Even though the need for digital holograms will always be present, the need for compressing virtual holographic objects are also very important due to its usefulness in many different applications including 3D visualization of objects, video gaming, and training simulations. This thesis is focused on virtual objects for this reason.

Chapter 3

Proposed Method

The wish is to further understand the factors of lossless compression on holograms generated on a computer with virtual objects. The objective for this is to better understand the factors for effective compression of holograms generated from virtual objects. Since there are not a lot of studies on compressing holograms that are not physically recorded, it would be of great interest to investigate compression with virtually recorded holograms for computer created hologram applications.

This study is purely focused on investigating lossless compression of holograms in which the virtual objects used for recording are mainly two-dimensional shapes. Two-dimensional shapes are used as a starting point for compression study to understand factors concerning holograms in general — the compression of holograms is generally the compression of the two-dimensional plate that contains the holographic fringes. Since the study is based on compressing holograms and not on viewing, preserving hologram quality is required for it to be easily adaptable to different forms of three-dimensional displays. Further studies can be done to target compression on specific output devices. Since most conventional lossless compression schemes are based on text-compression, it seems ill-suited to test their effectiveness for compressing computer-generated holograms. However, the one-dimensional schemes also take advantage of spatial information and compression ideas. Therefore, it provides good insight on the study of effectiveness. The following compression schemes were selected for investigation:

- Arithmetic Encoding
- Burrows-Wheeler Compression Scheme
- Huffman Encoding
- LZSS
- LZW
- PPMc
- PPMd

Arithmetic and Huffman are mainly bit reduction schemes: they reduce the number of bits required to represent each input symbol. LZSS and LZW are dictionary-based schemes which store redundant input through a dictionary. Burrows-Wheeler and PPM scheme are context-based schemes which require the identification of either local and/or block redundancy to be effective. These are good building blocks for hologram analysis due to their variation in approaches.

Most of these schemes have been tested by Naughton et al [24]. However, Naughton used phase-shift holography, which involved physical recording of the scene. Using a different recording setup, the tools may have a different effect on compressing the holograms. In addition, lossy methods were not used since the purpose is to not lose any information during compression. To test lossy techniques, an examination on resulting image quality would be required. In order to test the image quality of the resulting hologram, a viewer would be required to reconstruct and measure the hologram at various positions. Creating a viewer and doing various measurements would be out of the scope of this thesis.

There are also various lossless two-dimensional compression schemes that are useful when compressing images. The reason for not using these for the test is to try and find factor affecting the compression of holograms. The focus is on the search of factors that assist with building a good hologram compression scheme and not on testing two-dimensional compression schemes.

Finally, we point out a very important additional factor, the precision required to represent the data values. At each point in the hologram both a phase value and an intensity value are recorded and there is a choice to be made for the precision with which to record these numbers. We investigate how the compression rate varies with this choice.

3.1 Recording Setup

From the first chapter, it is known that using a physical setup to create holograms is non-trivial due to the physical limitations of the scene setup. Easily-diffusible objects and translucent objects are not easy to find as well as objects that are small enough for the item to be fully recorded on the holographic film. The physical environment must be in a dark room and must be without jitter, which is something that becomes inconvenient. In this sense, the advantage of using computers to generate holograms becomes attractive, especially with the increased computing power and understanding of holograms. In the proposed setup, the following substitutes the physical recording environment:

- the reference beam is replaced by a virtual light source at a predetermined distance and location relative to the hologram plane and object
- the object is replaced with a model of an object or a picture on a computer
- the need for lenses or object beam are replaced with using traced rays (an approach borrowed from graphical raytracing) to determine the correct amplitude and phase information from the object to the hologram plane
- no knowledge of the film is required since it's only using a virtual hologram plane

- polarization and other physical factors can be adjusted with a few parameters

For the actual recording environment, the recording of the hologram will be done with the Cortical Cafe Computer-Generated Hologram Maker software available on the internet [5]. Its type of hologram is of transmission quality; the resulting real image can be printed on a transparency and illuminated by a regular laser pointer. It is similiar to a Fourier hologram except it was designed not to use a 2D Fourier Transform on the image and perform reconstruction through the lens. In a typical physical setup, the reconstructed object lies on the hologram plane. However, this application is modified for holograms to be projected on a diffusible viewing area from a certain depth so it cannot be considered a real Fourier hologram. The scene would be recorded in the following way:

1. setup the location of the virtual holographic plate, the virtual reference beam, and the virtual object with each point on the plate having the same amplitude and phase (1 and 0 respectively)

2. for each sampled point on the plate and the object, calculate the amplitude and phase of light traveling from the object point to the hologram plane point

3. sum the calculated amplitude and phase to the complex wavefront (real and imaginary) point on the plate point with any previously calculated complex wave on the same point

This method is similiar to a raytracing setup except it calculates complex wavefronts as output instead of intensity. The need for diffusion is minimal since a traditional transmission setup usually requires highly translucent objects where diffusion is minimal. However, small randomizations in phase can also be used to model the absorption of light. Different wavelengths can also be used in the calculation for different laser pointer wavelengths.

3.2 Hologram Viewing - Transmission Holograms

The viewing of holograms in this setup can be done by printing the real image on any regular transparency using either a laser printer or on an inkjet printer through printing a filtered binary image of the real image. A typical inkjet printer has a resolution of at least 300 to 600 dots per inch, which is enough for viewing a small hologram of less than a square inch. Laser printers would be of better quality since they can represent more shades of gray similiar to the Kinoform hologram display discussed in the previous chapter. Figure 3.1 are reconstructed images of holograms from [5]. Even though the output format is currently on physical transparencies, the complex wavefront information can be easily used for computer-generated viewing of the hologram to spatial-light modulators.

These holograms are viewable within a size of approximately one square inch. This is due to the fact that the object used for recording is assumed to be the same size of the print size of the output transparency, which is exactly one square inch. Holograms suffer from having the reconstructed image the same size as the actual object's size unless illuminated with a different wavelength. It is also sensitive to resizing as proven by Naughton et al in [24].

3.3 Settings Critical to Virtual Setup

From previous sections on virtual hologram recording, it can be seen that a virtual setup involves as much parameter tuning as in a physical setup. However, these setups can be easily tuned to various values once they are implemented. The following are a few critical parameters involved in this study.

3.3.1 Pixel Density

The term "pixel density" used in this setup is the amount of evenly spaced sample points in 2D that represent the points on the transparency. The resulting printout would be a square inch since printers print by dots (pixels) per inch. In general, the higher the pixel density, the better the image quality. No image may result from illuminating the transparency if the printed hologram's resolution does not match the printable pixel density range of the printer, or if the printer is not of good printing quality. Another item of concern would be the sampling rate on the transparency. The x and y sampling rate will determine exactly which point on the one inch hologram are to be printed, and the virtual position of the plate point to calculate the distance of light from plate point to object point. If the sampled points are too far apart, the laser may not be able to produce any holographic images. Therefore, these discretization problems should be noted.

3.3.2 Depth

The hologram plate generation application in this study does not use lenses to focus holograms—it makes the hologram plane itself act as a lens so that images can be seen at a certain depth. Therefore, this setting becomes important for the focusing of images. If not set at a depth where the focus of the whole image is accounted for, part or all of the illuminated hologram may lose focus and become blurry.

3.3.3 Wavelength

The wavelength of light is always one of the most important factors affecting the recording of light. This is the main factor for what type of laser to use for recording and reconstructing the object. The application used allows user-entered wavelengths. Also, since this is a virtual scene, other issues such as polarization of light do not require fine-tuning - they are just equations if there are requirements to change. For this study, the waves will be sinusoidal.

3.3.4 Object Sampling Rate

The number of virtual object samples are critical to the creation of holograms. These sample points directly represent the reconstructed points on the outputted image. Therefore, they must be small enough to allow adequate viewing of the final outputted image.

3.3.5 Phase Randomization

In a physical setup, the diffusion of light from objects does not require any special setup. Due to the difficulty to properly model diffusion in light on computers, a random phase is usually introduced in addition to the calculation of each plate point. This will usually result in a more uniform looking hologram. However, since the phases added to each point are randomized, there is no guarantee that problems such as added noise won't occur. This is one of the toughest minor problems to solve when dealing with virtual holograms.

3.3.6 Recording Time

In terms of recording time, it is a factor that depends on the algorithm that is used to produce the hologram and the hardware available to the recording computer at the time. Typically, modern computers contain threading capabilities through multiple cores or through modern GPU architectures. Virtual holograms of not very complex scenes can be virtually recorded in minutes or seconds. However, the widespread use of these capabilities are only beginning and requires time to evolve.

3.4 Summary

The recording setup for virtual holograms becomes very advantageous for users to create holograms for different objects without the need to have anything more than a computer to create. However, the transferring of recording in a physical scene to recording in a virtual scene are not without its inherent problems. Problems with recording time and possible loss of information due to discretization are only some of the notable problems that occur with computerized holograms. Image size is also another issue with such setup. However, these problems are slowly thinning away with newer technology. In time, virtual hologram generation will be easy and effective for widespread use.

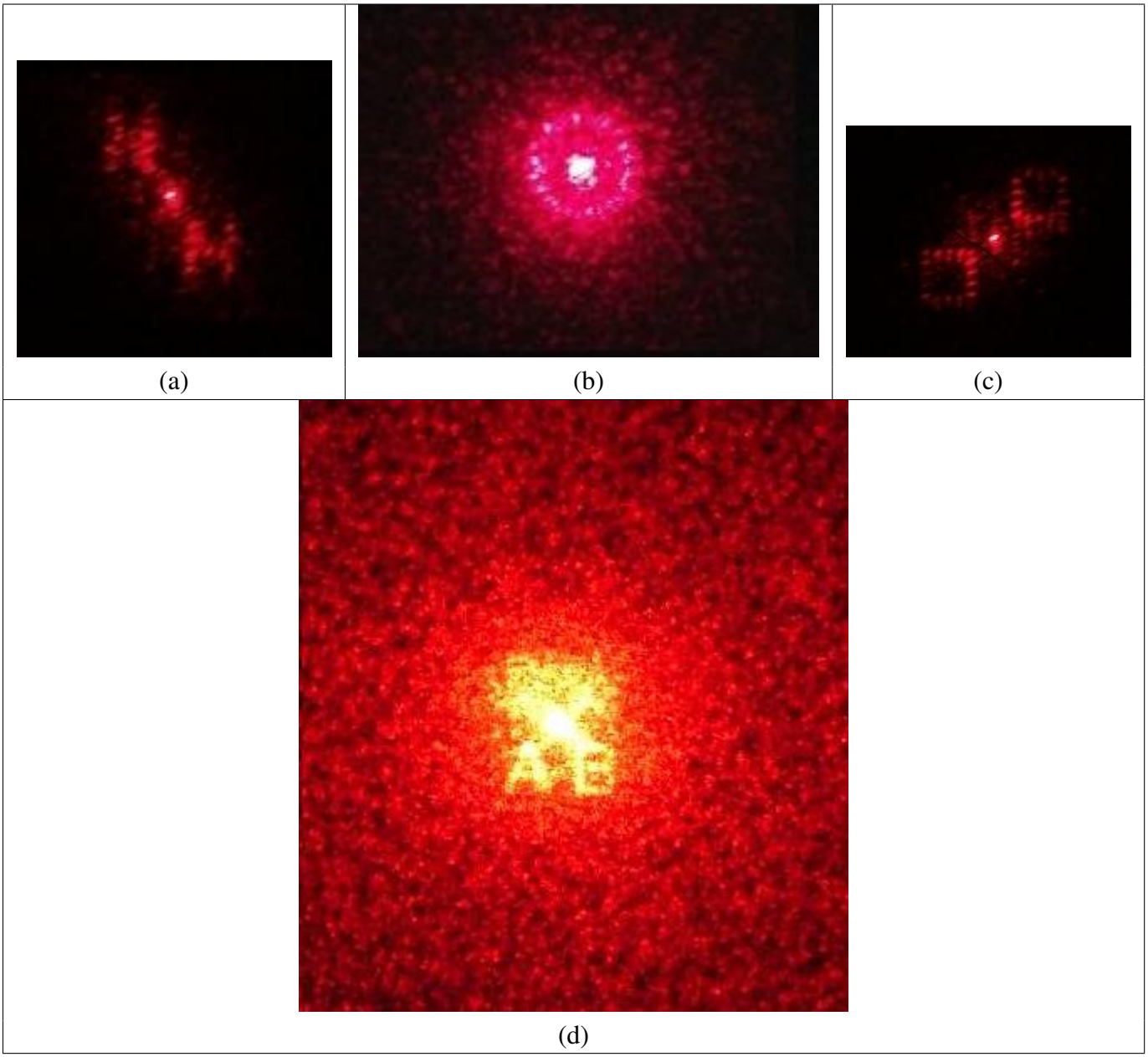


Figure 3.1: Images of reconstructed holograms from [5]. (a) The letter “H”. (b) A big circle. (c) A square. (d) An “AB”.

Chapter 4

Images Used For Compression

The holograms were created using the Cortical Cafe Computer-Generated Hologram Maker (CGH Maker) available on the Cortical Cafe Computer-Generated Hologram Construction Kit site[5]. It is a traditional transmission type hologram, where the resulting image can be printed on a regular transparency and can be displayed using a red laser pointer of 630 to 690 nanometres.

It contains settings such as recording length, pixel density (pixel per inch), phase randomizing, scaling, wavelength, and sampling rate. The maker takes mainly 2D gif images (the object in question is a flat 2D picture) or 3D describable XML descriptions as input. Other than the real image, it can also print the imaginary, amplitude, phase, and intensity as gif images.

For the purpose of this thesis, the settings in Figure 4.1 are for hologram creation. Outside of those identified above, everything else are set on default (as shown in Figure 4.1). The sample rates are the default sample rates suggested by the pixel density when chosen to make use of the even spread of pixels per inch from a typical printer (300, 600, 1200 dots per inch). Phase randomizing is not used to act as a diffuser for each object point. This term would allow highly variable values that can affect with compression. There are a few reasons for not setting the phase randomizing parameter:

1. It would cause strong variations in data which makes the resulting hologram harder to compress.

2. It can be easily added back during transparency printing or viewing on an electronic device.

3. It is not a fully reliable diffuser— it does not accurately describe how diffusion really works and cause more speckle noise.

The depth in the application is specified as the resulting viewing distance from where the reconstructed object will be focused. The optimal view depth calculation is selected to avoid focusing problems which a manually selected depth may create. Centering the object in front of the hologram is done to avoid any loss of the final holographic image since most of the objects used in this thesis are two-dimensional GIF images. For output types, the real image is the only thing used for experiments since that is the final product used for hologram viewing.

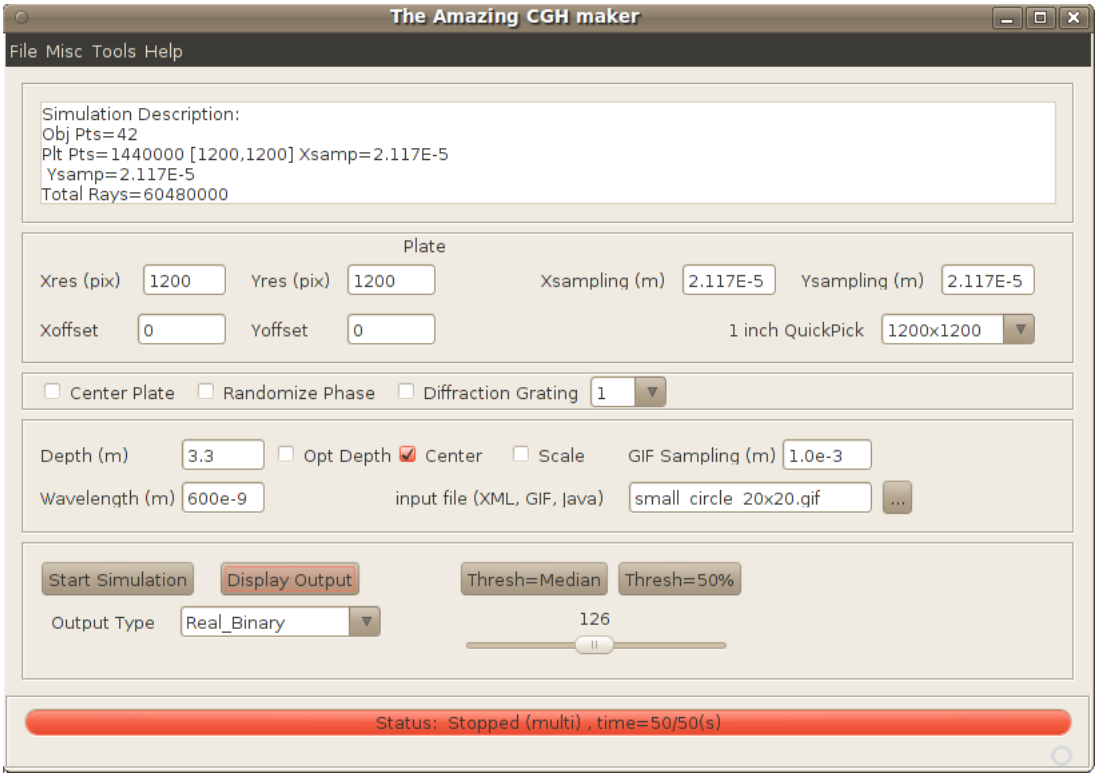


Figure 4.1: Cortical Cafe CGH Maker Application screen from [5].

Parameter	Value		
Pixel Density (pixels per inch)	300x300	600x600	1200x1200
X Sampling Rate (m)	8.467×10^{-5}	4.233×10^{-5}	2.117×10^{-5}
Y Sampling Rate (m)	8.467×10^{-5}	4.233×10^{-5}	2.117×10^{-5}
Randomize Phase	No	No	No
Optimal Viewing Depth Calculation	Yes		
Centered Object in Front of Plate	Yes		
Output Type	Real Image		
Wavelength (nm)	630		

Table 4.1: Parameters for hologram creation.

The resulting real images are in Appendix B using images from Appendix A. The images are split into two groups: The first one being simple images that contain only one shape per image, the second having more than one shape on each image (known as complicated images).

4.1 Holographic Data Analysis

Reducing redundancy is key to compression. Therefore, analyzing redundant patterns that occur in holograms allows a better understanding of inherent factors to effective compression of holograms. Some of the patterns include reducing redundancy of bytes to identifying redundant patterns that may be used for better compression. Since most compression schemes treat the input data as bytes instead of what they fully represent, understanding this type of redundancy assists prediction of the results of certain compression schemes. This section includes analysis of hologram redundancy through quantitative analysis and by visual inspection.

4.1.1 Test to Locate Hologram Redundancy

The first step to understanding holograms are to identify the amount of redundant pieces of data from the input. Thus, a count of repeated data is usually a good first step to understanding how much compression can be performed in the optimal case for pixel redundancy, and how close it should be used to compare with experimental results. The first set of checks that should be done are pixel redundancy checks. For real images or just amplitude or phase file(s), each real or phase/amplitude value (for ease of description) will be considered a pixel. Through pixel redundancy the sets of simple and complicated images from different pixel densities, it would give a good idea of the ideal compressed file size.

The following table is the table of redundant pixels (a repeated data value) for the amplitude and phase being in one file. In this setup, the phase and amplitude are double values.

From the table, it can be seen that more than half the pixels can be removed in the ideal case. Therefore, we can expect a compression ratio close to that value to be near the maximum compression ratio in terms of pixel redundancy. However, that is only for the case of pixel redundancy. There are other methods such as context redundancy that hasn't been counted at all. It seems a bit strange that there are no redundant pixels in complicated images but there is

Table 4.2: Average Redundancy Pixel Percentage for Amplitude and Phase in One File— Phase and Amplitude Each a Double Value

Type of Images	Resolution	Average Redundant Pixel Percentage
Simple	300	43.76
Complicated	300	0
Simple	600	44.07
Complicated	600	0
Simple	1200	44.29
Complicated	1200	0

a simple reason in this setup: from simple inspection, it is apparent that the simple images are mainly regular shapes. Each point on the hologram plane are the sum of the interference of light traveling from each object point to the plane point and the reference beam to the plane point. With the GIF images used being a plain 2D shape, the final complex wavefront over some plane points may contribute to the same amplitude and phase values. However, complicated images are of irregular shape and does not get the same amplitude and phase values when recorded. To see the difference in pixel redundancy due to image complication, observe the 300x300 redundancy percentages on simple images:

Table 4.3: Redundancy Pixel Table for Amplitude and Phase in One Files for Simple 300x300 Resolution— Phase and Amplitude Each a Double Value

Original Image from Appendix	Redundant Pixel Percentage
A.1(a) 1 Point	95.55
A.1(b) Big Circle	36.17
A.1(c) Small Circle	45.8
A.1(d) Letter A	15.48
A.1(e) Letter H	40.89
A.1(f) Square	53.61
A.1(g) Triangle	18.81

From observing the redundancy percentages and the original images from A.1, it can be seen that the single point, circles, letter H, and square contain the highest redundancy in terms of pixels. These are fairly symmetric shapes thus their redundancies are highest. To see the difference between pixel redundancy with combining and separating amplitude and phase, another redundancy count is used on files where amplitude and phase are separated. Since complicated images do not contain a lot of redundancy, simple images are the only ones that are counted.

It can be seen that redundancy in amplitude and phase are almost the same. This means there are multiple chances for good compression in terms of pixel redundancy. The next one is a pixel redundancy count on real images represented as byte values.

It is easy to see that using double precision floating point values to represent amplitude and phase values is not a good way of reducing redundancy. Fortunately, most compression

Table 4.4: Average Redundancy Pixel Table of Different Images for Amplitude and Phase in Different Files for Simple Images— Phase and Amplitude Each a Double Value

Resolution	Amplitude— Average Redundant Pixel Percentage	Phase— Average Redundant Pixel Percentage
300	40.48	47.03
600	40.97	47.17
1200	41.07	47.51

Table 4.5: Average Redundancy Pixel Table for Real Images of Different Resolution and Type for Amplitude and Phase in Different Files— Each Pixel is a Byte

Type of Images	Resolution	Average Redundant Pixel Percentage
Simple	300	0.998
Complicated	300	0.9978
Simple	600	0.9995
Complicated	600	0.9994
Simple	1200	0.9998
Complicated	1200	0.9998

schemes compress data byte by byte as default. Therefore, a redundancy count of the input file at the byte level should be performed. After counting bytes for files of 300x300 real images as well as the amplitude and phase files, it was discovered that all the files contained redundancy around 99.98%. Therefore, it is theoretically possible to reduce the original files by that percentage when compressing. However, it appears strange for all files to contain the same amount of redundancy. To investigate further, a check on the number of redundant bytes is done. From looking at the number of redundant bytes of different files, it seems there is no uniform trend to simple and complicated images. The distribution of redundant bytes for the two different types of images do not look different— the numbers look almost the same. Thus, there is no apparent way of identifying any pattern for byte redundancy.

4.1.2 Looking for Redundancy Through Inspection

Other than looking at repeated pixels, it would be of interest to visually inspect any redundant patterns from the real images, amplitude maps, and phase maps. For local patterns, the dictionary or context-based compression schemes are more than enough to prove redundancy in terms of small repeated patterns. However, there are some patterns that would be of more interest. Since the amplitude and phase maps are similiar looking for different pixel densities, Appendix C contains only amplitude and phase maps for 600x600 pixel per inch images. Through close visual inspection on the real images and amplitude/phase maps from Appendix B and C, it can be seen there are obvious repeated patterns in the real images of the simple-

imaged holograms and of the phase maps of holograms created from all images. For simple images, it is easy to see the real images for them contain obvious symmetric and fractal-like patterns. The symmetric patterns are due to the nature of the original images - they are symmetric. Thus, the resulting amplitude and phase maps are symmetrical as seen in Appendix C.1. Reducing symmetric and fractal patterns would be a good way to reduce redundancy for real images. However, this is not without its difficulties. Fast, automatic, and effective lossy or lossless fractal compression schemes are not yet available which means this method is not very practical.

Another observation would be the phase maps: It is obvious that they look like the original image. This is due to the nature of phase calculation and the object type (it is a flat 2D object). Since each phase point is mainly calculated as the sum of distances from the hologram plane point to all the object points that interferes with the reference beam, it results in a similiar looking image of the original image. With the original image being smaller in terms of pixel count than the resulting resolution of the hologram, finding a mapping between the original object points and the resulting points on the phase map may help reduce the file size given the phase map's resolution being significantly higher than the original image's size. However, this method of redundancy reduction would be very complicated since this method would involve a lot of checks to find the mappings.

4.2 Summary

From the previous analysis, it is apparent that there are optimally many ways to reduce redundancy in a file and therefore achieve compression. However, given all the possible forms of redundancy, it is not always easy to reduce redundancy based on these observations. Many factors such as the distribution of pixels and difficulty in reduction time would cause solutions to be impractical. Even though the analysis does not provide a straightforward solution to reducing redundancy, it would be a good guideline to match against compression schemes used for compression. As well, it would be a good idea to keep these factors in mind for the designing of new algorithms to compress holograms.

Chapter 5

Experiments

This chapter shows the experiments on compressing complex wavefront, and amplitude/phase files. Simple observations from the results are done after each test. A detailed analysis is done in the next chapter.

5.1 General Setup

The amplitude and phase values are outputted into two sets of files with amplitude and phase each being a 64-bit double precision floating point value: The first being amplitude and phase being entered into the same file, while the second one is with amplitude and phase separated into separate files. This will allow the evaluation of the effects of real images, amplitudes, and phases of a hologram in hologram construction. For the real images, the GIF images will then be transferred into a raw Portable Grayscale Map (pgm) image. The image resolution of the images used to create the holograms are 30×30.

The test images are mainly split into two main sets and three subsets. The first main section are simple images— images that are from the CGH Maker website's sample files. These are mainly images where one object is drawn so theoretically the resulting image should be evenly distributed and not as complicated. The second set are images drawn with more than one object that ranges from concentric circles to uneven shapes where the amplitude and phase distribution would not be as even as simpler images. The subsets are split by pixel density in pixels per inch: 300x300, 600x600, 1200x1200. This is used to test the effectiveness of compression on various-sized holograms. The images used to create the holograms are shown in Appendices A.1, A.3, and A.2.

5.1.1 System Setup

The following are the system specifications for the machine used to create the hologram:

- Machine: Sun Virtual Box Version 3.1.6r59338
- Operating System: Ubuntu 10.04
- Processor: One of two cores on an Intel Core 2 Duo CPU T7500 at 2.2GHz

- RAM: 256MB

5.1.2 Compression Scheme Setups

The holograms are all first compressed using their respective default settings. The compression schemes used are Arithmetic Encoding, Huffman Encoding, Burrows-Wheeler Compression Scheme (noted as BWT or BWCS), Prediction by Partial Matching (PPM), LZSS, and LZW. The source code for Arithmetic Encoding, Huffman Encoding, LZSS, and LZW are from Michael Dipperstein's webpage available on the internet [13]. The PPM code used is from Moffat's implementation of PPM [23]. The BWT code is a modified version of Mark Nelson's code [26]. The modified version can compress input using various block sizes. For BWCS, the default setting is a block size of 200000 bytes. An extra run-length encoder is also done before starting the Burrows-Wheeler Transform step to increase compressibility. For LZSS, the default setting is for a window size of 4096 bytes, a 512 byte max match length, and a maximum uncoded character size of 4 bits. LZW is set with a maximum dictionary size of default size 8 megabytes. For PPM, both the PPMc and PPMd schemes are used as a first experiment. The default is a context level of 4 with a maximum space for contexts as 8 megabytes. More PPM schemes or dictionary sizes will be experimented if the first ones prove effective in compression.

5.1.3 Criteria

Compression options for further experiments will be adjusted as more results shed light to the effectiveness of each scheme. The effectiveness of each compression scheme contains three main factors: Compression ratio, compression time, and decompression time. The compression ratio is:

$$CompressionRatio = \frac{CompressedFileSize}{UncompressedFileSize}$$

Therefore, an image with a compression ratio of 0.5 means that the image was compressed to half its original size. The compression time is measured in milliseconds. The criteria for a reliable compression scheme is as follows: the compression scheme must first achieve compression within a reasonable amount of time. The effectiveness of compression should be evaluated by the compression ratio and the time it takes to compress since a file that takes too long to compress for a near non-compression ratio is unrealistic for use.

5.2 Experiments

5.2.1 Experiment 1— First Experiment with Real Images

The first experiment was a simple compression test for real images using only a few images from the CGH Maker website (the images are in appendix A.1 (b), (d)-(f), and A.3. The pixel densities for the images are 50x50, 300x300, and 600x600 pixels per inch and the ratios are averaged together just as a first experiment on compression ratios. Table 5.1 are the results.

Table 5.1: Compression ratios of various compression schemes created with holograms of various images in figure A.1.

Compression Scheme	Average Compression Ratio
Arithmetic	0.79
BWT	0.68
Huffman	0.82
LZSS	0.91
PPMC1	0.78
PPMC2	0.72
PPMC3	0.70
PPMC4	0.67
PPMD1	0.67
PPMD2	0.75
PPMD3	0.69
PPMD4	0.68

The results from the first experiment indicates that real images work best with BWT and PPM. As PGM images, their values only vary from 0 to 255. From the previous chapter, it was discovered that the distribution of pixel values from different files are almost the same. Therefore, it is apparent that Huffman and Arithmetic Encoding have limited capabilities. Both forms of encoding requires a certain skewness in terms of pixel distribution through a small set of data for compression to be effective. Therefore, it can be seen that compression of the real images would not be close to optimal. LZSS is weak due to a possible reason: the variations in values involved in the local sliding windows may be so great that creating a dictionary offers no major redundancy reduction. Even though it seems like the same argument can be used for PPM, the difference between PPM and LZ algorithms is the use of the Arithmetic Encoder for PPM schemes. Thus, a usual reduction in Arithmetic Encoding could mean a possible reduction for PPM schemes (the BWT code used in this thesis also use Arithmetic Encoding as the entropy encoder). The BWT seems to be at an advantage in this setup since it compresses data by blocks and employs a number of reduction methods such as Arithmetic Encoding, Run-length Encoding, and Move-to-Front Encoding after the use of the Burrows-Wheeler Transform. This is because the Burrows-Wheeler method is the only method that transforms the input data into something compressible through reordering input.

5.2.2 Experiment 2— Full Test with Real Images

The second set of tests uses all the holograms built from A.1 and A.3. These simple images all share one simple similarity: they all contain one simple shape. Therefore, the built holograms all have a symmetric structure. A custom set of more complicated images (images having more than one shape on the image) are created from A.2 for hologram compression testing. From

the first experiment, it is apparent that BWT and PPM are suitable candidates for compressing holograms. Therefore, further options for the two schemes are performed. In this setup, various BWT block sizes are tested while different PPM context levels are tested with a context space limit of 8 megabytes. Also, LZW is added to the experiment. The file sizes for each pixel density are: 88Kb for 300x300 images, 352Kb for 600x600 images, and 1.4Mb for 1200x1200 images. The BWT block sizes are around 13%, 19%, 25%, 50%, 75%, and 100% of the image size. These values are big enough such that there are enough related data in each block to allow for effective compression ratios. Tables 5.2, 5.3, and 5.4 are the results of compression ratio, compression speed, and decompression time of the real images. Note that empty entries are for block sizes that are too small to be considered in the test.

Table 5.2: Average compression ratios for various compression schemes on real images of holograms created with A.1, A.3, and A.2. Resolutions are in pixels per inch (ppi).

Compression Scheme	**Blocksize/ Context Order**	**Compression Ratio of Simple Images**			**Compression Ratio of Complicated Images**		
		300x300ppi	***600x600ppi***	***1200x1200ppi***	***300x300ppi***	***600x600ppi***	***1200x1200ppi***
Arithmetic		0.85	0.84	0.84	0.89	0.88	0.88
Huffman		0.85	0.85	0.84	0.90	0.89	0.89
LZSS		1.07	1.09	1.09	1.12	1.12	1.12
LZW		0.98	0.97	0.94	1.12	1.09	1.07
BWT	Default: 200000	0.63	0.80	0.80	0.93	0.93	0.92
	11700	0.87			0.94		
	17100	0.85			0.94		
	23500	0.86			0.93		
	46000	0.84	0.85		0.93	0.93	
	68500	0.75	0.82		0.93	0.93	
	91000	0.63	0.83		0.93	0.93	
	181000		0.81	0.82		0.93	0.92
	270000		0.73	0.79		0.93	0.92
	361000		0.62	0.80		0.92	0.92
	721000			0.79			0.92
	1081000			0.71			0.92
	1441000			0.60			0.92
PPM	C1	0.91	0.92	0.92	0.97	0.96	0.96
	C2	0.76	0.80	0.86	1.04	1.02	1.01
	C3	0.66	0.66	0.86	1.05	1.04	1.04
	C4	0.65	0.84	0.88	1.05	1.04	1.04
	C6	0.65	0.87	0.90	1.04	1.04	1.04
	C7	0.65	0.83	0.92	1.04	1.04	1.04
	D1	0.94	0.95	0.95	1.00	0.99	0.98
	D2	0.78	0.83	0.89	1.08	1.05	1.04
	D3	0.68	0.68	0.88	1.08	1.07	1.07
	D4	0.67	0.86	0.90	1.08	1.08	1.07
	D6	0.67	0.90	0.93	1.08	1.08	1.07
	D7	0.67	0.86	0.95	1.08	1.08	1.07

Table 5.3: Average compression time for various compression schemes corresponding to Table 5.2.

Compression Scheme	**Blocksize/ Context Order**	**Compression Time of Simple Images (in milliseconds)**			**Compression Time of Complicated Images (in milliseconds)**		
		300x300ppi	***600x600ppi***	***1200x1200ppi***	***300x300ppi***	***600x600ppi***	***1200x1200ppi***
Arithmetic		80	283	1034	80	284	1100
Huffman		50	161	637	46	191	639
LZSS		2048	9319	33969	3664	9350	37224
LZW		89	459	3176	90	474	2744
BWT	Default: 200000	394	1481	5959	403	1626	6416
	11700	373			390		
	17100	381			486		
	23500	380			409		
	46000	391	1544		430	1744	
	68500	439	1610		433	1687	
	91000	447	1527		436	1627	
	181000		1846	7083		1859	6887
	270000		2061	7791		1751	6934
	361000		2463	7214		1901	7667
	721000			6599			7003
	1081000			7800			6661
	1441000			9831			6727
PPM	C1	130	473	1654	177	581	2097
	C2	213	1221	6847	480	3340	14194
	C3	293	1783	8711	624	3289	12973
	C4	369	2047	9153	770	3053	12344
	C6	409	1961	7041	1063	3551	13587
	C7	443	1886	7767	810	3046	11264
	D1	129	479	1880	186	624	2230
	D2	314	2170	13417	491	3616	15116
	D3	266	1591	8293	556	2809	10810
	D4	459	2501	9536	899	3586	13891
	D6	473	2399	9044	1034	3451	13294
	D7	531	2357	9411	1473	5584	19734

Table 5.4: Average decompression time for various compression schemes corresponding to Table 5.2.

Compression Scheme	**Blocksize/ Context Order**	**Decompression Time of Simple Images (in milliseconds)**			**Decompression Time of Complicated Images (in milliseconds)**		
		300x300ppi	***600x600ppi***	***1200x1200ppi***	***300x300ppi***	***600x600ppi***	***1200x1200ppi***
Arithmetic		44	171	683	51	180	713
Huffman		23	93	363	24	176	391
LZSS		16	53	210	13	60	211
LZW		24	84	319	26	81	347
BWT	Default: 200000	257	1587	3026	516	2729	3213
	11700	320			269		
	17100	297			246		
	23500	233			263		
	46000	194	917		329	887	
	68500	297	719		487	976	
	91000	377	772		453	926	
	181000		1881	2763		1661	3316
	270000		2010	2927		2816	3287
	361000		927	2921		1697	3486
	721000			3889			4599
	1081000			3030			3919
	1441000			3779			5229
PPM	C1	154	487	1689	217	631	2097
	C2	247	1400	7819	480	2850	11743
	C3	306	1623	7931	651	3149	12040
	C4	353	1934	7681	713	3040	11401
	C6	353	1934	7681	713	3040	11401
	C7	446	1921	7286	850	2840	10313
	D1	146	477	1799	219	660	2287
	D2	246	1407	7893	501	2849	11783
	D3	296	1560	8104	600	3020	11946
	D4	356	2019	8059	701	3009	11324
	D6	436	2040	7346	949	3014	11239
	D7	451	1891	7637	820	2824	10806

From this setup, it is easy to see that effective compression ratios still belong to BWT and PPM schemes. For the other schemes, their compression and decompression time are very fast compared to BWT and PPM— most of them can finish within a second. However, their compression ratio pales in front of higher level BWT block sizes and context levels higher than 2 for PPM schemes. It is apparent that dictionary-based schemes such as the Limpel-Ziv schemes does not work well since the variability of data is too great to keep the dictionary size small. Looking at the compression speed in simple images for BWT, a 10% or more improvement in compression ratio, it will take 7 times or more compression time versus Arithmetic or Huffman. The time it takes to compress the image becomes 9 or more times for a 20% improvement in compression ratio, which is also the maximal achievable compression ratio. The same case goes for decompression time. Looking at the compression ratio and time for smaller pixel density, using a higher PPM context level or BWT block size would still perform well against Arithmetic or Huffman schemes. Even though it is more than 5 times slower, it still performs within 1 or 2 seconds which can be easily compensated by the faster computers than the ones used in this setup and by parallel architectures.

For complicated images, however, the compression ratio situation is totally different. Even though none of the images achieved good compression, Huffman and Arithmetic Encoding became better choices than BWT and PPM. As seen from the previous chapter, byte redundancy is prominent amongst all types of real images. However, byte redundancy does not equate to good compression. Low compression rates from dictionary-based or context-based compression schemes suggest that there are not enough repeated patterns in local input for that reduction in byte or local/block context redundancy to work well. Since the real image is the real value created from the combination of amplitude and phases, it is easy to see that a random distribution in local data would equate in lower compression for real images. Therefore, the better method would be to reduce the number of bits used to represent each pixel— the reason why Arithmetic and Huffman Encoding wins overall in this scenario.

5.2.3 Experiment 3— Test with Amplitude and Phase as Same Files

The next set of tests are done with compressing amplitude and phase together as a single file. The advantage of compressing amplitude and phase instead of real images is the ability to fine tune parameters such as the viewing angle, scaling, and viewable depth. The amplitudes and phases are saved in the order of "amplitude then phase" for each pixel. The size of the files for different pixel densities are: 1.4Mb for 300x300, 5.6Mb for 600x600, and 22.5Mb for 1200x1200 images. The BWT block sizes percentage compared to the original file size for 300x300 files are 14%(default), 25%, 50%, 75%, 100%, 75%, and 100%, which should allow effective compression ratios. For 600x600, the upper limit for block size is capped at around 71% since the block size becomes too large to for an effective compression time (more than 6 minutes per file). The block sizes for 600x600 are 3.5%, 6%, 13%, 19%, 36%, 46%, 57%, and 71%. For 1200x1200, the upper limit becomes 17.7% and the block sizes decrease in mainly 3% intervals for smaller block sizes. PPM settings are 8Mb context space limit with different orders of PPMc and PPMd just like the previous experiment on real images. Tables 5.5, 5.6, and 5.7 are the results of compression ratio, compression speed, and decompression time of the amplitude and phase in one file:

NOTE: The 1200x1200 pixel density compression data for LZW are not available since it breaks the maximum dictionary size limit of 8 megabytes.

Table 5.5: Average compression ratios for various compression schemes on the amplitudes and phases of holograms created with A.1, A.3, and A.2. The amplitude and phase are stored in one single file. Resolutions are in pixels per inch (ppi).

Compression Scheme	**Blocksize/ Context Order**	**Compression Ratio of Simple Images**			**Compression Ratio of Complex Images**		
		300x300ppi	***600x600ppi***	***1200x1200ppi***	***300x300ppi***	***600x600ppi***	***1200x1200ppi***
Arithmetic		0.92	0.92	0.92	0.96	0.96	0.96
Huffman		0.93	0.93	0.93	0.97	0.96	0.96
LZSS		0.77	0.90	0.97	1.12	1.12	1.12
LZW		0.88	0.90		1.11	1.13	
BWT	Default: 200000	0.68	0.70	0.72	0.97	0.97	0.97
	361000	0.67	0.70	0.71	0.97	0.97	0.97
	721000	0.64	0.69	0.71	0.97	0.97	0.97
	1081000	0.59	0.67	0.70	0.97	0.97	0.97
	1441000	0.52	0.68	0.70	0.97	0.97	0.97
	2001000	0.51	0.64	0.70	0.97	0.97	0.97
	2600000	0.51	0.65	0.69	0.97	0.97	0.97
	3200000	0.51	0.64	0.69	0.97	0.97	0.97
	4000000	0.51	0.61	0.69	0.97	0.97	0.97
PPM	C1	0.83	0.85	0.88	0.99	0.99	0.99
	C2	0.67	0.74	0.77	1.06	1.06	1.06
	C3	0.66	0.69	0.72	1.06	1.06	1.06
	C4	0.66	0.70	0.73	1.06	1.06	1.06
	C6	0.68	0.71	0.74	1.05	1.05	1.05
	C7	0.68	0.71	0.75	1.05	1.05	1.05
	D1	0.83	0.87	0.90	1.01	1.00	1.00
	D2	0.69	0.77	0.80	1.08	1.08	1.08
	D3	0.67	0.71	0.74	1.09	1.09	1.09
	D4	0.68	0.72	0.75	1.08	1.08	1.08
	D6	0.69	0.72	0.76	1.08	1.08	1.08
	D7	0.69	0.73	0.76	1.08	1.08	1.08

Table 5.6: Average compression time for various compression schemes corresponding to 5.5.

Compression Scheme	**Blocksize/ Context Order**	**Compression Time of Simple Images (in milliseconds)**			**Compression Time of Complex Images (in milliseconds)**		
		300x300ppi	***600x600ppi***	***1200x1200ppi***	***300x300ppi***	***600x600ppi***	***1200x1200ppi***
Arithmetic		847	3301	13229	813	3206	12876
Huffman		320	1381	5753	333	1426	5594
LZSS		21304	103030	445754	32766	131389	535079
LZW		3714	29409		2919	16114	
BWT	Default: 200000	6173	25414	101986	7833	31284	126437
	361000	6636	26254	99284	8134	31660	128007
	721000	6611	26809	103263	12703	32244	129139
	1081000	8240	28300	112823	9526	35809	142236
	1441000	8990	31520	121860	8896	34004	139077
	2001000	8016	27466	113524	8461	33653	132311
	2600000	7963	27299	120463	8496	33967	134616
	3200000	8089	31987	108443	8311	34010	137377
	4000000	8083	38571	129091	9037	37430	148189
PPM	C1	3070	18624	45010	7794	51309	72619
	C2	14499	40490	178360	18307	75204	306580
	C3	9314	40564	172531	18447	74160	294624
	C4	11240	43839	180721	16466	65527	264056
	C6	7633	30670	129626	14287	57203	228444
	C7	6850	29014	122421	14016	54599	214757
	D1	3060	13484	58733	5689	20677	77197
	D2	8023	38449	163007	17296	71261	295987
	D3	9076	38723	165150	17399	70713	284480
	D4	8283	35760	149966	16263	65484	261401
	D6	7574	30846	132176	14589	58163	231619
	D7	6950	28989	124749	13727	54530	219259

Table 5.7: Average decompression time for various compression schemes corresponding to 5.5.

Compression Scheme	**Blocksize/ Context Order**	**Decompression Time of Simple Images (in milliseconds)**			**Decompression Time of Complex Images (in milliseconds)**		
		300x300ppi	***600x600ppi***	***1200x1200ppi***	***300x300ppi***	***600x600ppi***	***1200x1200ppi***
Arithmetic		660	2756	10881	820	2993	11713
Huffman		341	1444	6174	393	1616	6373
LZSS		189	816	3541	193	800	3286
LZW		306	1399		391	2577	6880
BWT	Default: 200000	5371	13984	54177	7032	53336	78027
	361000	3403	13356	54570	8218	19609	78451
	721000	5154	13410	61720	9729	53329	79647
	1081000	3460	22874	60600	8521	32447	85944
	1441000	4716	21990	56889	7236	42694	80577
	2001000	6367	20586	53427	8207	32031	80271
	2600000	4361	36931	53361	9617	31251	82284
	3200000	13490	35103	54177	9719	20171	81937
	4000000	3347	11921	54984	9814	31987	80224
PPM	C1	2373	11153	44846	4481	16697	66046
	C2	10389	48793	208320	22414	93609	372421
	C3	12349	56136	226693	26074	105163	431237
	C4	11181	49224	176364	24523	90280	388441
	C6	11436	48263	187744	21043	85704	341893
	C7	11607	53379	191016	26197	93853	372880
	D1	4530	20133	92286	7764	30646	127087
	D2	11968	56309	252786	27857	112420	450090
	D3	13607	59060	250824	23104	92437	382511
	D4	9064	38604	162181	17797	70433	283094
	D6	8266	34209	143444	17294	69206	268957
	D7	14064	57063	238081	21347	82037	318871

From the tables, it seems that simple images are still best suited for BWT or PPM in terms of compression ratio. Compared to arithmetic encoding, for 20% or more improvement in compression ratio, it takes 8 times or more in terms of compression speed for BWT (10 times for 40% improvement), and 10 times or more for PPM. In terms of compression speed for simple hologram files, BWT outperforms PPM in terms of both compression/decompression speed and ratio. The other compression schemes did not achieve compression at all. With the increase in the amount of varying values of amplitudes and phases, the weakness of arithmetic and Huffman Encoding become apparent. Even a low block size for BWT schemes and low context level for PPM schemes contain lower compression ratios than most other compression schemes in any pixel density. The reason for this can be observed from the amplitudes and phase maps in Appendix C. In simple images, it is easy for the amplitude and phase to be as easily compressible as the real images since they contain a lot of repeated patterns, which attributes to their combination in a file to be easily compressible.

For complicated images, none of them achieved effective compression. As mentioned in the previous experiment with real images, the amplitude and phase maps indicate the data does not contain a stable pattern, which is why the most likely way to reduce compression size is through entropy encoders such as arithmetic and Huffman Encoding - a stage of BWT. That is why only Arithmetic, Huffman, and BWT schemes achieve some compression in complex images.

Overall, it is observed that at PPM context levels greater than 3 in simple images, the compression ratio worsens. This is due to the increase in extra contexts causing the resulting files to lower in terms of compression ratio. For complex images, PPMC1 is the only one that actually results in a reduced average file size. This shows that a context level of 1 is the maximum allowable context level for PPM compression. Since the original input file is an amplitude then phase type of layout for the data, this coincides with the fact that amplitude and phase values for complex images do not contain a lot of repeated patterns in the amplitudes and phases overall.

5.2.4 Experiment 4— Compression with Amplitude and Phase as Separate Files

The next set of tests are done with amplitude and phase as separate files. Since all the input files are split into two, the resulting file sizes are 704Kb for 300x300 files, 2.8Mb for 600x600 files, and 11.2Mb for 1200x1200 files. The upper block size percentage limits for BWT in compression are 40% for 1200x1200, and 100% for the rest. Every other setting are identical to the previous two experiments.

Table 5.8: Average compression ratios for various compression schemes on the amplitudes and phases of holograms created with A.1, A.3, and A.2. The amplitude and phase are stored and compressed in two different files. Resolutions are in pixels per inch (ppi).

Compression Scheme	**Blocksize/ Context Order**	**Compression Ratio of Simple Images**			**Compression Ratio of Complex Images**		
		300x300ppi	***600x600ppi***	***1200x1200ppi***	***300x300ppi***	***600x600ppi***	***1200x1200ppi***
Arithmetic		0.90	0.90	0.90	0.95	0.95	0.95
Huffman		0.91	0.91	0.90	0.96	0.96	0.96
LZSS		0.74	0.78	0.91	1.11	1.11	1.11
LZW		0.87	0.88	0.90	1.11	1.10	1.12
BWT	Default: 200000	0.67	0.70	0.71	0.96	0.96	0.96
	361000	0.65	0.70	0.71	0.96	0.96	0.96
	721000	0.53	0.68	0.71	0.96	0.96	0.96
	1081000	0.53	0.65	0.70	0.96	0.96	0.96
	1441000	0.53	0.66	0.70	0.96	0.96	0.96
	2001000	0.53	0.62	0.70	0.96	0.95	0.96
	2600000	0.53	0.57	0.69	0.96	0.95	0.96
	3200000	0.53	0.54	0.68	0.96	0.95	0.96
	4000000	0.53	0.54	0.66	0.96	0.95	0.95
PPM	C1	0.80	0.82	0.84	0.98	0.97	0.97
	C2	0.59	0.71	0.76	1.04	1.04	1.04
	C3	0.63	0.69	0.72	1.05	1.05	1.05
	C4	0.65	0.70	0.72	1.04	1.04	1.04
	C6	0.67	0.71	0.72	1.04	1.04	1.04
	C7	0.67	0.70	0.73	1.04	1.04	1.04
	D1	0.81	0.83	0.85	1.00	0.99	0.99
	D2	0.60	0.73	0.78	1.06	1.06	1.06
	D3	0.64	0.71	0.74	1.07	1.07	1.07
	D4	0.66	0.72	0.74	1.07	1.07	1.07
	D6	0.68	0.72	0.74	1.06	1.06	1.06
	D7	0.69	0.72	0.74	1.06	1.06	1.06

Table 5.9: Average compression time for various compression schemes corresponding to 5.8.

Compression Scheme	**Blocksize/ Context Order**	**Compression Time of Simple Images (in milliseconds)**			**Compression Time of Complex Images (in milliseconds)**		
		300x300ppi	***600x600ppi***	***1200x1200ppi***	***300x300ppi***	***600x600ppi***	***1200x1200ppi***
Arithmetic		1043	3836	15353	1039	4284	17194
Huffman		347	1433	5557	391	1471	5740
LZSS		26921	115614	546439	36830	143264	586814
LZW		3043	20960	174377	2366	13036	80551
BWT	Default: 200000	6747	28271	111041	8250	32186	128799
	361000	9057	36827	154827	9297	37304	155900
	721000	7686	30054	121534	8499	33671	135494
	1081000	7764	31940	127876	8409	34404	135917
	1441000	7821	31701	129836	8570	34284	137559
	2001000	7703	34860	129606	8489	34821	141529
	2600000	7653	37449	137663	8540	35073	143254
	3200000	7817	39034	137973	8666	36689	149340
	4000000	7659	39021	150510	8404	35576	143360
PPM	C1	2574	10361	42674	4606	16549	63874
	C2	6653	40436	185303	19080	80987	319689
	C3	9307	44261	189089	20327	82794	331091
	C4	9340	40659	171903	18980	76716	305817
	C6	8253	34816	146650	16707	66929	268811
	C7	7753	32470	135686	16360	63883	256081
	D1	2837	12173	50690	5037	18457	71297
	D2	8506	51803	217023	23650	98953	406666
	D3	12399	56946	223966	27470	109067	452776
	D4	12597	53010	239390	22644	96526	436406
	D6	10536	46024	198957	20010	79996	329021
	D7	10000	45389	194483	19516	75463	293049

Table 5.10: Average decompression time for various compression schemes corresponding to 5.8.

Compression Scheme	**Blocksize/ Context Order**	**Decompression Time of Simple Images (in milliseconds)**			**Decompression Time of Complex Images (in milliseconds)**		
		300x300ppi	***600x600ppi***	***1200x1200ppi***	***300x300ppi***	***600x600ppi***	***1200x1200ppi***
Arithmetic		754	3071	11586	783	3257	11953
Huffman		380	1559	6133	510	2909	7520
LZSS		174	646	2814	204	783	3726
LZW		474	1484	6776	439	1156	4589
BWT	Default: 200000	6463	23874	54410	10346	41224	77911
	361000	4709	17020	56050	6134	31234	73313
	721000	4309	15529	65786	7896	40476	84553
	1081000	3629	17340	73271	6376	31937	107803
	1441000	4134	19829	64309	5757	26100	88294
	2001000	3183	17599	59554	8609	29501	85619
	2600000	3730	13079	58239	7427	31023	85857
	3200000	4859	13937	58984	8329	31907	80633
	4000000	4754	16441	62131	9433	33414	85996
PPM	C1	2907	10077	38556	4759	15236	57673
	C2	7026	39844	179970	19363	78666	315177
	C3	9686	44299	185103	20480	80597	322383
	C4	9613	40853	166954	19144	74284	295749
	C6	8691	35917	145536	16966	65900	261116
	C7	8679	34907	147621	16616	63424	263203
	D1	3176	11661	45884	8660	28904	111271
	D2	12024	57006	259794	36310	224060	567943
	D3	14424	65461	273770	31554	111276	439547
	D4	11707	50444	221477	22516	90160	353863
	D6	10753	43036	190891	18330	71991	300691
	D7	11531	37053	151684	18387	75376	293446

For simple images, BWT became the apparent winner out of all compression schemes through separate compression of amplitude and phase values. It can be seen that at the default setting (block size of 200000 or less than 28% of the input file size in all pixel densities), the compression is already comparable to different levels of PPM and bests all other compression schemes. At a block size of 721000 (less than 75% of the original size for 300x300 files, less than 25% for 600x600 files, and less than 6.2% for 1200x1200 files), the block size already outperforms all other compression schemes. As BWT compression block sizes increases, the compression rate goes closer to somewhere between 0.5 and 0.7, suggesting at least a 30% reduction in file sizes. This is due to the separation of the symmetric amplitude and phase values which makes reorganizing data through BWT easier for compression.

For complicated images, it is apparent that only arithmetic, Huffman, and BWT schemes compressed anything. As said before, the phase and amplitude maps are highly variable, it is hard for context and dictionary schemes to work properly for that reason. Therefore, the only way to reduce anything is with bit reduction using entropy encoders.

In terms of performance time, the time it takes to compress an amplitude and phase file is 7 times slower than the arithmetic or Huffman coders. For simple images, this is an ignored factor since the compression has reached or almost reaches 60%— something that is better than most other schemes. The decompression time contains similiar differences. However, it takes more than 10 seconds to compress and decompress. This shows that careful consideration should be done when deciding on using the BWT for hologram compression.

5.2.5 Experiment 5– Compressing Amplitude and Phase File With Lower Bit Sizes Using BWCS

Since the previous experiments showed the Burrows-Wheeler Compression Scheme's effectiveness, another experiment on compressing amplitudes and phases were performed. Given that each amplitude and phase value were represented using 64-bit floating point numbers, this experiment sought to reduce the file size through reducing the number of bits used to represent each amplitude and phase value. This is a two part experiment where compression follows after reducing the file size through shortening the bit length of each value. The idea for the first part is as follows:

1. Find the maximum and minimum value for the amplitude or phase values.

2. Determine the range of values using the maximum and minimum.

3. For each amplitude or phase value, turn it into a fraction of the range (add it by the minimum if the minimum is negative, subtract by minimum otherwise making it a number from zero to the range value)

4. Multiply the number by the maximum value from a given bit size. For example, the maximum for 4 bits would be 2^4 or 16.

This way of mapping the values to a smaller range reduces the file size by more than 70 percent. Viewing tests showed that holograms were still viewable up until the bit length was 2 bits or less. Degradation started at around 5 bits. The reason behind the holograms still

being viewable is due to the resulting real image, where 8-bits are used to represent each value. Therefore, shortening the bits used for amplitude and phase does not affect physical viewing on a transparency as much. The second part of this experiment involved using the BWCS to compress the hologram. Two images, specifically, the square image shown in Appendix A.1(f) and A.2(d) were used. Since the reduced file sizes were small, the block size for compression was exactly the size of the size-reduced file.

From Figures 5.1 and 5.2, we see that the size of the resulting file is primarily determined by the precision used for the phase and intensity information. For example, at 16 bits the ration is about 25%. This is what is to be expected by retaining 16 out of 64 bits (16/64 = 0.25). Likewise at 3 bits a ratio of about 5% is obtained, which is again what would be expected (3/64 = 0.047). Note, however that meaningful additional compression of the amplitude is obtained for bit lengths less than 9.

It is apparent that bit reduction does reduce file size from Figures 5.1 and 5.2. However, it is easy to see that for the square image, compression can reduce as far as to 15% of the original file size. The complicated flower image did not lower in size after compression until it was lower than 10 bits, which is due to the reduced range of values available.

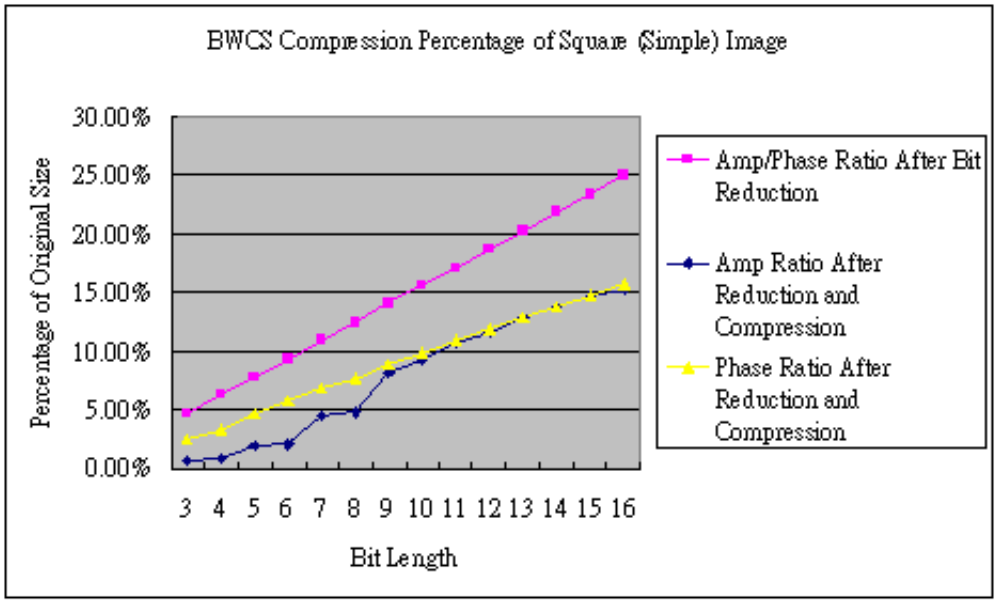


Figure 5.1: Graph comparing the percentage reduced from the original using bit-length reduction before and after compression for a simple image.

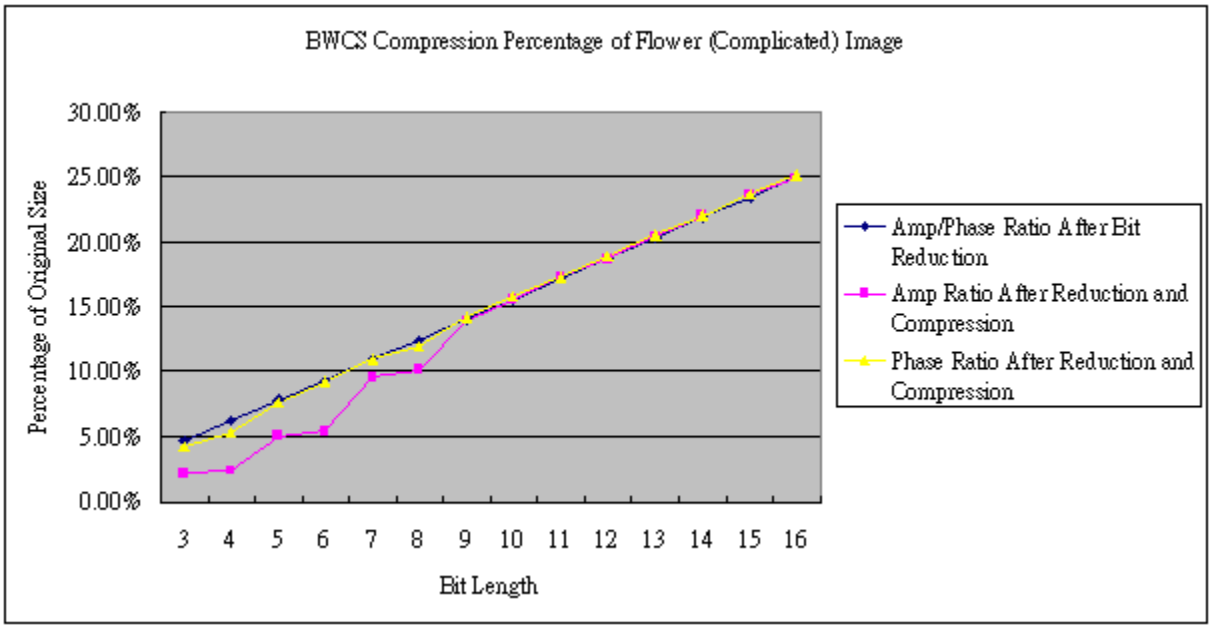


Figure 5.2: Graph comparing the percentage reduced from the original using bit-length reduction before and after compression for a complicated image.

5.3 Summary

By skimming the results, it appears that context-based compression such as PPM and BWT are mainly suitable for symmetric images that contain a stable pattern. For more complex images where different shapes are involved, it is easy to see that there is no definite trend for context finding algorithms to work. Therefore, byte value re-representation algorithms such as arithmetic and Huffman encoding are able to compress the data. However, the compression ratio is fairly weak due to the vast variation in values to re-represent. Thus, it shows that for complex scenes, compression is hard to achieve using modern algorithms.

The solution to better compression ratios is probably lowering precision on the amplitude and phase values. By mapping the 2^{64} amplitude and/or phase values into a smaller range, different values can be represented by a smaller set of values and make the holograms much more compressible. The possible size reduction of this is already shown in the last experiment. However, since this causes degradations on image quality and this thesis is focused on lossless hologram compression, further investigation is left as future work. The next chapter will go into a more detailed analysis of the advantages and disadvantages of discovered through the experiments and insight to factors that can possibly create a better compression scheme.

Chapter 6

Analysis of Results

From the results in the previous chapters, there are a lot of indication that using modern textual compression schemes on anything more than simple shapes will cause compression to fail due to the highly varying nature of hologram data. However, the results did confirm a few insights into hologram compression. The follow chapter will discuss those discoveries.

6.1 Compression Scheme Analysis

It can be easily noticed that simple holograms are more compressible than complicated holograms. For this reason, this section is split into two components: Simple and complicated holograms to show the different factors that affect compression in different types of holograms. There are more analysis done for simple holograms for each different compression scheme due to their compressibility.

6.1.1 Simple Images - Entropy Coders

Simple images are very easily compressible due to the highly symmetric structure that they contain. This means that it can make use of its pixel redundancy structure as shown in Chapter 4. Entropy encoders such as Huffman encoding and arithmetic encoding performed well due to the shortening of bits to represent each value. Arithmetic encoding in particular, outperforms Huffman encoding within 1% through the reduction of the whole data stream into a number for all cases. This shows the value of using arithmetic encoding as the more nominal bit reduction scheme in other compression schemes such as the Burrows-Wheeler Compression Scheme in this study.

6.1.2 Simple Images - Dictionary Coders

As for dictionary encoding, it can be easily seen from the results that compression fails greatly for real images. This is a problem that involves the nature of Limpel-Ziv schemes and the nature of simple holographic images. A real image is basically built by a combination of amplitudes and phases. As such, if either the phase or amplitude files contain little redundant patterns, the final image would look fairly hard for compression. However, studying the real,

amplitude, and phase images of simple holograms, it can be easily determined that all simple holograms contain a few things in common: Their amplitude and phase maps, as well as their real images, are mostly symmetric and contains fractal looking patterns. Therefore, the idea of any of the resulting holograms having few redundant patterns is not totally correct. Yet, it is the same nature of these simple holograms and of Limpel-Ziv schemes that affects compression in real images. To understand the phenomenon, one must look at the nature of Limpel-Ziv compression.

LZ schemes are based on textual compression - the input is processed row by row from left to right. As explained in Chapter 1, a sliding window is used to encase a set of input for encoding and repetition searching– it relies on pattern redundancy in local data. In this setup, characters (bytes) will always be processed from left to right. This means that symmetric images will not work for a few reasons:

1. If the image is symmetric from left to right (symmetric about y), the dictionary will have to add in the extra information from the symmetric right side character by character, since the data is not oriented in the same way as the left side (except for repeated characters).

2. If the image is symmetric from top to bottom (symmetric about x), the left to right pattern reading will not detect such a pattern.

3. If the image contains repeated patterns in rows due to symmetric properties in a very far away row, the sliding window may not catch such a repetition and re-encode the data– there might be no way to increase the sliding window size without compromising compression speed unless it uses a LZ78-like dictionary instead of search buffers.

For these reasons, the possibility of compression with LZ encoding schemes becomes very difficult for two-dimensional data. Outside of symmetry, non-redundant variations in data also causes problems for compression. In terms of LZW schemes, the non-redundant variations create a large dictionary that can cause poor compression ratios. However, looking at the compression ratios for the amplitude and phase files, around 20% of the original image size has been shaved off in the compressed files. This shows that compressing amplitude and phase files contain an advantage over real image compression in dictionary schemes. This can be quickly determined through inspection of the amplitude and phase maps from Appendix C. It can be seen that most amplitude and phase maps for simple holograms are symmetric about both x and y. The above reasons states that symmetric information is mostly ignored if the input is symmetric about y only or if the sliding windows is not enough to detect such a pattern. From looking at the amplitudes and phase maps, it is easily seen that the amplitude maps for simple holograms are mostly symmetric multiple times about y, which means it is easy for a large enough sliding window to compress some of the repeated patterns in each row. This is the main reason that contributes to better compression in smaller images (due to the smaller size). However, the amplitude map indicates that these holograms are symmetric about x multiple times, which means with a large enough sliding window and with it being symmetric about y as well, there are more repeated patterns that can be recorded. With a phase map, the reason is the same since most of the shapes are regular shapes.

Even though the amplitude and phase images are more compressible due to their symmetric properties, the combination of the two into a single real image causes irregularities that either

stretches or destroys repeated patterns if the image is not of an even x- and y-symmetric shape (an A for example), not allowing detection of the repeated pattern. In this case, the resulting real image suffers from irregularities and reduces compression ratio.

With a 3 to 7 times slower compression time than arithmetic encoding, a similiar compression/decompression time with Huffman, and being half the decompression speed of arithmetic encoding, LZW is better than entropy encoders and other dictionary coders in terms of a balance of compression ratio and speed. Since LZW is based on pointers in a dictionary, the compression time takes a bit longer when searching the dictionary for the longest match while decompression does not require searching. Compared to arithmetic encoding, the need to restore input using the same encoding approach is not required so it is slower in terms of decompression. However, no further tests are done with LZW since the compression ratio is poor as the pixel density gets higher and increases the dictionary storage size required for compressed files.

LZSS, on the other hand, is better than LZW, arithmetic, and Huffman in terms of decompression speed due to immediate O(n) speed recovery of data without the use of looking for data in a pointer-filled dictionary as in LZW. This is done through quickly referencing repeated values from a search buffer of pre-determined size. However, the compression time suffers the most out of all experimented compression schemes since it needs to start searching again at the beginning of the window each time a mismatch occurs. With row data varying greatly as in holograms, even small window sizes would cause such a decrease in compression size. However, no more tests are done with LZSS for two reasons: 1) the compression time in the default setting is too slow; and 2) the default window size only encompasses one row in 300x300 holograms, which means smaller window sizes would contribute to even lower compression ratios as the hologram pixel density becomes higher.

6.1.3 Simple Images - PPM

For PPM, the structure of how PPM searches for repeated patterns is similiar to LZW where a dictionary of contexts is stored. The difference however, is there is a limit on how many previous characters can be stored as a context. PPM takes advantage of both the capabilities of an entropy coder and a dictionary encoding scheme through using repeated contexts as frequencies for the entropy encoder. However, the weakness of dictionary schemes is still available— it still does not make use of the full symmetric properties of the holograms. Furthermore, using context tables replaces a search for long repetitions, changing the data into a match of short matching bytes instead. On the bright side, the transforming of long length matches into context tables has the added advantage of keeping the repeated patterns logged in a context table for much later rows to match— causing a reduction in redundant dictionary matches as in LZ schemes. The compression ratio is further reduced with the use of an entropy coder to shorten the input stream into a single number.

Selecting the right context size is important as the increase in context increases storage size for larger context tables. Since the probabilities of the contexts must be entered with the resulting number that represents the input stream, an increase in context size will also increase the storage size required for inputting contexts and their corresponding probabilities. As can be seen from the tables in the previous chapter, the larger the pixel density the weaker the compression ratio becomes. This is due to the increase in finer detail of the hologram resulting in an

increase in the amount of contexts. From the tables, it can be seen that for 300x300 holograms, PPMC6 is best suited for compression. However, this context setting is reduced to PPMC3 in 600x600 holograms, a 3 level reduction in context. For 1200x1200 pixels per inch holograms, the context level is best at around 2 for PPMC. For PPMD, the context level is usually best around context level 3 or 4.

Evidently, the increase in pixel density does not affect PPMD as much since PPMD attempts to evenly increase the skewness of each new symbol entered by starting them off as the ratio of the number of unique symbols to the total number of symbols. This starts off the new symbol contexts at a semi-skewed point and possibly results in less bits used to encode the output string. PPMC, whose probability is built on the dependence of the count of each unique symbol, does not provide a dependent start-off metric for increasing the skewness of the input data, which is the reason why PPMD is more stable in terms of keeping the best context level in higher pixel densities. PPMC, on the other hand, wins over PPMD in these experiments for compression ratios due the lack of a stable start-off distribution in holograms. In the case of simple holograms, since the data is mostly symmetric and input distribution is fairly skewed, PPMC shows a better compression ratio.

With the addition of an entropy encoder and a definite limit to context size, the search time is mainly bottlenecked by context table searching instead of input buffer searches. This ignores the long wait times from LZ algorithms where large window and file sizes can cause the compression time to increase rapidly. However, large contexts should be avoided to reduce context table match search time. Another item of concern would be the decompression time versus the compression time. The decompression time is very close to the compression time in all pixel densities since the same process to encode the input is used for decoding. For all other compression schemes, their decompression times are around or less than half their compression times. Some of this is a result of increased context which increases search time and encoded string length, PPM becomes an unfavorable trade between compression ratio and time versus other compression schemes.

6.1.4 Simple Images - BWT

Different from the previously mentioned compression schemes, the Burrows-Wheeler Compression Scheme (noted as BWCS or BWT), incorporates a variety of coding algorithms in order to create a more compressible scheme. This scheme includes the use of a move-to-front encoder, a run-length encoder, and then an arithmetic encoder in the same order to perform compression. However, this is all preceded by the use of a Burrows-Wheeler Transform on the input data to transform the data into a more compressible format for the next three steps. The Burrows-Wheeler Transform ignores the symmetric properties of the input data by seeking to increase local redundancy through lexographically reordering input symbols. Through lexographic reordering, the likelihood of the same symbols being beside each other are increased, thus increasing the likelihood of move-to-front and run-length encoding schemes to reduce the file size. The input is further reduced by reducing the bits through an entropy coder (in this case an arithmetic encoder). Thus, it is easy to deduce that the compression ratio of this compression scheme is better than other ones for simple holograms.

As seen in the compression ratio tables, the hologram size can be compressed to 50% or 60% of its original size when the whole file is considered as one single block. Comparing with

itself, using the whole block as the block size is also not a bad choice seeing that increasing the block size from 13% to 100% of the original file size only increases the compression time by less than 16%. The decompression time for the full block is even faster than smaller blocks for smaller pixel density files since it eliminates the need to transform encoded data to the original form block by block when memory can still handle larger blocks. For real images, Burrows-Wheeler decompression time is almost the same across different block sizes for 300x300 and 600x600 holograms while it is still very close (less than 40% increase in time) for 1200x1200 hologram. This is mainly due to the need to allocate a larger block of memory, in which case the smaller real images would suffer in this aspect. At the same time, using larger block sizes to compress/decompress amplitude and phase files ignores this problem by allowing more data to be processed at a time in turn increasing the performance time.

For amplitude and phase files, the separation of amplitude and phase files allowed better compression due to the allowance of the more symmetric amplitude and phase data to be compressed separately— assisting in putting same symbols together during sorting. The real image, which decreased in compressibility compared to the amplitude and phase files due to the decreased symmetry from merging the amplitude and phase, is also not too far from the amplitude and phase ratio (only 10% away) due to the transformation of data. However, it can still be more compressible if preprocessing was done to make use of the symmetric properties of simple holograms. Since it is mostly symmetric about x and y, the image can be first folded in the x or y direction before becoming compressed by BWCS. Currently, the best BWCS can do is make the image half its size. However, it can be even smaller for simple holograms due to folding. This will be more discussed in detail in later sections.

Comparing with other compression schemes, PPM is the closest contender with BWT in terms of compression ratio. In terms of compression size, at best PPM is around 10% weaker than BWT for smaller holograms. For 1200x1200 holograms, that gap closes even further for PPM to be 5% to 7% weaker in amplitude/phase file compressions. For real images, however, the reordering nature of BWT keeps the compression ratio near 60% for all pixel densities while PPM weakens by being 25% weaker than PPM at its best. This is due to the increase in contexts as discussed from the previous analysis on PPM. BWT, on the other hand, does not weaken in pixel density size if the block size is kept at the same percentage since it reorders and transforms data, and creates redundancies instead of relying on raw input redundancy. This is one of the strongest advantages that BWT have over other compression schemes. Against other compression schemes, the Burrows-Wheeler scheme is better being 20% to 40% more compressible across different pixel densities. As for performance speed, the compression time is about the same as PPM. The decompression time, on the other hand, shows that BWT is around half or less the speed of PPM except for PPM at context level 1. Therefore, PPM is still weaker in terms of compression ratio and decompression time.

However, being the winner over other lossless compression schemes is not without its disadvantages. Being a compression scheme that incorporates four different steps (five with the extra run-length encoder before the first four in this study), the compression time and decompression time increases much more than other compression schemes. Entropy coder compression comparison will be ignored since Burrows-Wheeler schemes include an entropy coder as one of the steps. For LZW, the compression time is better than BWT by 2 to 5 times across all pixel densities. For decompression, LZW is better by around 10 times. Against the 20% to 40% compression size improvement, this is a minimal problem seeing the resulting output is

still within a few seconds. Putting this in a faster machine and/or allowing BWT to adapt to parallel algorithms, the BWT schemes will run even faster and make the problem turn into a matter that is smaller than milliseconds.

6.1.5 Complicated Images

The complex patterns in complicated images do not contain a lot of redundant patterns. It was observed that bit reduction schemes were the only possible way to make irregular patterned files have any kind of compressibility outside of compressing the real images. The reason for this is mainly attributed to the fact that these are flat images— the phase and amplitude for each point on the object would be calculated from the same depth. Since the calculation of each point on the hologram plane are related to the distance between the object and reference waves, irregular shapes will create a less redundant distribution of amplitude and phase values. This increases variation in data and reduces compression ratios.

Three-dimensional objects, on the other hand, can create more repeated patterns for compression. Unless they are at the wrong angle (such as a flat side directly facing the holographic plate) or they are specially uneven objects, most three-dimensional objects in the real world or generated objects used in holography contain at least minor symmetric or repetitious properties through certain curve or evenly distributed surfaces. For example, a regular cable modem can contain different shapes (spheres for LED lights, a rectangular prism for the actual box, ringed-cyclinders for cable jack). If recorded at a slightly diagonal angle from the front, some of the amplitude/phase values can be repeated through the even distribution of depth in what would seem like an irregular 4-sided polygon in the 2D case. Other things such as the spherical LED lights or the reflective cable jack may also increase or decrease the likelihood of better compression. Therefore, for more complex scenes, it is better to use three-dimensional objects if the object of hologram creation is not a fairly regular shape or contains a lot of other shapes.

In terms of compression time, the entropy coders and BWCS performed just like their simple hologram compression counterparts. LZSS and PPM schemes suffered greatly from searching for matches and contexts, which made their run time a lot longer than simple holograms. Without the need to add a lot of new items to the dictionary, LZW was even faster than the compression of simple holograms. For compression time, PPM was the only one to suffer due to the need to restore the original data through the now large context sets. This shows that using heavy local data reliant schemes such as LZ and PPM schemes are not very valuable in terms of hologram compression.

6.2 Comparison of Compression Schemes with PSI Hologram Compression

Looking at the experimental results, a comparison with PSI compression schemes would be a good indication as to where virtually-recorded holograms stands in terms of lossless compression. Figure 6.1 contains the amplitude reconstruction of the objects used for compression. Tables 6.1 and 6.2 are compression ratios on PSI holograms from [20]. The tests were done on 2048x2048 pixels CCD cameras with a file size of 64Mb. The compression rates from [20] are

Table 6.1: Compression ratios of 5 holograms from [20] using amplitude and phase as one file.

Hologram No.	1	2	3	4	5	Average
Huffman	0.96	0.96	0.95	0.96	0.96	0.96
LZ77	0.81	0.96	0.51	0.85	0.83	0.79
LZW	1.00	1.00	0.85	1.00	1.00	0.97
BWT	0.57	0.98	0.28	0.63	0.58	0.61

transferred into compression ratio for better comparison with the data from the experiments. Since the explanation of why the complicated holograms created are not compressible was explained in the previous section, this section focuses on comparison with simple holograms.

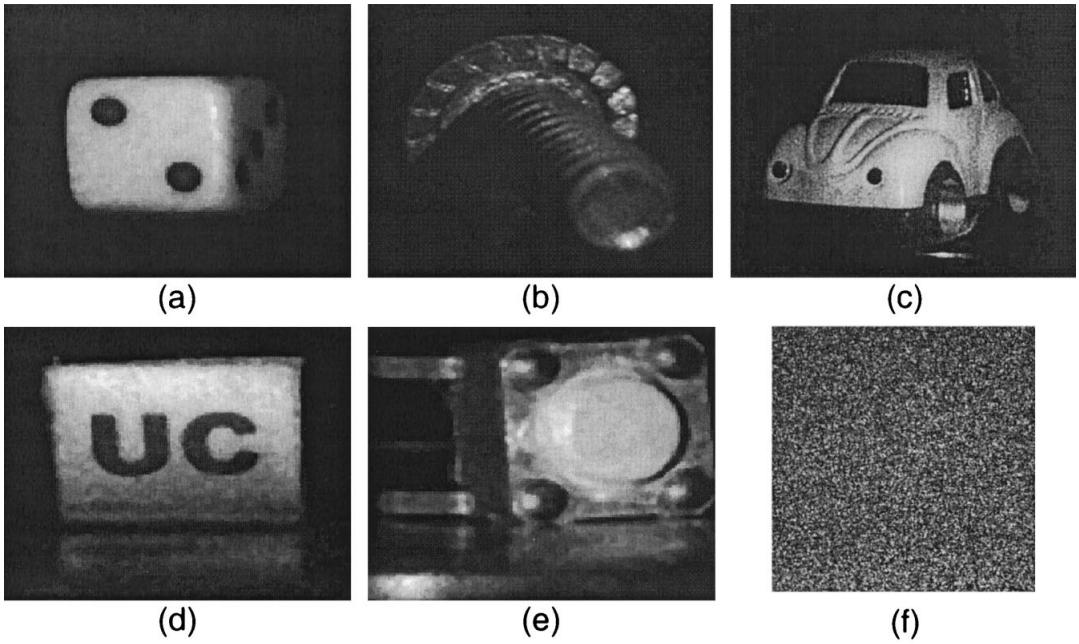


Figure 6.1: Amplitude Resconstruction of holograms and amplitude map of hologram used in [20]. (a)-(e) Amplitude reconstruction of holograms. (f) 512x512 Amplitude Map of (a).

From the table, it seems entering amplitude and phase in one file results in a similiar average compression ratio as the simple holograms used in this experiment. This is normal since mixing the two types of data reduces compressibility. For the separated amplitude and phase files, the average are even lower. There are two reasons for the lower compression ratio of PSI holograms as compared to simple holograms in this thesis: The removal of the conjugate and zero-order terms from the holograms using an in-line setup for PSI holograms, and the amount of bits used to represent the intensity.

The removal of the zero-order and conjugate terms creates a more uniform hologram that reduces a lot of noise. In turn, it allows the scene to contain less noise and thus allow better compression. Looking at the amplitude reconstruction of the holograms and their compression ratios, it is evident that the less reflective second image with the darkened screw 6.1(b), it is not as compressible as the other holograms. Using the same logic used to analyze holograms in

Table 6.2: Compression ratios of 5 holograms from [20] using real and imaginary data in separate files.

Hologram No.	1	2	3	4	5	Average
Huffman	0.42	0.96	0.84	0.41	0.41	0.61
LZ77	0.29	0.92	0.24	0.29	0.28	0.40
LZW	0.27	1.00	0.25	0.28	0.27	0.41
BWT	0.20	0.75	0.15	0.20	0.19	0.30

this thesis, the screw is a recorded at an angle of less symmetry. Most of the other holograms are either positioned directly facing the camera (increasing symmetry) and/or contain many gradual shades of white increasing (Figure 6.1(c)) the pixel redundancy. The dark screw, on the other hand, contains many different shades and the rings on the screw affects the distribution on the hologram, due to it being in an angle where the amplitude, phase, real, or imaginary values would not contain a good amount of repeated patterns. The holograms studied in this thesis do not employ PSI methods and therefore would result in a weaker compression ratio through the added noise from zero-order and conjugate images.

In the experiments done by Naughton et al, the CCD camera used allow a 10-bit storage size for pixels. Therefore, the intensity stored within the CCD cameras only store 10 bits of data versus the 64-bit double values calculated in this thesis. With the intensity values used being only 10-bits, precision is reduced for calculating amplitude, phase, real, or imaginary values. With the precision of intensity being reduced, some minor different values becomes the same due to the lost of precision, resulting in the same value for the calculation of other values. Therefore, compressibility is enhanced through this lost of precision. The holograms used in this thesis are all calculated from higher precision double values. Therefore, minuscule changes in the less significant bits of the calculated values would cause compression ratios to weaken, especially for finer pixel density holograms which contains much finer details and more variations in values. Therefore, this increases the precision for a decrease in compression ratio.

It was shown, however, that mapping the amplitude and phase values into a smaller range of values, and then compressing the holograms with BWCS were as good as compressing PSI holograms. The results shown in Experiment 5 of Chapter 5 clearly indicates that it can go as low as 5% of the original file size if the values were stored in 3 bits.

6.3 Summary of Post-Experiment Analysis

From the previous analyses, it can be shown that complicated two-dimensional images are not good holograms for use with compression. The fixed depth for hologram recording two-dimensional images causes fringes that contain very few repeatable patterns and decreases compression. For simple holograms, it was shown through the complex image and through different LZ and PPM schemes that items that rely heavily on local or ordered data fail compression due to the increase in size and match-search/context-adding time for more complicated holograms. For very simple ones, they are also compression ratio bottlenecked by their inability to make use of repeated patterns in directions other than row-by-row, left-to-right.

Entropy coders are reliable in terms of a stable but weak compression ratio and a fast performance time. This is due to their reduction of the number of bits required to represent the input symbols. In the case of this thesis, uncompressed input are mainly processed by byte, which means the number of different values are only 256— a small enough size for entropy encoders to reduce in terms of bit re-representation.

BWCS ignores direct local input redundancy by lexographically sorting a block of input. Once the block is sorted, the chances of the resulting output contains a lot of same-valued values to be directly beside each, allowing direct reduction of pixel redundancy through move-to-front and run-length encoding stages. In this sense, it is the most direct method that deals with pixel redundancy, which is the reason the compression ratios are much better in most cases. However, the block size usually determines the effectiveness of its compression ratio. At 100% of the original image as the block size, compression ratio reaches its maximum potential. In general, BWCS is a good choice for good compression ratios due to its effectiveness even for data that doesn't contain a lot of repeated or symmetric patterns— the scheme may ignore some symmetric properties after being sorted.

Mapping amplitude and phase values in a smaller range proved to be very effective. For a simple image, mapping the values to lower bit lengths allowed the file size to reduce to less than 15% of the original file. This is almost the same or better than the results presented by Naughton et al in [20]. However, there are more investigations to do before confirming that usefulness of this method. Mapping values into a smaller range is lossy. For printed transparencies, physical viewing is possible after reduction. However, it may be a different case using other output mediums such as a computer or a projector. Therefore, further investigation is required before this method can be employed for shortening the file sizes of the amplitude and phase files.

6.4 Final Suggestion of Factors for Effective Hologram Compression

In the case of simple holograms, the compression ratio could also be further reduced through taking advantage of the symmetric properties of simple holograms. For simple holograms, they are usually symmetric about x or y. For this fact, if a quad-tree or folding algorithm (an algorithm can recover data through symmetric mirroring the current side of the input) can reduce the size of the image by half or even more. For amplitudes or phases where the repeating patterns are larger and more regular due to the images used or symmetric shapes, an algorithm can be potentially invented to do one of two things: 1) search amplitude and phase files for repeating two-dimensional patterns and then mark down where the data symmetrically mirrors and then remove the mirrored-side. During restoring, one side is recovered through mirroring one from the remaining symmetric image. This is called folding. The other idea is to use quad-tree-like structures to figure out two-dimensional repeating patterns and store only one set of that pattern and mark down the starting coordinates of the next sets. By default, this at least removes half of the symmetric patterns in the holograms. In the case of highly symmetric data in holograms in both directions, the compression ratio can be lower than 0.25 due to the highly symmetric structure.

Another suggestion focuses on improving complicated hologram compression by dissecting the phase values into more manageable regular shapes. The idea is as follows:

1. Detect regular shape patterns from the hologram images that can suggest a decrease in compression ratio using shape recognition ideas such as the Hough transform;

2. Remove the shape from the hologram, mark the removed location and orientation, and replace the removed points with a single value (this increases memory load);

3. Compress the removed shape with a Burrows-Wheeler scheme and save it to the output file;

4. Continue removing and compressing shapes until a certain threshold; and

5. Compress the left-over hologram using a Burrows-Wheeler scheme.

For decompression, all that is needed is restoration of all compressed items and putting the shapes back into the original hologram. In this case, the regular shapes list can be built after arbitrarily testing different shapes. The threshold can also be experimentally tested. As for compression of the left-over hologram, the addition of a uniform value to removed shape points can allow the BWCS to easily compress the data into very small values— saving space for the removed shape. However, this way of adding data for compression would cause major time overhead and memory load due to the removal of shapes and compressing them individually outside of the original hologram. The same applies to decompression with the recovering of the shape and restoring the shape into the block of the restored original shapes-removed hologram.

This method is only effective for phase maps in holograms that were recorded from two-dimensional virtual objects. This is due to the nature of phases in the two dimensional sense: they look like the shape that was used to create the hologram. Therefore, by cutting the shape into more regular shapes, the symmetric properties of regular shapes can be preserved in hopes of increasing compression ratios. For amplitude maps, since there is no apparent trend with complicated holograms, it would be better to use an entropy coder to reduce the amount of bits.

The last suggestion is to modify the Burrows-Wheeler Transform for faster compression speeds, multi-directional reading of blocks, and changing the shape orientation of blocks. The improvements in speed can be easily done with parallel algorithms on parallel architectures due to the mutually exclusive nature of individual blocks. Changing the direction of block reading such as vertical, circular, or zig-zag allows the data to be more easily compressed depending on the orientation of the hologram. In this way, it would be easy for the compression scheme to search for higher compression ratios using different shapes or one fixed-shape. The reason the Burrows-Wheeler Transform is used is due to its direct dealing with pixel redundancy. Since the transform directly increases the chance of the same pixels to be beside each other, the compression ratio is bound to be better in most settings with a lot of redundant pixels. Allowing the scheme to choose which direction or shape to handle the blocks increases the chances of the compression ratio to increase through using the best order in which to read the data or consider blocks.

Overall, it seems like the best form of data to compress are the amplitude and phase data

in separate files due to the symmetric nature of simple holograms— amplitude and phase each contains a lot of symmetry. Therefore, merging them into a real image reduces the symmetry of the resulting image and as a result the compression ratio. The main advantage of storing only the real image is the file size for the printable real image is much smaller since the values are stored in 8-bits. Therefore, using only the real image is not good for computer hologram viewing applications so it would be better to store amplitude and phase if used for electronic viewing of holograms. By attempting to reduce the number of bits used for the amplitude and phase values, the file size can also be reduced greatly. However, there still requires further investigation to discover a suitable amount of bits to reduce which does not affect viewing quality for other output mediums.

6.5 Summary

The results of the experiments showed that the simple shaped objects used for recording are best suited for compression. In simple shaped holograms, the use of the symmetric properties of regular simple shapes gives a great advantage in terms of compression. It also seems that transforming the files using Burrows-Wheeler Transforms helps a lot in terms through the direct sorting of same pixels side-by-side. For complicated 2D scenes, compression is not useful due to the high variability of data from different shapes. Therefore, it loses in compression ratios due to scene complexity and lost of repeated patterns In this case, it is better to record the shape from an angle as a 3D object or not use 2D shapes as target virtual object. Overall, there are still a few ways in which the some of the special factors assist with better hologram compression. Taking the best from different compression schemes, the best merged compression scheme is to employ the BWCS on the data for the most part with the inclusion of 2D pattern recognition such as the quad-tree or folding method specified above.

Chapter 7

Conclusion

Compression of holograms are very different than regular image compression due to the need of compressing three-dimensional data such as real/imaginary values or amplitude/phase values instead of just an intensity value. Problems such as the precision of the associated values, compression ratio, and overall performance time, all affect the effective of hologram compression. Also, the nature of different holograms must be thoroughly investigated in order to design an effective method for compression holograms.

The effectiveness of other compression schemes such as PPM and LZ schemes proved to be effective up to a certain resolution. In general, the finer the resolution, the weaker the effectiveness due to the increased storage size of context tables, dictionaries, and sliding window matches. This results in an increase in storage size but also slows performance time. With the advancement of modern day technology and the increasing size of viewable holograms, compression schemes with poor compression ratios and slow performance times are unfavorable.

In this thesis, it was discovered that holograms created from symmetric two-dimensional shapes contain highly symmetric patterns in their wavefront distribution. Therefore, compression on these types of holograms resulted in better compression ratios than ones with irregular patterns. However, these ratios are still low compared to those presented in PSI methods. This is due to the removal of the conjugate and zero-order images in PSI, as well as decreased precision in PSI camera pixels. To take advantage of symmetry, schemes to reduce symmetric redundancy should be designed to decrease the output size.

The effects of symmetry were ignored after using the Burrows-Wheelers Transform to reorder the data. However, by directly trying to reorder the data in a way where the same symbols can be put beside each other, BWT succeeded in having the highest compression ratio over other conventional compression schemes. However, the effectiveness of these algorithms were still limited by their uni-directional textual nature to process input symbols. A better way for this to be more effective is through a multi-directional input symbol processing possibly using some from of shape detection algorithm.

The ultimate goal of compression is to reduce all possible form of redundancy without losing in the output's viewing quality. In lossless compression, the difficulty increases due to the requirement of not losing any information. The research done here is meant as a starting step for better understanding the inherent factors that constitute a good hologram compression scheme design. It is the hope of the author that the work presented here can be a start to the design of more compression algorithms for holograms.

7.1 Future Work

There are a lot of future works that can be done. Compression on holograms recorded with three-dimensional objects would be one of many logical next steps after this work. Two-dimensional shapes as target for recording is not fully sufficient for representing all factors that affect the compression of holograms. The implementation of the suggestions from the previous chapters (quadtree search for repeated patterns, symmetric image flipping, phase map shape removal, BWT improvements) would also be a possible next step. In particular, BWT modifications would be a topic of great interest for not just hologram compression but image compression as well since it becomes a two-dimensional compression algorithm that increases the likelihood of higher compression due to the selection of more likely pixels into the block. From this study, it was discovered that simple holograms contained many symmetric properties that can be exploited for size reduction. After the successful design of these test algorithms, a test on them for both two-dimensional shape and three-dimensional object holograms would be a good try on the effectiveness of the approaches. Another thing would be to create a hologram generator that removes the zero-order and conjugate image as in PSI. A hologram viewer implementation would also be advantageous for testing and trying lossy image compression methods such as JPEG or Frenelets to find an acceptable amount of information loss without affecting viewing quality.

The viewer application can also be used as a mechanism for evaluating image quality loss through mapping the amplitudes and/or phases into values representing coarser ranges (e.g. allow only 100 distinct phase values) as done in Experiment 5 of Chapter 5. This will result in certain values mapped into the same value. These newly re-represented holograms can then be compressed which can further reduce the file sizes and theoretically improve the compression ratios of the holograms. Seeing from the results of experiment done in Chapter 5, it can be seen that complicated images may not be very useful outside of having their file size reduced through remapping values. Therefore, investigation on more simple images and three-dimensional models should be the focus. Investigating remapping allows for the discovery of a metric defined as to how much degradation is accepted from mapping different ranges of values. At the same time, compression ratios of the newly mapped holograms can also be investigated.

Bibliography

[1] J. Abel, "Improvments to the Burrows-Wheeler Compression Algorithm: After BWT Stages", 2003, http://www.juergen-abel.info/Preprints/Preprint_After_BWT_Stages.pdf, 2010.

[2] T.C. Bell, J.G. Cleary, and I.H. Witten, Text Compression, Prentice-Hall, New Jersey, 1990.

[3] S.A. Benton, "On a method for reducing the information content of holograms," J. Opt. Soc. Am. **59**, pp 1545, 1969.

[4] M. Burrows and D.J. Wheeler, "A Block-sorting Lossless Data Compression Algorithm," Digital Systems Research Center Research Report 124, May 1994.

[5] Chipwich, "The CorticalCafe Computer Generated Hologram (CGH) Construction Kit", http://corticalcafe.com/prog_CGHmaker.htm, 2011.

[6] J.G. Cleary and I.H. Witten, "Data compression using adaptive coding and partial string matching," IEEE Transactions on Communications **32 (4)**, pp. 396402, April 1984.

[7] E. Darakis, T. J. Naughton, and J. J. Sorghan, "Compression defects in different reconstructions from phase-shifting digital holographic data," Applied Optics **38**, pp. 4579-4586, July 2007.

[8] E. Darakis and J. J. Sorghan, "Compression of interference patterns with aplication to phase shifting digital holography," Applied Optics **45**, pp. 2437-2443, April 2006.

[9] E. Darakis and J. J. Soraghan, "Compression of digital hologram sequences using MPEG-4," Proc. SPIE **7358**, 735811-1 to 735811-8, 2009.

[10] D.J. De Bitteto, "Holographic panoramic stereograms synthesized from white light recordings," Appl. Opt. **8**, pp. 1740-1741, 1970.

[11] Y.N. Denisyuk, "Photographic reconstruction of the optical properties of an object in its own scattered radiation field," Sov. Phys. - Dokl. **7**, pp. 543-545, 1962.

[12] S. Deorowicz, "Improvements to Burrows-Wheeler Compression Algorithm", preprint of Software - Practice and Experience, June 24, 2000.

[13] M. Dipperstein, "Michael Dipperstein's Page O' Stuff", http://michael.dipperstein.com, October 2010.

[14] D. Gabor, "A new microscope principle", Nature **161**, pp. 777-778, 1948.

[15] J. W. Goodman, Introduction to Fourier Optics, 3rd edition, Roberts & Company Publishers, Greenwood Village, 2005.

[16] E. Hecht, Optics, 4th edition, Addison-Wesley Publishing Company, Reading, MA, 2001.

[17] F. S. Hill, Jr., Computer Graphics Using OpenGL, 2nd edition, Prentice-Hall, New Jersey, 2001.

[18] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", In the proceedings of the I. R. E. **40(9)**, pp. 1098-1102, September 1952

[19] Y. Ichihashi, H. Nakayama, T. Ito, N. Masuda, T. Shimobaba, A. Shiraki, and T. Sugie, "HORN-6 special-purpose clustered computing system for electroholography", Optics Express **17(16)**, pp. 13895-13903, August 2009.

[20] B. Javidi and E. Tajahuerce, "Three-dimensional object recognition by use of digital holography," Optics Letter **25**, pp. 610-612, May 2000.

[21] A. W. Lohmann and D.P. Paris, "Binary Fraunhofer holograms generated by computer," Applied Optics **6**, pp. 1739-1748, 1967.

[22] M. Lucente, "Diffraction-Specific Fringe Computation for Electro-Holography," doctoral dissertation, Dept. of Electrical Eng. and Computer Science, MIT, 1994.

[23] A. Moffat, "Implementing the PPM data compression scheme," IEEE Transactions on Communications **38 (11)**, pp. 19171921, November 1990.

[24] T. J. Naughton, Y. Frauel, B. Javidi, and E. Tajahuerce, "Compression of digital holograms for three-dimensional object reconstruction and recognition," Applied Optics **41**, pp. 4124-4132, July 2002.

[25] T. J. Naughton, J. B. McDonald, and B. Javidi, "Efficient compression of Fresnel fields for Internet transmission of three-dimensional images," Applied Optics **42**, pp. 4758-4764, 2003.

[26] M. Nelson, "Data Compression with the Burrows-Wheeler Transform", Dr. Dodd's Journal, September 1996, `http://www.inference.phy.cam.ac.uk/is/donut/compression/BWTnelson.ps`, 2010.

[27] N. Pandey, D. P. Kelly, T. J. Naughton, and B. M. Hennelly, "Speed up of Fresnel transforms for digital holography using pre-computed chirp and GPU processing", In the proceedings of SPIE **7442(744205)**, 2009.

[28] K. Sadakane, "A Fast Algorithm for Making Suffix Arrays and for Burrows-Wheeler Transformation", In the proceedings of the IEEE Data Compression Confernce, Snowbird, Utah, pp. 128-138, March 1998.

[29] U. Schnars and W. Juptner, Digital Holography, Springer Publishing, New York, 2005.

[30] J.Seward, Bzip2 Homepage, `http://bzip.org/`, 2010.

[31] Dmitry Shkarin, "PPM: one step to practicality", Proceedings of the IEEE Data Compression Conference, Snowbird, Utah, pp. 202-211, 2002.

[32] C. Slinger, C. Cameron, and M. Stanley, "Computer-Generated Holography as a Generic Display Technology," Computer **38**, no. 8, pp. 46-53, Aug. 2005.

[33] E. Syahrul, J. Dubois, V. Vajnovszki, T. Saidani, and M. Atri, "Lossless Image Compression Using Burrows-Wheeler Transform (Methods and Techniques)", In the proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet-Based Systems, pp. 338 - 343, 2008.

[34] T. Welch, "A Technique for High-Performance Data Compression," IEEE Computer **17(6)**, pp. 8-19, June 1984.

[35] X. Xu, S. Solanki, X. Liang, S. Xu, R. B. A. Tanjung, Y. Pan, F. Farbiz, B. Xu, and T.-C. Chong, "Dynamic Display of 3D Objects in Real and Virtual Spaces with Computer-Generated Holography," Proc. VRCAI '08, December 2008.

[36] I. Yamaguchi and T. Zhang, "Phase-shifting digital holography," Optics Letter **22**, 1268-1270, August 1997.

[37] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," IEEE Transactions on Information Theory, **23(3)**, pp. 337-343, May 1977.

[38] "Holography," Wikipedia: The Free Encyclopedia, Wikimedia Foundation Inc. `http://en.wikipedia.org/wiki/Holography`, March 2011.

Appendix A

Images Used for Experiments

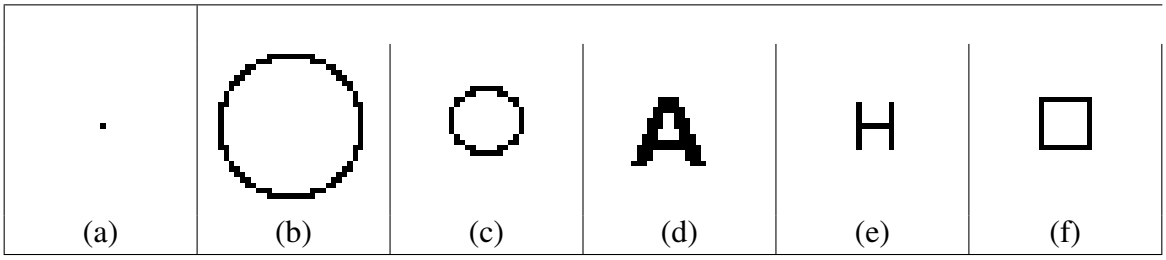


Figure A.1: 30x30 images from [5] for use as object for hologram creation by the program from [5]. (a) An image with one point. (b) A big circle. (c) A smaller 20x20 pixel per inch circle. (d) A letter “A”. (e) A letter “H”. (f) A square. Images are scaled 2 times larger than the original size for viewbility.

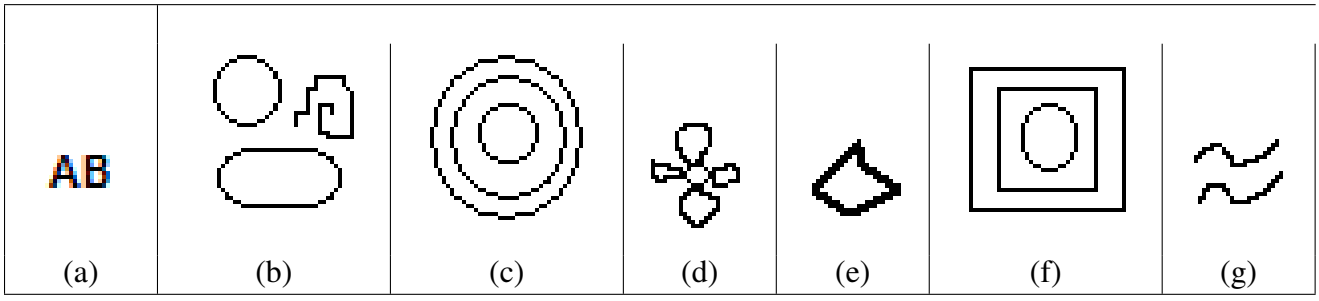


Figure A.2: 30x30 images used as object for hologram creation by the program from [5]. (a) An image with the letters “AB”. (b) An image with a circle, an ellipse, and a squiggly line. (c) Concentric circles. (d) A hand drawn flower. (e) A hand-drawn diamond. (f) Rectangle encased in another rectangle encased in an ellipse. (g) Two squiggly lines. Images are scaled 1.5 times larger than the original size for viewbility.

```
<!—
This is a sample XML file which defines a trianglar object for the MedCosm CGHMaker
program. Feel free to edit virtually anything. The worst that will happen is that
you'll break something!
—>
<PointSourceArray>

<PointSource x=''0.0000'' y=''0.0000'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0015'' y=''0.0015'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0030'' y=''0.0030'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0045'' y=''0.0045'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0060'' y=''0.0060'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0075'' y=''0.0075'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0090'' y=''0.0090'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0105'' y=''0.0105'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0120'' y=''0.0120'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0135'' y=''0.0135'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0150'' y=''0.0150'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0150'' y=''0.0135'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0150'' y=''0.0120'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0150'' y=''0.0105'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0150'' y=''0.0090'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0150'' y=''0.0075'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0150'' y=''0.0060'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0150'' y=''0.0045'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0150'' y=''0.0030'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0150'' y=''0.0015'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0150'' y=''0.0000'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0135'' y=''0.0000'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0120'' y=''0.0000'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0105'' y=''0.0000'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0090'' y=''0.0000'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0075'' y=''0.0000'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0060'' y=''0.0000'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0045'' y=''0.0000'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0030'' y=''0.0000'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>
<PointSource x=''0.0015'' y=''0.0000'' z=''4'' amp=''1'' wavelength=''630e–9'' phase=''–1.25''/>

</PointSourceArray>
```

Figure A.3: XML description of a triangle.

Appendix B

Real Images for Holograms in Experiments

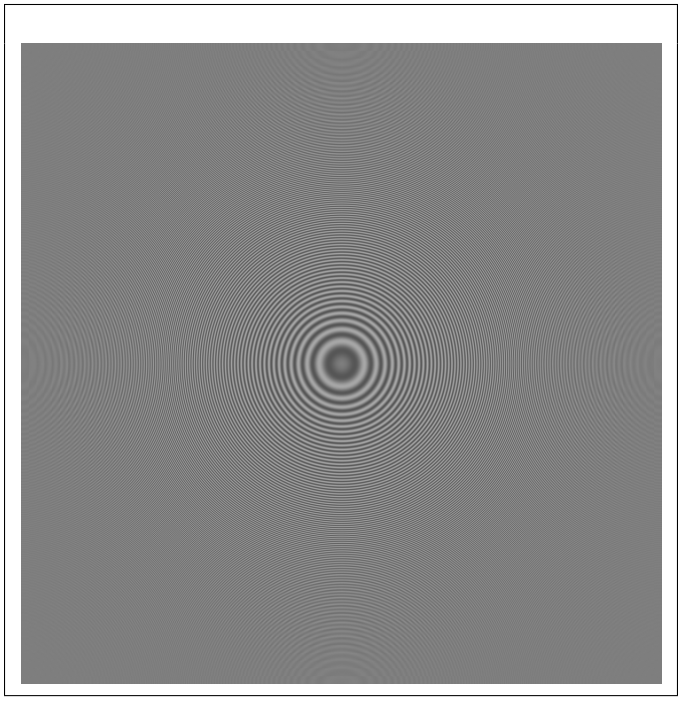


Figure B.1: Real image for 1200x1200 pixels per inch hologram created from A.1(a). Image is scaled 0.2 times of their original size.

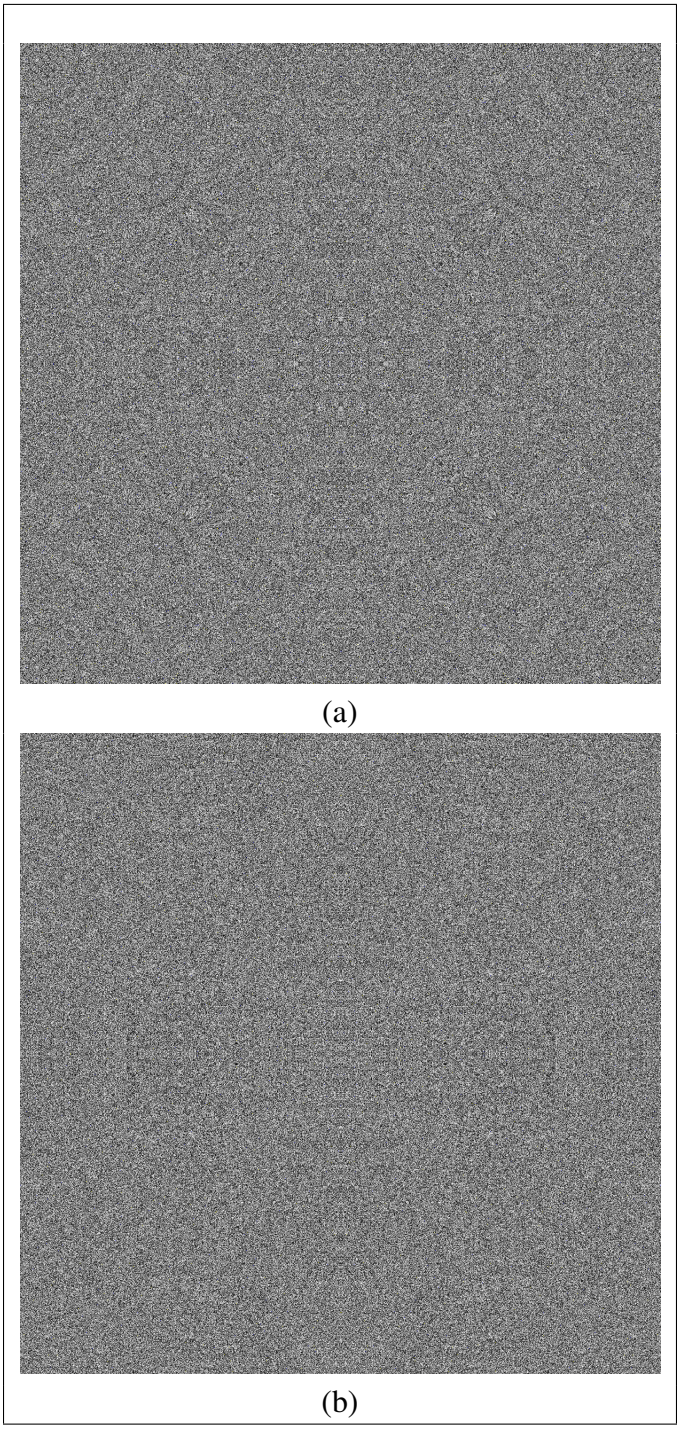
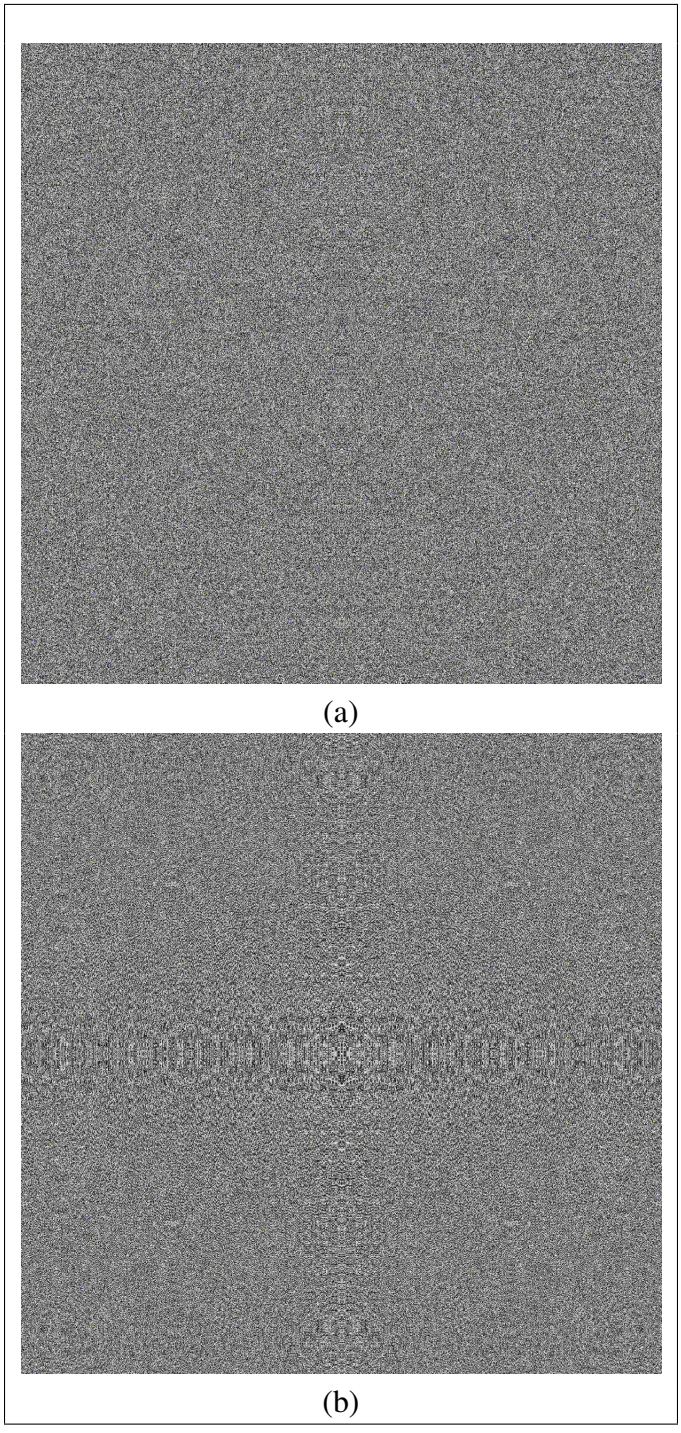


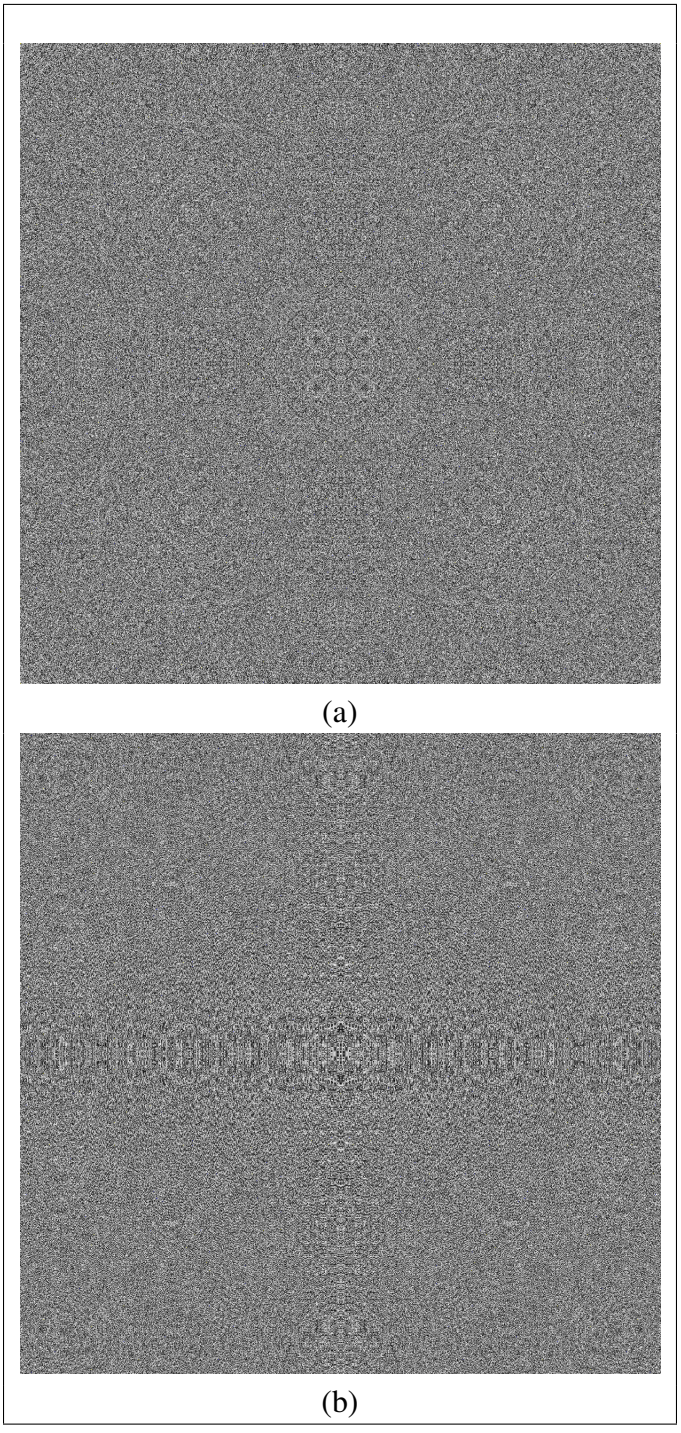
Figure B.2: Real images for 1200x1200 pixels per inch holograms created from images in A.1. Real images (a) and (b) refers to holograms generated from A.1 (b) and (c) respectively. Images are scaled 0.2 times of their original size.



(a)

(b)

Figure B.3: Real images for 1200x1200 pixels per inch holograms created from images in A.1. Real images (a) and (b) refers to holograms generated from A.1 (d) and (e) respectively. Images are scaled 0.2 times of their original size.



(a)

(b)

Figure B.4: Real image for 1200x1200 pixels per inch hologram created from A.1 and A.3. Real image (a) refers to hologram generated from A.1(f). (b) is hologram real image generated from A.3. Image is scaled 0.2 times of the original size.

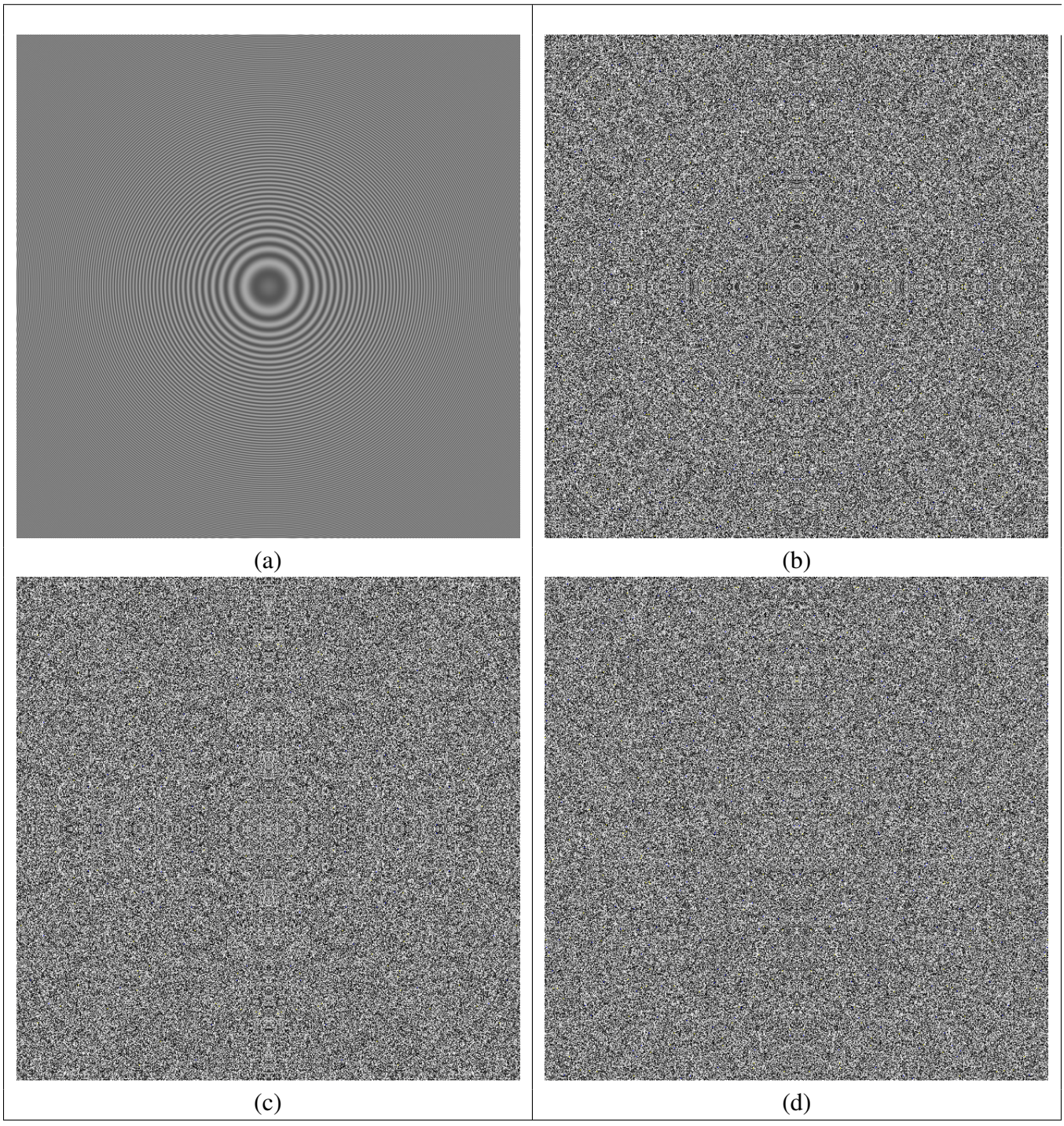
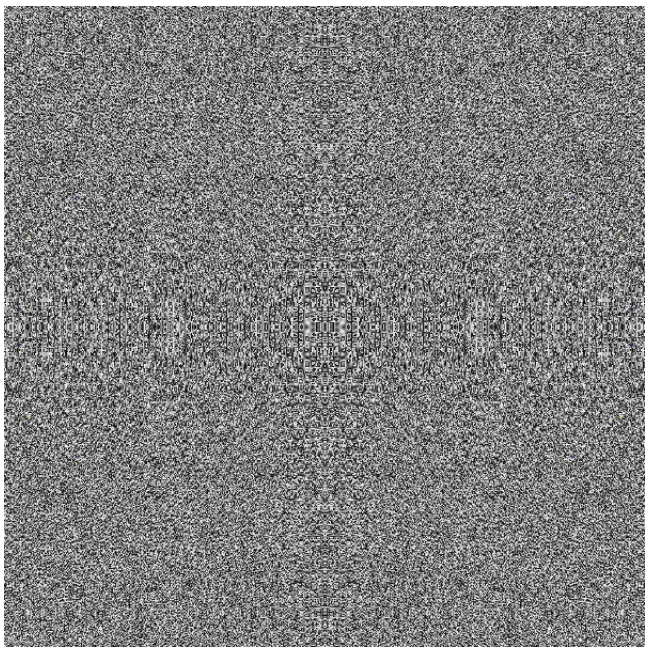
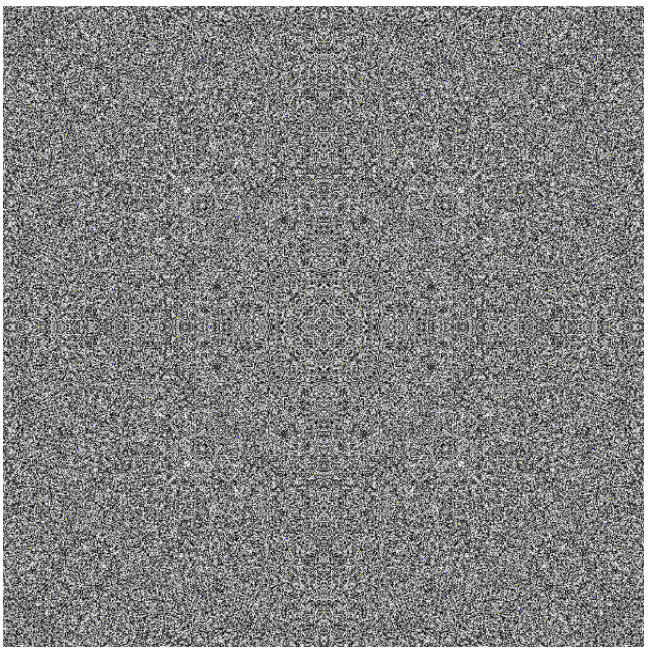


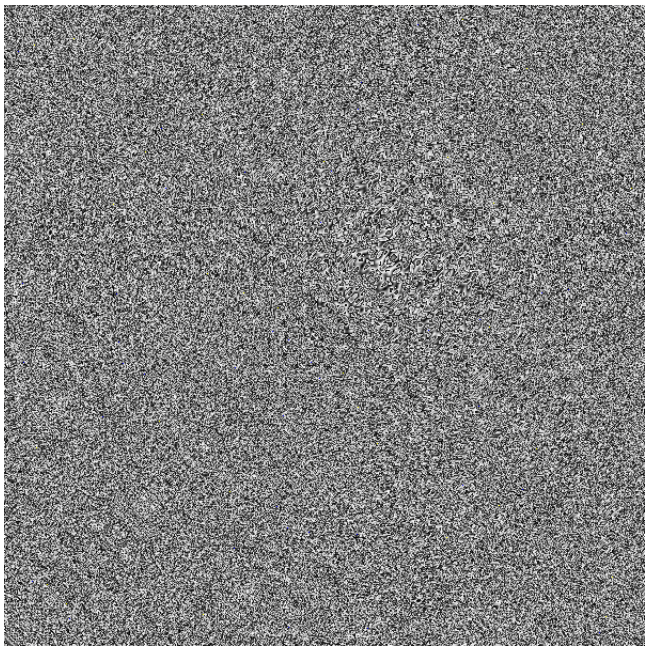
Figure B.5: Real images for 600x600 pixels per inch holograms created from images in A.1. Real images (a)-(d) refers to holograms generated from A.1 (a)-(d) respectively. Images are scaled 0.4 times of their original size.



(a)



(b)



(c)

Figure B.6: Real images for 600x600 pixels per inch holograms created from images in A.1 and A.3. Real images (a) and (b) refers to holograms generated from A.1 (e) and (f) respectively. (c) 600x600 pixel per inch real image of hologram for A.3. Images are scaled 0.4 times of their original size.

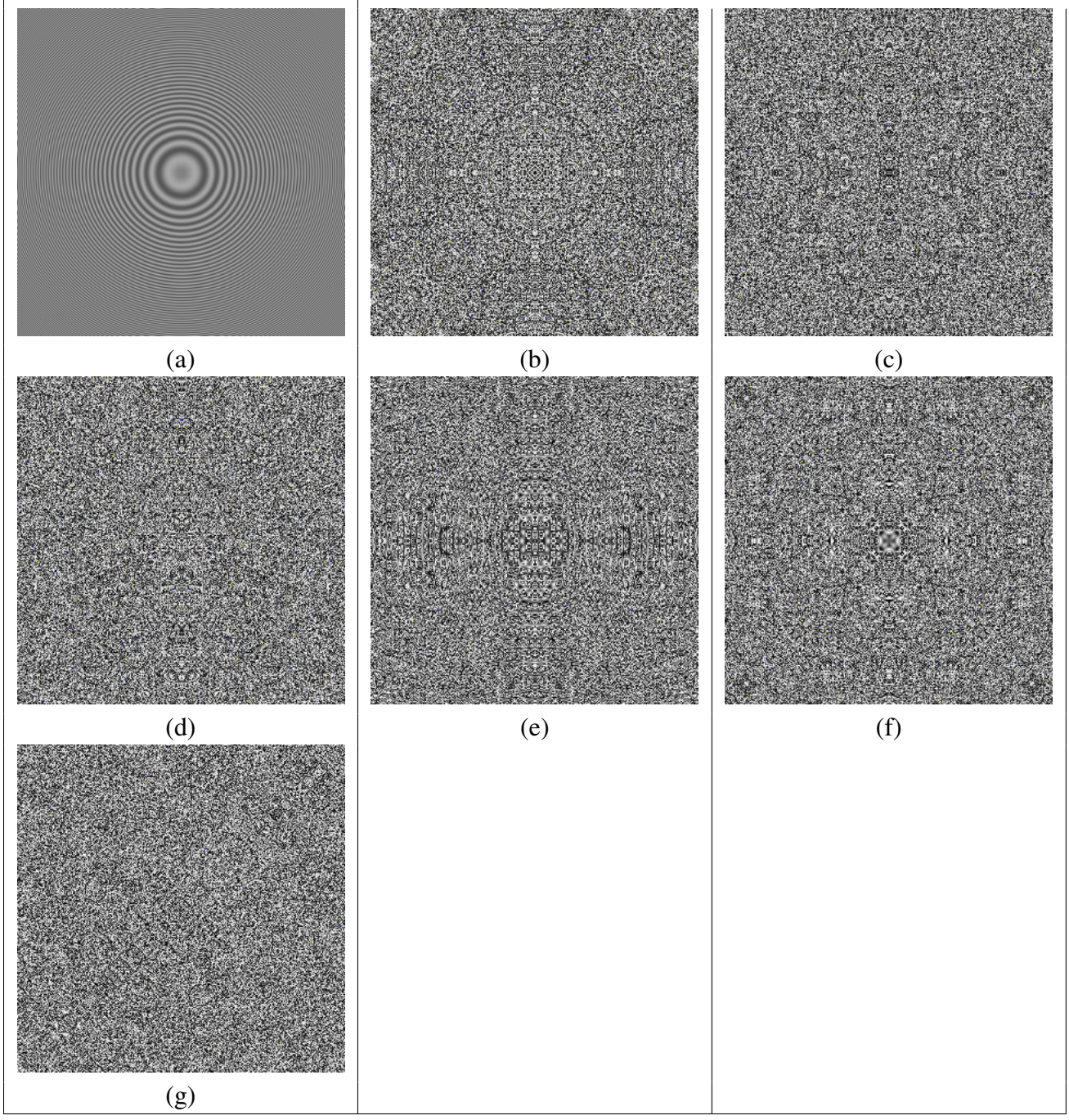


Figure B.7: Real images for 300x300 pixels per inch holograms created from images in A.1 and A.3. Real images (a)-(f) refers to holograms generated from A.1 (a)-(f) respectively. (g) Real image of hologram for A.3. Images are scaled 0.5 times of their original size.

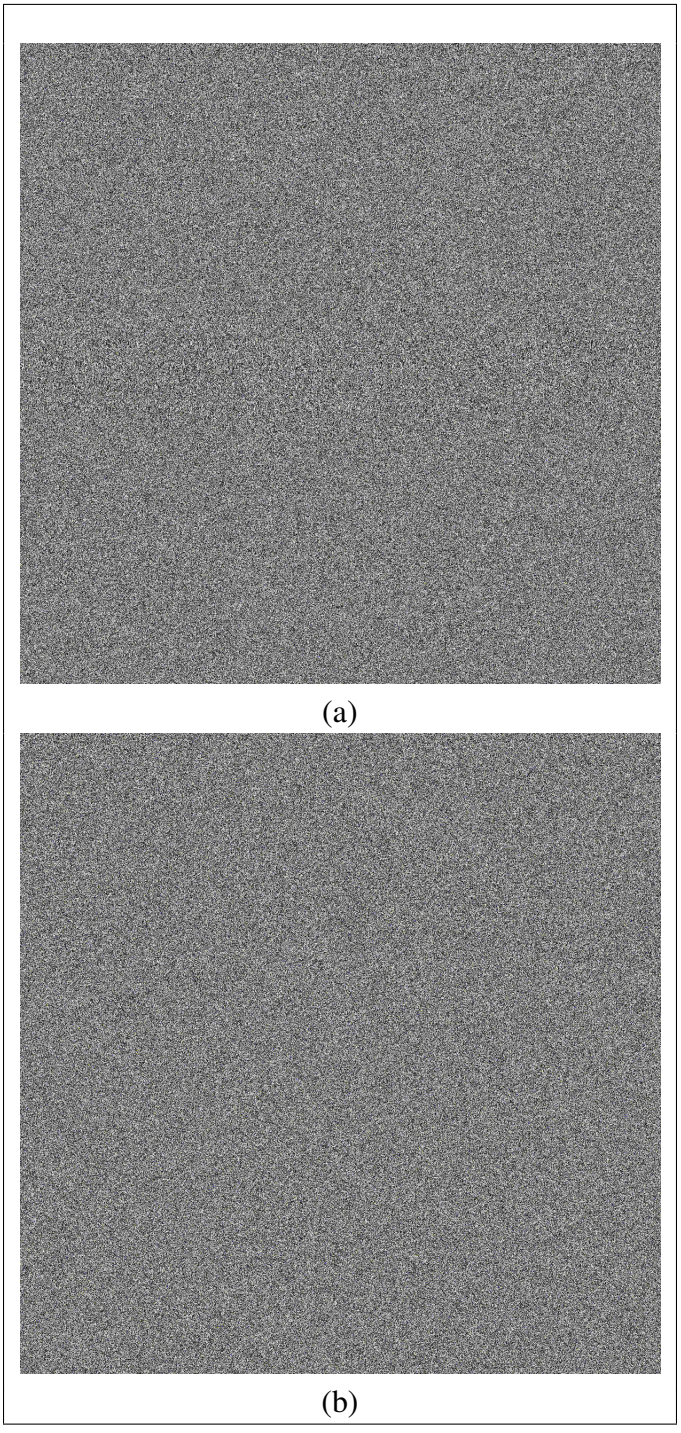


Figure B.8: Real images for 1200x1200 pixels per inch holograms created from images in A.2. Real images (a) and (b) refers to holograms generated from A.2 (a) and (b) respectively. Images are scaled 0.2 times of their original size.

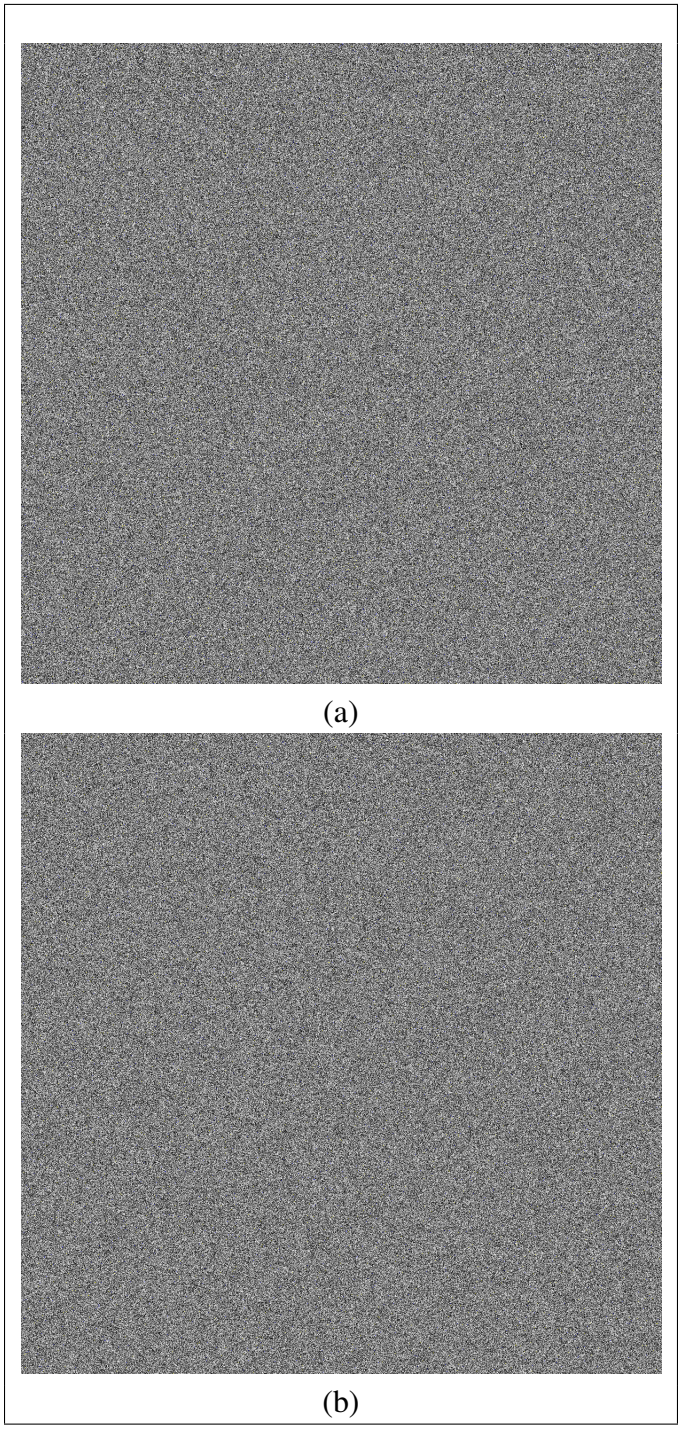
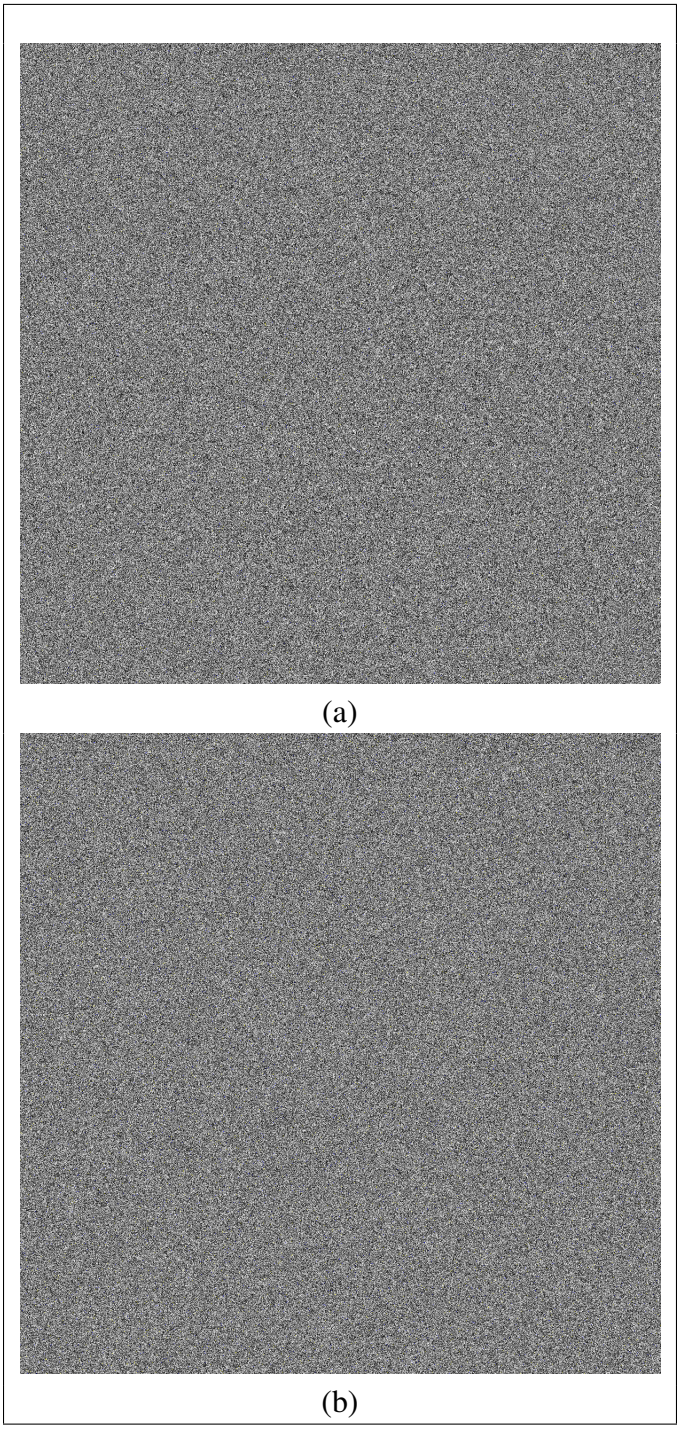


Figure B.9: Real images for 1200x1200 pixels per inch holograms created from images in A.2. Real images (a) and (b) refers to holograms generated from A.2 (c) and (d) respectively. Images are scaled 0.2 times of their original size.



(a)

(b)

Figure B.10: Real images for 1200x1200 pixels per inch holograms created from images in A.2. Real images (a) and (b) refers to holograms generated from A.2 (e) and (f) respectively. Images are scaled 0.2 times of their original size.

Figure B.11: Real image for 1200x1200 pixels per inch hologram created from A.2. Image is scaled 0.2 times of the original size.

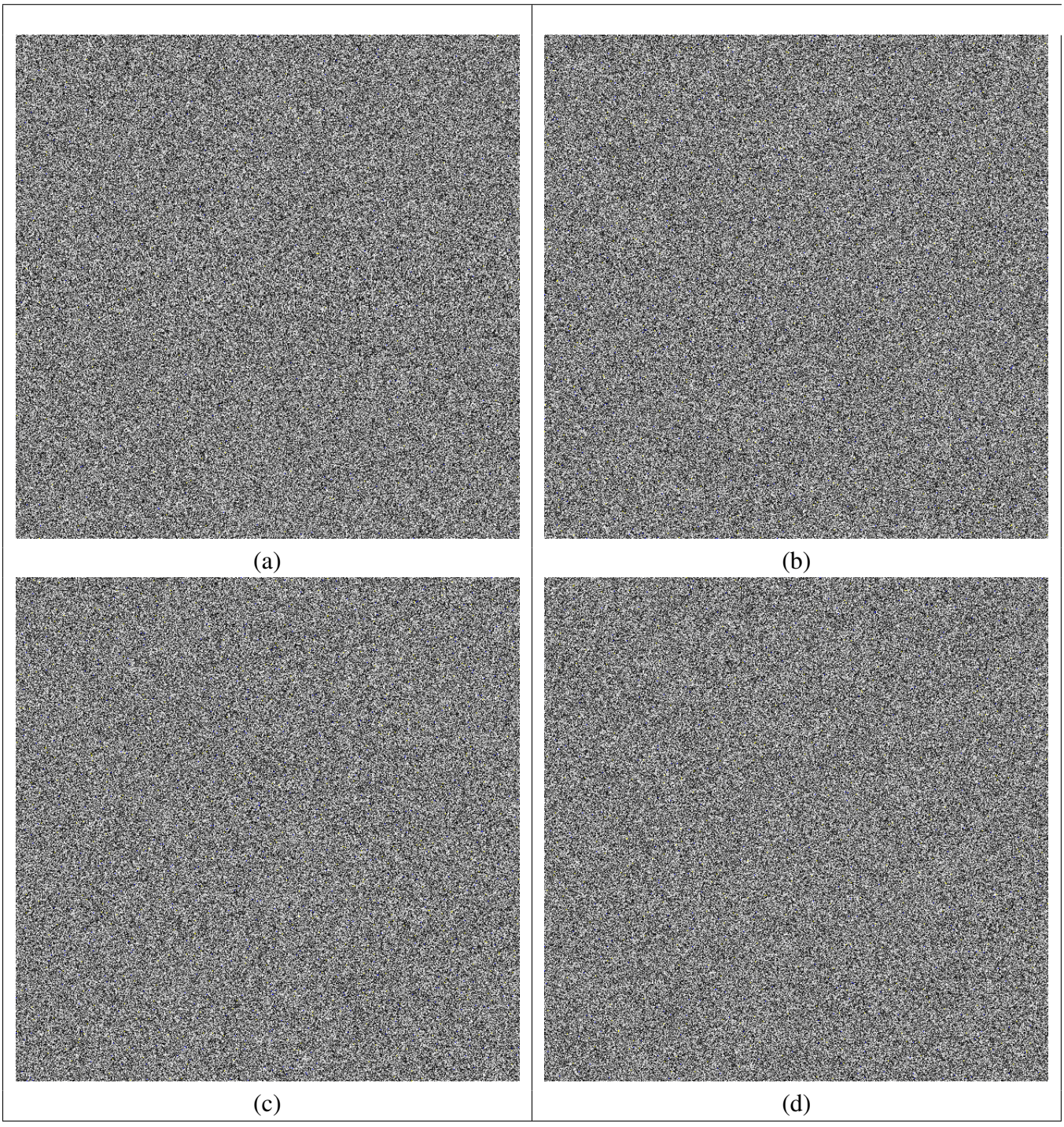
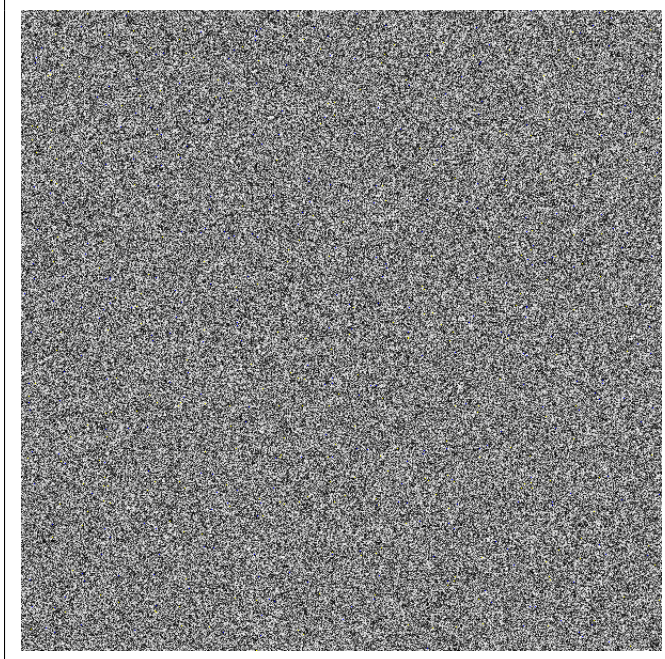
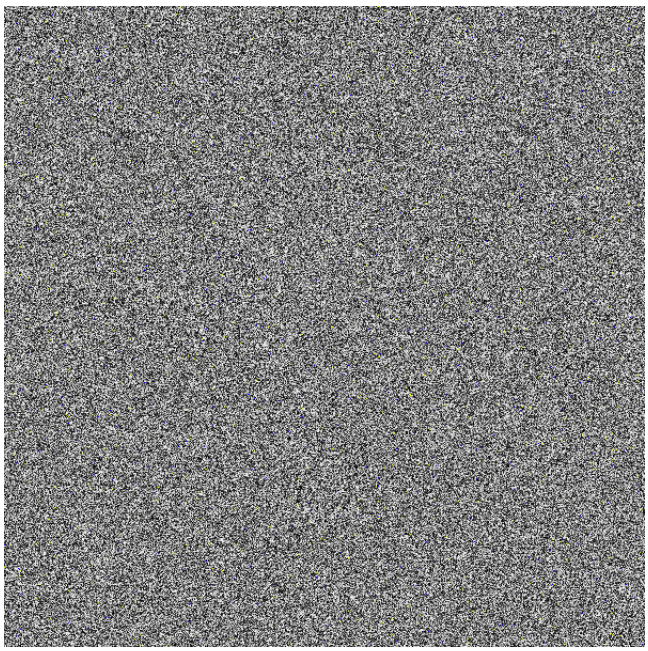


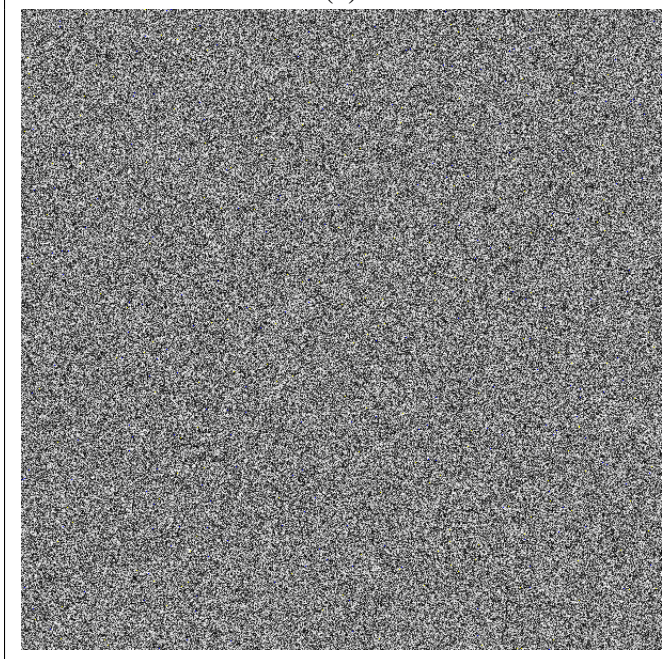
Figure B.12: Real images for 600x600 pixels per inch holograms created from images in A.2. Real images (a)-(d) refers to holograms generated from A.2 (a)-(d) respectively. Images are scaled 0.4 times of their original size.



(a)



(b)



(c)

Figure B.13: Real images for 600x600 pixels per inch holograms created from images in A.2. Real images (a)-(c) refers to holograms generated from A.2 (e)-(g) respectively. Images are scaled 0.4 times of their original size.

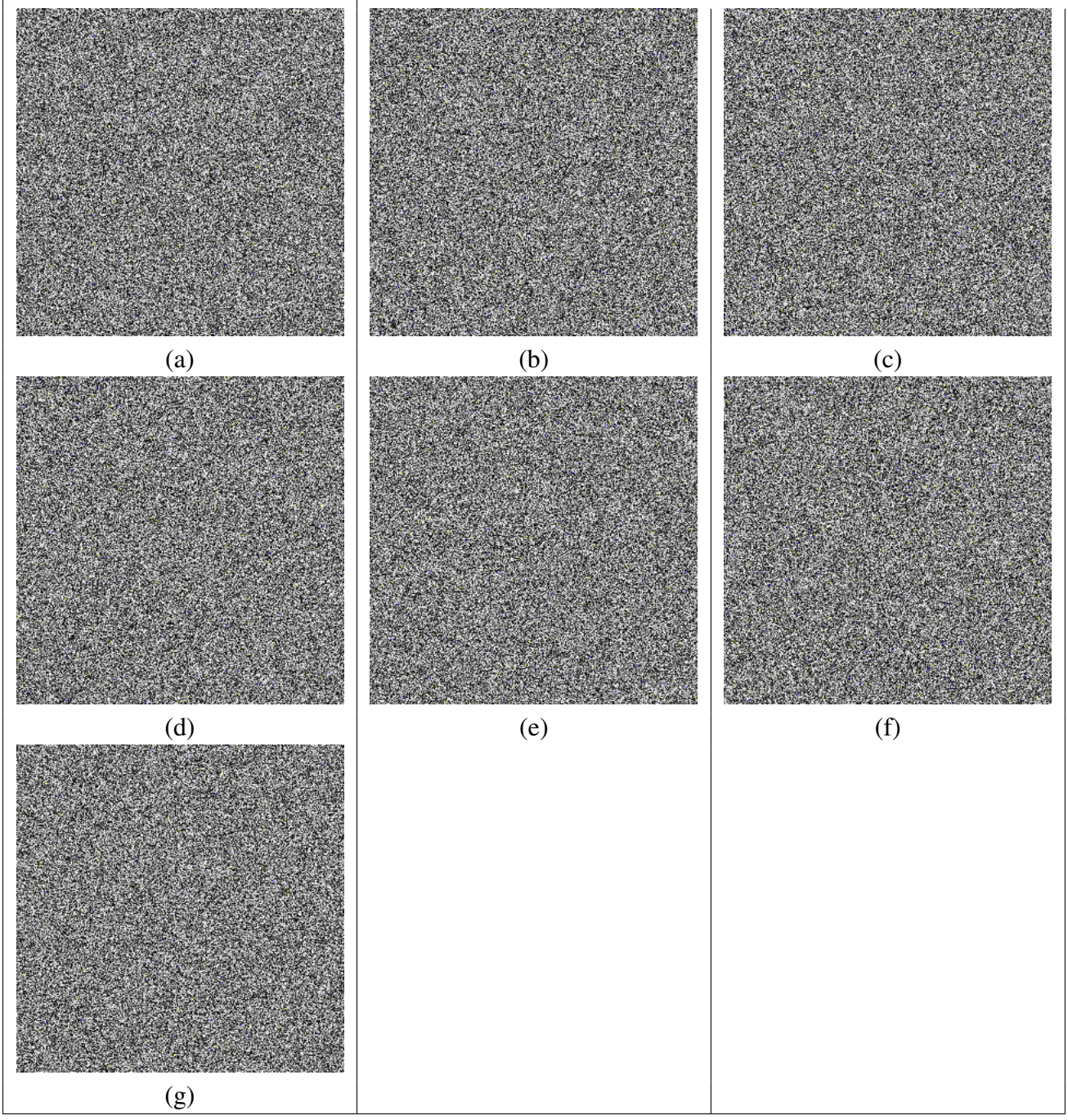
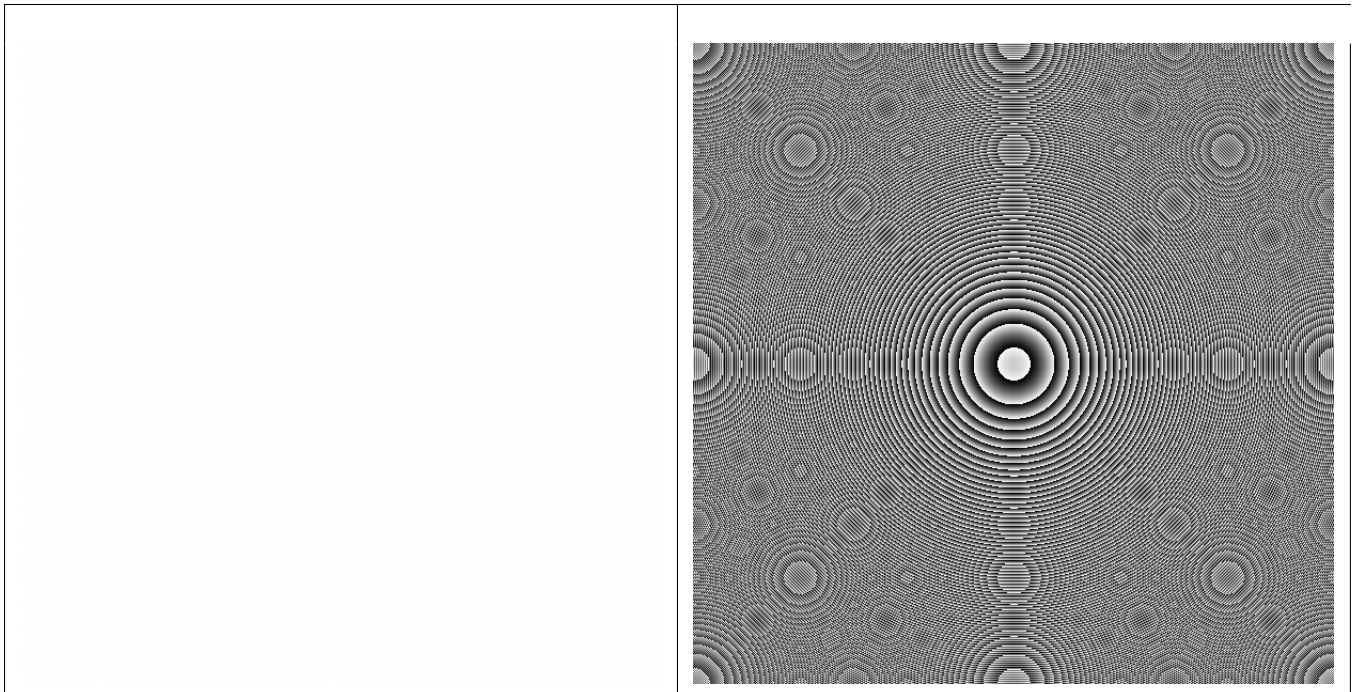


Figure B.14: Real images for 300x300 pixels per inch holograms created from images in A.2. Real images (a)-(g) refers to holograms generated from A.2 (a)-(g) respectively. Images are scaled 0.5 times of their original size.

Appendix C

Amplitude and Phase Maps for Holograms in Experiments

C.1 Amplitude and Phase Maps for Simple Images



(a) (b)

Figure C.1: Amplitude and phase map for 600x600 pixels per inch hologram created from image in A.1. (a) Amplitude map for A.1(a). (b) Phase map for A.1(a). Images are scaled 0.4 times of their original size. Note that the amplitude map is fully white.

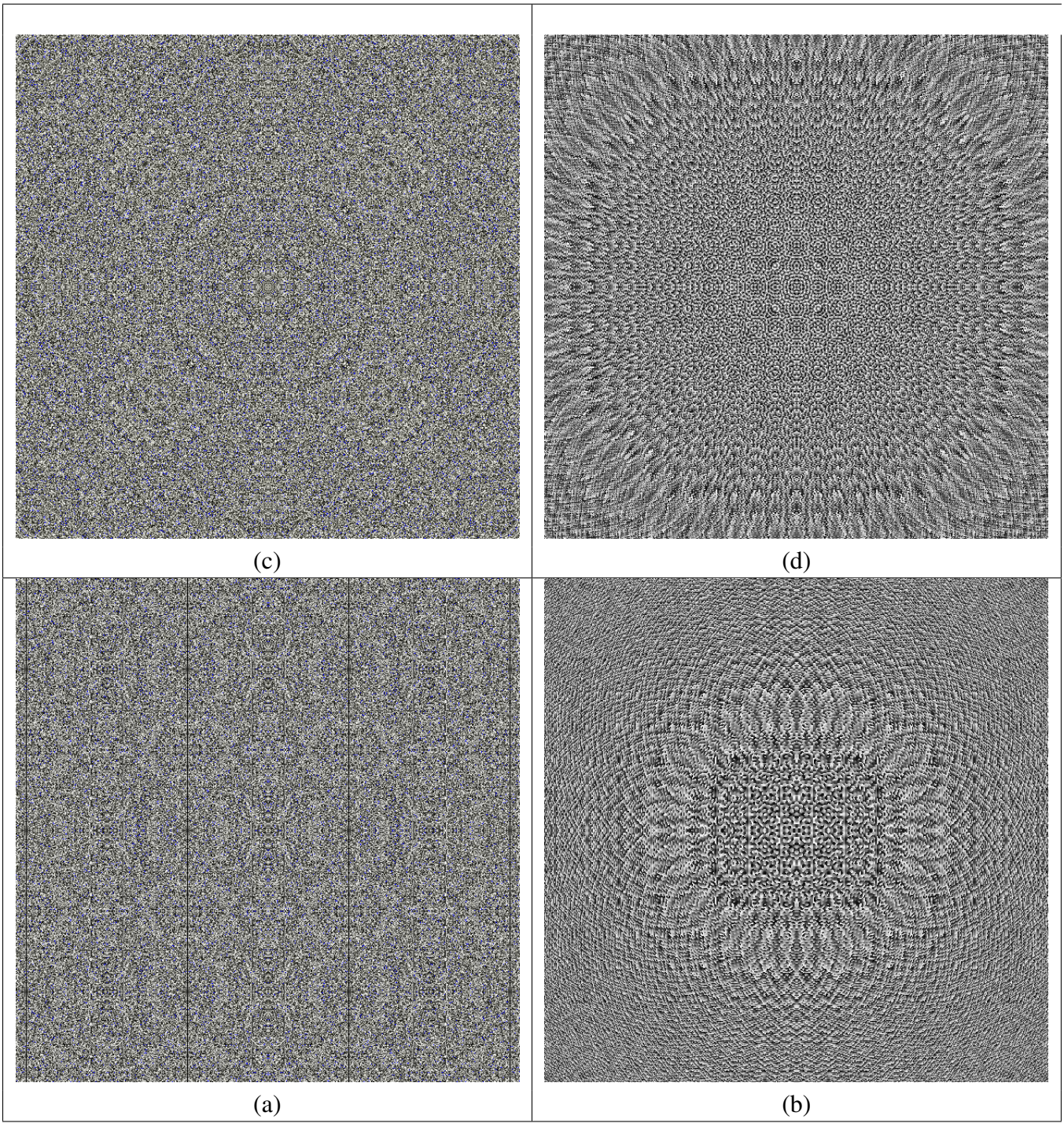


Figure C.2: Amplitude and phase map for 600x600 pixels per inch holograms created from images in A.1. (a) Amplitude map for A.1(b). (b) Phase map for A.1(b). (c) Amplitude map for A.1(c). (d) Phase map for A.1(c). Images are scaled 0.4 times of their original size.

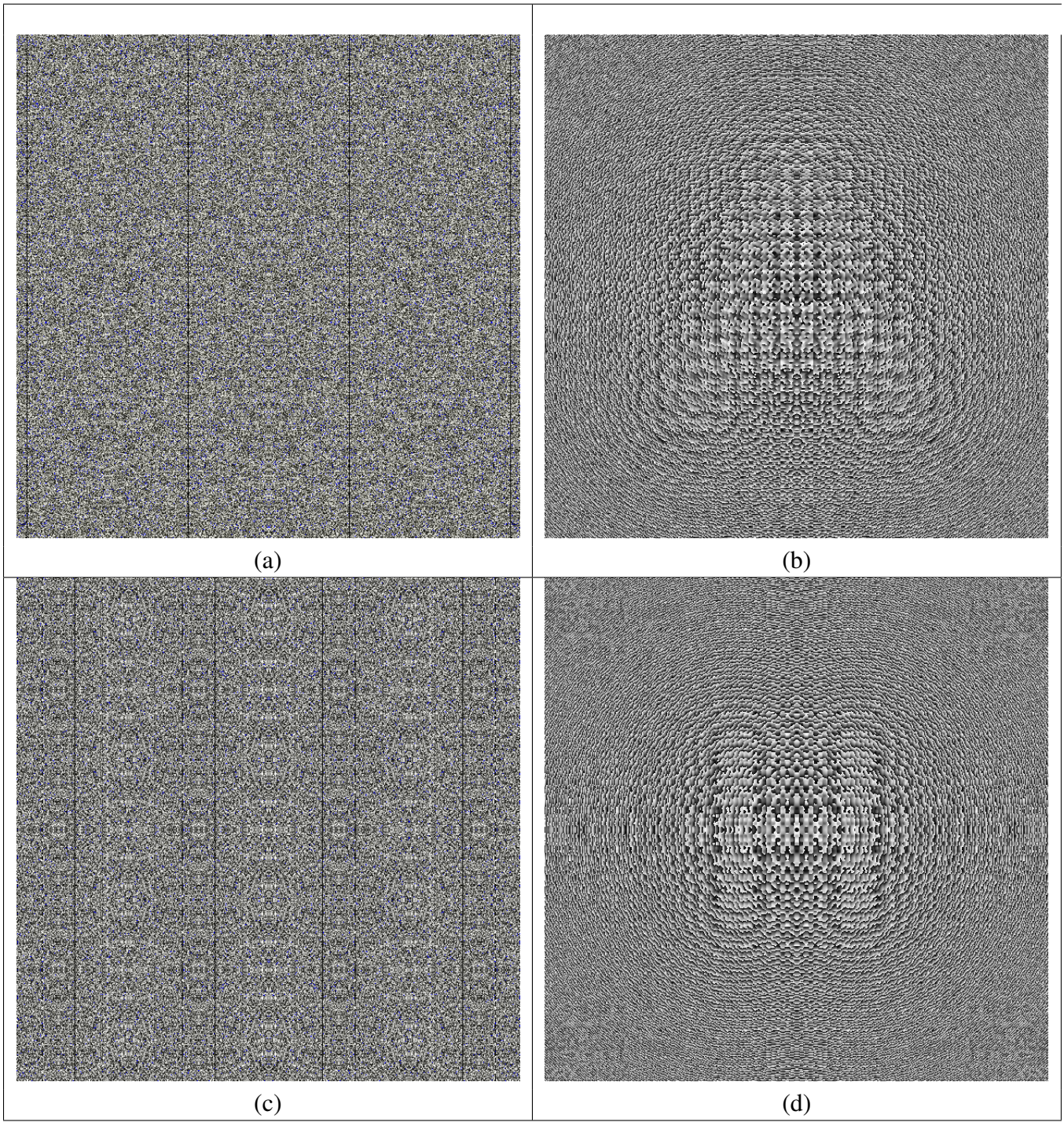


Figure C.3: Amplitude and phase map for 600x600 pixels per inch holograms created from images in A.1. (a) Amplitude map for A.1(d). (b) Phase map for A.1(d). (c) Amplitude map for A.1(e). (d) Phase map for A.1(e). Images are scaled 0.4 times of their original size.

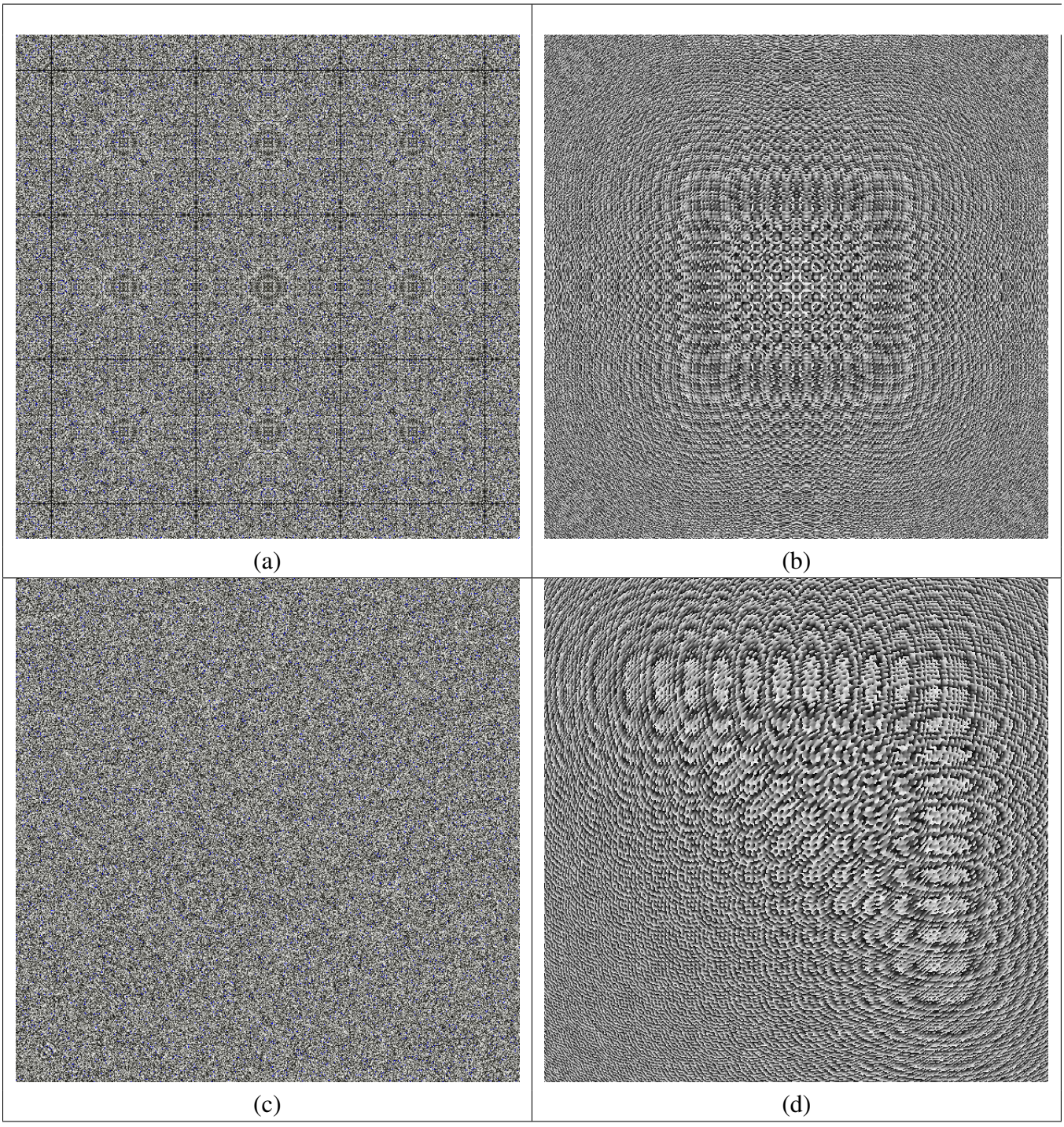
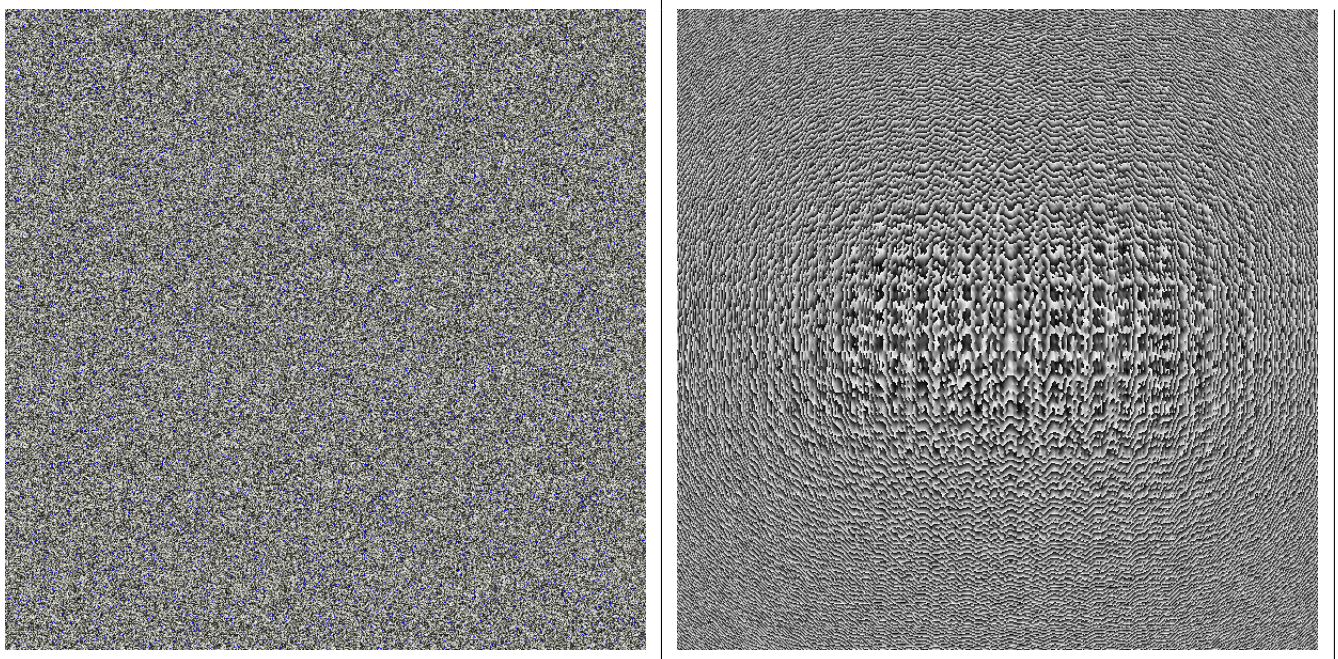


Figure C.4: Amplitude and phase map for 600x600 pixels per inch holograms created from A.1 and A.3. (a) Amplitude map for A.1(f). (b) Phase map for A.1(f). (c) Amplitude map for A.3. (d) Phase map for A.3. Images are scaled 0.4 times of their original size.

C.2 Amplitude and Phase Maps for Complicated Images



(a) (b)

Figure C.5: Amplitude and phase map for 600x600 pixels per inch holograms created from images in A.2. (a) Amplitude map for A.2(a). (b) Phase map for A.2(a). Images are scaled 0.4 times of their original size.

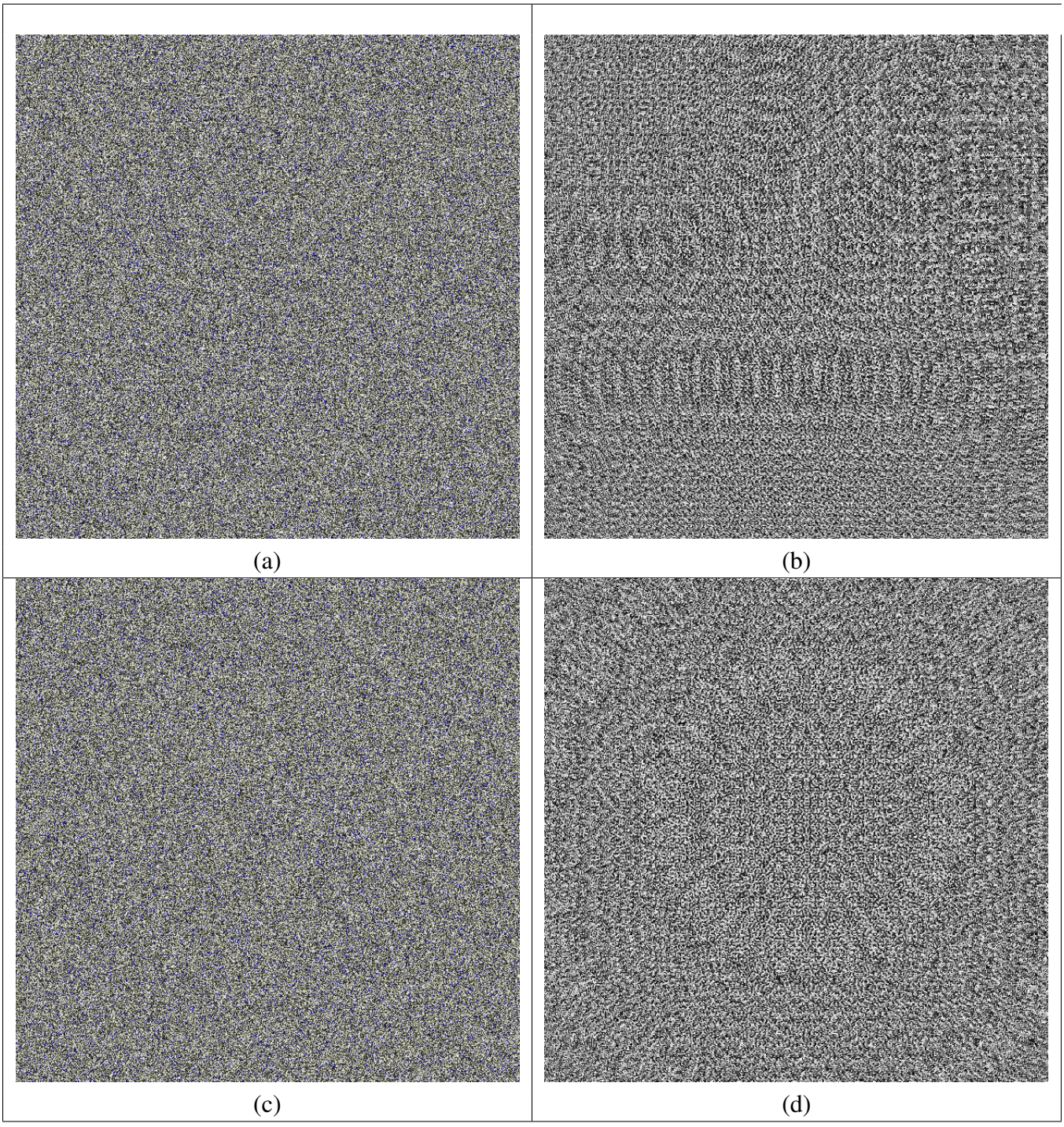


Figure C.6: Amplitude and phase map for 600x600 pixels per inch holograms created from images in A.2. (a) Amplitude map for A.2(b). (b) Phase map for A.2(b). (c) Amplitude map for A.2(c). (d) Phase map for A.2(c). Images are scaled 0.4 times of their original size.

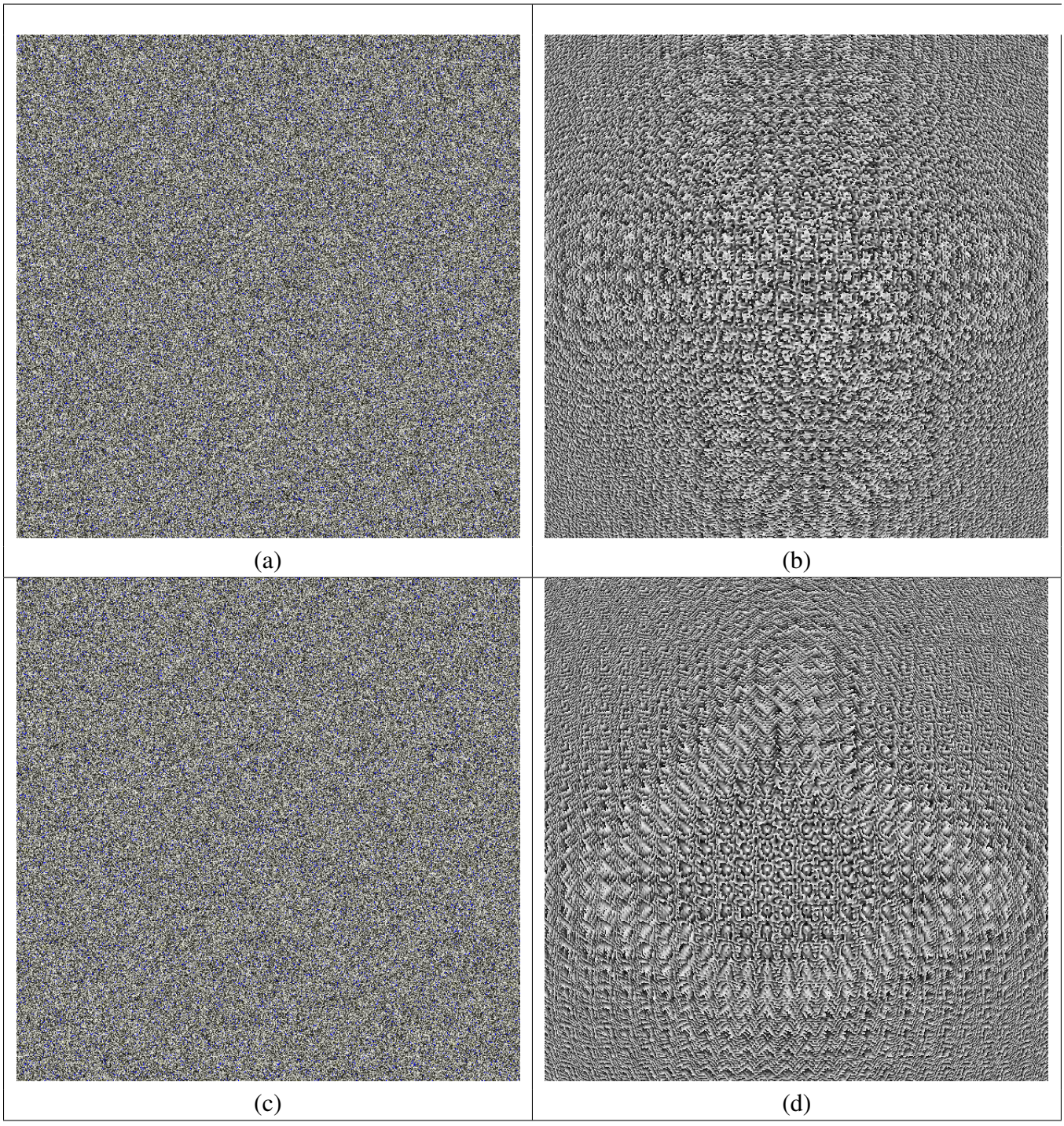


Figure C.7: Amplitude and phase map for 600x600 pixels per inch holograms created from images in A.2. (a) Amplitude map for A.2(d). (b) Phase map for A.2(d). (c) Amplitude map for A.2(e). (d) Phase map for A.2(e). Images are scaled 0.4 times of their original size.

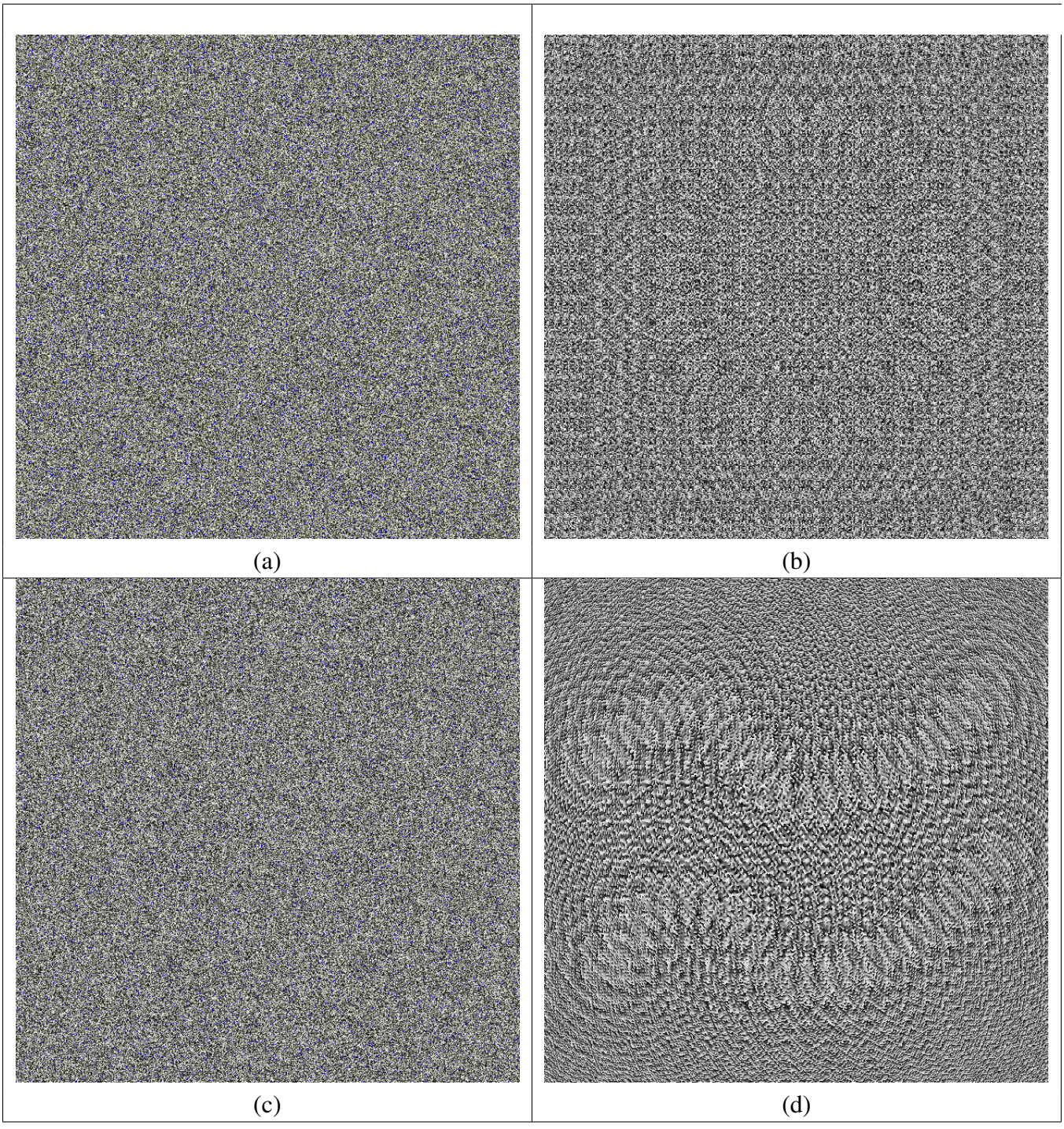


Figure C.8: Amplitude and phase map for 600x600 pixels per inch holograms created from A.2. (a) Amplitude map for A.2(f). (b) Phase map for A.2(f). (c) Amplitude map for A.2(g). (d) Phase map for A.2(g). Images are scaled 0.4 times of their original size.

Curriculum Vitae

Name:	Andrew Chan
Post-Secondary Education:	University of Western Ontario London, ON September 2009 - May 2011 M.Sc. (Computer Science) *Supervisor:* Pr. Stephen M. Watt University of Western Ontario London, ON September 2002 - June 2009 Honours B.Sc. (Computer Science - Minor in Game Development)
Honours and Awards:	2003, 2006 Dean's Honour List
Related Work Experience:	Teaching and Research Assistant Department of Computer Science The University of Western Ontario London, ON September 2009 - January 2011 Software Development Intern IBM Software Lab Markham, ON October 2007 - September 2008