Towards Quantum Computer Generated Holography

Master in Optics and Quantum Information

Author: Daniele Giunchi

Supervisor: Prof. Fabio Antonio Bovino

A.A. 2021-2022

Contents

Computer-generated holography (CGH) generates holograms using numerical algorithms, which have been extensively researched and applied in various fields. This thesis investigates CGH and introduce a new approach called quantum computer-generated holography (QCGH), exploring how quantum computing principles, algorithms, and potential applications can be applied to CGH. After reviewing the basics of holography, including analogue and digital hologram recording and reconstruction methods, we examine the numerical algorithms used in CGH. The concept of QCGH is then introduced, highlighting its potential advantages over traditional CGH. We propose a quantum version of Gerchberg-Saxton algorithm as a novel solution for QCGH, taking advantage of quantum computing's properties. As first example of QCGH, we detail the reasoning behind the implementation of the quantum circuit. Finally, the potential applications of QCGH in fields such as 3D display technologies, holographic data storage, optical microscopy, and remote sensing are discussed, considering ongoing advances in quantum computing hardware and potential algorithmic improvements.

Chapter 1

Introduction

Holography is a process for capturing and reconstructing three-dimensional (3D) images, resulting in a more realistic and immersive visual experience than conventional 3D rendering. Its creation dates back to 1947 by Dennis Gabor [GABOR, 1948]. Since then, it has developed and branched out into other sectors, including computer-generated holography (CGH), which creates holograms devoid of the usage of physical objects by using numerical algorithms. Various domains, including optical microscopy, metrology, data storage, and display technologies, have benefited substantially from CGH research [Goodman, 2005, Benton and Bove Jr, 2008, Poon, 2006]. In this work, we propose to pave the way for a new field, quantum computer-generated holography (QCGH), that results from the combination of CGH and the development of quantum computing. This thesis examines the principles, methods, and applications of CGH and QCGH, giving information about their effectiveness and prospective future developments. To process information in ways that classical computers cannot, quantum computing uses quantum physics concepts like superposition and entanglement [Nielsen and Chuang, 2002]. We will first go through the basics of holography, including how to record and rebuild holograms using analogue and digital methods. The fundamentals of CGH are next discussed, with an emphasis on the fundamental numerical hologram generation techniques, such as the Gerchberg-Saxton (GS) algorithm and Stochastic gradient descent (SD) algorithm [Gerchberg, 1972, Yang et al., 1994, Chen et al., 2019]. We also examine these algorithms' challenges, including the demands on memory and processing power and the trade-offs between image quality and computational effectiveness. Then, we discuss the idea of QCGH and consider any benefits it might have over conventional CGH. By utilising the capabilities of quantum computing and addressing several drawbacks present in traditional CGH algorithms, QCGH may enable quicker and more effective hologram creation. We suggest the quantum Gerchberg-Saxton (QSG) algorithm as the first algorithm candidate for QCGH. The QSG is based on a quantum adaptation of the GS that has been altered to benefit from the unique features of quantum computing, such as quantum parallelism and exponential speedups [Shor, 1999, Grover, 1996]. We create the quantum circuit detailing the role of each section and run an initial version on a quantum simulator. We will delve into the model and the role of the different entities in CGH, to establish a mapping between such entities and quantum circuit components. In our final section, we discuss other fields in which holography impacts (and so quantum holography), such as optical microscopy, holographic data storage, remote sensing, 3D display technologies. We also consider the future of QCGH in the context of ongoing developments in quantum algorithmic advancements such as quantum machine learning (QML) approaches.

1.1 Background

Dennis Gabor invented holography(Figure 1.1), a technique that captures and reproduces an object's three-dimensional (3D) structure, in 1947 to improve electron microscopy [GABOR, 1948]. However, holography did not find practical uses until the invention of the laser in 1960, such as in data storage and 3D displays with the work of Leith et al. [Leith and Upatnieks, 1964]. The invention of digital holography techniques in the 1960s pushed its possibilities even further, allowing computers to generate holograms(Goodman et al. [Goodman, 1967]). CGH has become an essential tool in various applications, including microscopy, metrology, and 3D displays [Poon, 2006]. Conversely, recently, a quantum approach was described to do holography [Töpfer et al., 2022], and quantum principles were applied to holography [Abouraddy et al., 2001, Sokolov et al., 2001, Defienne et al., 2021]. Continuous advances in computer capacity and the pursuit of more efficient and precise methods for creating holo-

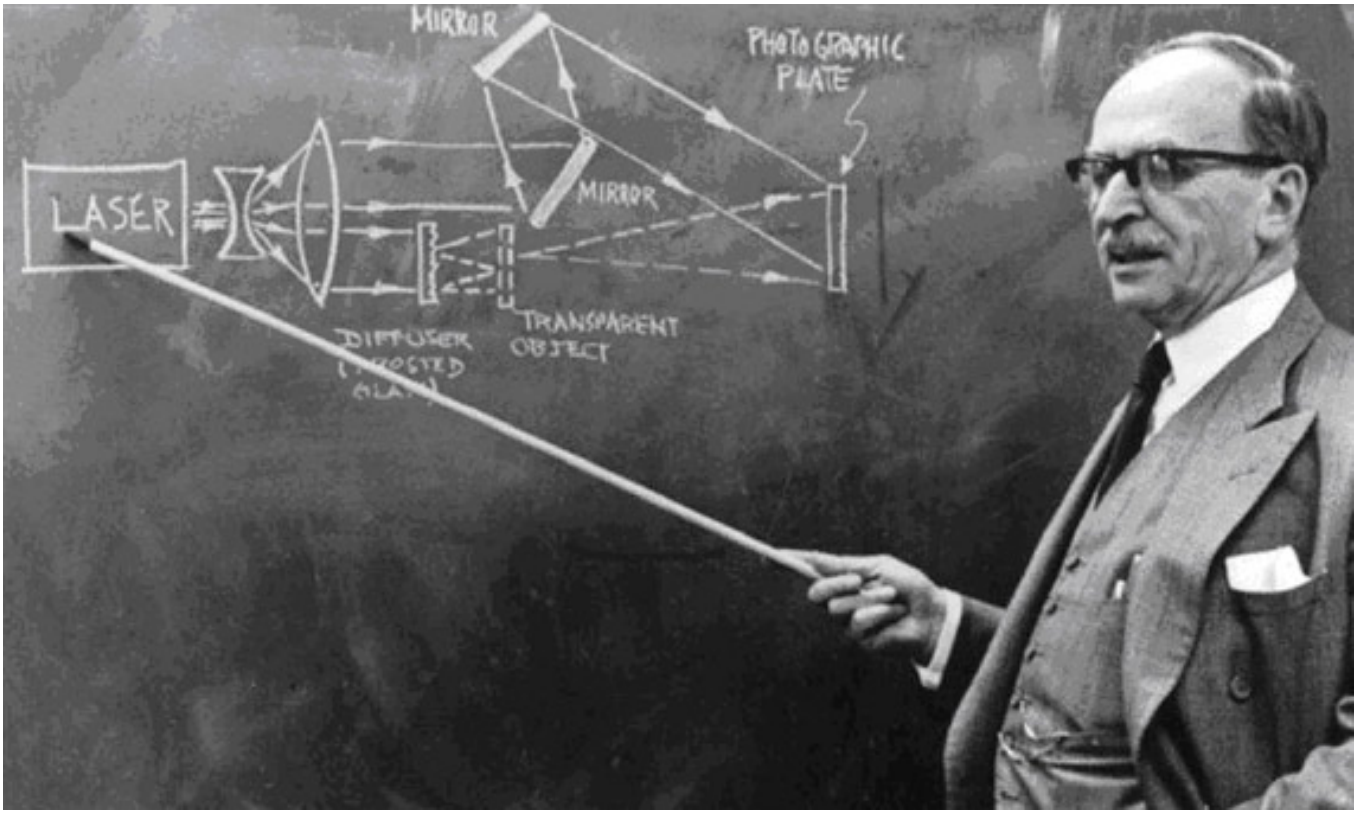

Figure 1.1: Dennis Gabor explaining an off-axis hologram. Credit: AIP Emilio Segrè Visual Archives, Physics Today Collection.

grams have marked the growth of CGH algorithms [Matsushima, 2020]. The introduction of the Fourier transform holography technique [Waters, 1968, Lu, 1968], the development of iterative phase retrieval algorithms (for example Gerchberg-Saxton [Gerchberg and Saxton, 1973]), and the introduction of wavefront encoding techniques([Brown and Lohmann, 1969, Lee, 1978]) were all significant milestones in the development of CGH algorithms. In recent years, quantum computing, a breakthrough technique that uses quantum mechanics principles to tackle complicated problems quicker and more efficiently than conventional computers, has shown in [Nielsen and Chuang, 2001]. To our knowledge, no approach for CGH uses quantum computing principles. As quantum computing has advanced, quantum algorithms have begun to appear, potentially overcoming some of the constraints of classical algorithms, and QCGH would use quantum parallelism and entanglement, potentially leading to faster computing times as theorized in computer graphics [Lanzagorta and Uhlmann, 2005a, Lanzagorta and Uhlmann, 2005b] and image processing [Caraiman and Manta, 2009, Ruan et al., 2021, Beach et al., 2003].

1.2 Objectives

This thesis builds upon the foundational principles of CGH and seeks to create a groundbreaking design for an algorithm for QCGH. Our research focuses on identifying how quantum computing can drive advances in holography. We outline our precise goals for this study as follows:

1. Literature Review: a thorough assessment of the body of knowledge on holography, CGH and quantum computing should be conducted to lay a firm foundation for understanding these topics. This review will examine our study's theoretical underpinnings, methods, and relevant works.

2. Algorithm Design: create a quantum algorithm based on iterative phase retrieval algorithms Gerchberg-Saxton. The holographic computation method must incorporate quantum computing concepts like superposition and entanglement.

3. Algorithm Implementation: implement a quantum circuit that can be run and used to retrieve phase to follow the

same step as the Gerchberg-Saxton algorithm.

4. Discussion and Future Works: we highlight how we map the entities and procedures of holographic computation into their quantum counterpart and we depict the implications, advantages and next steps in the implementation of the proposed algorithm. We describe possible applications of QCGH in various industries, including data storage, imaging, metrology, etc.

1.3 Scope and Limitations

The design of a quantum algorithm for CGH is the main contribution of this research. The research will review the different CGH selecting the most suitable to be the basis for the quantum version. We discuss the implementation with limits and future works for the QCGH algorithm. The state of quantum computing technology currently is one of this study's limitations. It takes a quantum computer with enough qubits and gate operations to construct a quantum algorithm for the QCGH. Since few such quantum computers are now available, future developments in quantum computing will determine the algorithm's scalability and applicability. The intricacy of the QCGH algorithm design is another drawback. Creating a suitable algorithm necessitates a thorough knowledge of CGH and quantum computing, which may limit the research's generalizability to researchers with weak backgrounds in these areas. The research will only implement and run the QCGH algorithm using quantum computing simulators. These simulators may not precisely reflect how the algorithm performs on actual quantum computers, although they can nevertheless offer valuable insights into the program's performance. Despite the extensive scope of this study, a few limitations should be noted:

- The quantum algorithm created in this paper will be tested on a simulator, which might not accurately represent the complete spectrum of quantum computing hardware.
- The study might not be able to fully utilise the benefits of QCGH due to the present constraints of quantum computing technology, such as error rates and decoherence.
- The practical deployment of QCGH in real-world applications may confront difficulties, such as integrating quantum computing technology with existing optical systems and developing appropriate holographic materials.

Despite these limitations, this research aims to provide the groundwork for future research and development in QCGH and its possible applications.

Chapter 2

Literature Review

Holography is a method that makes it possible to capture three-dimensional images by keeping crucial details about the amplitude and phase of light waves. Physicist Dennis Gabor, who later won the Nobel Prize for his work on it in "A new microscopic principle" [GABOR, 1948], first proposed this approach in 1948. After that, this method has made significant strides in various fields, including medical imaging technology, businesses that use it to store massive amounts of digital data, and creative endeavours like art.

2.1 From Holography to Computer-Generated Holography

2.1.1 Principles of Holography

Holography is based on the principles of interference, diffraction, and coherence of light waves. Unlike traditional photography, which only captures the amplitude of light waves, holography records both the amplitude and phase. This is achieved by generating an interference pattern between the reference wave, direct light from the source, and the object wave, which is light dispersed by the object, respectively [Leith and Upatnieks, 1965]. Object lighting, interference, and recording are essential to creating a hologram. Normally, when we illuminate an object, the light scatters and creates a wave containing object information (object wave); such wave is defined by the amplitude (intensity)and the phase containing shape information. When we take a photo, we record the variation of light intensity.

$$I(x, y) \propto |\psi_p(x, y)|^2 \tag{2.1}$$

where $\psi_p(x, y)$ is the two-dimensional complex amplitude. In the photograph, the information of phases is lost, shown also by Equation 2.1. Holograms are created by lighting an item with a coherent light source—typically a laser—divided into two beams by a beam splitter in the illumination step: the object beam and the reference beam. The object wavefront is formed by the light dispersed from the object after the object beam illuminates the object. The object wavefront is referenced in phase by the reference beam, which does not interact with the object [Hariharan, 1996]. The object and reference wavefronts merge and interfere with one another in the interference stage to produce an interference pattern. Bright and dark fringes make up this pattern, reflecting light waves' constructive and destructive interference. The interference pattern encodes the object wavefront's amplitude and phase data required for the 3D picture reconstruction. In the recording stage, the interference pattern is recorded on a photosensitive medium, such as a photographic film or a digital sensor. The hologram is permanently recorded due to the material's reaction to the interference pattern's light intensity. When lighted by an appropriate reference beam, this recording may be utilised to recreate the original 3D scene. The recorded hologram is lit by a reference beam that is the same as the one used for recording to rebuild the three-dimensional image. To recreate the original object wave, which then propagated to the observer's eye and creates a virtual image of the object, the interference pattern on the hologram diffracts the reference beam [Hariharan, 1996]. The fact that the recorded hologram contains both the amplitude (brightness) and phase information of the light dispersed from the object is a critical characteristic of holography. When the hologram is lit with the proper reference beam, which is necessary since this phase information stores the depth and spatial layout of the item, the original 3D picture may be recreated.

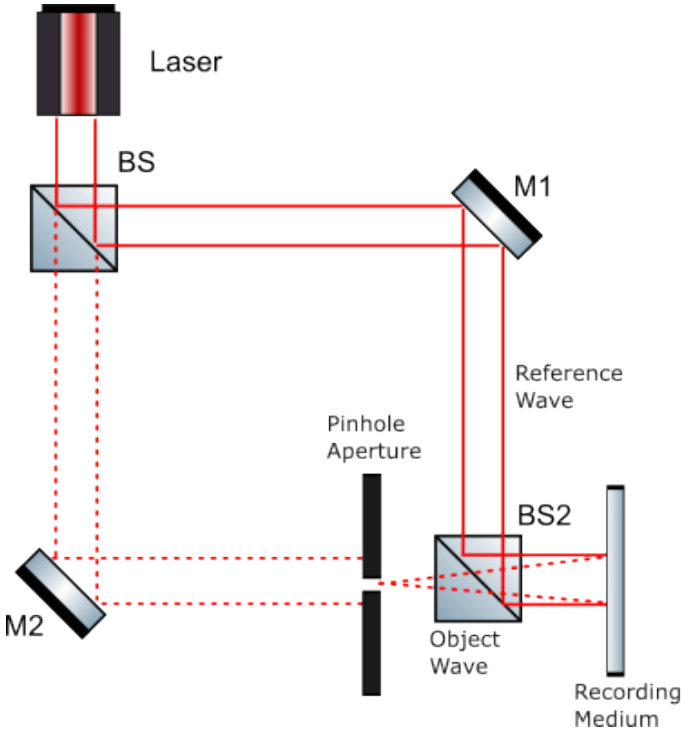

Figure 2.1: The basic principle of holography. A laser light illuminates an object, and a diffraction pattern is recorded when it interferes with the reference ray. Using the reference light and the interference pattern, we can reconstruct the 3D image of the object.

The hologram can generally be explained by recording each point that constitutes the entire object. For example Figure 2.1 shows how a laser is split into two waves by a beamsplitter, and then recombined by the use of mirrors and another beamsplitter. One ray (object wave) will hit the pinhole representing the object, and the other will hit the recording medium (reference wave). If ψ_0 is the object wave's field distribution and ψ_r is the reference, we will record the interference between them $|\psi_0 + \psi_r|^2$ in the medium. Since the pinhole aperture can be modelled by a $\delta(x, y)$, by Fresnel diffraction, we have on the medium an object wave known as paraxial spherical wave(Equation 2.2).

$$\psi_0(x, y; z_0) = \delta(x, y) * h(x, y; z_0)) = e^{-jk_0 z_0} \frac{jk_0}{2\pi z_0} e\left[-\frac{jk_0}{2z_0}(x^2 + y^2)\right] \tag{2.2}$$

As the reference wave propagates, maintaining the phase

$$\psi_r = ae^{-jk_0 z_0} \tag{2.3}$$

Giving an hologram with amplitude:

$$|\psi_r + \psi_0|^2 = \left|ae^{-jk_0 z_0} + e^{-jk_0 z_0} \frac{jk_0}{2\pi z_0} e^{\left[-\frac{jk_0}{2z_0}(x^2+y^2)\right]}\right|^2 \tag{2.4}$$

Such an equation can be simplified from complex to real if we have a real hologram and the resulting formula (not reported here) is a sinusoidal function called Fresnel zone plate (FZP), that is, the hologram with a distance z_0 from the recording medium. Studying FZP, we are able to identify the local fringe frequency and notice its increase with the spatial coordinate. Thus, local frequencies have information on z, so from it we can deduce the distance of the point source. Many hologram forms have arisen depending on the locations, quantities, and angles of the reference and object beams utilised in holographic recording. Different holograms may exist, including reflection, transmission, single beam, off-axis, and on-axis. Holograms can be categorized into two categories: phase holograms [Zhang and Yamaguchi, 1998, Yamaguchi and Zhang, 1997] and amplitude holograms [Wyrowski, 1990].Phase holograms modify the phase of the reference beam, whereas amplitude holograms alter the amplitude of the reference beam during reconstruction. Although a 3D picture may be reconstructed using any form of hologram, phase holograms frequently provide

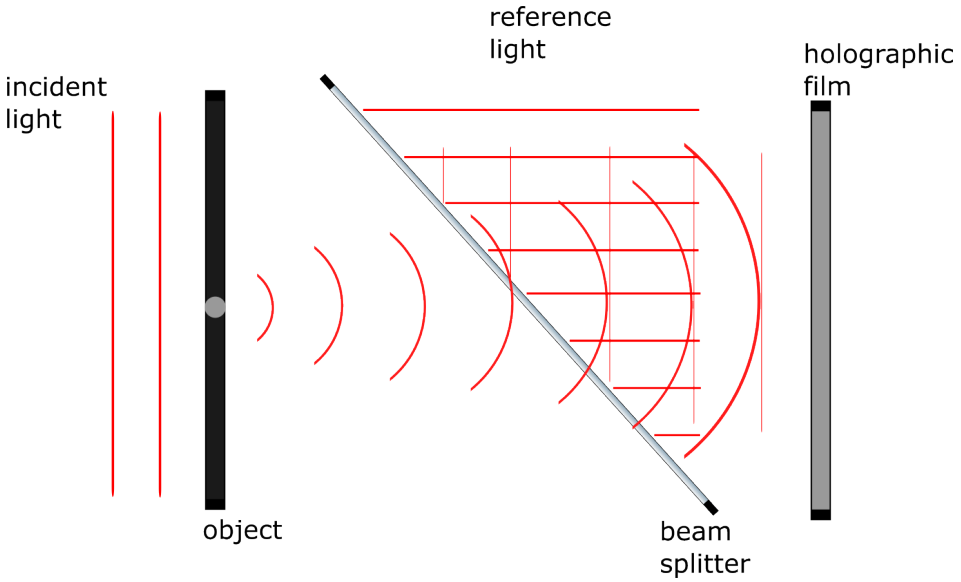

Figure 2.2: Off-Axis Holography principle.

higher image quality and less noise. Numerous uses for holography have been discovered in several industries, including interferometry, microscopy, entertainment, and data storage [Coufal et al., 2000]. Holography has been employed in the arts and entertainment to produce eye-catching 3D pictures and displays for exhibits, concerts, and performances [Matthews and Nairn, 2023]. Holographic data storage systems have been created to store vast amounts of data in a small footprint, potentially outperforming more conventional storage approaches [Ashley et al., 2000]. Holographic methods in microscopy have made it possible to acquire 3D data from biological samples with great resolution and little sample damage [Vanligten and Osterberg, 1966, Bally, 2013, Simon et al., 2008]. Holography has been used in interferometry to examine how things deform and move under various circumstances, revealing important details about the mechanical behaviour of the materials [Ostrovsky and Butusov, , Heflinger et al., 1966, Rastogi, 2013].

2.1.2 Holograms Classification

2.1.2.1 Off-axis vs On-axis Holography

On-axis and off-axis holography are two types of holography that may be distinguished depending on the angle of the object and reference beams. The object and reference beams in on-axis holography have a null angle between them. In contrast, off-axis holography creates a fringe pattern in the hologram by slightly angling the reference and object beams.

Off-axis holography As shown in Figure 2.2, off-axis holography is based on the interference of two coherent light beams. One interacts with the object of interest (object beam) while the other remains undisturbed (reference beam). Off-axis holography assures the formation of unique, non-overlapping spectral components in the Fourier domain by adding an angle between these beams. This spatial separation allows for extracting complicated wavefront data without needing phase-shifting or several exposures, which speeds up the holographic process. Off-axis holography uses a laser source to provide the requisite coherent light beams. These beams are divided by a beam splitter, which angles the reference beam to the object beam. Following their interaction with the object, both beams interfere with a digital recording medium, such as a charge-coupled device (CCD) or a complementary metal-oxide-semiconductor (CMOS) sensor, resulting in an off-axis hologram. Following that, sophisticated digital processing techniques like Fourier transformation and filtering allow for separating and reconstructing the necessary complex wavefront.

On-axis holography On-axis holography is when the object and reference waves have a zero angle, as shown in Figure 2.3. Even when no object is in view, off-axis holography generates fringe patterns, commonly cosine fringe patterns. These fringe patterns record the object's phase and may be utilised to extract the necessary term from the

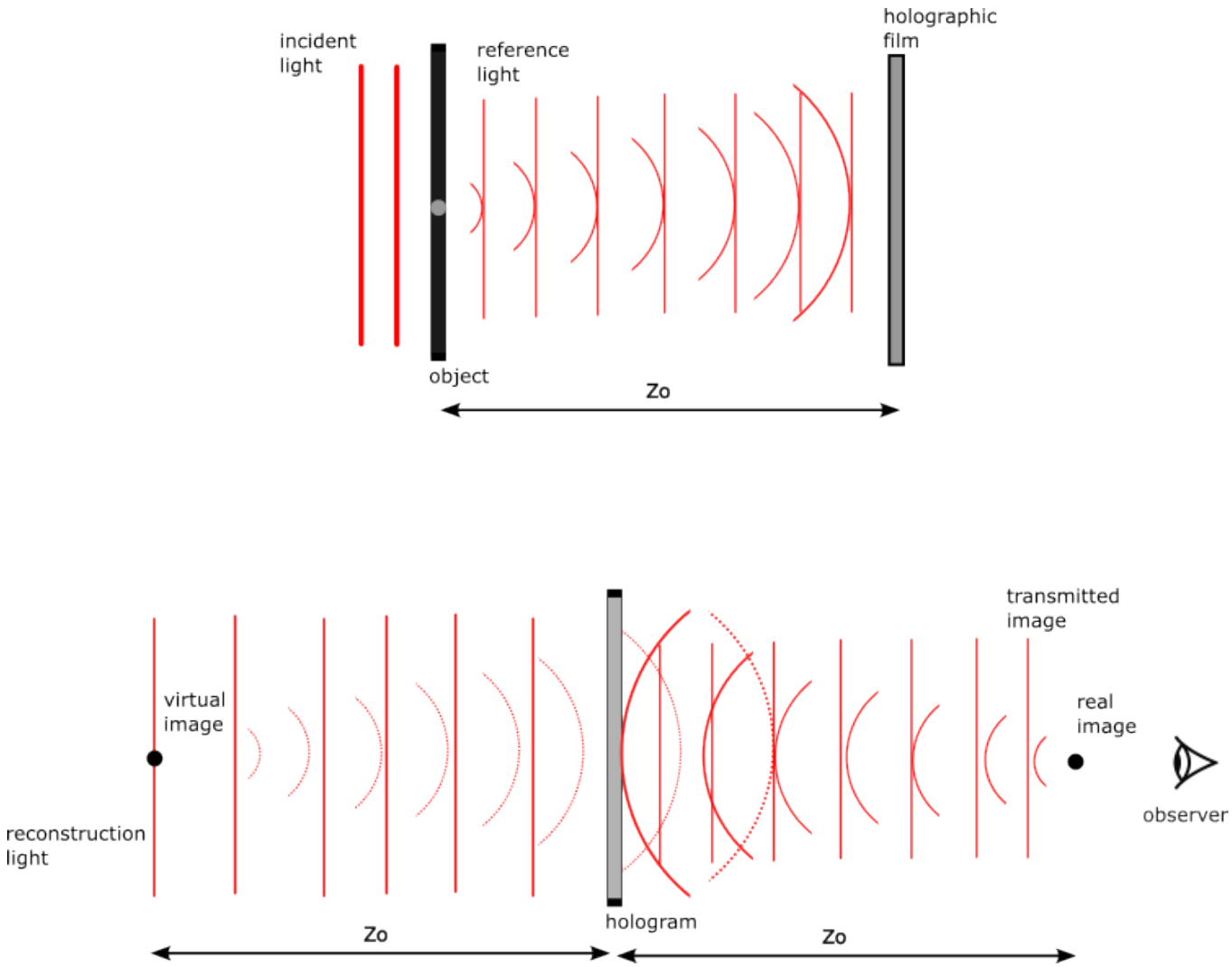

Figure 2.3: On-Axis Holography principle.

hologram. A Fourier transform is performed on the hologram to digitally isolate the required term, which changes the multiplication into correlation owing to the Fourier transform features. The hologram's Fourier transform has several orders of the term, with the first two being the DC terms placed at the original Fourier transform plane. The carrier frequency may be eliminated by cropping and centring the plus one order term, and an inverse Fourier transform can be used to return to the picture plane, resulting in the required term multiplied by the square of the reference wave.

2.1.2.2 Image Holography

An image hologram is a particular kind in which the object beam is made up of the light emanating from an actual picture of the item produced by a lens. This allows for recording the interference pattern between a reference beam and a genuine picture. An image hologram's light field dispersion during recording may be written as follows:

$$t(x,y) = |\psi_i(x,y) + \psi_r e^{j*k_0*sin\theta*x}|^2 \tag{2.5}$$

Where $\psi_r e^{j*k_0*sin\theta*x}$ is the off-axis reference beam, and $\psi_i(x,y)$ is the complex field of the picture generated on the hologram plane. The genuine picture is recreated onto the hologram plane using the same reference beam orientation. As chromatic aberration is reduced when the picture is on the hologram plane, this enables the reconstruction of image holograms using white light. Additionally, the twin image is rebuilt on the holographic plane. Therefore, off-axis recording is required to distinguish between the twin picture and the intended reconstructed image.

Diffraction calculations can be used to model the reconstruction of an image hologram while simulating the image hologram. The actions are:

1. Fresnel diffraction models the desired object and determines the field distribution at the hologram plane $\psi_i(x, y)$.
2. To mimic the recording process, add an off-axis reference beam using the formula $\psi_r e^{j*k_0*sin\theta*x}$.
3. Use Fresnel diffraction for several wavelengths to propagate the hologram field to the reconstruction distance during reconstruction.
4. To create a white-light reconstruction on the holographic plane, add the monochromatic reconstructions.

This basic simulation may be expanded to incorporate twin images and various diffraction orders, among other effects. Reconstruction will demonstrate that due to little chromatic aberration, the genuine picture sharpness is retained on the hologram plane even under polychromatic light.

2.1.3 Digital Holography

Given that the recording strategies are the same, the main distinction between optical and digital holography is replacing the recording medium with an electrical device, such as a charge-coupled device (CCD). To rebuild the hologram, optical interference fringes captured by the CCD are converted into a two-dimensional digital signal and then processed digitally. Hologram construction and reconstruction may be fully numerically simulated using digital techniques. A display device can then receive the hologram for optical reconstruction. To replicate diffraction between the hologram and the diffraction plane or the observation plane in each of these scenarios, we must first model interference between the object and the hologram at the front end of the process. Digital holography has the benefit of focusing, which allows the 3D scene to be rebuilt plane by plane inside the computer. Because multiple planes of the item may be visualised and analysed independently, a more complete and accurate reconstruction of the 3D object is possible. The focusing effect is especially effective when the item has complex or variable forms, allowing for greater visualisation and comprehension of the object's structure and features. The capacity to record dynamic or time-varying objects in real-time is a significant advancement in digital holography. Microscopy, biological imaging, and industrial inspection all need the gathering and analysis of dynamic 3D situations. In addition, it is possible to handle moving holograms and real-time or near-real-time reconstruction of 3D scenes. Employing improved algorithms and computational approaches for hologram processing and picture reconstruction is another key achievement in digital holography. Among these approaches are iterative, phase retrieval, and wavefront shaping algorithms. These algorithms and approaches increase hologram quality and accuracy, speed up the reconstruction process, and allow for correcting aberrations and distortions in holographic pictures—digital holography impacts the hologram recording and reconstruction phase.

2.1.3.1 Hologram Recording

The interference pattern is immediately recorded using a digital camera in digital holography. The strength of the interference pattern is captured by the digital camera and turned into a digital hologram. The digital hologram provides information on the object wavefront's complex field, including amplitude and phase. The recorded hologram is commonly saved as a two-dimensional array of complex integers, with each element representing a hologram pixel. The dynamic range and resolution of the digital camera used to record the hologram are critical in determining the quality of the reconstructed hologram.

2.1.3.2 Hologram Reconstruction

In digital holography, optical hologram reconstruction entails lighting the recorded hologram with a coherent laser beam that acts as the reference wave. The hologram diffuses the incident light, reconstructing the object wavefront and creating a three-dimensional representation of the object. Several methods, such as holographic displays [Yaraş et al., 2010] or spatial light modulators [Efron, 1994, Casasent, 1977], can be used to visualise the rebuilt image as shown in Figure 2.4 and 2.5.

In digital holography, computational holographic reconstruction entails utilising computational tools to analyse the recorded hologram and retrieve the three-dimensional picture of the item. Methods that perform Fourier transforms, filtering, and numerical propagation(an example of propagation is 2.6) can be used to compute the object wavefront from the recorded hologram. The rebuilt picture can be viewed on a computer screen or other digital device.

Figure 2.4: Hologlass, a product from Looking Glass Factory (Courtesy from Looking Glass Factory)

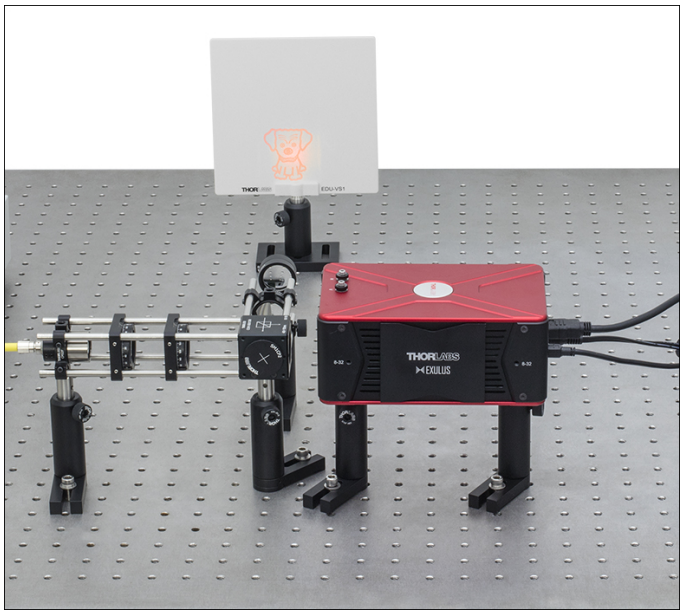

Figure 2.5: Spatial Light Modulators serve to change the phase of light (Courtesy from ThorLabs).

$$u(x,y) = \int_0^{2\pi} \int_0^{\infty} u_0(x,y) \frac{e^{ikr}}{r} \cos(\theta) dx dy \tag{2.6}$$

2.2 Computer Generated Holography

Making holograms with computers is called computer-generated holography, or CGH. Calculating the interference pattern between a reference wave and an object wave produces CGH holograms. A spatial light modulator (SLM) is used to show the digital picture that was created from the interference pattern. The SLM has a hologram of the item by diffracting the light. Compared to conventional holography, CGH provides several benefits. CGH holograms may be constantly changed and made in any size and form. CGH holograms can also be utilised to produce holograms of things that aren't there.

Holographic light transport (HLT), a method for mimicking the movement of light in holographic systems, is the basis for CGH. HLT is based on the wavefront reconstruction principle, which asserts that a wavefront's amplitude and phase at any location in space may be calculated given those values at two separate planes. HLT makes it possible to create holographic displays, develop novel holographic algorithms, and research holographic systems. Modelling

the intricate interactions between light and matter in holographic systems is one of the trickiest problems in HLT. Diffraction, reflection, and refraction are possible components of these interactions. HLT algorithms need to be able to represent these interactions to create precise, realistic simulations. The computational complexity of the issue is another challenge with HLT. Solving many differential equations is required to mimic light transmission in a holographic system. HLT algorithms must be effective and scalable to be used in real applications.

2.2.1 Early steps of CGH

2.2.1.1 The Detour-Phase Hologram

The detour-phase hologram is a milestone of CGH that was developed in the 1960s [Brown and Lohmann, 1966, Lohmann and Paris, 1967, Brown and Lohmann, 1969] before the availability of grey-scale displays. It allowed the reconstruction of holographic images from binary (opaque or transparent) holograms. The hologram is divided into an array of cells, each containing a simple binary pattern. Such a pattern in each cell controls the amplitude and phase of the light passing through. Amplitude control is achieved by varying the transparent window size in each cell while Phase control is achieved by slightly shifting the position of each window, causing a "detour phase". The complex hologram field is discretized into cells to generate a detour-phase hologram, with amplitude and phase values per cell. Then, the amplitude is represented by the window size in each cell. The phase is represented by shifting the window proportionally to the desired phase value. The window shifting introduces an optical path length difference, creating the detour phase. The binary cells are tiled to create the full hologram pattern. Upon illumination, each cell diffracts light with the encoded amplitude and phase. The aggregate diffraction pattern reconstructs the holographic image despite using only a binary hologram. The detour-phase method was an early clever technique to work around limitations in hologram display technology. It enabled reconstructing holographic images from binary patterns that could be printed. However, the diffraction efficiency is low compared to true gray-scale holograms.

2.2.1.2 The Kinoform Hologram

The kinoform hologram [Lesem et al., 1969] is a type of CG hologram that records only the phase information of a complex hologram. Kinoforms allowed the reconstruction of holographic images more efficiently than prior binary holograms. In a kinoform, the amplitude component of the hologram's complex field is ignored and set to a constant value. Only the phase is retained and encoded into a surface profile that modulates the phase of the incident light. This is based on the concept that randomly scrambling the amplitude while retaining the original phase allows recognisable reconstruction for many objects.

The kinoform hologram can be expressed as the simple formula:

$$Hp(x, y) = e^{-j\phi(x,y)} \tag{2.7}$$

with $\phi(x, y)$ that represents the phase component of the complex hologram field. Early kinoform holograms were generated optically by recording the interference pattern of an object and reference beam on a photographic plate. After development, the plate was bleached to convert the amplitude hologram into a phase hologram.

For computer generation, the process is simplified. The spectrum of the target object is computed. Then the phase is extracted and used as the kinoform pattern. This phase-only hologram can be displayed on a phase-modulating SLM.

The benefit of the kinoform is greatly improved diffraction efficiency compared to binary holograms. In principle, all of the incident light is directed into the reconstructed image, rather than wasted in a zero-order beam. However, the reconstruction quality depends on the object having a fairly uniform spectrum amplitude.

2.2.2 Algorithms for Computer-generated holography

The computer techniques used to make holograms have evolved alongside the discipline of holography. CGH has evolved as a dominant technique for producing holograms with the arrival of powerful computers and complex algorithms [Hirsch et al., 1971]. This section digs into the basics of computer-generated holography, focusing on the algorithms that create very realistic three-dimensional images. The act of recreating the interference pattern of light waves that would occur if a coherent light source lighted an item is at the heart of CGH. The necessity to properly represent wavefronts and their interactions while minimising computer complexity and resource needs drove the creation of CGH algorithms. Several approaches have been devised and modified to meet these issues. The Fourier Transform

approach, which relies on the inherent link between a hologram's spatial and frequency domains, was one of the early methods for CGH. The FT technique produces a frequency domain hologram by calculating the Fourier transform of the object's complex amplitude distribution. To address the limitations of the FT, researchers created several iterative CGH algorithms. One such approach is the Gerchberg-Saxton algorithm [Gerchberg and Saxton, 1973], which iteratively refines the hologram by alternating between the object and hologram planes, enforcing constraints on the amplitude and phase of the light field. To increase its convergence and efficiency, the GS method has been expanded and changed multiple times [Zhao and Chi, 2020, Wu et al., 2021b].

Deep learning has lately opened up new opportunities for developing CGH algorithms. To learn the intricate mapping between object representations and their related holograms, neural networks have been used, resulting in efficient and accurate hologram synthesis [Eybposh et al., 2020, Horisaki et al., 2018, Wu et al., 2021a, Khan et al., 2021]. Although their scalability and generalisation capabilities are still under investigation, these deep learning-based algorithms have proven tremendous potential for real-time and high-resolution hologram creation. In the next sections, we detail various algorithms, noting their advantages, disadvantages, and applications. Finally, we examine the most recent advances in deep learning-based CGH approaches, highlighting their possible effect on the field's future.

2.2.2.1 Iterative Fourier Transform and Iterative Phase Retrieval Algorithms

The kinoform hologram introduced previously records only the phase information of a hologram's complex field. While this improves diffraction efficiency, noise often degrades the reconstructed image. The iterative Fourier transform algorithm (IFTA) provides a technique to optimize the phase-only kinoform to improve image quality. Iterative Phase Retrieval Algorithms (IPRAs) are a type of IFTA that focus on object's phase information from its intensity distribution. These techniques update the phase distribution iteratively until it converges to a solution that meets the stated criteria. IPRAs work based on an initial estimate of the phase distribution, which might be random or based on prior information about the object. The method then alternates between estimating the hologram's complex field from the phase distribution and updating the phase distribution based on the estimated complex field. This procedure is done until the phase distribution converges to a solution that meets the specified constraints. Iterative phase retrieval methods have received much attention due to their capacity to produce high-quality computer-generated holograms. By limiting the amplitude and phase of the light field in both the object and hologram planes, these methods seek to modify the phase distribution of a hologram iteratively. In this part, we describe the essential iterative phase retrieval algorithms, expanding on their working principles, computational complexity, processing time, and the approaches used in their creation. One of the first and most well-known iterative phase retrieval algorithms is the Gerchberg-Saxton [Gerchberg and Saxton, 1973]. It works by alternating between the object and hologram planes while placing amplitude limits on the wavefront in each cycle. The GS algorithm starts with an estimate of the complex amplitude distribution of the object and repeatedly refines the phase distribution in the hologram plane. The GS algorithm's principal strength is its simplicity; however, it converges slowly and may not always attain the global minimum of the error function [Fienup, 1982]. The author created the error reduction (ER) and input-output (IO) algorithms, which are modified versions of the GS technique, to solve the shortcomings of the GS algorithm. The ER algorithm tightly maintains amplitude limits, but the IO algorithm modifies the update rule to enhance convergence time. Both techniques performed better than the original GS algorithm regarding convergence rate and reconstruction accuracy. However, the computational complexity of holograms remains an issue, especially for large-scale holograms. The hybrid input-output (HIO) approach, which combines the capabilities of the ER and IO algorithms, is another notable improvement in iterative phase retrieval methods [Fienup, 1982]. In subsequent rounds, the HIO algorithm alternates between the ER and IO update rules to balance the benefits of stringent amplitude constraint compliance with quick convergence. Although the HIO approach has been successfully applied to various CGH applications, it may still suffer from local minima stagnation, demanding more algorithmic modifications and optimisations [Marchesini et al., 2003]. Researchers have created various modifications and variants of iterative phase retrieval methods to improve their performance in certain applications over the years. Among them are adaptive step-size approaches, which dynamically change the update rule to minimise stagnation and speed up convergence [Bauschke et al., 2002]. The number of iterations and the hologram size dictate the processing time of iterative phase retrieval techniques. Researchers have investigated parallel processing options such as putting algorithms on graphics processing units (GPUs) or field-programmable gate arrays (FPGAs) to speed up computing [Nishi et al., 2005, Lenart and Owall, 2005]. Despite advances in iterative phase retrieval techniques, finding the global minimum of the error function remains difficult because these algorithms are sensitive to initial condition selection and can become caught in local minima. Researchers

have examined global optimisation techniques, such as simulated annealing and genetic algorithms, to direct the search process towards the global minimum to overcome this constraint [Ramsey et al., 2005, Wen et al., 2005]. Iterative phase retrieval techniques have shown substantial promise for producing high-quality computer-generated holograms. These methods have achieved excellent reconstruction accuracy and resolution by iteratively refining the phase distribution of the hologram and setting limits on the amplitude and phase of the light field. On the other hand, their computational complexity and sensitivity to beginning circumstances remain issues that need continual study and optimisation.

Gerchberg–Saxton Algorithm The Gerchberg-Saxton (GS) algorithm is one of the most widely used IPRAs. The GS algorithm's fundamental steps are as follows:

1. Begin by calculating an initial estimate of the phase distribution, $\phi(0)$.
2. Using the Fourier transform, determine the complex field of the hologram, $E(0)$, using the predicted phase distribution.
3. Set the complex field's amplitude to the observed intensity distribution, $|E(0)| = \sqrt{I}$.
4. Using the inverse Fourier transform, calculate the updated estimate of the phase distribution, (1), from the updated complex field, $E(1)$.
5. Steps 2-4 must be repeated until the phase distribution converges to a solution that meets the set criteria.

Python implementation of the GS algorithm is the Appendix A.1

In this implementation, I represents the measured intensity distribution, $\phi(0)$ represents the starting estimate of the phase distribution, max_iter represents the maximum number of iterations, and tol represents the convergence tolerance. The function returns the rebuilt phase distribution, ϕ. NumPy and SciPy routines for Fourier transformations are used to build the approach. There are several alternative techniques for phase retrieval, each with its own set of pros and limitations: angular spectrum method (ASM), Fresnel transformation algorithm (FTA) and wavelet-based holography(WBH). ASM is based on 2D Fourier transform, while FTA calculates light diffraction with the wavefront that is not parallel to the hologram plane, by splitting into a set of plane waves that travel across the hologram. Finally, WBH describes a method for rebuilding holograms by extracting information using wavelet decomposition.

Computational Complexity and Processing Time The computational cost of Fourier-based techniques is determined by the Fast Fourier Transform (FFT) technology utilised to produce holographic patterns. The complexity of FFT is $O(NlogN)$, where N is the number of discrete points in the object wavefront. As a result, the amount of input data and available computer resources significantly impact processing time [Cooley and Tukey, 1965]. IFTA techniques sometimes need many optimisation algorithm rounds, including numerous FFT computations, increasing overall complexity. Despite this, the additional processing cost is typically outweighed by the improved diffraction efficiency and stability of phase-only holograms.

Techniques to Improve Processing Time and Efficiency Several ways to improve The processing speed and efficiency of Fourier-based algorithms for CGH have been studied. Among these techniques are:

- Parallel computing and GPU computing: Researchers have implemented Fourier-based approaches on Graphics Processing Units (GPUs) and other parallel architectures to take advantage of the parallel nature of FFT calculations [Moreland and Angel, 2003], resulting in a considerable decrease in processing time.
- Compressive sensing techniques have been used to minimise the number of samples required for hologram creation, lowering the computing complexity and processing time of Fourier-based systems [Rivenson et al., 2013].
- LUT (look-up table) methods: LUT-based approaches have been suggested to accelerate iterative optimisation algorithms by precompiling and storing complicated exponential values or phase increments, lowering trigonometric calculations during the optimisation process [Kim and Kim, 2008].

2.2.2.2 Other Techniques

Aside from Fourier-based approaches and iterative phase retrieval algorithms, various alternative techniques for CGH have been developed. These approaches are intended to overcome the shortcomings of the prior methods, such as computational complexity and processing time. This part will look at several alternatives, such as direct fringe pattern-generating methods and deep learning-based algorithms.

Deep Learning-based Algorithms Deep learning breakthroughs in recent years have opened up new opportunities for developing CGH algorithms. To learn the intricate mapping between object representations and their related holograms, neural networks have been used, resulting in very efficient and accurate hologram synthesis [Shimobaba et al., 2022, Rivenson et al., 2019]. Convolutional neural networks (CNNs) have been used to generate holograms, with the network learning the inverse process of light propagation [Horisaki et al., 2018] or phase-recovery and holographic image restoration [Rivenson et al., 2018], compression phase [Jiao et al., 2018] or end-to-end process [Eybposh et al., 2020]. CNN-based hologram creation has shown promise for real-time and high-resolution hologram generation while requiring substantially less computer complexity than existing approaches. However, extending these technologies to large-scale holograms remains a difficulty. For CGH, generative adversarial networks (GANs) have been used [Lee et al., 2018, Chen et al., 2018], in which the generator network generates holograms that are difficult for a discriminator network to differentiate from actual holograms. GAN-based approaches have shown potential in producing high-quality holograms with complicated object representations, but they are susceptible to training instability and mode collapse [Creswell et al., 2018].

Overview of Neural CGH Methods There are two approaches to neural CGH methods: data-driven and model-driven. Using vast datasets of known instances, data-driven techniques train neural networks to learn the mapping between the input data and the intended holographic patterns [Rivenson et al., 2019]. On the other hand, model-driven techniques incorporate CGH physical principles into neural network architecture, eliminating the requirement for large amounts of training data [Liu et al., 2023a, Zheng et al., 2023, Chen et al., 2021].

Data-driven Neural CGH To learn the mapping between input items and their associated holographic patterns, deep convolutional neural networks (CNNs) are typically utilised in data-driven neural CGH techniques [Shi et al., 2021]. With training data of matching holographic patterns and pairs of input objects, these networks are taught using supervised learning techniques. Fully convolutional networks (FCNs), generative adversarial networks (GANs), and autoencoders [Wu et al., 2021a, Shimobaba et al., 2017] are some of the designs for data-driven neural CGH that have been used in inferring multiple fringe pattern [Kang et al., 2021] or for stereoscopic images [Chang et al., 2022]. These designs have been discovered to produce high-quality holograms with fewer artefacts and noise compared to conventional CGH methods.

Model-driven Neural CGH Model-driven neural CGH approaches incorporate holographic concepts directly into network design by including layers that perform Fresnel propagation, Fourier transforms, and phase retrieval [Choi et al., 2021, Rivenson et al., 2018]. This method eliminates the dependency on massive training datasets and can potentially increase the network's generalisation and interpretability. The DeepCGH architecture, for example, adds a Fourier transform layer into a CNN and learns the connection between input objects and their associated phase-only holograms [Eybposh et al., 2020].

Computational Complexity and Processing Time The computational cost of neural CGH approaches is determined by the neural network's architecture and size and the optimisation algorithm utilised during the training phase. Training neural networks may be computationally costly, necessitating powerful GPUs and vast quantities of RAM. However, once trained, a neural network's forward pass for hologram creation has a low computational cost, making it suited for real-time applications. The training phase, which can take hours or even days depending on the amount of the training dataset, the neural network topology, and the available computational resources, dominates the processing time of neural CGH approaches. However, due to the reduced complexity of the forward pass, the production of holograms using a neural network is often quicker than standard CGH techniques once trained.

Techniques to Improve Processing Time and Efficiency Several ways to improve the processing speed and efficiency of neural CGH systems have been proposed, including:

- Transfer learning may exploit pre-trained neural networks, lowering the quantity of training data and time necessary to attain acceptable performance [Huang et al., 2022a].
- Pruning and quantization of neural networks: These approaches may be used to minimise the size and complexity of neural networks, resulting in quicker processing times and lower memory needs [Shortt et al., 2006, Huang et al., 2022b].
- Parallel and GPU-based computing: Neural CGH, like classic CGH approaches, may benefit from parallel and GPU-based computation to speed up the training and inference procedures [Chen et al., 2022, Rivenson et al., 2018, Liu et al., 2023b].

2.3 Quantum Computing

Quantum computing is an interdisciplinary area that merge quantum physics, computer science, and information theory and provides major advances in information processing by utilising the unique characteristics of quantum physics. This computing discipline extends beyond the bounds of classical computing and promises to address issues that classical computers cannot answer [Shor, 1999, Grover, 1996]. Quantum computing, a branch of physics and computer science, uses the principles of quantum mechanics. The concept of quantum state is as a central concept encapsulating information about a quantum system, unlike classical physics. Vectors represent a quantum system in a complex Hilbert space, and Schrödinger's equation governs its evolution [Schrödinger, 1926]. The quantum bit or qubit, is the quantum counterpart of a conventional bit. Quantum algorithms use fundamental quantum mechanics principles to tackle specific problems more efficiently than conventional algorithms. Shor's method for integer factorization [Shor, 1999] and Grover's algorithm for unstructured database search [Grover, 1996] are two of the most well-known quantum algorithms. Shor's technique is especially significant because it threatens commonly used cryptographic systems like RSA, which rely on the difficulty of factoring big numbers [Rivest et al., 1978]. Grover's technique outperforms traditional search algorithms by a factor of four, providing considerable benefits for various optimisation issues.

2.3.1 Basics of Quantum Mechanics for Quantum Computing

This section provides a high-level review of quantum mechanics, emphasising its applications in quantum computing and possible applications in computer-generated holography.

2.3.1.1 Superposition and Entanglement in Qubits

Superposition and entanglement are two important concepts in quantum physics. A quantum system's ability to hold many states concurrently is known as superposition. This might be represented by complex-valued wave functions that outline the probability amplitudes of distinct states [Dirac, 1958]. Superposition applies to qubits and allows them to exist in a linear combination of the base states $|0>$ and $|1>$. This characteristic allows quantum computers to do numerous computations simultaneously, laying the groundwork for their remarkable processing capability [Deutsch, 1985]. Entanglement, a unique quantum correlation [Einstein et al., 1935, Bell, 1964] binds the states of two or more qubits even when separated by huge distances and may also be observed in qubits. Entanglement is a critical resource for quantum algorithms and protocols [Jozsa and Linden, 2003, Steane, 1998, Ekert and Jozsa, 1998].

2.3.2 Quantum Gates and Circuits

Because it has the potential to answer difficulties that traditional computers find intractable [Steane, 1998, Williams et al., 1998], quantum computing has expanded fast in recent years. The theoretical underpinnings of quantum computing are utilised to design and test quantum computer hardware and algorithms. These models can significantly be used by computer-generated holography science. This section will go through one significant quantum computing model that we used as a basic approach as the largely diffuse method to describe algorithms [Montanaro, 2016]: the quantum circuit model [Yao, 1993]. We use quantum circuits to describe CGH algorithm, and we do not treat alternatives such

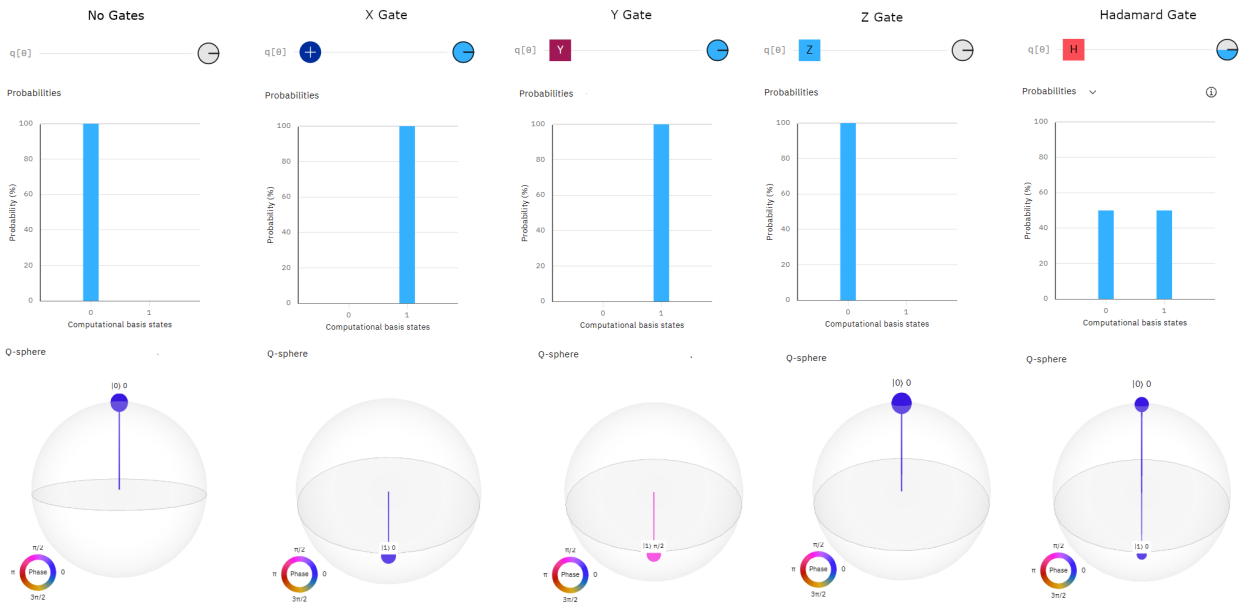

Figure 2.6: Pauli Gates (X, Y, Z) and their effects on a single qubit.

as adiabatic quantum computing [Albash and Lidar, 2018], topological quantum computing [Freedman et al., 2003], and measurement-based quantum computing [Raussendorf et al., 2003, Briegel et al., 2009]. The most widely studied quantum computing model is the quantum circuit model [Nielsen and Chuang, 2010]. In a quantum circuit analogous to a classical circuit, a computation is represented by a series of quantum gates, measurements, initializations of qubits to predetermined values, and maybe additional operations. DiVincenzo's criteria [DiVincenzo, 2000] are the minimal set of operations that a circuit must be capable of carrying out on the qubits to allow quantum computing.

Quantum gates execute unitary transformations on qubit states. Unitary matrices represent these gates and operate on one or more qubits. Quantum gates, comparable to conventional logic gates, are the building blocks of quantum computing. They manipulate qubit states via unitary transformations and execute reversible operations on them [Brylinski and Brylinski, 2002]. The Pauli-X, Pauli-Y, Pauli-Z, Hadamard, and Controlled-NOT (CNOT) gates are common quantum gates, as shown in Figure 2.6 and 2.7. Quantum circuits are a succession of time-ordered operations to a qubit's starting state. The output of a quantum circuit is produced by measuring the ultimate state of the qubits, which collapses their superposition into a distinct classical value. The quantum circuit model is a valuable tool for creating quantum algorithms.

Typically, a quantum circuit is shown as a sequence of horizontal lines, with each line representing a qubit. In between the lines are symbols that symbolise quantum gates. At the ends of the lines, certain symbols are used to signify measurements. Quantum gates are depicted as symbols on the qubit line and they can include vertical lines to connect to other qubit lines. Superposition and entanglement between qubits can be achieved and visualised with a proper combination of gates as shown in Figure 2.8.

2.3.3 Quantum Algorithms

Quantum computing has the potential to alter several facets of computer research fundamentally. CGH is an interesting and promising area among the many applications that quantum algorithms promise to improve. In industries like photography, 3D display technologies, and optical information processing, CGH is crucial. Quantum algorithms provide a unique chance to overcome the computational issues that have long prevented the mainstream application of CGH techniques. The capacity to rapidly calculate complex holograms has major implications for the progress of these fields. The junction of quantum computing and CGH is examined in this work, along with the theoretical underpinnings, prospective benefits, and practical consequences of using quantum algorithms in this area. Creating quantum algorithms for CGH necessitates a thorough grasp of quantum computing and CGH fundamentals. Many methodologies have been offered in the literature to describe and analyse light propagation

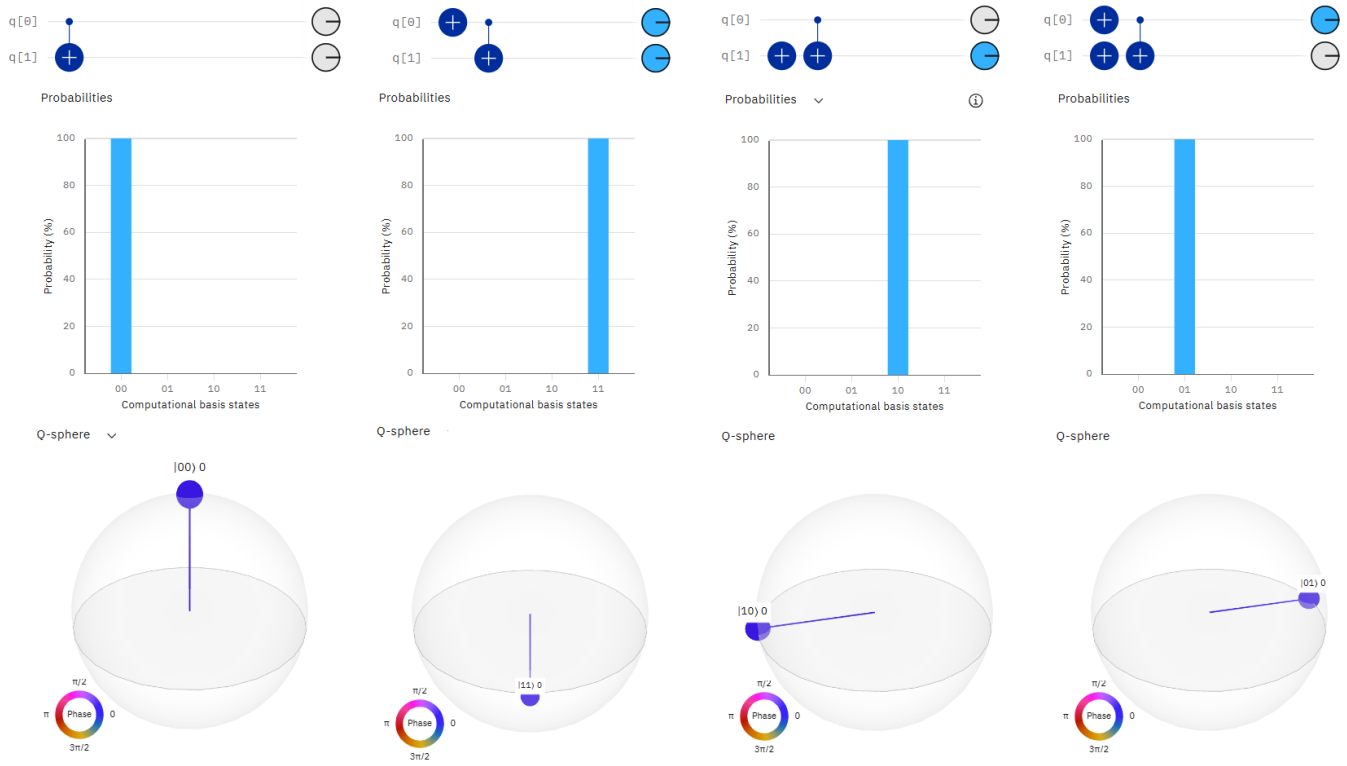

Figure 2.7: Control-NOT works on two qubits, control and trigger. In this figure, we show its effects on the four initial states $|00\rangle$, $|10\rangle$, $|01\rangle$, $|11\rangle$, achieved in this case with states $|00\rangle$ and X gates.

in holography, including scalar diffraction theory [Porter, 1970], Fourier optics [Goodman, 2005], and numerical simulations [Matsushima and Shimobaba, 2009]. These approaches serve as the foundation for constructing and analysing holographic algorithms and may be customised to use quantum computing's unique characteristics. Several quantum algorithms can be used to CGH-related difficulties. The quantum Fourier transform (QFT) and quantum discrete Fourier transform (QDFT), for example, are components that may potentially benefit CGH applications that rely on Fourier optics and related methods. Another interesting area of research in quantum algorithms for CGH is the development of quantum search and optimisation techniques like Grover's algorithm and its variations. These algorithms have shown the potential to significantly accelerate search and optimisation tasks when compared to classical algorithms, and they could be applied to various aspects of CGH, such as phase retrieval, aberration correction, and the design of optimal encoding schemes [Fienup, 1982, Gerchberg and Saxton, 1973, Bryngdahl and Wyrowski, 1990]

2.3.3.1 Shor's Algorithm

Shor's algorithm developed in 1994, is one of the most significant achievements in quantum computing [Shor, 1999]. This revolutionary algorithm demonstrated that a quantum computer could factor large integers exponentially faster than the most well-known classical algorithms, with important implications for the security of cryptographic systems based on integer factorization complexity, such as the widely used RSA cryptosystem [Rivest et al., 1978]. Shor's algorithm gave crucial insights into the design and analysis of quantum algorithms, which may be applied to other fields, such as computer-generated holography (CGH). The quantum Fourier transform (QFT) [Coppersmith, 2002], a quantum equivalent of the conventional discrete Fourier transform (DFT), lies at the heart of Shor's method. The QFT enables effective manipulation of phase information in quantum states, which is critical for Shor's algorithm's success [Nielsen and Chuang, 2010]. On a quantum computer, the QFT has been demonstrated to be more efficient than its classical equivalent, since it may be completed in n^2 steps, as opposed to $O(n \log n)$ steps for the fast Fourier transform (FFT), a frequently used implementation of the DFT. This efficiency boost might be used in CGH applications that rely on Fourier-based approaches, such as Fourier holography and related numerical methods. Shor's technique also emphasises the significance of utilising quantum computing's inherent parallelism and coherence to solve problems more efficiently than traditional approaches. These qualities might possibly be used in the context of CGH to perform

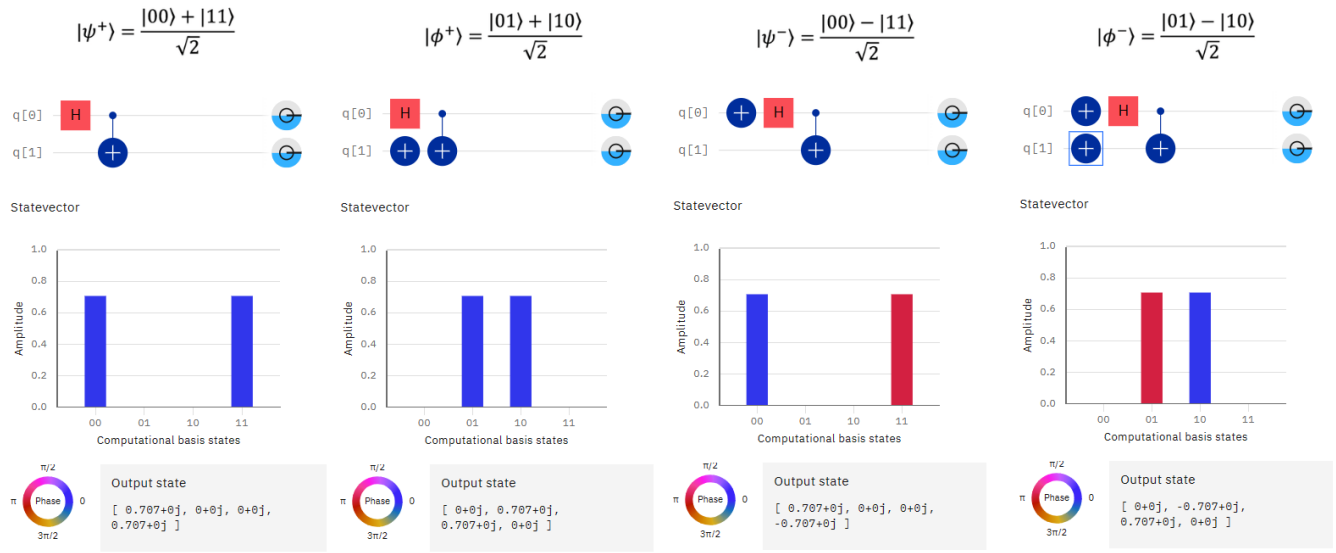

Figure 2.8: Bells state written with quantum gates.

parallel calculations of holographic interference patterns, resulting in quicker and more efficient hologram creation .Furthermore, quantum state coherence may aid in developing innovative holographic techniques that exploit quantum entanglement and other quantum phenomena inaccessible in classical computing. While Shor's method has not been directly applied to CGH, various academics have developed quantum algorithms that may be beneficial for CGH applications: quantum phase estimation technique, which is a major component of Shor's algorithm, as a potential tool for phase retrieval in holography. Another area of interest is the creation of quantum algorithms for solving linear systems of equations, such as the Harrow, Hassidim, and Lloyd (HHL) quantum algorithm [Harrow et al., 2009]. Under certain situations, this algorithm, which uses QFT and other quantum approaches, has shown the ability to solve linear problems tenfold quicker than conventional methods. This might allow for the efficient solution of linear equations originating from the discretization of the diffraction integral or other numerical methods employed in holography in the setting of CGH.Finally, Shor's algorithm was critical in proving the potential capabilities of quantum computing and gave vital insights into the design of quantum algorithms. While Shor's algorithm has not been directly used to CGH, its underlying ideas, such as QFT and quantum parallelism, might be used to produce more efficient holographic algorithms and procedures.

2.3.3.2 Grover's Algorithm

Grover's method, developed in 1996 by Lov Grover, is another significant achievement in quantum computing with possible implications for CGH [Grover, 1996]. Grover's quantum search technique can search an unsorted database or solve combinatorial optimisation problems rapidly. It outperforms conventional search algorithms by needing just $O(\sqrt{n})$ queries to identify a target item in a database of n elements instead of $O(n)$ queries for classical search algorithms. This acceleration has far-reaching consequences for various computing issues, including parts of CGH such as phase retrieval, aberration correction, and the construction of optimum encoding schemes. Many CGH applications rely on phase retrieval, which entails retrieving the phase information of a complex-valued hologram from intensity measurements [Fienup, 1982]. This topic is frequently framed as a search or optimisation problem, to find the phase distribution that minimises the error between the measured and calculated hologram intensities [Wyrowski and Bryngdahl, 1988]. With its quadratic speedup, Grover's technique might theoretically be used to rapidly explore the space of possible phase distributions, resulting in a quicker and more accurate phase retrieval procedure in CGH applications. Another key feature of CGH that might improve from Grover's approach is aberration correction. Optical aberrations, or imperfections in the light wavefront, can decrease the clarity of the reconstructed holographic picture [Kreis, 2006]. Correction of these aberrations frequently entails solving a difficult optimisation problem to determine the ideal combination of parameters that minimises the distortions [Goodman, 2005].Grover's technique, with its inherent speedup for search and optimisation issues, might possibly be used to quickly determine the ideal parameters for aberration correction, hence improving CGH system performance. Another area where Grover's

approach might potentially be used is in constructing optimum encoding schemes for CGH. Encoding strategies are used to encode complex-valued holograms with a real-valued or binary-valued function that may be projected on a spatial light modulator (SLM) or other holographic display device. The encoding strategy can significantly influence the quality and efficiency of the holographic reconstruction, and identifying the best scheme frequently requires searching or optimising across a vast set of options. Grover's approach might be used to rapidly search the space of encoding schemes, perhaps resulting to better holographic reconstructions and lower computing costs in CGH applications.

2.3.3.3 Quantum Fourier Transform

Quantum Fourier transform (QFT)-based methods is crucial in developing QCGH algorithms. One of the most important uses of QFT in QCGH could be the calculation of wavefront modifications, providing entanglement between qubits, which reflects the need for an interference pattern between qubits in generating holograms. In conventional methods, wavefront modifications require the assessment of Fresnel and Fraunhofer diffraction integrals, which are computationally costly procedures [Oikawa et al., 2011, Umul, 2005]. Quantum methods for fast Fourier transform (FFT) have also been developed and might be used to speed up the computation of diffraction integrals in QCGH. These methods provide a quantum equivalent of the conventional FFT, allowing the Fourier transform to be efficiently computed on a quantum computer. However, it is crucial to note that implementing QFT-based approaches in QCGH algorithms offers several obstacles. Implementing QFT may induce defects in the produced holograms due to the intrinsic vulnerability of quantum systems to noise and decoherence [Nielsen and Chuang, 2010]. To limit the impacts of noise and increase the quality of the reconstructed holograms, the use of quantum error correction techniques [Lidar and Brun, 2013] with surface codes [goo, 2023] or toric codes [Andreasson et al., 2019] may be required.

2.3.3.4 Iterative Quantum Phase Estimation

A fundamental quantum procedure called Iterative Quantum Phase Estimation (IQPE) is used in quantum computer-generated holography (QCGH) to reconstruct complicated wavefronts. While many quantum algorithms are designed to run faster than conventional equivalents, IQPE is primarily concerned with improving accuracy. As holography depends on recording detailed phase information by nature, it is an essential tool for producing high-quality holographic pictures. The IQPE algorithm, a quantum version of the classical phase estimation algorithm, is designed to estimate the phase of a quantum state with high accuracy. It is a sequence of the following steps:

1. Initialization: An initial phase estimate is necessary for the IQPE method. This approximation is fed into the supplementary register during the preparation of a superposition of states in the work register.

2. The IQPE method iteratively improves the phase estimate during iteration. Each cycle includes applying a controlled unitary operation. Often, the quantum gate in QCGH represents the unitary transformation of interest. Depending on the desired accuracy, different iterations may be necessary.

3. Following the iterations, the work register is subjected to a quantum Fourier transform (QFT). The output of this step gives the work register's most precise estimate of the phase.

4. Measurement: The work register is measured in a computational basis. The result produces an approximation of the quantum state's phase.

Since holography necessitates precise phase retrieval, the IQPE algorithm's ability to provide accurate phase estimates could be a valid tool or a basis for such a task. The iterative nature of IQPE allows for progressively finer phase approximations, ensuring that the final holographic image is detailed and accurate.

2.3.3.5 Hybrid Quantum-Classical Approaches

Quantum computers can do complicated computations tenfold quicker than classical computers, making them suited for computationally heavy jobs like CGH. However, both quantum and conventional resources must be used to fully realise the promise of quantum computers. The concept of quantum-classical variational algorithms (Figure 3.1), which combine quantum resources for addressing particular subproblems with conventional optimisation loops, is a

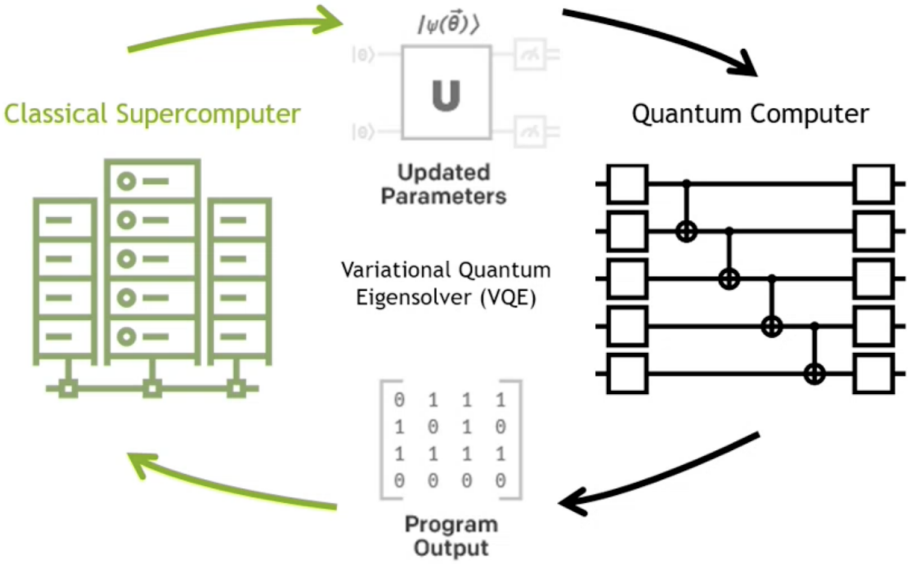

Figure 2.9: Variational quantum eigensolver architecture of a hybrid classical-quantum computer(Courtesy from NVidia).

major component of hybrid quantum-classical techniques [McClean et al., 2016, Endo et al., 2021]. QAOA and VQE are two examples. These algorithms have been applied in various disciplines, including combinatorial optimisation, quantum chemistry, and machine learning, highlighting the adaptability and promise of hybrid quantum-classical techniques. Hybrid quantum-classical algorithms may solve optimisation issues faster than conventional algorithms alone related to CGH, as can happen in stochastic gradient descent approach [Chen et al., 2019].

2.3.3.6 Quantum Machine Learning Algorithms

To create unique algorithms that can outperform conventional approaches in a variety of applications, the discipline of quantum machine learning (QML) integrates the concepts of quantum computing with machine learning techniques [Biamonte et al., 2017, Schuld and Petruccione, 2018, Schuld et al., 2015]. In the context of CGH, QML presents a viable approach for enhancing the effectiveness and performance of holographic algorithms and methods and investigating new uses and capabilities made possible by the quantum computing paradigm. The development of quantum algorithms for image reconstruction and processing is one possible use of QML in CGH. Deep learning and other classical machine learning approaches have previously shown the ability to enhance holographic picture reconstruction by learning to correct for numerous flaws and artefacts in the holographic process [Rivenson et al., 2018]. Quantum algorithms based on the Harrow-Hassidim-Lloyd (HHL) could potentially be used to solve the linear inverse problems that frequently arise in holographic image reconstruction, potentially providing significant speedup over classical methods under certain conditions. Moreover, to create more accurate and cheap algorithms for phase retrieval and aberration correction in CGH applications, QML might possibly integrate quantum search and optimisation approaches with machine learning models. To learn the complicated links between holographic patterns and the underlying phase distributions or aberrations, quantum neural networks, which are quantum equivalents of conventional neural networks, might be used. QML can potentially aid in developing novel holographic methods and applications that benefit from the special properties of quantum computing. It is possible to utilize quantum machine learning for producing and reconstructing holograms as some classical neural network approach with previously unheard-of resolution, sensitivity, and information capacity by combining CGH using quantum entanglement and other quantum phenomena to improve imaging performance. More studies should be directed towards building QML algorithms designed especially for CGH applications and investigating the more significant implications of quantum computing and machine learning for holography and related domains. We list and briefly discuss some of the most important QML algorithms here:

1. QSVM (Quantum Support Vector Machine): QSVM [Rebentrost et al., 2014] is a quantum variant of the

traditional Support Vector Machine (SVM) technique used for classification and regression. QSVM transforms input data into a high-dimensional feature space and solves an optimisation problem to determine the optimum hyperplane that divides the classes. The quantum version uses quantum parallelism to speed the computation of the kernel matrix, a fundamental component of SVM.

2. Quantum Principal Component Analysis (QPCA): QPCA [Lloyd et al., 2014] is a quantum approach for dimensionality reduction and feature extraction, similar to classical PCA. QPCA uses quantum phase estimation to compute the eigenvectors and eigenvalues of the covariance matrix more efficiently than classical approaches, allowing for quicker extraction of main components.

3. Quantum Boltzmann Machines (QBMs) [Amin et al., 2018] are a quantum counterpart of conventional Boltzmann Machines, which are generative, probabilistic models used for unsupervised learning. QBMs employ quantum annealing or other quantum optimisation approaches to sample from the system's energy distribution more effectively, possibly resulting in faster convergence and more accurate models.

4. Quantum Approximate Optimisation Algorithm (QAOA) and Variational Quantum Eigensolver (VQE): VQE and QAOA are hybrid quantum-classical algorithms that are used to solve optimisation challenges, such as those encountered in machine learning5. These techniques employ parameterized quantum circuits to approximate the ground state of a cost function Hamiltonian, changing the parameters repeatedly via classical optimisation until convergence [Peruzzo et al., 2014, Zhou et al., 2020].

5. Quantum Neural Networks (QNNs): QNNs [Kak, 1995] are a type of quantum algorithm that is influenced by classical neural networks. To express the mappings between input and output data, QNNs employ parameterized quantum circuits. In a way similar to classical deep learning, the parameters are updated using a mix of quantum and classical optimisation algorithms.

Chapter 3

Methodology

3.1 From CGH to QCGH

CGH necessitates the resolution of complicated numerical issues, such as phase retrieval and diffraction pattern calculation, which can be computationally costly. By delivering more efficient methods for tackling these challenges, quantum computing can potentially treat CGH issues with some advantages. For example, the quantum phase estimation technique [Kitaev, 1995] might be modified to estimate better the optical wavefront phase in CGH than conventional approaches. Similarly, the quantum Fourier transform a crucial component of Shor's algorithm, might accelerate diffraction pattern calculation [Nielsen and Chuang, 2010]. While the use of quantum computing in CGH is not yet proposed, recent advances in quantum algorithms and hardware imply that it might become an additional tool for hologram creation in the near future.

A comprehensive approach that involves issue formulation, classical and quantum algorithm design, implementation and simulation environment and performance measurements is required to develop and test QCGH algorithms. This section summarises the techniques employed in the QCGH inquiry, focusing on pertinent research and innovative techniques. The QCGH problem formulation includes constructing a hologram using a quantum computer's computational resources, including designing and implementing quantum algorithms that can mimic classical algorithm counterparts or be novel. For example, this task may be seen as an optimisation problem, with the optimisation variables standing in for the hologram parameters and the objective function calculating the difference between the intended and recreated optical fields. How this problem is framed will determine how well conventional and quantum algorithms are developed. Iterative Fourier transform algorithms (IFTA) and Gerchberg-Saxton algorithms (GS) are two design methods for traditional CGH algorithms that have been discussed in the literature. These algorithms may inspire the design of QCGH algorithms function and offer perceptions into the characteristics and issues of the holography problem.

To tackle the holography problem more efficiently than conventional methods, quantum CGH algorithm design necessitates the development of quantum algorithms that use the special properties of quantum computers. The implementation and simulation environment is essential to assess and contrast QCGH algorithms. For the creation and testing of QCGH algorithms, current quantum computing platforms like IBM Q [1], Google Quantum AI [2], and Rigetti Computing [3] offer cloud-based access to quantum processors and simulators [Castelvecchi, 2017]. Additionally, resources and tools for building and simulating quantum circuits are available through open-source software libraries like Qiskit [4] and Cirq [5]. In the following sections, we depict problem formulation, conventional and quantum algorithm design, implementation and simulation environment, which are all part of the process for investigating quantum computer-generated holography.

[1] https://research.ibm.com/interactive/system-one/
[2] https://quantumai.google/
[3] https://www.rigetti.com/
[4] https://qiskit.org/
[5] https://quantumai.google/cirq

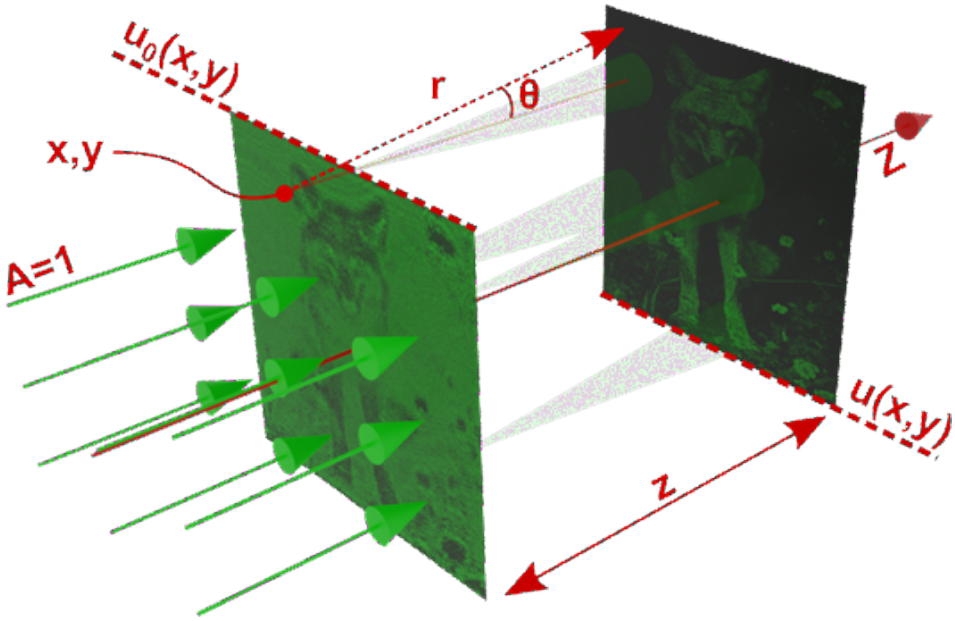

Figure 3.1: Complex field from holographic plane to reconstruction plane (Image courtesy from Kaan Aksit Computer-Generated Holography site[7]).

3.2 Problem Formulation

The primary goal of QCGH is to reconstruct the phase and amplitude information of a wavefront using quantum algorithms, as classic CGH does. Reconstructing the complex-valued wavefront, given a series of intensity measurements, is the goal of the phase retrieval problem in holography. The wavefront can be written as a complex-valued function, f(x), where x stands for the wavefront's location. The intensity readings are correlated with the wavefront's Fourier transform's squared magnitude, or |F[f(x)]|2, where F stands for the Fourier transform operator. Such a complex-valued function is also called "complex field" - and we refer to this term in this study It can be discretely represented as a grid of points with a position in the 3D space where we associate a complex number (or two real numbers). In classic CGH, the complex field is achieved as a set of sorted couples of floating point numbers; in the context of QCGH, we must find a suitable representation that also considers some properties of holography. One method could be using quantum states for each wavefront point (upon an analogous discretization of the complex field) to represent the wavefront and store the amplitude and phase information in the probability amplitudes of the quantum state. Along with the representation of the wavefront, other aspects need to be treated to generate the hologram with a quantum algorithm; we can take inspiration from the classic IPRA algorithms (as described in 2.2.2.1) to determine the different stages that our quantum algorithm need to contain. As the first steps into this new approach, we consider well-established studies with deep historical roots and the opportunity to evolve into various variants over time, enhancing accuracy while reducing spatial and temporal complexity. We accurately depict the design and implementation of the GS algorithm, which is a good representative of the iterative phase retrieval methods.

3.2.1 Classical CGH Algorithm Design

In-depth descriptions of the standard CGH algorithm design are provided in this part, emphasising key elements, including object representation, holographic computing, and optimisation. First and foremost, the object is digitally represented while creating a standard CGH algorithm. The wavefront with the object's visual information is often represented as a complex field in some spatial coordinates, and the complex values denote the amplitude and phase of the object wavefront. The wavefront is generally considered a plane, following Fraunhofer diffraction representation. Following that, the hologram computation is performed, which comprises computing the complex-valued hologram, $H(u)$, given the object representation, $O(x)$. The formula is $H(u) = F[O(x) * h(x)]$, where F is the Fourier transform, the operation is the convolution operation, and h(x) is the impulse response of the diffraction integral. The diffraction

[7]https://kaanaksit.com/odak/course/computer_generated_holography/

integral, which relates the object wavefront to the hologram via a linear transformation, is extensively employed in holographic computation. So, in general, we refer to the hologram, diffraction pattern or complex field in hologram space as the record of the interference pattern that captures information about the amplitude and phase of the light field after it interacts with an object. Such terms represent the same entity. Amongst the CGH techniques based on iterative optimisation approaches, we selected the Gerchberg-Saxton algorithm. Such an approach starts with a guess for the holographic phase and updates iteratively. The amplitude information of the hologram is normally set to satisfy the restrictions imposed by some conditions, such as the display devices or manufacturing procedures. On the other hand, non-iterative CGH techniques, such as the direct binary search (DBS) algorithm and the simulated annealing algorithm, generate the hologram directly without relying on an initial guess, in addition to iterative optimisation approaches. Thus, the design of typical CGH algorithms includes three key components: the digital representation of the object, the holographic computation based on the diffraction integral, and the optimisation process to improve reconstruction quality and deal with practical constraints.

3.2.2 Gerchberg-Saxton Algorithm

The Gerchberg-Saxton (GS) technique is an iterative phase retrieval method that calculates the phase of an unknown complex-valued field based on two intensity measurements taken in two separate planes: the hologram plane, and the image reconstruction plane. Both planes contain the evolving wavefront and are represented mathematically by a complex field with amplitude and phase that vary with some wave propagation methods. The algorithm begins with a complex field with amplitude values equal to the calculated wavefront pattern in the hologram plane or an initial constant amplitude value and random beginning phase values. We replace the complex field's amplitude with the incoming light source's amplitude, which is one if a collimated laser source is utilised. Propagation transmits the complex field formed with these values to the reconstruction plane. Later, the amplitude of the complex field in the reconstruction plane is substituted with the amplitude values of the intended target pictures. This new complex field is returned to the hologram plane using the same propagation equation (but with the opposite direction), where the amplitude and phase values are modified to the wave propagation mechanism used. The amplitude is again replaced with the initial value in the hologram plane. This forward and backward propagation process is repeated until the reconstruction produced by the phase-only hologram is satisfactory. The picture's complexity that has to be rebuilt, the desired degree of precision, and the measured data's signal-to-noise ratio (SNR) are some of the variables that determine how many iterations the Gerchberg-Saxton (GS) method needs to run. Generally speaking, more iterations are needed to reconstruct pictures with more complexity or at higher accuracy levels. Nevertheless, overfitting—a situation in which the algorithm begins to simulate the noise in the measured data rather than the signal—can also result from increasing the number of repetitions. The GS algorithm often iterates between 10 and 100 times. Some applications may call for more or fewer iterations. Such a code is in Appendix A.1.

Algorithm 1 Gerchberg-Saxton Phase Retrieval

```
1:  procedure GerchbergSaxton(image_magnitude, num_iterations)
2:      Initialize random phase for object object_phase
3:      for iteration ← 1 to num_iterations do
4:          Calculate Fourier transform: object_fourier
5:          Inverse Fourier transform: object_estimate
6:          Update object phase: object_phase
7:          Calculate object magnitude: object_magnitude
8:          Replace magnitude: object_fourier
9:          Inverse Fourier transform: object_estimate
10:         Update object phase: object_phase
11:     end for
12:     return object_estimate
13: end procedure
```

GS function takes an input image magnitude (the known magnitude information), and the number of iterations for the algorithm. We start by initializing a random phase for the object, our initial guess. The magnitude of the initial guess is the known image magnitude, and then the loop starts:

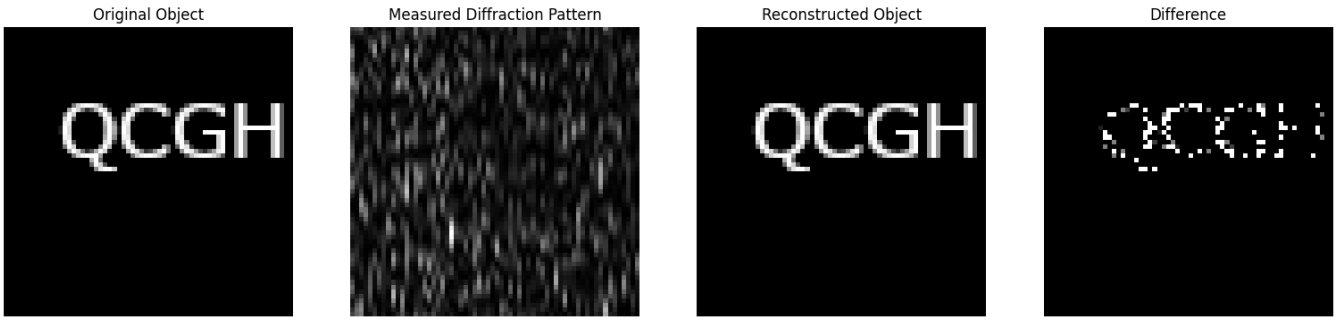

Figure 3.2: Comparison of the reconstructed and target values after 50 iterations, with a Qiskit version.

1. We calculate the FT of the object by combining the magnitude with the current phase.
2. We take the IFT to get a new object estimate.
3. the phase is updated using the angle of the complex-valued object estimate.
4. We replace the calculated magnitude with the known image magnitude (that is known), retaining the updated phase.
5. We repeat the IFT to get a new object estimate and update the phase.

After the specified number of iterations, the function returns the final estimate of the object, which should now have a phase that better matches the original phase. The example usage at the end demonstrates how to generate a sample magnitude image, run the Gerchberg-Saxton algorithm, and visualize or save the retrieved image. Figure 3.2 shows the original image, the diffraction pattern generated and the reconstructed image.

3.3 Gerchberg-Saxton and complex Transport Function

At its core, the Gerchberg-Saxton algorithm relies on several fundamental parameters to perform its reconstruction, following a more realistic model. In the next implementation (in which we use Odak [8] [Akşit and Kavaklı, 2023, Kavaklı and Akşit, 2022, Kavakli et al., 2022] toolkit for Python), we start by defining the wavelength of the incident light, a critical component in optical systems. The algorithm leverages this information to calculate the wavenumber, aiding in the subsequent computations. Other vital parameters encompass spatial resolution, represented as pixel dimensions, and the propagation distance within the optical system. These parameters, fine-tuned to the specifics of the problem, are instrumental in the image reconstruction and holography processes. The code for this section is in Appendix A.2. We start with preparing the target image, the focal point of the reconstruction process. Employing PyTorch, an open-source machine learning framework, we load the image and seamlessly transform it to meet the algorithm's prerequisites. Initially, the image is resized to match the desired resolution, ensuring the reconstruction process adheres to the intended dimensions. Furthermore, grayscale conversion is performed, yielding a monochromatic image representation. This transformed image is converted into a PyTorch tensor, ready for the forthcoming computational steps. The core of the Gerchberg-Saxton algorithm unfolds iteratively. Holograms are generated over a predetermined number of iterations, and the reconstruction process is simulated with mathematical precision. The algorithm relentlessly updates the amplitude and phase of the holograms in each iteration. Finally, the implementation offers a window into the algorithm's capabilities by rendering the reconstructed image and holograms. This visual representation aids in comprehending the results and assessing the effectiveness of the Gerchberg-Saxton algorithm. The final image, in this case with a more detailed model, is depicted in Figure 3.3.

3.4 Quantum CGH Algorithm

The quantum version of the iterative Fourier transform algorithm for CGH can be split into the same stages as the classic version. In addition, it is necessary to understand the representation of some entities essential for diffraction

[8] https://github.com/kaanaksit/odak

pattern creation and image reconstruction. Firstly, we remark on the difference between the classic version of GS and a possible QGS:

Complex Field Representation Classical Gerchberg-Saxton (CGS): In the classical version, the complex field is represented as a 2D array of real numbers (as complex numbers). Each array element corresponds to a point in the complex plane, capturing both amplitude and phase information.
Quantum Gerchberg-Saxton (QGS): In the quantum version, the complex field could be represented using quantum states. Quantum states, such as qubits or quantum registers, could store the amplitude and phase information in a quantum superposition, allowing for the quantum properties of superposition and entanglement to be utilized.

Interference Pattern CGS: The classical version of the algorithm calculates the interference pattern, which represents the intensity distribution resulting from the object's superposition and reference waves. The interference pattern is usually computed using a Fourier transform of the complex field.
QGS: In the quantum version, interference patterns could be calculated using quantum Fourier transform operations, or with operations that entangle qubits. Quantum Fourier transforms are designed to process quantum states and could simulate interference patterns by manipulating the quantum states corresponding to the object and reference waves.

Transport Equation CGS: The transport equation in the classical version describes the propagation of the complex field from the object plane to the hologram plane or to the hologram plane and the image reconstruction plane and back. It involves convolution operations and phase retrieval steps to update the complex field iteratively.
QGS: In the quantum version, simulating the transport equation could involve quantum gates for phase shifting and controlled operations for interference calculations. Quantum gates and entanglement can be leveraged to perform the transport equation efficiently.

Iteration Process CGS: Classic Gerchberg-Saxton iterates between the hologram and reconstruction planes, updating the complex field in each iteration. The algorithm repeats this process until a convergence criterion is met.
QGS: The quantum version of Gerchberg-Saxton would also involve an iterative process where quantum operations update the quantum states representing the complex field.

3.4.1 Quantum Gates and Techniques for Quantum Gerchberg-Saxton Algorithm

In this section, we try to define some quantum information tools when dealing with quantum circuit environments necessary to implement specific processes in QGS.

Superposition Gate (Hadamard - H): Purpose: The Hadamard gate is used to create a superposition state. For a single qubit, applying H transforms the qubit from the $|0\rangle$ state to an equal superposition of $|0\rangle$ and $|1\rangle$ states.
Application: In the Quantum Gerchberg-Saxton Algorithm, superposition can be useful when initializing the quantum states representing the complex field, allowing for quantum parallelism during calculations.

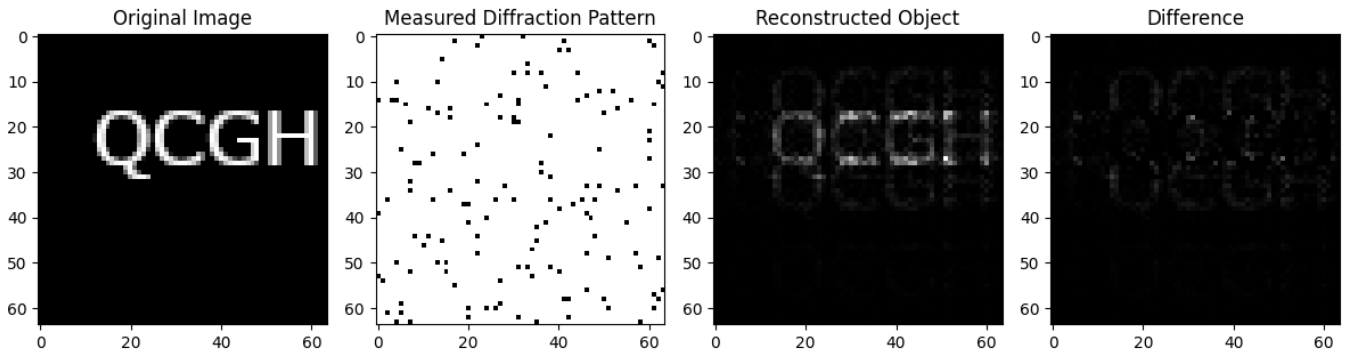

Figure 3.3: Comparison of the reconstructed and target values after 50 iterations, with a Odak version.

Entanglement Gate (CNOT, or other controlled gates): Purpose: The CNOT (Controlled-NOT) gate creates entanglement between two qubits. When the control qubit is in a superposition, it entangles the target qubit's state based on the control qubit's state.
Application: In the algorithm, we can entangle qubits to simulate the interference between the object and reference waves, creating interference patterns.

Phase Kickback (CZ and H): Purpose: The CZ (Controlled-Z) gate creates phase kickback by transferring the phase information from one qubit to another. We can manipulate the phase information by applying Hadamard gates before and after.
Application: Phase kickback can be used to simulate the propagation of waves and reconstruct the complex field in quantum holography.

Quantum Phase Estimation (H, U3, and CX): Purpose: Quantum Phase Estimation is a crucial technique for estimating phase information. H prepares qubits in superposition, U3 gates perform unitary transformations representing the hologram propagation, and CX gates apply controlled operations.
Application: these gates and techniques are employed to estimate the phase of the complex field iteratively.

Quantum Fourier Transform (H, Controlled-Phase, and Measurement): Purpose: The QFT is essential for frequency domain transformations and pattern recovery. H gates prepare qubits, and controlled-phase gates apply phase shifts.
Application: In quantum holography, QFT is useful for simulating interference patterns and transforming between spatial and frequency domains. Measuring the result yields the recovered holographic data.

3.4.2 Designing the Quantum Gerchberg-Saxton Algorithm

The Gerchberg-Saxton algorithm could be designed with the following challenges and implemented using qubits and Qiskit. The first step is representing the complex field in a quantum algorithm structure. The complex field could be represented by a set of qubits that can be grouped in a quantum register, with the qubit states encoding the amplitude and phase values. For example, θ and ϕ rotation gates could encode amplitude and phase. The θ rotation gate ($R\theta$), is a quantum gate that rotates the state of a qubit around the Bloch sphere, acting on the amplitude. The ϕ rotation gate ($R\phi$), is another quantum gate used for single-qubit rotations, introducing a phase shift to the quantum state of the qubit without changing the probability amplitudes of the states. In this way, when using more qubits, the real and imaginary components could be directly encoded in qubit basis states. Additional ancilla qubits may be needed for intermediate values. Secondly, tackling the propagation mechanism: such a step essentially performs a Fourier transform on the complex field. Quantum Fourier transforms could achieve this propagation between wavefronts represented by qubits registers. Thirdly, we need to translate propagation equations into unitary transformations, such an approach will simulate the wavefront propagation (and backpropagation) from hologram plane and reconstruction plane. Conditional operations like CNOTs and Toffolis could implement setting amplitude or phase constraints. Lastly, all the processes must be wrapped with a cycle acting as an iterative approach. The iterative optimization process aims to minimize the error between the target and the reconstructed state. It could formulate a variational quantum algorithm, updating parameters to minimize loss: loss calculation and classical optimization loop around quantum propagation steps. To detail how to pursue our goal, the following fundamental actions need to be taken:

- Mapping continuous math onto discrete qubit operations.
- Achieving high fidelity coherent transformations.
- Information exchange between classical and quantum steps.
- Quantifying if/where quantum speedups can be gained.

In the initial approach of the QGS algorithm, we can create two complex planes, one for the hologram and one for the reconstruction, and suppose that the wavefront moves iteratively from the hologram plane to the reconstruction plane. We allocate two sets of qubits to represent the requisite complex fields, and we internally entangle the qubits

at each complex plane. We may map the field intensities by setting the amplitude of the qubits to be measured in the $|1\rangle$ state, and the phase of the qubit represents the field phases. We can achieve propagation for all qubits by phase shifting. Phase shifting, entanglement, and superposition can all be used to create interference patterns. Although a series of regulated and Hadamard gates can accomplish the same goal, QFT is a strong contender to provide such features. After that, the circuit is repeated for the required number of times with the updated amplitude and phase data saved in the quantum states and arranged in registers.

3.5 Quantum Fourier Transform for Gerchberg-Saxton

The QFT creates interference patterns and manipulates quantum states as part of the algorithm's iterative process. The QFT acts as a change of basis, transforming computational basis states into equally weighted superpositions of all basis states. When measured, this results in interference effects, as the amplitudes can constructively or destructively interfere. The specific interference pattern depends on the number of qubits and measurements performed. Additionally, the controlled phase rotations in the QFT entangle the target qubit with the control qubit. This gradually builds up entanglement across all the qubits in the register. The QFT can be implemented using a variety of quantum circuits. The Hadamard gate, a single-qubit gate, sets a qubit in a superposition of the $|0\rangle$ and $|1\rangle$ states. A Z gate is applied to the second qubit by the controlled-Z gate, a two-qubit gate when the first qubit is in the $|1\rangle$ state. In Appendix A.3 we provide the Qiskit example of the QFT circuit for a 4-qubit register. A quantum state will be in a superposition of all potential Fourier transform basis states after the QFT circuit has been applied to it.

3.6 Simulation Environment

In this section, we describe the software packages and tools utilized to study GS algorithm in CGH and the implementation of QGS circuit in QCGH.

3.6.1 Qiskit

Qiskit is an open-source Python library enabling researchers, educators, and developers to interface with quantum computers at the circuit level. Quantum circuits serve as executable programs defining qubit behaviour on quantum processors. Qiskit provides tools for constructing, manipulating, and executing quantum circuits to make practical quantum computing accessible. Fundamentally, Qiskit represents quantum computations as circuits comprising sequences of quantum logic gates applied to qubits. It offers classes for building and visualizing quantum circuits, leveraging Python's mature numerical and scientific computing stacks through integration with NumPy and SciPy. Qiskit's circuit model enables composing, transpiling, and optimizing circuits on quantum processing units. For circuit execution, Qiskit grants access to simulated backends for exact and noisy emulation and real quantum processors via the cloud. Its provider mechanism allows connecting with resources from partners across the quantum computing industry. Qiskit provides a unified interface to manage time allocation, job submission, and result retrieval across accounts and hardware providers. Building upon this low-level circuit foundation, Qiskit provides higher-level tools for constructing quantum programs and applications, including domains like chemistry, optimization, finance, and machine learning. Coupled high-level programming enables integrating quantum as co-processors for hybrid classical-quantum workflows: visualization, monitoring, and debugging tools aid in developing large quantum codebases. Qiskit's extensible open-source design fosters an ecosystem of tools, algorithms, and hardware support. It empowers users to engage directly with quantum circuits to characterize, understand, and design around noise, errors, and device imperfections critical to applied research.

3.7 Circuit Creation

The quantum Gerchberg-Saxton (QGS) algorithm represents the first attempt to leverage quantum computing principles to tackle a classic problem of computational holography. At its core, the Gerchberg-Saxton algorithm is used for phase retrieval, a process critical in reconstructing information.

We assemble the algorithm considering the different stages mapped on the quantum counterpart. The QGS algorithm utilizes quantum circuits to manipulate quantum states representing elements in the complex field, performing

Figure 3.4: Quantum circuit that aims to mimic Gerchberg-Saxton algorithm. The QFTs that entangle the qubits in quantum registers representing complex planes and phase change acting as transport equation are visible. Such a circuit visualises only three iterations.

operations parallelising certain aspects of the interference process. Unlike the classical Gerchberg-Saxton algorithm, the quantum version introduces quantum gates, and quantum Fourier transforms.

In a quantum Gerchberg-Saxton iteration, quantum registers initialise quantum states representing the complex field in the holographic plane with initial amplitudes and phases. These quantum states are then propagated between the hologram and reconstruction planes using quantum gates, creating interference patterns. In this version, we simplified different stages, such as the propagation, even if in a later section, we propose a FresnelTransportGate as a starting point for a more complex solution for the wavefront propagation. The target intensities in the reconstruction plane are set during each iteration. Finally, quantum measurements extract the amplitudes or the phases from the quantum state. We identified different challenges in generating the circuit: firstly, we must map intensities into angles for the qubits. Secondly, we need a way to set θ rotations, between every unidirectional propagation cycle. Moreover, we should define a propagation function, and finally, we need to provide a mechanism that excludes wrong results from the outcome. We provide the code in the Appendix A.4

In our circuit, as depicted in a four qubits version in Figure 3.4, we defined three sections that appear to be the stages in the Gerchberg Saxton algorithm: initialization phase, interference creation, and propagation. The last two phases are then repeated until the final generation of the diffraction pattern and reconstruction. Here, we describe the functionalities mapped into the functions defined in the code. We have developed the script by established software development principles and best practices, aiming to ensure modularity and facilitate the code's comprehensibility.

- gerchberg_saxton(iterations): This is the main function implementing the quantum Gerchberg-Saxton algorithm. It sets up quantum registers for hologram, reconstruction, and eventually classical registers for measurement results. It initializes target intensities, initial intensities, and initial phases. The function then initializes the

circuit by setting the initial intensities and phases and propagates information back and forth between the hologram and reconstruction planes for a specified number of iterations. It simulates the quantum circuit and extracts amplitudes or phases based on the phase flag. We swap the quantum register at each iteration as modifying the previous registers becomes the new complex field.

- initialize_amplitudes_and_phases(qc, qubits, intensities, phases): This function initializes the qubits to $|0\rangle$ and sets their amplitudes based on the provided intensities and phases.
- propagates (qc, from_qubits, to_qubits): This function creates entanglement with QFT and propagates information from one set of qubits to another by applying phase shifts and controlled-Z gates between the qubits. We select as a naive solution a constant phase, but a more careful modelling would accept parameters to calculate the right phase for each qubit.
- set_intensities(qc, qubits, target_intensities): This function sets the intensities of the qubits based on the target intensities.

The code then includes functions for estimating amplitudes and phases via Bloch sphere functions. To compare this quantum version of the code with the classic Gerchberg-Saxton algorithm: The code initializes quantum states with amplitudes and phases in the hologram plane, that correspond to the initialization in GS, where the hologram plane is set with constant amplitudes and phase is random. After applying QFT that makes a correlation between all the qubits in the complex plane, it iteratively propagates these states to the reconstruction plane and back, applying phase shifts and controlled gates to simulate interference alternating QFT and inverse QFT. That corresponds to GS algorithm where FFT is used for interference pattern, and phase modification is applied according to parameters such as wavenumber, distance, wavelength, etc... In our version, we simplify such a section, but we describe and realise a FresnelQuantumGate that paves the way for future, more complex quantum propagation models. At the end of each forward propagation (hologram to reconstruction plane), the target intensities in the reconstruction plane are set as the GS algorithm recommends. Finally, we extract the amplitudes or phases so that a diffraction pattern can be used for image reconstruction. Thus, we identify some key differences from the classic Gerchberg-Saxton algorithm:

The classic algorithm is entirely classical, while this code uses a quantum circuit for phase retrieval. The classic algorithm is well-established and widely used in optics, whereas this quantum version is experimental and exploratory. This quantum version involves quantum gates, QFT, and Bloch sphere representation, which is not part of the classical algorithm. The quantum version may take advantage of quantum parallelism in some way, potentially providing speedup compared to classical methods. Overall, this code represents an attempt to apply quantum principles to a classic problem in holography. However, further refinement and analysis are required to determine the practicality and advantages of such an approach.

3.7.1 Fresnel Transport Gate

A tailored quantum gate called the FresnelDiffractionGate is made to mimic the process of Fresnel diffraction, which is a key concept in wave optics. The two essential factors characterising this gate are the wave number (k) and the propagation distance (distance). These characteristics have a direct relationship to Fresnel diffraction as a physical process.
Wave Number (k): this parameter of Fresnel diffraction, serves during propagation applied to the phase variations, and the gate must be able to apply a quantum rotation (Ry) based on this wave number. Propagation Distance (distance): Fresnel diffraction occurs as a wavefront travels from one plane to another, and distance is important. This gate incorporates the distance parameter to account for the phase shifts. The quantum operations within the gate, specifically the Ry rotation and the Hadamard (H) gate, simulate the wavefront's transformation through this distance.
By incorporating these parameters, the FresnelDiffractionGate bridges the quantum circuit to the physical process of Fresnel diffraction, enabling the quantum simulation of this optical phenomenon. This gate is a testament to how quantum computing can be applied to model and understand complex physical phenomena more accurately.

3.7.2 Visualization

We visualise the states (with bar charts for the decider quantities) of such a quantum system using Qiskit's Bloch sphere data visualisation module. We compute and display the amplitudes, phases, and goal intensities for a set of

Algorithm 2 Quantum Gerchberg-Saxton Algorithm

Input:

Number of qubits
Iterations
Target intensities
Initial intensities
Initial phases

Output:

Extracted phases/amplitudes

function GERCHBERGSAXTON

Initialize quantum registers and classical registers

Create a quantum circuit

for each iteration **do**

Initialize amplitudes and phases for the hologram plane

Propagate from the hologram to the reconstruction plane

Set target intensities in the reconstruction plane

if last iteration **then**

Break

end if

Propagate from the reconstruction to the hologram plane

Set target intensities in the hologram plane

end for

end function

iterations. The Bloch sphere is an effective tool for comprehending the quantum states and how they change during the GS algorithm. We can see how the algorithm is doing as it refines the quantum state to meet the goal intensities by displaying these states visually. We provide the code in Appendix A.5.

Chapter 4

Discussion

4.1 Considerations on Gerchberg-Saxton algorithm

To summarize, the GS algorithm is an iterative approach for determining the phase of a complex-valued wavefront from two intensity measurements taken in different planes. The algorithm operates by alternating two steps: the first one is the propagation of the complex field from the hologram plane to the reconstruction plane, and the second is the rectify of the complex field to match the target intensity. The process starts with a random guess for the hologram phases. The propagation step involves taking the complex field's Fourier transform, propagating the Fourier amplitudes through a system defined by its transfer function, and then taking the inverse Fourier transform. The error correction step is carried out by comparing the propagated field intensity to the second intensity measurement. The iterative process that involves such steps minimises the error between the target image and the reconstruction image. The GS algorithm is an effective phase retrieval tool. It is relatively easy to apply and is frequently very effective. However, the technique might be sluggish to converge and susceptible to measurement noise. The Gerchberg-Saxton algorithm has the following advantages: easy implementation, quite effective, phases determination of a complex-valued wavefront from two intensity measurements. Anyway, it suffers from the following drawbacks: slow convergence, susceptibility to noise, and initial guess dependency that might be problematic for convergence.

4.2 Complex Field in Holography

CGH represents the complex field using a series of pairs of real numbers, each representing the amplitude and phase of a single point in the complex field. In the quantum iteration of the process, a collection of qubits represents the complex field. The field's amplitude at each point represents the chance that a qubit is in the $|1\rangle$ state, and the qubit's phase is utilised to indicate the field's phase at each point.

The field's propagation is implemented via a series of quantum gates. Using X or Y gates for probabilities and Z gates for phases, the qubits are rotated into a superposition of the $|0\rangle$ and $|1\rangle$ states. To propagate the field, we employed QFT, and phase shift gates were applied to the qubits in a certain order.

To measure the qubits in the complex field, the conventional method is the projective measurement (tomography) using the $|0\rangle$ / $|1\rangle$ basis. Depending on the projective measurement, the qubits are projected onto the basis. However, in our implementation, we chose to use Bloch sphere Qiskit functionalities, as in the previous case, measuring directly destroys the phase value. A possible alternative solution is to measure on a different basis, such as circular (Y) or Hadamard (X). Measuring the complex field can be achieved in Qiskit, by doing a quantum state tomography, achievable by using the 'state_tomography_circuits', 'StateTomographyFitter' functions.

Before the quantum version of the GS method can be employed in real-world applications, a few issues need to be resolved: The algorithm is intricate and necessitates several qubits, depending on the resolution of the complex field we want to achieve. Due to this, applying the procedure on an existing quantum computer is challenging. Compared to the traditional version, the algorithm needs more complex implementations of the propagation step. We started implementing a FresnelDiffraction gate that takes input parameters such as the distance from the planes and the wavenumber. However, propagation is a highly complicated topic and convoluted methods were developed. In addition, the algorithm is noise-sensitive. The GS algorithm's quantum variant is more noise-sensitive than its conventional counterpart. The qubits are more prone to errors, so numerous error correction strategies have been developed. The

quantum variant of the GS algorithm is a potential new method for CGH despite these difficulties. It might speed up the computation, and the quantum version of the GS algorithm will be more useful as quantum computers get stronger.

4.2.1 Comparison Quantum-Classic

Quantum algorithms claim some advantages over the classic algorithm, supported by a quantum hardware capable of achieving superposition and entanglement, leveraging resources not present in classic hardware.
Parallelism: Quantum computers inherently process multiple states simultaneously through superposition, allowing faster manipulation of quantum states than classical sequential processing.
Efficient Linear Algebra: Quantum gates efficiently perform complex linear algebra operations, such as phase rotations and quantum Fourier transforms. These operations are crucial in Gerchberg-Saxton, potentially offering speedup over classical matrix operations.
Speedup for Quantum Transformations: Quantum Fourier transforms can be performed in logarithmic time on a quantum computer, whereas classical algorithms require polynomial time, providing an advantage for large data sizes.
Quantum Phases: Quantum algorithms naturally handle phase information, which is crucial in algorithms like Gerchberg-Saxton. Quantum phase rotations are precise and can be efficiently applied.
Quantum Memory: Quantum states are stored in quantum bits (qubits), enabling exponential data representation compared to classical bits. This is advantageous for large datasets.
Quantum Interference: Quantum states can undergo constructive or destructive interference, facilitating complex amplitude manipulations that are hard to emulate classically.
Potential Speedup: While the quantum advantage is problem-specific, quantum algorithms can potentially provide exponential speedup for specific problems, such as those involving Fourier transforms.
On the other hand, we have longer experience with classic algorithms, and there are some advantages over quantum computing:
Accessibility: Classical computing is mature and widely accessible, while quantum computers are still in their infancy. Developing and debugging algorithms on classical platforms is currently more straightforward.
Universal Problem Solving: Classical computers can solve a broader range of problems due to their general-purpose nature, while quantum computers excel at specific tasks that exploit quantum phenomena.
Cost and Resources: Currently, quantum computations are resource-intensive, requiring specialized hardware and noise mitigation techniques. Classical computations are more cost-effective for many problems.
Precision and Stability: Classical computations are more stable and less prone to errors than quantum computations, which are sensitive to noise and decoherence.
Interoperability: Classical software libraries and languages are well-established and interconnected, simplifying software development, integration, and reuse.

4.2.2 Results

Our circuit can run on a laptop machine with an Intel I9 CPU with 64 GB of RAM, and we tested on four qubits. A circuit with nine qubits was created, but the result is not provided. The results show some reconstructions correct after a variable number of iterations, as displayed in Figure 4.1. The qubit phase changes accordingly with the code provided, which can be considered a starting point to improve both the interference pattern and the propagation mechanism. Indeed, in this study, we aim to step into such a new field. As discussed in the next section, as part of improving such stages of the GS algorithm, different approaches can be explored, like other iterative phase retrieval algorithms or even single-shot diffraction pattern generation. Although the algorithm we created can be scaled by modifying one parameter, we noticed that the quantum simulation memory requirement increases quickly, saturating the memory. Simultaneously, in the classical computer, a longer time is required to produce the outcomes. Future work consists of improving both the circuit and the measurement, and eventually testing on quantum hardware via IBM account.

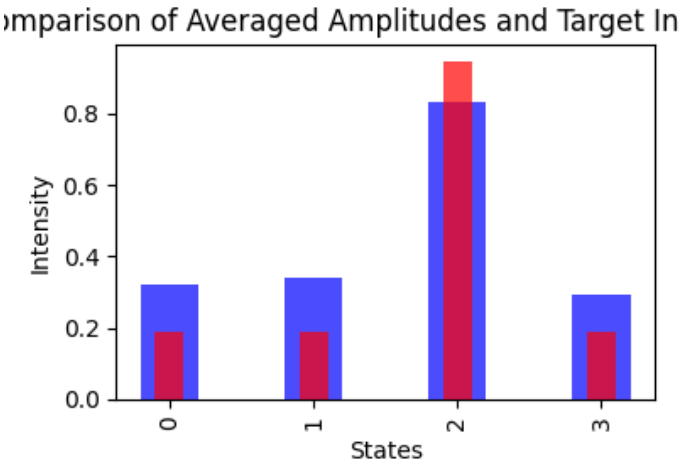

Figure 4.1: Comparison between amplitudes and target intensities after 35 iterations.

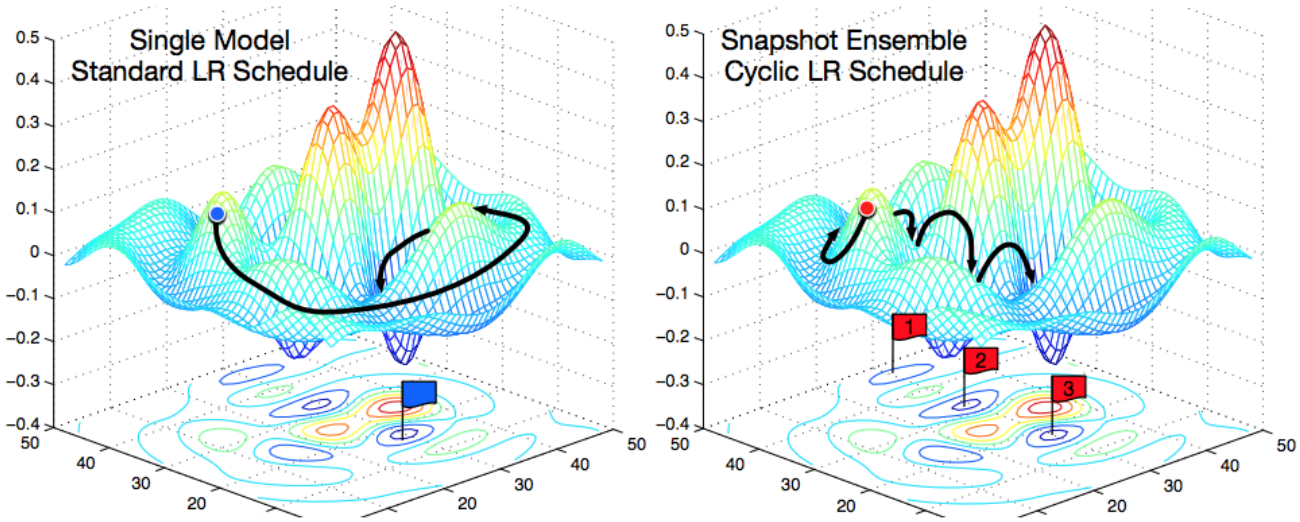

Figure 4.2: Visual representation of SGD approach. (Courtesy image from Feng Wang blogpost[1]))

4.3 Future Works

4.3.1 Stocastic Gradient Descent

In GS alternatives, the main data structures are the same, as transport equations must update complex planes, but there are different approaches to creating the right diffraction pattern. For example, we can consider such an approach as a minimization of a loss function that calculates the difference between the target and reconstruction image. An iterative optimisation process is the stochastic gradient descent (SGD). The SGD algorithm starts with a purely arbitrary assumption for the reconstruction picture. The method then moves toward the loss function's negative gradient to update the reconstruction image repeatedly. Because the SGD technique is stochastic, it does not take a deterministic route to the loss function's minimum. The SGD algorithm is an effective tool for reconstructing images. This process is repeated until the loss function converges to the specified tolerance. The technique can, however, be sensitive to measurement noise and problematic to converge. In addition, the outcome depends on initial conditions, and the optimal solution is not guaranteed. A single intensity measurement may be utilised to rebuild pictures using the SGD approach (Figure 4.2). In contrast, conventional image reconstruction techniques call for several intensity

[1]https://towardsdatascience.com/https-medium-com-reina-wang-tw-stochastic-gradient-descent-with-restarts-5f511975163

measurements. Recreating pictures from noisy data using the SGD method is also possible. This makes it a technology with much potential for holographic imaging in practical settings. Given this benefit and drawback of SGD and considering that it is an algorithm used massively in machine learning, a possible quantum-based approach could consider the application of recent quantum machine learning algorithms such as quantum stochastic gradient descent (QSGD) [Sweke et al., 2020]. In Sweke's work, for example, is presented a version in which SGD is used in a hybrid quantum-classical system. In Appendix A.6 is provided SGD applied to holography.

4.3.2 Quantum version of SGD

The quantum version of the stochastic gradient descent algorithm is similar to the classical version. Still, it uses quantum circuits to represent the complex field and the loss function. QSGD minimises a cost or loss function that quantifies the difference between the quantum circuit's predictions and target values. The quantum circuit's parameters, represented by a vector θ, must be optimized to minimize this cost function. It employs stochastic optimization. It processes training data in mini-batches to approximate the gradient of the cost function. Each mini-batch contains a subset of the training data. In the initialization stage, the quantum circuit with parameters θ is used to prepare quantum states, compute quantum observables or perform other quantum computations. The outcomes of these computations are used to estimate the gradient of the cost function to the parameters θ. QSGD leverages the principles of quantum measurement to estimate gradients. The quantum circuit is prepared in a certain state, and measurements are performed. The measurement outcomes, often represented as probabilities or expectation values, provide information about the gradient components concerning the parameters θ. Then, it updates the parameters θ based on the estimated gradient information. The update step is similar to classical SGD. The learning rate and the gradient sign are used to update each parameter. The algorithm repeats until a stopping criterion is met. This can be a maximum number of iterations, a convergence threshold, or other criteria defined by the user. Such workflow is analogous to the GS loop and so can replace the current implementation with a reduced effort. However, as such an algorithm runs on quantum hardware, it inherits a susceptibility to noise and errors. Therefore, quantum error mitigation techniques are required to reduce noise effects on the estimated gradients. We provide in Appendix A.7 with a possible approach of quantum SGD in Qiskit.

4.3.3 Odak Library for CGH and Odak Quantum Plugin

Odak is an open-source Python library that enables research and development in CGH. A key capability Odak provides is the simulation of light propagation and diffraction effects, which are integral physical phenomena underlying holographic reconstruction. At the core of Odak is the odak.wave module, which implements angular spectrum propagation of monochromatic coherent light waves between parallel planes. This allows for the modelling of diffraction and interference effects critical to CGH. Odak.wave natively represents complex optical field data using NumPy arrays, enabling integration with Python's ecosystems for scientific computing and automatic differentiation. Key components of odak.wave include functions to propagate wavefields at specified distances, apply aperture functions, model lens effects, and interface field data with common image representations. Odak.wave permits flexible definitions of parameters, including wavelength, sampling rates, and physical display dimensions. Upon this diffraction simulation framework, Odak incorporates tools for calculating optimal hologram patterns to reconstruct desired 3D scenes. This includes the generation of point light distributions representing the scene. Light from these reconstructed points can be back-propagated to a hologram plane, aiming to determine a phase-only hologram pattern that best reconstructs the scene upon forward re-propagation. Odak provides common objective functions, such as maximizing power in reconstruction peaks while minimizing background noise. Gradient-based optimization leveraging odak.wave's differentiable propagation can efficiently solve for the optimal scene-representing hologram. To enable applied CGH research, Odak integrates with several physical holographic display platforms. A key target display is the HoloPlay module, which accepts phase-only holograms to reconstruct mid-air 3D scenes. Odak provides functions to interface optimized holograms with HoloPlay's input stream for real-time video display. Support for phase-only spatial light modulators like those from Meadowlark Optics is also provided.

To enhance the capabilities of Quantum Stochastic Gradient Descent (QSGD) and facilitate seamless integration with quantum machine learning frameworks, we can explore the possibility of introducing a "Quantum Odak Plugin." This plugin would be designed to leverage the strengths of both QSGD and the Odak framework, which employs tensors in PyTorch, a popular deep-learning framework. The proposed plugin could be implemented using a dedicated

library called TorchQuantum, which would provide the bridge between quantum circuits and tensor-based operations. By integrating TorchQuantum into Odak, we would enable the seamless use of quantum circuits within the PyTorch ecosystem, mapping Gerchberg Saxton and SGD algorithms to their quantum versions.

4.3.4 TorchQuantum

TorchQuantum is an open-source library that facilitates the development of quantum machine learning models by integrating quantum circuits into PyTorch's deep learning ecosystem. It enables the construction of hybrid quantum-classical neural networks using PyTorch's familiar interfaces. At its core, TorchQuantum provides QuantumCircuit modules that encapsulate parameterized quantum circuit objects for usage in PyTorch neural networks. Common quantum logic gates are implemented to construct variational circuits, integrating PyTorch's automatic differentiation mechanisms to support the optimization of quantum parameters through backpropagation. For the execution of quantum circuits, TorchQuantum integrates with several quantum simulators and real quantum hardware, including IBM Quantum systems. Native PyTorch integrations are included for quantum emulators like Aer, PyTorchQVM, and PennyLane. Using TorchQuantum's interfaces, developers can leverage familiar PyTorch workflows for data loading, loss calculation, optimization, and model checkpointing to train quantum models. TorchQuantum incorporates common quantum machine learning applications, including quantum classifiers, autoencoders, and generative models, to provide starting points for developing novel quantum neural network architectures. Pre-built datasets and benchmarking tasks tailored for quantum machine learning are also provided. While nascent, TorchQuantum aims to promote research and education in quantum machine learning by lowering barriers to exploring quantum-enhanced deep learning techniques. It brings quantum circuits into GPU-accelerated training regimes familiar to deep learning practitioners. TorchQuantum is supported by active open-source community building integrations and applications.

4.3.5 Future Applications

Because of its promising skills in handling complicated and computationally heavy tasks, quantum computer-generated holography (QCGH) has the potential impact on a variety of sectors. This section investigates the prospective applications of QCGH and future research paths in this sector, stressing the foundations that are valuable for these goals. In addition to 3D displays [Tay et al., 2008, Pan et al., 2015], and augmented reality (AR) systems [He et al., 2019, Li et al., 2016, Maimone et al., 2017], one of the key uses of QCGH is in the presentation and storage of information. Holography allows for the storing and retrieval of three-dimensional (3D) information, with capacity and resolution improvements over typical two-dimensional (2D) approaches [Psaltis and Burr, 1998, Hesselink et al., 2004, Burr et al., 2001]. The principle behind this technology can be split into different required stages:

1. Laser Light Source: holographic data storage typically uses a laser light source, (red, green, or blue). It provides the coherent light required for recording and retrieving holographic data.

2. Spatial Light Modulator (SLM): An SLM encodes and modulates data onto the laser beam. It controls the light's phase and amplitude to create the interference patterns necessary for holography. The data, often in binary pixels (0s and 1s), is encoded on the SLM.

3. Beam Splitter: The beam is split into two beams—the reference beam and the data beam. The reference beam is directed to illuminate the entire storage medium uniformly, while the data beam encodes the SLM information.

4. Storage Medium: The storage medium is typically a thick, photosensitive material like a crystal or a polymer. This medium can record and store the holographic interference pattern created by the data beam and the reference beam.

5. Hologram Formation: When the data beam interferes with the reference beam on the storage medium, a 3D interference pattern or hologram is formed. The intensity and phase variations of the light at each point in the medium are recorded as a hologram. This process occurs for multiple data pages in the same volume of the storage medium.

6. Multiplexing: that means that multiple holograms (representing different data pages) can be stored at different angles (changing reference beam angle) within the same volume of the storage medium.

7. Data Retrieval: To retrieve data, a similar laser light source is used to illuminate the storage medium with the reference beam. This reconstructs the interference patterns and allows the data to be read. The data pages can be selectively retrieved by adjusting the angle of the reference beam.

QCGH can expand on these capabilities by allowing for the creation of more efficient and adaptable holograms, perhaps leading to improvements in high-density high-transfer data storage [Orlov et al., 2004]. Another possible use for QCGH is optical metrology and sensing. Holographic methods have been used to measure physical quantities precisely and detect phenomena such as temperature, tension, and strain [Asundi, 2011, Bennett et al., 1976, Paturzo et al., 2018]. The use of QCGH in various applications can increase accuracy, sensitivity, and speed, hence driving developments in domains like material research, structural health monitoring, and biomedicine. In addition, QCGH has potential uses in security and authentication. Because of their intricate patterns and the complexity of counterfeiting [Vather et al., 2018, Spagnolo and De Santis, 2011, Cheng et al., 2014], holograms are frequently utilised as security features in banknotes, passports, and other precious things [Nomura et al., 2005]. QCGH can allow the construction of more advanced and secure holographic features, boosting the security against counterfeiting and fraud, by harnessing the computing capabilities of quantum computers. Several additional research avenues should be examined to fully realise the promise of QCGH in these applications. First, developing efficient and effective hybrid quantum-classical holographic algorithms is critical to reaping the benefits of quantum computing. As indicated in the preceding section, hybrid quantum-classical techniques provide a promising path for this purpose, and additional study should be conducted to investigate their potential in the context of QCGH. Second, QCGH algorithm scalability should be considered, since it is critical to their actual implementation. Quantum computers with more qubits and lower error rates are required to solve more difficult holography issues. At the same time, developing fault-tolerant quantum algorithms and error correction techniques will be critical to providing scalable QCGH solutions. Third, investigating innovative quantum computing paradigms and hardware designs may open up new avenues for QCGH. Continuous-variable quantum computing (CVQC) [Adesso et al., 2014], for example, has shown potential in modelling and solving issues in quantum optics [Cerf et al., 2007]. Finally, multidisciplinary collaboration among researchers in quantum computing, holography, and application-specific domains is critical for developing QCGH. Interdisciplinary research can help create more effective and customised QCGH solutions, thereby expediting the realisation of their potential advantages by encouraging a greater knowledge of each discipline's underlying concepts and problems.

Chapter 5

Conclusion

We thoroughly explored the fields of holography, digital holography, and computer-generated holography (CGH) in this thesis. An extensive literature analysis covering the fundamentals of holography, the development of holographic methods, and the crucial shift from analogue to digital holography served as our first step in building a strong foundation. As we dug into its complexities, we investigated the early iterations of CGH, including the Detour-Phase Hologram and the Kinoform Hologram. Furthermore, we thoroughly investigated CGH algorithms, such as Iterative Fourier Transform and Iterative Phase Retrieval, explaining their importance in building intricate 3D holographic structures. The design and implementation of a quantum version of the Gerchberg-Saxton Algorithm was the primary contribution of this study. With its design for quantum computers, this quantum method promises quicker and more effective phase retrieval, opening up new possibilities for digital holography. To implement this innovative quantum method, we spoke about using quantum gates, the Quantum Fourier Transform, and the creation of quantum circuits with Qiskit.As we conclude this thesis, it is clear that there is a lot of potential in combining digital holography with quantum computing. Exciting applications in complex field reconstruction await the development of quantum stochastic gradient descent, quantum variants of the Gerchberg-Saxton Algorithm, and holography research utilising cutting-edge libraries such as Odak and TorchQuantum. The enormous potential of quantum-assisted holography in a variety of domains, from imaging to data storage and beyond, is depicted by this study, which acts as a first step towards the union of quantum and holographic technologies.

Bibliography

[goo, 2023] (2023). Suppressing quantum errors by scaling a surface code logical qubit. *Nature*, 614(7949):676–681.

[Abouraddy et al., 2001] Abouraddy, A. F., Saleh, B. E., Sergienko, A. V., and Teich, M. C. (2001). Quantum holography. *Optics Express*, 9(10):498–505.

[Adesso et al., 2014] Adesso, G., Ragy, S., and Lee, A. R. (2014). Continuous variable quantum information: Gaussian states and beyond. *Open Systems & Information Dynamics*, 21(01n02):1440001.

[Akşit and Kavaklı, 2023] Akşit, K. and Kavaklı, K. (2023). Flexible modeling of next-generation displays using a differentiable toolkit. In *Practical Holography XXXVII: Displays, Materials, and Applications*, volume 12445, pages 131–132. SPIE.

[Albash and Lidar, 2018] Albash, T. and Lidar, D. A. (2018). Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002.

[Amin et al., 2018] Amin, M. H., Andriyash, E., Rolfe, J., Kulchytskyy, B., and Melko, R. (2018). Quantum boltzmann machine. *Physical Review X*, 8(2):021050.

[Andreasson et al., 2019] Andreasson, P., Johansson, J., Liljestrand, S., and Granath, M. (2019). Quantum error correction for the toric code using deep reinforcement learning. *Quantum*, 3:183.

[Ashley et al., 2000] Ashley, J., Bernal, M.-P., Burr, G. W., Coufal, H., Guenther, H., Hoffnagle, J. A., Jefferson, C. M., Marcus, B., Macfarlane, R. M., Shelby, R. M., et al. (2000). Holographic data storage technology. *IBM journal of research and development*, 44(3):341–368.

[Asundi, 2011] Asundi, A. (2011). *Digital holography for MEMS and microsystem metrology*, volume 7. Wiley Online Library.

[Bally, 2013] Bally, G. (2013). *Holography in Medicine and Biology: Proceedings of the International Workshop, Münster, Fed. Rep. of Germany, March 14–15, 1979*, volume 18. Springer.

[Bauschke et al., 2002] Bauschke, H. H., Combettes, P. L., and Luke, D. R. (2002). Phase retrieval, error reduction algorithm, and fienup variants: a view from convex optimization. *JOSA A*, 19(7):1334–1345.

[Beach et al., 2003] Beach, G., Lomont, C., and Cohen, C. (2003). Quantum image processing (quip). In *32nd Applied Imagery Pattern Recognition Workshop, 2003. Proceedings.*, pages 39–44. IEEE.

[Bell, 1964] Bell, J. S. (1964). On the einstein podolsky rosen paradox. *Physics Physique Fizika*, 1(3):195.

[Bennett et al., 1976] Bennett, J., Anderson, A., McInnes, P., and Whitaker, A. (1976). Microwave holographic metrology of large reflector antennas. *IEEE Transactions on Antennas and Propagation*, 24(3):295–303.

[Benton and Bove Jr, 2008] Benton, S. A. and Bove Jr, V. M. (2008). *Holographic imaging*. John Wiley & Sons.

[Biamonte et al., 2017] Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. (2017). Quantum machine learning. *Nature*, 549(7671):195–202.

[Briegel et al., 2009] Briegel, H. J., Browne, D. E., Dür, W., Raussendorf, R., and Van den Nest, M. (2009). Measurement-based quantum computation. *Nature Physics*, 5(1):19–26.

[Brown and Lohmann, 1969] Brown, B. and Lohmann, A. (1969). Computer-generated binary holograms. *IBM Journal of research and Development*, 13(2):160–168.

[Brown and Lohmann, 1966] Brown, B. R. and Lohmann, A. W. (1966). Complex spatial filtering with binary masks. *Applied optics*, 5(6):967–969.

[Brylinski and Brylinski, 2002] Brylinski, J.-L. and Brylinski, R. (2002). Universal quantum gates. *Mathematics of quantum computation*, 79.

[Bryngdahl and Wyrowski, 1990] Bryngdahl, O. and Wyrowski, F. (1990). I digital holography–computer-generated holograms. In *Progress in optics*, volume 28, pages 1–86. Elsevier.

[Burr et al., 2001] Burr, G. W., Jefferson, C. M., Coufal, H., Jurich, M., Hoffnagle, J. A., Macfarlane, R. M., and Shelby, R. M. (2001). Volume holographic data storage at an areal density of 250 gigapixels/in. 2. *Optics Letters*, 26(7):444–446.

[Caraiman and Manta, 2009] Caraiman, S. and Manta, V. I. (2009). New applications of quantum algorithms to computer graphics: the quantum random sample consensus algorithm. In *Proceedings of the 6th ACM conference on Computing frontiers*, pages 81–88.

[Casasent, 1977] Casasent, D. (1977). Spatial light modulators. *Proceedings of the IEEE*, 65(1):143–157.

[Castelvecchi, 2017] Castelvecchi, D. (2017). Ibm's quantum cloud computer goes commercial. *Nature*, 543(7644).

[Cerf et al., 2007] Cerf, N. J., Leuchs, G., and Polzik, E. S. (2007). *Quantum information with continuous variables of atoms and light*. World Scientific.

[Chang et al., 2022] Chang, C., Wang, D., Zhu, D., Li, J., Xia, J., and Zhang, X. (2022). Deep-learning-based computer-generated hologram from a stereo image pair. *Optics Letters*, 47(6):1482–1485.

[Chen et al., 2022] Chen, H., Huang, L., Liu, T., and Ozcan, A. (2022). Fourier imager network (fin): A deep neural network for hologram reconstruction with superior external generalization. *Light: Science & Applications*, 11(1):254.

[Chen et al., 2021] Chen, N., Wang, C., and Heidrich, W. (2021). Holographic 3d particle imaging with model-based deep network. *IEEE Transactions on Computational Imaging*, 7:288–296.

[Chen et al., 2018] Chen, X., Zhang, H., Zhu, T., Yao, Y., Jin, D., and Fei, P. (2018). Generative adversarial network (gan) enabled on-chip contact microscopy. *bioRxiv*, page 478982.

[Chen et al., 2019] Chen, Y., Chi, Y., Fan, J., and Ma, C. (2019). Gradient descent with random initialization: Fast global convergence for nonconvex phase retrieval. *Mathematical Programming*, 176:5–37.

[Cheng et al., 2014] Cheng, C.-J., Hwang, W.-J., Zeng, H.-Y., and Lin, Y.-C. (2014). A fragile watermarking algorithm for hologram authentication. *Journal of display technology*, 10(4):263–271.

[Choi et al., 2021] Choi, S., Gopakumar, M., Peng, Y., Kim, J., and Wetzstein, G. (2021). Neural 3d holography: learning accurate wave propagation models for 3d holographic virtual and augmented reality displays. *ACM Transactions on Graphics (TOG)*, 40(6):1–12.

[Cooley and Tukey, 1965] Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301.

[Coppersmith, 2002] Coppersmith, D. (2002). An approximate fourier transform useful in quantum factoring. *arXiv preprint quant-ph/0201067*.

[Coufal et al., 2000] Coufal, H. J., Psaltis, D., Sincerbox, G. T., et al. (2000). *Holographic data storage*, volume 8. Springer.

[Creswell et al., 2018] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65.

[Defienne et al., 2021] Defienne, H., Ndagano, B., Lyons, A., and Faccio, D. (2021). Polarization entanglement-enabled quantum holography. *Nature Physics*, 17(5):591–597.

[Deutsch, 1985] Deutsch, D. (1985). Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117.

[Dirac, 1958] Dirac, P. (1958). The principles of quantum mechanics, clarendon. *Oxford. Dederichs P H.(1972). Solid State Physics*, 27:135.

[DiVincenzo, 2000] DiVincenzo, D. P. (2000). The physical implementation of quantum computation. *Fortschritte der Physik: Progress of Physics*, 48(9-11):771–783.

[Efron, 1994] Efron, U. (1994). *Spatial light modulator technology: materials, devices, and applications*, volume 47. CRC press.

[Einstein et al., 1935] Einstein, A., Podolsky, B., and Rosen, N. (1935). Can quantum-mechanical description of physical reality be considered complete? *Physical review*, 47(10):777.

[Ekert and Jozsa, 1998] Ekert, A. and Jozsa, R. (1998). Quantum algorithms: entanglement–enhanced information processing. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 356(1743):1769–1782.

[Endo et al., 2021] Endo, S., Cai, Z., Benjamin, S. C., and Yuan, X. (2021). Hybrid quantum-classical algorithms and quantum error mitigation. *Journal of the Physical Society of Japan*, 90(3):032001.

[Eybposh et al., 2020] Eybposh, M. H., Caira, N. W., Atisa, M., Chakravarthula, P., and Pégard, N. C. (2020). Deepcgh: 3d computer-generated holography using deep learning. *Optics Express*, 28(18):26636–26650.

[Fienup, 1982] Fienup, J. R. (1982). Phase retrieval algorithms: a comparison. *Applied optics*, 21(15):2758–2769.

[Freedman et al., 2003] Freedman, M., Kitaev, A., Larsen, M., and Wang, Z. (2003). Topological quantum computation. *Bulletin of the American Mathematical Society*, 40(1):31–38.

[GABOR, 1948] GABOR, D. (1948). A new microscopic principle. *Nature*, 161(4098):777–778.

[Gerchberg and Saxton, 1973] Gerchberg, R. and Saxton, W. (1973). Wave phase from image and diffraction plane pictures. *Image processing and computer-aided design in electron optics*, pages 66–81.

[Gerchberg, 1972] Gerchberg, R. W. (1972). A practical algorithm for the determination of plane from image and diffraction pictures. *Optik*, 35(2):237–246.

[Goodman, 1967] Goodman, J. W. (1967). Digital image formation from electronically detected holograms. In *Computerized Imaging Techniques*, volume 10, pages 176–181. SPIE.

[Goodman, 2005] Goodman, J. W. (2005). *Introduction to Fourier optics*. Roberts and Company publishers.

[Grover, 1996] Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219.

[Hariharan, 1996] Hariharan, P. (1996). *Optical Holography: Principles, techniques and applications*. Cambridge University Press.

[Harrow et al., 2009] Harrow, A. W., Hassidim, A., and Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502.

[He et al., 2019] He, Z., Sui, X., Jin, G., and Cao, L. (2019). Progress in virtual reality and augmented reality based on holographic display. *Applied optics*, 58(5):A74–A81.

[Heflinger et al., 1966] Heflinger, L., Wuerker, R., and Brooks, R. E. (1966). Holographic interferometry. *Journal of Applied Physics*, 37(2):642–649.

[Hesselink et al., 2004] Hesselink, L., Orlov, S. S., and Bashaw, M. C. (2004). Holographic data storage systems. *Proceedings of the IEEE*, 92(8):1231–1280.

[Hirsch et al., 1971] Hirsch, P. M., Jordan Jr, J. A., and Lesem, L. B. (1971). Method of making an object dependent diffuser. US Patent 3,619,022.

[Horisaki et al., 2018] Horisaki, R., Takagi, R., and Tanida, J. (2018). Deep-learning-generated holography. *Applied optics*, 57(14):3859–3863.

[Huang et al., 2022a] Huang, L., Yang, X., Liu, T., and Ozcan, A. (2022a). Few-shot transfer learning for holographic image reconstruction using a recurrent neural network. *APL Photonics*, 7(7).

[Huang et al., 2022b] Huang, Y., Zhu, Y., Qiao, X., Su, X., Dustdar, S., and Zhang, P. (2022b). Towards holographic video communications: A promising ai-driven solution. *IEEE Communications Magazine*.

[Jiao et al., 2018] Jiao, S., Jin, Z., Chang, C., Zhou, C., Zou, W., and Li, X. (2018). Compression of phase-only holograms with jpeg standard and deep learning. *Applied Sciences*, 8(8):1258.

[Jozsa and Linden, 2003] Jozsa, R. and Linden, N. (2003). On the role of entanglement in quantum-computational speed-up. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 459(2036):2011–2032.

[Kak, 1995] Kak, S. C. (1995). Quantum neural computing. *Advances in imaging and electron physics*, 94:259–313.

[Kang et al., 2021] Kang, J.-W., Park, B.-S., Kim, J.-K., Kim, D.-W., and Seo, Y.-H. (2021). Deep-learning-based hologram generation using a generative model. *Applied Optics*, 60(24):7391–7399.

[Kavaklı and Akşit, 2022] Kavaklı, K. and Akşit, K. (2022). Introduction to odak: a differentiable toolkit for optical sciences, vision sciences and computer graphics. In *Frontiers in Optics*, pages FTu1A–1. Optica Publishing Group.

[Kavakli et al., 2022] Kavakli, K., Walton, D. R., Antipa, N., Mantiuk, R., Lanman, D., and Akşit, K. (2022). Optimizing vision and visuals: lectures on cameras, displays and perception. In *ACM SIGGRAPH 2022 Courses*, pages 1–66.

[Khan et al., 2021] Khan, A., Zhijiang, Z., Yu, Y., Khan, M. A., Yan, K., and Aziz, K. (2021). Gan-holo: Generative adversarial networks-based generated holography using deep learning. *Complexity*, 2021:1–7.

[Kim and Kim, 2008] Kim, S.-C. and Kim, E.-S. (2008). Effective generation of digital holograms of three-dimensional objects using a novel look-up table method. *Applied Optics*, 47(19):D55–D62.

[Kitaev, 1995] Kitaev, A. Y. (1995). Quantum measurements and the abelian stabilizer problem. *arXiv preprint quant-ph/9511026*.

[Kreis, 2006] Kreis, T. (2006). *Handbook of holographic interferometry: optical and digital methods*. John Wiley & Sons.

[Lanzagorta and Uhlmann, 2005a] Lanzagorta, M. and Uhlmann, J. (2005a). Quantum rendering: an introduction to quantum computing, quantum algorithms and their applications to computer graphics. In *ACM SIGGRAPH 2005 Courses*, pages 1–es.

[Lanzagorta and Uhlmann, 2005b] Lanzagorta, M. and Uhlmann, J. K. (2005b). Hybrid quantum-classical computing with applications to computer graphics. In *ACM SIGGRAPH 2005 Courses*, pages 2–es.

[Lee, 1978] Lee, W.-H. (1978). Iii computer-generated holograms: Techniques and applications. In *Progress in optics*, volume 16, pages 119–232. Elsevier.

[Lee et al., 2018] Lee, Y.-H., Kim, D.-W., and Seo, Y.-H. (2018). Hologram generation method using dcgan. In *Proceedings of the Korean Society of Broadcast Engineers Conference*, pages 209–210. The Korean Institute of Broadcast and Media Engineers.

[Leith and Upatnieks, 1964] Leith, E. N. and Upatnieks, J. (1964). Wavefront reconstruction with diffused illumination and three-dimensional objects. *Josa*, 54(11):1295–1301.

[Leith and Upatnieks, 1965] Leith, E. N. and Upatnieks, J. (1965). Holograms: their properties and uses. *Optical Engineering*, 4(1):3–6.

[Lenart and Owall, 2005] Lenart, T. and Owall, V. (2005). Xstream-a hardware accelerator for digital holographic imaging. In *2005 12th IEEE International Conference on Electronics, Circuits and Systems*, pages 1–4. IEEE.

[Lesem et al., 1969] Lesem, L., Hirsch, P., and Jordan, J. (1969). The kinoform: a new wavefront reconstruction device. *IBM Journal of Research and Development*, 13(2):150–155.

[Li et al., 2016] Li, G., Lee, D., Jeong, Y., Cho, J., and Lee, B. (2016). Holographic display for see-through augmented reality using mirror-lens holographic optical element. *Optics letters*, 41(11):2486–2489.

[Lidar and Brun, 2013] Lidar, D. A. and Brun, T. A. (2013). *Quantum error correction*. Cambridge university press.

[Liu et al., 2023a] Liu, K., Wu, J., He, Z., and Cao, L. (2023a). 4k-dmdnet: diffraction model-driven network for 4k computer-generated holography. *Opto-Electronic Advances*, pages 220135–1.

[Liu et al., 2023b] Liu, Q., Chen, J., Qiu, B., Wang, Y., and Liu, J. (2023b). Dcpnet: a dual-channel parallel deep neural network for high quality computer-generated holography. *Optics Express*, 31(22):35908–35921.

[Lloyd et al., 2014] Lloyd, S., Mohseni, M., and Rebentrost, P. (2014). Quantum principal component analysis. *Nature Physics*, 10(9):631–633.

[Lohmann and Paris, 1967] Lohmann, A. W. and Paris, D. (1967). Binary fraunhofer holograms, generated by computer. *Applied optics*, 6(10):1739–1748.

[Lu, 1968] Lu, S. (1968). Generating multiple images for integrated circuits by fourier-transform holograms. *Proceedings of the IEEE*, 56(1):116–117.

[Maimone et al., 2017] Maimone, A., Georgiou, A., and Kollin, J. S. (2017). Holographic near-eye displays for virtual and augmented reality. *ACM Transactions on Graphics (Tog)*, 36(4):1–16.

[Marchesini et al., 2003] Marchesini, S., He, H., Chapman, H. N., Hau-Riege, S. P., Noy, A., Howells, M. R., Weierstall, U., and Spence, J. C. (2003). X-ray image reconstruction from a diffraction pattern alone. *Physical Review B*, 68(14):140101.

[Matsushima, 2020] Matsushima, K. (2020). *Introduction to Computer Holography: Creating Computer-Generated Holograms as the Ultimate 3D Image*. Springer Nature.

[Matsushima and Shimobaba, 2009] Matsushima, K. and Shimobaba, T. (2009). Band-limited angular spectrum method for numerical simulation of free-space propagation in far and near fields. *Optics express*, 17(22):19662–19673.

[Matthews and Nairn, 2023] Matthews, J. and Nairn, A. (2023). Holographic abba: Examining fan responses to abba's virtual "live" concert. *Popular Music and Society*, pages 1–22.

[McClean et al., 2016] McClean, J. R., Romero, J., Babbush, R., and Aspuru-Guzik, A. (2016). The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023.

[Montanaro, 2016] Montanaro, A. (2016). Quantum algorithms: an overview. *npj Quantum Information*, 2(1):1–8.

[Moreland and Angel, 2003] Moreland, K. and Angel, E. (2003). The fft on a gpu. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 112–119.

[Nielsen and Chuang, 2002] Nielsen, M. A. and Chuang, I. (2002). Quantum computation and quantum information.

[Nielsen and Chuang, 2001] Nielsen, M. A. and Chuang, I. L. (2001). Quantum computation and quantum information. *Phys. Today*, 54(2):60.

[Nielsen and Chuang, 2010] Nielsen, M. A. and Chuang, I. L. (2010). *Quantum computation and quantum information*. Cambridge university press.

[Nishi et al., 2005] Nishi, S., Shiba, K., Mori, K., Nakayama, S., and Murashima, S. (2005). Fast calculation of computer-generated fresnel hologram utilizing distributed parallel processing and array operation. *Optical review*, 12:287–292.

[Nomura et al., 2005] Nomura, T., Tajahuerce, E., Matoba, O., and Javidi, B. (2005). Applications of digital holography for information security. *Optical and Digital Techniques for Information Security*, pages 241–269.

[Oikawa et al., 2011] Oikawa, M., Shimobaba, T., Masuda, N., and Ito, T. (2011). Computer-generated hologram using an approximate fresnel integral. *Journal of Optics*, 13(7):075405.

[Orlov et al., 2004] Orlov, S. S., Phillips, W., Bjornson, E., Takashima, Y., Sundaram, P., Hesselink, L., Okas, R., Kwan, D., and Snyder, R. (2004). High-transfer-rate high-capacity holographic disk data-storage system. *Applied Optics*, 43(25):4902–4914.

[Ostrovsky and Butusov,] Ostrovsky, Y. I. and Butusov, M. M. *Interferometry by holography*.

[Pan et al., 2015] Pan, Y., Liu, J., Li, X., and Wang, Y. (2015). A review of dynamic holographic three-dimensional display: algorithms, devices, and systems. *IEEE Transactions on Industrial Informatics*, 12(4):1599–1610.

[Paturzo et al., 2018] Paturzo, M., Pagliarulo, V., Bianco, V., Memmolo, P., Miccio, L., Merola, F., and Ferraro, P. (2018). Digital holography, a metrological tool for quantitative analysis: Trends and future applications. *Optics and Lasers in Engineering*, 104:32–47.

[Peruzzo et al., 2014] Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., Aspuru-Guzik, A., and O'brien, J. L. (2014). A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):4213.

[Poon, 2006] Poon, T.-C. (2006). *Digital holography and three-dimensional display: Principles and Applications*. Springer Science & Business Media.

[Porter, 1970] Porter, R. P. (1970). Diffraction-limited, scalar image formation with holograms of arbitrary shape. *JOSA*, 60(8):1051–1059.

[Psaltis and Burr, 1998] Psaltis, D. and Burr, G. W. (1998). Holographic data storage. *Computer*, 31(2):52–60.

[Ramsey et al., 2005] Ramsey, J., Hall, T., and Bidnyk, S. (2005). Poged: A fast alternative to computer generated holograms design by simulated annealing. *Diffractive Optics, Warsaw, Poland*, pages 23–24.

[Rastogi, 2013] Rastogi, P. K. (2013). *Holographic interferometry: principles and methods*, volume 68. Springer.

[Raussendorf et al., 2003] Raussendorf, R., Browne, D. E., and Briegel, H. J. (2003). Measurement-based quantum computation on cluster states. *Physical review A*, 68(2):022312.

[Rebentrost et al., 2014] Rebentrost, P., Mohseni, M., and Lloyd, S. (2014). Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503.

[Rivenson et al., 2013] Rivenson, Y., Stern, A., and Javidi, B. (2013). Overview of compressive sensing techniques applied in holography. *Applied optics*, 52(1):A423–A432.

[Rivenson et al., 2019] Rivenson, Y., Wu, Y., and Ozcan, A. (2019). Deep learning in holography and coherent imaging. *Light: Science & Applications*, 8(1):85.

[Rivenson et al., 2018] Rivenson, Y., Zhang, Y., Günaydın, H., Teng, D., and Ozcan, A. (2018). Phase recovery and holographic image reconstruction using deep learning in neural networks. *Light: Science & Applications*, 7(2):17141–17141.

[Rivest et al., 1978] Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

[Ruan et al., 2021] Ruan, Y., Xue, X., and Shen, Y. (2021). Quantum image processing: opportunities and challenges. *Mathematical Problems in Engineering*, 2021:1–8.

[Schrödinger, 1926] Schrödinger, E. (1926). An undulatory theory of the mechanics of atoms and molecules. *Physical review*, 28(6):1049.

[Schuld and Petruccione, 2018] Schuld, M. and Petruccione, F. (2018). *Supervised learning with quantum computers*, volume 17. Springer.

[Schuld et al., 2015] Schuld, M., Sinayskiy, I., and Petruccione, F. (2015). An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185.

[Shi et al., 2021] Shi, L., Li, B., Kim, C., Kellnhofer, P., and Matusik, W. (2021). Towards real-time photorealistic 3d holography with deep neural networks. *Nature*, 591(7849):234–239.

[Shimobaba et al., 2022] Shimobaba, T., Blinder, D., Birnbaum, T., Hoshi, I., Shiomi, H., Schelkens, P., and Ito, T. (2022). Deep-learning computational holography: A review. *Frontiers in Photonics*, 3:8.

[Shimobaba et al., 2017] Shimobaba, T., Endo, Y., Hirayama, R., Nagahama, Y., Takahashi, T., Nishitsuji, T., Kakue, T., Shiraki, A., Takada, N., Masuda, N., et al. (2017). Autoencoder-based holographic image restoration. *Applied Optics*, 56(13):F27–F30.

[Shor, 1999] Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332.

[Shortt et al., 2006] Shortt, A. E., Naughton, T. J., and Javidi, B. (2006). Compression of optically encrypted digital holograms using artificial neural networks. *Journal of Display Technology*, 2(4):401–410.

[Simon et al., 2008] Simon, P., Lichte, H., Formanek, P., Lehmann, M., Huhle, R., Carrillo-Cabrera, W., Harscher, A., and Ehrlich, H. (2008). Electron holography of biological samples. *Micron*, 39(3):229–256.

[Sokolov et al., 2001] Sokolov, I., Kolobov, M., Gatti, A., and Lugiato, L. (2001). Quantum holographic teleportation. *Optics Communications*, 193(1-6):175–180.

[Spagnolo and De Santis, 2011] Spagnolo, G. S. and De Santis, M. (2011). Holographic watermarking for authentication of cut images. *Optics and Lasers in Engineering*, 49(12):1447–1455.

[Steane, 1998] Steane, A. (1998). Quantum computing. *Reports on Progress in Physics*, 61(2):117.

[Sweke et al., 2020] Sweke, R., Wilde, F., Meyer, J., Schuld, M., Fährmann, P. K., Meynard-Piganeau, B., and Eisert, J. (2020). Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum*, 4:314.

[Tay et al., 2008] Tay, S., Blanche, P.-A., Voorakaranam, R., Tunç, A., Lin, W., Rokutanda, S., Gu, T., Flores, D., Wang, P., Li, G., et al. (2008). An updatable holographic three-dimensional display. *Nature*, 451(7179):694–698.

[Töpfer et al., 2022] Töpfer, S., Gilaberte Basset, M., Fuenzalida, J., Steinlechner, F., Torres, J. P., and Gräfe, M. (2022). Quantum holography with undetected light. *Science advances*, 8(2):eabl4301.

[Umul, 2005] Umul, Y. Z. (2005). Equivalent functions for the fresnel integral. *Optics express*, 13(21):8469–8482.

[Vanligten and Osterberg, 1966] Vanligten, R. F. and Osterberg, H. (1966). Holographic microscopy. *Nature*, 211(5046):282–283.

[Vather et al., 2018] Vather, D., Naydenova, I., Cody, D., Zawadzka, M., Martin, S., Mihaylova, E., Curran, S., Duffy, P., Portillo, J., Connell, D., et al. (2018). Serialized holography for brand protection and authentication. *Applied optics*, 57(22):E131–E137.

[Waters, 1968] Waters, J. P. (1968). Three-dimensional fourier-transform method for synthesizing binary holograms. *JOSA*, 58(9):1284–1288.

[Wen et al., 2005] Wen, M., Yao, J., Wong, D. W., and Chen, G. C. (2005). Holographic diffuser design using a modified genetic algorithm. *Optical Engineering*, 44(8):085801–085801.

[Williams et al., 1998] Williams, C. P., Clearwater, S. H., et al. (1998). *Explorations in quantum computing*. Springer.

[Wu et al., 2021a] Wu, J., Liu, K., Sui, X., and Cao, L. (2021a). High-speed computer-generated holography using an autoencoder-based deep neural network. *Optics Letters*, 46(12):2908–2911.

[Wu et al., 2021b] Wu, Y., Wang, J., Chen, C., Liu, C.-J., Jin, F.-M., and Chen, N. (2021b). Adaptive weighted gerchberg-saxton algorithm for generation of phase-only hologram with artifacts suppression. *Optics express*, 29(2):1412–1427.

[Wyrowski, 1990] Wyrowski, F. (1990). Diffraction efficiency of analog and quantized digital amplitude holograms: analysis and manipulation. *JOSA A*, 7(3):383–393.

[Wyrowski and Bryngdahl, 1988] Wyrowski, F. and Bryngdahl, O. (1988). Iterative fourier-transform algorithm applied to computer holography. *JOSA A*, 5(7):1058–1065.

[Yamaguchi and Zhang, 1997] Yamaguchi, I. and Zhang, T. (1997). Phase-shifting digital holography. *Opt. Lett.*, 22(16):1268–1270.

[Yang et al., 1994] Yang, G.-z., Dong, B.-z., Gu, B.-y., Zhuang, J.-y., and Ersoy, O. K. (1994). Gerchberg–saxton and yang–gu algorithms for phase retrieval in a nonunitary transform system: a comparison. *Applied optics*, 33(2):209–218.

[Yao, 1993] Yao, A. C.-C. (1993). Quantum circuit complexity. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361. IEEE.

[Yaraş et al., 2010] Yaraş, F., Kang, H., and Onural, L. (2010). State of the art in holographic displays: a survey. *Journal of display technology*, 6(10):443–454.

[Zhang and Yamaguchi, 1998] Zhang, T. and Yamaguchi, I. (1998). Three-dimensional microscopy with phase-shifting digital holography. *Optics letters*, 23(15):1221–1223.

[Zhao and Chi, 2020] Zhao, T. and Chi, Y. (2020). Modified gerchberg–saxton (gs) algorithm and its application. *Entropy*, 22(12):1354.

[Zheng et al., 2023] Zheng, H., Peng, J., Wang, Z., Shui, X., Yu, Y., and Xia, X. (2023). Diffraction model-driven neural network trained using hybrid domain loss for real-time and high-quality computer-generated holography. *Optics Express*, 31(12):19931–19944.

[Zhou et al., 2020] Zhou, L., Wang, S.-T., Choi, S., Pichler, H., and Lukin, M. D. (2020). Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067.

Appendix A

Code

In the appendix, we list the code written for this thesis.

A.1 Gerchberg-Saxton algorithm

Original idea of the Gerchberg Saxton:

```
import numpy as np
from scipy.fftpack import fft2, ifft2
def gs_algorithm(I, phi0, max_iter=1000, tol=1e-6):
    """
    Implements the Gerchberg-Saxton algorithm for phase retrieval.
    Parameters:
    -----------
    I : ndarray
        Measured intensity distribution.
    phi0 : ndarray
        Initial estimate of the phase distribution.
    max_iter : int, optional
        Maximum number of iterations.
    tol : float, optional
        Tolerance for convergence.
    Returns:
    --------
    phi : ndarray
        Reconstructed phase distribution.
    """
    # Initialize variables
    phi = phi0
    E = np.sqrt(I) * np.exp(1j * phi)
    prev_error = np.inf
    # Iterate until convergence or maximum number of iterations
    for i in range(max_iter):
        # Estimate complex field from phase distribution
        E_hat = fft2(E)
        # Update amplitude of complex field
        E = np.sqrt(I) * np.exp(1j * np.angle(E_hat))
        # Update phase distribution
        phi = np.angle(ifft2(E))
        # Check for convergence
```

```
        error = np.mean(np.abs(np.exp(1j * phi) - E))
        if error < tol:
            break
        # Check for increase in error
        if error > prev_error:
            print(f"Error␣increased␣at␣iteration␣{i}")
            break
        prev_error = error
    return phi
```

Algorithm used to produce images in Chapter 3 idea of the Gerchberg Saxton:

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image


def GS_algorithm(F, I, max_iter):
    # F: The known amplitude of the object
    # I: The measured intensity pattern
    # max_iter: Maximum number of iterations

    M, N = F.shape
    G = np.random.rand(M, N) + 1j * np.random.rand(M, N) # Initialize guess for the
        ↪ phase
    G = np.exp(1j * np.angle(G)) # Convert the guess to a complex field with unit
        ↪ magnitude

    for iteration in range(max_iter):
        # Step 1: Compute the Fourier transform of the guess
        G_hat = np.fft.fftshift(np.fft.fft2(G))

        # Step 2: Replace the magnitude with the measured intensity
        G_hat = np.sqrt(I) * np.exp(1j * np.angle(G_hat))

        # Step 3: Compute the inverse Fourier transform
        G_new = np.fft.ifft2(np.fft.ifftshift(G_hat))

        # Step 4: Replace the magnitude with the known amplitude
        G_new = F * np.exp(1j * np.angle(G_new))

        # Update the guess for the complex field
        G = G_new

    return G

# Example usage:
if __name__ == '__main__':
    # Load an image to use as the object amplitude
    image_path = "test.png" # Replace with the path to your image
    object_amplitude = np.array(Image.open(image_path).convert("L")) # Load image and
        ↪ convert to grayscale
```

```
    # Measure the diffraction pattern
    object_phase = np.random.rand(*object_amplitude.shape) * 2 * np.pi
    object_field = object_amplitude * np.exp(1j * object_phase)
    diffraction_pattern = np.abs(np.fft.fftshift(np.fft.fft2(object_field))) ** 2

    # Use the GS algorithm to reconstruct the object's phase
    reconstructed_field = GS_algorithm(object_amplitude, diffraction_pattern, max_iter
        ↪ =100)

    # Calculate the absolute difference between the original and reconstructed images
    difference_image = np.abs(object_amplitude - np.abs(reconstructed_field))

    # Display the results
    plt.figure(figsize=(12, 8))

    plt.subplot(141)
    plt.title("Original Object")
    plt.imshow(object_amplitude, cmap='gray')
    plt.axis('off')

    plt.subplot(142)
    plt.title("Measured Diffraction Pattern")
    plt.imshow(diffraction_pattern, cmap='gray')
    plt.axis('off')

    plt.subplot(143)
    plt.title("Reconstructed Object")
    plt.imshow(np.abs(reconstructed_field), cmap='gray')
    plt.axis('off')

    plt.subplot(144)
    plt.title("Difference")
    plt.imshow(difference_image, cmap='viridis')
    plt.axis('off')

    plt.show()
```

A.2 Gerchberg-Saxton Transport Function Code

Parameters for the transport function.

```
wavelength = 515e-9
k = wavenumber(wavelength)
dx = 0.000008
resolution = [64, 64]
distance = 0.15
```

Transformation of the images:

```
image_path = 'test.png'
image = Image.open(image_path)
```

```
# Define the transformation to apply to the image
transform = transforms.Compose([
    transforms.Resize((resolution[0], resolution[1])),
    transforms.Grayscale(num_output_channels=1),
    transforms.ToTensor()
])

# Apply the transformation to the image
target = transform(image)
```

Propagation of the beam:

```
for i in t:
    hologram = zero_pad(hologram)
    recon_field = propagate_beam(hologram, k, distance, dx, wavelength, propagation_type)
        ↪
    recon_field = crop_center(recon_field)
    recon_field = set_amplitude(recon_field, target)
    recon_field = zero_pad(recon_field)
    hologram = propagate_beam(recon_field, k, -distance, dx, wavelength,
        ↪ propagation_type)
    hologram = crop_center(hologram)
    hologram = set_amplitude(hologram, amplitude)
```

A.3 Quantum Fourier Transform for Gerchberg-Saxton Code

Example of a QQFT in qiskit for 4 bits.

```
import qiskit
from math import pi

qc = qiskit.QuantumCircuit(4)

# Apply hadamard gate to first qubit
qc.h(0)

# Controlled phase rotations on remaining qubits
qc.cp(pi/2, 0, 1)
qc.cp(pi/4, 0, 2)
qc.cp(pi/8, 0, 3)

qc.cp(pi/4, 1, 2)
qc.cp(pi/8, 1, 3)

qc.cp(pi/8, 2, 3)
```

A.4 Quantum Gerchberg-Saxton Algorithm

The code for the quantum-GS algorithm.

```
from qiskit.visualization import circuit_drawer
from qiskit.visualization.state_visualization import _bloch_multivector_data
```

```
from time import sleep
import tqdm
import matplotlib.pyplot as plt
import matplotlib.cm as cm

# Hologram and reconstruction planes
num_qubits = 4
iterations = 100

previous_amplitudes = [0]*num_qubits
previous_phases = [0]*num_qubits

# Target intensities
target_intensities = [0.2]*num_qubits
target_intensities[num_qubits//2] = 1.0


# Initial intensities
initial_intensities = [1.0*np.pi]*num_qubits

# Initial phases
initial_phases = np.random.rand(num_qubits)

def gerchberg_saxton(iterations = 5):

    hologram = QuantumRegister(num_qubits, 'hologram')
    reconstruction = QuantumRegister(num_qubits, 'reconstruction')

    # Create classical registers for measurement results
    reconstruction_measure = ClassicalRegister(num_qubits, 'reconstruction_measure')
    holographic_measure = ClassicalRegister(num_qubits, 'holographic_measure')

    qc = QuantumCircuit(hologram, reconstruction, holographic_measure,
        ↪ reconstruction_measure)

    def initialize_amplitudes_and_phases(qc, qubits, intensities, phases):
        """Sets the initial intensities and phases and applies QFT to the qubits."""


        # Initialize the qubits with intensities and phases the first time
        for i, qubit in enumerate(qubits):
            amplitude = intensities[i]*np.pi # 0 if 0, pi if 1
            previous_amplitudes[i] += amplitude
            phase = phases[i]
            previous_amplitudes[i] += phase

            qc.ry(amplitude, qubit)
            qc.rz(phase, qubit)

        # interference pattern # replaced by QFT
        #for qubit in qubits:
```

```
        # qc.h(qubit)


        return qc


    def set_intensities(qc, qubits, target_intensities):
        """Sets the intensities of the qubits."""

        for i, qubit in enumerate(qubits):
            amplitude = target_intensities[i]*np.pi # 0 if 0, pi if 1
            qc.ry(amplitude-previous_amplitudes[i], qubit)

        return qc

    def propagate(qc, from_qubits, to_qubits, h2r = True):
        """Propagates information from one set of qubits to another by applying phase
            ↪ shifts."""

        #QFT
        qft = QFT(num_qubits=num_qubits).to_gate()
        if(h2r):
            qc.append(qft, qargs=range(num_qubits))
        else:
            qc.append(qft, qargs=range(num_qubits,2*num_qubits))


        for i, qubit in enumerate(from_qubits):
            phase = -np.pi/16 if h2r else np.pi/16
            qc.rz(phase, i if h2r else num_qubits + i)

        #QFT
        qft = QFT(num_qubits=num_qubits, inverse= True).to_gate()
        if(h2r):
            qc.append(qft, qargs=range(num_qubits))
        else:
            qc.append(qft, qargs=range(num_qubits,2*num_qubits))

        for i in range(num_qubits):
            qc.swap(from_qubits[i], to_qubits[i])

        return qc

    print("Creating Circuit")
    pbar = tqdm.tqdm(total=iterations, desc='iterations', leave=False)

    # Initialize hologram plane
    initialize_amplitudes_and_phases(qc, hologram, initial_intensities,
        ↪ target_intensities)

    for i in range(iterations):
      pbar.update(1)
```

```
    # Propagate to reconstruction plane
    propagate(qc, hologram, reconstruction, True)

    # Set target intensities
    set_intensities(qc, reconstruction, target_intensities)

    if(i == iterations-1 ): break

    # Propagate to hologram plane
    propagate(qc, reconstruction, hologram, False)


    # Set target intensities
    set_intensities(qc, hologram, initial_intensities)

  pbar.close()

  # Simulate
  print("Simulating...")


  simulator = Aer.get_backend('statevector_simulator')
  job = execute(qc, simulator)
  result = job.result()
  statevector = result.get_statevector()
  bloch_data = (_bloch_multivector_data(statevector))


  amplitudes_h = []
  amplitudes_r = []
  phases_h = []
  phases_r = []
  for i in range(len(bloch_data)):
      q = bloch_data[i]
      theta = np.arccos(q[2])
      phi = np.arctan2(q[1],q[0])

      #hologram
      if(i < num_qubits):
          amplitudes_h.append(theta)
          phases_h.append(phi)
      else:
      #reconstruction
          amplitudes_r.append(theta)
          phases_r.append(phi)

  return amplitudes_h, phases_h, amplitudes_r, phases_r , qc
```

A.5 Quantum Gerchberg-Saxton Visualization

Here we provide the code for visualizing the algorithm results.

```
def visualize_states(amplitudes_h, phases_h, amplitudes_r, phases_r, target_intensities,
    ↪  count):
    def normalize(vector):
        magnitude = np.linalg.norm(vector)

        # Step 2: Normalize the vector by dividing each component by the magnitude
        return [component / (magnitude) for component in vector]
    print(amplitudes_r)
    plt.figure(figsize=(4, 3))
    plt.bar(range(num_qubits), normalize(amplitudes_r), label='Selected Amplitudes',
        ↪ alpha=0.7, color='b', width=0.4)
    plt.bar(range(num_qubits), normalize(target_intensities), label='Target Intensities',
        ↪  alpha=0.7, color='r', width=0.2)

    # Add labels and a legend
    plt.xlabel('States')
    plt.ylabel('Intensity')
    plt.title('Comparison of Averaged Amplitudes and Target Intensities')
    plt.xticks(rotation=90)
    plt.tight_layout()
    #plt.show()
    file_name = f"amplitudes_chart_{str(count).zfill(5)}.png"
    plt.savefig(file_name)
    plt.clf()

    plt.bar(range(num_qubits), phases_r) # last propagation put h into r
    plt.xlabel('qubit')
    plt.ylabel('Values')
    plt.title('Qubit Phases')
    plt.xticks(rotation=90)
    plt.tight_layout()
    #plt.show()
    file_name = f"phases_chart_{str(count).zfill(5)}.png"
    plt.savefig(file_name)
    plt.clf()

for i in range(1,iterations+1, 1):
    print("---------------------------- Iterations: ", i, "
        ↪ ------------------------------------")
    a_h, p_h, a_r, p_r, qc = gerchberg_saxton(i)
    visualize_states(a_h, p_h, a_r, p_r, target_intensities, i)
```

A.6 Stochastic Gradient Descent for Holography

Odak version for SGD for holographic reconstruction.

```
import torch
from odak.learn.wave import calculate_phase, calculate_amplitude, wavenumber,
    ↪ propagate_beam, produce_phase_only_slm_pattern, set_amplitude,
    ↪ generate_complex_field, linear_grating
from odak.learn.tools import zero_pad, crop_center, save_image,load_image
from odak import np
```

```
from tqdm import tqdm

wavelength = 515e-9
k = wavenumber(wavelength)

dx = 0.000008
resolution = [1080,1920]
distance = 0.15
square_size = 100
target = torch.zeros(resolution[0],resolution[1])
target[round((resolution[0]-square_size)/2):round((resolution[0]+square_size)/2),round((
    ↪ resolution[1]-square_size)/2):round((resolution[1]+square_size)/2)] = 1.

import matplotlib.pyplot as plt
import matplotlib.cm as cm
plt.imshow(target, cmap=cm.inferno)
plt.colorbar()
plt.show()

propagation_type = 'TR␣Fresnel'
slm_range = 2*np.pi
dynamic_range = 255

iteration_number = 100
t = tqdm(range(iteration_number), leave=False)

phase = torch.rand(resolution[0], resolution[1])
amplitude = torch.ones_like(target)
hologram = generate_complex_field(amplitude,phase)

loss_function = torch.nn.MSELoss(reduction='none')
alpha = 0.1

for i in t:
    hologram_padded = zero_pad(hologram)
    recon_field = propagate_beam(hologram_padded, k, distance, dx, wavelength,
        ↪ propagation_type)
    recon_field_cropped = crop_center(recon_field)
    recon_intensity = calculate_amplitude(recon_field_cropped)**2
    loss = loss_function(recon_intensity, target)
    loss_field = generate_complex_field(loss, calculate_phase(recon_field_cropped))
    loss_field_padded = zero_pad(loss_field)
    loss_propagated_padded = propagate_beam(loss_field_padded, k, -distance, dx,
        ↪ wavelength, propagation_type)
    loss_propagated = crop_center(loss_propagated_padded)
    hologram_updated = hologram - alpha * loss_propagated
    hologram_phase = calculate_phase(hologram_updated)
    hologram = generate_complex_field(amplitude, hologram_phase)
    t.set_description('Loss:{:.4f}'.format(torch.mean(loss)))

hologram_padded = zero_pad(hologram)
reconstruction = propagate_beam(hologram_padded, k, distance, dx, wavelength,
    ↪ propagation_type)
```

```
reconstruction = crop_center(reconstruction)

reconstruction_amp = calculate_amplitude(reconstruction)
reconstruction_intensity = (reconstruction_amp / reconstruction_amp.max())**2
save_image('reconstructed_image.png',reconstruction_intensity,cmin=0.,cmax=1.)

plt.imshow(reconstruction_intensity, cmap=cm.inferno)
plt.colorbar()
plt.show()

phase_hologram = calculate_phase(hologram)
phase_only_hologram = (phase_hologram % slm_range) / slm_range * dynamic_range
save_image('phase_only_hologram.png', phase_only_hologram)

plt.imshow(phase_only_hologram, cmap=cm.inferno)
plt.colorbar()
plt.show()
```

A.7 Quantum Stochastic Gradient Descent for Holography

A possible approach of QuantumSGD using Qiskit:

```
import qiskit
from qiskit.circuit.library import PhasedAmpQubit
from qiskit.circuit.quantumregister import QuantumRegister

import torch
from odak.learn.wave import calculate_phase, calculate_amplitude, wavenumber,
    ↪ propagate_beam, produce_phase_only_slm_pattern, set_amplitude,
    ↪ generate_complex_field, linear_grating
from odak.learn.tools import zero_pad, crop_center, save_image,load_image
from odak import np
from tqdm import tqdm

wavelength = 515e-9
k = wavenumber(wavelength)

dx = 0.000008
resolution = [1080,1920]
distance = 0.15
square_size = 100
target = torch.zeros(resolution[0],resolution[1])
target[round((resolution[0]-square_size)/2):round((resolution[0]+square_size)/2),round((
    ↪ resolution[1]-square_size)/2):round((resolution[1]+square_size)/2)] = 1.

import matplotlib.pyplot as plt
import matplotlib.cm as cm
plt.imshow(target, cmap=cm.inferno)
plt.colorbar()
plt.show()

propagation_type = 'TR Fresnel'
```

```
slm_range = 2*np.pi
dynamic_range = 255

iteration_number = 100
t = tqdm(range(iteration_number), leave=False)

qreg = QuantumRegister(resolution[0] * resolution[1])
circuit = qiskit.QuantumCircuit(qreg)

loss_function = torch.nn.MSELoss(reduction='none')
alpha = 0.1

for i in t:
    # propagate the field
    circuit.append(PhasedAmpQubit(k, distance, dx, wavelength, propagation_type), qreg)

    # set the amplitude
    circuit.set_amplitude(target, qreg)

    # propagate the field back
    circuit.append(PhasedAmpQubit(k, -distance, dx, wavelength, propagation_type), qreg)

    # measure the field
    counts = qiskit.execute(circuit, backend=qiskit.Aer.get_backend('qasm_simulator')).
        ↪ result().get_counts()

    # reconstruct the image
    reconstruction = []
    for i in range(resolution[0] * resolution[1]):
        amplitude = counts.get(str(i), 0) / (2**resolution[0] * resolution[1])
        phase = i * slm_range / dynamic_range
        reconstruction.append(complex(amplitude, phase))

    loss = loss_function(reconstruction, target)
    loss_field = generate_complex_field(loss, calculate_phase(reconstruction))

    # update the field
    for i in range(resolution[0] * resolution[1]):
        circuit.cx(i, i + 1)
        circuit.rz(alpha * loss_field[i], i)
        circuit.cx(i, i + 1)
```