Doctoral Programme in Bioengineering

Study of triplicator diffraction gratings and holograms displayed onto spatial light modulators

Shang Gao

Director of the thesis

Dra. María del Mar Sánchez López

Co-director of the thesis

Dr. Ignacio Moreno Soriano

Universidad Miguel Hernández de Elche

- 2025 –

This Doctoral Thesis, entitled "**Study of triplicator diffraction gratings and holograms displayed onto spatial light modulators**", is submitted under the format of **conventional thesis with the following quality indicators**:

- J. A. Davis, I. Moreno, **S. Gao**, M. M. Sánchez-López, D. M. Cottrell, "Multiplexing onto a spatial light modulator using random binary patterns", *Optical Engineering* 62(10), 103104 (2023). https://doi.org/10.1117/1.OE.62.10.103104 [JCR 1.1 (2023) in "Optics", journal 93/119, Q4].

- **S. Gao**, M. M. Sánchez-López and I. Moreno, "Experimental implementation of phase triplicator gratings in a spatial light modulator", *Chinese Optics Letters* 22(2), 020501 (2024). https://doi.org/10.3788/COL202422.020501 [JCR 3.3 (2023) in "Optics", journal 33/119, Q2].

- **S. Gao**, M. M. Sánchez-López and I. Moreno, "Analysis of diffraction gratings displayed in spatial light modulators at the Nyquist limit: Application to the triplicator grating", *Optics & Lasers in Engineering* 184, 108628 (2025). https://doi.org/10.1016/j.optlaseng.2024.108628 [JCR 3.5 (2023) in "Optics", journal 32/119, Q2].

- **S. Gao**, M. M. Sánchez-López, P. García-Martínez and I. Moreno, "Triplicator phase-only hologram and its use as a snapshot optical convolver and correlator", Journal of the European Optical Society-Rapid Publications, (2025) (Accepted). https://doi.org/10.1051/jeos/2025027 [JCR 1.9 (2023) in "Optics", journal 55/119, Q3].

Dra. María del Mar Sánchez López, director, and Dr. Ignacio Moreno Soriano, co-director of the doctoral thesis entitled **Study of triplicator diffraction gratings and holograms displayed onto spatial light modulators**

REPORT:

That Shang Gao has performed, under our supervision, the work entitled "**Study of triplicator diffraction gratings and holograms displayed onto spatial light modulators**" pursuant to the terms and conditions established in the Research Plan and following the Code of Good Practices of the Miguel Hernández University of Elche, successfully meeting the objectives planned for its public defence as a doctoral thesis.

In witness whereof we sign for all pertinent purposes, in Elche on April 08th 2025

Thesis director
Dr. María del Mar Sánchez López

Thesis co-director
Dr. Ignacio Moreno Soriano

Dra. Piedad de Aza Moya, Coordinator of the Doctoral Programme in Bioengineerng

REPORTS:

That Shang Gao has performed, under the supervision of our Doctoral Programme, the work entitled "**Study of triplicator diffraction gratings and holograms displayed onto spatial light modulators**" pursuant to the terms and conditions established in the Research Plan and following the Code of Good Practices of the Miguel Hernández University of Elche, successfully meeting the objectives planned for its public defence as a doctoral thesis.

In witness whereof I sign for all pertinent purposes, in Elche on April 08th 2025.

Prof. Dra. Piedad de Aza Moya

Coordinator of the Doctoral Programme in Bioengineering

Acknowledgements

With profound gratitude, I extend my sincerest appreciation to my esteemed directors, Dr. María del Mar Sánchez and Dr. Ignacio Moreno, for their invaluable guidance through my doctoral research. Their profound academic expertise and excellent insights laid a solid base for the completion of my PhD thesis. Their knowledge and wisdom have not only guided the direction of my doctoral studies with their patience and unwavering mentorship but have also let me think independently. Their selfless dedication to guidance has equipped me with a systematic understanding of optics and optoelectronics and has also made me understand how to break down complex scientific challenges into solvable steps, which will remain an enduring inspiration throughout my career. I am especially grateful for their kindness and care in daily life, which made me feel a deep sense of belonging while studying far from home.

I would also like to express my appreciation to Dr. Jeffrey Davis for the invaluable opportunity to share a publication with him. Additionally, I extend my gratitude to Dr. Pascuala García. Also, I am grateful to work together with her, and every visit of hers provided me with new insights and perspectives.

My sincere thanks also go to my colleagues, Dr. David Marco and Dr. Esther Nabadda. Though they completed postdoctoral work and successfully obtained doctoral degrees, respectively, ahead of me, I will always treasure the time that we were together.

I am eternally grateful to my mother, Ms. Yunli Shang, whose kindness and strength have been my greatest source of support. As a mother and a role model, she has instilled in me the values of perseverance and integrity, shaping me into someone who embraces challenges with courage and determination. She has taught me to live by the principles of "uprightness and sincerity "

I am also deeply appreciative of all those who have guided and supported me throughout my life—my professors during my master's studies in Photonics Engineering at Universidad Carlos III de Madrid, my early mentors at Yanbian University's Faculty of Physics, and my friends who have companied me with unwavering encouragement and emotional support.

Finally, to you, the reader of this thesis, I extend my sincerest gratitude for your reading.

Financing

The work presented in this Thesis has been possible thanks to the financial support received from Generalitat Valenciana (project ref. CIAICO/2021/276) and Ministerio de Ciencia e Innovación, Spain (project ref.: PID2021-126509OB-C22).

Table of Contents

Summary

This thesis comprises the design of phase-only triplicator diffraction gratings and computer-generated holograms (CGHs) and their encoding on phase-only liquid-crystal spatial light modulators (SLMs). Polychromatic CGHs are also developed, where the colour reconstructions are successfully realized by implementing chromatic dispersion control in the hologram design and synchronizing a high-speed SLM with and RGB laser.

SLMs are programmable optoelectronic liquid-crystal microdisplays that allow the shaping of light beams, in amplitude, phase and polarization, with high spatial resolution and in real time, thus opening up a wide range of possibilities for bioengineering applications. Here we focus on displaying phase-only functions, since they have ideally 100% diffraction efficiency. However, the performance of SLMs is affected by their pixelated structure and other deletory effects, like phase flicker, refresh rate and light efficiency, that must be taken into account. Therefore, this thesis addresses both the design and characteristics of phase-only diffractive optical elements and the principal aspects that affect the ideal SLM performance, which must be properly evaluated and compensated.

Triplicator gratings are 1x3 fan-out elements that generate three equally intense 0th and $\pm$1st diffraction orders. The optimum triplicator phase design has received renewed attention for the implementation of trifocal diffractive intraocular lenses. Also, combined with a spiral phase function, it has served to generate arrays of vortex beams. In this thesis we extend the triplicator phase profile to triplicator holograms, where we are able to obtain the target image on the triplicator orders: direct and inverted image on the $\pm$1st orders, and a delta function on the zero-order. In this way, we can realize an optical convolver and correlator by adding the hologram of the target image to the triplicator holograms. This optical element keeps the high efficiency of the optimum phase triplicator, what is of interest given the essential tool of correlators to check the likeness between two light patterns.

This thesis is presented in the conventional mode of a doctoral thesis report and comprises two parts. The first part is devoted to the materials and methods employed: liquid-crystal (LC) on silicon SLMs and Fourier optics theory. When illuminated with linearly polarized light parallel to the LC director, LC-SLMs modulate the phase of input light by addressing them from a computer with a gray level pattern displayed on the device pixelated screen. The displayed pattern then diffracts the incoming light beam and reconstructs the desired optical wavefront. Three different phase-only SLMs are employed in this work; their characteristics and calibration are described in detail. An automatic calibration method based on a polarization camera is proposed and demonstrated on a monopixel LC retarder. The theoretical methods supporting this thesis rely on Fourier analysis, light diffraction theory and Fourier optics. Fourier analysis (including Fourier series and Fourier transform) is briefly reviewed together with the relevant properties that provide the tools for analysing diffraction gratings, calculating CGHs, and modelling the pixelated structure of SLMs. The optical architectures employed to obtain the optical Fourier transform are introduced using the classical simplest approach of Fourier optics, where light beams are considered within the paraxial approximation and a two-dimensional Fourier relation is obtained for the diffracted field with respect to the field behind the diffractive element.

The second part of the thesis presents the analytical and experimental results, together with the corresponding discussions. First, using the Fourier series analysis, different phase-only triplicator designs are compared, including binary, multilevel and continuous phase profiles. They are implemented on a LC-SLM considering large periods (of 64 pixels) to verify their properties. The condition for generating a triplicator with high efficiency are discussed. The binary π phase grating proves to be a feasible option, with a theoretical

efficiency not so far from Gori's optimum continuous profile, and also close to the efficiency of the multilevel phase profile that adapts well to the SLM pixelated structure. Interestingly, the large number of pixels available in the device enable us to apply a random multiplexing approach, which provides a triplicator with less efficiency but free of higher harmonic orders. However, the pixels feature a dead zone and a limited effective area, thus resulting in a periodical physical structure of the SLM. We model it as an amplitude grating of period equal to the pixel pitch. Using the Fourier transform approach we demonstrate that this SLM pixelated structure yields diffraction parasite orders with weights given by a sinc envelope. Consequently, this effect will be relevant when encoding on the SLM phase gratings with short periods, particularly at the spatial resolution limit (Nyquist limit), i.e. period of two times the pixel pitch, which must be binary phase profiles. The convolutional Fourier approach gives a clear physical insight into this phenomenon, where the target diffraction pattern is replicated on the SLM parasite orders. Therefore, there is a superposition between orders of the different replicas. Analytical expressions for the diffraction orders' intensities are obtained in terms of the device fill factor and phase level of the binary grating. The triplicator's diffraction efficiency is also derived and verified experimentally for binary gratings with different periods. As the period decreases down to two pixels (Nyquist limit), the fringing field effect cannot be ignored. Therefore, making use of the fact that the fringing effect smooths the phase profile, we build a nonlinear phase profile model that fits the experimental intensity curves and provides the actual distorted phase implemented on the SLM due to fringing.

On the other hand, a systematic study on how to improve the reconstruction quality of phase-only CGHs is done as well. We compute the Fourier transform of the original image and retrieve only the phase (kinoform CGH) to implement it on the SLM, where advantage is taken of fast Fourier transform (FFT) kits in Python. For simple CGH computation, we explore several methods to improve the hologram intensity reconstruction, such as adding amplitude/phase noise to the original field, applying a limit window to the target area, and applying a nonlinear calculation to the original image. All the above methods aim at increasing the weight of the high-frequency component and flattening the intensity distribution in the reconstructed target field with high signal-to-noise ratio. The inverse Fourier transform algorithm (IFTA) is applied. Polychromatic CGHs are presented, where realistic reconstructions can be obtained. Red (R), green (G), and blue (B) lasers are modulated independently and synchronised to a 180 Hz LC-SLM so that, by properly scaling the RGB phase masks, we can compensate for the chromatic dispersion and recover the original colour object.

Finally, the triplicator phase profile is applied to CGHs, leading to the reconstruction of the target image at the customized diffraction orders. This provides simultaneously an optical convolver and a correlator. This is useful because when two identical images (in all aspects) are correlated, a delta function will be shown in the reconstructed field. However, if the two images have any difference, like phase information that is invisible, the profile of the delta function will be broadened.

This doctoral thesis thus contributes to the advance in the proper use of phase-only liquid-crystal SLMs for the accurate implementation of triplicator phase diffraction gratings and computer-generated holograms. Given the growing interest in using such programmable devices in optical imaging techniques, this thesis provides useful tools for bioengineering, telecommunications, or industry applications.

Resumen

Esta tesis comprende el diseño de redes de difracción triplicadoras puras de fase y hologramas generados por ordenador, así como su realización experimental con moduladores espaciales de luz (conocidos como SLMs, iniciales de las palabras en inglés *Spatial Light Modulators*). También se han desarrollado hologramas policromáticos, en los que las reconstrucciones de color se consiguen implementando el control de la dispersión cromática en el diseño del holograma y sincronizando un SLM de alta velocidad con un láser RGB.

Los SLM son micropantallas optoelectrónicas de cristal líquido programables que permiten modelar haces de luz, en amplitud, fase y polarización, con alta resolución espacial y en tiempo real, abriendo así un amplio abanico de posibilidades para aplicaciones de bioingeniería. Aquí nos centramos en la visualización de funciones de puras de fase, ya que tienen una eficiencia de difracción idealmente del 100%. Sin embargo, el rendimiento de los SLM se ve afectado por su estructura pixelada y otros efectos secundarios, como el parpadeo de fase, la frecuencia de refresco y la eficiencia, que deben tenerse en cuenta. Por ello, esta tesis aborda tanto el diseño y las características de los elementos ópticos difractivos puros de fase, así como los principales aspectos que afectan y limitan a su implementación física en SLMs, que deben ser adecuadamente evaluados y compensados.

Las redes de difracción triplicadoras son elementos divisores de haz de 1×3 que generan tres órdenes de difracción (orden cero y órdenes ±1) con la misma intensidad. El diseño óptimo de la red de fase triplicadora ha recibido renovada reciente atención para la implementación de lentes intraoculares difractivas trifocales. Además, combinado con una función de fase en espiral, ha servido para generar matrices de haces vorticiales. En esta tesis extendemos el perfil de fase de la red triplicadora a los hologramas triplicadores, de modo que se obtienen tres órdenes de ifracción que contienen las imágenes directa e invertida en los órdenes ±1, y una función delta en el orden cero. De este modo, podemos realizar un elemento óptico convolucionador y correlador simplemente añadiendo también el holograma de la imagen objetivo al holograma triplicador. Este elemento mantiene la eficiencia de la red optima triplicadora, y tiene interés ya que el correlador óptico es una herramienta esencial para comprobar la semejanza entre dos patrones de luz.

Esta tesis se presenta en el modo convencional de una memoria de tesis doctoral y consta de dos partes. La primera parte está dedicada a los materiales y métodos empleados: se introducen los dispositivos SLM de cristal líquido sobre silicio, conocidos como LCOS (de las iniciales de las palabras en inglés *Liquid-Crystal On Silicon*) y la teoría de la óptica de Fourier. Cuando se iluminan con luz polarizada linealmente paralela al director del cristal líquido, losdispositivos LCOS-SLM modulan la distribución de fase de la luz de entrada de forma controlada desde un ordenador mediante un patrón de niveles de gris. A continuación, el patrón de luz generado en la pantalla LCOS se difracta y reconstruye en el plano de Fourier el frente de onda óptico deseado. En este trabajo se emplean tres SLM puros de fase diferentes; sus características y calibración se describen en detalle. Se propone un método de calibración automática basado en una cámara de polarización y se demuestra en un retardador de cristal líquido de un único píxel. Los métodos teóricos que sustentan esta tesis se basan en el análisis de Fourier, la teoría de difracción de la luz y la óptica de Fourier. El análisis de Fourier (incluidas las series de Fourier y la transformada de Fourier) se revisa brevemente junto con las propiedades relevantes que proporcionan las herramientas para analizar las redess de difracción, calcular los hologramas generados por ordenador y modelizar la

estructura pixelada de los SLM. Las arquitecturas ópticas empleadas para obtener la transformada de Fourier óptica se introducen utilizando el enfoque clásico de la óptica de Fourier, en el que los haces de luz se consideran dentro de la aproximación paraxial y se obtiene una relación de Fourier bidimensional para el campo difractado.

La segunda parte de la tesis presenta los resultados analíticos y experimentales, junto con las correspondientes discusiones. En primer lugar, utilizando el análisis de series de Fourier, se comparan diferentes diseños de redes de difracción triplicadoras puras de fase, incluyendo perfiles de fase binarios, multinivel y continuos. Se implementan en un LCOS-SLM considerando periodos grandes (de 64 píxeles) y se verifican sus propiedades. Se discuten las condiciones para generar un triplicador con alta eficiencia. La red de diffración binaria de fase π binaria se muestra como una opción factible, con una eficiencia teórica no tan alejada del perfil continuo óptimo de Gori, y también cercana a la eficiencia del perfil de fase multinivel que se adapta bien a la estructura pixelada de los SLM. El gran número de píxeles disponibles en el dispositivo nos permite aplicar un enfoque de multiplexación aleatoria, que proporciona un triplicador con menor eficiencia pero libre de órdenes armónicos superiores. Sin embargo, los píxeles presentan una zona muerta y un área efectiva limitada, lo que da lugar a la estructura física periódica del SLM. Ésta se modeliza como una red de difracción de amplitud de periodo igual a la distancia entre píxeles. Utilizando el enfoque de la transformada de Fourier demostramos que esta estructura pixelada del SLM produce órdenes de difracción parásitos con pesos dados por una envolvente en forma de función seno-cardinal (sinc). En consecuencia, este efecto es relevante al codificar en el SLM redes de fase con periodos cortos, particularmente en el límite de resolución espacial (límite de Nyquist), es decir, periodos de dos píxeles, que forzosamente deben ser perfiles binarios de fase. El enfoque convolucional ofrece una visión física clara dele fenómeno, en el que el patrón de difracción del objetivo se replica en los órdenes parásitos de la SLM. Por lo tanto, en el caso de las redes de difracción, existe una superposición entre los órdenes de las distintas réplicas. Se obtienen expresiones analíticas para las intensidades de los órdenes de difracción en función del factor de llenado del dispositivo y del nivel de fase de la red binaria. También se deduce la eficiencia de difracción del triplicador y se verifica experimentalmente para redes binarias con diferentes periodos. A medida que el periodo disminuye hasta alcanzar los dos píxeles (límite de Nyquist), no puede ignorarse el efecto conocido como *fringing* que produce una distorsión de la función de fase implementada por el dispositivo, que suaviza el perfil de fase. Este efecto se evalúa mediante un modelo de perfil de fase no lineal que se ajusta a las curvas de intensidad experimentales y proporciona la fase distorsionada real implementada en el SLM.

Por otro lado, también se realiza un estudio sistemático sobre cómo mejorar la calidad de reconstrucción de los hologramas generados por ordenador puros de fase. Calculamos la transformada de Fourier de la imagen original aprovechando las funciones de transfomada de Fourier rápida (FFT del inglés, *Fast Fourier Transform*) existentes en código Pyhton, y recuperamos solamente la distribución de fase (holograma tipo kinoform), que se implementarla en el SLM. En el cálculo simple de los hologramas de fase, exploramos varios métodos para mejorar la reconstrucción de la intensidad, como añadir ruido de amplitud o fase a la función original, aplicar una ventana límite a la zona objetivo de reconstrucción o aplicar un cálculo no lineal a la imagen original. Todos los métodos anteriores tienen como objetivo aumentar el peso del componente de alta frecuencia y aplanar la distribución de intensidad en el campo reconstruido, con una elevada relación señal/ruido. Se aplica el algoritmo de transformada de Fourier inversa (IFTA, del inglés *inverse Fourier transform algorithm*). Se presentan también hologramas policromáticos, con los que obtener reconstrucciones realistas. Se utiliza un sistema combinado de tres láseres rojo (R), verde (G) y azul (B) que se modulan independientemente y se sincronizan con un LCOS-SLM de 180

Hz, de modo que, escalando adecuadamente las funciones de fase RGB, podemos compensar la dispersión cromática y recuperar el objeto en color original.

Por último, el perfil de fase de la red triplicadora se aplica a los hologramas en color, lo que conduce a la reconstrucción de la imagen objetivo en los órdenes de difracción personalizados. Esto proporciona simultáneamente un elemento óptico convolucionador y correlador en color. Esto es útil porque cuando dos imágenes idénticas (en todos los aspectos) están correlacionadas, se mostrará una función delta en el campo reconstruido. Sin embargo, si las dos imágenes tienen alguna diferencia, como información de fase que es invisible, el perfil de la función delta se ensanchará.

Esta tesis doctoral contribuye así al avance en el uso adecuado de SLMs de cristal líquido de sólo fase para la implementación precisa de rejillas de difracción de fase triplicadora y hologramas generados por ordenador. Dado el creciente interés en el uso de este tipo de dispositivos programables en técnicas de imagen óptica, esta tesis proporciona herramientas útiles para múltiples aplicaciones de bioingeniería, telecomunicaciones o industriales.

1. Introduction and objectives

This Doctoral Thesis report constitutes a compilation of the work developed by Shang Gao within the framework of the Doctoral Program in Bioengineering of the University Miguel Hernández of Elche (UMH). The work has been developed at the TecnOPTO Labs (https://tecnopto.umh.es/) of the UMH Institute of Bioengineering (https://bioingenieria.umh.es/).

The Thesis is framed within the field of Optics, specifically in the design of diffraction gratings and computer-generated holograms, and their physical realization with optoelectronic displays, namely phase-only liquid-crystal spatial light modulators.

In this first chapter of the report, the work carried out is introduced and the objectives that have been sought are described, as well as the materials and instruments used along the work

1.1. Motivation

Optical technologies are well-established resources recognized as key enablers for current societal trends, including manufacturing, telecommunications, digitalization, IoT, big data, artificial

intelligence or autonomous transportation [Pho-2019]. This fundamental role of Optics and Photonics is particularly relevant in biomedicine [Tuc-2010, Ngu-2022], serving as mature technologies impacting different applications that cover from diagnostic imaging devices, non-invasive procedures, visualization assisting tools in therapeutic protocols, surgical automatic devices, augmented vision systems, biosensors, biomarkers, etc.

Current modern optical techniques very often rely on the use of spatial light modulators (SLM), optoelectronic microdisplays that can manipulate light with high spatial and/or temporal resolution, opening up a wide range of possibilities for bioengineering research and applications. Among them, we can highlight their use in the following fields:

- in optical tweezers, where they are employed to create complex and dynamic optical traps [Pes-2020];
- in super-resolution microscopy, where they help improve resolution in techniques like Structured Illumination Microscopy (SIM) [Wen-2022] or Stimulated Emission Depletion (STED) microscopy [Nobel-2014, Gör-2018];
- in adaptive optics, where they are used to correct for aberrations to better visualize biological samples [Muñ-2025] and correct human vision [Soo-2024];
- in optical coherence tomography (OCT), where they provide the means to control the wavefront of light [Urr-2024];
- in optogenetics, where they are used to activate or deactivate with light specific neurons or other cells, enabling targeted photo stimulation [Nik-2008] and providing insights into neural circuits and biological processes [Lee-2024];
- in laser-based surgery, where they are used to shape laser beams for precise tissue ablation and control the focused spot at the end of multimode optical fibre to improve minimally invasive endoscopy [Lam-2024];
- in microfluidics, where they can be used to create dynamic optical patterns within microfluidic devices, enabling precise control of fluid flow and particle manipulation [Col-2019].

Thanks to their high spatial resolution, modern SLMs play a crucial role in all these applications by displaying a diffractive element, i.e., a computer-generated hologram (CGH), which implements the different required optical function. SLMs can manipulate the amplitude, the phase, or the polarization of light, and this is done spatially by addressing them from a computer with a gray level pattern that is displayed on their pixelated surfaces, which then diffract the incoming light beam to reconstruct the desired optical wavefront. In most cases the CGHs displayed on SLMs

are designed to encode phase-only functions, since they have in principle 100% light diffraction efficiency. However, this imposes restrictions when the required optical functions include complex amplitude (magnitude and phase) or polarization information. Other aspects that affect the SLM performance include:

- Resolution and pixel pitch: The finite pixel size of SLMs limits the spatial resolution. Smaller pixel pitches are desired for higher resolution. However, reducing the pixel pitch exacerbates pixel crosstalk between closely spaced pixels, consequently, achieving accurate phase modulation becomes more challenging.
- Refresh rate: The speed at which SLMs can update their encoded holograms affects the stability of diffraction patterns and restricts the frame rate of dynamic holograms, which is important for video applications.
- Phase modulation accuracy: Imperfections in phase modulation can introduce distortions and noise in the reconstructed holograms. These imperfections may arise from aberrations in the SLM panel, and from limitations and artifacts generated by the electronic addressing required to produce the optical modulation.
- Light efficiency: Inevitably, not all the incident light is diffracted into the desired reconstruction, leading to energy losses.

1.2. Objectives of the Thesis

Having all these points in mind, the main goals pursued in this work are:

- Apply methods of Fourier analysis to describe the diffraction pattern generated in the far field by diffraction gratings implemented on phase-only SLMs, and study different phase-only profiles.
- Experimentally test the realization of specific diffraction gratings, namely triplicator gratings, that generate three equally intense diffraction orders.
- Apply methods for the characterization of the retardance and phase-modulation to high-resolution liquid-crystal SLMs.
- Analyse the implications that the pixelated structure of SLMs causes on the reconstruction of diffraction gratings with periods approaching the resolution limit. Analyse the diffraction generated by Nyquist gratings, diffraction gratings with two pixels per period.

- Implement a multiwavelength SLM-based time-multiplexing method for generating colour diffractive elements.
- Study methods for the multiplexing of different optical functions onto a single hologram.
- Apply methods of computer-generated holography to calculate iterative Fourier transform phase-only functions that efficiently and accurately reconstruct a given pattern and apply them to display dispersion-compensated colour structured patterns.
- Apply the triplicator grating profile to computer-generated holograms to achieve convolution and correlation operations between RGB signals.

1.3. Thesis organization

This Thesis report is organised as follows: after this brief introductory chapter, Chapter 2 and Chapter 3 review respectively the main concepts of Fourier analysis and Fourier optics theory, as well as the related systems to achieve optical Fourier transforms through diffraction. In Chapter 4 phase-only parallel-aligned liquid-crystal SLMs are explained, including the description of the polarization configuration, the methods used for calibrating the phase modulation, and the description of the specific devices used along the Thesis.

Then, Chapter 5 introduces different designs of phase-only diffraction gratings, with particular emphasis on the triplicator gratings, which generate three equally-intense diffraction orders. Chapter 6 introduces different methods to calculate phase-only computer-generated holograms, including those based on adding random functions to the original pattern and iterative algorithms for improving their reconstruction.

Chapter 7 describes the general case when an arbitrary CGH is displayed, and the effect of the pixelated structure of SLMs is studied in detail. Then, the realization of diffraction gratings on pixelated devices is studied in Chapter 8, where the double periodicity, that of the grating and that of the SLM, must be considered. Chapter 9 presents the realization of colour diffractive optical elements and holograms. Finally, Chapter 10 contains the conclusions of the work.

Some chapters include the Python code of the programs developed along the work, and every chapter contains its own list of references cited within the chapter.

2. Fourier analysis

This chapter briefly reviews the main concepts of Fourier analysis required to study diffraction gratings. We consider for simplicity one dimensional functions in most of the chapter and describe the Fourier series and the Fourier transform. Important concepts as convolution, cross correlation, and sampling, and how they are related with the Fourier transform, are here introduced. These tools will be applied in later chapters. Finally, the last section in the chapter extends the Fourier analysis to two-dimensional signals, as typically required in Optics [Arf-1995, Goo-2005].

2.1. Fourier series of a periodic function

2.1.1. Real form of the Fourier series

The Fourier theorem is central in signal analysis. It states that any periodic function $g(t)$ with period p can be described as the sum of sinusoidal functions as:

$$g(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos(\omega_n t) + \sum_{n=1}^{\infty} b_n \sin(\omega_n t), \tag{2.1}$$

where $\omega_1 = 2\pi f_1 = 2\pi/p$ is the fundamental angular frequency ($f_1 = 1/p$ is the fundamental linear frequency). The harmonic angular frequencies are multiples of this fundamental frequency $\omega_n = n\omega_1$ (or equivalently the harmonic linear frequencies are $f_n = nf_1$.) The Fourier series coefficients in Eq. (2.1) are given by:

$$a_n = \frac{2}{p} \int_{-p/2}^{+p/2} g(t) \cos(n\omega_1 t) dt. \tag{2.2a}$$

$$b_n = \frac{2}{p} \int_{-p/2}^{+p/2} g(t) \sin(n\omega_1 t) dt. \tag{2.2b}$$

The coefficient a_0 defines the DC term, thus $a_0/2$ is the average or continuum value:

$$a_0 = \frac{2}{p} \int_{-p/2}^{+p/2} g(t) dt. \tag{2.3}$$

2.1.2. Complex form of the Fourier series

The complex form of the Fourier series is an equivalent decomposition in the form:

$$g(t) = \sum_{n=-\infty}^{\infty} c_n e^{i\omega_n t}, \tag{2.4}$$

where the complex Fourier coefficients are:

$$c_n = \frac{1}{p} \int_{-p/2}^{+p/2} g(t) e^{-in\omega_1 t} dt, \tag{2.5}$$

which are related to the real coefficients by

$$c_0 = \frac{a_0}{2}, \quad c_{n>0} = \frac{1}{2}(a_n - ib_n), \quad c_{n<0} = \frac{1}{2}(a_n + ib_n). \tag{2.6}$$

2.1.3. Fourier series of a train of pulses

One classical signal widely studied in Fourier analysis is the train of pulses as shown in Fig. 2.1. It is included here as a relevant example but also because it will be employed along this Thesis, especially when analysing binary diffraction gratings and the SLM pixelated structure.

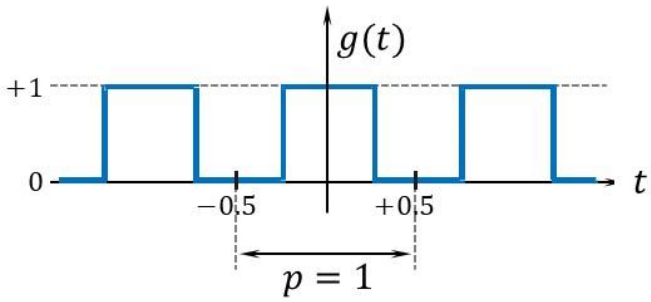

Fig. 2. 1. Signal $g(t)$ consisting of a train of pulses.

The function in Fig. 2.1 has period $p = 1$ (in arbitrary units), thus having a frequency $f_1 = 1$ (in arbitrary units, inverse of the period's units) and angular frequency $\omega_1 = 2\pi$. Within the central period, i.e. within the range $t \in \left[-\frac{p}{2}, +\frac{p}{2}\right]$, the function is defined as:

$$g_0(t) = \begin{cases} 1 \text{ if } t \in \left[-\frac{p}{4}, +\frac{p}{4}\right], \\ 0 \text{ if } t \in \left[-\frac{p}{2}, -\frac{p}{4}\right] \text{ or } t \in \left[+\frac{p}{4}, +\frac{p}{2}\right], \end{cases} \tag{2.7}$$

where $g_0(t)$ denotes $g(t)$ defined only in the central period interval.

Using this definition in Eqs. (2.2) and (2.3), the coefficients of the real form of the Fourier series expansion take the form:

$$a_n = 2\int_{-1/4}^{+1/4} \cos(n2\pi t)dt = \frac{2}{n\pi}\sin\left(\frac{n\pi}{2}\right) = \text{sinc}\left(\frac{n}{2}\right), \tag{2.8a}$$

$$b_n = 2\int_{-1/4}^{+1/4} \cos(n2\pi t)dt = 0, \tag{2.8b}$$

where the sine cardinal function (or sinc function) is defined as

$$\text{sinc}(x) \equiv \frac{\sin(\pi x)}{\pi x}. \tag{2.9}$$

Equations (2.8) show that the train of pulses in Fig. 2.1 has a DC component $a_0/2 = 1/2$

plus cosine terms with amplitudes a_n. Therefore, applying Eq. (2.1), its Fourier real expansion is given by

$$g(t) = \sum_{n=0}^{\infty} \text{sinc}\left(\frac{n}{2}\right)\cos(\omega_n t) = \frac{1}{2} + \frac{2}{\pi}\cos(2\pi t) - \frac{2}{3\pi}\cos(6\pi t) + \frac{2}{5\pi}\cos(10\pi t) - \cdots \quad (2.10)$$

Figure 2.2(a) illustrates the Fourier synthesis of the train of pulses as higher harmonic terms are added to the main frequency term.

For this train of pulses signal, the complex form of the Fourier expansion takes the following expression for the complex coefficients:

$$c_n = \frac{\sin\left(\frac{n\pi}{2}\right)}{n\pi} = \frac{1}{2}\text{sinc}\left(\frac{n}{2}\right). \quad (2.11)$$

Therefore, the same signal can be expressed as the following complex Fourier series:

$$g(t) = \frac{1}{2}\sum_{n=-\infty}^{\infty} \text{sinc}\left(\frac{n}{2}\right)e^{in2\pi t} = \frac{1}{2} + \frac{1}{\pi}e^{i2\pi t} - \frac{1}{3\pi}e^{i6\pi t} + \frac{1}{5\pi}e^{i10\pi t} - \cdots + \frac{1}{\pi}e^{-i2\pi t} - \frac{1}{3\pi}e^{-i6\pi t} + \frac{1}{5\pi}e^{-i10\pi t} - \cdots \quad (2.12)$$

Each exponential term in this complex Fourier expansion in Eq. (2.12) is associated with one angular frequency $\omega_n = n2\pi$, with $n = \pm 1, \pm 3, \pm 5$ .., each weighted by its corresponding coefficient c_n given in Eq. (2.11).

The signal representation in the frequency domain is its spectrum. It represents the Fourier coefficients in terms of the frequency content. In the case of the train of pulses, its spectrum is represented in Fig. 2.2(b). Because the train of pulses is a periodic function, its spectrum only contains discrete frequency values, the fundamental frequency ω_1 and its harmonics $\omega_n = n\omega_1$. This can be described using the delta (or impulse) function $\delta(t)$, which will be properly defined later, but which we can for the moment consider as a function that is zero everywhere except at the origin $t = 0$. Therefore, using delta functions $\delta(\omega - n\omega_1)$, each weighted by the $c_n = \frac{1}{2}\text{sinc}\left(\frac{n}{2}\right)$ term, the discrete Fourier spectrum can be described as the function:

$$G(\omega) = \frac{1}{2}\sum_{n=-\infty}^{\infty} \text{sinc}\left(\frac{n}{2}\right)\delta(\omega - n\omega_1). \quad (2.13)$$

Note that the delta functions are null everywhere except at $\omega = n\omega_1$, so $G(\omega)$ can be rewritten as

$$G(\omega) = \frac{1}{2}\,\mathrm{sinc}\left(\frac{\omega}{4\pi}\right) \sum_{n=-\infty}^{\infty} \delta(\omega - n\omega_1). \tag{2.14}$$

This relation shows that the spectrum of the train of pulses is a frequency comb modulated by the **sinc** function (represented as the dotted curve in Fig. 2.2(b)). We will refer to this characteristic when modelling the SLM pixelated structure.

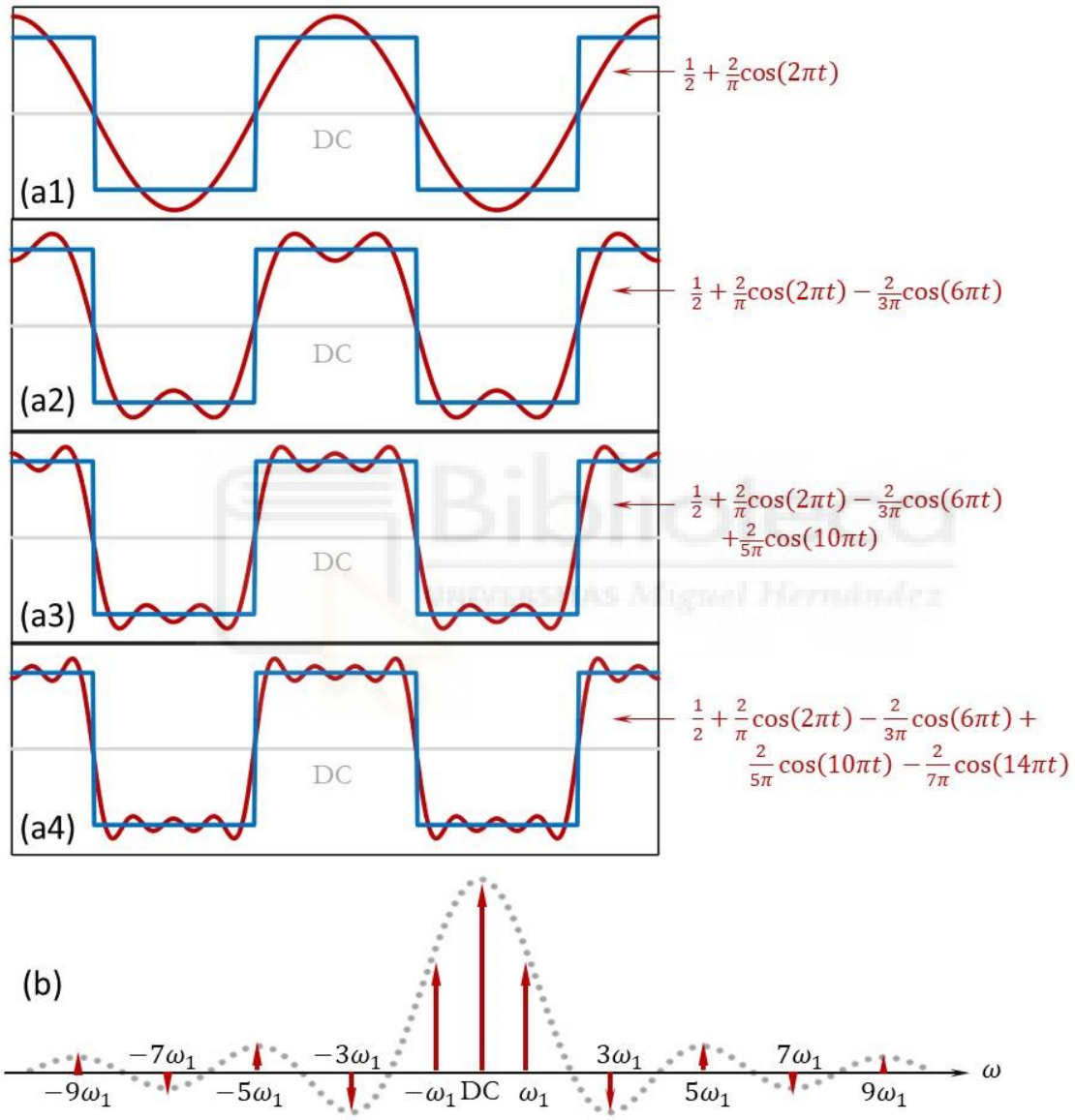

Fig. 2. 2. (a) Synthesis of the train of pulse signal as more harmonic terms are added to the first harmonic component. (b) Fourier frequency content of the train of the pulse signal.

2.2. The Fourier transform

The Fourier transform is the mathematical transform that decomposes an arbitrary function (not necessarily periodic) into its constituent frequencies. Given a function $g(t)$, its Fourier transform is defined as the function $G(\omega)$ given by the integral:

$$G(\omega) = \int_{-\infty}^{+\infty} g(t)e^{-i\omega t}dt, \tag{2.15}$$

The function $G(\omega)$ is the frequency spectrum of $g(t)$. In the case of periodic functions, it reproduces the discrete Fourier spectrum given by the Fourier series. However, as mentioned above, the Fourier transform can be applied to non-periodic functions, cases for which $G(\omega)$ is in general a continuous function of ω. The signal $g(t)$ can be recovered from its Fourier spectrum through the inverse Fourier transform as:

$$g(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} G(\omega)e^{+i\omega t}d\omega. \tag{2.16}$$

The Fourier transform operator $\mathcal{F}[\cdot]$ defined in terms of the linear frequency is:

$$G(f) = \mathcal{F}[g(t)] = \int_{-\infty}^{+\infty} g(t)e^{-i2\pi ft}dt, \tag{2.17}$$

while the inverse Fourier transform $\mathcal{F}^{-1}[\cdot]$ is given by

$$g(t) = \mathcal{F}^{-1}[G(f)] = \int_{-\infty}^{+\infty} G(f)e^{+i2\pi ft}df. \tag{2.18}$$

Equation (2.18) can be viewed as the synthesis of $g(t)$ by combining infinite exponential terms $e^{+i2\pi ft}$, which depends on the frequency f, weighted by the Fourier transform $G(f)$. This is equivalent to the Fourier series expansion in Eq. (2.4) for a periodic function, but the role of the discrete Fourier coefficients c_n is now taken by the continuous function $G(f)$.

2.2.1. Definition of functions and some Fourier transform pairs

Figure 2.3 shows some important Fourier transform pairs. An example particularly important is the impulse Dirac delta function already introduced in the previous section. This is a generalized function (distribution) used to model mathematically an idealized point, and it can be defined as

the Fourier transform of a constant function $g(t) = 1$, i.e.:

$$\delta(f) = \int_{-\infty}^{+\infty} e^{-i2\pi ft}dt = \begin{cases} \infty & \text{if } f = 0, \\ 0 & \text{if } f \neq 0. \end{cases} \tag{2.19}$$

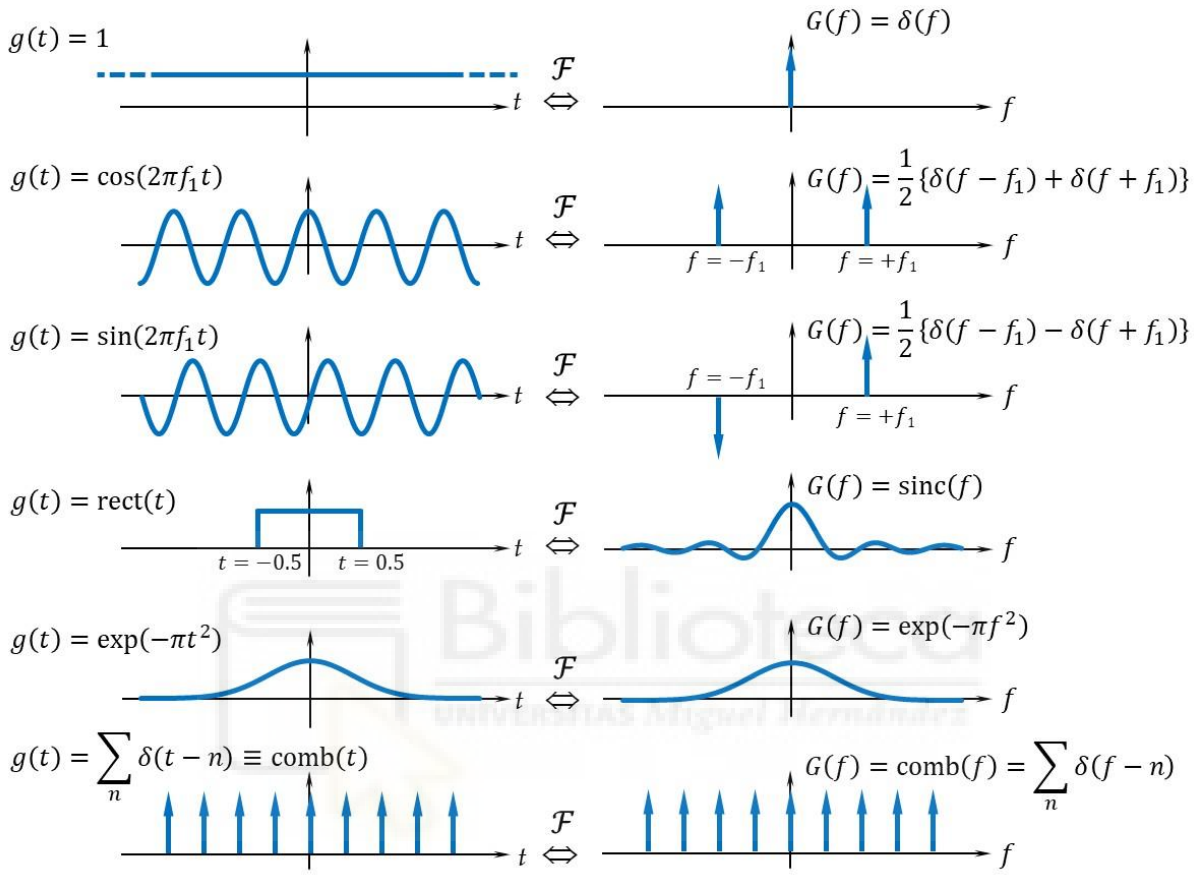

Fig. 2. 3. Some examples of Fourier transform pairs.

The cosine and sine functions lead to spectra in the form of two impulse delta functions because of the Euler relations, i.e.:

$$\mathcal{F}[\cos(2\pi f_1 t)] = \tfrac{1}{2}\mathcal{F}\left[e^{+i2\pi f_1 t} + e^{-i2\pi f_1 t}\right] = \tfrac{1}{2}\{\delta(f - f_1) + \delta(f + f_1)\}, \tag{2.20a}$$

$$\mathcal{F}[\sin(2\pi f_1 t)] = \tfrac{1}{2}\mathcal{F}\left[e^{+i2\pi f_1 t} - e^{-i2\pi f_1 t}\right] = \tfrac{1}{2}\{\delta(f - f_1) - \delta(f + f_1)\}, \tag{2.20b}$$

where the definitions in Eq. (2.17) and Eq. (2.19) were applied. The rectangle or pulse function is defined as

$$\text{rect}(t) = \begin{cases} 1 & \text{if } t \in \left(-\frac{1}{2}, +\frac{1}{2}\right), \\ 0 & \text{otherwise,} \end{cases} \tag{2.21}$$

whose Fourier transform is the sine cardinal function:

$$\mathcal{F}[\text{rect}(t)] = \text{sinc}(f). \tag{2.22}$$

The Gaussian function $\exp(-\pi t^2)$ has the property of being itself its Fourier transform, i.e.:

$$\mathcal{F}[\exp(-\pi t^2)] = \exp(-\pi f^2). \tag{2.23}$$

Finally, the comb function, defined as a train of impulses:

$$\text{comb}(t) \equiv \sum_{n=-\infty}^{+\infty} \delta\,(t-n), \tag{2.24}$$

is another function whose Fourier transform is a comb function itself:

$$\mathcal{F}[\text{comb}(t)] = \text{comb}(f) = \sum_{n=-\infty}^{+\infty} \delta\,(f-n). \tag{2.25}$$

2.2.2. Some properties of the Fourier transform: scaling, shift, and energy

Some important properties of the Fourier transform that will be used along the Thesis are the scaling property, the shift property, and the energy conservation property.

The **scaling property** indicates that given a function $g(t)$ with Fourier transform $G(f) = \mathcal{F}[g(t)]$, a scaled version $g(at)$ of this function (here a is a scalar value) has a Fourier transform that is scaled oppositely, since

$$\mathcal{F}[g(at)] = \frac{1}{|a|} G\left(\frac{f}{a}\right). \tag{2.26}$$

Figure 2.4 illustrates this property with an example based on the rect function. The scaled rectangle is $g(t) = \text{rect}(at)$ whose Fourier transform is $G(f) = \frac{1}{|a|}\text{sinc}(f/a)$.

When the scaling factor is selected $a = 1/2$, the rectangle function becomes two times wider, and its Fourier transform becomes a narrower and higher sinc function (the maximum central value is now $G(f = 0) = 2$). On the contrary, when the scaling factor is $a = 2$, the rectangle function becomes two times narrower, and its Fourier transform becomes a wider and lower sinc function (the maximum central value is now $G(f = 0) = 0.5$).

The Fourier transform **shift property** indicates that a lateral shift of the signal does not change the magnitude of its Fourier transform but adds a linear phase proportional to the amount

of shift. Similarly, adding a linear phase to the input signal provokes a frequency shift. This property is described mathematically as

$$\mathcal{F}[g(t-t_0)] = G(f)e^{-i2\pi t_0 f}, \tag{2.27a}$$

$$\mathcal{F}\left[g(t)e^{+i2\pi f_0 t}\right] = G(f-f_0). \tag{2.27b}$$

Finally, the **energy conservation theorem** states that the signal energy (defined as its magnitude squared) is the same in the direct space and in the frequency space:

$$\int_{-\infty}^{+\infty} |g(t)|^2 dt = \int_{-\infty}^{+\infty} |G(f)|^2 df. \tag{2.28}$$

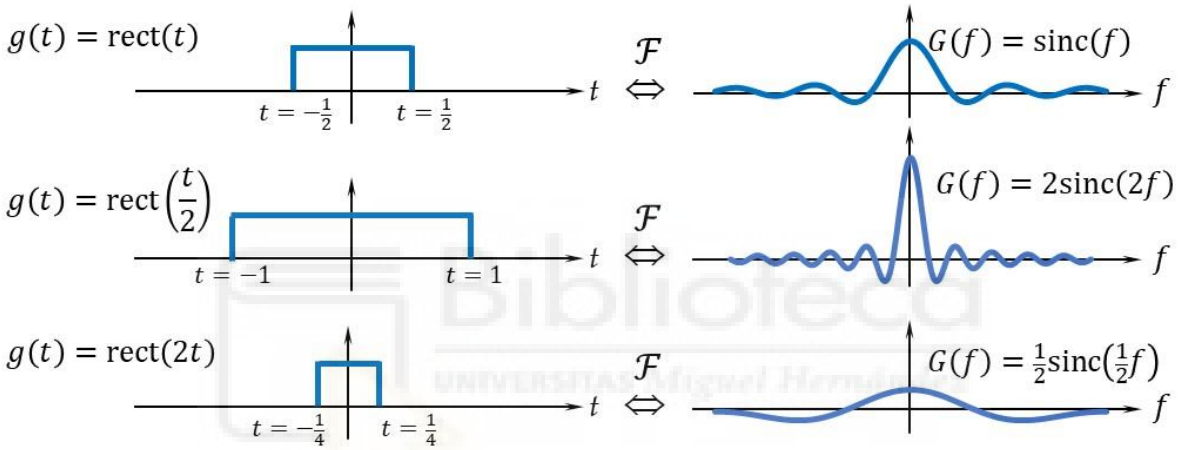

Fig. 2. 4. Scaling property of the rect - sinc Fourier transform pair.

2.2.3. The impulse function

The impulse function or Dirac delta function was already introduced in Section 2.2.1. It is a generalized function or distribution, used to model an idealized point. Regarding Fig. 2.4, the impulse function can be viewed as the Fourier transform of the rectangle function in the limit when the rectangle becomes wider and wider. Then, the sinc function becomes narrower and narrower, and in the limit, it provides the delta function defined as:

$$\delta(t) = \int_{-\infty}^{+\infty} e^{+i2\pi f t} df = \begin{cases} \infty & \text{if } t = 0, \\ 0 & \text{if } t \neq 0, \end{cases} \tag{2.29}$$

where now we consider a constant function to obtain through the inverse Fourier transform (i.e., the opposite transformation is applied compared to that in Eq. (2.19)).

The **normalization property** of the impulse function states that

$$\int_{-\infty}^{+\infty} \delta(t)dt = 1. \tag{2.30}$$

The **multiplication property** indicates that multiplying an impulse function by another function $g(t)$ gives a scalar value equal to the value of the function at the location of the impulse, and is zero everywhere else:

$$g(t)\delta(t - t_0) = g(t_0)\delta(t - t_0). \tag{2.31}$$

Finally, the **decomposition property** shows that any function can be viewed as the superposition of weighted impulse functions, i.e.

$$g(t) = \int_{-\infty}^{+\infty} g(\tau)\,\delta(t - \tau)d\tau. \tag{2.32}$$

2.3. Convolution and cross-correlation

Given two signals $g_1(t)$ and $g_2(t)$ their convolution is defined as the following operation:

$$g_1(t) * g_2(t) = \int_{-\infty}^{+\infty} g_1(\tau)\,g_2(t - \tau)d\tau. \tag{2.33}$$

Figure 2.5 illustrates the meaning of the convolution operation, showing that it represents the area below the product of one of the functions by a reflected version of the other function, as it is shifted along the axis.

The convolution theorem states that the Fourier transform of the convolution of two functions is equal to the product of the individual Fourier transforms:

$$\mathcal{F}[g_1(t) * g_2(t)] = G_1(f)G_2(f), \tag{2.34a}$$

where $G_1(f) = \mathcal{F}[g_1(t)]$ and $G_2(f) = \mathcal{F}[g_2(t)]$. In the same manner, the Fourier transform of the product of two functions is equal to the convolution of their Fourier transforms:

$$\mathcal{F}[g_1(t)g_2(t)] = G_1(f) * G_2(f). \tag{2.34b}$$

One relevant case is the convolution with an impulse delta function. If the delta function is shifted, then its convolution with another function is equal to a shifted version of this other

function:

$$g(t) * \delta(t - t_0) = g(t - t_0), \tag{2.35}$$

as illustrated in Fig. 2.6 for the case of a sinc function.

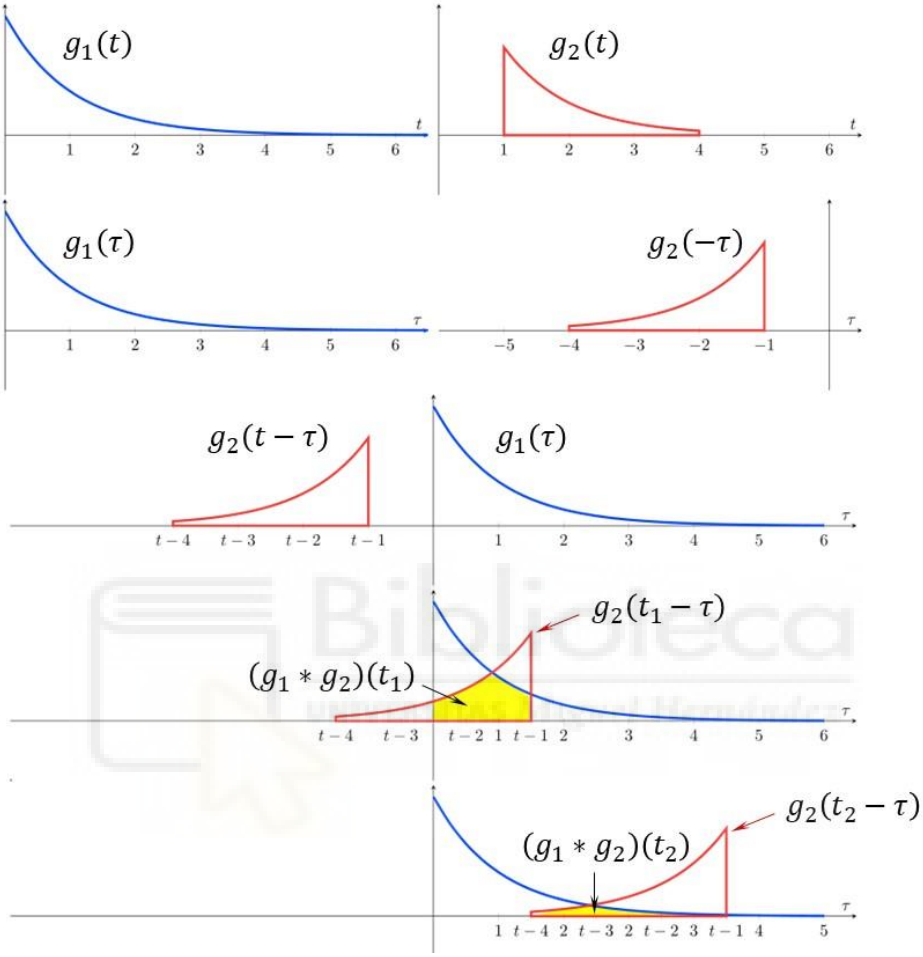

Fig. 2. 5. Illustration of the convolution of two functions.

The cross-correlation between two functions is a measure of the similarity between them. It is an operation closely related to the convolution, defined as:

$$g_1(t) \star g_2(t) = \int_{-\infty}^{+\infty} g_1^*(\tau)\, g_2(t+\tau) d\tau. \tag{2.36}$$

Comparing to Eq. (2.33), this shows that the cross-correlation between $g_1(t)$ and $g_2(t)$ is equivalent to the convolution between the function $g_1^*(-t)$ and $g_2(t)$, i.e.:

$$g_1(t) \star g_2(t) = g_1^*(-t) * g_2(t). \tag{2.37}$$

Using the Fourier transform definition in Eq. (2.17), it is straight showing that $\mathcal{F}[g^*(-t)] =$

$(\mathcal{F}[g(t)])^*$. Therefore, the Fourier transform of the correlation between $g_1(t)$ and $g_2(t)$ is related to the individual Fourier transforms $G_1(f)$ and $G_2(f)$ as:

$$\mathcal{F}[g_1(t) \star g_2(t)] = G_1^*(f)G_2(f). \tag{2.38}$$

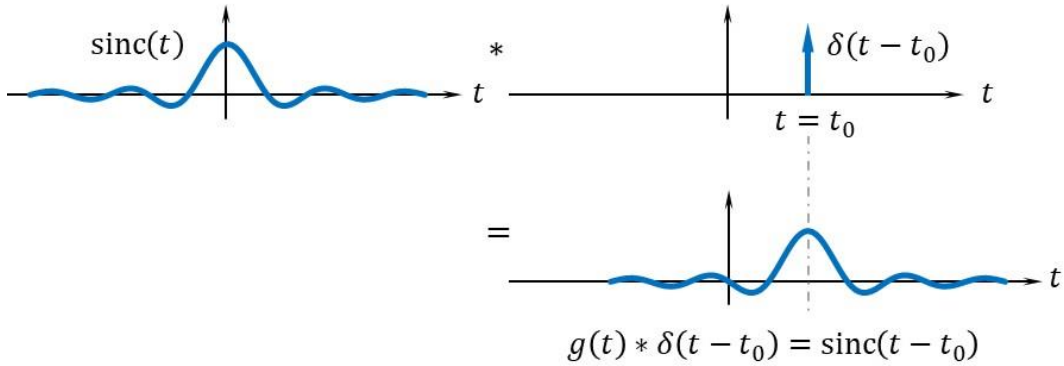

Fig. 2. 6. Illustration of the convolution of a sinc function with a shifted impulse function.

When the two functions are the same $g_1(t) = g_2(t) \equiv g(t)$, the correlation $g(t) \star g(t)$ is named as autocorrelation, and its Fourier transform is given by the squared magnitude $\mathcal{F}[g(t) \star g(t)] = G^*(f)G(f) = |G(f)|^2$.

2.4. Sampling

The impulse function and the convolution can be used to describe the effect of sampling a continuous function, which means discretizing the values taken by the function. Figure 2.7 illustrates the sampling process. A continuous function $g(t)$ with Fourier transform $G(f)$ is considered. In Fig. 2.7 $g(t)$ is plotted as a wide Gaussian-like function, thus leading to a $G(f)$ function of narrow Gaussian shape. The sampled version $g_S(t)$ of this function can be described as the multiplication of $g(t)$ by a scaled comb function, where the sampling rate is defined by the distance Δ between the delta functions in the comb function, i.e.:

$$g_S(t) = g(t) \sum_{n=-\infty}^{+\infty} \delta(t - n\Delta). \tag{2.39}$$

Considering the multiplication property of Eq. (2.31), Equation (2.39) can be written as

$$g_S(t) = \sum_{n=-\infty}^{+\infty} g(n\Delta)\delta(t - n\Delta), \tag{2.40}$$

which shows that the sampled function is a set of delta functions separated by a distance Δ, each one weighted by the value $g(n\Delta)$ that the function $g(t)$ takes at the location of the delta functions.

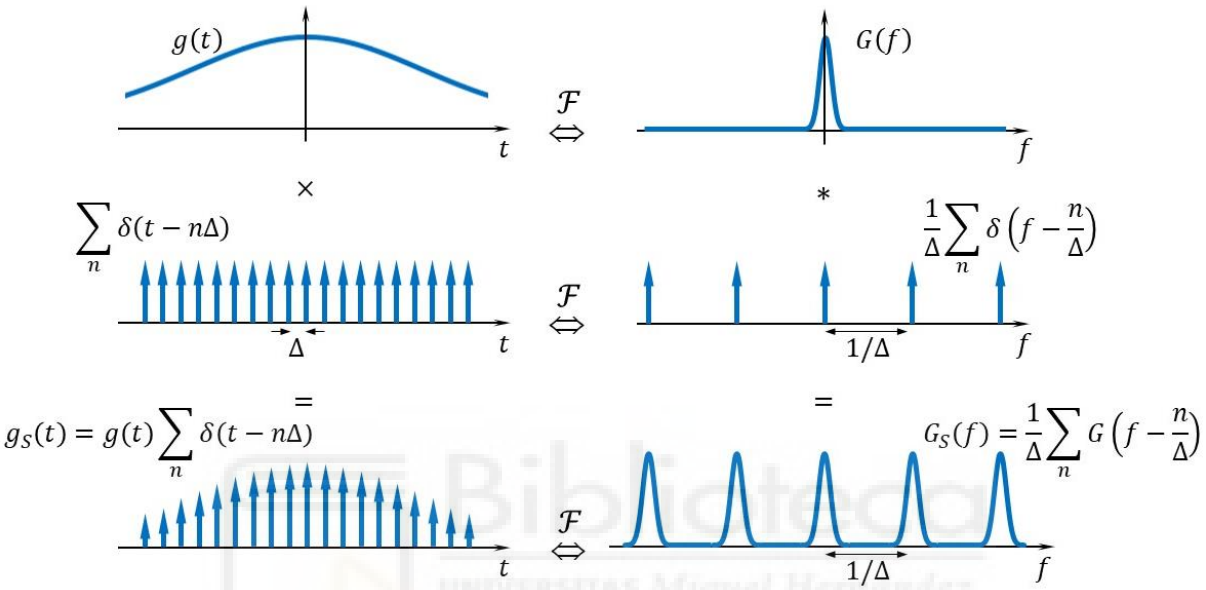

Fig. 2. 7. Illustration of sampling a function and its effect on the Fourier transform.

The scaling property of the Fourier transform, when applied to the comb function, leads to the following pair relation:

$$\sum_{n=-\infty}^{+\infty} \delta(t - n\Delta) = \frac{1}{\Delta}\mathrm{comb}\left(\frac{t}{\Delta}\right) \quad \overset{\mathcal{F}}{\Leftrightarrow} \quad \mathrm{comb}(f\Delta) = \sum_{n=-\infty}^{+\infty} \delta\left(f - \frac{n}{\Delta}\right). \tag{2.41}$$

This shows that the delta functions of the comb function in the Fourier domain are separated inversely to the separation of the delta functions of the comb function in the signal domain, as indicated in the second row in Fig. 2.7.

Therefore, considering the property in Eq. (2.34b) (i.e. the Fourier transform of the product of two functions), the Fourier transform of the sampled function $g_S(t)$ can be described as the convolution between the spectrum $G(f)$ of the continuous function and the comb function in the frequency domain:

$$G_S(f) = \mathcal{F}[g(t)] * \mathcal{F}\left[\sum_{n=-\infty}^{+\infty} \delta(t - n\Delta)\right] = G(f) * \left[\frac{1}{\Delta}\sum_{n=-\infty}^{+\infty} \delta\left(f - \frac{n}{\Delta}\right)\right]. \quad (2.42)$$

Finally, considering the convolution property of the delta functions in Eq. (2.35), it can be written as

$$G_S(f) = \frac{1}{\Delta}\sum_{n=-\infty}^{+\infty} G\left(f - \frac{n}{\Delta}\right). \quad (2.43)$$

This shows that the effect of sampling the signal provokes the generation of multiple replicas of the original spectrum in the Fourier transform, being the separation between replicas inversely proportional to the sampling rate Δ. This limits the rate at which a function can be sampled in what is known as the Whitaker-Shannon sampling theorem.

2.5. The two-dimensional Fourier transform

In Optics, the diffraction phenomena are described employing two-dimensional (2D) Fourier transforms, where functions are considered in two dimensions, i.e. $g(x,y)$, where $\mathbf{r} = (x,y)$ is a vector denoting the cartesian spatial coordinates in the plane of the diffractive element. The diffracted field depends on the 2D Fourier transform of $g(x,y)$, given by

$$G(u,v) = \mathcal{F}_{2D}[g(x,y)] = \iint_{-\infty}^{+\infty} g(x,y)e^{-i2\pi(ux+vy)}\,dxdy, \quad (2.44)$$

where $\mathcal{F}_{2D}$ indicates the 2D Fourier transform operator, and $\mathbf{u} = (u,v)$ is a vector whose coordinates u and v denote, respectively, the spatial frequencies in the horizontal and vertical directions. This relation can be written in a more compact way using the scalar product $\mathbf{u}\cdot\mathbf{r}$ as

$$G(\mathbf{u}) = \mathcal{F}_{2D}[g(\mathbf{r})] = \iint_{-\infty}^{+\infty} g(x,y)e^{-i2\pi\mathbf{u}\cdot\mathbf{r}}d\mathbf{r}. \quad (2.45)$$

When the function is separable in the (x,y) coordinates, i.e. $g(x,y) = g_X(x)g_Y(y)$, the 2D Fourier transform can be written as two separate Fourier transform integrals

$$G(u,v) = \left(\int_{-\infty}^{+\infty} g_X(x)e^{-i2\pi ux}dx\right)\left(\int_{-\infty}^{+\infty} g_Y(y)e^{-i2\pi vy}dy\right). \quad (2.46)$$

Therefore, it is also separable and can be written as $G(u,v) = G_U(u)G_V(v)$, i.e. as the product of

two one-dimensional Fourier transformations $G_U(u) = \mathcal{F}_{1D}[g_X(x)]$ and $G_V(v) = \mathcal{F}_{1D}[g_Y(y)]$.

Figure 2.8 illustrates this property for a rectangular function. The top row displays three rectangles with the same width along the horizontal (x) direction but variable height along the vertical (y) direction. The images on the bottom row show the magnitude of the corresponding 2D Fourier transforms (the central part has been saturated to clearly view the side lobes in the function). These examples correspond to the typical diffraction generated by a rectangular aperture. These simulations used images of 512×512 pixels. The rectangular shapes in the top row have 20×20 pixels, 20×40 pixels, and 20×512 pixels respectively. For a better visualization, the images in Fig. 2.8 were cropped to half its size. The simulations were made with a Python code that uses the 2D digital Fourier transform algorithm, included at the end of the chapter.

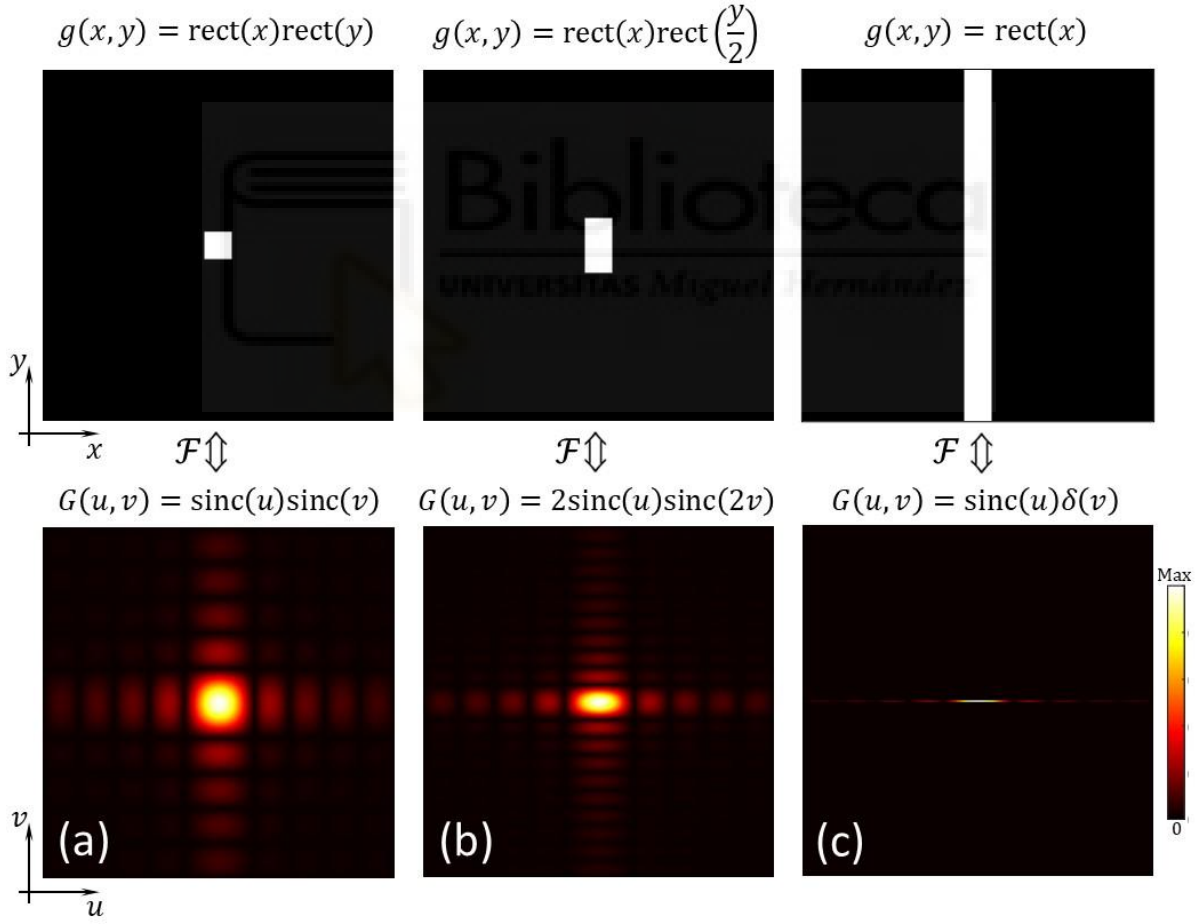

Fig. 2. 8. Several rectangle functions and the magnitude of their corresponding Fourier transforms.

In Fig. 2.8(a) the aperture is square, and the product of two equal sinc functions along the horizontal and vertical directions is observed in the magnitude of the Fourier transform. In Fig.

2.8(b) the aperture height is twice its width and consequently the sinc function along the vertical direction gets compressed by a factor of two. Finally, in Fig. 2.8(c), the rectangle is considered unlimited along the vertical direction, i.e. it reproduces a slit. In this case the sinc function along the vertical direction is so narrow that it can be considered a delta function $\delta(v)$. Note that the intensity of the diffraction patterns shown in the figure are presented using a pseudo-colormap ('hot'), to better visualize the variations (this colour code must not be confused with the diffraction pattern generated with polychromatic light, which will be analysed in other chapters of the Thesis).

A second example is shown in Fig. 2.9. Again, we consider an image of 512×512 pixels, and a central square of 64×64 pixels. In this central square we first consider horizontal rectangles 64 pixels wide in Figs. 2.9(a), 2.9(b), and 2.9(c), whose height is of 8, 4 and 2 pixels respectively, separated vertically by the same number of by black pixels. They, thus, mimic 1D binary amplitude diffraction gratings with periods of 16, 8 and 4 pixels respectively, limited by the square aperture of 64×64 pixels.

Despite the low number of periods (4, 8 and 16 in Figs. 2.9(a), 2.9(b), and 2.9(c) respectively), the magnitude of the corresponding Fourier transform illustrates the concentration on certain spatial frequencies along the vertical direction (along the v axis), which in the case of diffraction gratings correspond to the diffraction orders. Note how the separation between orders in the Fourier domain increases as the period is reduced, a consequence of the scaling property.

Figures 2.9(d), 2.9(e), and 2.9(f) illustrate equivalent results, but now using 2D arrays obtained by multiplying the grating by another version of it rotated by 90 degrees. The result is a grid of 4×4 squares of 8×8 pixels (Fig. 2.9(d)), a grid of 8×8 squares of 4×4 pixels (Fig. 2.9(e)), and a grid of 16×16 squares of 2×2 pixels (Fig. 2.9(e)). The magnitude of the corresponding Fourier transforms shows a 2D grid of orders. These simulations illustrate the situation that is encountered when using SLMs, which will be described in detail in the next chapters.

2.5.1. Two-dimensional convolution and correlation

The convolution and cross-correlation operations defined in Eqs. (2.33) and (2.36) respectively can be extended to two dimensional functions, being essential operations in image processing. The 2D convolution and the 2D correlation of functions $g_1(x,y)$ and $g_2(x,y)$ is defined respectively as:

$$[g_1 * g_2](x,y) = \int_{-\infty}^{+\infty} g_1(\tau_x,\tau_y)\, g_2(x-\tau_x, y-\tau_y) d\tau_x d\tau_y. \tag{2.47a}$$

$$[g_1 \star g_2](x,y) = \int_{-\infty}^{+\infty} g_1^*(\tau_x,\tau_y)\, g_2(x+\tau_x, y+\tau_y) d\tau_x d\tau_y. \tag{2.47b}$$

They are related to the 2D Fourier transforms of each individual functions as

$$\mathcal{F}_{2D}[g_1(\mathbf{r}) * g_2(\mathbf{r})] = G_1(\mathbf{u})G_2(\mathbf{u}), \tag{2.48a}$$

$$\mathcal{F}_{2D}[g_1(\mathbf{r}) \star g_2(\mathbf{r})] = G_1^*(\mathbf{u})G_2(\mathbf{u}), \tag{2.48b}$$

where $G_1(\mathbf{u}) = \mathcal{F}_{2D}[g_1(\mathbf{r})]$ and $G_2(\mathbf{u}) = \mathcal{F}_{2D}[g_2(\mathbf{r})]$.

One important property of the convolution is the shift property, which states that if $h(\mathbf{r})$ denotes the convolution $g_1(\mathbf{r}) * g_2(\mathbf{r})$, then the convolution of the same displaced functions results in a shifted version of the convolution, i.e.:

$$g_1(\mathbf{r}-\mathbf{r_1}) * g_2(\mathbf{r}-\mathbf{r_1}) = h(\mathbf{r}-\mathbf{r_1}-\mathbf{r_2}), \tag{2.49}$$

where $\mathbf{r_1} = (x_1, y_1)$ and $\mathbf{r_2} = (x_2, y_2)$ indicate the displacements of $g_1(\mathbf{r})$ and $g_2(\mathbf{r})$ respectively.

Figure 2.10 shows several examples of convolution and cross-correlation simulations. We use an identical image with a lotus pattern for $g_1(\mathbf{r})$ and $g_2(\mathbf{r})$, so the convolution and the auto-correlation functions are calculated. In Fig. 2.10(a) the two lotuses are located in the centre, with a size of $\mathbf{110 \times 110}$ pixels. The second column in Fig. 2.10(a) shows the simulation results of convolution and auto-correlation, respectively. Both of them are in the shape of a cloud with a brighter centre. The differences between them are: 1. The contour of the two clouds is a little different. 2. The auto-correlation shows a higher and narrower peak, as shown in their 3D plotting. The maximum value of auto-correlation is higher than that of the convolution, since in the former calculation, the area of the intersection between two patterns is larger than the latter.

Finally, we illustrate the convolution shift property. Figure 2.10(b) shifts one pattern to the left, and Fig. 2.10(c) shifts the other pattern to the top with respect to Fig. 2.10(b). Their corresponding convolution and auto-correlation are the same as Fig. 2.10(a) but with the different positions. In Fig. 2.10(b) the convolution and the correlation are shifted to the left the same amount as the shifted pattern is moved. In Fig. 2.10(c), since both lotus patterns are shifted, one horizontally and the other vertically, the resulting convolution and correlation appear centred at the addition of the two displacements, i.e., in diagonal direction. We will make use of this convolution/correlation shift property in Chapter 9.

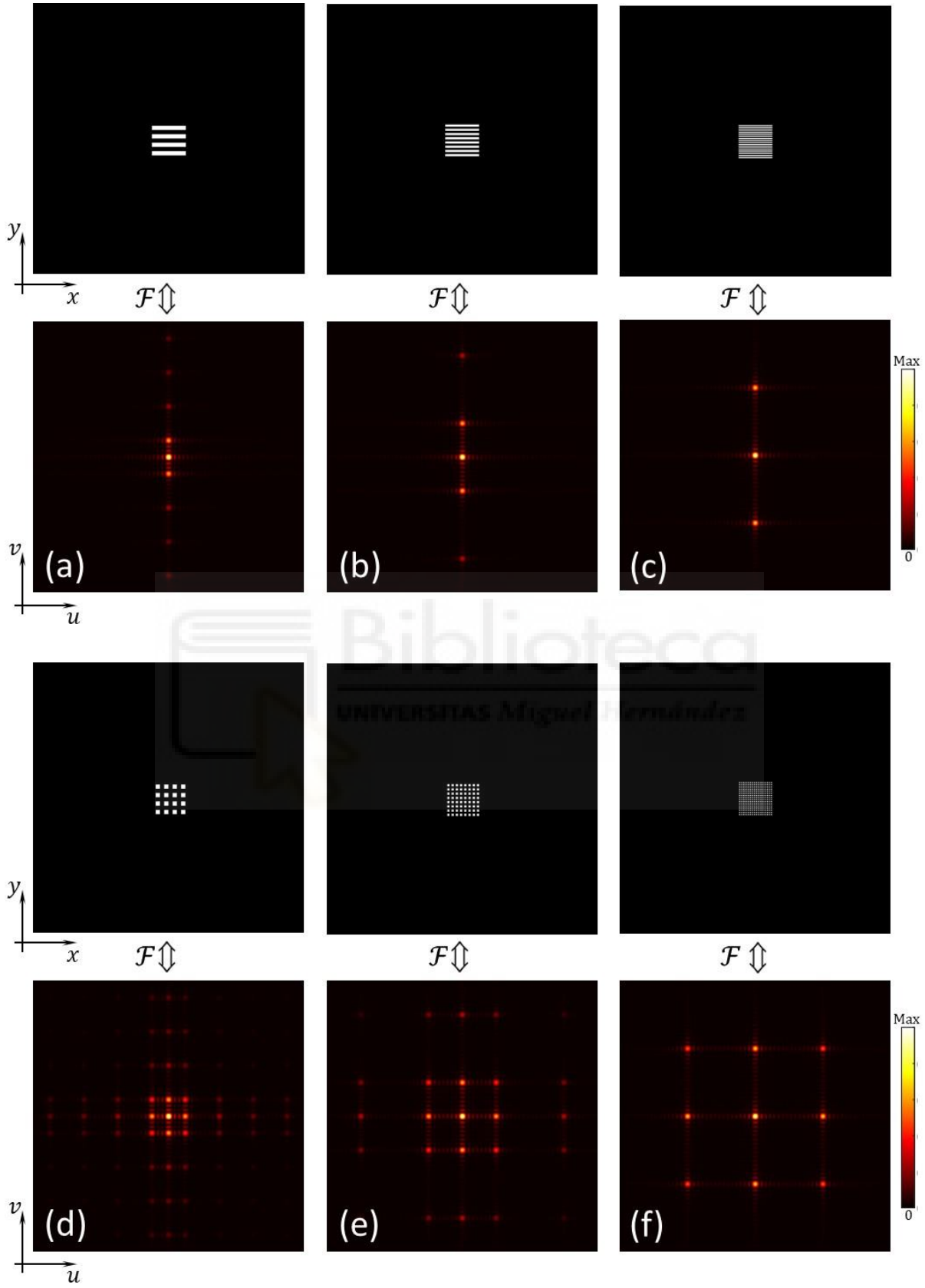

Fig. 2. 9. 1D binary amplitude gratings with periods of (a) 16 pixels, (b) 8 pixels, and (c) 4 pixels, and the magnitude of the corresponding Fourier transform. Equivalent results for 2D arrays obtained by multiplication of the binary grating with a 90 degrees rotated version of it. Periods of: (d) 16 pixels, (e) 8 pixels and (f) 4 pixels.

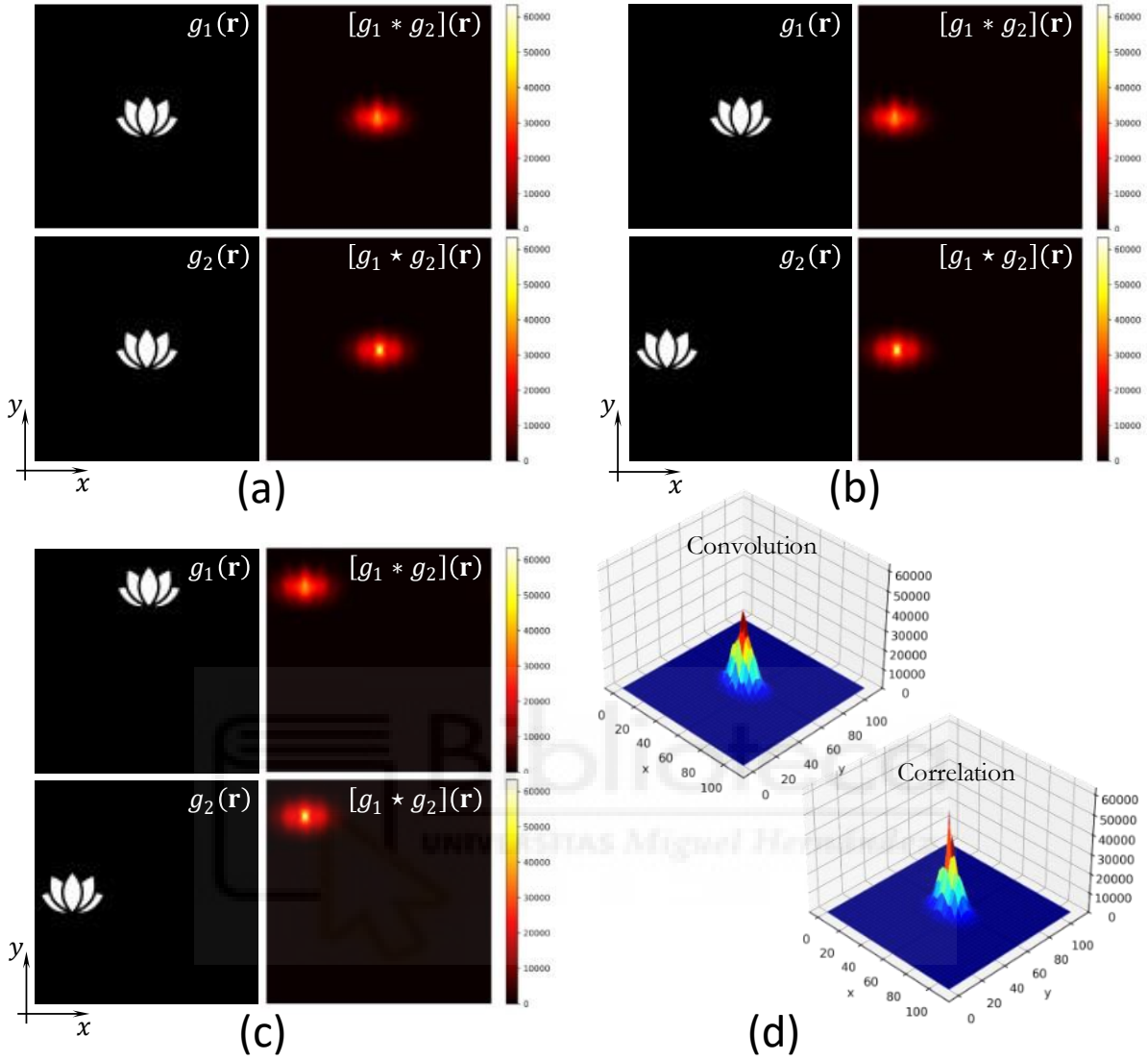

Fig. 2. 10. Simulated convolution and cross-correlation results between different combinations: (a) Two lotuses in the centre. (b) One is in the centre and the other shifts to the left. (c) One shifts to the left and the other shifts to the top. (d) 3D plots of the convolution and correlation functions in (a).

Python Codes

The next Python code was designed for generating the images in Figs. 2.8. It includes 2D fast Fourier transform (FFT) demonstrations of a square, a rectangle, and a slit:

Script 1.

```
import numpy as np
from scipy.fft import fft2, fftshift
import matplotlib.pyplot as plt

"""
```

```
- The image is a 2D array with 512*512 pixels, filled with 0 as background.
- Three shapes are defined: a square, a rectangle and a slit.
- The 2D FFT algorithm is applied to each shape
- The "fftshift" is required for shifting the zero spatial frequency to the centre in the frequency
domain.
- Finally, the magnitude of FFT for different shapes are saved..
"""
square=np.zeros((512,512))
width,height=square.shape
center=int(width/2)
width_c=10
square[center-width_c:center+width_c,center-width_c:center+width_c]=1
"""A rectangle is designed in the size of 20*20 pixels filled with 1. """
plt.imsave("square.png", square,cmap='gray')
square_fft=fftshift(fft2(square))
square_mag=abs(square_fft)/np.max(abs(square_fft))
plt.imsave("square_fft.png", square_mag,cmap='hot')
"""A rectangle is designed in the size of 20*40 pixels filled with 1. """
rect=np.zeros((512,512))
rect[center-2*width_c:center+2*width_c,center-width_c:center+width_c]=1
plt.imsave("rect.png", rect,cmap='gray')
rect_fft=fftshift(fft2(rect))
rect_fft_mag=abs(rect_fft)/np.max(abs(rect_fft))
plt.imsave("rect_fft.png", rect_fft_mag,cmap='hot')
"""A slit is designed in the size of 20*512 pixels filled with 1. """
slit=np.zeros((512,512))
slit[:,int(center)-int(1*width_c):int(center)+int(1*width_c)]=1
plt.imsave("slit.png", slit,cmap='gray')
slit_fft=fftshift(fft2(slit))
slit_mag=abs(slit_fft)/np.max(abs(slit_fft))
plt.imsave("slit_fft.png", slit_mag,cmap='hot')
```

This second Python was designed for generating the images in Fig. 2.9. It includes 2D fast Fourier transform (FFT) demonstrations of 1D multiple slits and 2D of rectangular apertures that resemble the pixels in an SLM.

Script 2.

```
import numpy as np
from scipy.fft import fft2, fftshift
import matplotlib.pyplot as plt

"""
- A square white aperture of 64*64 pixels is used at the centre of a 512*512 pixels image.
- Multiple slits along vertical and horizontal directions, separated evenly, to create an amplitude
grating.
- A 2D version is calculated as well.
"""
square=np.zeros((512,512))
width,height=square.shape
center=int(width/2)
width_c=32
period=16
half_period=int(period/2)
square[center-width_c:center+width_c,center-width_c:center+width_c]=1
for i in range(center - width_c, center + width_c, period):
    square[center - width_c:center + width_c, i:i + half_period] = 0
plt.imsave(f"Vertical slits p_{period}.png", square,cmap='gray')
"""2D FFT is applied to the shape of multiple vertical slits. """
square_fft=fftshift(fft2(square))
square_mag=abs(square_fft)/np.max(abs(square_fft))
plt.imsave(f"FFT_Vslits p_{period}.png", square_mag,cmap='hot')
"""2D FFT is applied to the shape of multiple horizontal slits. """
square_h=np.zeros((512,512))
width_h,height_h=square_h.shape
center_h=int(width_h/2)
square_h[center_h-width_c:center_h+width_c,center_h-width_c:center_h+width_c]=1
for i in range(center_h-width_c, center_h+width_c, period):
    square_h[i:i + half_period,center_h - width_c:center_h+width_c] = 0
plt.imsave(f"Horizontal slits p_{period}.png", square_h,cmap='gray')
```

```
square_h_fft=fftshift(fft2(square_h))
square_h_mag=abs(square_h_fft)/np.max(abs(square_h_fft))
plt.imsave(f"FFT_Hslits p_{period}.png", square_h_mag,cmap='hot')
"""2D FFT is applied to the shape of a grid resembling an SLM. """
SLM=np.zeros((512,512))
width_s,height_s=SLM.shape
center_s=int(width_s/2)
SLM[center_s-width_c:center_s+width_c,center_s-width_c:center_s+width_c]=1
for i in range(center_h-width_c, center_h+width_c, period):
    SLM[i:i + half_period,center_s - width_c:center_s+width_c] = 0
    SLM[center - width_c:center + width_c, i:i + half_period] = 0
plt.imsave(f"SLM p_{period}.png", SLM,cmap='gray')

SLM_fft=fftshift(fft2(SLM))
SLM_mag=abs(SLM_fft)/np.max(abs(SLM_fft))
plt.imsave(f"FFT_SLM p_{period}.png", SLM_mag,cmap='hot')
```

This third Python was designed for generating the images in Fig. 2.10. The FFT of the original patterns are computed, and the convolution and cross-correlation are calculated based on the magnitude and phase in the FT field.

Script 3.

```
import numpy as np
from scipy.fftpack import ifft2, ifftshift, fft2,fftshift
import matplotlib.pyplot as plt

im1=plt.imread("C:/Users/Laboratorio/Conv and Corr/lotus_center.png")
im2=plt.imread("C:/Users/Laboratorio/Conv and Corr/lotus_FH_0_S.png")
im3=plt.imread("C:/Users/Laboratorio/Conv and Corr/lotus_V_S.png")

im1shift=fftshift(im1[:,:,0])
im2shift=fftshift(im2[:,:,0])
im3shift=fftshift(im3[:,:,0])

h,w=im1shift.shape
"""No noise"""
file1="No noise"
# FFT of the original images.
FT_forward=fftshift(fft2(im1shift))
FT_forward_L=fftshift(fft2(im2shift))
FT_forward_V=fftshift(fft2(im3shift))
# Multiplication of FT fields with magnitude and phase, the complex conjugation operation is for
cross-correlation.
con1=FT_forward*FT_forward
cor1=FT_forward*np.conjugate(FT_forward)
# Inverse FFT of multiplication, where the reconstruction of convolution and cross-correlations are
obtained.
In_FT_con1=ifftshift(ifft2(con1))
In_FT_cor1=ifftshift(ifft2(cor1))
I_con1=abs(In_FT_con1)**2
I_cor1=abs(In_FT_cor1)**2

con2=FT_forward*FT_forward_L
cor2=FT_forward_L*np.conjugate(FT_forward)

In_FT_con2=ifftshift(ifft2(con2))
In_FT_cor2=ifftshift(ifft2(cor2))
I_con2=abs(In_FT_con2)**2
I_cor2=abs(In_FT_cor2)**2

con3=FT_forward_V*FT_forward_L
cor3=FT_forward_V**np.conjugate(FT_forward_L)

In_FT_con3=ifftshift(ifft2(con3))
In_FT_cor3=ifftshift(ifft2(cor3))
I_con3=abs(In_FT_con3)**2
I_cor3=abs(In_FT_cor3)**2

max_=I_cor1.max()
```

```
def plotim(I_con,I_cor,t):
    plt.figure()
    plt.imshow(I_con,vmax=max_,cmap="hot")
    plt.colorbar()
    plt.axis("off")
    plt.savefig(f"Convolution {file1} {t}.png", dpi=300, bbox_inches="tight")
    plt.close()

    x = np.arange(0, w)
    y = np.arange(0, h)
    X,Y=np.meshgrid(x,y)

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_surface(X, Y, I_con, cmap='jet')
    ax.set_zlim(0, max_)
    ax.view_init(elev=35, azim=-45)
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    plt.savefig(f"Convolution {file1} {t} 3d.png", dpi=300, bbox_inches="tight")
    plt.close()

    plt.figure()
    plt.imshow(I_cor,vmax=max_,cmap="hot")
    plt.colorbar()
    plt.axis("off")
    plt.savefig(f"Correlation {file1} {t}.png", dpi=300, bbox_inches="tight")
    plt.close()

    fig2 = plt.figure()
    ax2 = fig2.add_subplot(111, projection='3d')
    ax2.plot_surface(X, Y, I_cor, cmap='jet')
    ax2.set_zlim(0, max_)
    ax2.set_xlabel("x")
    ax2.set_ylabel("y")
    ax2.view_init(elev=35, azim=-45)
    plt.savefig(f"Correlation {file1} {t} 3d.png", dpi=300, bbox_inches="tight")
    plt.close()
plot1=plotim(I_con1, I_cor1, 1)
plot2=plotim(I_con2, I_cor2, 2)
plot1=plotim(I_con3, I_cor3, 3)
```

3. Light diffraction and Fourier optics

This chapter reviews the diffraction properties of light and its relation to the Fourier analysis presented in the previous chapter. This allows to introduce the basic concepts of Fourier optics in a way that is suitable for describing the spatial light modulators and the displayed diffractive elements. The optical architectures employed to obtain the optical Fourier transform are introduced and described. All the treatment is based on the classical simplest approach of Fourier optics, where light beams are considered within the paraxial approximation, where a two-dimensional Fourier relation is obtained for the diffracted field with respect to the field behind the diffractive element [Goo-2005].

3.1. Fresnel and Fraunhofer diffraction approximations

Let us consider the basic problem in optics where the wavefront right after a diffractive element is known, and we want to evaluate how this field propagates a certain distance d and generates a diffracted field. The solution is given by the Fresnel-Huygens principle of wave optics, which states that each point of the wavefront generates a spherical wave, and the envelope of these secondary

waves constitutes the new wavefront. Such a spherical wavefront is expressed mathematically as a function in the three-dimensional space given by [Sha-1999]

$$s(R) = \frac{e^{ikR}}{\lambda R}, \tag{3.1}$$

where $k = 2\pi/\lambda$ is the light wavenumber, λ is the wavelength and R is the distance from the point source to the observation point, as indicated in Fig. 3.1 (a). The exponential term in the numerator accounts for the wave's spherical form, while the λR term in the denominator accounts for the spreading of the amplitude as the wave moves away from the point source. Note that the function in Eq. (3.1) corresponds to a diverging spherical wave, where the wave leaves the point source. A spherical wave converging to the central point is expressed with the complex conjugated version of Eq. (3.1).

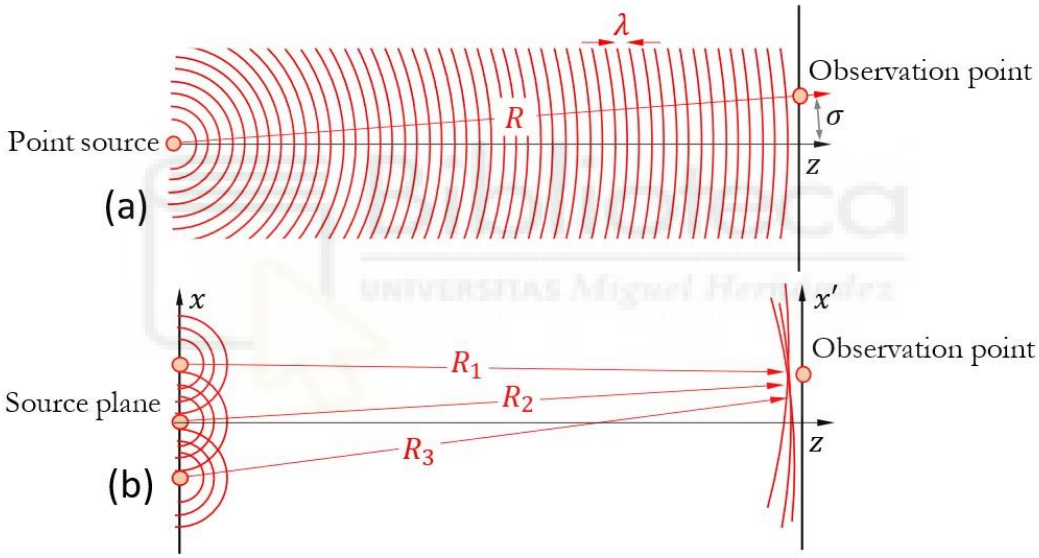

Fig. 3. 1. (a) Parabolic approximation of a spherical wave at large distances. (b) Scheme to calculate the propagated wavefront.

Light beams are considered in the paraxial approximation, where the beam propagates around the z axis with very small angles σ. Therefore, from Fig. 3.1 (a), $R^2 = z^2 + r^2$, where $r = |\mathbf{r}| = \sqrt{x^2 + y^2}$ located in a transversal plane with coordinates $\mathbf{r} = (x, y)$. In the small angle approximation, the quantity $\sigma \sim \tan(\sigma) = r/z$ is very small, and the distance R can be approximated as:

$$R = z\sqrt{1 + \left(\frac{r}{z}\right)^2} \cong z\left[1 + \frac{1}{2}\left(\frac{r}{z}\right)^2\right]. \tag{3.2}$$

Therefore, the spherical wave in Eq. (3.1) can be approximated as

$$s(\mathbf{r}) = \frac{e^{ikR}}{\lambda R} \cong \frac{e^{ikz}}{\lambda z} e^{i\frac{kr^2}{2z}} = \frac{e^{ikz}}{\lambda z} e^{i\frac{k}{2z}(x^2+y^2)} \tag{3.3}$$

where the denominator was approximated simply as $R \cong z$, while Eq. (3.2) was applied in the numerator, where the phase variations are more relevant. The relation in Eq. (3.3) is known as the **Fresnel approximation** or the parabolic approximation of the spherical wave [Sal-1991].

Assuming a continuous distribution of point sources located in another transversal plane with coordinates $\mathbf{r}_1 = (x_1, y_1)$, as illustrated in Fig. 3.1(b) by three points, each point generates a parabolic wave on the observation plane, with transversal coordinates $\mathbf{r}_2 = (x_2, y_2)$. This can be expressed mathematically as:

$$g'(\mathbf{r}_2) = \frac{e^{ikz}}{\lambda z} \iint_{-\infty}^{+\infty} g(\mathbf{r}_1) e^{i\frac{k}{2z}|\mathbf{r}_2 - \mathbf{r}_1|^2} d\mathbf{r}_1 \tag{3.4}$$

This equation is known as the Fresnel-Kirchhoff diffraction integral. Note that it can be written in the form of the two-dimensional convolution between the input wavefront $g(\mathbf{r})$ and the parabolic wave function $s(\mathbf{r})$ in Eq. (3.3) as

$$g'(\mathbf{r}) = g(\mathbf{r}) * s(\mathbf{r}). \tag{3.5}$$

By expanding the exponential term in Eq. (3.4), the diffraction integral is written as the following popular expression known as the **Fresnel diffraction approximation**

$$g'(\mathbf{r}_2) = \frac{e^{ikz}}{\lambda z} e^{i\frac{kr_2^2}{2z}} \iint_{-\infty}^{+\infty} g(\mathbf{r}_1) e^{-i\frac{2k\mathbf{r}_2 \cdot \mathbf{r}_1}{2z}} e^{i\frac{kr_1^2}{2z}} d\mathbf{r}_1. \tag{3.6}$$

If the propagation distance $z = d$ is large enough, the diffraction enters in the domain known as the **Fraunhofer approximation** or **far field approximation**. In this situation, the two quadratic exponentials in Eq. (3.6) change very slowly and can be ignored, and the wavefront takes the form:

$$g'(\mathbf{r}_2) = \frac{e^{ikd}}{\lambda d} \iint_{-\infty}^{+\infty} g(\mathbf{r}_1) e^{-i\frac{k\mathbf{r}_2 \cdot \mathbf{r}_1}{d}} d\mathbf{r}_1. \tag{3.7}$$

This relation shows the Fourier transform relation between the wavefront right after the diffractive element and the propagated field, as illustrated in Fig. 3.2. Equation (3.7) explicitly written in the transversal cartesian coordinates is given by

$$g'(x_2,y_2) = \frac{e^{ikd}}{\lambda d} \iint_{-\infty}^{+\infty} g(x_1,y_1) e^{-i\frac{2\pi}{\lambda d}(x_2x_1+y_2y_1)}\, dx_1 dy_1, \tag{3.8}$$

which, ignoring the normalization factor outside the integral, states that $g'(\mathbf{r}_2) = \mathcal{F}_{2D}[g(\mathbf{r}_1)]$ as defined in Eq. (2.45), where the spatial frequencies of the Fourier transform of $g(\mathbf{r}_1)$ are related to the spatial coordinates as $\mathbf{r}_2 = \lambda d\mathbf{u}$.

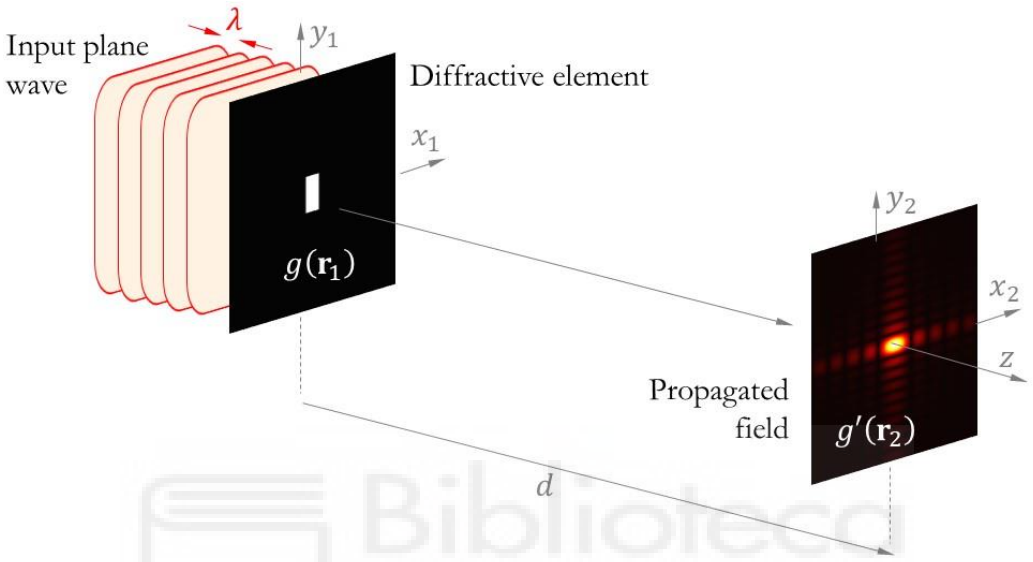

Fig. 3. 2. The far field diffraction within the Fraunhofer approximation.

3.2. Ray matrix operator algebra for Fourier optical systems

In practice, Fourier optics is based on using lenses to obtain the Fourier transform of diffractive elements in a shorter distance. The far field Fraunhofer approximation, which states that the diffracted field is equal to the Fourier transform of the field after the diffractive element, is more precise as the propagation distance is larger. Since a lens provides an image of infinity in its back focal plane, one can understand that the exact optical Fourier transform can be found by adding a converging lens to the scheme in Fig. 3.2.

The ray matrix operator algebra provides a very useful tool for analysing paraxial lens systems, useful to derive the classical geometric optics, but also to analyse diffractive systems [Mor-2005, Mor-2010]. Within this framework, a lens system is described by a 2×2 real valued matrix known as the ray transfer matrix (or ABCD matrix) that relates the position and angle coordinates (r, σ) of input and output rays, i.e.:

$$\begin{pmatrix} r_2 \\ \sigma_2 \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} r_1 \\ \sigma_1 \end{pmatrix}, \tag{3.9}$$

The two basic building blocks in optical systems are the free propagation and the lens, as shown in Fig. 3.3. A ray free propagation of an axial distance d keeps the angle and only modifies the ray height (as shown in Fig. 3.3(a)), so its ray transfer matrix is:

$$\mathbf{M}_{\wp}(d) = \begin{pmatrix} 1 & d \\ 0 & 1 \end{pmatrix}. \tag{3.10}$$

On the contrary, the lens modifies the angular coordinate, leaving unchanged the height coordinate (Fig. 3.3(b)). The corresponding ray matrix is:

$$\mathbf{M}_{\mathcal{L}}(f') = \begin{pmatrix} 1 & 0 \\ -1/f' & 1 \end{pmatrix}, \tag{3.11}$$

where f' denotes the lens focal length.

Note that optical systems have rotational symmetry, so here the r coordinate represents the radial coordinate in a transversal plane. The schemes in Fig. 3.3. represent a longitudinal cross section of the system so, without losing generality, we can assume r for instance as the x coordinate.

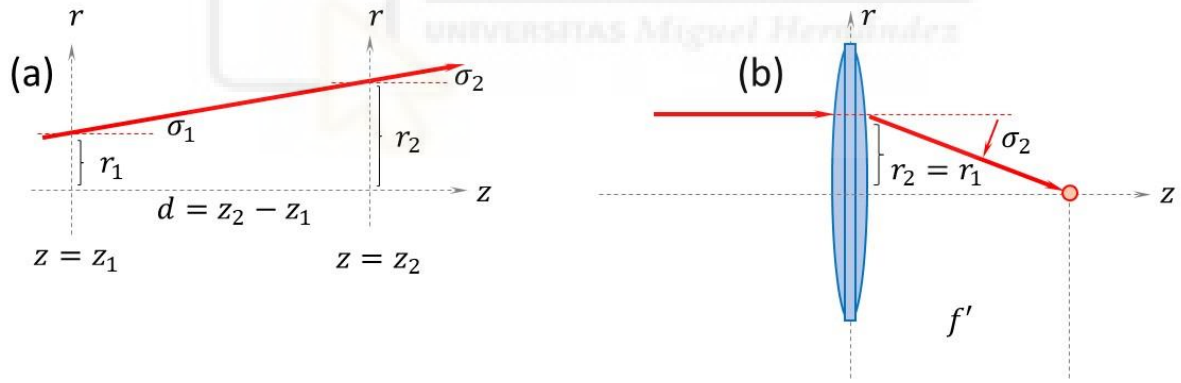

Fig. 3. 3. (a) Free propagation. (b) Lens action.

3.3. The 2f Fourier transform optical system

Figure 3.4(a) shows an especially relevant case, where light propagation is considered from the lens front focal plane to its back focal plane. This is known as the 2f system. The ray transfer matrix from plane F to plane F' involves the matrix product of the lens matrix between two free propagation matrices where the propagation distance is equal to the lens focal length, i.e.:

$$\mathbf{M}_{\mathcal{F}} = \begin{pmatrix} 1 & f' \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -1/f' & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & f' \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & f' \\ -1/f' & 0 \end{pmatrix}, \tag{3.12}$$

Figure 3.4(a) illustrates the system, showing how it converts a point source located at the front focal plane into a collimated output beam (green rays in Fig 3.4(a)), while an input collimated beam focuses on a point on the back focal plane (red rays in Fig 3.4(a)). Figures 3.4(b) illustrates when the input parallel rays have an angle σ_1 with respect to the optical axis, how the light spot is focused off-axis on the F' plane, with a location given by $x_2 = \sigma_1 f'$ where the paraxial small angle approximation is applied. The input ray passing through the centre of the lens, which is not deviated, provides this relation. Similarly, Fig. 3.4(c) illustrates if the point source in F is located off-axis a distance x_1, how the output is a set of tilted parallel rays with an angle $\sigma_2 = x_1/f'$. Therefore, the 2f system provides a one-to-one mapping between the rays' position and angle coordinates (x_1, σ_1) on plane F onto the angle and position coordinates (σ_2, x_2) of the rays at plane F'.

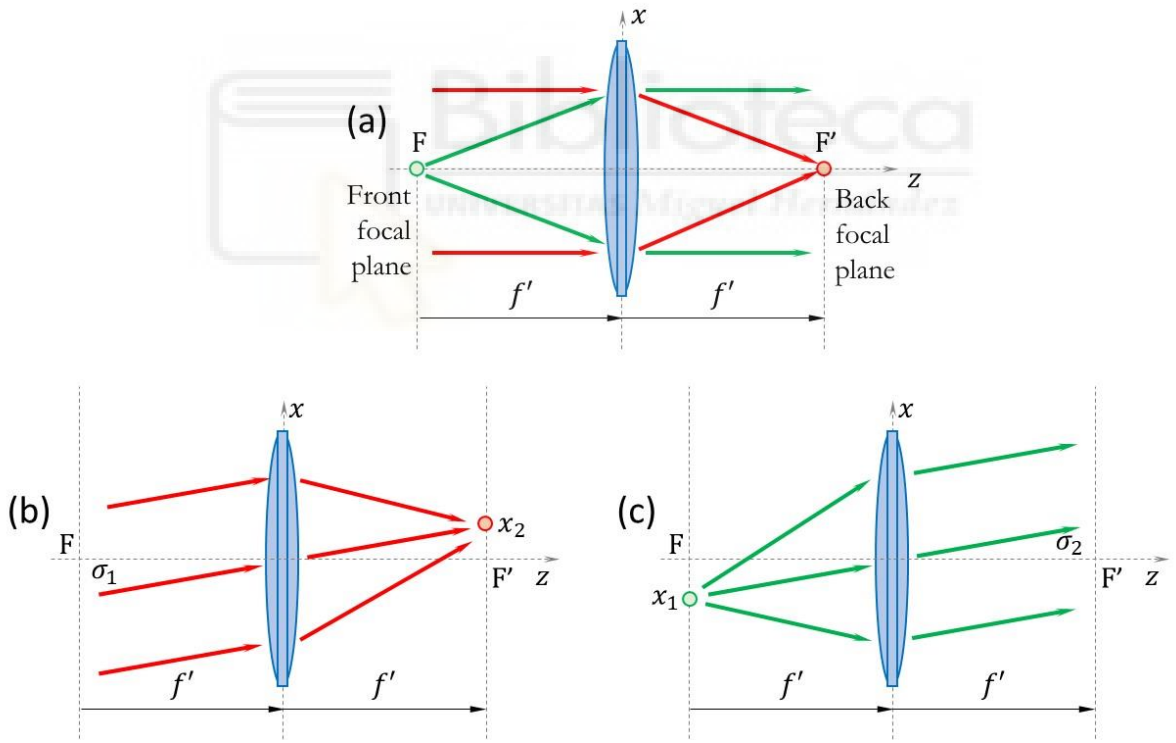

Fig. 3. 4. (a) 2f Fourier transform system. (b) Transformation of tilted parallel rays into an off axis point image. (c) Transformation of an off-axis light point source into tilted parallel rays.

This mapping is characteristic of the optical Fourier transform. Figure 3.5 illustrates the second case but viewed from the wave optics point of view, where the wavefronts are drawn. Figure 3.5(a) shows the 2f system transforming the input beam at plane F described by a wavefront

$g(x) = 1$ into a wavefront at plane F′ described by the wavefront in the form of a light impulse $g'(x) = \delta(x)$.

Figure 3.5(b) illustrates the situation when the input beam is tilted an angle σ. Since the distance between two wavefronts corresponds to a wavelength, (a 2π phase difference), the phase between points A and B in Fig. 3.5(b) inset must grow linearly with x, i.e., the wavefront at the front focal plane is given by $g(x) = e^{-i2\pi\sigma x/\lambda}$, which depends on the tilt angle σ. Since the beam at the back focal plane is a focused spot located at coordinate $x = \sigma f'$, it can be described as the wavefront $g'(x) = \delta(x - \sigma f')$.

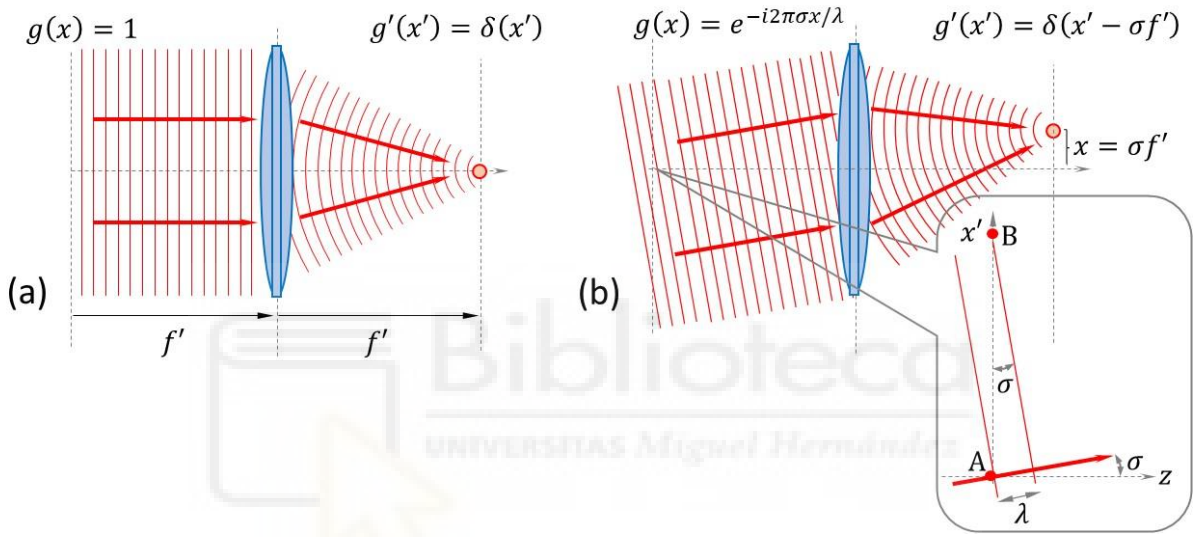

Fig. 3. 5. Wave optics view of the lens focusing of parallel rays (input plane wave).

Note this provides the Fourier transform relation described in Eqs. (2.27) in the previous chapter. Therefore, the following relation between wavefront $g'(x)$ at plane F' relative to wavefront $g(x)$ at plane F is found to be as $g'(x') = \mathcal{F}[g(x)]$, i.e., the wavefront at plane F' is the Fourier transform of the wavefront at plane F, where the spatial coordinates at F' are related to the spatial frequencies of the input wavefront as $x' = u\lambda f'$.

Note that the functions are two-dimensional, so this above relation is better described as

$$g'(\mathbf{r}') = \mathcal{F}_{2D}[g(\mathbf{r})]_{\mathbf{r}'=\lambda f'\mathbf{u}}, \tag{3.13}$$

where $\mathbf{r} = (x, y)$ are the transversal coordinates in the plane F, $\mathbf{r}' = (x', y')$ are the transversal coordinates in the plane F', and $\mathbf{u} = (u, v)$ are the spatial frequencies of the input wavefront.

One important property given by the $\mathbf{r}' = \lambda f'\mathbf{u}$ relation is that the **scale of the optical**

Fourier transform provided by the 2f system is proportional to the wavelength λ, i.e. a chromatic dispersion is produced in the diffracted field, and also to the lens focal length f'. Therefore, lenses with larger focal length can be used to obtain larger Fourier patterns.

The lens action viewed from the wave optics point of view transforms the collimated input beam into a converging spherical wavefront converging towards the back focal point. Therefore, the transmission of the lens must be a quadratic phase factor in Eq. (3.3) to yield this transformation, i.e.:

$$t_{\mathcal{L}}(\mathbf{r}) = e^{-i\frac{\pi r^2}{\lambda f'}}, \tag{3.14}$$

where f' is the lens focal length, $r = |\mathbf{r}| = \sqrt{x^2 + y^2}$ is the transversal radial coordinate.

The following equivalence relationships hold between the ray matrix operators of paraxial optics and the optical wavefront transformations:

	Ray matrix operator	Wavefront transformation	
Lens	$\mathbf{M}_{\mathcal{L}} = \begin{pmatrix} 1 & 0 \\ -1/f' & 1 \end{pmatrix}$	$g'(\mathbf{r}') = e^{-i\frac{\pi r^2}{\lambda f'}} g(\mathbf{r}),$	(3.15a)
Fourier transform	$\mathbf{M}_{\mathcal{F}} = \begin{pmatrix} 0 & f' \\ -1/f' & 0 \end{pmatrix}$	$g'(\mathbf{r}') = \left(\frac{1}{\lambda f'}\right)\mathcal{F}_{2D}[g(\mathbf{r})]_{\mathbf{r}'=\lambda f'\mathbf{u}}$	(3.15b)

3.4. Approximations to the optical Fourier transform

These previous relations allow to easily derive Fourier transform properties of systems applying simple matrix calculations. As practical examples let us consider two useful approximations.

Let us consider a diffractive optical element (DOE), defined by a transmission function $t(\mathbf{r})$ whose Fourier transform is $T(\mathbf{u}) = \mathcal{F}_{2D}[t(\mathbf{r})]$. If the DOE is placed on the F plane of a perfect 2f system like in Fig. 3.6(a), and it is illuminated with a perfectly collimated plane wave, the wavefront right after it is $g(\mathbf{r}) = t(\mathbf{r})$ and the wavefront at plane F' is given by $g'(\mathbf{r}') = \frac{1}{\lambda f'} T(\lambda f' \mathbf{u})$. The irradiance is proportional to the modulus squared of the wavefront function, i.e. a camera detector placed at the F' plane captures the function

$$I(\mathbf{r}') = \left(\frac{1}{\lambda f'}\right)^2 |T(\mathbf{r}' = \lambda f' \mathbf{u})|^2. \tag{3.16}$$

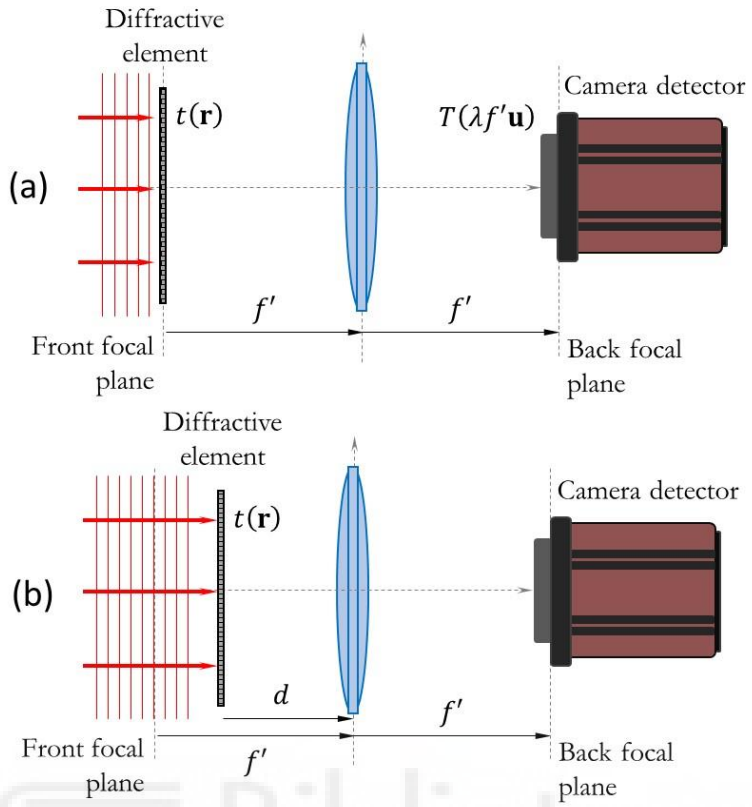

Fig. 3. 6. Optical setup to capture the irradiance of the Fourier transform of a diffractive element: (a) Exact 2f system. (b) System with additional quadratic phase.

3.4.1. Optical Fourier transform with quadratic phase

Figure 3.6(b) shows another case, where now the DOE is not placed exactly at F, but at a distance $d \neq f'$ from the lens. Using the ray matrix operators, the propagation from the DOE plane to the camera detector is described as

$$\mathbf{M}(d) = \begin{pmatrix} 1 & f' \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ -1/f' & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & d \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & f' \\ -1/f' & 1 - \frac{d}{f'} \end{pmatrix}. \tag{3.17}$$

This matrix can be further decomposed as

$$\mathbf{M}(d) = \begin{pmatrix} 1 & 0 \\ +\frac{1}{f'}\left(1 - \frac{d}{f'}\right) & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & f' \\ -\frac{1}{f'} & 0 \end{pmatrix} = \mathbf{M}_{\mathcal{L}} \cdot \mathbf{M}_{\mathcal{F}}. \tag{3.18}$$

This last equation reveals an important property of this system. The matrix on the right is the Fourier transform operator, which is then followed by a lens-like operator. Therefore, the output waveform is the Fourier transform of the input one but multiplied by a quadratic phase factor that vanishes only when $d = f'$, and it is denoted as follows:

$$g'(\mathbf{r}') = e^{+i\frac{\pi r^2}{\lambda f'}\left(1-\frac{d}{f'}\right)}\left(\frac{1}{\lambda f'}\right)\mathcal{F}_{2D}[g(\mathbf{r})]_{\mathbf{r}'=\lambda f'\mathbf{u}} \tag{3.19}$$

i.e., the optical Fourier transform is obtained at the lens focal plane F', but multiplied by the quadratic phase factor $\frac{\pi r^2}{\lambda f'}\left(1-\frac{d}{f'}\right)$ that vanishes if $d = f'$, corresponding to the exact 2f system.

In most cases, however, this phase is not relevant when only the irradiance of the beam is detected, like in Eq. (3.16), by placing a camera detector at the Fourier transform plane. Therefore, a system like in Fig. 3.6(b) measures an irradiance distribution as $I(\mathbf{r}') \propto |T(\mathbf{r}' = \lambda f'\mathbf{u})|^2$. This system has been used in parts of this Thesis.

3.4.2. The Fresnel and the Fraunhofer approximations

The Fresnel diffraction approximation, Eq. (3.6), and the far-field Fraunhofer approximation, Eq. (3.7), can be easily retrieved by decomposing the free propagation ray matrix as the following product:

$$\mathbf{M}_{\wp}(d) = \begin{pmatrix} 1 & d \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1/d & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & d \\ -1/d & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 1/d & 1 \end{pmatrix}. \tag{3.20}$$

This decomposition shows that the free space propagation can be viewed as three operators: two lens like operators and one Fourier transform operator in between. Using the correspondence in Eqs. (3.15) between the ray matrix operators and the wavefront mathematical transformations, the diffraction integral in Eq. (3.6) is directly retrieved as

$$g'(\mathbf{r}') = \left(\frac{1}{\lambda d}\right) e^{i\frac{\pi r'^2}{\lambda d}} \mathcal{F}_{2D}\left[g(\mathbf{r}) e^{i\frac{\pi r^2}{\lambda d}}\right]_{\mathbf{r}'=\lambda d\mathbf{u}}. \tag{3.21}$$

Finally, the far field Fraunhofer approximation is obtained when the propagation distance d is large enough so the quadratic phase factors in Eq. (3.21), corresponding to the two extreme matrices in Eq. (3.20), can be ignored. In other words, the free-space propagation matrix can be approximated only by the central Fourier transform type matrix, thus leading to the result in Eq. (3.7), where the output wavefront is related to the input one through

$$g'(\mathbf{r}') \cong \left(\frac{1}{\lambda d}\right) \mathcal{F}_{2D}[g(\mathbf{r})]_{\mathbf{r}'=\lambda d\mathbf{u}}. \tag{3.22}$$

Operating in the Fraunhofer approximation is useful because the scale of the Fourier transform can be changed simply by changing the distance d. This has been used in this Thesis, mainly to view the diffraction orders generated by diffraction gratings.

3.4.3. The convergent Fourier transform system

Finally, another configuration used in this Thesis is the convergent Fourier transform system shown in Fig. 3.7. In this case the diffractive element is illuminated with a converging beam. This input beam can be obtained simply by imaging a point source (for instance, a pinhole illuminated with a focused laser beam) with a positive lens, as indicated in the figure. The point of convergence is assumed at distance d from the DOE, where a diffuser screen can be placed to visualize the Fourier transform pattern, that can be imaged onto the detector by adding an objective lens to the camera.

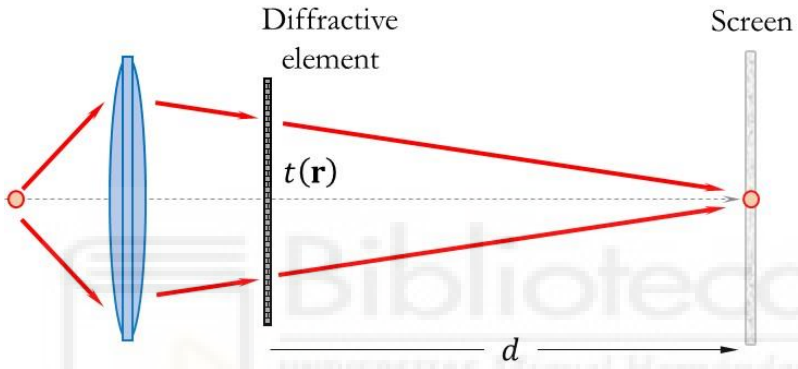

Fig. 3. 7. Converging Fourier transform system.

Having the DOE illuminated with the converging beam and then having a free propagation of distance d, can be regarded as only having the free propagation, but the DOE having a lens-like element right before it. Therefore, the converging illumination compensates for the first lens-like ray matrix in Eq. (3.20), so the ray matrix operator from the DOE plane to the screen plane can be written as

$$\mathbf{M}_C(d) = \begin{pmatrix} 1 & 0 \\ 1/d & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & d \\ -1/d & 0 \end{pmatrix}, \tag{3.23}$$

so the relation between the input and output wavefronts is given by

$$g'(\mathbf{r}') = \left(\frac{1}{\lambda d}\right) e^{i\frac{\pi r'^2}{\lambda d}} \mathcal{F}_{2D}[g(\mathbf{r})]_{\mathbf{r}'=\lambda d\mathbf{u}}. \tag{3.24}$$

Since the input beam illuminates the DOE, the wavefront right after it is given by $g(\mathbf{r}) = t(\mathbf{r})$ and a camera detector located a distance d captures an irradiance distribution

$$I(\mathbf{r}') = \left(\frac{1}{\lambda d}\right)^2 |T(\mathbf{r}' = \lambda d\mathbf{u})|^2. \tag{3.16}$$

This configuration has been used in the Thesis whenever the Fourier transform was very large to fit in the camera detector sensor. In that situation a diffuser screen was placed in the Fourier plane, and a camera with an added objective lens can be used to photograph the pattern. In addition, changing the location of the DOE relative to the lens changes the distance d and can be used to change the scale of the Fourier transform.

4. Phase-only spatial light modulators

Spatial Light Modulators (SLM) are pixelated optoelectronic devices that can modulate the intensity, the phase, or the state of polarization of the light beam. Their primary use is as micro-displays, to generate images to be displayed in projectors and head-mounted systems. However, thanks to their high resolution, with a large number of pixels and small pixel size, these devices have been widely used to provide laser beam shaping, aberration control, optical computing, etc through the display of diffractive optical elements (DOE), also known as holographic optical elements (HOE) when they are designed to produce complicated optical functions other than the classical diffraction gratings and diffractive lenses. In these applications, the parameter of interest to be modulated is the phase of the light beam. Phase-only SLMs are of interest because they can be used to shape light beams ideally with the maximum diffraction efficiency. Two major technologies compete in the market of SLM devices: 1) Digital Micromirror Devices (DMD), and 2) Liquid-Crystal Displays (LCD), illustrated in Fig. 4.1.

DMDs were introduced by the company Texas Instruments (TI) in the 80s. Sometimes referred to by the TI trademark, Digital Light Processing (DLP) systems, they consist of an array of individually addressed square micromirrors. Using micro electro-mechanical systems (MEMS)

technology, each micromirror can pivot about a hinge, tilting in opposite directions, giving two possible "on" and "off", states [Sch-2020]. These devices are, therefore, binary amplitude SLMs, although some coupled phase effects are observed due to the mirrors' tilt. Their use to display DOE is limited by the binary amplitude modulation (the maximum ideally diffraction efficiency is only about 20%). Since they do not produce phase modulation, the use of classical computer generated holography encoding techniques are required to display DOE. These methods typically use super-pixels (a set of various SLM pixels conforming one DOE pixel) to encode phases, thus reducing the effective available resolution. On the contrary, they present interesting properties like being wavelength and polarization insensitive (thus, they are not useful to modulate the state of polarization), and they can resist high powers. Their most valuable property is the capacity to be operated at remarkably high refresh rates, which can reach the scale of tens of kHz.

On the contrary, LCDs are based on the optical birefringence properties offered by liquid-crystal materials. Therefore, they can directly produce phase and polarization modulation at each pixel (no encoding is required), and the full spatial resolution can be exploited to efficiently display phase-only DOEs [Yan-2023]. They require working with polarized light and the phase modulation wavelength dispersion. Their most significant drawback is the slow refresh rate, limited to only tens of Hz with standard nematic LCDs, or up to about 200 Hz with ferroelectric liquid-crystal LCDs, these limited to a binary phase modulation.

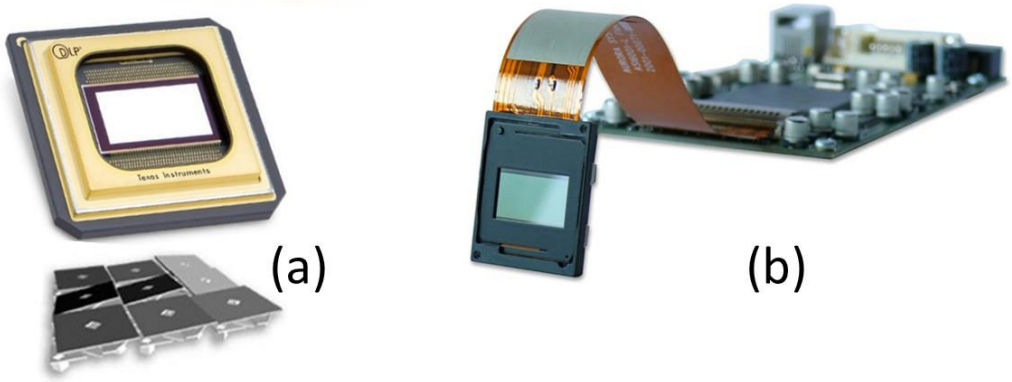

Fig. 4.1. Pictures of the two main SLM technologies: (a) a Digital Micromirror Device - DMD (image adapted from Texas Instruments), and (b) a Liquid-Crystal Display - LCD (image adapted from Holoeye Photonics).

4.1. Parallel-aligned liquid-crystal on silicon displays

Nowadays, the most common liquid crystal SLM configuration is known as the liquid-crystal on silicon (LCOS) display [Zha-2014]. These are reflective LCDs where a silicon backplane substrate

contains the electronic CMOS circuitry underneath the pixelated electrode array, while having in the outer surface a common ITO (Indium Titanium Oxide) transparent electrode (Fig. 4.2(a)). The LCOS structure allows significantly reducing the pixel size compared to transmission LCDs, giving the current high-resolution displays.

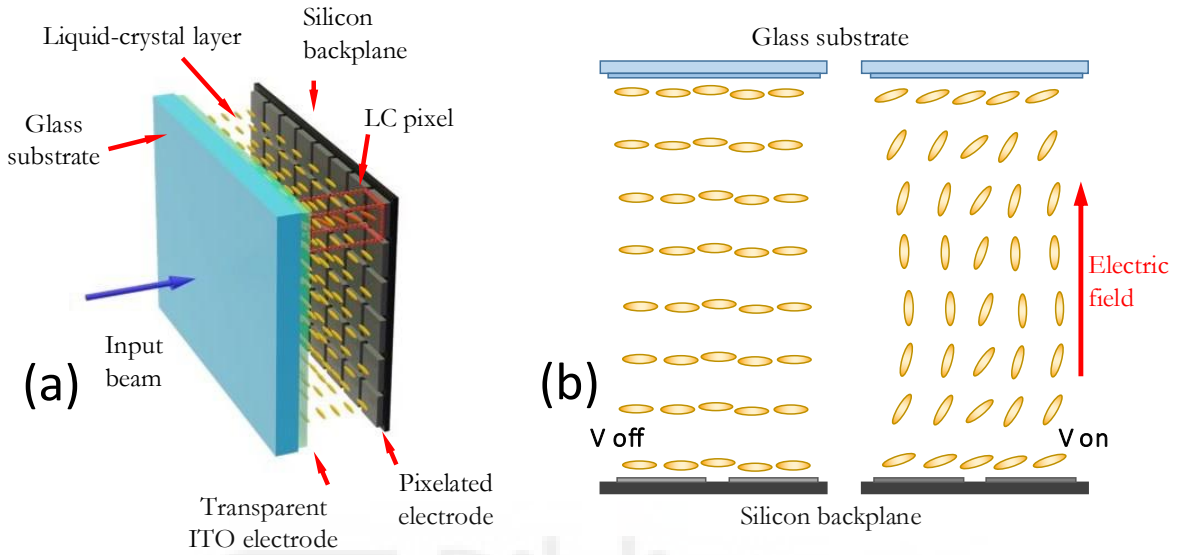

Fig. 4.2. (a) Scheme of the LCOS display. (b) Top view scheme of the parallel aligned LCOS configuration in the off and in the on state.

LCOS displays designed for phase-only operation typically use the nematic liquid-crystals in the parallel-aligned (PAL) configuration, as shown in Fig. 4.2(b). Nematic liquid-crystals have elongated molecules aligned along a preferential direction named the director axis. They are uniaxial birefringent materials, the director axis indicating the optical axis. They typically have positive birefringence, so the extraordinary refractive index is greater than the ordinary index ($n_e > n_o$). In the PAL configuration, the LC director axis is aligned homogeneously, parallel from the transparent electrode to the backplane. Therefore, when the PAL-LCOS device is off, the LC layer acts as an optical linear retarder having the slow axis oriented parallel to the director axis. When a voltage is applied to the electrodes, a transversal electric field is generated on the LC layer, causing the director axis to tilt towards the electric field. This causes reduction of the effective birefringence since the effective extraordinary refractive index tends to be equal to the ordinary index. As shown in the right part of Fig. 4.2(b), the director tilts more in the central part of the LC layer; LC molecules near the electrodes show higher resistance to tilt due to the alignment process. In addition, the electric field is not uniform due to the interpixel regions between backplane pixelated electrodes. This leads to a non-uniform orientation of the LC director in the interpixel

regions, causing the effect known as fringing.

Phase-only operation of nematic PAL LCOS displays is achieved using linearly polarized light oriented parallel to the LC director axis orientation in the off state. For this orientation, the input polarization is aligned with the extraordinary axis, and therefore the output beam does not change its polarization, but experiences a phase variation $\phi_e(V)$ proportional to the voltage dependant effective extraordinary index of refraction $n_e(V)$ as

$$\phi_e(V) = \frac{2\pi}{\lambda} n_e(V) t, \tag{4.1}$$

where λ is the wavelength of the light beam and t denotes the thickness of the LC layer (in a reflective device as LCOS, t is twice the physical thickness, to account for the beam's double pass). On the contrary, the perpendicular polarization component is aligned with the ordinary axis and experiences a constant phase $\phi_o = \frac{2\pi}{\lambda} n_o t$, independent of the value of applied voltage.

4.1.1. LCOS-SLMs commercial devices

Nowadays, there are several companies that supply SLMs with several distinctive characteristics. Along this Thesis we have worked with three different phase-only LCOS SLMs from different companies: (a) Hamamatsu model X10468-01, (b) Thorlabs model Exulus HD1, and (c) CAS Microstar model FSLM-2K39-P02. The three devices illustrated in Fig. 4.3.

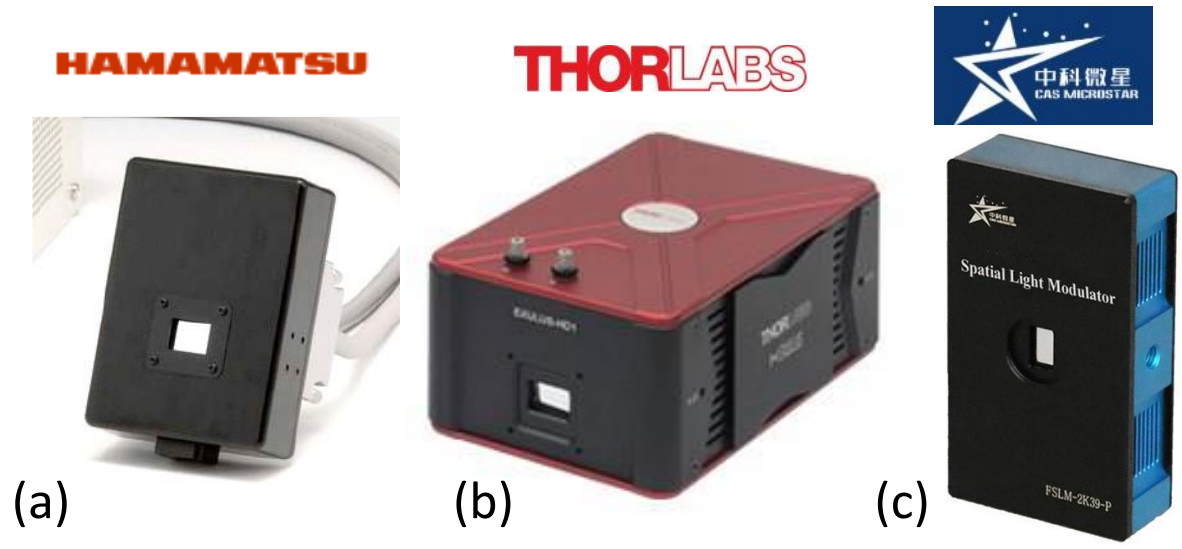

Fig. 4.3. Pictures of the three LCOS-SLMs used along the Thesis: (a), Hamamatsu model X10468-01 (b) Thorlabs model Exulus HD1 and (c) CAS Microstar model FSLM-2K39-P02.

For all three devices, the readout wavelength range covers the visible spectrum, and they work in 8 bits of grayscale, thus offering a total of 256 phase levels. Thorlabs and CAS Microstar devices have a full HD resolution with a small pixel pitch. The Hamamatsu SLM is the oldest device and has a relatively low resolution with a big pixel pitch, but it has the highest fill factor among the three. A similar panel active area occurs in Thorlabs and Hamamatsu devices, which are bigger than that of the CAS Microstar display. The Hamamatsu display has the electronics control in a separate box, while the Thorlabs and the CAS Microstar devices have a compact box including the panel and the electronics, thus being a much more convenient for small systems. Finally, only CAS Microstar has built-in functionality for synchronizing three different wavelength laser beams, which will be established in Chapter 9.

Table I provides the most relevant information of the three SLMs.

Table I. Basic parameters for three used SLMs.

	Hamamatsu X10468-01	Thorlabs Exulus HD1	CAS Microstar FSLM-2K39-P02
Wavelength range	400 - 700 nm	400 - 850 nm	420 - 650 nm
Grayscale	8 bits, 0-255	8 bits, 0-255	8 bits, 0-255
Spatial resolution	792 × 600 pixels	1920 × 1080 pixels	1920 × 1080 pixels
Pixel pitch	20 μm	6.4 μm	4.5 μm
Fill factor (a^2)	98%	93%	91.3%
Panel active area	15.8 × 12 mm^2	15.6 × 9.2 mm^2	9.64 × 4.86 mm^2

4.2. Retardance modulation calibration

One key aspect for the correct use of phase-only SLMs is an accurate calibration of the phase modulation properties. Many different interferometric, diffractive and polarimetric methods have been proposed. Here we use a simple standard classical method for determining the retardance of a linear retarder [Var-2013]. The retardance is defined as the difference between the extraordinary and the ordinary phases. In a nematic liquid crystal retarder (LCR), the retardance can be tuned since the extraordinary phase changes controlled with the applied voltage, i.e.:

$$\phi(V) = \phi_e(V) - \phi_o = \frac{2\pi}{\lambda}[n_e(V) - n_o]t. \tag{4.2}$$

Note that since nematic LCs typically have positive birefringence, this quantity is always positive and decreases as voltage is applied to the device.

The calibration procedure consists of measuring the intensity transmission of the LCR placed between parallel and crossed polarizers, oriented with 45° relative to the LC director axis. In this situation, the normalized output intensities are given by [Var-2013]:

$$i_{\parallel}(V) = \cos^2\left(\frac{\phi(V)}{2}\right), \qquad i_{\perp}(V) = \sin^2\left(\frac{\phi(V)}{2}\right), \tag{4.3}$$

where $i_{\parallel}$ and $i_{\perp}$ denote the transmission curves for parallel and for crossed polarizers, respectively. From these curves, the retardance ϕ can be retrieved as a function of the applied voltage as:

$$\phi(V) = 2\arctan\left(\sqrt{\frac{i_{\perp}(V)}{i_{\parallel}(V)}}\right), \tag{4.4}$$

where unwrapping of the $\phi(V)$ curve must be applied considering that the retardance is a function continuously decreasing as the voltage increases.

4.2.1. Polarization camera based automatic method of retardance calibration

The above-described procedure requires taking measurements versus the applied voltage for two orientations of the final polarizer. This is typically done by manually rotating the output polarizer. However, this forces us to take two measurements for each voltage, and measurements are affected by laser intensity fluctuations. This can be avoided by using a polarizing beam splitter and two detectors to make synchronous detection of the two polarization components. Here, we have developed a method using a polarization camera.

We have developed an automatic system that uses a single pixel liquid-crystal retarder from Thorlabs, model LCC1223-A. This is an LCR device with a clear aperture of 20 mm, having anti-reflection (AR) coatings for the wavelength range 350-700 nm. It is a full wave (has more than 2π retardance variation) uncompensated (does not reach zero retardance) variable retarder (Fig. 4.4(a)). It is controlled with the Thorlabs K-Cube™ controller model KLC101 (Fig. 4.4(b)), which allows applying a square-wave output voltage with adjustable amplitude 0 to ±25 VAC and adjustable frequency from 500 Hz to 10 kHz with 50% duty cycle. It has Python SDKs (Software development kits), for Python hardware programming.

The detector is a polarization camera Thorlabs Kiralux CS505MUP, which is a monochrome camera having 2448×2048 square pixels with 3.45 μm size. The image sensor incorporates a linear micro-polarizer array, integrated between a microlens array and the photodiodes. The polarizer

array is composed of wire grid polarizers fabricated directly on the sensor and arranged in a mosaic pattern. Each pixel has a polarizer which transmits one of the horizontal (H), vertical (V), diagonal (D), and antidiagonal (A) linear polarization components, creating a 4×4 square superpixel (Fig. 4.4(c)). Hence, it is possible to measure the state of polarization (SOP) of the light hitting the camera in terms of these four linear polarizations in one shot. We are able to retrieve the intensities for the parallel and crossed response on the polarization camera with respect to the polarization of P2 (in Fig. 4.5). The polarization camera is available for Python hardware programming as well, so it can be synchronized with the LCR device. As a result, we can take one measurement for one voltage in 3.67s, including both parallel and crossed configurations.

Fig. 4.4. Pictures of (a) Liquid-crystal retarder LCC1223-A, (b) K-CubeTM controller KLC101 and (c) Polarization camera Kiralux CS505MUP.

Figure 4.5 shows a scheme of the automatic calibration experimental setup we built for the LCR calibration procedure, where we developed a Python project to synchronize sending voltages to LCR and capturing images on the polarization camera. This is a transmissive setup due to the transmissive property of the LCR. A semiconductor green laser of wavelength 532 nm is used as the light source. An attenuator located at the laser output is used to adjust the intensity level. Then

the beam is spatially filtered with a microscope objective and a pinhole and collimated with lens (L1). Then we add a system composed of a first polarizer (P1), a quarter-wave plate (QWP), and a second polarizer (P2). Polarizer P1 ensures that the laser beam is linearly polarized. Then the QWP is oriented to generate circularly polarized light, so polarizer P2 can be oriented in any arbitrary orientation without changing the intensity impinging on the LCR device. The LCR was mounted onto a rotatable mount, so the optical axis was selected oriented at 0° in the laboratory coordinate. P2 is oriented at −45°. A second lens (L2) is used to control the focus beam and project it onto the camera, and the four polarization panels given by the camera are captured. During this process, the LCR is addressed with a bipolar signal of frequency 1kHz and voltage values from 0V to 5V.

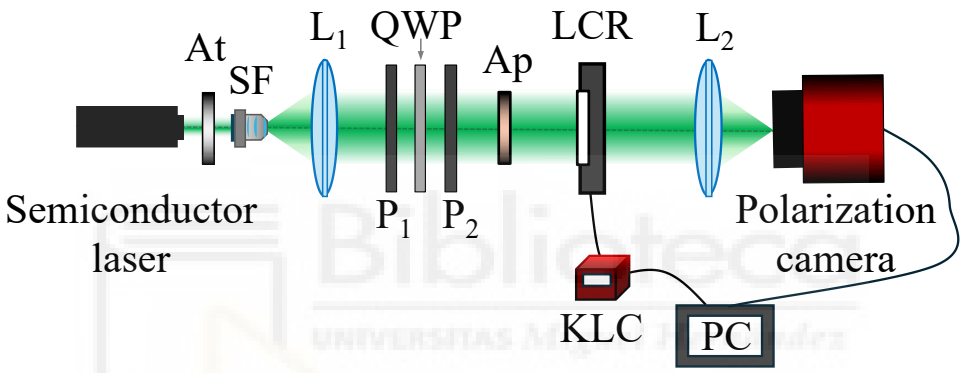

Fig. 4.5. Experimental setup for automatic calibration. P: linear polarizer; L: convergent lens; QWP: quarter wave plate; SF: spatial filter; Ap: circular aperture; At: variable attenuator; LCR: liquid crystal retarder; KLC: LCR controller.

Figures 4.6(a-c) show the experimental captures on the polarization camera for 0V, 1V, and 3V conditions, respectively. In each capture, there are four panels, and in each panel, corresponding to the vertical (V), diagonal (D), anti-diagonal (A) and horizontal (H) linear polarization components. Thus, the D and A components give the crossed and parallel intensities required to apply Eqs. (4.3-4.4). In each case we measure the intensities by averaging the sum of the values of the pixels in a circular shape around the light spot. Figure 4.6(d) shows the experimental normalized crossed $i_{\perp}$ and parallel $i_{\parallel}$ intensities versus voltage from 0V to 5V, measured in steps of 0.05 volts. The curves for parallel and crossed intensities follow the expected oscillating behaviour, showing almost two complete oscillations (indicating a retardance variation slightly less than 4π radians). The oscillations are faster for voltages between 0.8 and 2 volts, indicating the typical situation where the retardance variation is non-linear with respect to the applied voltage. Three voltages circled in green have consistent intensity changes in panels D and

A compared to the captures in Fig. 4.6(a-c). The retardance of LCR is shown in Fig. 4.6(e), which is retrieved according to Eq. (4.4). The maximum retardance occurs for zero voltage, and for the green laser of wavelength 532 nm, is of 3.43π radians. There is no variation until a threshold voltage value of about 0.6 volts. Then, there is a rapid retardance reduction between 0.8 and 2 volts. For higher voltage values, the retardance continues to decrease slowly but does not reach zero retardance (due to the LC molecules near the electrodes, which are not able to tilt).

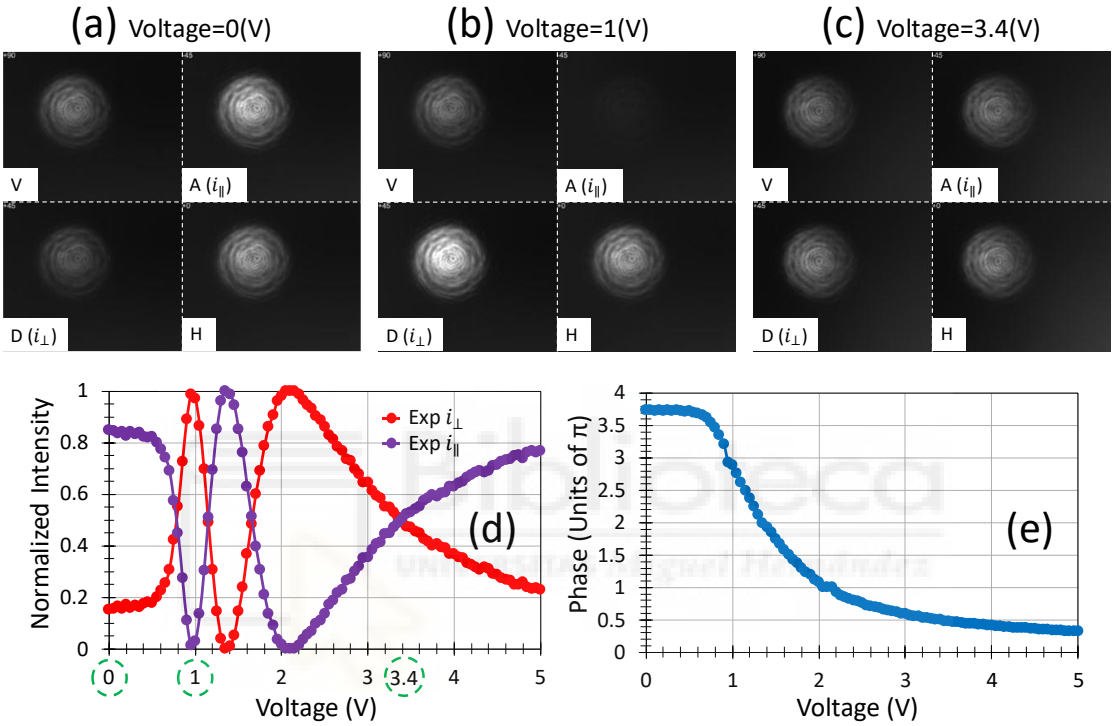

Fig. 4.6. Experimental captures of polarization camera with 4 panels for (a) 0V, (b) 1V, and (c) 3.4V. (d) Experimental normalized intensity for crossed and parallel configuration versus voltage. (e) Retardance of LCR versus voltage.

4.3. SLM phase modulation calibration

As mentioned, phase-only LCOS SLMs are pixelated linear retarders. Therefore, the same standard procedure described above can be applied to calibrate the retardance variations. The SLM must be placed between parallel and crossed polarizers, oriented with 45º relative to the LC director axis. Here, the parameter of control of the modulation is the gray level addressed from a computer. Since the three SLMs described in Section 4.1 are 8-bit devices, the grayscale ranges from 0 to 255, and the electronics of the device convert them into the corresponding voltage values.

The application of these SLMs to display phase-only DOE depends on the phase depth modulation (or simply the phase modulation). This is obtained as the retardance variation with

respect to a reference retardance value, which here we select as the value for gray level 255, i.e.:

$$\varphi(g) = \phi(g) - \phi(g = 255). \tag{4.5}$$

The implementation of DOE with maximum diffraction efficiency requires that the maximum phase modulation reaches 2π radians, and a linear variation of the phase with gray level.

Figure 4.7 shows the experimental setup employed to calibrate the SLMs phase modulation. The display is set between two polarizers (P2 and P3). We used a He-Ne laser of wavelength 632.8 nm for Thorlabs Exulus and Hamamatsu SLMs, and a semiconductor laser of wavelength 662 nm for the CAS Microstar FSLM-2K39-P02 (this device works in a mode synchronized with a diode laser source, as will be discussed in Chapter 9). The input intensity is adjusted by an attenuator (At). Like in the previous system, the laser beam was expanded through a spatial filter (SF) and collimated by a lens (L1). After that, circularly polarized light is generated by combining a linear polarizer P1 and a quarter wave plate (QWP) with its neutral axis at 45° relative to P1's transmission axis. As a result, the intensity after linear polarizer P2 remains constant regardless of its orientation, which allows us to illuminate the LCOS-SLM with different polarization orientations without changing the light intensity on the display. Finally, the beam reflected from the SLM is focused by a lens (L2) on a camera. The polarization camera described in the previous section could be used, but in some early works we used a standard camera (Basler, with 1390×1038 pixels of 4.65 μm pixel size) with another third polarizer (P3) that captures the diffraction pattern. A circular aperture (Ap) is used to adjust the beam diameter to the SLM area.

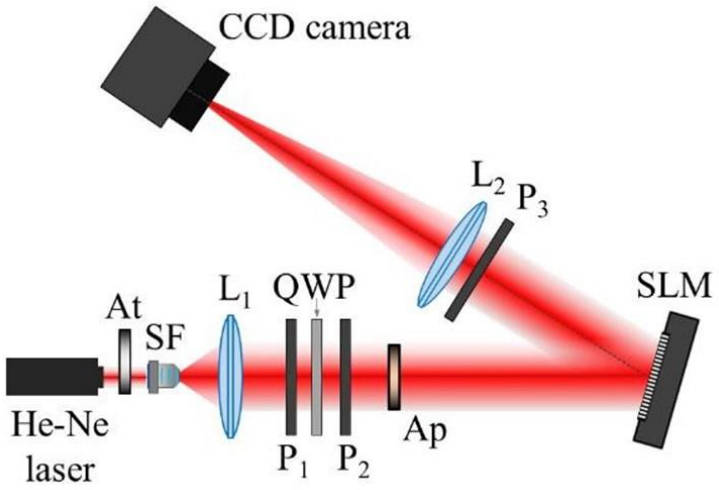

Fig. 4.7. Experimental setup for standard calibration [Gao-2024a]. P: linear polarizer; L: convergent lens; QWP: quarter wave plate; SF: spatial filter; Ap: circular aperture; At: variable attenuator.

Figure 4.8 shows the experimental normalized transmission (red dots for $i_{\perp}(g)$, purple dots for $i_{\parallel}(g)$) compared to the expected curves (red curve for $i_{\perp}(g)$, purple curve for $i_{\parallel}(g)$) and the phase modulation (blue dots) for three SLMs. For the same He-Ne laser of wavelength 632.8 nm, the phase modulations for Hamamatsu SLM (Fig. 4.8(a)) and Thorlabs SLM (Fig. 4.8(b)) are 2.34π and 1.96π, respectively, and CAS Microstar SLM (Fig. 4.8(c)) has a 1.78π phase modulation with an input of semiconductor laser of wavelength 662 nm.

Note that the three SLMs show a linear phase modulation response versus addressed gray level. Despite LC retardance does not respond linearly to the applied voltage (as illustrated in the case of the LCR in Fig. 4.6(e)). These devices are prepared so the gray level to voltage conversion compensates the voltage to phase modulation relation, so finally there is a linear relation of the phase modulation versus gray level.

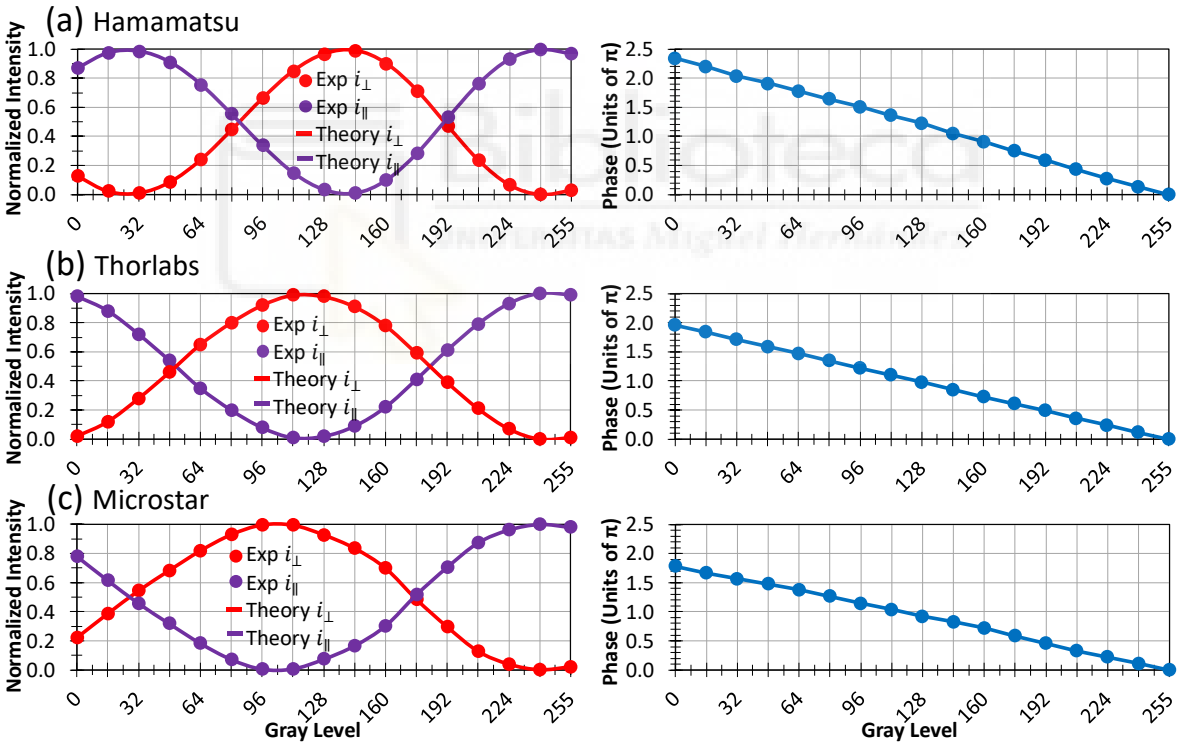

Fig. 4.8. LCOS-SLM's normalized transmission for parallel and crossed polarizers (theory in solid curves, experiment in dots) and phase modulation (blue dots) versus the addressed grey level: (a) Hamamatsu. (b) Thorlabs. (c) CAS Microstar.

4.3.1. Pixel crosstalk by fringing field effect

Phase-only LCOS-SLMS are typically affected by different deletory effects that affect their performance as pure tunable pixelated retarders. The most relevant are: 1) the deformation of the

backplane, which introduces aberration; 2) multiple internal beam reflections, which cause Fabry--Perot type interferences; 3) flicker or temporal phase fluctuations caused by the digital addressing of the devices, which causes a reduction of the diffraction efficiency; and 4) pixel crosstalk caused by the fringing field effect. This last is the more severe problem in modern high-resolution displays, since the small pixel size makes such devices more affected by this problem.

This is a crosstalk between neighbouring pixels, which leads to phase profile smoothing and a reduction of the phase difference, especially when the voltage difference between pixels is high [Lin-2013]. This effect occurs because the external electrode is common and there is no physical barrier between the LC layer for different pixels. Therefore, when two neighbouring pixels are addressed with different voltages, the electric field generated in the LC layer does not show an abrupt transition. Instead, it shows a distribution which results in a smooth variation of the LC director axis distribution.

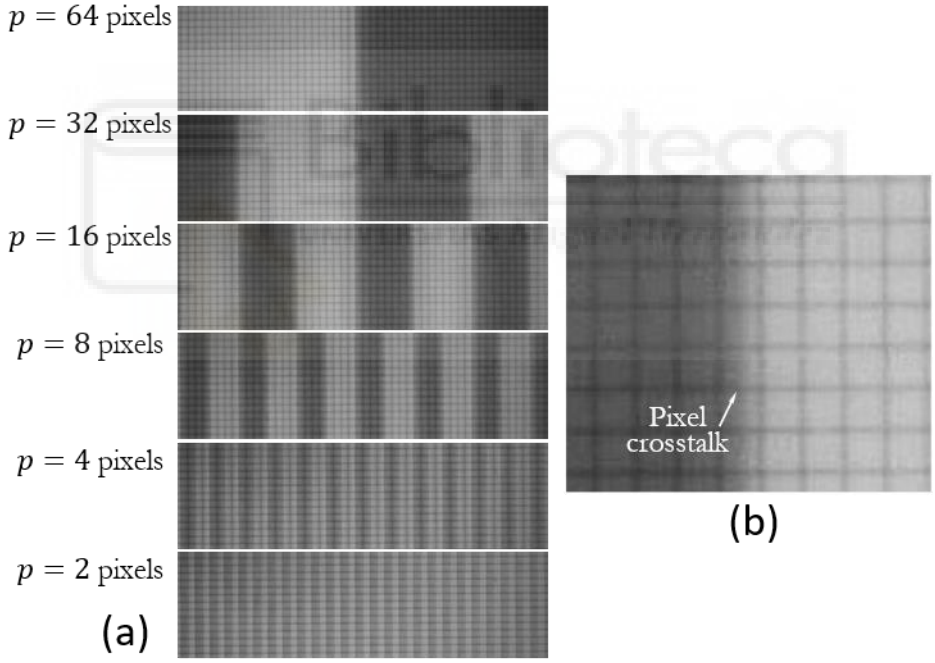

Fig. 4.9. Images of the Thorlabs Exulus-HD1 LCOS-SLM viewed through a polarizing microscope, showing the pixels of the device. (a) The SLM is addressed with vertical bar images with different periods (p). (b) Magnified image of the transition between the two regions, showing the pixel crosstalk caused by the fringing field effect.

Figure 4.9 shows this effect, where the Thorlabs Exulus-HD1 SLM screen was viewed through a polarizing microscope with a 20X objective. The pixels are clearly visible. Although some LCOS devices may have a reflective interpixel gap, the picture shows the usual case where the interpixel gap has no reflective coating and thus absorbs light. The SLM is addressed with an

image of vertical stripes consisting of two different gray levels (a Ronchi grating). These vertical stripes are viewed as bright and dark areas in the polarizing microscope.

As noted in the magnified image in Fig. 4.9(b), the transition from one region to the other is not sharp, which is a typical indication of the pixel crosstalk caused by the fringing field effect affecting the LCOS-SLM [Lin-2013]. Let us highlight that the calibration technique used in Fig. 4.8 applies images of uniform gray level. Therefore, no pixel crosstalk is produced, and the phase calibration is accurate. This is not the case of other phase calibration techniques relying on diffraction gratings, which might lead to errors if the SLM is affected by this pixel crosstalk [Mor-2021], as it will be shown in the next chapter. In fact, the images in Fig. 4.9(a) clearly show a contract reduction when $p = 4$ and $p = 2$ pixels, indicating that with these low values, the fringing field effect prevents the LC from being oriented properly to produce the expected transition.

Python Code

We developed a Python project to synchronize the hardware: an LCR with KLC and a polarization camera, for automating the whole calibration process. Figure 4.10 shows the directory of the project, the folder "Native_64_lib" and the file "KLCCommandLib_x64.dll" are necessary libraries for the camera and KLC, respectively. The Python file "windows_setup.py" is for adding the folder containing the DLLs to the current project path. The files "initialKLCfor1.py" and "closeKLCfor1.py" are designed to turn on and off the KLC, which are executed once. This prevents activating and deactivating the device repeatedly during continuous measurements. The file "OneKLCwithCamera.py" is the console of this project for adjusting the parameters of devices and taking measurements. In the end, we use the file "BatchReadImage.py" to process the saved images for each voltage and save the results in an Excel file.

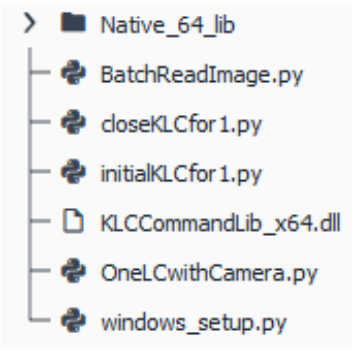

Fig. 4. 10. Directory of the Python project for hardware control of LCR with KLC and polarization camera.

Script 1. windows_setup

```
import os
import sys

def configure_path():
    is_64bits = sys.maxsize > 2**32
    relative_path_to_dlls = os.sep
    if is_64bits:
        relative_path_to_dlls += 'Native_64_lib'
    else:
        relative_path_to_dlls += '32_lib'

    absolute_path_to_file_directory = os.path.dirname(os.path.abspath(__file__))

    absolute_path_to_dlls = os.path.abspath(absolute_path_to_file_directory + os.sep +
relative_path_to_dlls)

    os.environ['PATH'] = absolute_path_to_dlls + os.pathsep + os.environ['PATH']

    try:
        os.add_dll_directory(absolute_path_to_dlls)
    except AttributeError:
        pass
```

Script 2. OneKLCwithCamera

```
try:
    from windows_setup import configure_path
    configure_path()
except ImportError:
    configure_path = None

import numpy as np
from PIL import Image, ImageDraw, ImageFont
from thorlabs_tsi_sdk.tl_camera import TLCameraSDK
from thorlabs_tsi_sdk.tl_camera_enums import SENSOR_TYPE
from thorlabs_tsi_sdk.tl_polarization_processor import PolarizationProcessorSDK
import initialKLCfor1
import closeKLCfor1
from KLCCommandLib import *
import time
import asyncio
import nest_asyncio

nest_asyncio.apply()
start_time = time.time()
handle=initialKLCfor1.initialKLC(1) # Call function “initialKLC(1)” to turn on KLC, and the serial
number is returned to “handle”.
print("KLC´s handle:",handle)
vols2=np.arange(0,5,0.1) # Set continuous input voltages from 0V to 5V with 0.1V of one step.
# Config the parameter values of KLC.
f=1000 # Set the frequency of the signal to enable one channel.
mode=2 # The mode of sending voltages to LCR. 1. continuous; 2. cycles.
cyclenumber=1 # If “cycles” is selected, the number of cycles (1~ 2147483648) can be changed.
delay=1000  # The delay between two voltages, 1~ 2147483648[ms].
precycle_rest=0 # The delay time before the cycle starts 0~ 2147483648[ms].
hdl=handle
# The function controls KLC to send voltages to LCR.
async def sendVoltage(vols):
    l=len(vols)
    volarr =  (c_float * len(vols))(*vols)
    print("what is happening")
    # Set the target input voltages to the available KLC by indexing handle “hdl”.
    if(klcSetOutputLUT(hdl, volarr, l)<0):
        print("klcSetOutputLUT failed")
    # Set the predefined frequency to one channel of the available KLC.
    klcSetFrequency1(hdl, f)
```

```
    # Set all the parameters that are defined before.
    if(klcSetOutputLUTParams(hdl, mode, cyclenumber, delay, precycle_rest)<0):
        print("klcSetOutputLUTParams failed")

    print("----------------Current output voltage is:",vols)
    # Create a task for the camera, and it calls and executes “camera()” while KLC is working, which
is the core of synchronization.
    task=asyncio.create_task(camera())

    if(klcStartLUTOutput(hdl)<0):
        print("klcStartLUTOutput failed")
    # KLC and camera stop working at the same time and prepare for the next voltage.
    await task
# Count execution time for naming the captures.
def my_count(s, i=[0]):
    i[0] += 1
    return i[0]
# This function turns on/off the polarization camera and makes it save the captures.
async def camera():
    # Call built-in models/packages to detect the available camera.
    with TLCameraSDK() as camera_sdk, PolarizationProcessorSDK() as polarization_sdk:
        available_cameras = camera_sdk.discover_available_cameras()
        if len(available_cameras) < 1:
            raise ValueError("no cameras detected")
        # Open camera by camera handle “available_cameras[0]”.
        with camera_sdk.open_camera(available_cameras[0]) as camera:
            camera.frames_per_trigger_zero_for_unlimited = 1  # Set the number of frames for one
capture, 0 means start the camera in continuous mode.
            camera.image_poll_timeout_ms = 1000  # This time is used for retrieving image on camera to
frame, the unit is ms.
            camera.arm(2)
            # Set the width and height for a frame.
            image_width = camera.image_width_pixels
            image_height = camera.image_height_pixels

            camera.issue_software_trigger()
            # Receive the content for each frame.
            frame = camera.get_pending_frame_or_null()
            if frame is not None:
                print("frame received!")
            else:
                raise ValueError("No frame arrived within the timeout!")

            camera.disarm()
            # Call the built-in polarization model depending on the camera´s colour mode. We are
scaling from 12 bits (property of the camera) to 8 bits (compatible with the PIL Python package) in
this project.
            if camera.camera_sensor_type is not SENSOR_TYPE.MONOCHROME_POLARIZED:
                raise ValueError("Polarization processing should only be done with polarized cameras")

            camera_polar_phase = camera.polar_phase
            camera_bit_depth = camera.bit_depth
            print(f"camera_polar_phase at top left:{camera_polar_phase}, and the bit
depth:{camera_bit_depth}")
            # Open built-in polarization processor.
            with polarization_sdk.create_polarization_processor() as polarization_processor:
                unprocessed_image = frame.image_buffer.reshape(image_height, image_width)  # This is
the raw image data in each frame.
                unprocessed_image = unprocessed_image >> camera_bit_depth - 8  # Create an empty 2D
array “output_quadview” for the quadview having 4 panels for 4 different states of polarization.
                output_quadview = np.zeros(shape=(image_height, image_width))  # initialize array for
QuadView data
                #  Rearrange the intensity distribution of the original image data and assign the new
distribution to “output_quadview”. Put pixels (intensity) with the same transmission axis together.
                output_quadview[0:int(image_height / 2), 0:int(image_width / 2)] = \
                    unprocessed_image[0::2, 0::2]  # (0,0): top left rotation == camera_polar_phase
                output_quadview[0:int(image_height / 2), int(image_width / 2):image_width] = \
                    unprocessed_image[0::2, 1::2]  # (0,1): top right rotation
                output_quadview[int(image_height / 2):image_height, 0:int(image_width / 2)] = \
                    unprocessed_image[1::2, 0::2]  # (1,0): bottom left rotation
                output_quadview[int(image_height / 2):image_height, int(image_width / 2):image_width]
= \  unprocessed_image[1::2, 1::2]  # (1,1): bottom right rotation
```

```
                # Display QuadView with the labels of the polarization state in each panel.
                quadview_image = Image.fromarray(output_quadview)
                img = quadview_image.convert('RGB')
                draw=ImageDraw.Draw(img)
                text1="+0"
                text2="+90"
                text3="+45"
                text4="-45"
                font=ImageFont.truetype("arial.ttf",50)
                draw.text((0,0), text2, (255,255,255),font=font)
                draw.text((1224,1024), text1, (255,255,255),font=font)
                draw.text((0,1024), text3, (255,255,255),font=font)
                draw.text((1224,0), text4, (255,255,255),font=font)
                k=my_count(1)
                img.save(f"vol_{k}.tif")
# Send the voltage sequence one by one to KLC, and then the camera can save the capture for every
voltage.
async def main():
    length = len(vols2)
    for count in range(length):
        vols1 = [vols2[count]]
        await sendVoltage(vols1)
# This script starts from the main function.
if __name__ == "__main__":
    asyncio.run(main())
# Turn off the open KLC.
print("The measurement is done.")
closeKLCfor1.closeKLC(handle)

end_time = time.time()
execution_time = end_time - start_time
print("Execution time:", execution_time, "seconds")
```

Script 3. initialKLCfor1

```
try:
    from KLCCommandLib import *
except OSError as ex:
    print("Warning:",ex)

def initialKLC(a):
    print("*** Initial KLC ***")
    if a:
        try:
            # Call the built-in function to detect the device number of the available KLC.
            devs = klcListDevices()
            if(len(devs)<=0):
               print('There is no devices connected')
               exit()
            sn1 = devs[0][0] # Retrieve the serial number of detected devices.
            print("connect ",sn1)
            hdl1 = klcOpen(sn1, 115200, 3)  # Open a KLC by serial number, and a handle number is
returned. This handle is passed to a specific variable when this function is called, which is the ID
for a KLC.

            if(hdl1<0):
                print("open ", sn1, " failed")
                exit()
            if(klcIsOpen(sn1) == 0):
                print("klcIsOpen failed")
                klcClose(hdl1)
                exit()

        except Exception as ex:
            print("Warning:", ex)

  return hdl1
```

Script 4. closeKLCfor1

```
try:
    from KLCCommandLib import *
    import time
    import initialKLC
except OSError as ex:
    print("Warning:",ex)
# Turn off KLC by passing its ID (handle) to this function.
def closeKLC(handle1):
    try:
        hdl1=handle1
        klcStopLUTOutput(hdl1)
        print("stop LUT")
        klcClose(hdl1)
        print("*** start to end ***")
    except Exception as ex:
        print("Warning:", ex)
    print("*** End ***")
```

Script 5. BatchReadImage

```
import cv2
import os
import numpy as np
import matplotlib.pyplot as plt
from openpyxl import Workbook, load_workbook
import time
# Read the saved images to retrieve the intensity, which is shown in Figs. 4.6(a-c).
os.chdir("C:/Users/Laboratorio/AutoMeasureLCwithPolorizationCamera/monitor/monitor_mor")
excelName=input("Please input a filename for saving data:")
start_time = time.time()
filename=0
maxinumber=50
while filename<maxinumber+1:

    im=cv2.imread(str(filename)+".tif")
    im1=im[:,:,1]
    print(filename)
    im45withlabel=im1[1150:1850,250:1050]
    im_45withlabel=im1[100:800,1470:2170]
    # Crop the centre part of the target panels into a square area.
    im45=im45withlabel[350:500,230:400]
    im_45=im_45withlabel[350:500,230:400]
    plt.subplot(2, 2, 1)
    plt.imshow(im45withlabel)
    plt.title('+45')
    plt.subplot(2, 2, 2)
    plt.imshow(im_45withlabel)
    plt.title('-45')

    plt.subplot(2, 2, 3)
    plt.imshow(im45)
    plt.title('+45')
    plt.subplot(2, 2, 4)
    plt.imshow(im_45)
    plt.title('-45')

    def calIntensity(row, col,x,y,r,pol):
        count=0
        sumintensity=0
        for i in range(rows):
            for j in range(cols):
                distance_squared = (i - x0) ** 2 + (j - y0) ** 2
                if distance_squared < r ** 2:
                    if pol=="45":
                        sumintensity += im45[i, j]
                    else:
                        sumintensity += im_45[i, j]
```

```
                count+=1
                return sumintensity/count

    print(f"The sum of the intensity id {sumintensity} and the number of the pixel is {count} ")
    print(f"Then the average of the intensity is {sumintensity/count:.2f}")
rows, cols = np.shape(im45)
x0,y0=rows/2,cols/2
r=20
# Calculate crossed intensity
cal45=calIntensity(rows, cols, x0, y0, r,"45")
rows_45, cols_45 = np.shape(im_45)
x0_45,y0_45=rows_45/2,cols_45/2
# Calculate parallel intensity.
cal_45=calIntensity(rows_45, cols_45, x0_45, y0_45, r,"-45")
# Create an empty Excel file to save retrieved data. The results are shown in Figs. 4.6(d-e).
if os.path.exists(excelName+ ".xlsx"):
    wb=load_workbook(excelName+r".xlsx")
    ws=wb.active
    ws['D1'] = "I (45)-BG"
    ws['E1'] = "I (-45)-BG"
    ws['F1'] = "Sum"
    ws['H1'] = "I_c"
    ws['I1'] = "I_p"
    ws['J1'] = "SQRT"
    ws['K1'] = "2*ATAN"
    ws['L1'] = "phase(π)"
    ws['M1'] = "unwrapped phase(π)"
    ws.cell(filename+2,2,cal45)
    ws.cell(filename+2,3,cal_45)
    i=2
    # Set the formula in the specific cells, which are following Eqs. (4.3-4.4).
    while i<maxinumber+3:
        ws[f'D{i}'] = f"=B{i}-MIN(B:B)"
        ws[f'E{i}'] = f"=C{i}-MIN(C:C)"
        ws[f'F{i}'] = f"=E{i}+D{i}"
        ws[f'H{i}'] = f"=D{i}/F{i}"
        ws[f'I{i}'] = f"=E{i}/F{i}"
        ws[f'J{i}'] = f"=SQRT(H{i}/I{i})"
        ws[f'K{i}'] = f"=2*ATAN(J{i})"
        ws[f'L{i}'] = f"=K{i}/PI()"
        i+=1
    wb.save(excelName+r".xlsx")
else:
    wb=Workbook()
    ws=wb.active
    ws['D1'] = "I (45)-BG"
    ws['E1'] = "I (-45)-BG"
    ws['F1'] = "Sum"
    ws['H1'] = "I_c"
    ws['I1'] = "I_p"
    ws['J1'] = "SQRT"
    ws['K1'] = "2*ATAN"
    ws['L1'] = "phase(π)"
    ws['M1'] = "unwrapped phase(π)"
    ws.cell(1,1,"ImageName")
    ws.cell(1,2,"pol=+45")
    ws.cell(1,3,"pol=-45")
    ws.cell(1,7,"Voltage")
    vols=np.arange(0,5.1,0.1)
    for k  in range(len(vols)):
          ws.cell(k+2, 1, vols[k]*20)
          ws.cell(k+2, 7, vols[k])
    ws.cell(filename+2,2,cal45)
    ws.cell(filename+2,3,cal_45)
    i=2
    while i<maxinumber+3:
        ws[f'D{i}'] = f"=B{i}-MIN(B:B)"
        ws[f'E{i}'] = f"=C{i}-MIN(C:C)"
        ws[f'F{i}'] = f"=E{i}+D{i}"
        ws[f'H{i}'] = f"=D{i}/F{i}"
        ws[f'I{i}'] = f"=E{i}/F{i}"
        ws[f'J{i}'] = f"=SQRT(H{i}/I{i})"
```

```
            ws[f'K{i}'] = f"=2*ATAN(J{i})"
            ws[f'L{i}'] = f"=K{i}/PI()"
            i+=1
        wb.save(excelName+r".xlsx")
    filename+=1
end_time = time.time()
execution_time = end_time - start_time
print("Execution time:", execution_time, "seconds")
```

5. Diffraction gratings: Fourier analysis and experimental realization

This chapter explains the characterization of several typical diffraction gratings theoretically and their realizations on SLMs. The condition for generating a triplicator grating, a diffraction grating generating three equally intense 0th and ±1st diffraction orders, will be highlighted for different gratings, and their theoretical diffraction efficiencies are also verified in experiments. In addition, we designed a quantized phase triplicator profile, which adapts to the discrete pixel structure of SLMs, with a high triplicator efficiency. Finally, we also consider the large number of pixels available in modern SLMs to apply a random multiplexing approach.

5.1. Phase-only gratings

Phase-only gratings do not absorb light, thus leading to an improved diffraction efficiency compared to amplitude gratings. Thus, several techniques have been developed in the last decades to design and fabricate blazed, binary phase, and multilevel gratings [Mai-1990, Wal-1990] or even

continuous phase-only designs [Rom-2010].

In this section, we review the simplest classical phase-only grating designs (blazed, binary, and sinusoidal) and show their implementation on an SLM pixelated device [Gao-2023, Gao-2024a]. In general, when the size of the diffraction elements is close to the operating wavelength we should apply the electromagnetic polarization theory. However, in our case, we can work within the scalar diffraction theory because the pixel size of the SLM is much larger than the wavelength of the laser sources (see Chapter 4).

Hence, the paraxial Fourier optics domain is considered, where a scalar phase-only diffraction grating with period p can be expanded as a Fourier series [Goo-2005]. The Fourier series formalism was summarized in Section 2.1 for periodic functions in time. The functions describing diffraction gratings are periodic in space. Thus, applying Eqs. (2.4-2.5) where now the spatial coordinate x is considered as a variable instead of time t, and using $\omega_m = m\omega_1$ where $\omega_1 = 2\pi/p$, the scalar phase diffraction grating with a spatial period p can be directly written as:

$$g(x) = e^{i\varphi(x)} = \sum_{m=-\infty}^{+\infty} c_m \, e^{im2\pi x/p}, \tag{5.1}$$

whose coefficients are given by

$$c_m = \frac{1}{p} \int_{-p/2}^{+p/2} g(x) \, e^{-im2\pi x/p} \, dx. \tag{5.2}$$

Each term $e^{im2\pi x/p}$ in Eq. (5.1) represents a tilted plane wave that generates a diffraction order with index m. The intensity of each order is given by $I_m = |c_m|^2$.

As we mentioned in the Introduction, this Thesis analyses in particular the triplicator diffraction grating, which is defined to provide three equally intense diffraction orders (the 0th and ±1st orders), i.e.: $I_0 = I_1 = I_{-1}$ with the diffraction efficiency given by [Gao-2023, Gao-2024a]:

$$\eta_{0\pm1} = I_0 + I_1 + I_{-1}. \tag{5.3}$$

5.1.1. The blazed phase grating

A classical phase-only design is the blazed phase grating [Gao-2023, Gao-2024a]. It features a linear phase profile $\varphi(x) = \pm 2\pi x/p$. This "saw-tooth" profile is illustrated (modulo 2π) in Figs. 5.1(a-b) for the positive and negative blazed gratings. Therefore, inserting $e^{\pm i2\pi x/p}$ as $g(x)$ into Eq. (5.2)

yields the Fourier coefficients: $c_m = \mathrm{sinc}(m \pm 1)$, where $\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$. Only $m = \pm 1$, for the positive and negative blazed phase profiles, gives $c_{\pm 1} = 1$ ($I_{\pm 1} = 1$), which means a single $\pm$1st diffraction order with 100% efficiency.

Figure 5.1 shows that the positive and negative blazed phase profiles with 2π phase modulation generate a single $\pm$1st order in their Fourier spectrum, and there are no other high orders (Figs. 5.1(a-b)). This single order can be realized by encoding the corresponding gray level images onto the phase-only SLM as shown in Fig. 5.1(c). Obviously, if a uniform gray level is encoded (as shown on top of panel(c)) the non-modified input beam is obtained i.e. a single 0th order. Then, displaying on the SLM a positive (or a negative) blazed phase profile, all the input light should be transferred to the +1st (or to the –1st) order from the 0th order. This is justified in the experiments in Fig. 5.1(d), where the gratings period is $p = 64$ pixels. Note that no high orders appear in our experimental results, but only $\pm$1st orders.

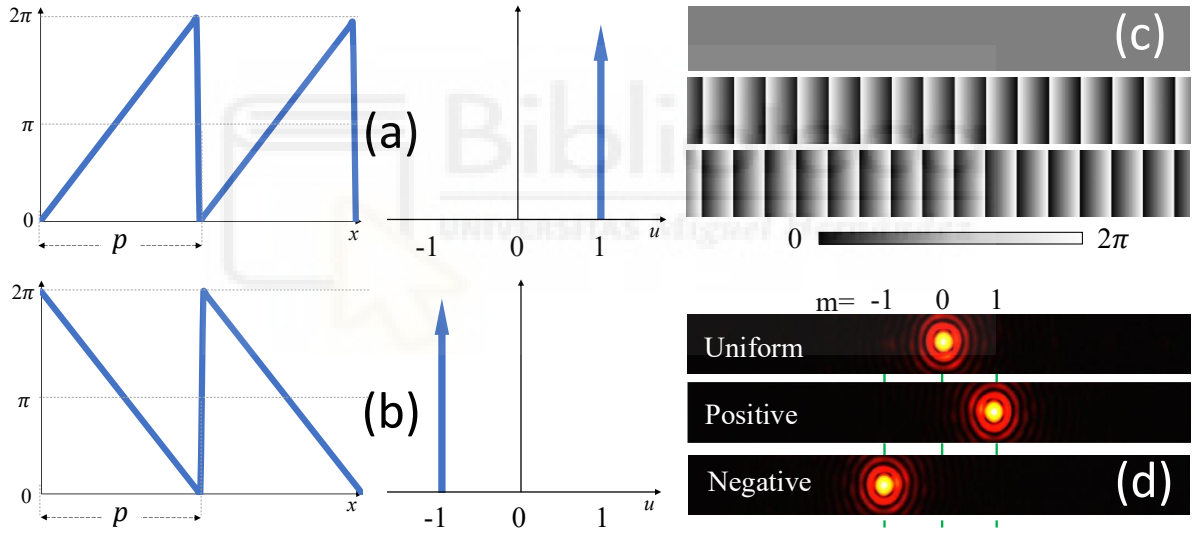

Fig. 5.1. Blazed phase profile with 2π phase modulation and corresponding Fourier spectrum [Gao-2023] (a) positive blazed; (b) negative blazed. (c) Gray level images, and (d) experimental diffraction patterns for the uniform grating, the positive, and the negative blazed grating.

5.1.2. The binary phase grating

The binary phase grating is interesting not only because it is simpler to fabricate than the blazed grating, but also because it serves to calibrate the phase modulation in SLMs [Lu-1994]. In addition, it is a feasible option to generate a triplicator with high diffraction efficiency [Gao-2023, Gao-2024a], as explained below. The binary phase profile within a period can be expressed as:

$$\varphi(x) = \begin{cases} 0, & \text{if } 0 \le x \le \dfrac{p}{2} \\ \varphi, & \text{if } \dfrac{p}{2} \le x \le p \end{cases}, \tag{5.4}$$

where φ denotes the phase difference between the two levels in the grating and each level occupies half the period. The Fourier coefficients (Eq. (5.2)) for this grating are given by

$$c_0 = \frac{1}{2}\left(e^{i\varphi} + 1\right), \tag{5.5a}$$

$$c_{m\neq 0} = \frac{1}{2}\left(e^{i\varphi} - 1\right)\operatorname{sinc}\left(\frac{m}{2}\right), \tag{5.5b}$$

where $\operatorname{sinc}(x) = \sin(\pi x)/(\pi x)$. Thus, the intensity of the 0th and ±1st orders is

$$I_0 = |c_0|^2 = \frac{1}{2}(1 + \cos\varphi) = \cos^2\left(\frac{\varphi}{2}\right), \tag{5.6a}$$

$$I_{\pm 1} = \left|c_{\pm 1}\right|^2 = \frac{2}{\pi^2}(1 - \cos\varphi) = \frac{4}{\pi^2}\sin^2\left(\frac{\varphi}{2}\right). \tag{5.6b}$$

Figures 5.2(a-b) illustrate the binary phase profile with $\varphi = \pi$ and its corresponding Fourier spectrum. Figure. 5.2(c) shows the theoretical and experimental normalized intensity of the 0th and ±1st orders as a function of the phase difference φ [Gao-2023, Gao-2024a]. As the phase attains π, I_0 reaches zero and $I_{\pm 1}$ reach their maximum intensity of $I_{\pm 1} = 4/\pi^2$, thus diffracting 80.1% of the input light into the ±1st orders. This is why the π phase binary grating is widely applied for transferring intensity from the 0th order mainly to the ±1st order, with little intensity being transferred to the higher odd orders and no intensity in the even orders due to its duty cycle of $1/2$. On the other hand, note that these curves intersect at $\varphi = 0.64\pi$ and $\varphi = 1.36\pi$, where $I_0 = I_{\pm 1} = 0.288$, thus rendering a triplicator with theoretical diffraction efficiency $\eta_{0\pm 1} = 86.4\%$. Note that the experimental points on the left part of the curve are slightlty lower that the theory. This intensity leak is caused by the fringing effect in the SLM. Here the impact is minor because the grating has a large period, but in Chapter 8 we show how relevant it is when the grating has a small period. Figures 5.2(d-e) show, for various phase values, the gray level images addressed to the SLM, and the experimental diffraction patterns captured with the camera. We can see that the diffraction patterns for $\varphi = 0.64\pi$ and $\varphi = 1.36\pi$ reproduce a triplicator, both with $\eta_{0\pm 1}$~85%, close to the theoretical value. Also, we confirm that the input light is transferred from the 0th order (pattern $\varphi = 0$) to the ±1st order (pattern $\varphi = \pi$). All these results were obtained with gratings encoded with $p = 64$ pixels per period. In this way, the fringing effect is minimized [Mor-2021]. The images in Fig. 5.2(d) are deliberately saturated to make higher orders clearly visible.

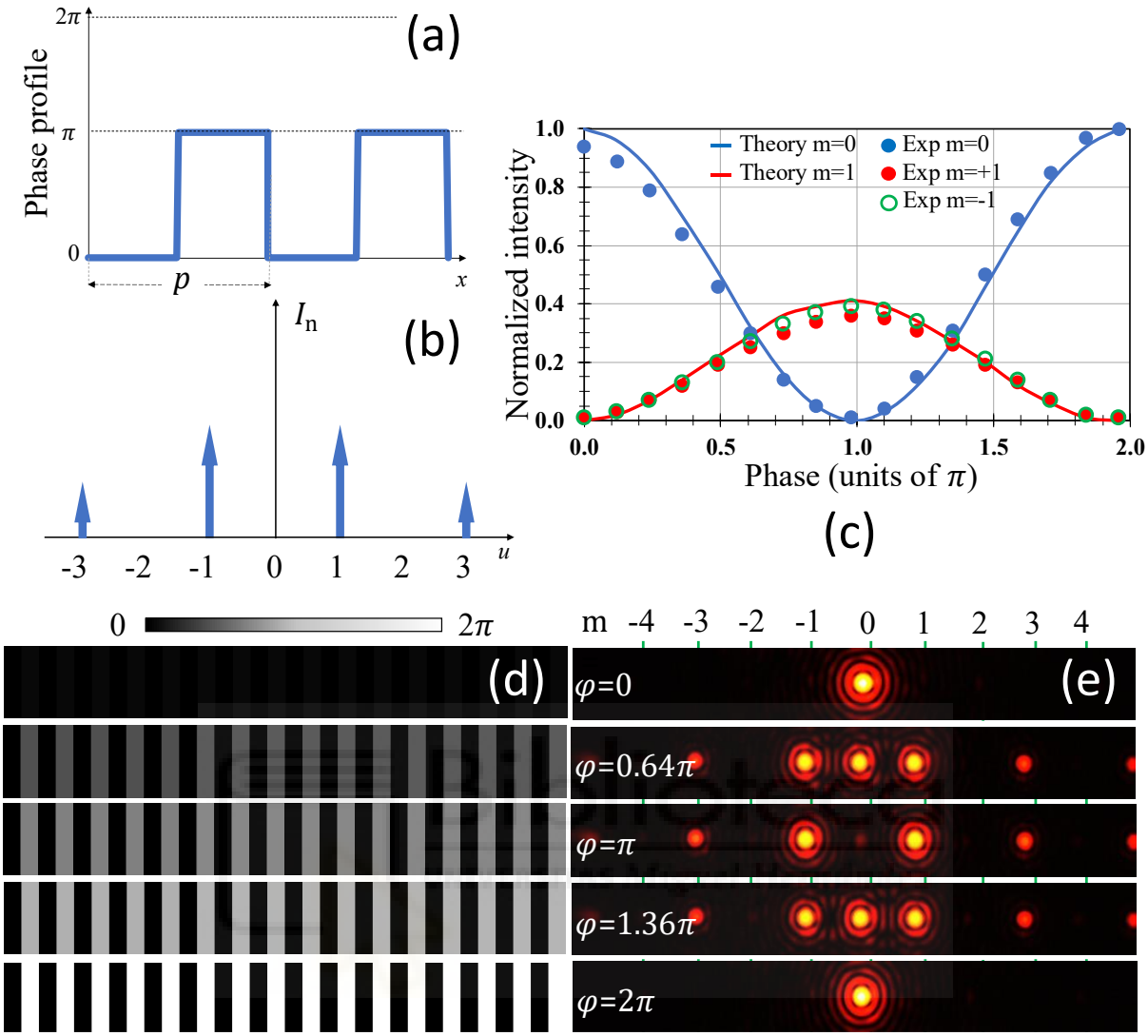

Fig. 5.2. (a) Binary phase profile and (b) the Fourier spectrum with a π-phase modulation [Gao-2023]. (c) Theoretical (solid lines) and experimental (dots) normalized intensity of the 0th and ±1st orders versus phase level. (d) Gray level images addressed to the SLM and (e) the corresponding experimental diffraction patterns for different phase values.

5.1.3. Sinusoidal phase grating

Another classical phase-only grating often found in the literature is the sinusoidal phase profile given by

$$\varphi(x,a) = \pi + a\pi \cdot \cos\left(\frac{2\pi x}{p}\right). \tag{5.7}$$

where values $a \in [0,1]$. Here the phase varies sinusoidally with x, with an elongation $\pm a\pi$, around a central value that we chose π, so that the phase profile $\varphi(x,a)$ is within the range $[0,2\pi]$. Figure 5.3(a) illustrates the phase profile for the case where $a = 0.457$. When using Eq. (5.7) into Eq. (5.2), it is found that the Fourier coefficients coincide with the Bessel functions [Goo-2005]:

$$J_n(x) = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{i(x\sin\beta - n\beta)}\,dx, \tag{5.8}$$

where $x = a\pi$. Thus, the intensity of the diffraction orders is given by $I_n = |J_n(a\pi)|^2$. Figure 5.3(b) shows the Fourier spectrum for case $a = 0.457$. Note that the 0th and ±1st orders have the same intensity and also note the less intense ±2nd and ±3rd orders in the Fourier spectrum. This situation corresponds to the sinusoidal phase triplicator with the highest efficiency [Gao-2023, Gao-2024a].

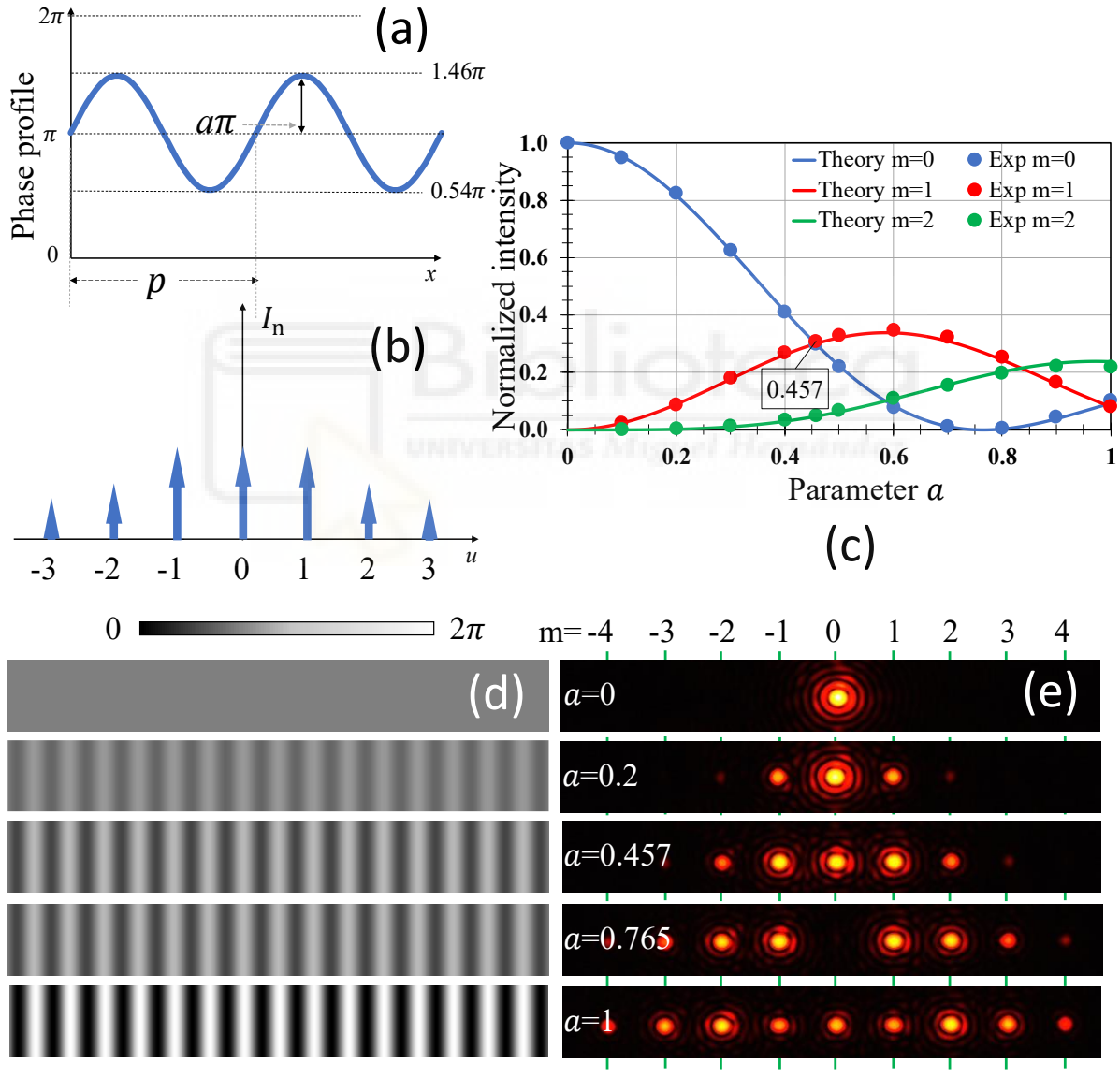

Fig. 5.3. (a) Sinusoidal phase profile for $a = 0.457$, and (b) the Fourier spectrum [Gao-2023]. (c) Theoretical (solid lines) and experimental (dots) normalized intensity of the 0th, 1st, and 2nd orders versus parameter a. (d) Gray level images addressed to the SLM and (e) corresponding experimental diffraction patterns for different values of parameter a.

To demonstrate it, we plot in Fig. 5.3(c) the theoretical and experimental normalized intensity of the 0th, 1st, and 2nd orders versus parameter a. Most of the light remains on the 0th order for low parameter values, and we found that the intensity curves of the 0th and ±1st orders intersect at two parameter values: $a = 0.457$ and $a = 1$. Both render a triplicator, but the one at $a = 0.457$ is the most convenient, with a theoretical diffraction efficiency $\eta_{0\pm1} = 90\%$ close to the optimal Gori's triplicator.

We confirm these results experimentally [Gao-2023, Gao-2024a]. Figures 5.3(d-e) show the gray level images addressed to the SLM and the experimental diffraction patterns, again for gratings with a period of $p = 64$ pixels. The diffraction pattern for $a = 0.457$ confirms a triplicator with high efficiency and with most of the noncontributing light concentrated on the ±2nd orders. As parameter a increases, light is diffracted onto higher orders, as observed in the patterns with $a = 0.765$ and $a = 1$. Note that the zero-order cancels for $a = 0.765$, in agreement with the theoretical I_0 curve in Fig. 5.3(c). The pattern for $a = 1$ is a triplicator with a much lower efficiency, where in fact the ±2nd orders are more intensive.

5.2. Gori's optimum phase triplicator grating

In 1998 Gori *et al.* [Gor-1998] derived the phase grating profile providing a triplicator with optimal efficiency, whose solution can be written as:

$$\varphi(x, a) = \pi + \arctan\left(a \cdot \cos\left(\frac{2\pi x}{p}\right)\right), \tag{5.9}$$

where again we have added π to the original solution to have phase values in the range $[0,2\pi)$. This solution provides the maximum efficiency of the 0th and ±1st orders altogether, i.e. it maximizes $\eta_{0\pm1}$ in Eq. (5.3) and provides intensities at these diffraction orders given by

$$I_0 = \frac{4}{\pi^2}[K(-a^2)]^2, \tag{5.10a}$$

$$I_{\pm1} = \frac{4}{\pi^2 a^2}[E(-a^2) - K(-a^2)]^2, \tag{5.10b}$$

where $K(m)$ and $E(m)$ are the complete elliptic integrals of the first and second kind, respectively. Parameter a controls the phase profile and the relative intensity between the 0th order and the ±1st orders. Figure 5.4(a) shows the phase profiles for three values of a. Note the profile is sinusoidal for $a = 1$ but it tends to be a binary profile for large values of a. For very low values,

the phase modulation is reduced, reaching a uniform phase in the limit $a \to 0$ that renders only the 0th order. The case $a = 2.65$ is the optimal triplicator profile, with three equi-intense 0th and ±1st orders, whose Fourier spectrum is illustrated in Fig. 5.4(b).

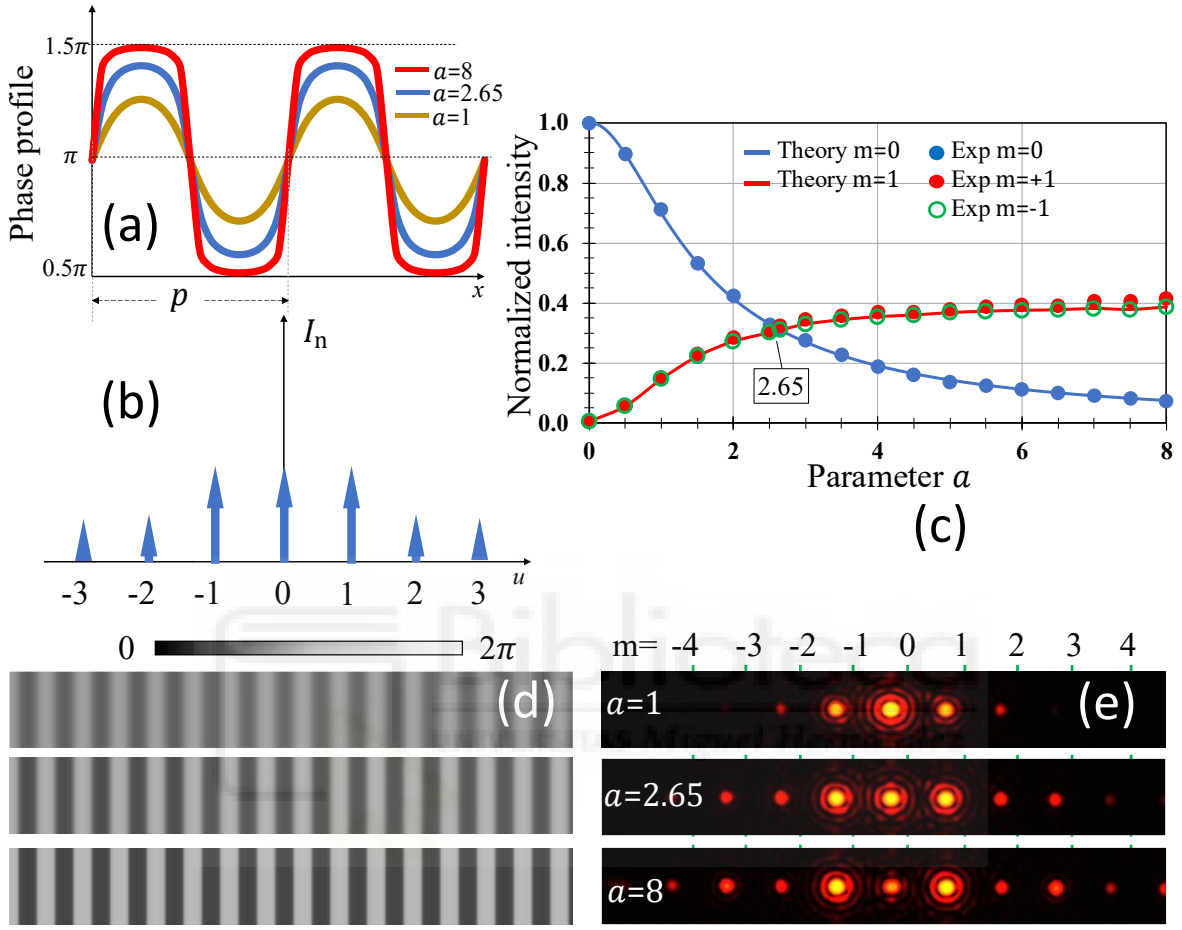

Fig. 5.4. (a) Gori's phase profile [Gao-2023] for $a = 1$, $a = 2.65$, and $a = 8$, and (b) the Fourier spectrum for $a = 2.65$. (c) Theoretical (solid lines) and experimental (dots) normalized intensity of the 0th and ±1st orders versus a. (d) Gray level images addressed to the SLM and (e) experimental diffraction patterns for different parameters a.

The theoretical normalized intensity of the 0th and ±1st orders, Eqs. (5.18), are depicted in Fig. 5.4(c) as a function of parameter a, together with the experimental measurements [Gao-2023, Gao-2024a]. To our knowledge, these results are the first experimental verification of Gori's intensity curves versus a. Figures 5.4(d-e) show the gray level images addressed to the SLM and the experimental diffraction patterns for three different values of a. Again, the gratings are displayed with 64 pixels per period; thus, the quantization effect is negligible. For $a = 2.65$, where the I_0 and $I_{\pm 1}$ curves in Fig. 5.4(c) intersect, the expected optimum triplicator is obtained with a measured experimental efficiency $\eta_{0\pm 1} = 94.7\%$, somewhat above the theoretical value in [Gor-1998]. Note that although the three central orders are the most intense, higher orders are also

visible. The diffraction pattern for $a = 8$, with most of the intensity in the ± 1st orders, resembles a binary grating.

5.3. Borghi's phase triplicator grating

We also explore the four-level phase triplicator derived by Borghi *et al.* [Bor-2001], which features unequal spacing for each phase level. This phase profile is characterized by three parameters as indicated in Fig. 5.5(a): two phase shifts φ_1 and φ_2 that are added and subtracted from a certain value, which we select again π, and the width r of the two central steps relative to the grating's period p. The coefficients of the 0th order and the ± 1st orders are given by [Bor-2001]:

$$c_0 = 4\left[r\cos(\varphi_1) + \left(\frac{1}{4} - r\right)\cos(\varphi_2)\right], \tag{5.11a}$$

$$c_{\pm 1} = \frac{4}{2\pi}\{\sin(\varphi_1)[1 - \cos(2r\pi)] + \sin(\varphi_2)\cos(2r\pi)\}. \tag{5.11b}$$

The solution that maximizes the triplicator efficiency in the scalar regime was found to be: $r = 0.0641p$, $\varphi_1 = 0.153\pi$, and $\varphi_2 = 0.356\pi$, leading to the phase profile and the Fourier spectrum as depicted in Figs. 5.5(a-b). The theoretical efficiency $\eta_{0\pm1} = 91.05\%$ is only slightly lower than that of the optimal Gori's triplicator ($\eta_{0\pm1} = 92.6\%$).

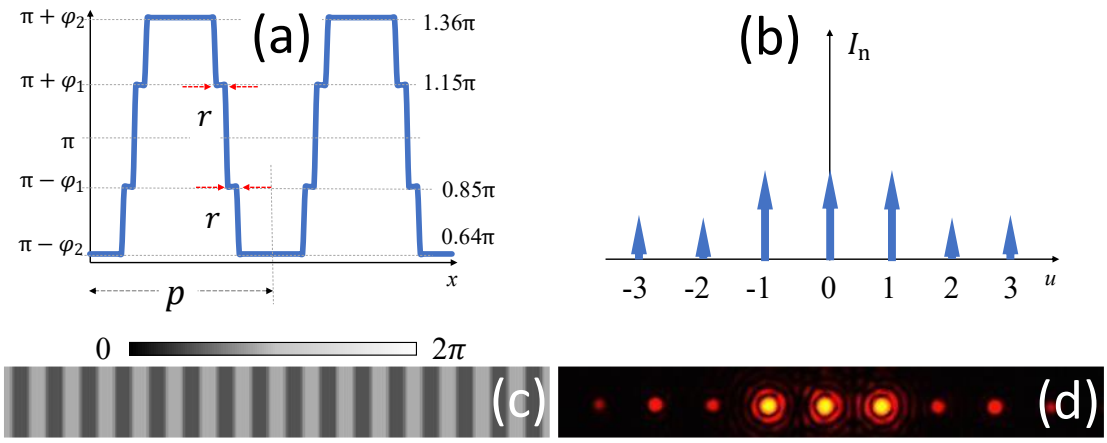

Fig. 5.5. (a) Four-level phase profile with unequal period spacing and (b) the corresponding Fourier spectrum [Gao-2023]. (c) Gray level image addressed to the SLM and (d) the experimental diffraction pattern of the four-level phase triplicator.

This kind of profile was intended for lithographic fabrication where it can be produced only

with two exposure steps. Here we reproduce it with the SLM (which has equispaced pixels) [Gao-2023, Gao-2024a] we used again a period of $p = 64$ pixels, so $r_{exp} = 0.0641p \approx 4$ pixels. The addressed gray level image and the experimental diffraction pattern are shown in Figs. 5.5(c-d).

5.4. Quantized phase triplicator gratings

The SLM is a pixelated display that can only implement discrete phase levels [Gao-2023, Gao-2024a]. Therefore, the continuous phase profiles of the sinusoidal triplicator and Gori's triplicator previously presented require a large number of pixels per period. Our gratings in the previous sections were encoded with $p = 64$ pixels per period, so that the quantization does not significantly affect the results. In this section we analyze triplicator phase gratings with a discrete number (N) of phase levels per period; we call it quantized phase triplicator. This design can be a convenient choice for achieving a high diffraction efficiency in applications that require a small period. But, having in mind their implementation with a pixelated SLM, the phase levels must be regarded as equispaced, as shown in Fig. 5.6(a), where we consider N different phase levels $(\varphi_0, \varphi_1, \ldots, \varphi_{N-1})$ located at positions $(x_0, x_1, \ldots, x_{N-1})$ within one period (p). The length of each phase segment is $\Delta = p/N$, which corresponds to a single pixel in the SLM.

In Figs. 5.7(b-d)) we can see three examples of such quantized phase profiles. The general quantized phase grating can be described as a summation of rectangular functions, each with a given phase level:

$$g(x) = \sum_{j=0}^{N-1} e^{i\varphi_j} \cdot \mathrm{rect}\left(\frac{x - x_j}{\Delta}\right), \tag{5.12a}$$

$$\mathrm{rect}\left(\frac{x - x_j}{\Delta}\right) = \begin{cases} 1, \text{if } |x - x_j| \le \dfrac{\Delta}{2} \\ 0, \text{if } |x - x_j| > \dfrac{\Delta}{2} \end{cases} \tag{5.12b}$$

where each pixel is assigned a phase φ_j within a rectangle of width Δ, $j = 0, 1, \ldots, N-1$ and where the centre of each pixel is given by

$$x_j = \frac{p}{2} \cdot \left(\frac{(2j+1)}{N} - 1\right). \tag{5.13}$$

The Fourier coefficients $c_m(N)$ in Eq. (5.2) are now

$$c_m = \frac{1}{N}\mathrm{sinc}\left(\frac{m}{N}\right)\sum_{j=0}^{N-1} e^{i\varphi_j} e^{-im2\pi x_j/p}, \tag{5.14}$$

and the corresponding intensity of the diffraction orders is:

$$I_m^{(N)} = \frac{\mathrm{sinc}^2\left(\frac{m}{N}\right)}{N^2} \times \left[N + \sum_{j\neq k=0}^{N-1} 2\cos\left(\varphi_j - \varphi_k - \frac{2\pi m(j-k)}{N}\right)\right]. \tag{5.15}$$

By computing Eq. (5.15), for each value of N we seek for the phases φ_j that provide a triplicator response, with a higher diffraction efficiency $\eta_{0\pm1}$ and the least intensity variation among the three orders [Gao-2023, Gao-2024a]. Figure 5.7(a) shows the triplicator efficiency $\eta_{0\pm1}$ versus N, and Fig. 5.7(b) plots the efficiency of the second orders ($\eta_{\pm2} = I_2 + I_{-2}$) and the third orders ($\eta_{\pm3} = I_3 + I_{-3}$).

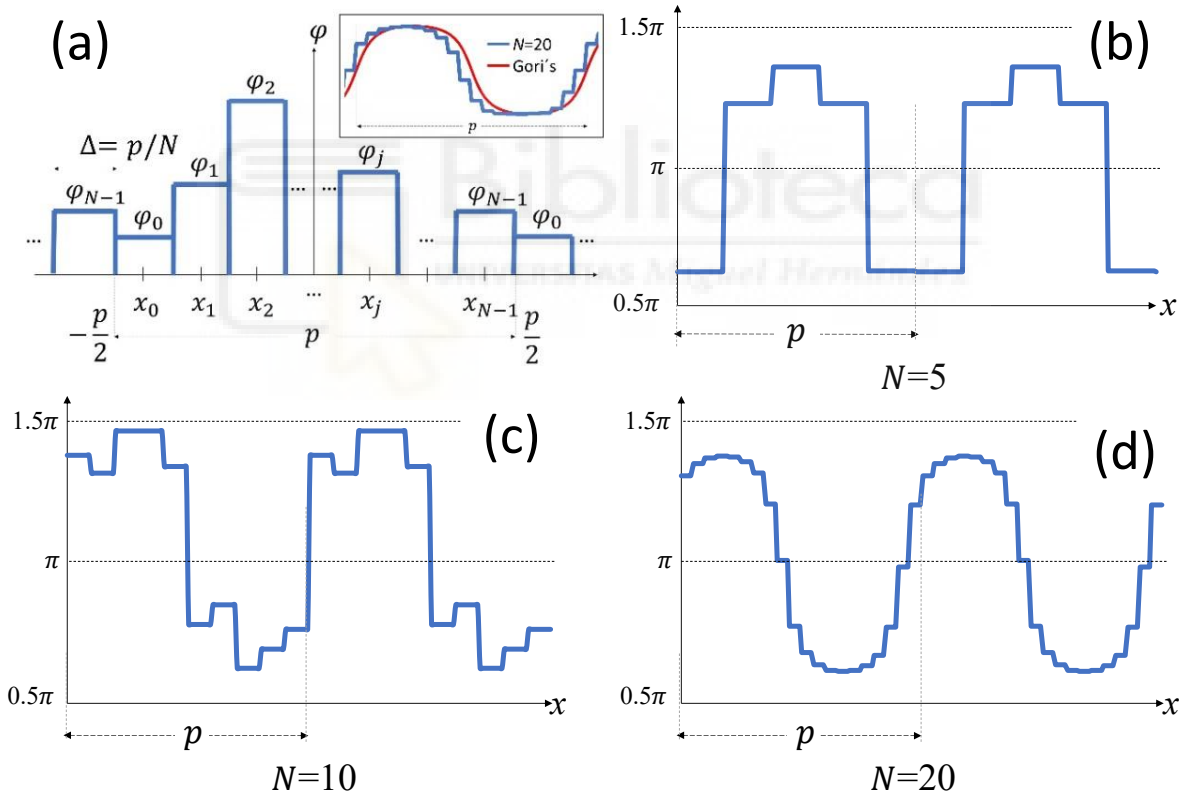

Fig. 5.6. (a) Quantized phase profile [Gao-2024a]. Inset, for $N = 20$ versus Gori's. Specific profiles for triplicators with (b) $N = 5$, (c) $N = 10$ and (d) $N = 20$ [Gao-2023].

These curves reveal that, for low odd values $N = 3$ and $N = 5$, the triplicator efficiency is lower than that of the binary triplicator, $\eta_{0\pm1} = 86.4\%$. For $N = 3$, the ±3rd orders vanish because the phase profile converges to a binary grating with a 1/3 duty cycle [Mar-2007]. For $N = 4$, the

solution is equivalent to the binary case, but every two pixels are assigned the same phase level. Note that this is different from the four-phase solution proposed in Section 5.3, which considers different lengths of the grating period for each phase level. Thus, it is not possible to implement it with four equispaced pixels per period. Only when the number of levels reaches $N = 6$, the efficiency of the binary triplicator is surpassed, and then it slowly increases with N. We simulate up to $N = 20$, for which $\eta_{0\pm1} = 92\%$ is very close to that of Gori's triplicator and whose quantized phase profile approaches the continuous one (inset, Fig. 5.6(a)). This behaviour is confirmed experimentally in Figs. 5.7(c-d), which shows the gray level images addressed to the SLM and the experimental diffraction patterns for $N = 2, 3, 5, 20$. They all feature the three central equi-intense orders, but also many other higher harmonic orders. The diffraction pattern with $N = 3$ has no $\pm$3rd orders but brighter 2nd orders in agreement with the numerical curves $\eta_{\pm2}$ and $\eta_{\pm3}$ in Fig. 5.7(b). The diffraction pattern with $N = 5$ shows more harmonic orders than those with lower N, which agrees with the fall of $\eta_{0\pm1}$ at $N = 5$ in Fig. 5.7(a).

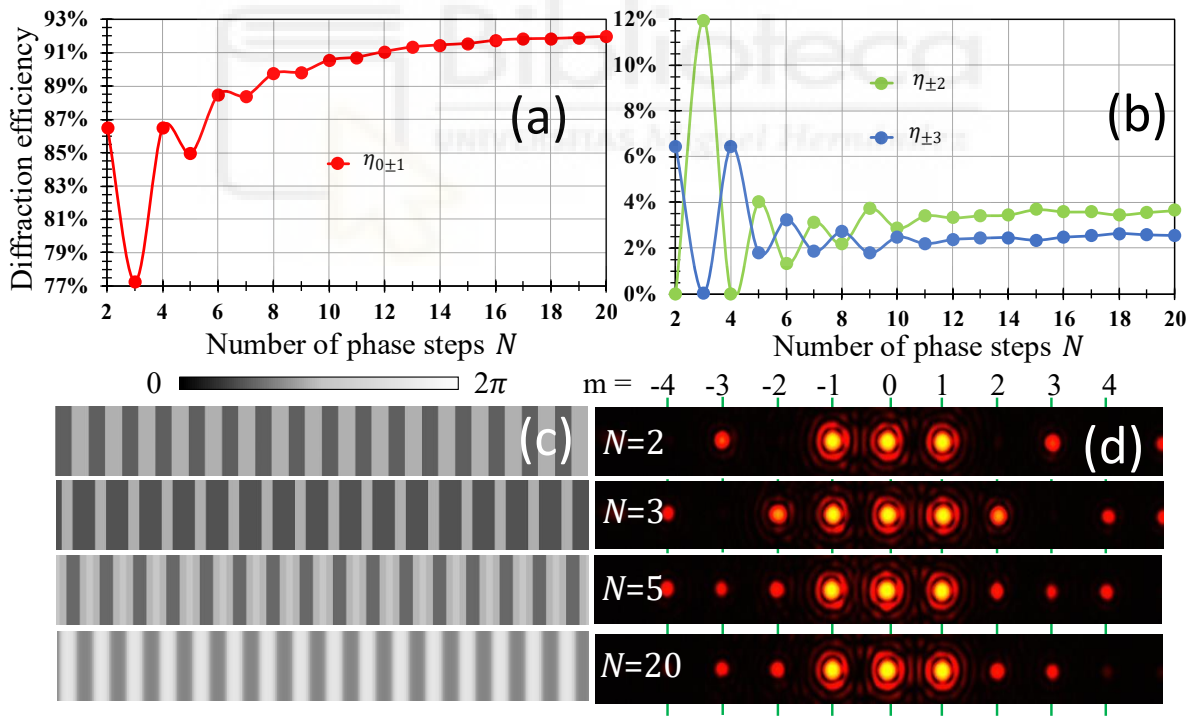

Fig. 5.7. The simulated triplicator diffraction efficiency of the quantized phase grating in terms of N [Gao-2023, Gao-2024a], (a) $\eta_{0\pm1}$ (b) $\eta_{\pm2}$, $\eta_{\pm3}$. (c) Gray level images addressed to the SLM and (d) the corresponding experimental diffraction patterns for $N = 2, 3, 5, 20$.

5.5. The random multiplexing approach

The large number of pixels available in high-resolution SLMs allows the use of a random technique to multiplex several phase functions onto a single phase-only mask [Dav-2023]. The multiplexing approach is illustrated in Fig. 5.8. We first create two complementary random binary amplitude patterns $A(x,y)$ and $a(x,y)$, where each pixel is randomly given a value 1 or 0. These two patterns are complementary, i.e., one pattern selects certain pixels of the image (by assigning them a value 1), and the other pattern selects the rest of the pixels in the image. Therefore, together they cover the entire image. In Fig. 5.8, left column, black and white denote binary amplitude values 0 and 1 in the random patterns.

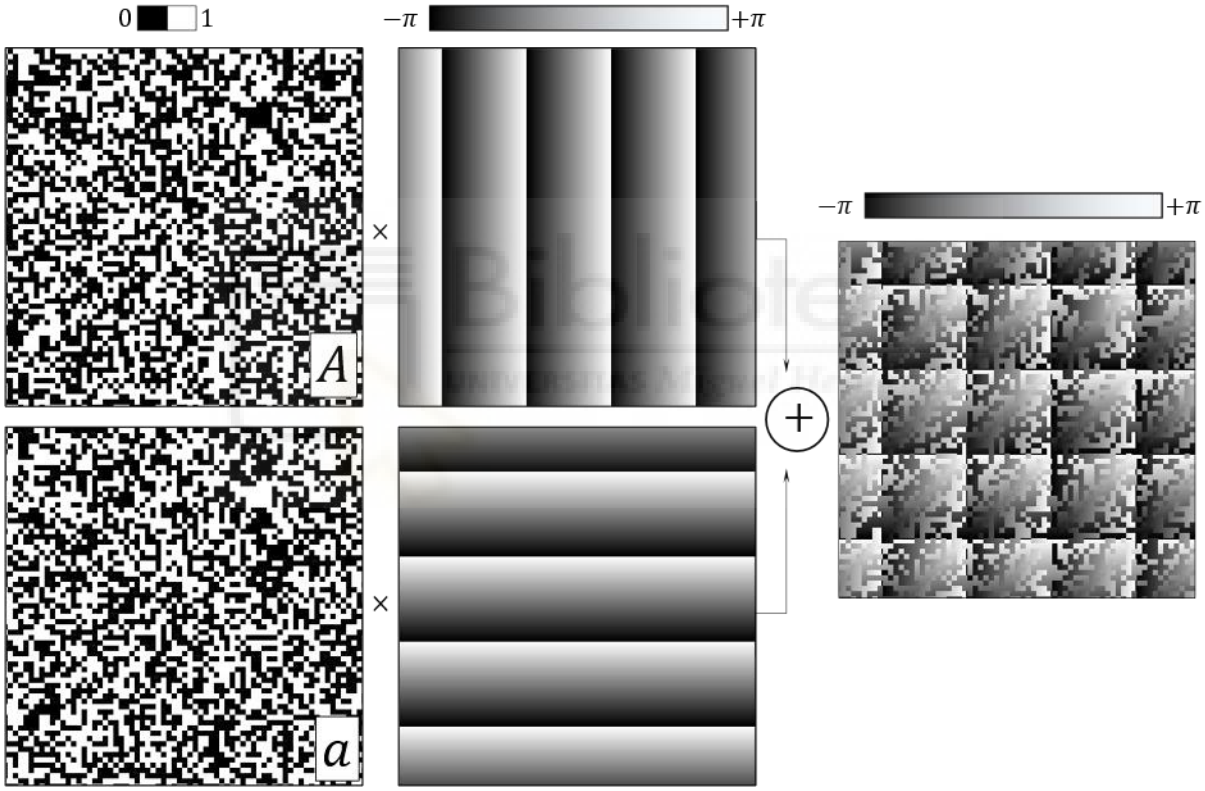

Fig. 5.8. Design of the multiplexed phase function using the random approach [Dav-2023]. The left column shows two complementary binary amplitude masks $A(x,y)$ and $a(x,y)$. They multiply linear phases (central column) to produce two different outputs at different locations, here in horizontal and vertical directions. The corresponding results are added to provide the final multiplexed phase-only function.

Now two different phase functions are encoded onto each binary pattern. For that purpose, we simply multiply one phase function by the binary mask $A(x,y)$ and the other one by the complementary binary mask $a(x,y)$. For instance, Fig. 5.8. shows the encoding of one different blazed diffraction grating onto each pattern, one diffracting in the horizontal direction and the

other in the vertical direction. The resulting multiplexed phase pattern $F(x,y)$ can be written as

$$F(x,y) = A(x,y)e^{i\gamma x} + a(x,y)e^{-i\gamma y}. \tag{5.16}$$

Note that the simple addition of the two blazed gratings $e^{i\gamma x} + e^{-i\gamma y}$ to produce two diffraction orders results in a complex-valued function which cannot be directly implemented in a phase-only SLM. On the contrary, the multiplexed function $F(x,y)$ in Eq. (5.16) is a phase-only function that can be displayed directly on the SLM.

5.5.1. The pixel crosstalk problem

The random patterns $A(x,y)$ and $a(x,y)$ in Fig. 5.8 used in this multiplexing approach are patterns that vary at the pixel level [Dav-2023]. Therefore, when the multiplexed phase mask $F(x,y)$ shown in the figure is displayed on an SLM, there are pixels which might have a high phase difference with respect to neighbour pixels, thus being affected by the pixel crosstalk caused by the fringing field effect described at Section 4.3.1 in Chapter 4.

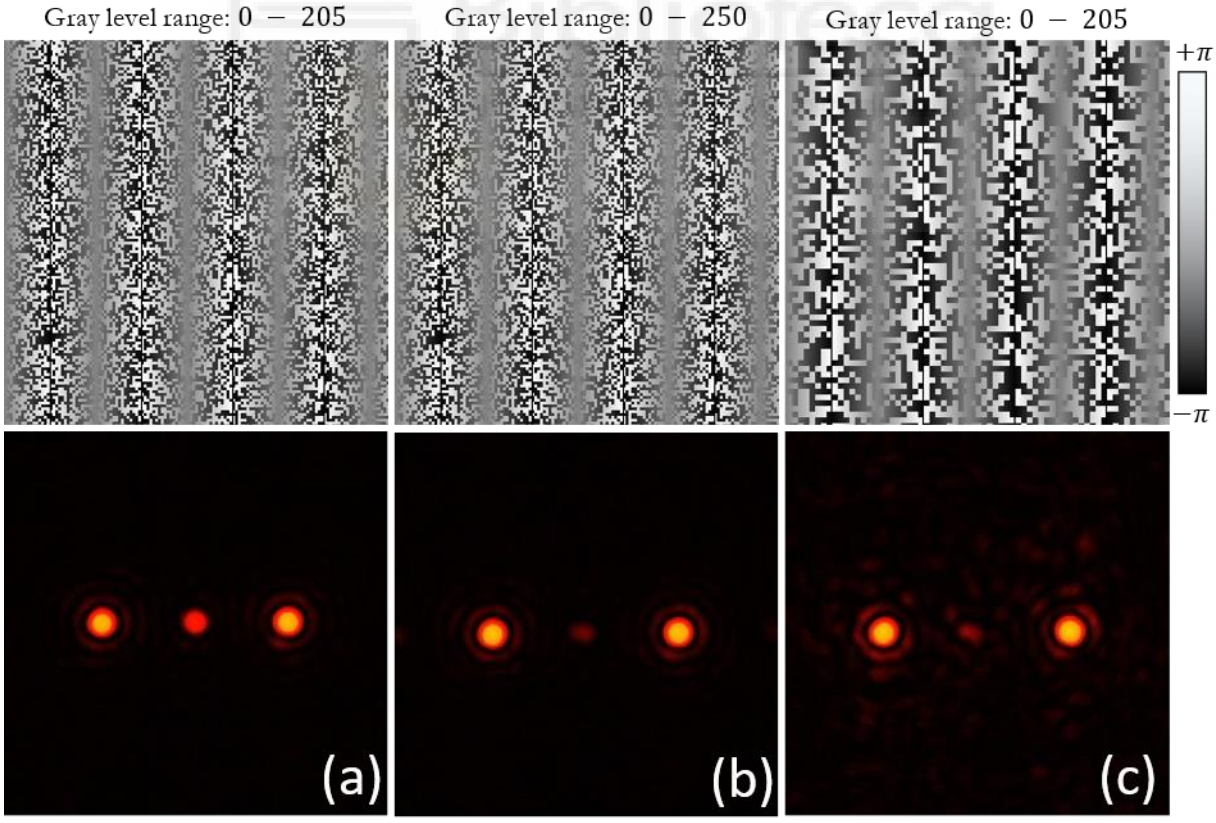

Fig. 5.9. Experimental results of multiplexing two opposite blazed gratings in the Hamamatsu SLM [Dav-2023]. Result using a random pattern with one pixel size (a) when the maximum gray level is adjusted to 2π phase modulation ($g = 205$) and (b) when the maximum gray level is increased ($g = 250$) to enlarge the phase modulation. (c) Result using a random pattern with 2×2 macro-pixels and the original phase modulation depth ($g = 205$). On top of each

experimental capture, a detail of the corresponding phase mask is shown.

Figure 5.9 shows some experimental results obtained with the Hamamatsu SLM, where two blazed gratings diffracting horizontally in opposite directions were encoded. Let us remind that the phase modulation depth for this SLM at the He-Ne laser wavelength $\lambda = 633$ nm exceeds 2π. A phase modulation of 2π is obtained for the addressed gray level $g = 205$. The output in Fig 5.9(a) shows the two expected focused spots, but a strong DC order is also present in the centre as a result of the fringing field effect.

We discuss two possible methods for minimizing this unwanted DC order [Dav-2023]. The easiest way is to increase the phase depth of the SLM. In the result shown in Fig 5.9(b), the same pattern as in Fig. 5.9(a) was displayed in the SLM, but the maximum gray level was increased to 250, which corresponds approximately to a 2.5π phase modulation measured in the calibration (see Section 4.3) This change does not affect the lower voltages required for the lower phases, but increases the voltages required for higher phases. The fringing field effect reduces the slope of the ideal 2π phase jump [Mos-2019]. Increasing the maximum value in the addressed gray level ramp pattern increases the slope of the phase jump and compensates the fringing field effect while satisfying the blazing requirements. As a result, the DC term is practically eliminated, as shown in Fig. 5.9(b).

Another approach to reduce the DC term caused by the fringing field effect [Dav-2023] is shown in Fig 5.9(c). Here the maximum gray level is kept at $g = 205$ but the size of the random dots used for the binary amplitude patterns is increased. Instead of creating the random patterns over the entire image array assigning a different value to each pixel, we make the random pattern with larger "macropixels" consisting of 2×2 individual pixels. The result shows how the intensity of the DC term is reduced as well. The advantage of this approach is that it can be applied to SLMs whose phase modulation range does not exceed 2π. The disadvantage is that the effective number of random pixels is reduced, and the background noise increases.

Figure 5.10 shows additional experiments that disclose the great flexibility provided that this simple random multiplexing approach compared to standard phase diffraction profiles. Figures 5.10(a-c) show how easy is to change the relative strength of each of the two multiplexed blazed gratings, simply by changing the percentage of pixels assigned to each binary amplitude mask. In Fig. 5.10(a) the same result as in Fig. 5.9(b) is shown, where 50% of the pixels are assigned to each of the two blazed gratings. Therefore, the two ±1st orders are generated with the same intensity. In Figs. 5.10(b-c), the percentage of pixels assigned to the +1st order is changed from 25% to 75%

(vice versa for the −1 order), and the experiment clearly shows the asymmetry in the diffraction orders intensity.

Figure 5.10(d) shows the experiment corresponding to the design presented in Fig 5.8, where a multiplexed phase mask is generated with $N = 2$ blazed gratings oriented in orthogonal directions, thus leading to two diffraction orders in horizontal and vertical directions. Note that the non-ideal phase modulation caused by the fringing effect causes other weaker (but noticeable) orders. Finally, Figs. 5.10(e-f) show the experimental result when multiplexing phase masks with $N = 4$ and $N = 8$ different blazed gratings of different percentage and orientations, so a pattern of diffraction orders is generated in the Fourier transform plane.

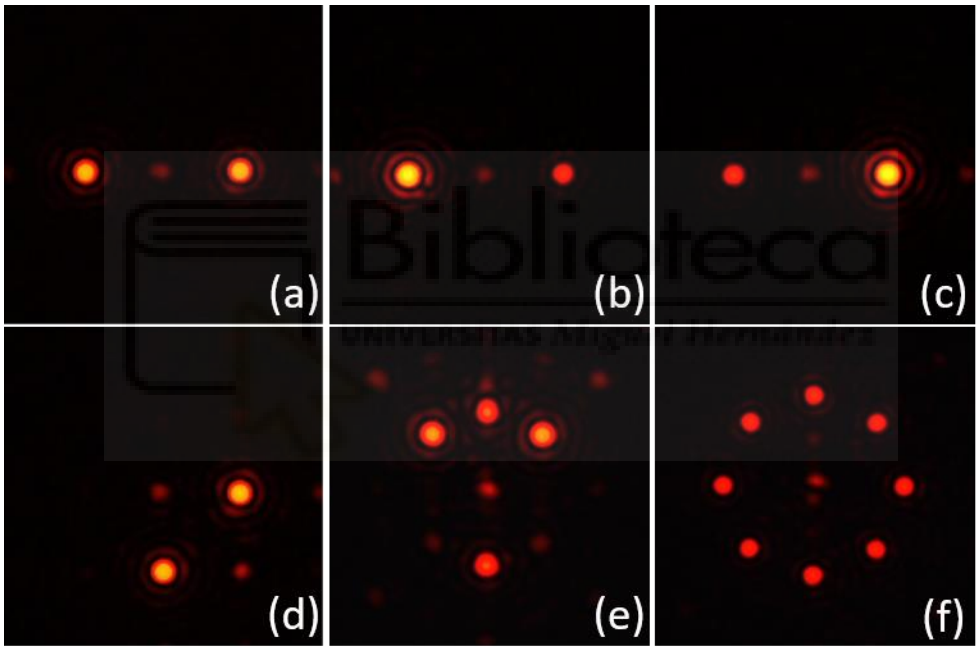

Fig. 5.10. Experimental results of multiplexing blazed phase gratings [Dav-2023]. (a-c) Result where the two random patterns have the same number of pixels: (a) 50% of pixels for each output, (b) 75% of the pixels for left output,(c) 25% of pixels for left output. (d-e) Result for different number (N) of multiplexed blazed phase gratings with different orientations: (d) $N = 2$, (e) $N = 4$ and (f) $N = 8$.

5.5.2. Random multiplexed triplicator grating

The random multiplexing approach is another feasible method for generating a triplicator grating. In this case, a multiplexed mask is generated with N=3 patterns: those shown in Fig. 5.1(c), i.e. two opposite blazed gratings together with the uniform gray level that generates a single 0th order

Figure 5.11 shows the gray level images addressed to the SLM using this approach, for

different macropixel sizes, and the corresponding experimental diffraction patterns. They are all triplicators, but in case 2×2 and, even more noticeable, in case 1×1, the zero-order is too bright due to pixel crosstalk and additional higher orders appear. This effect is minimized when the randomness is done on a larger macropixel. As shown, in the results for 4×4 and 8×8, the random approach redistributes the light not contributing to the target 0th and ±1st orders as background noise.

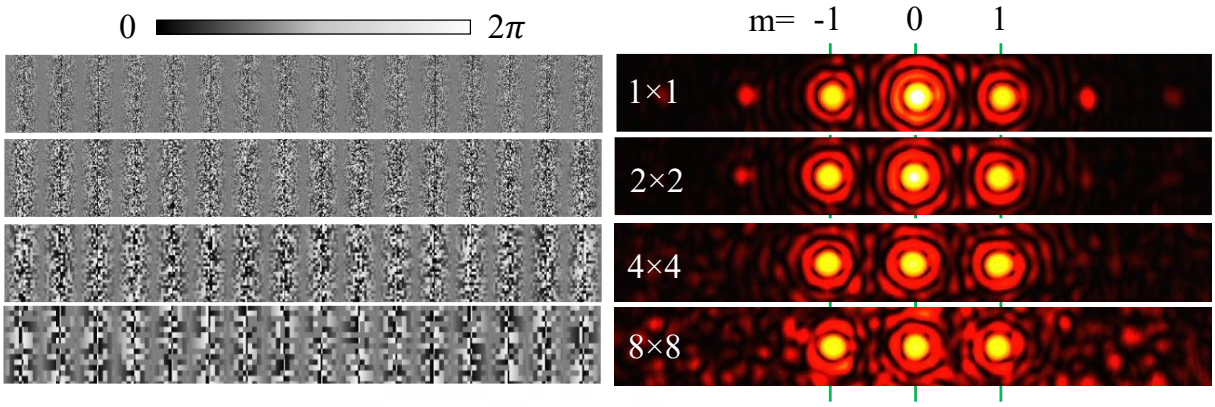

Fig. 5.11. Gray level images addressed to the SLM and the experimental diffraction pattern for different macropixel sizes $N \times N$ $(N = 1, 2, 4, 8)$ [Gao-2023].

6. Phase-only Fourier transform computer-generated holograms

The design of computer-generated holograms (CGHs) has a long history that goes back to shortly after the invention of the laser [Loh-2008]. Here we will focus directly on the realization of phase-only CGHs designed to be implemented with phase-only SLMs. We introduce Fourier transform (FT) CGHs, i.e., holograms designed to provide a desired pattern by diffraction in the FT domain. They are calculated using the inverse FT to generate a phase-only function that generates the desired object pattern through an optical FT obtained by diffraction.

Using only the phase of the FT provokes a distortion of the reconstructed pattern. Therefore, several techniques and algorithms have been developed, which are useful to reduce and compensate for this distortion. Here we review some simple methods based on adding amplitude and/or phase random noise to the original object. We also introduce iterative Fourier transform algorithms (IFTA), which provide a better CGH reconstruction. These methods will be employed and extended in Chapter 9 to implement CGHs with chromatic dispersion control.

6.1. Fourier transform phase-only CGHs

6.1.1. The kinoform CGH

Let us first introduce the kinoform [Les-1969], the simplest form of a phase-only CGH. This kind of hologram is designed by calculating the inverse FT of the pattern to be generated and discarding its magnitude. The phase of the inverse FT is directly the phase-only hologram. When this hologram is illuminated by a plane wave, the light diffraction produces a direct FT in the far field, which recovers the original function. However, the fact that the magnitude information has been discarded provokes a distortion of the optically reconstructed pattern. This distortion is typically an edge enhancement.

To illustrate this effect, we refer to the simulation in Fig. 6.1. Here we consider a simple object, a circular shape. The diameter is of 70 pixels, embedded in an image array of 1024×1024 pixels. The figure shows the images cropped to the middle area of 256×256 pixels. The FT of the circle shows a magnitude having the typical Airy pattern characteristic of circular apertures, and the phase distribution shows rings that alternate a π phase shift. Note that the energy is concentrated around the centre of the Fourier transform pattern, which corresponds to the low spatial frequencies of the image of the circle. The kinoform phase-only CGH is generated by discarding this magnitude information and normalizing it to one. Then, performing a direct Fourier transform provides an intensity that takes the shape of the original object, but edge enhanced.

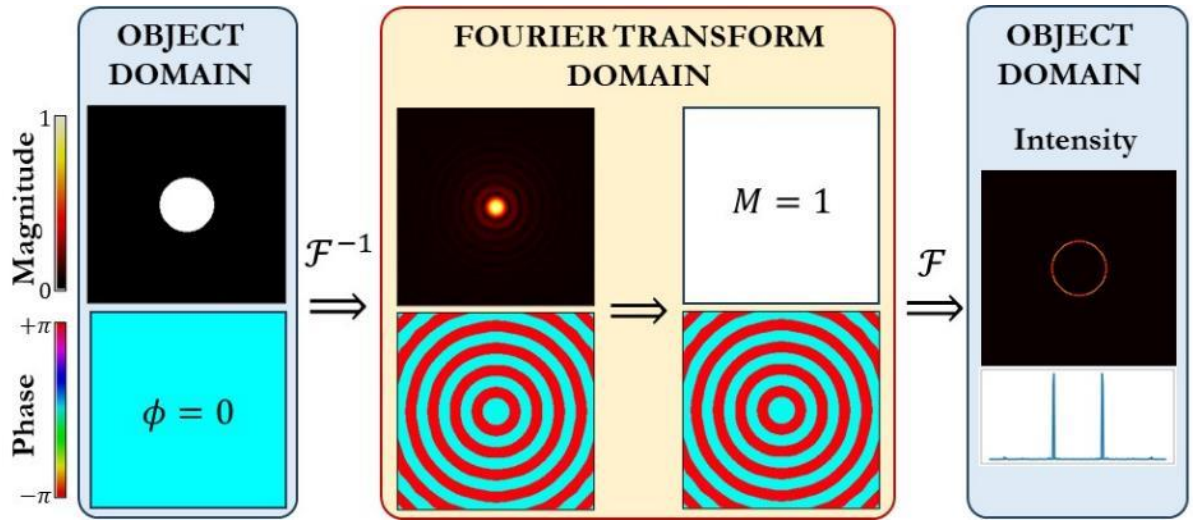

Fig. 6.1. Illustration of the design of a simple FT phase-only CGH (**kinoform**). First, the inverse FT is calculated. Secondly, the FT magnitude is discarded. After a direct FT, the intensity (magnitude squared) shows an edge-enhanced version of the original pattern. The inset at the right bottom of the hologram reconstruction shows the intensity profile along a horizontal line.

This edge enhancement effect can be understood since the edge of the binary pattern is the part of the image that requires the highest spatial frequencies. Thus, the magnitude normalization gives the same weight to the high and low frequencies, thus resulting in an effective high frequency reinforcement, leading to the edge enhanced effect.

6.1.2. The random magnitude approach

A quite simple but effective approach to avoid the edge enhancement effect was proposed in [Dav-1994]. It consists of multiplying the original object (the circle) by a random pattern, where each pixel is assigned randomly to values 0 and 1. The result, as shown in Fig. 6.2, looks like the circle but the inner part is filled with black and white pixels ("salt and pepper" noise). Therefore, the inner part of the circle is filled with many edges.

The inverse FT shows differences compared with the kinoform case illustrated in Fig. 6.1. While the magnitude is quite like the Airy pattern, and it is only affected by some noise in higher frequencies, the phase distribution looks clearly different to that in Fig. 6.1, now showing a continuous variation of the phase, and only showing the binary circular pattern in the centre. This phase distribution is the new phase-only CGH. Then, the direct FT leads to an intensity (magnitude squared) distribution that reproduces the filled circle, as shown in the last column of Fig. 6.2. Its intensity, though, is far from being uniform and it is instead filled with large intensity variations, as shown in the intensity profile. In addition, some noticeable noise is also present outside the circle reconstruction.

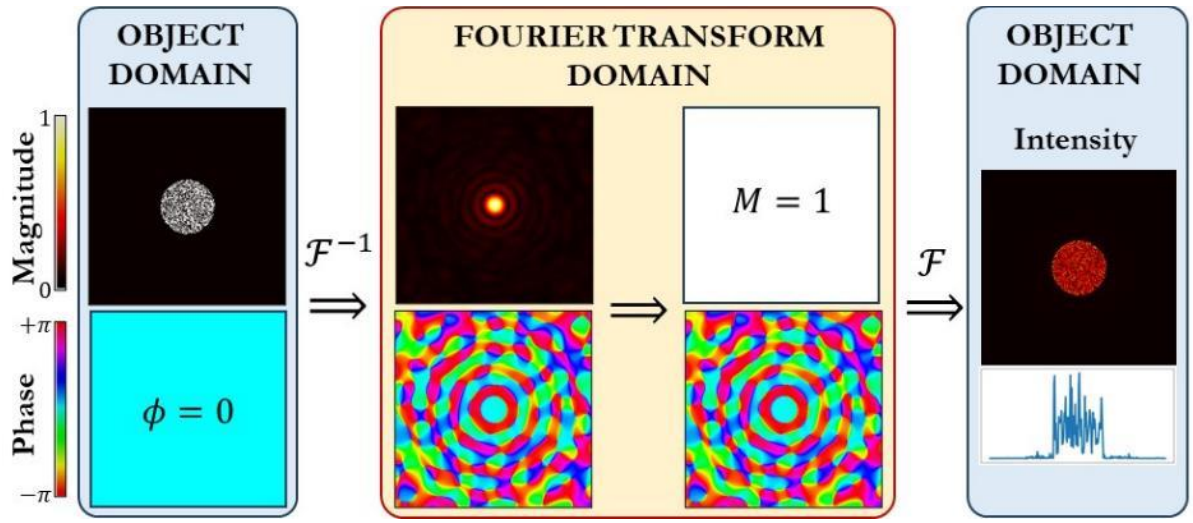

Fig. 6.2. Illustration of the design of a simple FT phase-only CGH with the **random magnitude approach**. A binary random pattern is multiplied by the original object before calculating the inverse FT. The inset at the right bottom of the hologram reconstruction shows the intensity profile along a horizontal line.

6.1.3. The random phase approach

An alternative and more efficient approach can be used by adding a random phase to the object. This is illustrated in Fig. 6.3. Now the object magnitude is left as the original. However, a random phase is considered, where each pixel is added a random phase value in the range $[-\pi, +\pi]$.

This random phase pattern acts as a diffuser, generating that the magnitude of the inverse FT is now distributed all over the frequency plane, instead of being concentrated at the centre as an Airy pattern. Note that now the phase-only CGH does not show the characteristic binary circular pattern shown in the two previous cases.

Therefore, the effect of normalizing the energy of the spatial frequencies by discarding the magnitude is now eliminated since the energy is distributed all over the frequency domain. This is shown in the hologram reconstruction, which again shows the circle filled. Note that now the circle looks brighter than in the previous case. The intensity profile shows how the mean value is greater, and the variations are reduced. Nevertheless, this result still shows a noticeable noise within the circle, where the intensity should be constant, and outside the circle, where the intensity should be null.

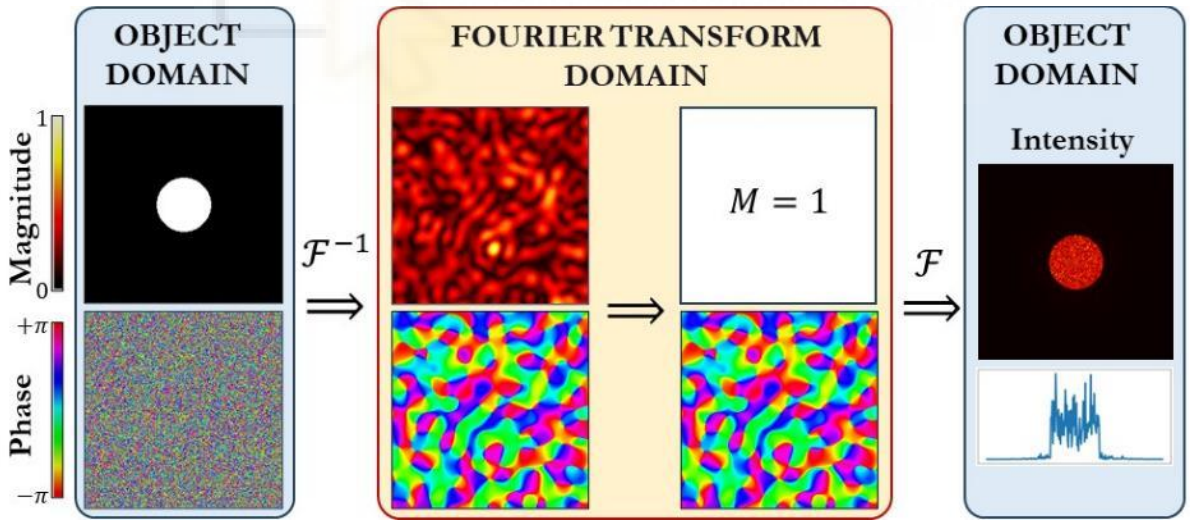

Fig. 6.3. Illustration of the design of a simple FT phase-only CGH with the **random phase approach**. A binary random pattern is multiplied by the original object before calculating the inverse FT. The inset at the right bottom of the hologram reconstruction shows the intensity profile along a horizontal line.

These three approaches have been used all along the Thesis we have used to design phase-only holograms. Figure 6.4 illustrates another example of an arbitrary pattern, an elephant shape,

where the phase-only CGH design has been calculated with the kinoform, with the binary magnitude random approach, and with the random phase approach. Like in the previous example, the kinoform reconstruction reproduces the edges, the binary random magnitude approach provides a better reconstruction, but less intense and noisier than the one provided by the random phase approach.

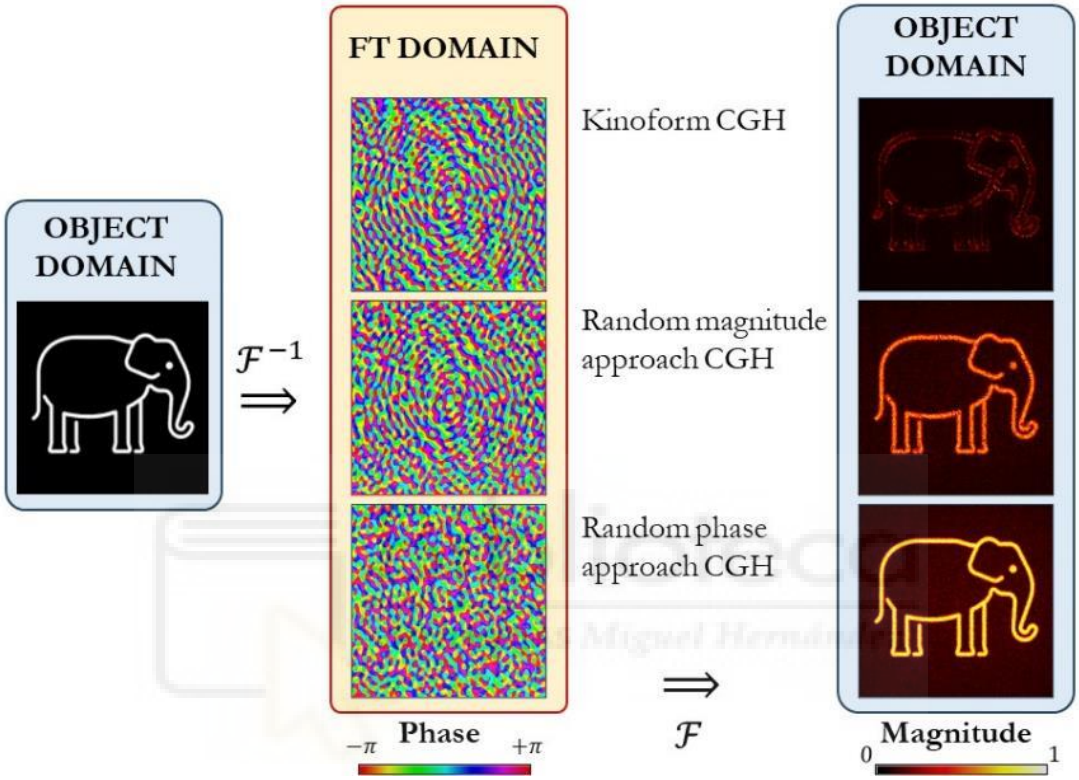

Fig. 6.4. Illustration of the phase-only CGHs for an arbitrary pattern using the kinoform, the binary random magnitude approach and the random phase approach. The right column here gives the magnitude of the hologram reconstruction.

6.2. Iterative Fourier transform algorithms

The previous approaches to designing phase-only CGH can be improved by using iterative Fourier transform algorithm, known as IFTA or the Gerchberg–Saxton (GS) algorithm [Ger-1972]. They consist of performing multiple FT operations and imposing restrictions in the object and the Fourier domain.

6.2.1. Full image IFTA

Figure 6.5 illustrates the IFTA algorithm, here applied to the circle. Starting from the object domain, the inverse FT is calculated. Then the FT domain restriction is imposed, i.e., the magnitude must be one in all pixels. The Fourier transform is then calculated. This is the first

iteration ($N = 1$) of the algorithm. Note that this corresponds to the same situation as in Fig. 6.1 where the kinoform CGH was calculated. There, we only considered the intensity distribution after performing the FT. However, note that there is also a phase distribution, which shows certain circular symmetry but also certain randomness distribution.

This phase pattern is the starting point for the second iteration. Here the second iteration starts by restoring the target magnitude together with the phase of the CGH reconstruction given by the first iteration. The inverse FT is calculated by starting a new iteration. Note how in this second iteration the magnitude and phase distributions in the Fourier domain still very much resemble the Airy pattern, although some distortion is already visible. At the end of this second iteration the magnitude already shows a full circle (the edge-enhanced version in the first iteration was eliminated), but the result is very noisy, as shown in the intensity profile.

The last row in Fig. 6.5 illustrates the result after 25 iterations, which shows a much brighter and uniform magnitude in the circle reconstruction. Note how, even after these 25 iterations, the magnitude and the phase in the FT domain still show distributions that resemble the Airy pattern, mainly in the centre.

Of course, using the phase random approach as the starting point for the IFTA algorithm provides a faster convergence. This case is illustrated in Fig. 6.6. Now the Airy pattern disappears in the first iteration. The magnitude distribution in the FT domain is distributed all over the plane, and at the end of the first iteration, the circle already appears full. Note in the intensity profile how the mean value increases with the number of iterations. The circle reconstruction also appears brighter with the number of iterations. Nevertheless, the reconstruction magnitude shows important fluctuations within the circle and noise outside it.

6.2.2. Windowed IFTA

A method to reduce the noise level in the reconstruction region is a variation of the IFTA algorithm that employs a window [Fet-1997]. Figure 6.7 illustrates this variation called windowed IFTA. A region is defined around the reconstruction area. The regular IFTA algorithm is applied to all the pixels inside the defined window. However, the target magnitude restitution is not applied to the pixels outside the window. Since the hologram reconstruction must be null in this region, this does not influence within the window region. However, allowing the algorithm to have this magnitude freedom in the pixels outside the window makes the noise be expelled from within the window.

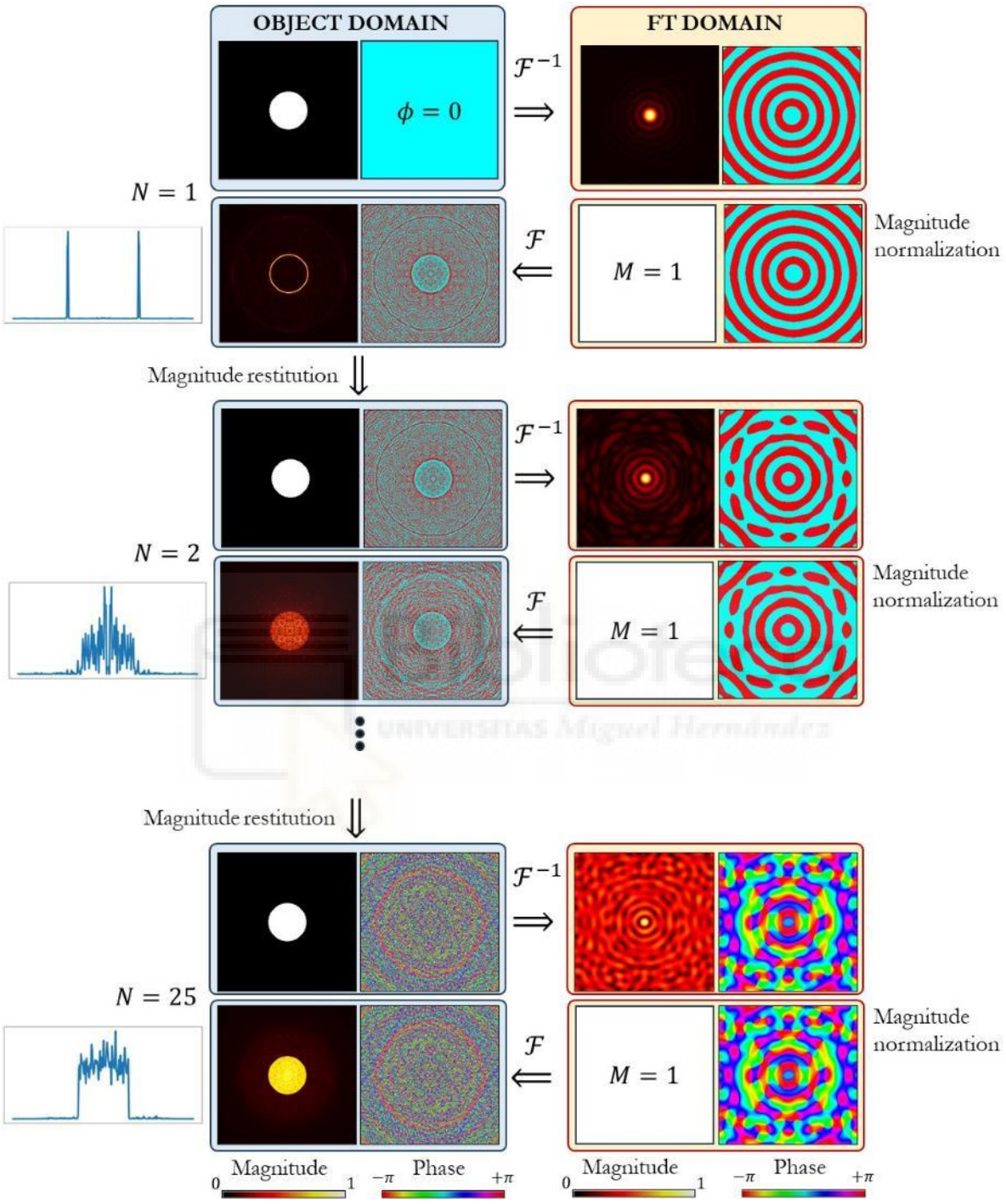

Fig. 6.5. Illustration of the IFTA algorithm for iterations $N = 1, N = 2$ and $N = 25$, and the kinoform CGH is used as a starting point. The intensity profile of the reconstruction is shown on the left.

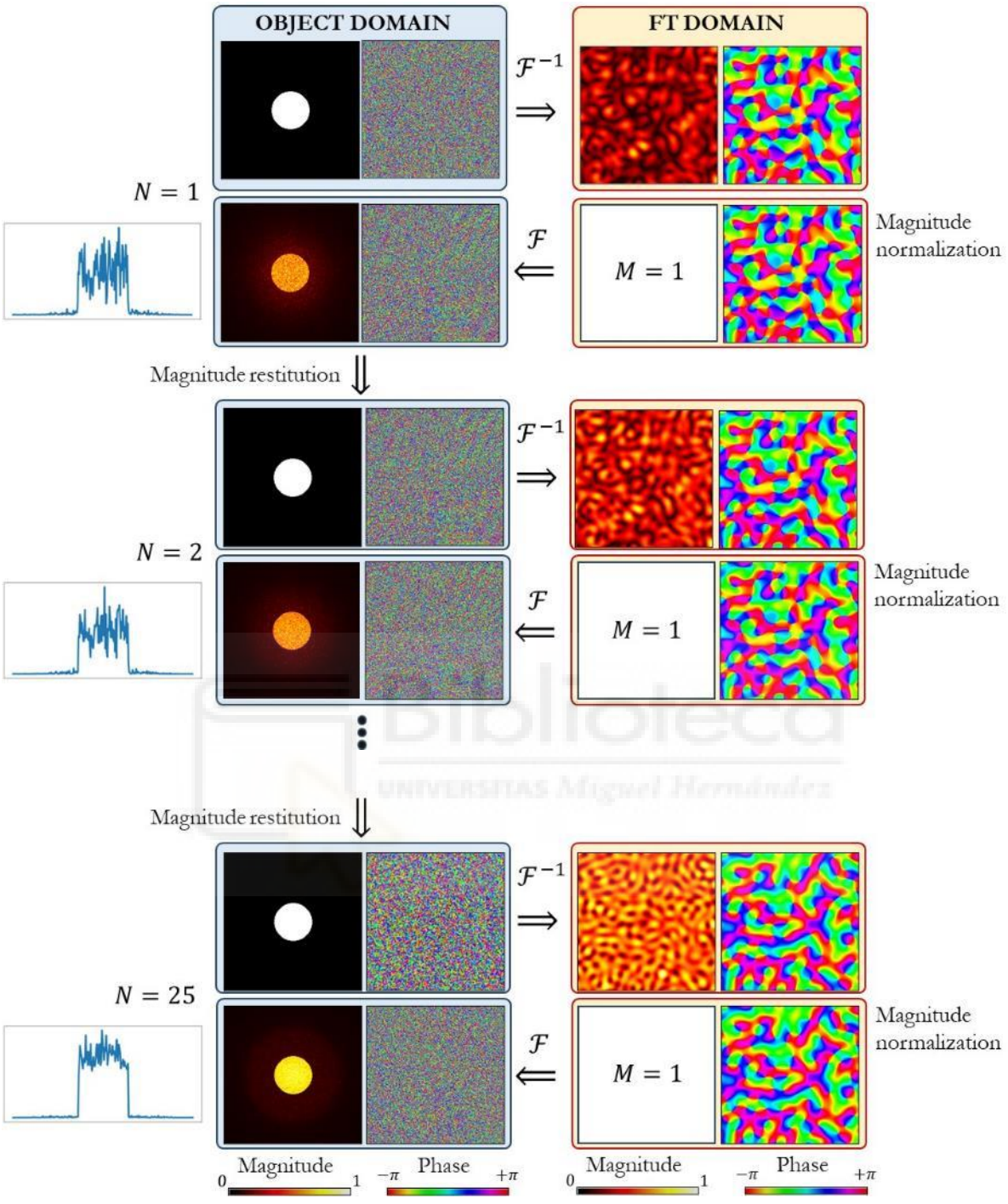

Fig. 6.6. Illustration of the IFTA algorithm for iterations $N = 1, N = 2$ and $N = 25$, and the starting point is the CGH generated with the random phase approach. The intensity profile of the reconstruction is shown on the left.

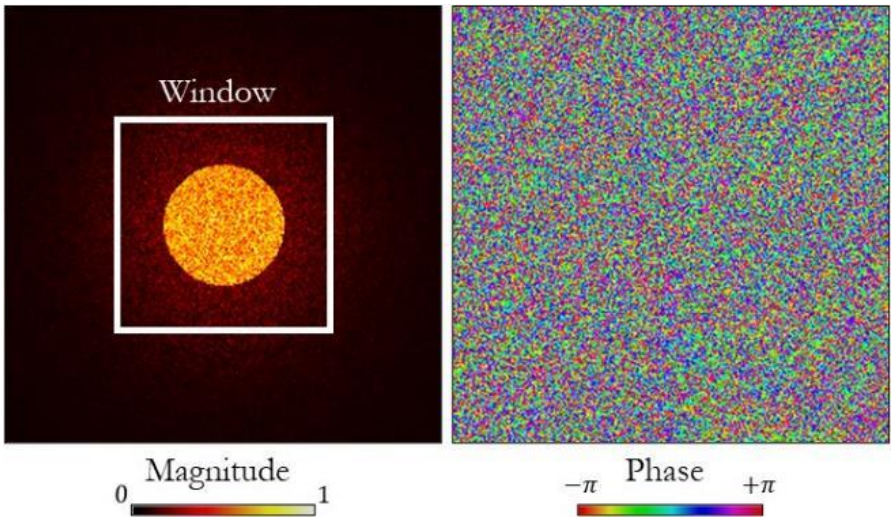

Fig. 6.7. Illustration of the windowed IFTA algorithm. After each iteration, the target magnitude is only restituted in the pixels within the window. Outside the window, the pixels keep the magnitude coming from the previous iteration.

Figure 6.7 shows the magnitude and phase result after the first iteration in Fig. 6.5. The windowed IFTA algorithm restitutes the original object only within the window, while keeping the phase, but also the magnitude, in the pixels outside the window. The result is that the target object, located within the window, is better reproduced than in the standard IFTA algorithm, with the only cost of increasing the noise in the region outside the window.

Figure 6.8 illustrates results comparing the standard IFTA and the windowed IFTA algorithms applied to the case of the circular object. In both cases, a random phase is added in the first step. The figure compares the magnitude retrieved after 1, 5, 10, 20, and 40 iterations. We indicate with variables n_1 and n_2 the number of iterations in each case. These calculations were made in an image of 1024×1024 pixels, drawing a circle with a diameter of 387 pixels. The window is a square of 424×424 pixels. These results illustrate how the windowed IFTA provides a better result within the defined window, where the circle is reproduced with outstanding good fidelity and the background is practically zero. Note how the profile becomes flatter as the number of iterations n_2 increases. Also note the darkness of the inner part of the window where the circle is not reproduced, denoting the absence of noise in this region. The noise in the areas outside the window is greater than in the case of the standard IFTA. Let us finally highlight that the images and the profiles illustrated in Fig. 6.8 correspond to the magnitude, so the intensity result will show a better contrast. Finally, Fig. 6.9 shows the results obtained when cascading a number (n_1) of standard IFTA iterations with n_2 windowed IFTA iterations.

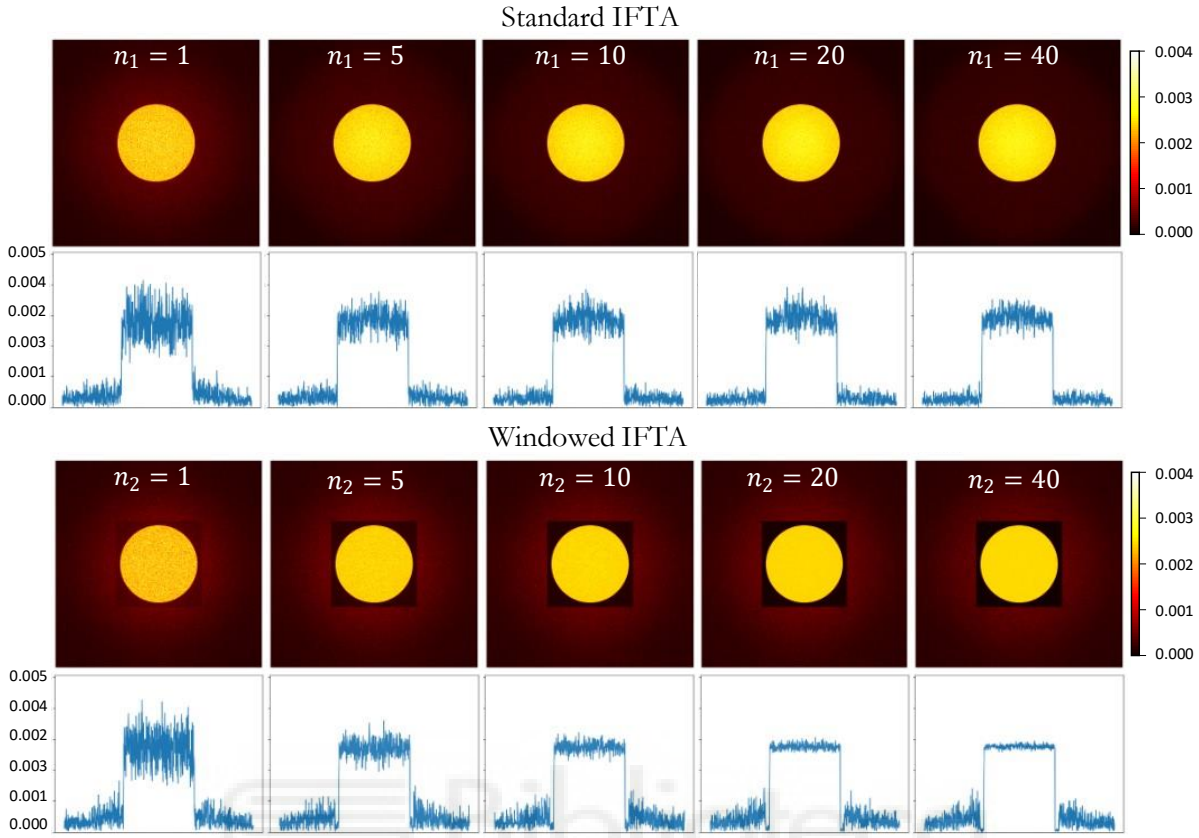

Fig. 6.8. Comparison of the standard IFTA and the windowed IFTA. Results show the magnitude of the hologram reconstruction as well as the profile of this magnitude along the central horizontal line.

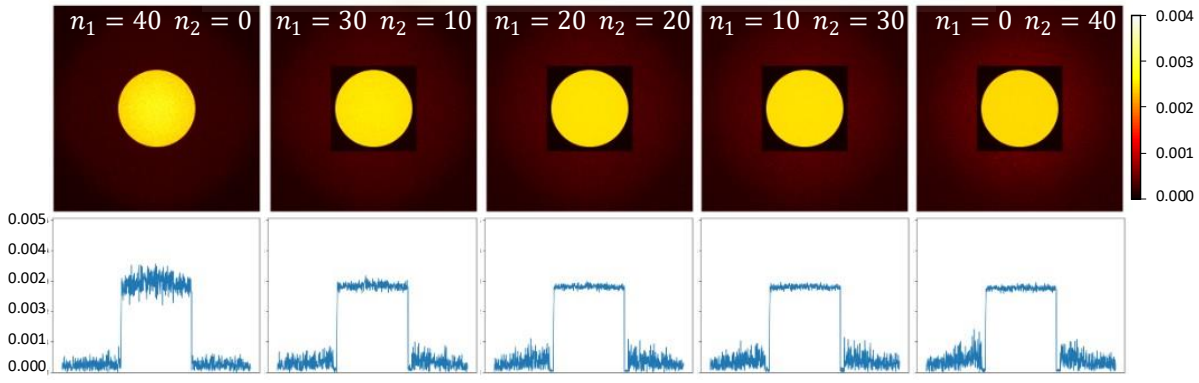

Fig. 6.9. Comparison of the results obtained by applying first a number n_1 of iterations with the standard IFTA followed by a number n_2 of windowed IFTA iterations.

To have a quantitative comparison of the resulting holograms, we have considered two metrics that compare the original object (the circle with unit transmission) and the different reconstructions shown in Figs. 6.8 and 6.9. The first metric is the difference between the reconstructed circle and the original one, hence, it is called DIF metric. Let $g_{obj}(x,y)$ and

$g_{rec}(x,y)$ denote respectively the magnitude of original object and the magnitude of the reconstruction. The difference between the two images is calculated as the quadratic mean error in the intensity distribution, i.e.:

$$DIF = \sqrt{\sum_{x,y\in W}\left\{\frac{\left[g_{obj}(x,y)\right]^2}{\langle i_{obj}\rangle} - \frac{[g_{rec}(x,y)]^2}{\langle i_{rec}\rangle}\right\}^2}, \tag{6.1}$$

where the summation is performed only in the pixels within the defined window, $x,y \in W$ (in this case a square window of 424×424 pixels), and where each signal is normalized to have the same average intensity within the window W by dividing each one in Eq. (6.1) by the values,

$$\langle i_{obj}\rangle = \sum_{x,y\in W}\left[g_{obj}(x,y)\right]^2, \qquad \langle i_{rec}\rangle = \sum_{x,y\in W}[g_{rec}(x,y)]^2\,. \tag{6.2}$$

This normalization takes into account about the change in the mean intensity of the original and reconstructed signal caused by the magnitude discard used in the IFTA algorithms. Figure 6.10 shows the difference intensity value at each pixel within the window (each summand in Eq. (6.1)) for the different combinations of the standard and windowed IFTA algorithm shown in Figs. 6.8 and 6.9. It shows that the difference is large when only one IFTA iteration is applied, but it is reduced as the number of iterations increases. The last row corresponds to different combinations of standard and windowed IFTA iterations.

A second metric usually applied to quantify the quality of the CGH reconstruction is the signal-to-noise ratio (SNR) between the intensity within the window and outside the window, defined as:

$$SNR = \frac{\sum_{x,y\in W}[g_{rec}(x,y)]^2}{\sum_{x,y\notin W}[g_{rec}(x,y)]^2}. \tag{6.3}$$

While the DIF metric provides quantitative information about the fidelity of the CGH reconstruction compared to the desired object, the SNR metric provides information about how efficient the reconstruction is.

Figure 6.11 provides numerical calculations of these two metrics for the simulation results presented in Figs. 6.8 and 6.9. The graph in Fig. 6.11(a) shows that the metric difference diminishes as the number of iterations increases, giving a better result with the windowed IFTA than with the standard IFTA. The graph in Fig. 6.11(b) shows how the SNR increases with the number of iterations for the standard IFTA while it takes a more constant lower value for the windowed IFTA.

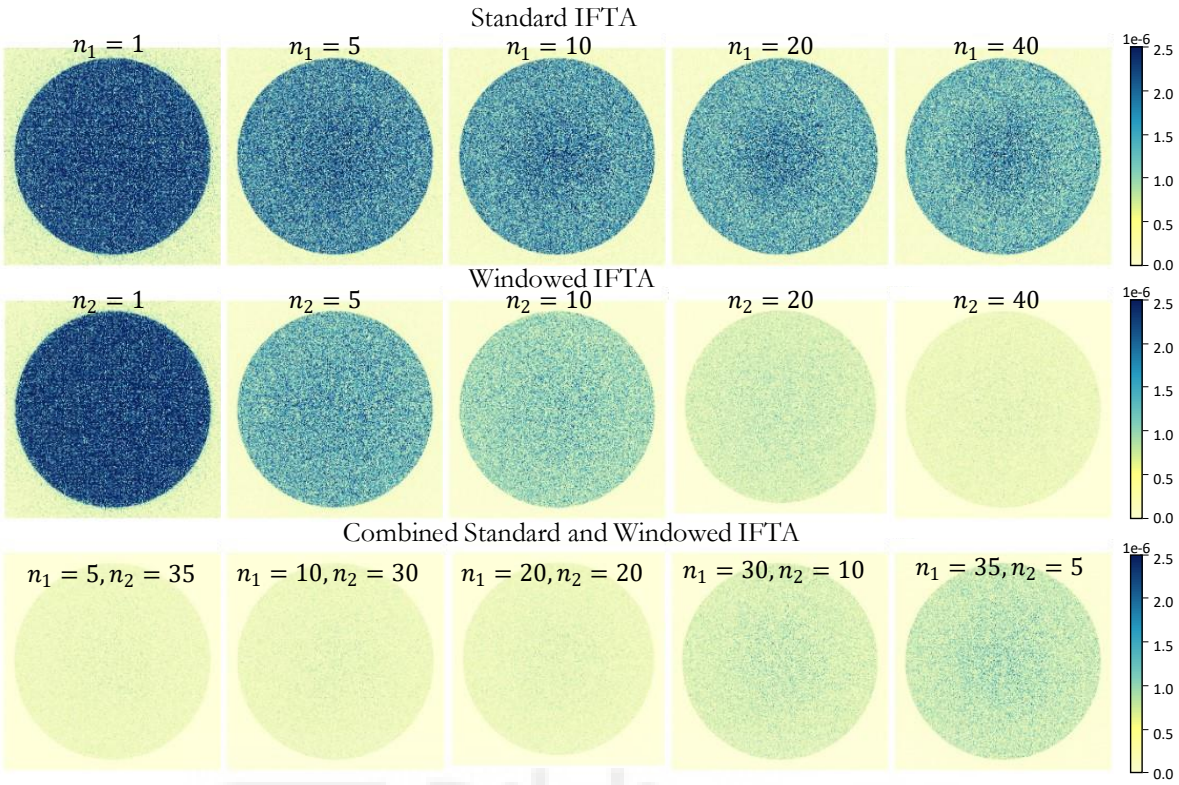

Fig. 6.10. Computed intensity difference for different IFTA combinations.

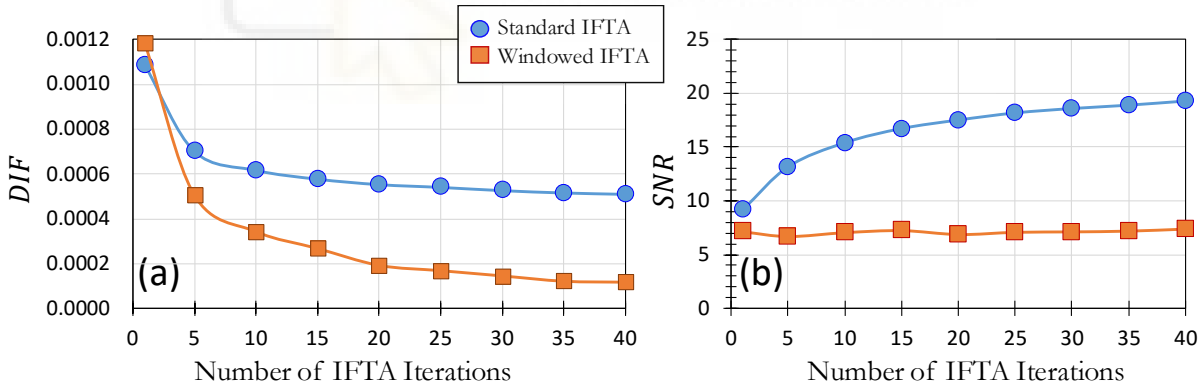

Fig. 6.11. Quality metrics comparison for the CGH reconstruction of the circle object calculated with the standard and with the windowed IFTA. (a) DIF, (b) SNR.

Figure 6.12 provides equivalent results when the total number of iterations is kept constant, equal to 40 iterations, but the standard and windowed IFTA iterations are combined. As it can be seen, combining the two IFTA provides a trade-off between having a good fidelity hologram reconstruction (low DIF) achieved with the windowed IFTA, while maintaining the higher SNR provided by the standard IFTA.

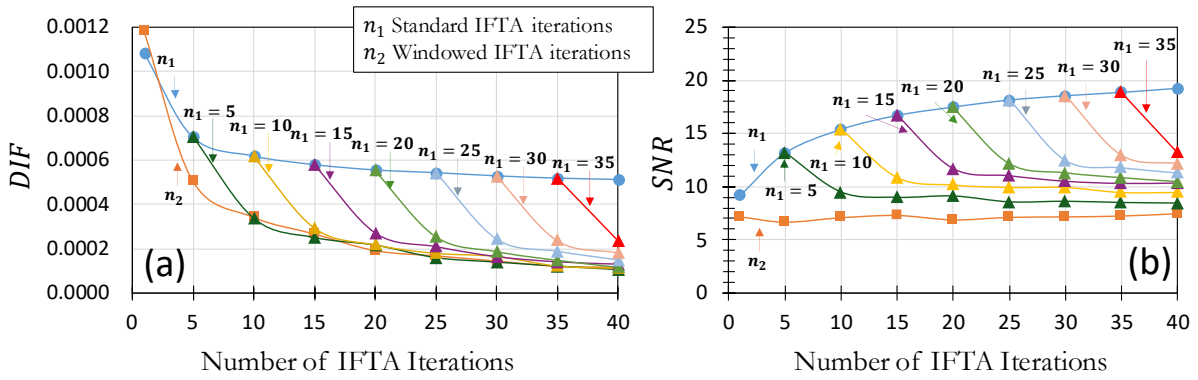

Fig. 6.12. Quality metrics comparison (a) DIF, (b) SNR versus the total number of iterations, for the standard IFTA (n_1 iterations) and for the windowed IFTA (n_2 iterations). n_1 changing from 5 to 35 represents the number of iterations computed by the standard IFTA in the combined IFTA curves (indicated as triangles).

6.3. Multiple realizations and temporal integration

Another approach useful to reduce the noise levels, which can be applied when using SLMs, is based on the temporal integration of several holograms generated [Ama-1995]. The idea here is to avoid the use of the IFTA algorithm, which is computationally consuming. Instead, it is substituted by several phase-only CGHs calculated with different realizations of the random phase approach (Section 6.1.3). Each realization requires a single Fourier transform calculation. Calculating the same CGH with different realizations of the random phase, generates the same pattern in all cases, but a different noise in each case. Therefore, integrating several of these reconstructions results in an averaging of the intensity of the reconstruction that leads to a reduction of the noise levels. Since the phase-only hologram can be changed in real time in the SLM, these different realizations of the same hologram can be displayed sequentially, and a slow detector (for instance the human eye) can make the temporal integration.

Figure 6.13 presents some simulations of this process. These results are calculated by designing t simple phase-only holograms of the same object, the circle, with different realizations of the added random phase noise. For each realization, the reconstructed intensity $i_t(x,y)$ is calculated, and then they are averaged. The images in Fig. 6.13 show the square root of this averaged intensity, thus they illustrate the corresponding magnitude and are comparable to the results presented in the previous sections. As seen in the images in Fig 6.13, and in the corresponding profiles, as the number of realizations increases, the noise levels within the circle are reduced, leading to a flat profile. However, there is also a significant noise around the circle.

Figures 6.14 and 6.15 show respectively the difference between the target and the

reconstructed intensity, and the DIF and SNR parameters as the number (t) of averaged realizations increases. The results show that the temporal integration technique provides results with DIF and SNR comparable to those obtained with the windowed IFTA.

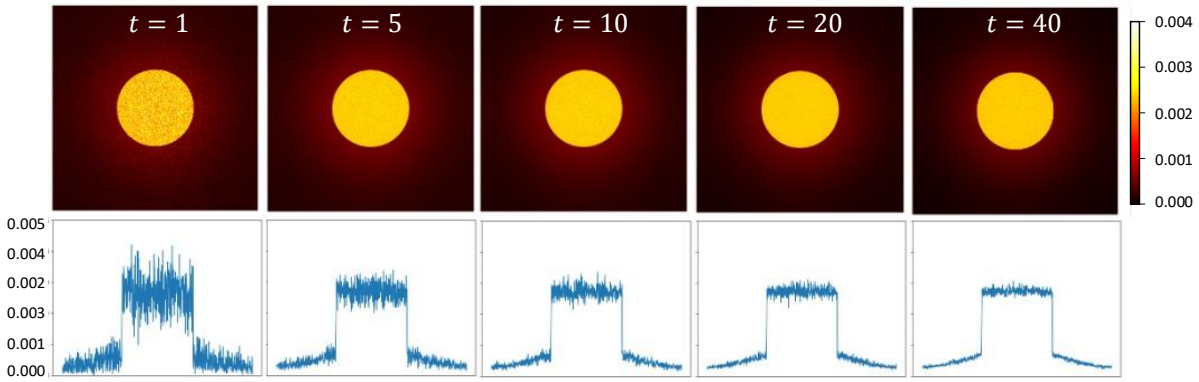

Fig. 6.13. Comparison of the results obtained with the multiple temporal integration method for different number of average realizations.

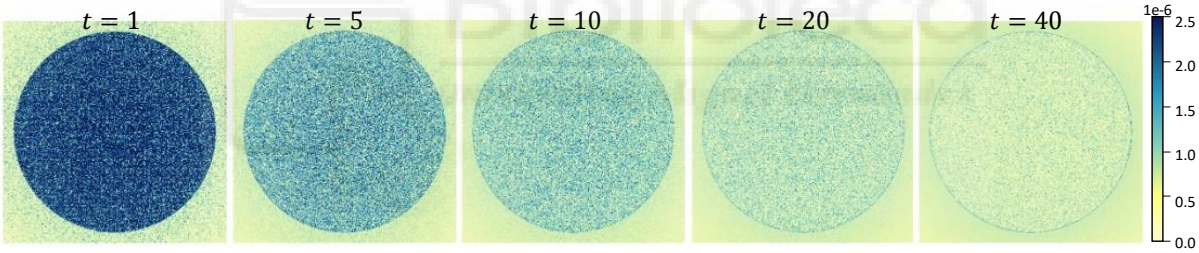

Fig. 6.14. Computed intensity difference for the temporal integration method with different t number of average realizations.

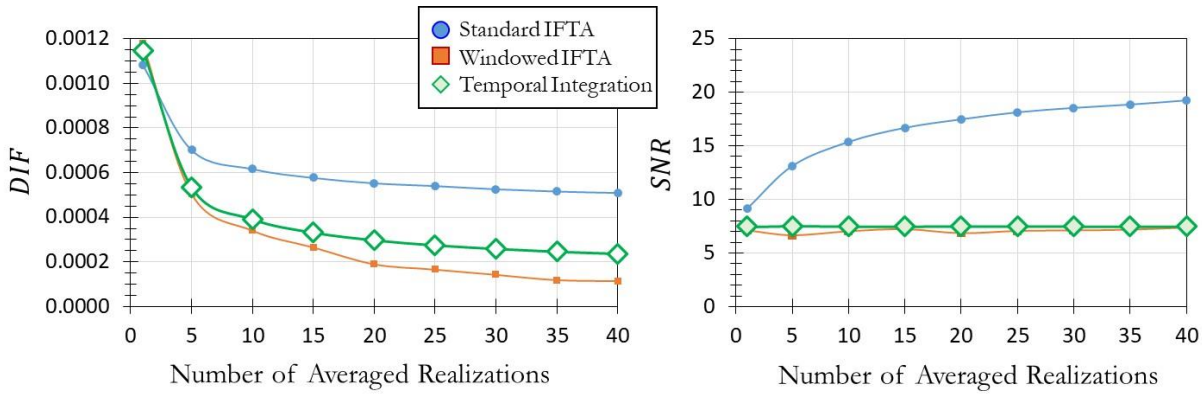

Fig. 6.15. Comparison of the quality metrics for the CGH reconstruction for the circle object calculated with the temporal integration method and compared to the results with the standard and with the windowed IFTA. (a) DIF, (b) SNR.

6.4. The triplicator hologram

Here we go a step beyond the optimum triplicator grating profile presented in Chapter 5 and apply it to a phase-only Fourier transform (FT) hologram. We extend the phase profile in Eq. (5.7) to an arbitrary phase hologram. Let us assume a phase-only Fourier transform (FT) hologram $\phi(\mathbf{x})$, calculated according to any of the methods outlined before, and let us assume that its reconstruction is the function $g(\mathbf{x}') = \mathcal{F}\left[e^{i\phi(\mathbf{x})}\right]$. Here $\mathbf{x} = (x, y)$ and $\mathbf{x}' = (x', y')$ denote the spatial coordinates at the hologram and at the reconstruction planes. Since the optical Fourier transform is assumed to be obtained with a lens having a focal length f, then $\mathbf{x}' = \lambda f \mathbf{u}$, with $\mathbf{u} = (u, v)$ denoting the hologram spatial frequencies.

Now, we consider a new triplicator phase function $\varphi_h(\mathbf{x})$ that is calculated as

$$\varphi_h(\mathbf{x}) = \pi + \arctan[a\cos(\phi(\mathbf{x}) + \gamma x)]. \tag{6.4}$$

This is exactly the same formula as in Eq. (5.7), but the linear term $2\pi x/p$ used there as the argument of the cosine function is now changed to include the hologram's phase function $\phi(\mathbf{x})$. The term $\gamma x = 2\pi x/p$ is kept as an additional linear phase [Wyr-1990] that is used to spatially separate the different terms in the hologram reconstruction, as will be shown later.

A well-known result from computer-generated holography is that the modification of the phase profile of a phase-only hologram leads to a Fourier series expansion due to the phase 2π periodicity [Mor-1995], so the modified hologram can be expressed as

$$e^{i\varphi_h(\mathbf{x})} = \sum_{m=-\infty}^{+\infty} c_m e^{im[\phi(\mathbf{x})+\gamma x]}, \tag{6.5}$$

where c_m are the Fourier coefficients of the phase profile transformation, i.e. they are given by Eq. (5.2).

Since the phase transformation in Eq. (6.4) is given by the triplicator profile, the most important coefficients are c_{+1}, c_0 and c_{-1}, the three having the same energy and together carrying 93% of the total energy. In [Mar-2019] it was demonstrated that these three terms are related as $c_{\pm 1} = ic_0$ and $c_0 \cong 0.55$. Therefore, if the rest of the terms are ignored, Eq. (6.5) can be approximated as

$$e^{i\varphi_h(\mathbf{x})} \cong ic_0 e^{i\phi(\mathbf{x})} e^{i\gamma x} + c_0 + ic_0 e^{-i\phi(\mathbf{x})} e^{-i\gamma x}. \tag{6.6}$$

This relation shows that the three main terms of the triplicator hologram are the original phase hologram $\phi(\mathbf{x})$, its complex-conjugated version, and a DC (constant) component. Note that the

linear phases $e^{\pm i\gamma x}$ in the ± 1st orders have opposite signs.

The corresponding Fourier transform is therefore given by

$$\mathcal{F}\left[e^{i\varphi_h(\mathbf{x})}\right] \cong ic_0 g(\mathbf{x}' - \mathbf{x}'_0) + c_0\delta(\mathbf{x}') + ic_0 g^*(-\mathbf{x}' - \mathbf{x}'_0). \tag{6.7}$$

This shows that the triplicator hologram reconstruction provides the original function $g(\mathbf{x}')$ shifted laterally a distance $\mathbf{x}'_0 = \lambda f\gamma/2\pi$. The linear term γx added in Eq. (6.4) makes the direct reconstruction appear centred at $\mathbf{x}' = \mathbf{x}'_0$. But another inverted and complex-conjugated version $g^*(-\mathbf{x}' - \mathbf{x}'_0)$ appears in the hologram reconstruction, now centred at $\mathbf{x}' = -\mathbf{x}'_0$ due to the opposite sign of the linear phase affecting the $m = -1$st order in Eq. (6.6). Finally, the DC order results in a delta function centred on axis $(\mathbf{x}' = 0)$.

Figure 6.16 illustrates some experimental results obtained with the Exulus SLM. Here a phase-only hologram is designed using the random phase approach explained in subsection 6.1.3, here applied to an object with the shape of a lotus flower. A linear phase is added to the original hologram calculated from a centred object (or alternatively the hologram is calculated from a shifted object). The phase of a lens is added to the hologram, so there is no need for an additional physical lens to obtain the hologram reconstruction. Figure 6.16(a) shows the experimental result when the original hologram $\phi(\mathbf{x})$ is displayed, which shows the effective generation of the flower. Figure 6.16(b) shows the equivalent result when the triplicator profile $\varphi_h(\mathbf{x})$ is displayed on the SLM. Now the three terms described in Eq. (6.7) are observed. Note that the inverted version of the flower has the same energy and direct reconstruction. The zero-order provides a bright spot focalization, corresponding to the delta function in Eq. (6.7).

Finally, it is shown how these triplicator holograms can be used to easily produce optical convolver and correlator elements. To show this property, a new phase function is calculated as $[\varphi_h(\mathbf{x}) - \phi(\mathbf{x})]_{mod2\pi}$. Therefore, when this new phase hologram is displayed on the SLM, the resulting phase function can be written as

$$e^{i\varphi_h(\mathbf{x})}e^{-i\phi(\mathbf{x})} \cong ic_0 e^{i\gamma x} + c_0 e^{-i\phi(\mathbf{x})} + ic_0 e^{-i2\phi(\mathbf{x})}e^{-i\gamma x}, \tag{6.8}$$

so its Fourier transform is given by the convolution of $g^*(-\mathbf{x}')$ with the three terms in Eq. (6.7), i.e.:

$$\mathcal{F}\left[e^{i\varphi_h(\mathbf{x})-i\phi(\mathbf{x})}\right] \cong ic_0\delta(\mathbf{x}' - \mathbf{x}'_0) + c_0 g^*(-\mathbf{x}') + ic_0[g^* * g^*](-\mathbf{x}' - \mathbf{x}'_0). \tag{6.9}$$

The corresponding experiment is shown in Fig. 6.16(c). The first term in Eq. (6.9) is the autocorrelation $[g \star g](\mathbf{x}' - \mathbf{x}'_0)$, which takes the form of a delta function since the holograms are

phase-only functions. This is shown as the bright spot that appears on the right part of Fig. 6.16(c). The central term in Eq. (6.9) reproduces the inverted object, as seen in the central part in Fig. 6.16(c). Finally, the last term of Eq. (6.9) is the autoconvolution of the inverted object, which in this case takes the shape of a broad noisy cloud of light.

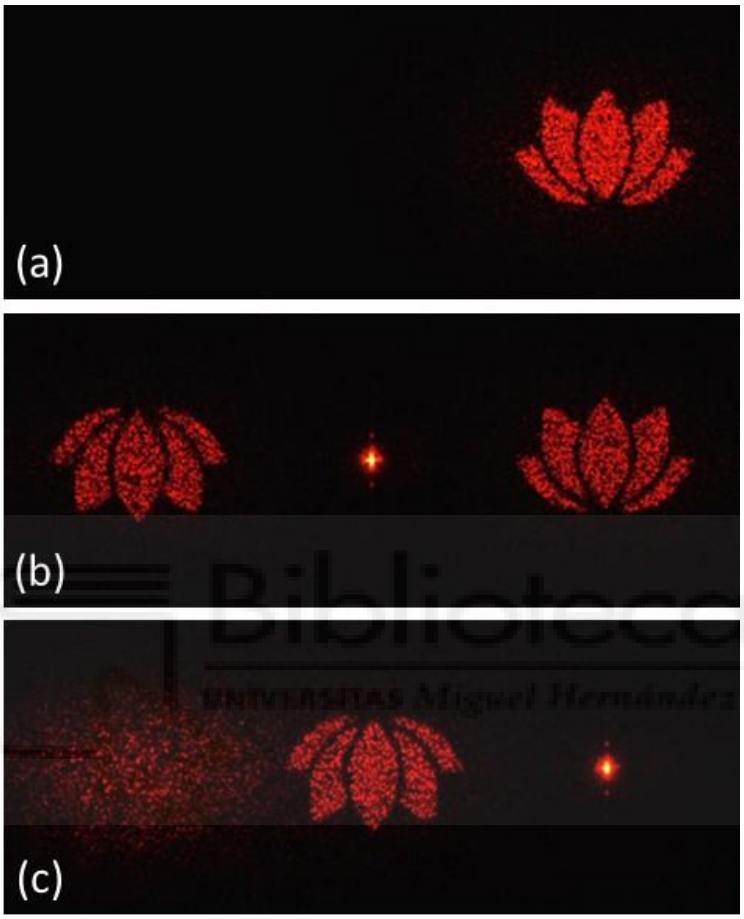

Fig. 6.16. Comparison of the hologram reconstruction of (a) a phase-only hologram $\boldsymbol{\phi(x)}$ designed with the random phase approach, (b) the corresponding triplicator hologram $\boldsymbol{\varphi_h(x)}$, and (c) the combination $[\boldsymbol{\varphi_h(x) - \phi(x)}]_{mod2\pi}$. In all three cases, a lens phase function is added at the end to focus the beam onto the camera detector.

Python Code

The codes used for calculation of the IFTA holograms included in this chapter are included in Chapter 9, where these methods here are extended to RGB images and the corresponding RGB holograms. Here we provide only the code to simulate the temporal integration presented in Section 6.3.

```
"""
Simulates the temporal integration of multiple CGHs

"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft2, fftshift,ifft2,ifftshift
import time

start_t=time.time()
Iterations = [1, 5, 10, 15, 20, 25, 30, 35, 40]
# Read the path of the original image.
original_path2="C:/Users/Laboratorio/MakeHologram/FFT_CGH_thesis/SNR_Diff/Ch6/One circle_1024.png"
original=plt.imread(original_path2)
# Demonstrate the object in the red channel.
Target=original[:,:,0]
h,w=Target.shape
l = 300
c_w,c_h = w//2,h//2
lh,lw = h-2*l,w-2*l # The size of the window.
original_nor=Target/np.sum(Target[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2])

count = 0
for count in Iterations:
    Intensity = 0
    for n in range(count):
        # Creates the array of random numbers [0-1]
        # RandomImage = Image.new("L", (h,w), "black")
        Rand = np.random.uniform(0,1,(h,w))
        RandomPhase = np.exp(1j*2*np.pi*Rand)

        Field = Target * RandomPhase
        # FOURIER TRANSFORM
        FT_Field = fftshift(fft2(fftshift(Field)))
        FT_Magnitude = np.abs(FT_Field)
        FT_Phase = np.angle(FT_Field)
        # MAGNITUDE = 1
        FT_Field = np.exp(1j*FT_Phase)
        # INVERSE FOURIER TRANSFORM
        Field = ifftshift(ifft2(ifftshift(FT_Field)))
        Magnitude = np.abs(Field)
        # Sum over all the intensity.
        Intensity += np.square(Magnitude)
    # Average the total intensity and calculate magnitude.
    I_average = Intensity/count
    M_average = np.sqrt(I_average)
    # SNR in the window to the outside noise.
    SNR_r = np.sum(I_average[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2])/(np.sum(I_average)-
np.sum(I_average[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]))
    # Difference in the window with respect to the original intensity.
    I_average_nor=I_average/np.sum(I_average[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2])
    diff_r = original_nor[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]-I_average_nor[c_h-lh//2:c_h+lh//2,
c_w-lw//2:c_w+lw//2]
    D_r = np.sqrt(np.sum((diff_r)**2))

    print(f"t {count}: SNR={SNR_r} Diff={D_r}")
    # Show the difference.
    plt.figure()
    plt.imshow(abs(diff_r),vmax=3.3e-6,cmap="YlGnBu")
    plt.colorbar()
    plt.axis("off")
    plt.savefig(f"I Diff for t_{count} avr.png", dpi=300, bbox_inches='tight')
    plt.close()
    # Show the magnitudes.
    plt.figure()
    plt.imshow(M_average,vmax=0.004,cmap='hot')
    plt.colorbar()
    plt.axis('off')
    plt.savefig(f"Rec_M _count_{count} avr.png", dpi=300, bbox_inches='tight')
    plt.close()
    # Plot a profile.
```

```
    row = h//2
    profile_M = M_average[row,:]

    plt.figure()
    plt.plot(profile_M)
    plt.gca().xaxis.set_visible(False)
    plt.ylim(0,0.005)
    plt.savefig(f"Rec_M profile_{count} avr.png", dpi=300, bbox_inches='tight')
    plt.close()
#---------------------------------
end_t=time.time()
print(f"Time consuming {end_t-start_t}s")
```

7. Holograms displayed on pixelated SLMs

SLMs are popular devices for displaying programmable holograms. However, the performance will be affected by the pixelated structure of these devices. In this chapter, we use the Fourier transform theory to analyse the effects of the SLM's pixelated structure when displaying a hologram. As will be shown, this Fourier approach provides a clear physical insight into the obtained diffraction pattern.

First, we consider no hologram displayed on the SLM and analyse the diffraction pattern stemming from the device alone [Gao-2024b, Gao-2025]. By regarding the device as an amplitude grating, we demonstrate that the SLM pixelation yields diffraction orders and brings a sinc envelope that affects the orders' weights. This theoretical prediction is experimentally verified.

Second, the case where an arbitrary Fourier transform hologram is displayed on the SLM is considered. Multiple replicas of the hologram are then obtained in the Fourier plane. Using the Fourier transform approach they are interpreted as the convolution between the target hologram and the diffraction orders stemming from the SLM itself. We show that the weights of these

replicas are given by the sinc envelope brought by the pixelated structure.

7.1. Fourier transform analysis

The general convolutional approach applied here provides physical insights into how the diffraction pattern is affected [Arr-1999]. For simplicity, we model the SLM as a 1D amplitude grating $g_{SLM}(x)$ of a period equal to the pixel pitch (Δ), with amplitude 1 in the effective size of the pixels and amplitude 0 in the dead zones [Gao-2024b, Gao-2025]. Considering Section 2.5, it is straightforward to see that its generalization for the two-dimensional SLM case is the product of two such functions along the x and y coordinates. The pixel effective size where light modulation is possible is defined as $a\Delta$, where $a < 1$. The device fill factor (F) is defined as the ratio of the effective pixel area $(a\Delta)^2$ and the entire pixel area Δ^2, hence $F = a^2$. In the experimental results presented in this chapter the Exulus-HD1 SLM has been used. As mentioned in Chapter 4, this device has a pixel size of $\Delta = 6.4\ \mu m$, and its fill factor is $F = 93\%$, thus $a = 0.964$.

As illustrated in Fig. 7.1 (a), the SLM's pixelated structure can be described as the convolution of a rectangle function of width $a\Delta$ and a comb function that sets the center of each pixel at positions separated by a distance Δ, i.e.

$$g_{SLM}(x) = \mathrm{rect}\left(\frac{x}{a\Delta}\right) \otimes \sum_{n=-\infty}^{\infty} \delta(x - n\Delta), \tag{7.1}$$

where $\mathrm{rect}(x) = 1$ if $|x| \leq 1/2$, and $\mathrm{rect}(x) = 0$ elsewhere, and $n = 0, \pm 1, \pm 2, ..$, is the integer index denoting the diffraction order generated by the SLM's pixelated structure. The convolution operation in Eq. (7.1) replicates the rectangle function centered on each delta function of the summatory.

The diffraction pattern generated in the Fourier plane is given by the Fourier transform ($\mathcal{F}\{\cdot\}$) of the function in Eq. (7.1). Using the properties desribed in Section 2.3, this can be written as:

$$G_{SLM}(u) = \mathcal{F}\{g_{SLM}(x)\} = a\,\mathrm{sinc}(ua\Delta) \cdot \sum_{n=-\infty}^{\infty} \delta\left(u - \frac{n}{\Delta}\right) = \sum_{n=-\infty}^{\infty} a\,\mathrm{sinc}(an)\,\delta\left(u - \frac{n}{\Delta}\right), \tag{7.2}$$

where u stands for the spatial frequency and $\mathrm{sinc}(x) \equiv \sin(\pi x)/(\pi x)$. This equation illustrates that the grid pattern of the SLM's pixelated structure (without any function displayed on the device)

generates a set of diffraction orders, given by the delta functions $\delta\left(u-\frac{n}{\Delta}\right)$, each one weighted by an amplitude coefficient $a\,\mathrm{sinc}(an)$. Thus, the intensity of each of these diffraction orders is given by

$$I_n = a^2\,\mathrm{sinc}^2(an). \tag{7.3}$$

Note that the limited size of the SLM is being ignored (the summation in Eq. (7.1) extends infinitely). The finite size of the SLM could be accounted for by multiplying $g_{SLM}(x)$ with a wide **rect** function in Eq. (7.1). The resulting effect in the Fourier transform is to convert the delta functions in Eq. (7.2) into very narrow **sinc** functions, without altering their amplitudes. Therefore, for simplicity, we do not consider this effect.

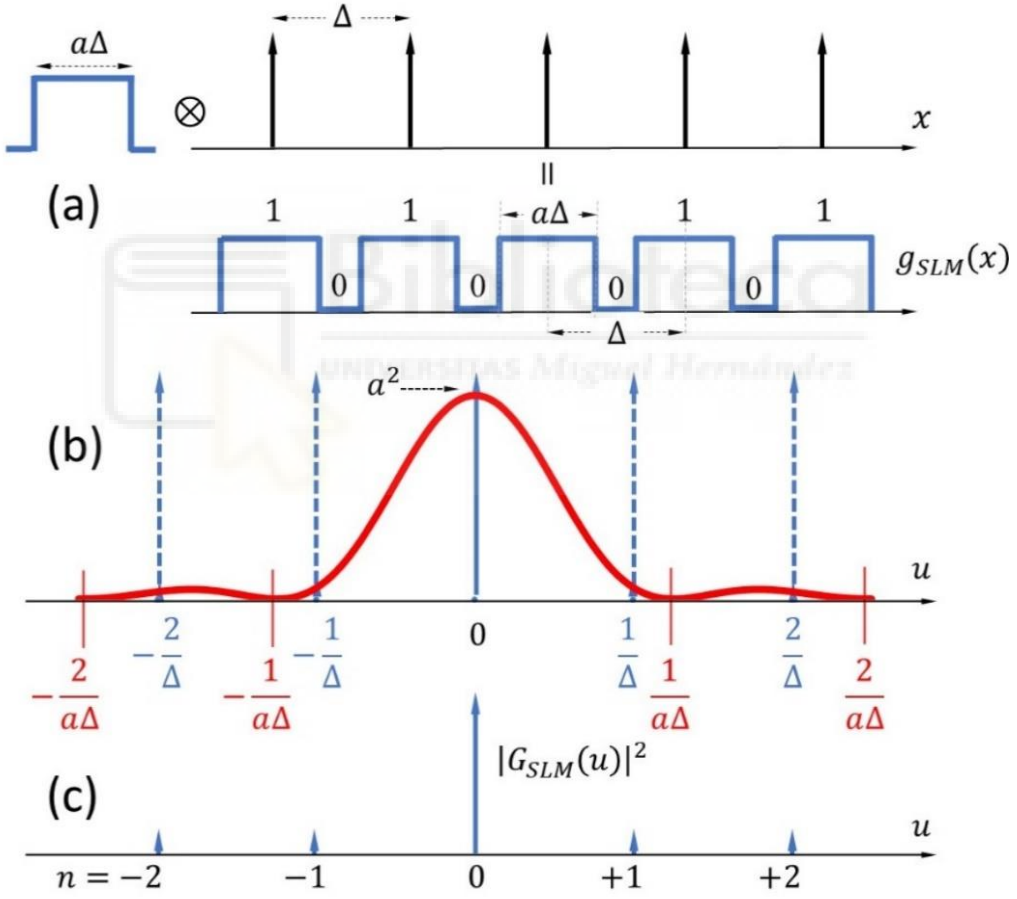

Fig. 7.1. Analysis of the SLM's pixelated structure [Gao-2024b, Gao-2025]: (a) Representation of the SLM function $g_{SLM}(x)$ as the convolution of a rectangle function with a comb function, Eq. (7.1), and representation of its Fourier transform $G_{SLM}(u)$: (b) Square of the envelope **sinc** function (red curve) and **comb** function (dashed delta functions), and (c) multiplication of the above two terms to provide the intensity distribution of the diffraction orders generated by the SLM's pixelated structure.

7.2. The sinc envelope function

Figures 7.1(b) and 7.1(c) illustrate the physical interpretation of Eqs. (7.2) and (7.3), where we have drawn the intensity of the envelope function $a^2\,\mathrm{sinc}^2(ua\Delta)$ (red curve) together with the comb function (dashed lines denoting the delta functions), and their multiplication results in the weighted solid blue lines [Gao-2024b, Gao-2025].

The diffraction pattern caused by the SLM's pixelated structure consists of discrete diffraction orders located at spatial frequencies $u = n/\Delta$, where $n = 0, \pm1, \pm2, \ldots$ The zeros of the envelope function $\mathrm{sinc}^2(ua\Delta)$ appear at spatial frequencies $u = \pm\frac{1}{(a\Delta)}, \pm\frac{2}{(a\Delta)}, \ldots$. Therefore, the first zero lies at a value of u larger than the position $1/\Delta$ of the first delta function $(n = 1)$. The greater the fill factor, the closer to one is parameter a, and the closer is the zero of the envelope function to the first delta function, thus almost cancelling the diffraction order. The limit $a = 1$ corresponds to a display without dead zones, where the minimas of the sinc function coincide with the positions of the delta functions. Consequently, there would not be diffraction orders, only a central 0th order. Since the fill factor of our Exulus-HD1 device is $F = a^2 = 0.93$ with $a = 0.964$, considering Eq. (7.3) where $a^2\,\mathrm{sinc}^2(a) = 0.00028$ and regarding the SLM as a 2D amplitude grating, we expect to observe a bright zero-order with relative intensity $I_0 = a^4 = 86.36\%$ surrounded by four dim lateral 1st diffraction orders with relative intensity $I_1 = 0.03\%$. The experimental result is shown in Fig. 7.2(b), which displays the diffraction pattern captured by the camera when a uniform gray level is encoded on the SLM.

7.3. The multiple replicas

Now, let us consider an arbitrary phase function $g_d(x)$ encoded on the SLM, whose Fourier transform is $G_d(u) = \mathcal{F}\{g_d(x)\}$. The displayed function becomes:

$$g(x) = \mathrm{rect}\left(\frac{x}{a\Delta}\right) \otimes \left(g_d(x) \sum_{n=-\infty}^{\infty} \delta(x - n\Delta) \right). \tag{7.4}$$

As explained in Section 2.4, the term in brackets describes the sampling of the phase profile $g_d(x)$, at the center of each pixel $x = n\Delta$ $(n = 0, \pm1, \pm2, \ldots)$. Then, the convolution with the rectangle function outside the brackets assigns the sampled value to the effective pixel length [Gao-2024b, Gao-2025].

The Fourier transform of the displayed function in Eq. (7.4) is given by

$$G(u) = \mathcal{F}\{g(x)\} = a\,\mathrm{sinc}(a\Delta u) \cdot \left(G_d(u) \otimes \sum_{n=-\infty}^{\infty} \delta\left(u - \frac{n}{\Delta}\right) \right). \tag{7.5}$$

Figure 7.2(a) illustrates the physical interpretation of Eq. (7.5). The convolution of $G_d(u)$ with the comb function implies that $G_d(u)$ is replicated centred at the spatial frequencies $u = n/\Delta$. This is schematically represented in Fig. 7.2(a) by the yellow blocks $G_d(u)$ drawn on top of each delta function. The envelope sinc function (red curve) multiplies the result of this convolution and gives the weight of each replica (indicated by the shadow gray regions).

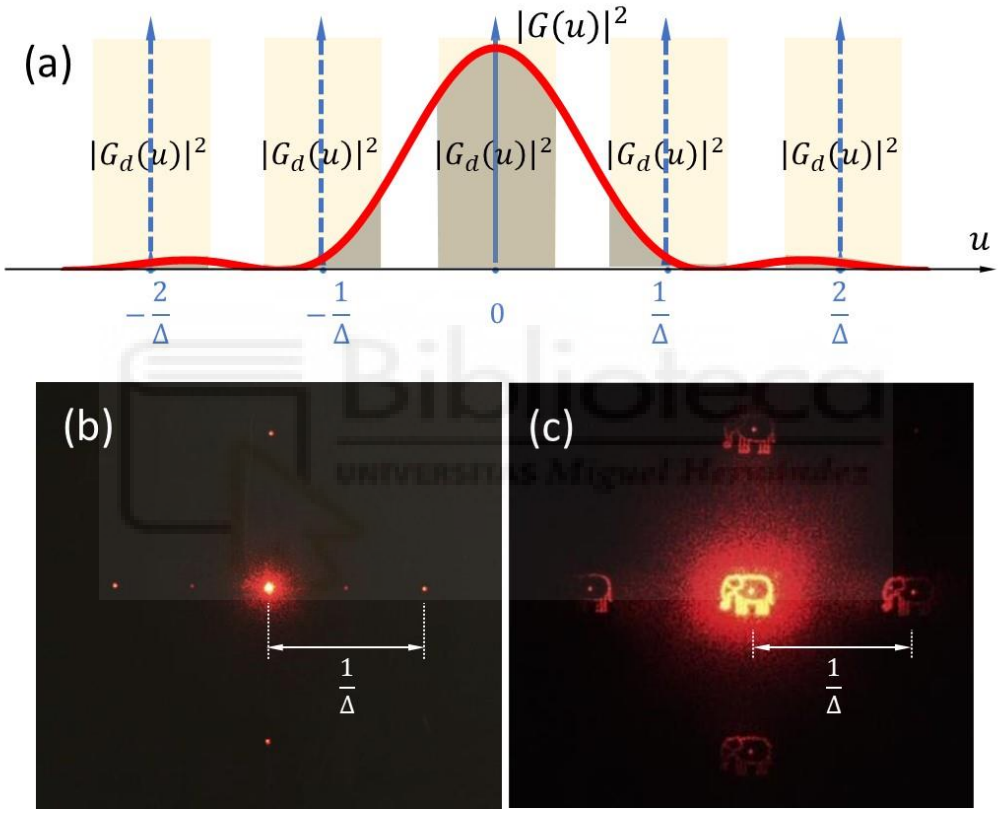

Fig. 7.2. (a) Representation of the intensity (modulus square) of Eq. (7.5), with an indication of the square of the envelope sinc function (red curve), the comb function (dashed delta functions) and replicas of $G_d(u) = \mathcal{F}\{g_d(x)\}$ (yellow boxes). Gray boxes indicate the reconstruction modulated by the envelope function. (b) Experimental capture when the SLM displays a uniform gray level. (c) Experimental capture when the SLM displays the hologram of an elephant picture [Gao-2024b, Gao-2025].

To illustrate experimentally the role of the sinc envelope, we select as function $G_d(u)$ the picture of an elephant. Its inverse Fourier transform is calculated numerically, and its phase distribution is displayed on the SLM [Gao-2024b, Gao-2025]. Figure 7.2(c) shows the experimental effective hologram reconstruction of a bright elephant centred on the zero-order, surrounded by four faint replicas centred on the locations of the ±1st horizontal and vertical orders that are

generated by the SLM pixelated structure (Fig. 7.2(b)). Note that the replica centred on the horizontal 1st order has a brighter left side, while that centred on the horizontal −1st order is brighter on its right side, in agreement with the shape of the sinc envelope in the regions around $u = +1/\Delta$ and $u = -1/\Delta$, respectively.

This effect is of great importance and must be corrected in holographic displays [Stat-2023]. One simple method consists in pre-compensating the object to be reconstructed by the hologram. Other methods have been proposed to minimize its impact, based on using different pixel shapes [Stat-2021] or distributions [Yam-2024], or by using multiple input beams [Che-2017]. However, most SLMs use the standard arrays of rectangular pixels.

Note that SLMs having a large fill factor, i.e., values of a approaching 1, seem in principle more valuable, since they provide higher light efficiency. However, the envelope sinc function decreases rapidly as we observe far from the axis. This would make the center of the hologram appear brighter than the sides. On the contrary, SLMs with smaller fill factor (lower values of a) provide a slowly-evolving envelope function, and therefore a wider area of almost uniform intensity around the axis. This makes them useful for displaying holograms without requiring the compensation techniques cited above.

8. Diffraction gratings displayed on pixelated SLMs at the Nyquist limit

This chapter considers diffraction gratings displayed on pixelated SLMs at their spatial resolution limit (Nyquist limit). These are gratings with binary phase profiles. We consider two theoretical approaches to analyse such binary diffraction gratings: the Fourier series and the Fourier transform. As derived in Chapter 7, the latter approach reveals the effect of a sinc function and multiple replicas of the target diffraction pattern due to the SLM's pixelation. Instead of using an elephant hologram as the target pattern, in this chapter the function $g_d(x)$ encoded on the SLM is a diffraction grating.

Analytical expressions for the complex amplitude coefficients and diffraction orders' intensities are obtained in terms of the pixel size (Δ), fill factor (a^2), and phase difference (φ) between two phase levels in the grating. We show that the convolutional Fourier transform approach probes very useful to gain physical insight into the different contributions to the diffraction orders [Gao-2024b, Gao-2025].

Experimental realizations are included, where the normalized intensities of the 0th, 1st, and

2nd orders are measured as a function of the gray level, for gratings with different periods. As the period decreases down to two pixels (Nyquist limit), the fringing effect becomes very noticeable. We consider that its main consequence is a smoothing of the ideal square phase profile. Therefore, we then introduce a nonlinear phase profile model to fit the experimental intensity curves.

This model considers unbalanced and asymmetric phase profiles governed by four parameters. Their proper selection (through fitting the experimental intensity data) allows us to obtain the smoothed phase profile that is actually implemented on the SLM due to fringing.

Finally, we identify the conditions to obtain a Nyquist triplicator in terms of the device's fill factor and phase level. The effect of the pixel crosstalk (fringing) on the diffraction efficiency and on the conditions to obtain a Nyquist triplicator are also discussed.

8.1. The Fourier series approach

We first use the Fourier series approach to analyse a binary phase grating displayed on the SLM at the Nyquist limit [Gao-2024b, Gao-2025]. Figure 8.1 defines the Nyquist binary phase grating, with a period of only two pixels, $p = 2\Delta$, i.e., including the interpixel dead zones. As considered in Chapter 7, the pixel effective size is $a\Delta$, with a^2 being the device's fill factor and $a < 1$.

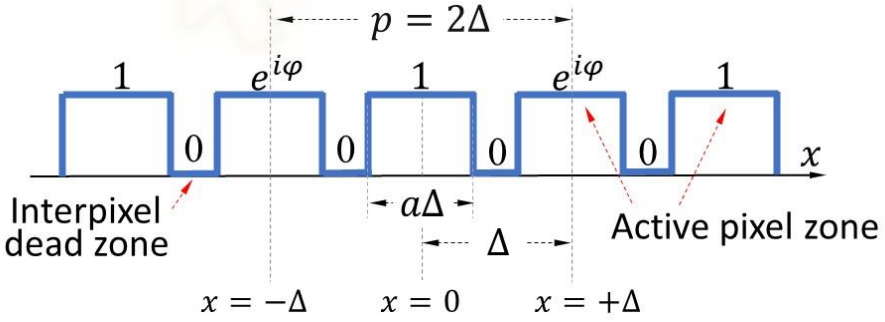

Fig. 8. 1. Nyquist phase grating (period $p = 2\Delta$) with phases 0 and φ displayed on an SLM with interpixel dead zones [Gao-2025].

The complex amplitude function defining one period $p = 2\Delta$ of the Nyquist phase grating in Fig. 8.1, from $x = -\Delta$ to from $x = +\Delta$, is given by the following relation:

$$g_0(x)=\begin{cases}1 & \text{if } x\in\left(-\frac{a\Delta}{2},+\frac{a\Delta}{2}\right),\\ 0 & \text{if } x\in\left(-\Delta\left(1-\frac{a}{2}\right),-\frac{a\Delta}{2}\right] \text{ or } x\in\left[+\frac{a\Delta}{2},+\Delta\left(1-\frac{a}{2}\right)\right),\\ e^{i\varphi} & \text{if } x\in\left(-\Delta,-\Delta\left(1-\frac{a}{2}\right)\right] \text{ or } x\in\left[+\Delta\left(1-\frac{a}{2}\right),+\Delta\right).\end{cases} \tag{8.1}$$

According to the Eq. (2.5) in Chapter 2, the Fourier coefficients of this periodic function are given by

$$c_m=\frac{1}{2\Delta}\int_{-\Delta}^{+\Delta} g_0(x)e^{-im\pi x/\Delta}dx, \tag{8.2}$$

where m is an integer index denoting the diffraction order. Noting that $g_0(x)$ is an even function, these Fourier coefficients can be calculated as:

$$c_m=\frac{1}{\Delta}\int_0^{+\Delta} g_0(x)\cos\left(\frac{m\pi x}{\Delta}\right)dx=\left[\frac{1}{\Delta}\int_0^{a\Delta/2}\cos\left(\frac{m\pi x}{\Delta}\right)dx\right]+\left[\frac{e^{i\varphi}}{\Delta}\int_{\Delta\left(1-\frac{a}{2}\right)}^{\Delta}\cos\left(\frac{m\pi x}{\Delta}\right)dx\right]. \tag{8.3}$$

After straightforward calculations, the Fourier coefficients obtained are:

$$c_m=\frac{a}{2}\left[1+(-1)^m e^{i\varphi}\right]\mathrm{sinc}\left(\frac{ma}{2}\right). \tag{8.4}$$

The intensity of the diffraction orders is proportional to the squared modulus of the Fourier coefficients, $I_m=|c_m|^2$, resulting in:

$$I_{m\,even}=a^2\mathrm{sinc}^2\left(\frac{ma}{2}\right)\cos^2\left(\frac{\varphi}{2}\right), \tag{8.5a}$$

$$I_{m\,odd}=a^2\mathrm{sinc}^2\left(\frac{ma}{2}\right)\sin^2\left(\frac{\varphi}{2}\right). \tag{8.5b}$$

Since $\mathrm{sinc}(0)=1$, the zero-order intensity is $I_0=a^2\cos^2(\varphi/2)$. The first and second orders have intensities $I_{\pm1}=a^2\mathrm{sinc}^2(a/2)\sin^2(\varphi/2)$ and $I_{\pm2}=a^2\mathrm{sinc}^2(a)\cos^2(\varphi/2)$, respectively.

As shown above, the Fourier series approach provides a simple derivation of the Fourier coefficients and diffraction orders' intensities of the Nyquist grating. However, this simple approach does not highlight the role of the SLM's pixelation and the resulting contributions to each diffraction order. This is why in the next section we consider the convolutional approach for analysing the Nyquist grating.

8.2. The convolutional approach

Whereas the diffraction generated by an arbitrary function $g_d(x)$ displayed on the SLM was discussed in Section 7.3, we now consider that this function is a diffraction grating [Gao-2024b, Gao-2025]. We assume a binary phase diffraction grating (Ronchi grating) with phases of zero and φ, and with period $p = k\Delta$, where k is an integer number ($k = 2$ for the Nyquist grating). Thus, the function $g_d(x)$ to be displayed on the SLM is:

$$g_d(x) = g_0(x) \otimes \sum_{m=-\infty}^{\infty} \delta(x - mk\Delta), \tag{8.6}$$

where $g_0(x)$ is the function defining the binary phase grating in a single period, which can be written as

$$g_0(x) = e^{i\varphi}\text{rect}\left(\frac{x}{k\Delta}\right) + \left(1 - e^{i\varphi}\right)\text{rect}\left(\frac{2x}{k\Delta}\right). \tag{8.7}$$

Now, the Fourier transform $G_d(u) = \mathcal{F}\{g_d(x)\}$ is given by:

$$G_d(u) = \frac{1}{k\Delta} G_0(u) \sum_{m=-\infty}^{\infty} \delta\left(u - \frac{m}{k\Delta}\right) = \sum_{m=-\infty}^{\infty} a_m\, \delta\left(u - \frac{m}{k\Delta}\right), \tag{8.8}$$

where $G_0(u) = \mathcal{F}\{g_0(x)\}$ is the Fourier transform of the grating's single period and a_m are the amplitude coefficients given by $a_m = G_0(m/k\Delta)/k\Delta$. Note that the binary grating, contrary to the situation of the elephant hologram in Fig. 7.2, provides high-order harmonic terms that lie at spatial frequencies that extend further from the separation $\pm 1/\Delta$ between the diffraction orders generated by the SLM's pixelated structure. Therefore, the superposition of different terms cannot be ignored when encoding binary phase gratings of a small period.

The Fourier transform of Eq. (8.7) is the function $G_0(u)$ given by:

$$G_0(u) = e^{i\varphi} k\Delta \text{sinc}(k\Delta u) + \left(1 - e^{i\varphi}\right)\frac{k\Delta}{2}\text{sinc}\left(\frac{k\Delta u}{2}\right). \tag{8.9}$$

Therefore, the coefficients a_m of the function $G_d(u)$ defined in Eq. (8.8)) read:

$$a_m \equiv \frac{G_0\left(\frac{m}{k\Delta}\right)}{k\Delta} = e^{i\varphi}\text{sinc}(m) + \frac{1 - e^{i\varphi}}{2}\text{sinc}\left(\frac{m}{2}\right) = \begin{cases} \frac{1}{2}\left(1 + e^{i\varphi}\right) & \text{if } m = 0, \\ \frac{1}{2}\left(1 - e^{i\varphi}\right)\text{sinc}\left(\frac{m}{2}\right) & \text{if } m \neq 0. \end{cases} \tag{8.10}$$

Note that $a_m = 0$ for all even values of m, except for $m = 0$. Now, applying Eq. (8.8) to Eq. (7.5), the diffraction pattern of the binary phase grating encoded on the SLM can be expressed as:

$$G(u) = a\,\mathrm{sinc}(a\Delta u)\left[\left(\sum_{m=-\infty}^{\infty} a_m \delta\left(u - \frac{m}{k\Delta}\right)\right) \otimes \left(\sum_{n=-\infty}^{\infty} \delta\left(u - \frac{n}{\Delta}\right)\right)\right]. \tag{8.11}$$

This relation shows the effect of the SLM dead zones [Gao-2024b, Gao-2025]. The diffraction orders generated by the displayed grating (m index) are replicated with the centre on each diffraction order stemming from the SLM pixelation (n index). Consequently, a superposition between orders of the different replicas occurs provided k is an integer value, being the strongest effect when $k = 2$. Note that blazed phase gratings with fractional values k have been proposed, leading to asynchronously sampled phase gratings [Dav-2002]. Here we consider integer k values; hence, there is a superposition of orders generated at each replica, and Eq. (8.11) can be rewritten as:

$$G(u) = a\,\mathrm{sinc}(a\Delta u) \sum_{q=-\infty}^{\infty} b_q\, \delta\left(u - \frac{q}{k\Delta}\right). \tag{8.12}$$

Therefore, the diffraction pattern of the binary grating displayed on the SLM can be written as a set of diffraction orders located at spatial frequencies $u = q/k\Delta$, multiplied by the envelope function that accounts for the SLM's pixel size, where each order q has an amplitude b_q that results from the superposition of all the a_m coefficients (given in Eq. (8.10)) fulfilling the condition $m = q - kn$:

$$b_q = \sum_{n=-\infty}^{\infty} a_{m=q-kn}. \tag{8.13}$$

Let us consider the Nyquist grating, $k = 2$. The contribution to the central order $q = 0$ requires summing all the terms in Eq. (8.13) that fulfil the relation $m = -2n$. These include $(n = 0,\ m = 0)$ which is the zero-order of the displayed grating centred on the zero-order of the SLM grating, but also $(n = +1,\ m = -2)$, which is the negative second order of the displayed grating centered on the positive SLM first order, and $(n = -1,\ m = +2)$ which is the positive second order of the displayed grating centred on the SLM negative first order, etc. Thus, $b_{q=0} = a_0 + a_2 + a_4 \ldots + a_{-2} + a_{-4} \ldots$. However, for a perfect binary grating with a $1/2$ duty cycle in a two-pixel period, all even orders vanish except the zero-order (Eq. (8.10)), and only a_0 contributes to this summation. Hence, using Eq. (8.13) in Eq. (8.12) leads to the coefficient c_0 of the central order:

$$c_0 = G(u = 0) = \frac{a}{2}\left(1 + e^{i\varphi}\right). \tag{8.14a}$$

Similarly, we can derive the amplitude coefficient of the 1st diffraction order $q = +1$, located at $u = 1/(2\Delta)$ by using the condition $m = 1 - 2n$ in the summatory of Eq. (8.13). This order appears right in between the zero and the first order generated by the SLM [Gao-2024b, Gao-2025]. In this case, this diffraction order receives the contribution of two main terms: $(n = 0, m = +1)$ that is the positive first order of the displayed grating centred on the SLM zero-order, but also $(n = +1, m = -1)$ which is the negative first order of the displayed grating centred on the positive SLM first order. Other weaker higher-order terms contribute: $(n = -1, m = +3)$, $(n = +2, m = -3)$, etc. Therefore, for this order the summation in Eq. (8.13) is $b_{q=1} = a_1 + a_{-1} + a_3 + a_{-3} + \cdots$ and, in this case, Eqs. (8.12) and (8.13) lead to the overall coefficient:

$$c_1 = G\left(u = \frac{1}{2\Delta}\right) = \frac{a}{2}\left(1 - e^{i\varphi}\right)\mathrm{sinc}\left(\frac{a}{2}\right). \tag{8.14b}$$

Finally, we also consider the 2nd order, $q = +2$, which appears located at $u = 1/\Delta$, i.e., right at the position of the first order generated by the SLM. This order receives the contribution of all terms fulfilling $m = 2 - 2n$ in the summatory in Eq. (8.13), which again are all even. Therefore, only a_0 contributes and the result is

$$c_2 = G\left(u = \frac{1}{\Delta}\right) = \frac{a}{2}\left(1 + e^{i\varphi}\right)\mathrm{sinc}(a). \tag{8.14c}$$

The corresponding intensities of the 0th, 1st, and 2nd orders, for the binary phase grating with phase difference φ and period $p = 2\Delta$ encoded on an SLM of pixel pitch Δ and fill factor parameter a, are thus given by the following analytical expressions:

$$I_0 = |c_0|^2 = a^2\cos^2\left(\frac{\varphi}{2}\right), \tag{8.15a}$$

$$I_{\pm 1} = |c_1|^2 = a^2\mathrm{sinc}^2\left(\frac{a}{2}\right)\sin^2\left(\frac{\varphi}{2}\right), \tag{8.15b}$$

$$I_{\pm 2} = |c_2|^2 = a^2\mathrm{sinc}^2(a)\cos^2\left(\frac{\varphi}{2}\right), \tag{8.15c}$$

where we applied the problem symmetry to also include the negative diffraction orders. This convolutional derivation is longer than the common Fourier series analysis in Section 8.1, but provides an extremely useful physical interpretation, as illustrated next in the experimental verification. These expressions of the intensities coincide with those in Eqs. (8.5) evaluated at $m = 0, \pm 1, \pm 2, \pm 3 \ldots$.

8.3. Diffraction gratings displayed at the Nyquist limit

Let us now discuss the diffraction orders' intensities of the binary phase grating with two pixels per period that were derived in the previous section [Gao-2025]. Figure 8.2(a) shows the intensities of these orders as a function of the phase modulation φ for different values of a. The limit $a \to 1$ corresponds to a non-pixelated SLM, where the relations in Eqs. (8.15) recover those of the standard binary phase grating in Section 5.1 i.e. $I_0(a=1) = \cos^2(\varphi/2)$, $I_{\pm1}(a=1) = (4/\pi^2)\sin^2(\varphi/2)$ and $I_{\pm2} = 0$. All cases in Fig. 8.2(a) feature a null zero-order and a maximum efficiency of the first orders when $\varphi = \pi$. However, note that the maximum of I_0 decreases much faster than the maximum of $I_{\pm1}$ as parameter a is reduced. Therefore, the maximum of $I_{\pm1}$ relative to the maximum value of I_0 is higher than the value $4/\pi^2 = 0.405$ expected for a binary phase Ronchi grating. This behaviour was already demonstrated in [Dav-2006] using a transmission SLM, where the effect was stronger due to the smaller value of a compared to that in our device. Note also the second orders stemming from the SLM pixelization, which are stronger as a is smaller.

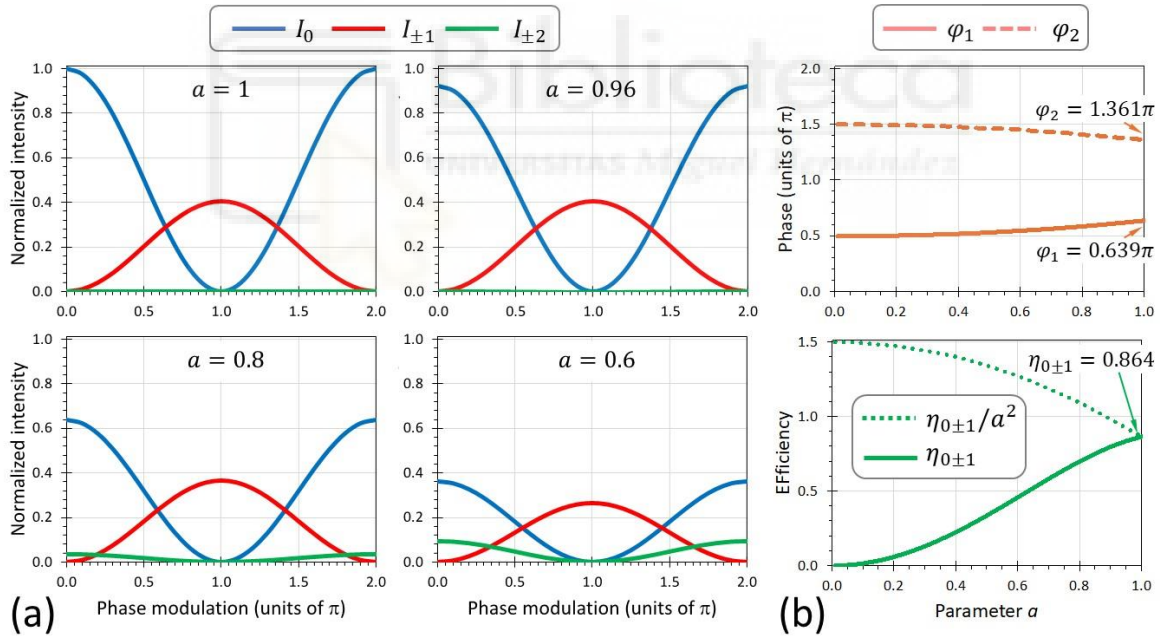

Fig. 8. 2. (a) Intensity of the zero, first, and second orders as a function of the phase modulation φ for different values of the a parameter. (b) Phases φ_1 and φ_2 that provide the triplicator response in Eq. (8.17) and its diffraction efficiency as a function of the a parameter [Gao-2025].

Equations (8.15) provide the ratio between the first and the zero-order as a function of the phase modulation [Gao-2025]:

$$\frac{I_{\pm 1}}{I_0} = \mathrm{sinc}^2\left(\frac{a}{2}\right)\tan^2\left(\frac{\varphi}{2}\right). \tag{8.16}$$

This result is relevant because this ratio I_1/I_0 has been sometimes used to calibrate the SLM phase modulation φ [Lu-1994, Már-2005]. However, the sinc term in Eq. (8.16) is not obtained if the SLM pixelization is ignored. Although this has minimal effect if the displayed gratings have a large number of pixels per period, it can lead to a wrong calibration when encoding gratings with a very low number of pixels per period.

According to Eq. (8.16) the triplicator condition $(I_1/I_0 = 1)$ for the Nyquist binary grating reads:

$$\tan^2\left(\frac{\varphi}{2}\right) = \frac{1}{\mathrm{sinc}^2\left(\frac{a}{2}\right)}. \tag{8.17}$$

Hence, SLM devices with different fill factors would require different phase level values φ of the binary grating to achieve the Nyquist triplicator. If phase φ is selected to fulfil Eq. (8.17), then the overall diffraction efficiency for the three target diffraction orders in the triplicator is given by

$$\eta_{0\pm 1} = I_0 + I_{+1} + I_{-1} = 3a^2\cos^2\left(\frac{\varphi}{2}\right). \tag{8.18}$$

For a binary triplicator grating with no dead zones $(a = 1)$, Eq. (8.17) provides two possible phase values $\varphi = 0.639\pi$ and $\varphi = 1.361\pi$, and a total efficiency $\eta_{0\pm 1} = 86.4\%$ [Gao-2024]. Instead, according to the value $a = 0.964$ in our device, the phase values that render the triplicator are slightly different: $\varphi = 0.629\pi$ and $\varphi = 1.371\pi$, and the efficiency is $\eta_{0\pm 1} = 84.4\%$.

Figure 8.2(b) shows the dependence on the a parameter of the two possible phase values solution of Eq. (8.17) that render the triplicator response. It shows as well the triplicator efficiency $\eta_{0\pm 1}$ and the ratio $\eta_{0\pm 1}/a^2$ as a function of a. While $\eta_{0\pm 1}$ represents the triplicator efficiency relative to the input beam, the ratio $\eta_{0\pm 1}/a^2$ is the triplicator efficiency relative to the total intensity transmitted by the SLM. It is interesting to note their opposite evolution with a. Namely, whereas $\eta_{0\pm 1}$ increases with the a parameter, $\eta_{0\pm 1}/a^2$ increases for decreasing a, thus leading to weaker higher orders relative to the three central target orders.

We display binary phase gratings of different periods on the SLM [Gao-2025]. The final goal is to reproduce a triplicator grating. Figure 8.3 shows the diffraction pattern of four binary phase triplicator gratings with periods $p = 16\Delta$, $p = 8\Delta$, $p = 4\Delta$, and the Nyquist limit $p = 2\Delta$. In all cases, the displayed grating has one gray level equal to zero while the second gray level was varied until reaching the triplicator condition. The required value is different in each case, namely: $g =$

$80, 88, 104$, and 136 respectively.

The gratings were displayed on the SLM vertically aligned, so the diffraction orders lie horizontally. The captures in Fig. 8.3 feature the three target bright diffraction orders at the centre of the diffraction pattern. Other non-target orders are visible too. The captures in Fig. 8.3 were deliberately saturated to clearly visualize the non-target orders. Figure 8.3(a), where $p = 16\Delta$, clearly shows the three bright central target orders, with the 1st orders lying at spatial frequencies $u = \pm 1/(16\Delta)$. The figure also indicates the spatial frequency $1/\Delta$ where the first order stemming from the SLM's pixelated structure appears (as shown in Fig. 7.2(b)). The second and other even orders are very weak (ideally for the binary grating they should be zero). In Figs. 8.3(b) and 8.3(c), where the period is reduced to $p = 8\Delta$ and $p = 4\Delta$, the diffraction orders are separated by $1/(8\Delta)$ and $1/(4\Delta)$ respectively. Finally, Fig. 8.3(d) shows the Nyquist grating with $p = 2\Delta$, where now only a single diffraction order located at $u = 1/(2\Delta)$ appears between the zero-order and the first order generated by the SLM (the latter located at $u = 1/\Delta$).

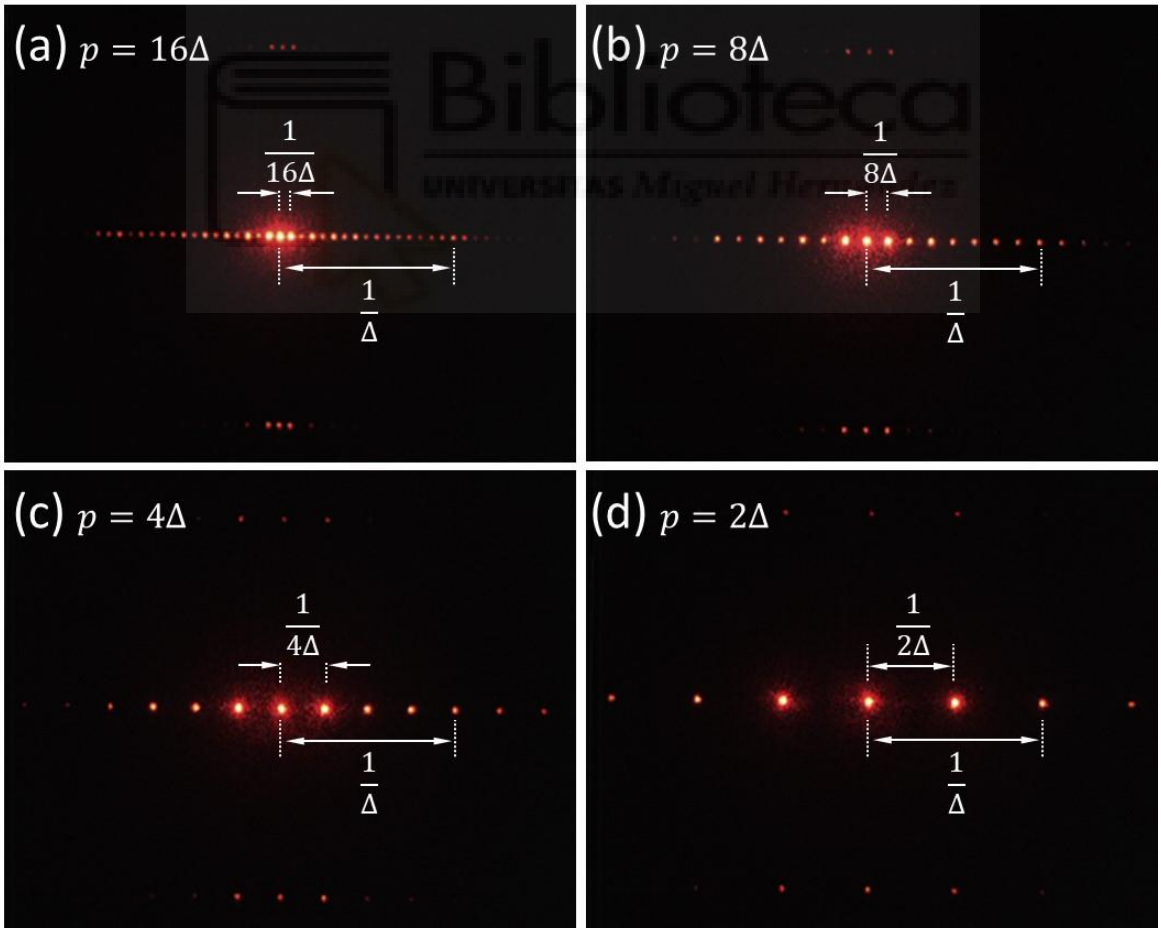

Fig. 8. 3. Diffraction pattern of the binary phase triplicator gratings displayed with different periods and gray levels [Gao-2025]: (a) $p = 16\Delta$, $g = 80$, (b) $p = 8\Delta$, $g = 88$, (c) $p = 4\Delta$, $g = 104$, (d) $p = 2\Delta$, $g = 136$.

According to the convolutional approach described in Section 8.2, the diffraction pattern generated by the binary grating is replicated at the location of the SLM orders. The captures in Fig. 8.3 show these replicas at the diffraction orders located at $\pm 1/\Delta$ along the vertical direction, where the three central triplicator orders are clearly visible. Along the horizontal direction, however, a superposition occurs between the orders of the displayed grating replicated at the SLM diffraction orders. This effect is mostly important in Fig. 8.3(d) where the shortest period (Nyquist grating) is considered. Note that for cases $p = 16\Delta$, $p = 8\Delta$ and $p = 4\Delta$, the diffraction pattern is described by Eq. (8.12), but the superposition of orders in the summation of Eq. (8.13) is different in each case, since it depends on how many pixels in one period (k value).

8.4. Fringing effect and smoothing of the phase profile

However, we must consider another effect strongly affecting the diffraction efficiency of SLMs operating at the Nyquist limit: the fringing-field effect, which leads to pixel crosstalk due to the non-uniform distribution of the LC director in the pixel area [Mor-2021]. This effect has been shown to reduce the diffraction efficiency of binary phase gratings displayed with spatial frequencies close to the SLM spatial resolution limit due to smoothing of the phase profile and a reduction of the maximum phase depth [Lin-2013, Már-2005]. Hence, the fringing cannot be ignored when displaying gratings with short periods like those in Fig. 8.3(b-d).

We measured the intensity of the 0th, 1st, and 2nd orders as a function of the variable gray level for binary phase gratings with different periods (the other gray level in the grating was kept at $g = 0$) [Gao-2025]. The dots in the curves of Fig 8.4 indicate the experimental data. The diffraction orders' intensities are normalized to the intensity of the zero-order when $g = 0$, where $I_0 = a^2$ (Eq. (8.15a)). Figure 8.4(a) shows the result for the grating with period $p = 64\Delta$. This is a large enough period; thus, the distortion caused by the fringing effect is minimal. In addition, since the diffraction angle is small, the three target diffraction orders lie close together at the centre and the sinc envelope does not significantly affect. Therefore, the intensity of these diffraction orders can be well approximated by the relations of the standard binary phase grating, i.e.,

$$I_0 = a^2 \cos^2\left(\frac{\varphi}{2}\right), \tag{8.19a}$$

and

$$I_{\pm 1} \cong \frac{4a^2}{\pi^2} \sin^2\left(\frac{\varphi}{2}\right). \tag{8.19b}$$

These relations ignore the fringing effect and the interpixel dead zone, except for the a^2 fill factor used to normalize the maximum achievable value. The corresponding curves, indicated in Fig. 8.4(a) with the label 'no fringing', bear quite good agreement with the experimental data for $p = 64\Delta$, assuming the phase modulation presented in Fig. 4.8 in Section 4.3. The binary triplicator grating response is obtained at the two expected gray levels, $g = 80$ and $g = 172$. Nevertheless, the fringing effect can be noticed by looking at the intensity of the zero-order for $g = 255$, which does not recover the same value as for $g = 0$ due to the smoothing of the phase profile [Lin-2013]. Another interesting aspect to note is the total extinction of the zero-order for gray level $g = 128$, which is an indication that the SLM is not affected significantly by other effects like flicker or phase fluctuations.

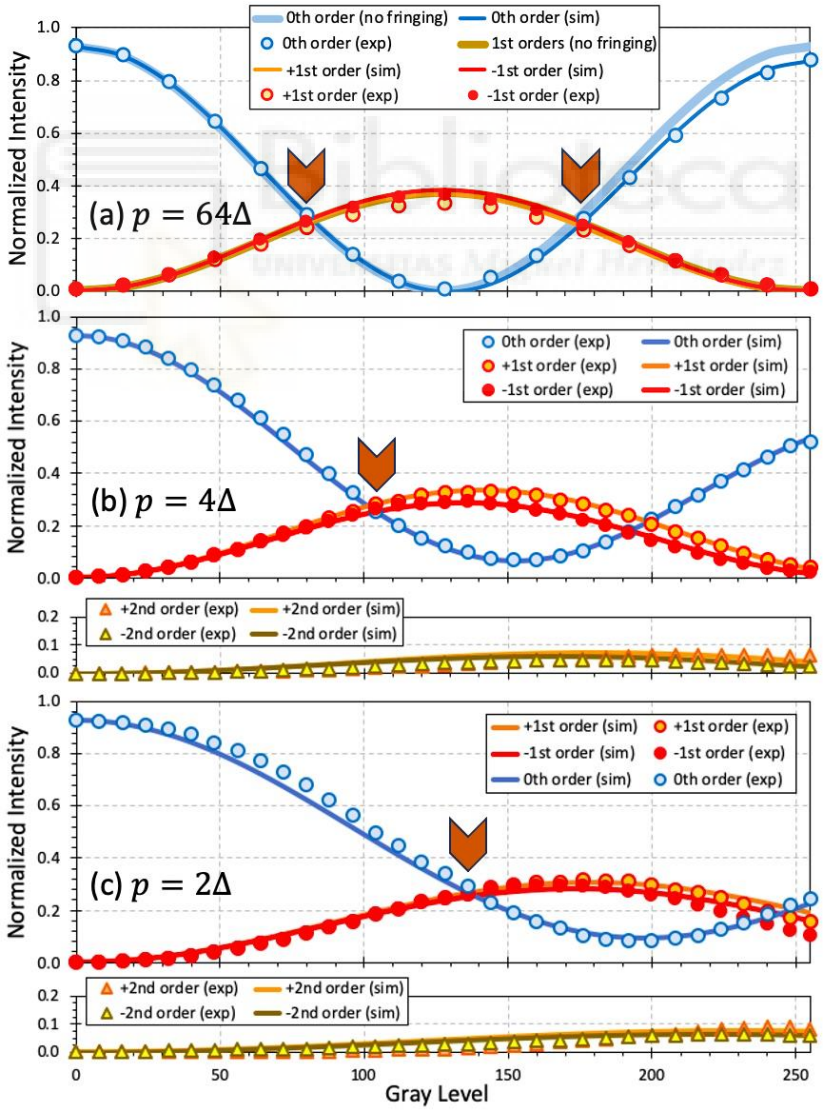

Fig. 8. 4. Experimental normalized intensities of the 0th, 1st, and 2nd orders (circle and triangular dots) for the binary

phase grating with period of (a) $p = 64$ pixels, (b) $p = 4$ pixels, and (c) $p = 2$ pixels. The arrow indicates the lowest gray level where the triplicator is obtained. The continuous lines indicate the fit obtained with the fringing model [Gao-2025].

However, when decreasing the period to $p = 4\Delta$ (Fig. 8.4(b)), and furthermore to $p = 2\Delta$ (Fig. 8.4(c)), the SLM's pixelated structure and the fringing strongly affect the results. Another typical indication of the fringing-field effect is that the gray level required to yield the minimum zero-order intensity, and the maximum first-order intensity shifts to higher gray values [Mor-2021, Mos-2019]. This is clearly observed in Figs. 8.4(b) and 8.4(c), where these minimum and maximum occur at gray levels far from the value $g = 128$ expected from the phase modulation calibration in Fig. 4.8. The presence of significative second diffraction orders is also a clear signature of the pixel crosstalk. In fact, they are clearly visible in the captures of Fig. 8.3.

There exist powerful techniques capable of calculating the LC director's distribution inside the LC layer to account for the fringing effect [Mos-2019, Bou-2000]. However, these methods require considerable computational effort and knowledge of the physical parameters of the LC material, information that is not generally available to users of commercial SLMs. Here, instead, we follow a simplified procedure introduced in [Mor-2021] which makes use of the fact that the fringing effect smooths the binary grating phase profile.

Hence, the phase profile $\psi(x, g)$ produced by the SLM is retrieved from the measured diffraction orders intensities as a function of the gray level g (Fig. 8.4) by assuming a dependency as [Gao-2025]:

$$\psi(x, g) = m(g)\psi_1(x), \tag{8.20}$$

where $\psi_1(x)$ is a periodic function given by

$$\psi_1(x) = \frac{1}{C}\arctan\{A\cos[f(x)]\}, \tag{8.21}$$

where the argument $f(x) = 2\pi x/p$, so $\psi_1(x)$ is periodic with period p. The numerical parameter A controls the smoothing of the function $\psi_1(x)$, and C is a normalization constant equal to the maximum value of the $\arctan$ term in Eq. (8.21) (which depends on the selected value of A), so the values of $\psi_1(x)$ range between -1 and $+1$. Figure 8.5(a) illustrates how the profile $\psi_1(x)$ changes from square to sinusoidal for values of A from 1000 to 1.

The multiplicative factor $m(g)$ in Eq. (8.20) provides the phase depth variation, i.e., the peak-to-peak variation of the phase profile $\varphi(x, g)$ as a function of gray level g. This term is selected as $m(g) = M\pi g/g_{2\pi}$, where $g_{2\pi}$ is the gray level leading to a 2π phase modulation (in our

case, $g_{2\pi} = 255$ according to Fig. 4.8. The parameter M accounts for a possible reduction of the maximum phase depth that the grating can reach ($M = 1$ means that there is no phase depth reduction due to fringing). This way, ref. [Mor-2021] showed that the intensities of the diffraction orders generated by the binary gratings displayed on the SLM could be quantitatively described with particularly good accuracy simply by adjusting the numerical parameters A and M.

However, the SLM used in [Mor-2021] had a pixel size more than three times larger than the pixel size of the Exulus SLM device here employed. The experimental curves in Fig. 8.4(b) and 8.4(c) show features that cannot be explained accurately with such a simple approach [Gao-2025]. In particular, the asymmetric behaviour of positive and negative orders implies some asymmetry in the phase profile. To account for this asymmetry, we consider the smoothing of the phase profile different when going from lower to higher phase level than in the opposite direction. Therefore, we introduce two parameters, A_L and A_R, one for the left and another for the right part of the period. This is illustrated in the two top rows of Fig. 8.5(b). Secondly, the fact that the zero-order is never cancelled in these curves is an indication that the phase profile is not balanced with respect to the zero phase, i.e., the mean phase value is not zero. We account for this effect by changing the function $f(x)$ in Eq. (8.21) to be $f(x) = \arctan[B2\pi x/p]$, normalized to its maximum value. The numerical constant B leads to a function $f(x)$ linear with x for values below $B = 0.1$. However, increasing the value of B leads to a nonlinear function that results in a phase profile where the width of the positive phases is narrower than the width of the negative phases, as shown in the two last cases in Fig. 8.5(b).

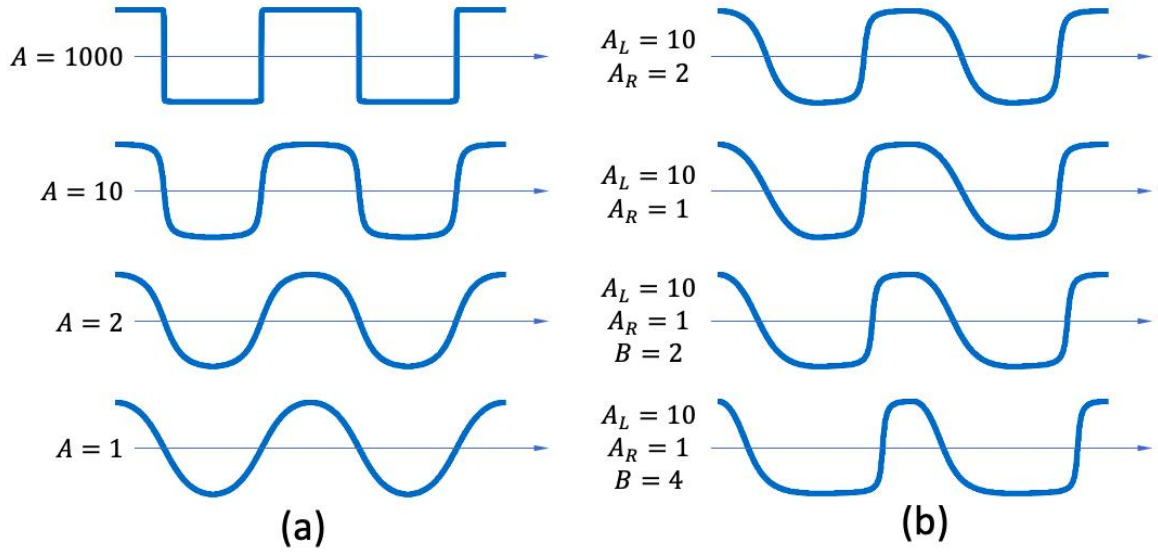

Fig. 8. 5. Illustration of the effect of the parameters that model the fringing effect on the phase profile $\psi_1(x)$ [Gao-2025] (a) Simplified model in Eq. (8.21) with a single parameter A. (b) Refined asymmetric model with parameters A_L,

A_R and B.

Therefore, we use four numerical parameters (A_L, A_R and B, to define the phase profile $\psi_1(x)$, and M to account for the phase peak-to-peak variation with g) to define the phase profiles $\psi(x,g)$ [Gao-2025] These phase profiles are multiplied by the amplitude profile accounting for the SLM pixelated structure (in all cases we consider the value $a = 0.964$). Then, the Fourier coefficients of each phase profile $\psi(x,g)$ are calculated as a function of g and its modulus squared value is compared to the experimental data. The numerical parameters that best fit the experimental curves in Fig. 8.4 are searched by minimizing the absolute error $\varepsilon = \frac{1}{N}\sum_N |i_n^{num} - i_n^{exp}|$ between the calculated (i_n^{num}) and the experimental (i_n^{exp}) intensity for each of the N measurements obtained for each gray level and for each measured diffraction order. Table I gives the numerical constants for the three cases in Fig. 8.4 corresponding to gratings with $p = 64\Delta$, $p = 4\Delta$, and $p = 2\Delta$.

Table I. Numerical constants of the fringing model that best fits the experimental data [Gao-2025].

	A_L	A_R	B	M	ε
$p = 64\Delta$	72.88	46.51	0.01	1.00	1.10%
$p = 4\Delta$	6.76	4.20	1.47	0.98	0.72%
$p = 2\Delta$	4.83	3.58	1.85	0.87	1.38%

The graphs in Fig. 8.4 include the corresponding best fit curves. For $p = 64\Delta$, the best fit represents only a slight variation with respect to the relations in Eqs. (8.19) (no fringing' curves in Fig. 8.4(a)) [Gao-2025]. Nevertheless, considering the fringing effect provides a better fit of the experimental data, with an error below $\varepsilon = 1.1\%$. Figure 8.6(a) shows the corresponding phase profile derived from these values, represented in one period (64 pixels). Despite its smooth edges, the period is very large, so the binary phase grating profile is still relatively well implemented, as indicated by the large values of $A_L = 72.88$ and $A_R = 46.51$. Therefore, the binary profile is dominant, and smoothing makes only a small variation in the diffraction efficiency curves.

In Figs. 8.4(b) and 8.4(c), however, the discrepancies between the experimental curves and the 'no fringing' curves are clear. Hence, the best fit curves accounting for fringing are now relevant. For $p = 4\Delta$ the best fit is achieved with much lower values $A_L = 6.76$ and $A_R = 4.20$. Therefore, now the smoothing represents a much greater fraction of the period. Their different

values lead to a very good fit of the differences between the intensities of the ± 1st diffraction orders. Note also that the best fit is obtained with a nonlinear unbalanced parameter $B = 1.47$. The corresponding phase profile shown in Fig.8.6(b) shows a narrower variation for positive phases than for negative phases. Finally, a very small reduction of the phase depth, $M = 0.98$, is also required to reach an error of only $\varepsilon = 0.72\%$.

Finally, the best fit for the Nyquist grating ($p = 2\Delta$), Fig. 8.4(c), is obtained for values $A_L = 4.83$ and $A_R = 3.58$, $B = 1.85$, and a reduced phase depth $M = 0.87$. The corresponding phase profile shown in Fig. 8.6(c) clearly shows the strongly smoothed and unbalanced profile, with reduced phase depth. In this case, the error between the experimental data and the best fit is $\varepsilon = 1.38\%$.

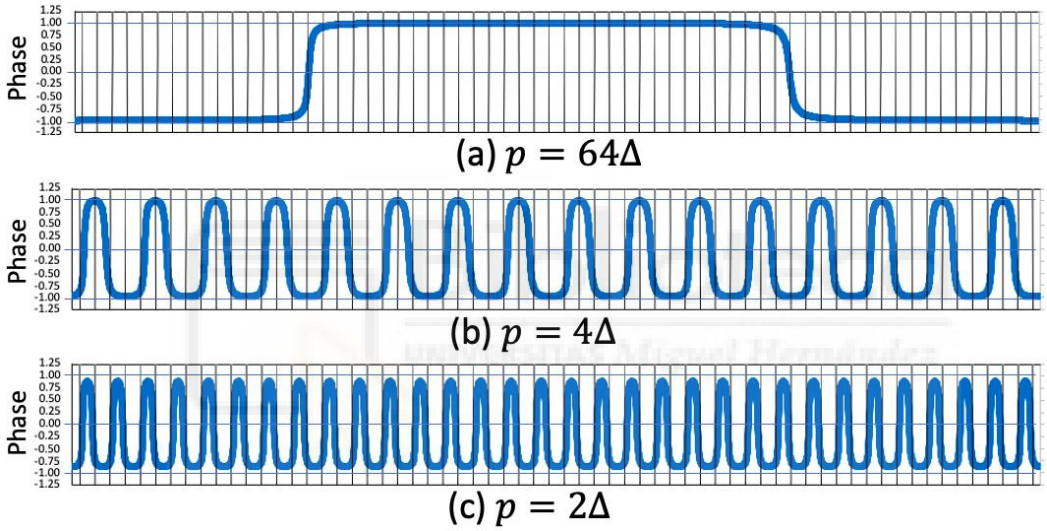

Fig. 8. 6. Phase profiles that best fit the curves in Fig. 8.4 for binary gratings with periods of $p = 64$, $p = 4$ and $p = 2$ pixels [Gao-2025]. The profiles are represented for the maximum gray level $\psi(x, g = 255)$ and in units of π. The dead zones are indicated as black lines.

Nevertheless, despite the strong influence of the fringing effect, the triplicator grating condition is found in all three cases at the gray level values where the 0th and 1st orders' intensity curves intersect. For each grating, the gray level meeting the triplicator condition is indicated by an arrow in Fig. 8.4. For the case with $p = 64\Delta$, two gray levels meet the triplicator condition, namely $g = 80$ and $g = 176$. However, the asymmetry of the phase profile eliminates the second gray level for the cases with $p = 4\Delta$ and $p = 2\Delta$ due to the different intensities of the $+1$st and -1st orders. The triplicator condition occurs at $g = 104$ in Fig. 8.4(b) and at $g = 136$ in Fig. 8.4(c). The measured triplicator diffraction efficiency $\eta_{0\pm1} = I_0 + I_{+1} + I_{-1}$ is $\eta_{0\pm1} = 79.5\%$, (which is

different from the efficiency given in Chapter 5, since it is normalized by the fill factor here) for the grating with period $p = 64\Delta$, $\eta_{0\pm1} = 80.1\%$ for $p = 4\Delta$, and $\eta_{0\pm1} = 81.7\%$ for $p = 2\Delta$. These values are lower than those expected from Eq. (8.18), namely: 84.7%, 84.6%, 84.4%, respectively, being this reduction attributed to the phase profile distortion caused by fringing. Interestingly, we found an opposite evolution with the period for the experimental and theoretical diffraction efficiency. Let us note that the theoretical efficiency is lower as $p \to 2\Delta$, whereas the experimental efficiency is larger as $p \to 2\Delta$. This is also attributed to the phase profile distortion due to fringing.

Python Code

The following Python scripts allow us to simulate the smoothed phase profile that is produced by the SLM according to Eqs. (8.20-8.21) in terms of the numerical parameters A_L, A_R, B, and M, which are discussed in Section 8.4. Then, we can calculate the normalized intensities of the 0th, 1st, and 2nd orders for the binary phase grating based on these simulated phase profiles to fit the experimental results shown in Fig. 8.4.

Script 1.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator
"""
In this section, we only simulate the influence of parameters AL and AR on the phase profile keeping
other parameters constant, and how they affect the normalized intensities of the 0th order and the 1st
orders.
1. AL and AR are in the range of (1000, 2, 50), and they can be explored separately.
2. M and B are initialled with 1.007 and 0.1. respectively.
3. Some other necessary parameters are defined and initialled:
    Fill factor a,
    Roundness A for linear phase function,
    Diffraction order m,
    Period p,
    gray level gray (phase value).
"""
AL=[1000,2,50]
AR=AL
M=1.007
B=0.1
a=[0.964,1,0.7]
A=1000
m=[0,1,-1]
p=1
gray=np.arange(0,256,1)
x=np.linspace(-0.5, 0.5,640)  # For linear
x_=np.linspace(0, 0.5,320)   # For nonlinear
ac=[]   # An empty array to save the normalized intensities associated with different fill factors.
# A loop that is for going through different fill factors.
for a_ in range (len(a)):
    # The linear f(x) in Eq. (8.21).
    f_linear=2*x/p
    # Define a nonlinear f(x) as f_1, which is designed based on f_linear for showing the difference
between linear and non-linear phase functions.
    f_1=np.arctan(B*(f_linear/2))
```

```
    max_index_1 = np.argmax(f_1)
    max_value_1 = f_1[max_index_1]
    f_n=f_1/max_value_1     # Normalized f_1.
    # The phase profile which is calculated based on linear f_linear.
    PhaseProfile=np.arctan(A*np.cos(f_linear*np.pi))/(np.pi)
    # A nonlinear f(x) which is similar to f_1, but this one is for calculating a nonlinear phase
profile. Because the phase profile will not be symmetric this is for the negative/left part of it.
    f_=np.arctan(B*(x_/p))
    max_index_ = np.argmax(f_)
    max_value_ = f_[max_index_]
    f_L=f_/max_value_
    # The positive/right part.
    f_r=np.arctan(B*(-x_/p))
    min_index_r = np.argmin(f_r)
    min_value_r = f_r[min_index_r]
    f_R=f_/min_value_r

    Mc=[]   # An empty array to save the normalized intensities associated with different parameters
M.
    # A loop that is for going through different parameters AL/AR.
    for n in range(len(AL)):
        # The negative/left part of the phase profile which is calculated based on nonlinear f_L.
        PhaseProfile_L=np.arctan(AL[n]*np.cos(f_L*np.pi))/(np.pi)
        max_index_L = np.argmax(PhaseProfile_L)
        max_value_L = PhaseProfile_L[max_index_L]
        PhaseProfile_L=PhaseProfile_L/max_value_L
        # The positive/right part.
        PhaseProfile_R=np.arctan(AL[n]*np.cos(f_R*np.pi))/(np.pi)
        max_index_R = np.argmax(PhaseProfile_R)
        max_value_R = PhaseProfile_R[max_index_R]
        PhaseProfile_R=PhaseProfile_R/max_value_R
        # Combine left and right together.
        PhaseProfile_n=np.concatenate((PhaseProfile_L[::-1], PhaseProfile_R))
        # Simulate the binary amplitude grating resembling the pixelation structure of SLM.
        pixelation=[0] * len(x)
        for i in range(len(x)):
            if x[i]<len(x)/2 and x[i]>a[a_]/2:
                pixelation[i]=0
            elif x[i]>-len(x)/2 and x[i]<-a[a_]/2:
                pixelation[i]=0
            else:
                pixelation[i]=1
        # The phase is resacled by parameter M.
        PhaseProfile_M=PhaseProfile_n*M

        mc=[]  # An empty array to save the normalized intensities associated with different
diffraction orders m.
        # A loop that is for going through different diffraction orders m.
        for k in range(len(m)):
            cc=[]    # An empty array to save the coefficients associated with different phase values
gray.
            for j in range(len(gray)):
                gray_M=M*gray[j]/255    # The phase modulation range, which is manipulated by
parameter M.
                phaseProfile_C=PhaseProfile_M*gray_M
                ex=m[k]*np.pi*f_linear
                # The integrand of Fourier series: the multiplication of SLM´s pixelation function,
the nonlinear phase profile, and an exponent function.
                phase=pixelation*np.exp(1j*phaseProfile_C*np.pi)*np.exp(1j*ex)
                cm=np.sum(phase)
                cc.append(cm)   # The coefficient value for one of the orders we requested is based on
the Fourier series integral.
            mc.append(cc)   # The coefficient value for all the orders we requested.
        Im=mc*np.conjugate(mc)/len(x)**2   # The normalized intensities for all the orders we
requested.
        Mc.append(Im)   # The normalized intensities of all the orders for all the requested AL/AR.
    ac.append(Mc)   # The normalized intensities of all the orders for all the requested AL/AR with
different fill factors.

# Showing the normalized intensities of all the orders for all the requested AL/AR with different fill
factors in terms of phase values (gray level).
plt.figure()
```

```
plt.plot(gray,ac[0][0][0],label=f"A={AL[0]},a={a[0]},order={m[0]}")
plt.plot(gray,ac[0][0][1],label=f"A={AL[0]},a={a[0]},order={m[1]}")
plt.plot(gray,ac[0][0][2],label=f"A={AL[0]},a={a[0]},order={m[2]}")

plt.plot(gray,ac[0][1][0],label=f"A={AL[1]},a={a[0]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[0][1][1],label=f"A={AL[1]},a={a[0]},order={m[1]}",linestyle='--')
plt.plot(gray,ac[0][1][2],label=f"A={AL[1]},a={a[0]},order={m[2]}",linestyle='--')
ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc="best")
plt.grid(True, linestyle='--')
plt.ylim([0,1])
plt.xlim([0,255])
plt.show()

plt.figure()
plt.plot(gray,ac[0][0][0],label=f"A={AL[0]},a={a[0]},order={m[0]}")
plt.plot(gray,ac[0][0][1],label=f"A={AL[0]},a={a[0]},order={m[1]}")
plt.plot(gray,ac[0][1][0],label=f"A={AL[1]},a={a[0]},order={m[0]}",linestyle=':')
plt.plot(gray,ac[0][1][1],label=f"A={AL[1]},a={a[0]},order={m[1]}",linestyle=':')
plt.plot(gray,ac[0][2][0],label=f"A={AL[2]},a={a[0]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[0][2][1],label=f"A={AL[2]},a={a[0]},order={m[1]}",linestyle='--')
ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc="best")
plt.grid(True, linestyle='--')
plt.ylim([0,1])
plt.xlim([0,255])
plt.show()

plt.figure()
plt.plot(gray,ac[1][0][0],label=f"A={AL[0]},a={a[1]},order={m[0]}")
plt.plot(gray,ac[1][0][1],label=f"A={AL[0]},a={a[1]},order={m[1]}")
plt.plot(gray,ac[1][1][0],label=f"A={AL[1]},a={a[1]},order={m[0]}",linestyle=':')
plt.plot(gray,ac[1][1][1],label=f"A={AL[1]},a={a[1]},order={m[1]}",linestyle=':')

plt.plot(gray,ac[1][2][0],label=f"A={AL[2]},a={a[1]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[1][2][1],label=f"A={AL[2]},a={a[1]},order={m[1]}",linestyle='--')

ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc="best")
plt.grid(True, linestyle='--')
plt.ylim([0,1])
plt.xlim([0,255])
plt.show()

plt.figure()
plt.plot(gray,ac[2][0][0],label=f"A={AL[0]},a={a[2]},order={m[0]}")
plt.plot(gray,ac[2][0][1],label=f"A={AL[0]},a={a[2]},order={m[1]}")
plt.plot(gray,ac[2][1][0],label=f"A={AL[1]},a={a[2]},order={m[0]}",linestyle=':')
plt.plot(gray,ac[2][1][1],label=f"A={AL[1]},a={a[2]},order={m[1]}",linestyle=':')

plt.plot(gray,ac[2][2][0],label=f"A={AL[2]},a={a[2]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[2][2][1],label=f"A={AL[2]},a={a[2]},order={m[1]}",linestyle='--')

ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc="best")
plt.grid(True, linestyle='--')
plt.ylim([0,0.5])
plt.xlim([0,255])
plt.show()
```

Script 2.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator
"""
In this section, we only simulate the influence of parameter B on the phase profile keeping other
```

```
parameters constant, and how they affect the normalized intensities of the 0th order and the 1st
orders.
1. B is in the range of [0.1, 6, 3]
2. AL and AR are 1000, and M is 1.007, respectively.
3. The functions and variables are the same as in the simulation for parameters AL/AR.
"""
B=[0.1,6,3]
M=1.007
AL=1000
AR=AL
a=[0.964,1,0.7]
A=1000
m=[0,1,-1]
p=1
gray=np.arange(0,256,1)
x=np.linspace(-0.5, 0.5,640)
x_=np.linspace(0, 0.5,320)
ac=[]
for a_ in range (len(a)):
    f_linear=2*x/p
    Mc = []
    for n in range(len(B)):
        f_1=np.arctan(B[n]*(f_linear/2))
        max_index_1 = np.argmax(f_1)
        max_value_1 = f_1[max_index_1]
        f_n=f_1/max_value_1

        PhaseProfile=np.arctan(A*np.cos(f_linear*np.pi))/(np.pi)
        max_index = np.argmax(PhaseProfile)
        max_value = PhaseProfile[max_index]

        f_=np.arctan(B[n]*(x_/p))
        max_index_ = np.argmax(f_)
        max_value_ = f_[max_index_]
        f_L=f_/max_value_

        PhaseProfile_L=np.arctan(AL*np.cos(f_L*np.pi))/(np.pi)
        max_index_L = np.argmax(PhaseProfile_L)
        max_value_L = PhaseProfile_L[max_index_L]
        PhaseProfile_L=PhaseProfile_L/max_value_L

        f_r=np.arctan(B[n]*(-x_/p))
        min_index_r = np.argmin(f_r)
        min_value_r = f_r[min_index_r]
        f_R=f_/min_value_r

        PhaseProfile_R=np.arctan(AR*np.cos(f_R*np.pi))/(np.pi)
        max_index_R = np.argmax(PhaseProfile_R)
        max_value_R = PhaseProfile_R[max_index_R]
        PhaseProfile_R=PhaseProfile_R/max_value_R

        PhaseProfile_n=np.concatenate((PhaseProfile_L[::-1], PhaseProfile_R))

        pixelation=[0] * len(x)
        for i in range(len(x)):
            if x[i]<len(x)/2 and x[i]>a[a_]/2:
                pixelation[i]=0
            elif x[i]>-len(x)/2 and x[i]<-a[a_]/2:
                pixelation[i]=0
            else:
                pixelation[i]=1
        PhaseProfile_M=PhaseProfile_n*M
        mc=[]
        for k in range(len(m)):
            cc=[]
            for j in range(len(gray)):
                gray_M=M*gray[j]/255
                phaseProfile_C=PhaseProfile_M*gray_M
                ex=m[k]*np.pi*f_linear
                phase=pixelation*np.exp(1j*phaseProfile_C*np.pi)*np.exp(1j*ex)
                cm=np.sum(phase)
                cc.append(cm)
```

```
            mc.append(cc)
        Im=mc*np.conjugate(mc)/len(x)**2
        Mc.append(Im)
    ac.append(Mc)

plt.figure()
plt.plot(gray,ac[0][0][0],label=f"B={B[0]},a={a[0]},order={m[0]}")
plt.plot(gray,ac[0][0][1],label=f"B={B[0]},a={a[0]},order={m[1]}")
plt.plot(gray,ac[0][0][2],label=f"B={B[0]},a={a[0]},order={m[2]}")

plt.plot(gray,ac[0][1][0],label=f"B={B[1]},a={a[0]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[0][1][1],label=f"B={B[1]},a={a[0]},order={m[1]}",linestyle='--')
plt.plot(gray,ac[0][1][2],label=f"B={B[1]},a={a[0]},order={m[2]}",linestyle='--')
ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc="best")
plt.grid(True, linestyle='--')
plt.ylim([0,1])
plt.xlim([0,255])
plt.show()

plt.figure(3)
plt.plot(gray,ac[0][0][0],label=f"B={B[0]},a={a[0]},order={m[0]}")
plt.plot(gray,ac[0][0][1],label=f"B={B[0]},a={a[0]},order={m[1]}")
plt.plot(gray,ac[0][1][0],label=f"B={B[1]},a={a[0]},order={m[0]}",linestyle=':')
plt.plot(gray,ac[0][1][1],label=f"B={B[1]},a={a[0]},order={m[1]}",linestyle=':')

plt.plot(gray,ac[0][2][0],label=f"B={B[2]},a={a[0]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[0][2][1],label=f"B{B[2]},a={a[0]},order={m[1]}",linestyle='--')

ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc="best")
plt.grid(True, linestyle='--')
plt.ylim([0,1])
plt.xlim([0,255])
plt.show()

plt.figure(4)
plt.plot(gray,ac[1][0][0],label=f"B={B[0]},a={a[1]},order={m[0]}")
plt.plot(gray,ac[1][0][1],label=f"B={B[0]},a={a[1]},order={m[1]}")
plt.plot(gray,ac[1][1][0],label=f"B={B[1]},a={a[1]},order={m[0]}",linestyle=':')
plt.plot(gray,ac[1][1][1],label=f"B={B[1]},a={a[1]},order={m[1]}",linestyle=':')

plt.plot(gray,ac[1][2][0],label=f"B={B[2]},a={a[1]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[1][2][1],label=f"B={B[2]},a={a[1]},order={m[1]}",linestyle='--')

ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc="best")
plt.grid(True, linestyle='--')
plt.ylim([0,1])
plt.xlim([0,255])
plt.show()

plt.figure(5)
plt.plot(gray,ac[2][0][0],label=f"B={B[0]},a={a[2]},order={m[0]}")
plt.plot(gray,ac[2][0][1],label=f"B={B[0]},a={a[2]},order={m[1]}")
plt.plot(gray,ac[2][1][0],label=f"B={B[1]},a={a[2]},order={m[0]}",linestyle=':')
plt.plot(gray,ac[2][1][1],label=f"B={B[1]},a={a[2]},order={m[1]}",linestyle=':')

plt.plot(gray,ac[2][2][0],label=f"B={B[2]},a={a[2]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[2][2][1],label=f"B={B[2]},a={a[2]},order={m[1]}",linestyle='--')

ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc="best")
plt.grid(True, linestyle='--')
plt.ylim([0,0.5])
plt.xlim([0,255])
plt.show()
```

Script 3.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator
"""
In this section, we only simulate the influence of parameter M on the phase profile keeping other
parameters constant, and how they affect the normalized intensities of the 0th order and the 1st
orders.
1. M is in the range of [0.8, 1.007, 0.5]
2. AL and AR are 1000, and B is 0.1, respectively.
3. The functions and variables are the same as in the simulation for parameters AL/AR.
"""

M=[0.8,1.007,0.5]
AL=1000
AR=AL
B=0.1
a=[0.964,1,0.7]
A=1000
m=[0,1,-1]
p=1
gray=np.arange(0,256,1)
x=np.linspace(-0.5, 0.5,640)
x_=np.linspace(0, 0.5,320)
ac=[]
for a_ in range (len(a)):
    f_linear=2*x/p
    f_1=np.arctan(B*(f_linear/2))
    max_index_1 = np.argmax(f_1)
    max_value_1 = f_1[max_index_1]
    f_n=f_1/max_value_1

    PhaseProfile=np.arctan(A*np.cos(f_linear*np.pi))/(np.pi)
    max_index = np.argmax(PhaseProfile)
    max_value = PhaseProfile[max_index]

    f_=np.arctan(B*(x_/p))
    max_index_ = np.argmax(f_)
    max_value_ = f_[max_index_]
    f_L=f_/max_value_

    PhaseProfile_L=np.arctan(AL*np.cos(f_L*np.pi))/(np.pi)
    max_index_L = np.argmax(PhaseProfile_L)
    max_value_L = PhaseProfile_L[max_index_L]
    PhaseProfile_L=PhaseProfile_L/max_value_L

    f_r=np.arctan(B*(-x_/p))
    min_index_r = np.argmin(f_r)
    min_value_r = f_r[min_index_r]
    f_R=f_/min_value_r

    PhaseProfile_R=np.arctan(AR*np.cos(f_R*np.pi))/(np.pi)
    max_index_R = np.argmax(PhaseProfile_R)
    max_value_R = PhaseProfile_R[max_index_R]
    PhaseProfile_R=PhaseProfile_R/max_value_R

    PhaseProfile_n=np.concatenate((PhaseProfile_L[::-1], PhaseProfile_R))

    pixelation=[0] * len(x)
    for i in range(len(x)):
        if x[i]<len(x)/2 and x[i]>a[a_]/2:
            pixelation[i]=0
        elif x[i]>-len(x)/2 and x[i]<-a[a_]/2:
            pixelation[i]=0
        else:
            pixelation[i]=1
    Mc=[]
    for n in range(len(M)):
        PhaseProfile_M=PhaseProfile_n*M[n]
        mc=[]
```

```
        for k in range(len(m)):
            cc=[]
            for j in range(len(gray)):
                gray_M=M[n]*gray[j]/255
                phaseProfile_C=PhaseProfile_M*gray_M
                ex=m[k]*np.pi*f_linear
                phase=pixelation*np.exp(1j*phaseProfile_C*np.pi)*np.exp(1j*ex)
                cm=np.sum(phase)
                cc.append(cm)
            mc.append(cc)
        Im=mc*np.conjugate(mc)/len(x)**2
        Mc.append(Im)
    ac.append(Mc)

plt.figure()
plt.plot(gray,ac[0][0][0],label=f"M={M[0]},a={a[0]},order={m[0]}")
plt.plot(gray,ac[0][0][1],label=f"M={M[0]},a={a[0]},order={m[1]}")
plt.plot(gray,ac[0][0][2],label=f"M={M[0]},a={a[0]},order={m[2]}")

plt.plot(gray,ac[0][1][0],label=f"M={M[1]},a={a[0]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[0][1][1],label=f"M={M[1]},a={a[0]},order={m[1]}",linestyle='--')
plt.plot(gray,ac[0][1][2],label=f"M={M[1]},a={a[0]},order={m[2]}",linestyle='--')
ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.15), ncol=3)
plt.grid(True, linestyle='--')
plt.ylim([0,1])
plt.xlim([0,255])
plt.show()

plt.figure(3)
plt.plot(gray,ac[0][0][0],label=f"M={M[0]},a={a[0]},order={m[0]}")
plt.plot(gray,ac[0][0][1],label=f"M={M[0]},a={a[0]},order={m[1]}")
plt.plot(gray,ac[0][1][0],label=f"M={M[1]},a={a[0]},order={m[0]}",linestyle=':')
plt.plot(gray,ac[0][1][1],label=f"M={M[1]},a={a[0]},order={m[1]}",linestyle=':')

plt.plot(gray,ac[0][2][0],label=f"M={M[2]},a={a[0]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[0][2][1],label=f"M={M[2]},a={a[0]},order={m[1]}",linestyle='--')

ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.15), ncol=3)
plt.grid(True, linestyle='--')
plt.ylim([0,1])
plt.xlim([0,255])
plt.show()

plt.figure(4)
plt.plot(gray,ac[1][0][0],label=f"M={M[0]},a={a[1]},order={m[0]}")
plt.plot(gray,ac[1][0][1],label=f"M={M[0]},a={a[1]},order={m[1]}")
plt.plot(gray,ac[1][1][0],label=f"M={M[1]},a={a[1]},order={m[0]}",linestyle=':')
plt.plot(gray,ac[1][1][1],label=f"M={M[1]},a={a[1]},order={m[1]}",linestyle=':')

plt.plot(gray,ac[1][2][0],label=f"M={M[2]},a={a[1]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[1][2][1],label=f"M={M[2]},a={a[1]},order={m[1]}",linestyle='--')

ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.15), ncol=3)
plt.grid(True, linestyle='--')
plt.ylim([0,1])
plt.xlim([0,255])
plt.show()

plt.figure(5)
plt.plot(gray,ac[2][0][0],label=f"M={M[0]},a={a[2]},order={m[0]}")
plt.plot(gray,ac[2][0][1],label=f"M={M[0]},a={a[2]},order={m[1]}")
plt.plot(gray,ac[2][1][0],label=f"M={M[1]},a={a[2]},order={m[0]}",linestyle=':')
plt.plot(gray,ac[2][1][1],label=f"M={M[1]},a={a[2]},order={m[1]}",linestyle=':')

plt.plot(gray,ac[2][2][0],label=f"M={M[2]},a={a[2]},order={m[0]}",linestyle='--')
plt.plot(gray,ac[2][2][1],label=f"M={M[2]},a={a[2]},order={m[1]}",linestyle='--')
```

```
ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.15), ncol=3)
plt.grid(True, linestyle='--')
plt.ylim([0,0.5])
plt.xlim([0,255])
plt.show()
```

Script 4.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator
from scipy.optimize import minimize
"""
In this section, we will use our model to fit the experimental results. The model is the nonlinear
phase profile function in terms of AL, AR, B, and M.
The normalized intensities of diffraction orders will be calculated based on this model. In addition,
we will use the "minimize" algorithm to obtain a minimal RMSE between the experimental and model
results.
Once the RMSE is small enough, we are able to fit the experimental results well.
"""
# We define a model function, which can be called with several necessary arguments: an array of gray
level, roundness AL/AR, parameters B, and M.
# Then the function will return the normalized intensities of diffraction orders.
def model(gray,AL,AR,B,M,order):
    p=1
    a=0.964
    m=[order] # 0,-+1
    x_=np.linspace(0, 0.5,320)

    f_=np.arctan(B*(x_/p))
    max_index_ = np.argmax(f_)
    max_value_ = f_[max_index_]
    f_L=f_/max_value_

    PhaseProfile_L=np.arctan(AL*np.cos(f_L*np.pi))/(np.pi)
    max_index_L = np.argmax(PhaseProfile_L)
    max_value_L = PhaseProfile_L[max_index_L]
    max_value_L = max(max_value_L, 1e-10)
    PhaseProfile_L=PhaseProfile_L/max_value_L

    f_r=np.arctan(B*(-x_/p))
    min_index_r = np.argmin(f_r)
    min_value_r = f_r[min_index_r]
    f_R=f_/min_value_r

    PhaseProfile_R=np.arctan(AR*np.cos(f_R*np.pi))/(np.pi)
    max_index_R = np.argmax(PhaseProfile_R)
    max_value_R = PhaseProfile_R[max_index_R]
    max_value_R = max(max_value_R, 1e-10)
    PhaseProfile_R=PhaseProfile_R/max_value_R

    PhaseProfile_n=np.concatenate((PhaseProfile_L[::-1], PhaseProfile_R))

    PhaseProfile_M=PhaseProfile_n*M
    x=np.linspace(-0.5, 0.5,640)
    pixelation=[0] * len(x)
    for i in range(len(x)):
        if x[i]<len(x)/2 and x[i]>a/2:
            pixelation[i]=0
        elif x[i]>-len(x)/2 and x[i]<-a/2:
            pixelation[i]=0
        else:
            pixelation[i]=1
    mc=[]
    for k in range(len(m)):
        cc=[]
        for j in range(len(gray)):
            gray_M=M*gray[j]/255
```

```
            phaseProfile_C=PhaseProfile_M*gray_M
            ex=m[k]*np.pi*2*x/p
            phase=pixelation*np.exp(1j*phaseProfile_C*np.pi)*np.exp(1j*ex)
            cm=np.sum(phase)
            cc.append(cm)
        mc.append(cc)
    Im=mc*np.conjugate(mc)/len(x)**2
    return Im.real

gray_y = np.array([0,8,16,24,32,40,48,56,64,72,80,88,96
,104,112,120,128,136,144,152,160,168,176,184,192,200,208,216
,224,232,240,248,255])    # Experimental gray levels.
y_data = np.array([0.930,0.926,0.921,0.913,0.896,0.875,0.845,0.815
,0.772,0.729,0.683,0.623,0.567,0.499,0.448,0.384,0.341,0.294,0.230
,0.192,0.158,0.132,0.102,0.090,0.081,0.085,0.094,0.102,0.128,0.154
,0.188,0.222,0.243])
y_data_1 = np.array([0.000,0.001,0.004,0.009,0.017,0.027,0.041
,0.053,0.073,0.090,0.110,0.135,0.156,0.188,0.203,0.233,0.248,0.263
,0.288,0.296,0.309,0.309,0.316,0.311,0.311,0.297,0.280,0.270,0.247
,0.224,0.195,0.169,0.157])
y_data_N1 = np.array([0.000,0.001,0.004,0.009,0.017,0.027,0.041
,0.054,0.073,0.089,0.110,0.134,0.157,0.186,0.206,0.230,0.243,0.260
,0.276,0.286,0.291,0.294,0.291,0.286,0.273,0.258,0.242,0.218,0.196
,0.173,0.146,0.124,0.105])
# We define an RMSE function, which can be called with several necessary arguments: a group of
initialized parameters AL/AR, B, and M, the experimental gray levels, and corresponding normalized
diffraction order intensities.
# Then the function will return the optimal RMSE value between the model´s and the experimental
normalized intensities of diffraction orders.
def RMSE(params,gray_y, y_data,y_data_1, y_data_N1):
    AL,AR,B,M = params
    y_pred_0 = model(gray_y, AL,AR,B,M,0)
    rmse_opt_0 = np.sqrt(np.mean((y_data - y_pred_0[0]) ** 2))

    y_pred_1 = model(gray_y, AL,AR,B,M,1)
    rmse_opt_1 = np.sqrt(np.mean((y_data_1 - y_pred_1[0]) ** 2))

    y_pred_N1 = model(gray_y, AL,AR,B,M,-1)
    rmse_opt_N1 = np.sqrt(np.mean((y_data_N1 - y_pred_N1[0]) ** 2))

    rmse_opt = 1*rmse_opt_0 + 1*rmse_opt_1 + 1*rmse_opt_N1
    return  rmse_opt
# Function to update the initial guess dynamically based on previous results
def update_initial_guess(prev_results, tolerance=1e-5):
    if prev_results is not None:
        # Apply logic to update initial guess dynamically, e.g., by perturbing parameters slightly
        AL_opt, AR_opt, B_opt, M_opt = prev_results
        # Slight perturbation of optimal values
        initial_guess = [
            AL_opt + np.random.uniform(-0.5, 0.5),
            AR_opt + np.random.uniform(-0.5, 0.5),
            B_opt + np.random.uniform(-0.1, 0.1),
            M_opt + np.random.uniform(-0.05, 0.05)
        ]
    else:
        # Default initial guess if no previous results
        initial_guess = [4.067, 3.0394, 3.1, 0.8803]
    return initial_guess
# Initial guess and bounds
initial_guess = [4.067, 3.0394, 3.1, 0.8803]  # Initialize AL, AR, B, and M.
bounds = [(0, 6), (0, 5), (0, 3.1), (0, 0.9)]  # The bounds for AL, AR, B, and M.
# Fitting loop with L-BFGS-B method
tolerance = 1e-5  # Threshold for RMSE improvement
max_iterations = 20  # Maximum number of iterations
prev_results = None  # To store previous iteration results

for iteration in range(max_iterations):
    # Perform minimization with the current initial guess
    result = minimize(RMSE, initial_guess, method='L-BFGS-B', args=(gray_y, y_data, y_data_1,
y_data_N1), bounds=bounds)

    AL_opt, AR_opt, B_opt, M_opt = result.x
```

```
    minimized_rmse = result.fun

    print(f"Iteration {iteration+1}: AL={AL_opt:.4f}, AR={AR_opt:.4f}, B={B_opt:.4f}, M={M_opt:.4f},
RMSE={minimized_rmse:.4f}")
    # Calculate individual RMSE values for logging
    rmse_0 = np.sqrt(np.mean((y_data - model(gray_y, AL_opt, AR_opt, B_opt, M_opt, 0)[0]) ** 2))
    rmse_1 = np.sqrt(np.mean((y_data_1 - model(gray_y, AL_opt, AR_opt, B_opt, M_opt, 1)[0]) ** 2))
    rmse_N1 = np.sqrt(np.mean((y_data_N1 - model(gray_y, AL_opt, AR_opt, B_opt, M_opt, -1)[0]) ** 2))
    print(f"RMSE values - 0th: {rmse_0:.4f}, 1st: {rmse_1:.4f}, -1st: {rmse_N1:.4f}")
    # Stop if all individual RMSEs meet tolerance criteria
    if all(abs(rmse) < tolerance for rmse in [rmse_0, rmse_1, rmse_N1]):
        print("Converged to minimal RMSEs for all orders")
        break
    # Update previous results and initial guess for the next iteration
    prev_results = [AL_opt, AR_opt, B_opt, M_opt]
    initial_guess = update_initial_guess(prev_results)
# Now we can call the model function by inputting the optimal AL, AR, B, and M to see how close our
fitting is to the experimental results.
gray=np.arange(0,256,1)
print(AL_opt, AR_opt, B_opt,M_opt)
I_fitted = model(gray, AL_opt, AR_opt, B_opt,M_opt,0)
I_fitted_1 = model(gray, AL_opt, AR_opt, B_opt,M_opt,1)
I_fitted_N1 = model(gray, AL_opt, AR_opt, B_opt,M_opt,-1)

plt.figure()
plt.scatter(gray_y, y_data, label='Experimental Data m=0th', color='blue')
plt.plot(gray, I_fitted[0], label='Optimized Fit m=0th', color='green')

plt.scatter(gray_y, y_data_1, label='Experimental Data m=1st', color='orange')
plt.plot(gray, I_fitted_1[0], label='Optimized Fit m=1st', color='pink')

plt.scatter(gray_y, y_data_N1, label='Experimental Data m=-1st', color='purple')
plt.plot(gray, I_fitted_N1[0], label='Optimized Fit m=-1st', color='red')
plt.legend()
ax_n = plt.gca()
ax_n.yaxis.set_minor_locator(MultipleLocator(0.02))
plt.legend(loc="best")
plt.grid(True, linestyle='--')
plt.ylim([0,1])
plt.xlim([0,255])
plt.show()
plt.xlabel('Gray Level')
plt.ylabel('Normalized Intensity')
plt.show()
```

9. Diffraction gratings and holograms with chromatic control

This final chapter of the Thesis presents the generation of polychromatic diffractive elements in a special system developed by the company CAS Microstar, which combines their SLM device model FSLM-2K39-P02 introduced in Chapter 4, with an RGB diode laser. This is a three-laser diode source, with wavelengths of 662 nm (R channel), 518 nm (G channel), and 449 nm (B channel). These lasers can be switched on/off at frequencies that match the SLM display frequency, where the SLM can be switched at 180Hz, despite being a nematic LC display. Therefore it can be synchronized with the three lasers, each laser switching at 60 Hz, so that each wavelength is affected by different sequentially displayed phase patterns. By properly designing and scaling these phase masks the chromatic dispersion that characterizes diffraction can be compensated, thus creating diffraction gratings and CGH that generate polychromatic and black-and-white diffraction patterns. Therefore, in this Chapter we extend some of the studies presented in the previous chapters to now consider polychromatic computer-generated holography.

9.1. Description of the RGB laser – SLM system

Figures 9.1(a) and 9.1(b) show a picture of the CAS Microstar SLM and the multiwavelength laser. The side view of both devices is shown in Fig. 9.1(c), where there are three ports to connect each of the three lasers to the SLM, so the synchronized mode can be selected. We used a StellarNet spectrometer (model STN-BLK-C-SR) to measure the spectral characteristics of the multiwavelength laser, in which the wavelengths 662 nm (R channel), 518 nm (G channel), and 449 nm (B channel) gain the most counts, as shown in Fig. 9.1(d).

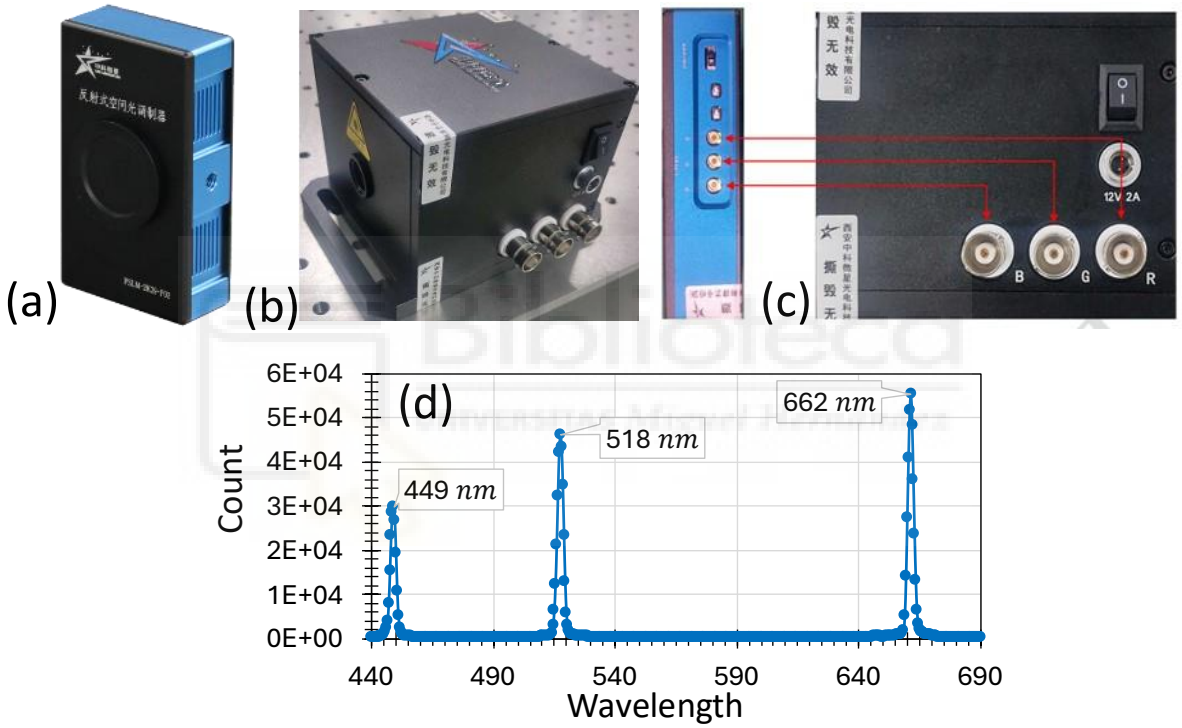

Fig. 9.1. Pictures of the CAS Microstar (a) SLM and (b) multiwavelength laser diode. (c) Side view of both devices showing the connection ports to operate in the synchronized mode. (d) Spectral characteristics of the multiwavelength laser diode.

Figure 9.2 shows several snapshots of the professional software accompanying the CAS Microstar SLM. Two languages are available, Chinese and English, which can be chosen in the "Help" menu. The option "SLM Select" lists many models of SLM that should be selected properly. In the Picture window, you can check the condition of the grayscale images and CGHs that are encoded on the SLM. "Experimental Options" offers 10 experimental functions (see Fig 9.2(b)), where only "Image Playback" is available for customers to upload their customized

hologram images or videos. Hence, this is the option chosen in this chapter. For instance, if "Diffraction of interference" is selected six subitems will be available and listed in the yellow dashed rectangle shown in Fig. 9.2(a). "Picture Window" displays a double-slit, whose size can be adjusted by "Aperture(px)" and gray level is controlled by "Gray2", where "Gray1" controls the gray level outside the slits. "Space(px)" changes the distance between the two slits, and commands "Move(px)" and "Rotate" control the position and rotation of two slits. Some general grayscale images are included as built-in options shown in Figs. 9.2(c-h).

Figure 9.3 shows the "Image Playback" having two options: "Image player" and "Video player". We use the former option since it is useful for encoding customized holograms, such as the RGB CGHs that will be developed to modulate three input wavelengths in a synchronized way. If multiple CGHs are uploaded and played at a speed of "Rate(FPS)", the corresponding reconstructions are displayed dynamically.

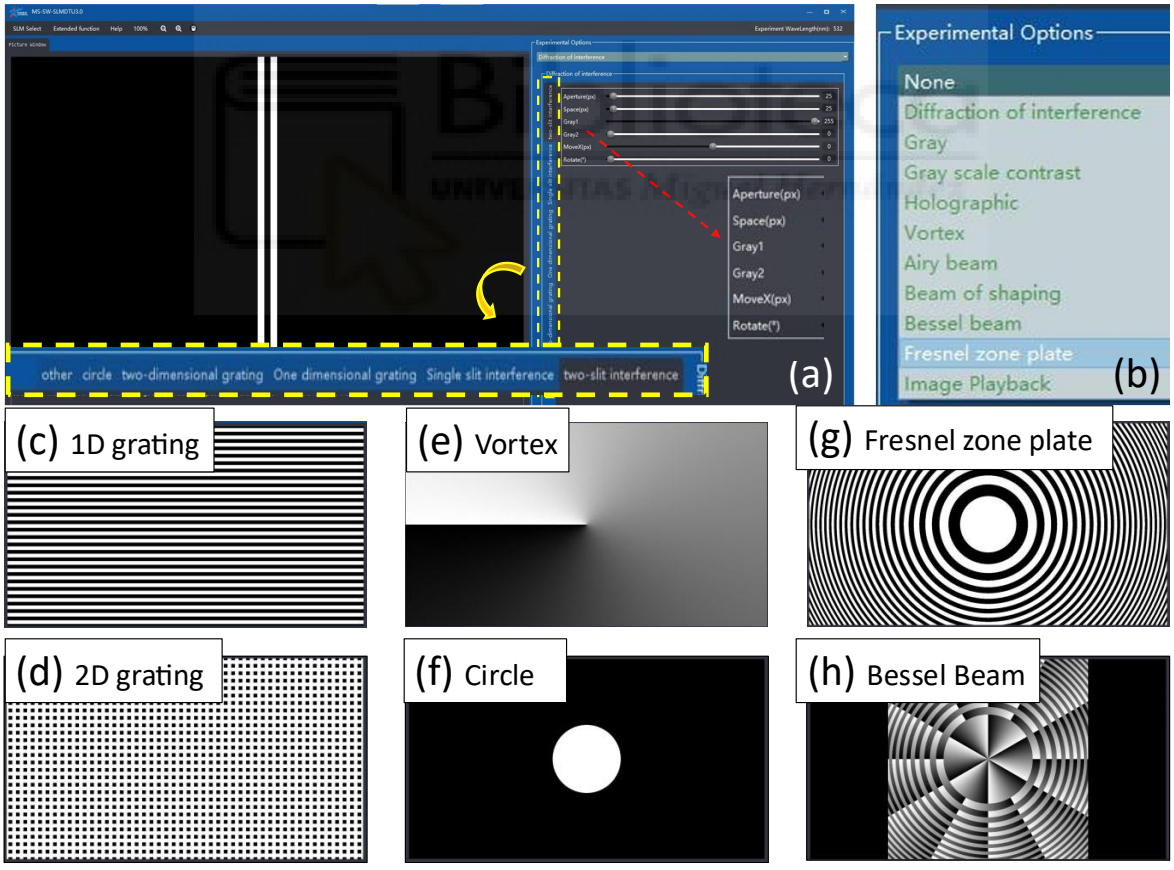

Fig. 9.2. (a) Screenshot of the software, where the subitems are marked in the yellow dashed rectangle. (b) Experimental options available in the software. Built-in grayscale images (c) 1D and (d) 2D grating, (e) Vortex, (f) Circle, (g) Fresnel zone plate, and (h) Bessel beam.

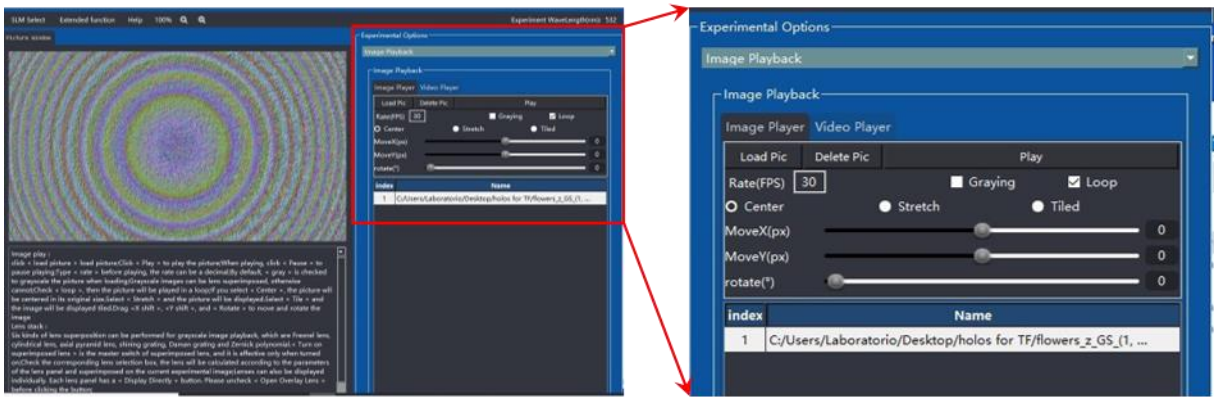

Fig. 9.3. Screenshot of the option "Image Playback", and its enlarged menu.

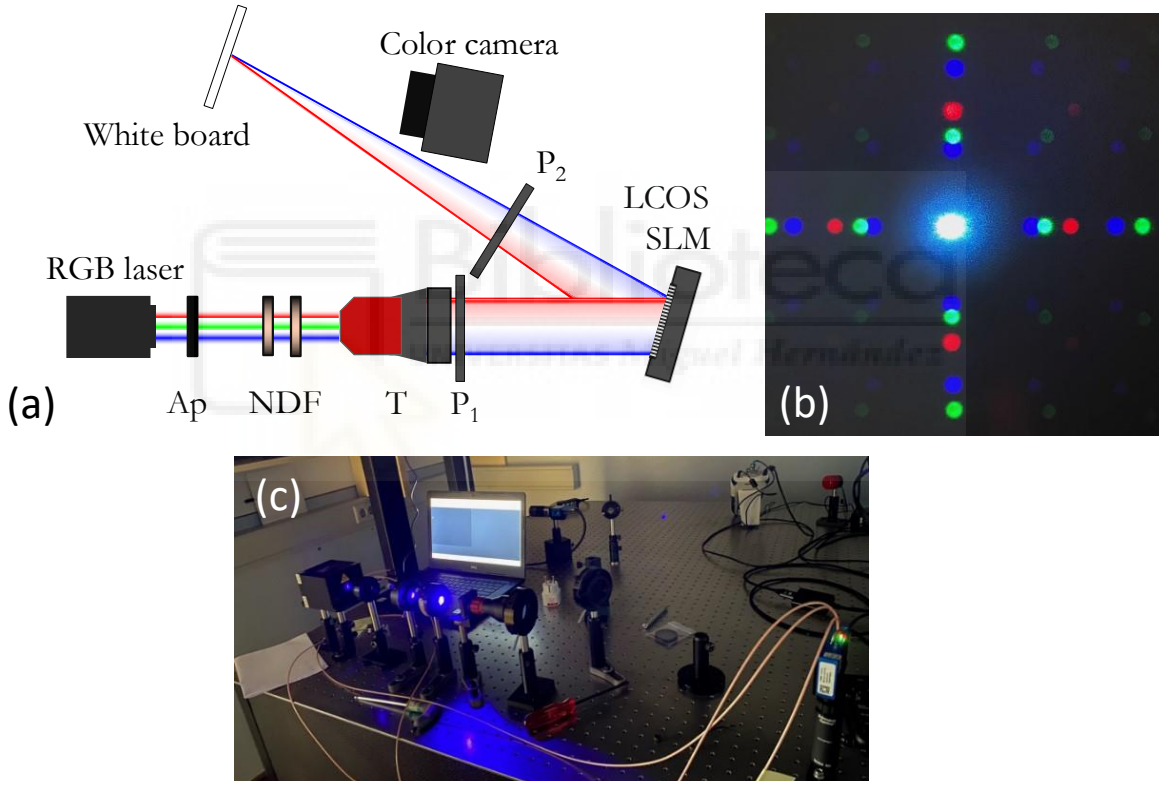

Fig. 9.4. (a) Scheme of the optical setup: Ap: circular aperture, NDF: variable neutral density filter, P: linear polarizers, T: telescope. (b) Picture of the multiwavelength beam reflected by the SLM inherent pixelated structure, showing the 2D pattern of diffraction orders affected by chromatic dispersion. (c) Experimental setup.

Figure 9.4(a) shows a scheme of the optical setup in Fig. 9.4(c). Because the three RGB lasers are not perfectly aligned, it is not possible to use a standard spatial filter. Instead, we use a telescope to magnify and collimate the beam. The Fourier transform is then projected onto a whiteboard, and a colour camera is used to capture the projected pattern on the whiteboard. As will be

described in Section 9.2, a diffractive lens is encoded on each RGB component to focus the Fourier transform onto the whiteboard.

9.2. RGB computer-generated holograms

In this section, we introduce polychromatic holography. We consider the hologram designs introduced in Chapter 6 for monochromatic light but extend to RGB CGHs. In Chapter 6 we simulated with/without the window and combined techniques for improving the quality of the reconstructions. We found that IFTA techniques improve the reconstruction of the phase-only CGH in one monochromatic channel. When we apply those IFTA algorithms to the R, G, and B channels of the original image, we are able to obtain an RGB CGH containing three monochromatic channels that yield the polychromatic reconstruction.

However, as described in Chapter 3, diffraction in the paraxial domain scales with the wavelength. Therefore, a CGH illuminated with different wavelengths produces the same reconstruction in the FT domain, but with a size that is proportional to λ. In the case of a diffraction grating this is the cause of chromatic dispersion, i.e. the separation of colours in the diffraction orders. Figure 9.4(b) shows the diffraction pattern generated when the RGB laser illuminates the SLM. The pixelated structure of the device acts as an amplitude grating that separates the three lasers, creating a characteristic 2D pattern of diffraction orders, being the red laser diffracted at the largest angle and the blue laser at the smallest angle. The diffraction pattern inherent to the pixelated device was analysed in Chapter 7 in the monochromatic situation.

As we mentioned in the previous section, RGB CGHs are the core of synchronizing the Microstar SLM with the multiwavelength laser. Because there are three grayscale images, each of them saved in the corresponding channel of the RGB CGHs, we are able to modulate different input wavelengths independently. As a result, we can compensate for chromatic dispersion, rebuild the original objects in RGB colour, or generate RGB diffraction patterns. Because of the wavelength scaling of the diffraction pattern, a colour hologram reconstruction requires compensating the chromatic dispersion by rescaling the CGH with the opposite scale. In our case, instead of rescaling the CGH, which implies interpolating the phase values between pixels, we rescale the RGB components of the desired colour object. Thus, if the target object is described with three RGB components $r(x,y)$, $g(x,y)$ and $b(x,y)$, and the SLM is illuminated with the three corresponding wavelengths λ_r, λ_g and λ_b, then the new rescaled versions of the target object´s R

and B components are calculated as:

$$r'(x,y) = \frac{\lambda_g}{\lambda_r} r(x,y), \qquad b'(x,y) = \frac{\lambda_g}{\lambda_b} b(x,y). \tag{9.1}$$

Consequently, the different IFTA procedures are applied to the new rescaled RGB object $[r'(x,y), g(x,y), b'(x,y)]$, leading to a phase-only FT CGH consisting of three RGB components $[R'(u,v), G(u,v), B'(u,v)]$, which is displayed on the SLM to generate the colour reconstruction when synchronized with the multiwavelength laser source.

Figure 9.5 illustrates this process. The goal here is to design an RGB hologram that generates a white circle. Because we only want to draw the circle edges, without the inner part, it is enough to consider the simple Fourier transform kinoform (described in Section 6.1.1), since it provides directly the edge enhanced version. The target circle components are rescaled according to Eqs. (9.1) to compensate for the chromatic dispersion. Thus, the diameter of the circle is reduced in the R component relative to the G component, while it is enlarged in the B component. The phase of the Fourier transform of these circles shows, as expected, a shape in the form of rings centred on the axis.

As shown in Fig. 9.4(a), we do not use here a glass lens to obtain the Fourier transform. Instead, we add a diffractive lens on each RGB component. This way, we avoid having to use achromatic lenses. The diffractive lens on each component is also rescaled with the wavelength, to assure the same focal plane for the three wavelengths, thus focusing the Fourier transform on the same plane. Then, each RGB phase is added to the corresponding diffractive lens, and the result is combined in one colour RGB image, namely, the final RGB composite CGH, as shown in the right part of Fig. 9.5(a). This colour image is loaded on the SLM and synchronized with the RGB laser.

Figures 9.5(b) and 9.5(c) show the experimental results. In Fig. 9.5(b) the circles had not been rescaled prior to the calculation of the hologram. Therefore, the RGB reconstructions scale with wavelength, giving a red circle bigger than the green circle, which itself is bigger than the blue circle. Note, however, that the wavelength compensation was applied to the added diffractive lenses in order to make the same focal length for the three wavelengths. This is why the three circles in Fig. 9.5(b) are focused on the same plane. Finally, Fig. 9.5(c) shows the phase holograms calculated with the compensated diffractive lens on each channel and with the rescaled circles, so the chromatic dispersion is compensated. Now we obtain a white circle, demonstrating that all three wavelengths give the same scale in the hologram reconstruction.

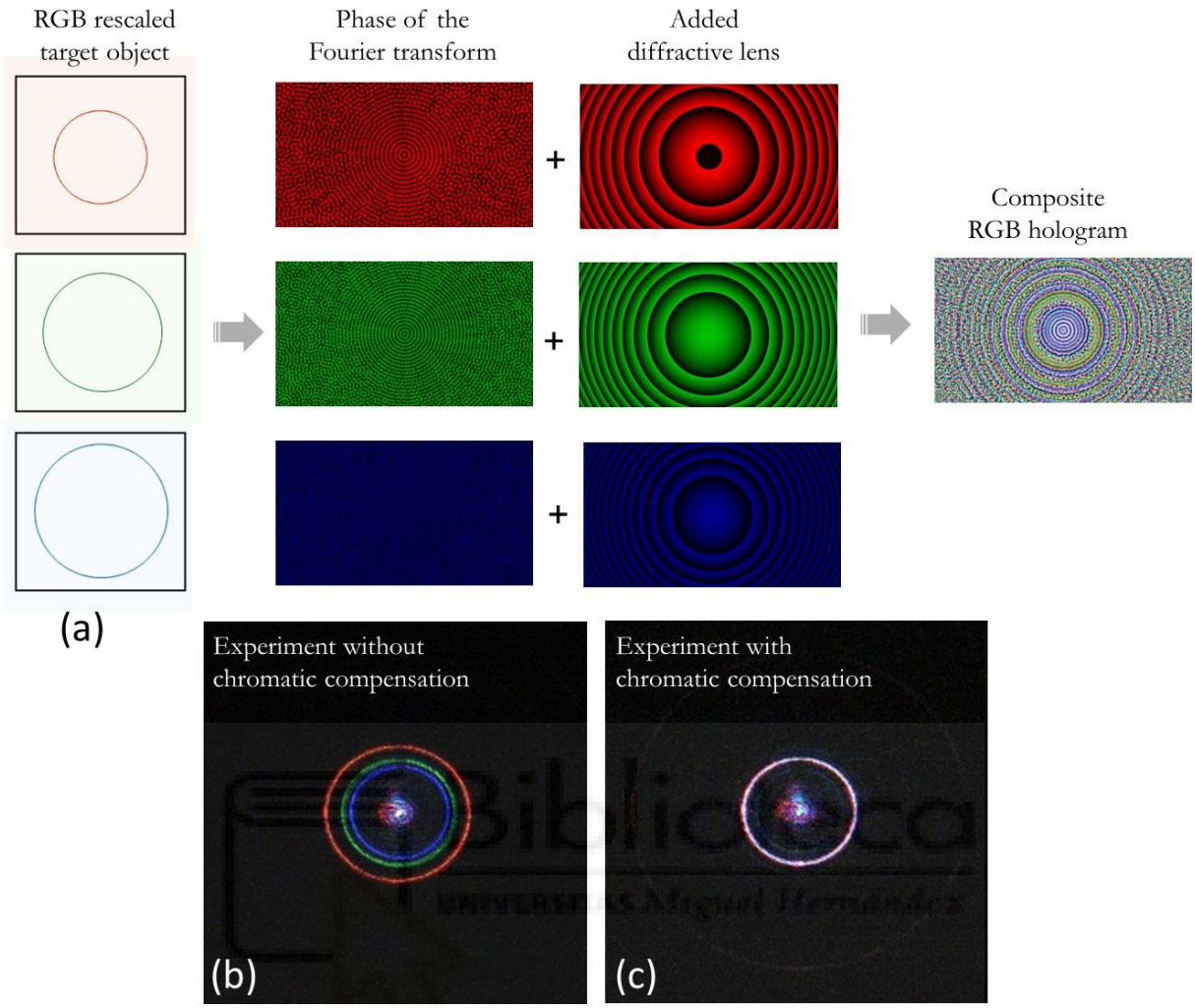

Fig. 9.5. (a) Process of calculating the RGB hologram. (b) CGH reconstruction without chromatic compensation. (c) CGH reconstruction with chromatic compensation.

9.2.1. Inverse Fourier Transform algorithm for RGB CGHs

In the IFTA procedure, the phase information of the original object can be refined in multiple loops. In each loop, the forward and inverse FT of the original object are computed in the object plane and Fourier plane alternatively. The difference between the computation of RGB CGHs and monochromatic CGHs is that we need to apply the IFTA to three channels containing different objects for generating independent R, G, and B CGHs and compensate dispersion when encoding on the SLM.

Therefore, the following preprocess is necessary for the simulation of multiple channels and the experimental generation of RGB CGHs.

Preprocess the original image (object):

1. Read the original objects in R, G, and B channels separately.
2. Only for the experiment: Rescale the object in the R and B channels, whose phase holograms are for red and blue wavelengths, with respect to the G channel, according to Eq. (9.11) and as shown in Fig. 9.5(a).
3. Apply the random phase noise approach (described in Section 6.1.3) to different channels with different object fields, which are the squared root intensity of the original object.

After that, we feed the processed channels to the IFTAs as we discussed in Chapter 6. Note that for the simulation, the rescaling step is not needed.

As shown in Fig. 9.6, the size of the original image is 1920×1920 pixels, the target pattern is a combination of three solid overlapping circles with pure red, green, and blue colours. The size of the target pattern is (500×500 pixels) quite smaller than the full size of the image, so that we chose a window size of 560×560 pixels to properly enclose the target area. Figure 9.6 shows the magnitude of the original image and the corresponding simulation magnitudes computed by the standard IFTA ($n_1 = 1, 40$), the windowed IFTA ($n_2 = 1, 40$), and the combined IFTA (marked by (n_1, n_2) pair). The standard IFTA improves the multiple magnitude distributions in the target area and mitigates the noise outside the target from $n_1 = 1$ to $n_1 = 40$. The windowed IFTA also improves the area inside the window as the iteration increases, but at the cost of outside noise. Note that the case with $n_2 = 40$ iterations leads to a smaller variation in multiple magnitudes with respect to $n_1 = 40$. For the combined IFTA, with a total iteration of 40, if the iteration of the standard IFTA grows the magnitude will increase but with a bigger variation. If the iteration of the windowed IFTA grows, then the flatness of the target area improves but with much more noise outside the window. Therefore, the case $n_1 = 20$, $n_2 = 20$ is an optimal option with a smaller variation and a higher magnitude. These results are all reconstructed in the three channels, in agreement with the monochromatic results in Chapter 6.

To quantify the reconstruction quality of the different methods, we calculate the signal-to-noise ratio (SNR) and the difference (DIF) between the reconstructed intensity and the original intensity, which were defined in Chapter 6. Here, we calculate the mean of SNR among the three channels for the polychromatic situation.

$$SNR = \frac{1}{3}(SNR_R + SNR_G + SNR_B), \tag{9.2}$$

where SNR_R, SNR_G, and SNR_B is the signal-to-noise ratio in each channel.

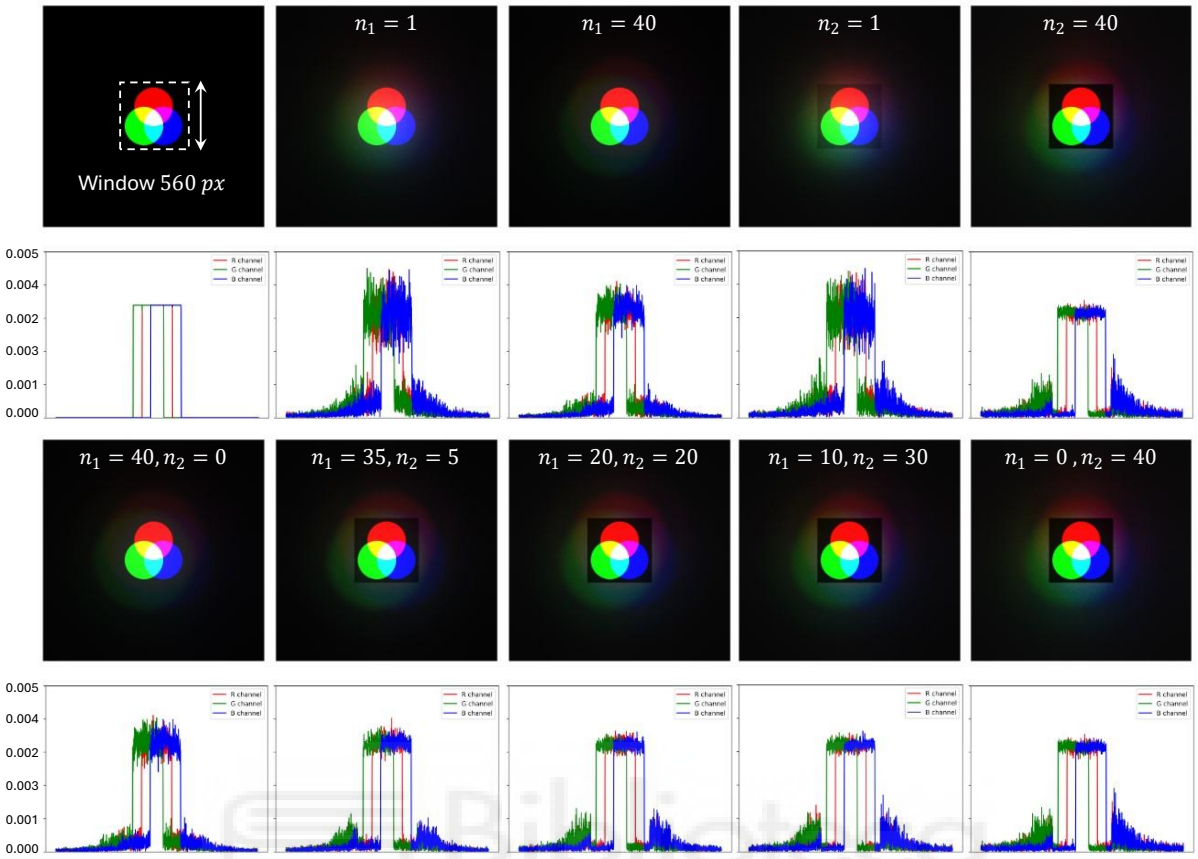

Fig. 9.6. The original RGB pattern field with a window size of 560×560 pixels, and its simulated magnitude computed by the standard IFTA ($n_1 = 1, 40$), by the windowed IFTA ($n_2 = 1, 40$), and by the combined IFTA marked as the (n_1, n_2) pair, where the total iteration is 40.

The DIF metric is calculated as the average of the DIF parameters in the three channels:

$$DIF = \frac{1}{3}(DIF_R + DIF_G + DIF_B), \tag{9.3}$$

where DIF_R, DIF_G, and DIF_B are the DIF parameters in each channel.

Figure 9.7 establishes the DIF and SNR metrics versus the total iteration number for the standard IFTA (n_1 in blue dots) and the windowed IFTA (n_2 in orange squares), and the combined IFTA (n_1 in triangles with a number). The DIF of the windowed IFTA yields better reconstructions which are much closer to the original intensity, since the n_2 curve (Fig. 9.7(a)) is lower than the n_1 curve of the standard IFTA. Oppositely, the SNR of the standard IFTA increases as the iteration number n_1 increases, being always higher than that of the windowed IFTA for the same iteration number n_2. The latter fluctuates around $SNR = 7$ after $n_2 = 5$ iterations. These two tendencies are in agreement with the simulations in the first and second row of Fig. 9.6.

Let us examine the SNR and DIF metrics for the combined IFTA versus the total iteration in Fig. 9.7. Each triangle curve in a different colour (marked by n_1 with a number) represents the combined computation, where the specific number is the iteration n_1 computed by the standard IFTA, then followed by the windowed IFTA. For a certain total iteration, when the iteration number n_1 increases, the SNR (Fig. 9.7(b)) is improved but DIF (Fig. 9.7(a)) becomes larger compared to the windowed IFTA. Therefore, we notice that if the total iteration is large, the ratio n_1/n_2 indicates the trade-off between SNR and DIF. For instance, if the total iteration is 40, the case (n_1=20, n_2=20) has a relatively high SNR and lower DIF , with $n_1/n_2 = 1$, shown in the last two rows of Fig. 9.6.

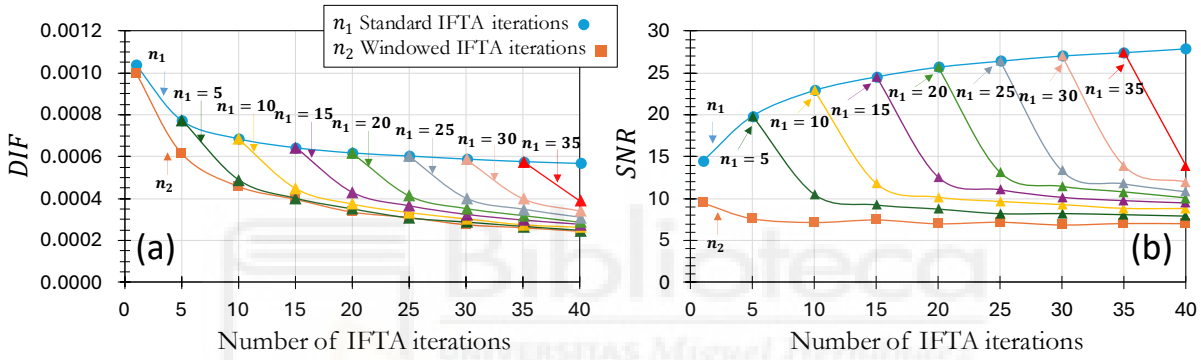

Fig. 9.7. (a) DIF and (b) SNR versus the total number of iterations, for the standard IFTA (n_1 iterations) and for the windowed IFTA (n_2 iterations). n_1 changing from 5 to 35 represents the number of iterations computed by the standard IFTA in the combined IFTA curves (indicated as triangles).

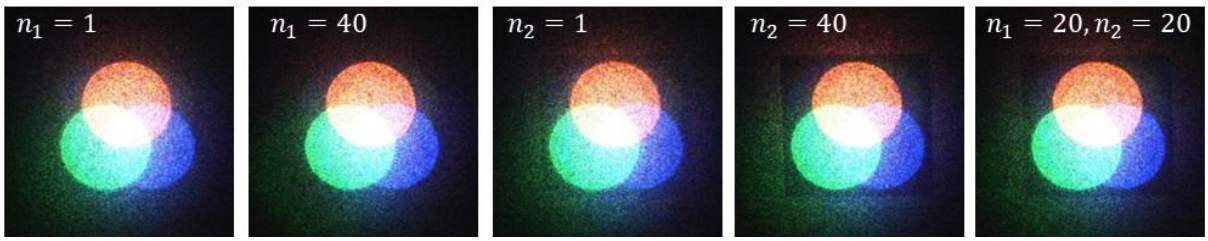

Fig. 9.8. Experimental captures of the RGB pattern for the holograms computed by standard IFTA (n_1=1, 40),windowed IFTA (n_2=1, 40), and combined IFTA (n_1=20, n_2=20).

Figure 9.8 presents the experimental reconstructions of the RGB CGHs computed by the standard IFTA, the windowed IFTA, and the combined IFTA. These results were captured using the experimental setup in Fig. 9.4, when the total iteration is 1 and 40. We can see that the result obtained when n_1=40 has less noise than that with n_1=1, the edge of the circles becomes sharper and the distribution is much more uniform. The result with n_2=40 has much more noise outside

the window and less inside noise compared to n_2=1. Also the edges and distribution are much improved. At the same time, the result with (n_1=20, n_2=20) has less outside noise than that with n_2=40, which agrees with the simulation in Fig. 9.6.

9.2.2. Iterative FT algorithm with enhanced original objects

In this section we extended the standard and windowed IFTA procedure with a more complicated and reddish scene, as shown in Fig. 9.9. In addition, we propose a new method to effectively improve the RGB CGHs of such images, where the powered input original field is exploited and, due to the nonlinear calculation, the texture sharpness and the colour transition are enhanced. Note that the human eye response is not linear with the intensity, and neither are many camera detectors. As a result, in the reconstructions, we could see much more boundaries, edges, and smooth transitions in colour.

As described in Section 9.2.1, in the preprocessing part of the IFTA procedure we apply random phase noise to the original image. Note that if the hologram is designed to visualize a given image, in principle the magnitude information is what must be the input to the algorithm, i.e., the square root (power 1/2) of the original intensity of the image. Now, we add an extra degree of freedom in the calculation which is the different power calculations of the original intensity of the image, and then calculate the field by making a square root of the powered intensity to feed into IFTA.

The target pattern is a daisy shown in Fig. 9.9, whose petals are fine and the colour transition is very natural, so it is a difficult image to reproduce holographically. The size of the target flower is 580×580 pixels and the window size is 680×680 pixels. The input object intensities are modified by different powers, and their corresponding normalized red channels are shown. For each p value, the image on the left is the powered intensity, while the image on the right is the normalized red channel. If p=1, the original intensity is not modified. Most of the pixels in the red channel tend to have high values (note that the red channel is presented such that the colourbar represents low values in white). When p increases from 2 to 4, the number of intense (high level) pixels decreases while the number of pixels carrying low value grows after normalization to the maximum value. This can be observed in how the flower appears much darker for $p = 4$ compared to $p = 1$. Therefore, the boundaries between petals become more visible in the image and, correspondingly, we can see the colour transits from 0 to 1 preventing a colour mismatch produced by the hologram procedure.

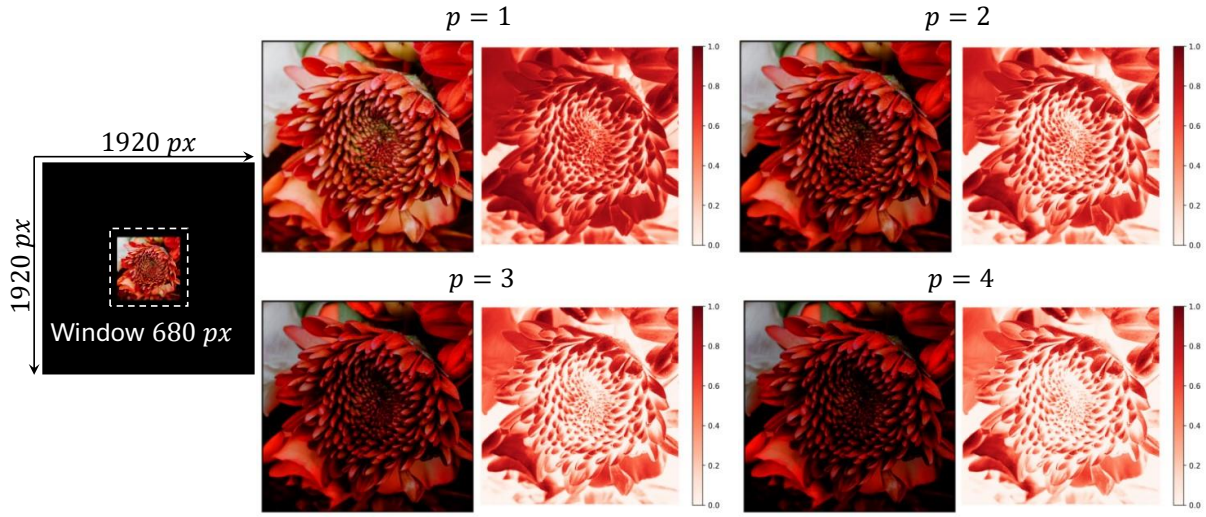

Fig. 9.9. The original daisy image and its different powered intensities for powers p = 1, 2, 3, 4. Right to each image the corresponding red channel is visualized with a colour code from white to red.

Figure 9.10 shows the computer-simulated reconstructions obtained with holograms designed for the standard IFTA (column (a)), windowed IFTA (column (b)), and their combination (column (c)). The total number of iterations is 40 in all cases. The images are shown cropped to the central part. Each row corresponds to different powers of the input intensity. For the first row where $p = 1$, the case $n_1 = 40$ gives a bright daisy and the reconstruction appears like saturated (Fig. 9.10(a)). The colour reproduction is not correct and the reconstruction appears much whiter than the original image in Fig. 9.9 (case $p = 1$), and with less bright red and white area representing edges, that induces a whiter reconstruction and ambiguous separation between petals. The case $n_2 = 40$ shows slightly less saturation effect (whiter images) with the least noise inside the window. The combined case ($n_1 = 20, n_2 = 20$) produces a better reconstruction than the other two methods, where the petals of the daisy are the clearest. The improvement in the latter two IFTA is in agreement with the previous RGB demonstration in Section 9.2.1.

The other results in Fig. 9.10 show how the saturation effect becomes alleviated as p increases, for all the IFTA methods. Increasing the value of p introduces larger variations in the intensity distributions of the RGB channels. Thus, the area carrying high intensity is smaller and details of the flower petals start to be strengthened. Consequently, the colour transition becomes more natural. Note how the windowed IFTA brings more white light outside the window, thus leading to a better colour reproduction of the flower pattern. The power value cannot be infinitely high, because the magnitude of the dim part area will be close to 0 after many power calculations, and then we would lose structures with low colour values.

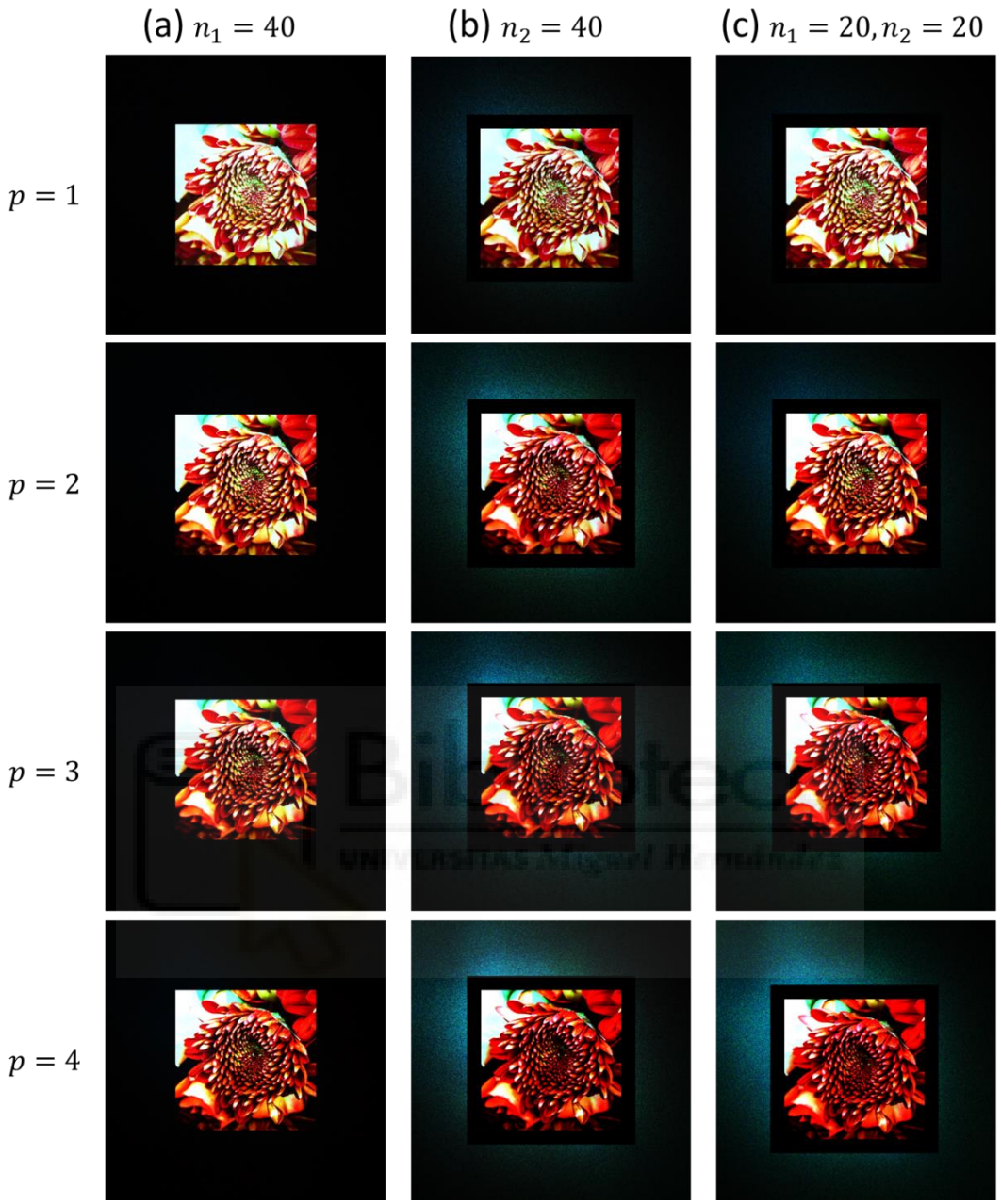

Fig. 9.10. Simulation of the reconstructed intensity of the daisy image for different power values p computed by (a) standard IFTA (b) windowed IFTA, and (c) combined IFTA.

Figure 9.11 shows the corresponding experimental results, where the hologram reconstruction is projected onto a white screen and a camera is used to capture the projected images. These reconstructions were obtained with high resolution images of 4032×3024 pixels, being a window size of 680×680 pixels. Figures 9.11(a-c) display the results obtained with standard, the windowed IFTA, and the combined IFTA, respectively, with a total iteration of 40. We can see that the experimental reconstructions are all in agreement with the simulations in Fig. 9.10.

Without the power transformation ($p = 1$ in the first row), we cannot distinguish narrow petals as clearly as with higher p values because they look whiter, like and over-exposure. The saturation decreases while increasing power value, as simulated in Fig. 9.10. Then many more details appear: we can recognize independent petals, and the colours are recovered as the original, where the white occurring on the tips of the petals and the main body of the daisy turns to red.

In all cases, these experimental images feature a white bright zero-order spot on the centre, which we ascribe to a non-ideal phase modulation response (reflection at the input surface, non-linear phase modulation, flicker, fringing,…) that may affect the employed SLM.

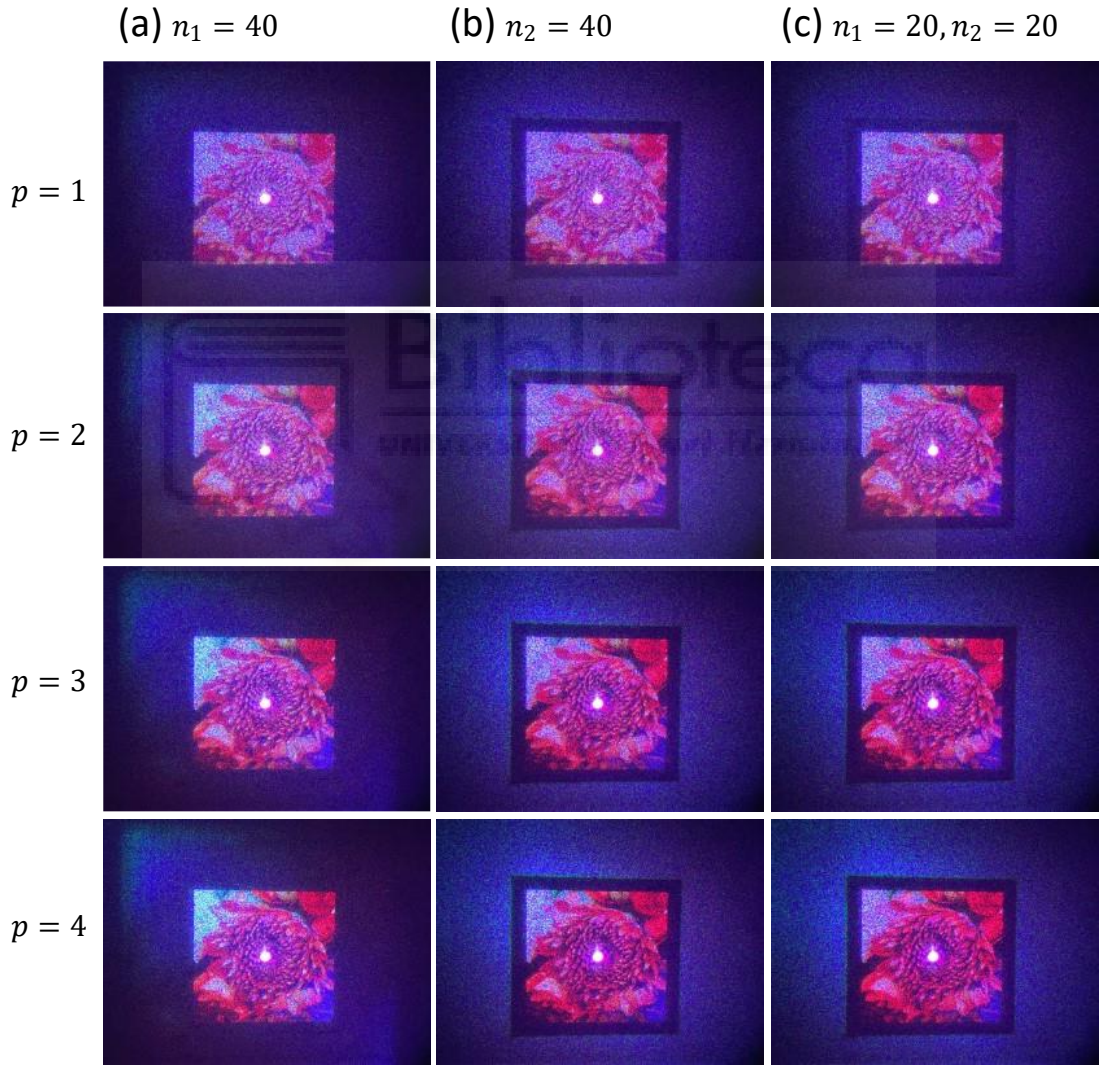

Fig. 9.11. Experimental captures of the daisy for the holograms computed by (a) standard IFTA (b) windowed IFTA, and (c) combined IFTA, with different power values p.

9.3. Correlator/convolver RGB gratings

This final section is devoted to the use of the above RGB system to encode diffraction gratings. Namely, we are interested in exploiting the Fourier transform properties described in Section 2.3 to combine computer-generated holograms and diffraction gratings that provide convolution/correlation operations between RGB signals.

Let us assume two phase-only Fourier transform CGHs $\psi_1(\mathbf{u})$ and $\psi_2(\mathbf{u})$ designed to be displayed on a phase-only SLM and produce a given function $g_1(\mathbf{x}) = \mathcal{F}[\exp(i\psi_1(\mathbf{u}))]$ and $g_2(\mathbf{x}) = \mathcal{F}[\exp(i\psi_2(\mathbf{u}))]$ in the respective hologram reconstructions. For simplicity, we use bold letters $\mathbf{x} = (x, y)$ and $\mathbf{u} = (u, v)$ to represent the transversal coordinates and spatial frequencies. Let us now consider two new phase-only functions calculated as the sum and the difference of these two phase holograms, i.e., $\psi_S(\mathbf{u}) = \psi_1(\mathbf{u}) + \psi_2(\mathbf{u})$, and $\psi_D(\mathbf{u}) = \psi_1(\mathbf{u}) - \psi_2(\mathbf{u})$. Therefore, these new phase-only holograms display the following complex products, which lead to the following Fourier transforms:

$$\mathcal{F}[\exp(i\psi_S(\mathbf{u}))] = \mathcal{F}\left[e^{i\psi_1(\mathbf{u})}e^{i\psi_2(\mathbf{u})}\right] = g_1(\mathbf{x}) * g_2(\mathbf{x}), \tag{9.4a}$$

$$\mathcal{F}[\exp(i\psi_D(\mathbf{u}))] = \mathcal{F}\left[e^{i\psi_1(\mathbf{u})}e^{-i\psi_2(\mathbf{u})}\right] = g_1(\mathbf{x}) * g_2^*(-\mathbf{x}) = g_1(\mathbf{x}) \star g_2(\mathbf{x}). \tag{9.4b}$$

The first result in Eq. (9.4a) is the convolution $[g_1 * g_2](\mathbf{x})$ between the two hologram reconstructions $g_1(\mathbf{x})$ and $g_2(\mathbf{x})$, while the second result in Eq. (9.4b) provides the cross-correlation $[g_1 \star g_2](\mathbf{x})$ between the two reconstructions.

The optical system based on the synchronized RGB laser and SLM provides the possibility to display holograms that generate time-multiplexed colour reconstructions. Here we use them to demonstrate these convolution and correlation properties in RGB images.

9.3.1. RGB CGHs with lateral shift

To illustrate the idea, Fig. 9.12 shows preliminary experimental results of an RGB CGH reconstruction with different conditions. In all cases, a single CGH $\psi(\mathbf{u})$ is calculated and designed to reconstruct a flower pattern. The hologram has been rescaled according to Eq. (9.1) to avoid the chromatic dispersion in the RGB channels, so the three hologram channels are designed as $\psi_R(\mathbf{u})$, $\psi_G(\mathbf{u})$ and $\psi_B(\mathbf{u})$. Figures 9.12(a) to 9.12(d) show the colour hologram reconstructions for different combinations of RGB channels. In Fig. 9.12(a) the RGB hologram is designed to be $[\mathrm{RGB}]_a = [\psi_R(\mathbf{u}) + \gamma_R u, 0, 0]$, where $[\mathrm{R}, \mathrm{G}, \mathrm{B}]$ denote the components of the colour hologram.

Here only the red component is displayed, and therefore only the red laser beam is modulated and the flower appears red. The constant γ_R determines the slope of a linear phase added to the original hologram in the horizontal direction. Taking into account the Fourier transform shifting property described in Section 2.2.2, this added linear phase produces a lateral horizontal shift of the hologram reconstruction, so the flower appears separated from the central DC order.

In Fig. 9.12(b) the hologram is designed as $[\mathrm{RGB}]_b = [0, \psi_G(\mathbf{u}) + \gamma_G u, 0]$, thus leading to a green reconstruction. The constant γ_G is related to γ_R so that the chromatic dispersion is compensated for the two wavelengths, i.e., $\gamma_R = \lambda_G \gamma_G / \lambda_R$, with $\lambda_R = 662$ nm and $\lambda_G = 518$ nm. In Fig. 9.12(c) the displayed hologram is $[\mathrm{RGB}]_c = [\psi_R(\mathbf{u}) + \gamma_R u, \psi_G(\mathbf{u}) + \gamma_G u, 0]$. Since the red and green laser beams are equally diffracted, the flower appears yellow. Finally, in Fig. 9.12(d) all three channels display the hologram, i.e., $[\mathrm{RGB}]_d = [\psi_R(\mathbf{u}) + \gamma_R u, \psi_G(\mathbf{u}) + \gamma_G u, \psi_B(\mathbf{u}) + \gamma_B u]$ where again the constant $\gamma_B = \lambda_G \gamma_G / \lambda_B$ with $\lambda_B = 449$ nm determines the slope of the linear phase added in the blue channel that compensates for the chromatic dispersion. The flower now appears white, as expected, because the three wavelengths are diffracted into the same flower pattern and the same position.

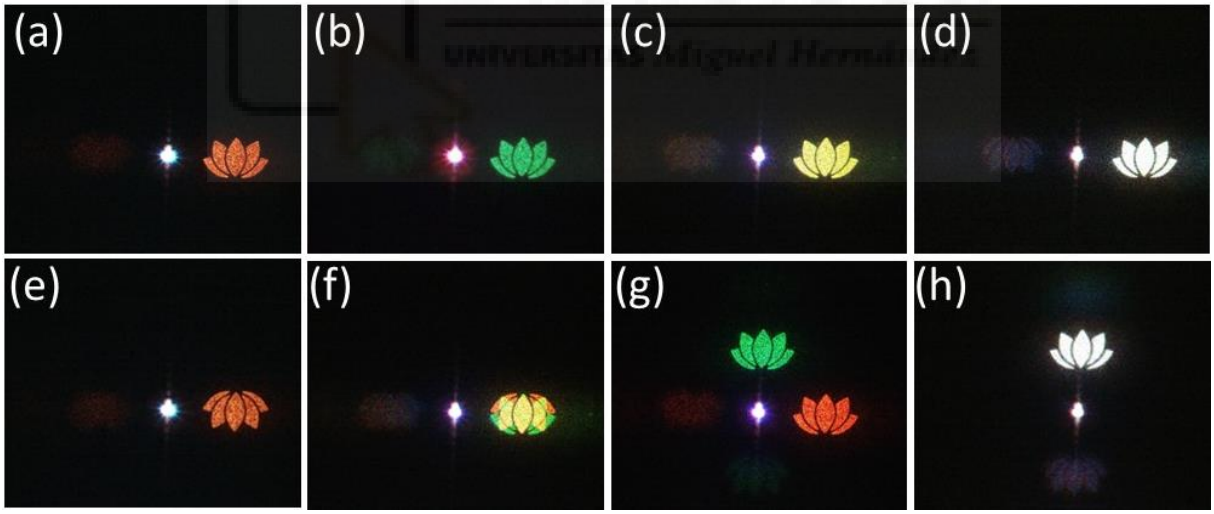

Fig. 9.12. Optical reconstruction when the following $[R, G, B]$ wavelength compensated holograms are displayed: (a) $[\psi_R(\boldsymbol{u}) + \gamma_R u, 0, 0]$, (b) $[0, \psi_G(\boldsymbol{u}) + \gamma_G u, 0]$, (c) $[\psi_R(\boldsymbol{u}) + \gamma_R u, \psi_G(\boldsymbol{u}) + \gamma_G u, 0]$, (d) $[\psi_R(\boldsymbol{u}) + \gamma_R u, \psi_G(\boldsymbol{u}) + \gamma_G u, \psi_B(\boldsymbol{u}) + \gamma_B u]$, (e) $[-\psi_R(\boldsymbol{u}) + \gamma_R u, 0, 0]$, (f) $[-\psi_R(\boldsymbol{u}) + \gamma_R u, \psi_G(\boldsymbol{u}) + \gamma_G u, 0]$, (g) $[\psi_R(\boldsymbol{u}) + \gamma_R u, \psi_G(\boldsymbol{u}) + \gamma_G v, 0]$, and (h) $[\psi_R(\boldsymbol{u}) + \gamma_R v, \psi_G(\boldsymbol{u}) + \gamma_G v, \psi_B(\boldsymbol{u}) + \gamma_B v]$.

Figure 9.12(e) shows the result when the complex conjugate version of the original phase hologram in Fig 9.12(a) is displayed, i.e. $[\mathrm{RGB}]_e = [-\psi_R(\mathbf{u}) + \gamma_R u, 0, 0]$. Note that the minus sign has been added only to the hologram, not to the linear phase. Thus, the reconstruction appears

red (only the red hologram is displayed), inverted horizontally and vertically compared to the result in Fig. 9.12(a) (due to the complex conjugation) and laterally shifted to the right (since the linear phase is the same as in Fig. 9.12(a)). The result in Fig. 9.12(f) is obtained for the colour hologram $[\mathrm{RGB}]_f = \left[-\psi_R(\mathbf{u}) + \gamma_R u, \psi_G(\mathbf{u}) + \gamma_G u, 0\right]$. The green reconstruction appears as shown in Fig. 9.12(b) while the red one appears inverted, leading to yellow only in the areas where they overlap.

Finally, Figs. 9.12(g) and 9.12(h) correspond to the colour holograms $[\mathrm{RGB}]_g = \left[\psi_R(\mathbf{u}) + \gamma_R u, \psi_G(\mathbf{u}) + \gamma_G v, 0\right]$ and $[RGB]_h = \left[\psi_R(\mathbf{u}) + \gamma_R v, \psi_G(\mathbf{u}) + \gamma_G v, \psi_B(\mathbf{u}) + \gamma_B v\right]$ respectively. They illustrate that by adding a linear phase proportional to the v coordinate, the hologram reconstruction is shifted vertically. In Fig. 9.12(g) the green component is shifted vertically while the red component is shifted horizontally, while in Fig. 9.12(h) the three components are shifted along v, thus leading to the white reconstruction in the vertical direction.

9.3.2. Optical convolution and correlation of RGB CGHs

The addition of linear phases shown in the previous section is useful to isolate the convolution/correlation terms when the sum or difference phase holograms in Eqs. (9.4) are displayed. This is illustrated in the experimental results in Figs. 9.13 and 9.14. In both figures, the left and the central columns show the optical reconstruction when the colour holograms $\psi_1(\mathbf{u})$ and $\psi_2(\mathbf{u})$ are displayed individually on the SLM. In Fig. 9.13, the hologram $\psi_1(\mathbf{u})$ generates the direct monochromatic reconstruction shifted horizontally, only in the red channel (Fig. 9.13(a1)), only in the green channel (Fig. 9.13(b1)), or only in the blue channel Fig. 9.13(c1)). Hologram $\psi_2(\mathbf{u})$ includes the three channels thus producing a white (polychromatic) reconstruction, which is shifted in the vertical direction.

The third column shows the reconstruction when the additive hologram $\psi_1(\mathbf{u}) + \psi_2(\mathbf{u})$ is displayed. The addition of the phase and the modulo 2π is calculated in each channel. Since each hologram reconstruction is shifted in a different direction, i.e., they are given by $g_1(x - x_0, y)$ and $g_2(x, y - y_0)$, where the displacements (x_0, y_0) are determined by the constants γ that multiply the linear phases, their convolution appears centred shifted in both directions, i.e. $[g_1 * g_2](x - x_0, y - y_0)$. This is shown in the results in the right column of Fig. 9.13, where the convolution of the flower results in a wide spot of light displaced in the diagonal direction.

Note that the result in Fig. 9.13(a3) shows the convolution in red light, while the flower object appears in cyan in the vertical direction. The reason is that the convolution is produced only

in the red channel, since the hologram $\psi_1(\mathbf{u})$ only includes this channel, and no convolution/correlation is happening across the channel. Therefore, the green and blue components of the hologram $\psi_2(\mathbf{u})$ convolve with the delta function on the axis, thus leading simply to its reconstruction in green and blue channels, showing a cyan flower. Figure 9.13(b) and 9.13(c) show equivalent results, but in Fig. 9.13(b3) the convolution is green, so the flower appears magenta (red + blue), while in Fig. 9.13(c3) the convolution appears blue and the flower appears yellow (red + green).

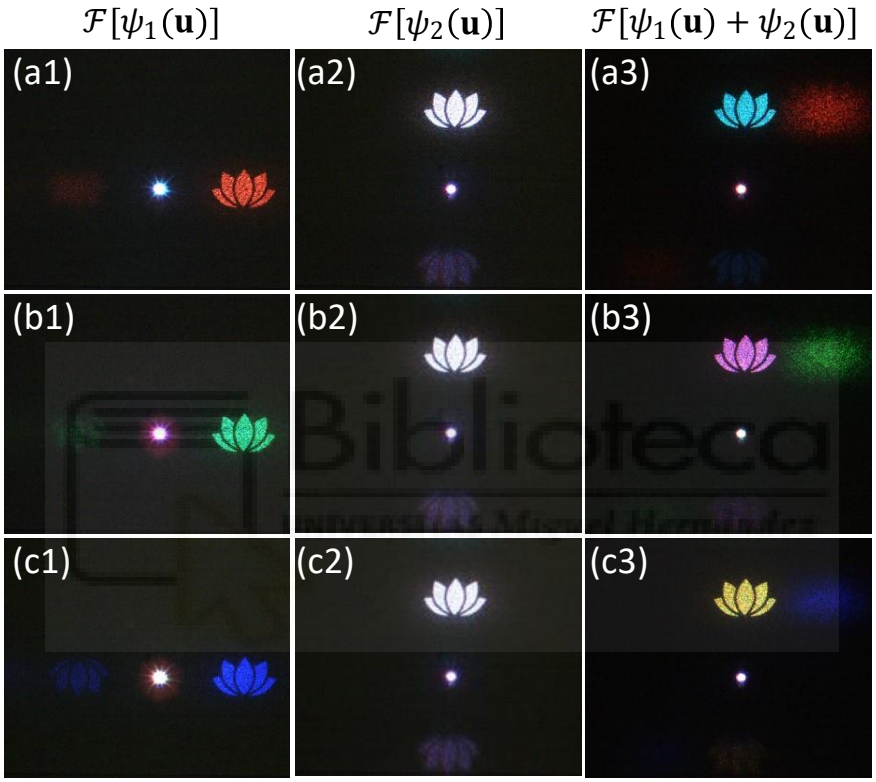

Fig. 9.13. Optical reconstruction for different combinations of $[R, G, B]$ wavelength-compensated holograms and their phase sum to provide the optical convolution.

Figure 9.14 shows equivalent results but now the complex conjugate of the hologram $\psi_1(\mathbf{u})$ is displayed, so its reconstruction appears downwards in the first column. In this situation, when the addition of the two holograms is calculated, the correlation of the original functions appears in the diagonal direction in agreement with Eq. (9.4b). This is clearly visible as a narrow bright spot in the upper right part in Figs. 9.14(a3), (b3) and (c3). Note that the auto-correlation peaks, as well as the remaining flower reconstructions, show equivalent colour content as the convolution results in Fig. 9.13. The colour of the light spot indicates the RGB component that is being correlated. When the input intensity is increased and the detector is highly saturated, the equivalent

results in Figs. 9.14(a4), (b4) and (c4) illustrate how the complementary colour (cyan, magenta, and yellow respectively) is obtained in the flower reconstruction in the vertical direction. Other weaker light spots occurring in the opposite diagonal direction are other harmonic orders caused by the non-ideal SLM phase modulation and its limited spatial resolution.

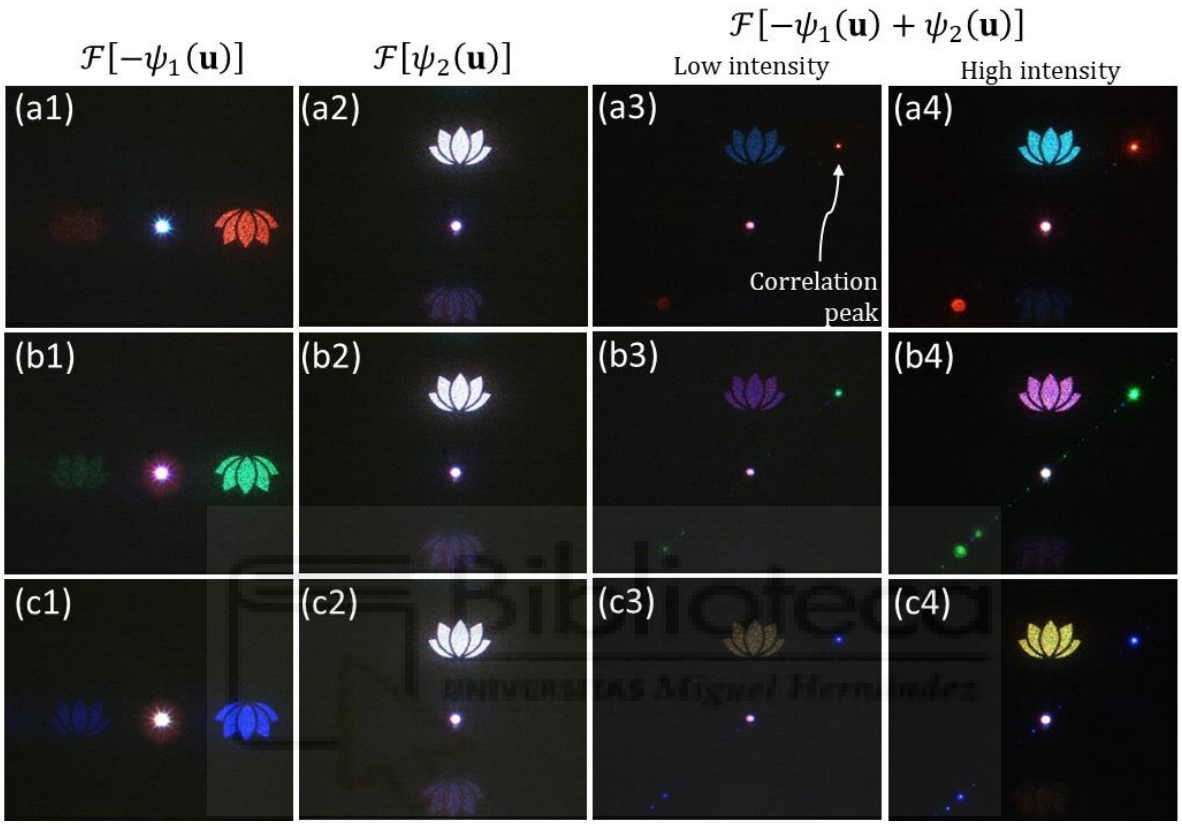

Fig. 9.14. Optical reconstruction for different combinations of $[R, G, B]$ wavelength-compensated holograms and their phase sum to provide the optical correlation.

9.3.3. Diffraction gratings with embedded CGHs

These previous results show the possibility of generating convolution and correlation functions with the proper combination of phase-only CGHs. However, as was shown in Chapter 5, by modifying the profile of the phase grating, the binary π phase profile provides two equally intense ± 1st diffraction orders, and the triplicator profile provides three diffraction orders (0th and ± 1st) with the same intensity. Then, if the phase $\psi(\mathbf{u})$ of a given phase-only CGH is modified according to such phase profiles, an equivalent distribution of diffraction orders can be generated, where each diffraction order (with index n) results in a phase term $\exp[in\psi(\mathbf{u})]$ weighted by the corresponding Fourier series coefficient. This procedure can be used to simultaneously obtain convolution and correlation in a single shot.

If the phase function $\psi(\mathbf{u})$, assumed to have values in the range $[-\pi, +\pi]$, is binarized to phase values 0 and φ, so a new phase-only hologram $\psi_{bin}(\mathbf{u})$ is defined as

$$\psi_{bin}(\mathbf{u}) = \begin{cases} 0 & \text{if } \psi(\mathbf{u}) < 0 \\ \varphi & \text{if } \psi(\mathbf{u}) \geq 0, \end{cases} \tag{9.5}$$

then the displayed hologram can be approximated by the main harmonic components. In general, the most intense terms will be the zero and the ± 1st orders, so the binarized hologram can be approximated with these three main terms as:

$$e^{i\psi_{bin}(\mathbf{u})} \cong c_1 e^{+i\psi(\mathbf{u})} + c_0 + c_{-1} e^{-i\psi(\mathbf{u})} \overset{\mathcal{F}}{\Rightarrow} c_1 g(\mathbf{x}) + c_0 \delta(\mathbf{x}) + c_{-1} g^*(-\mathbf{x}), \tag{9.6}$$

where $g(\mathbf{x}) = \mathcal{F}\{\exp[i\psi(\mathbf{u})]\}$ and where c_1, c_0 and c_{-1} are the Fourier series coefficients of the binary phase grating (Eq. (5.5b)). For instance, considering the phase level as the one for the binary triplicator ($\varphi = 0.64\pi$) then $|c_1| = |c_0| = |c_{-1}|$, and these three terms carry together 86.4% of the energy. If the phase is $\varphi = \pi$, then $c_1 = c_{-1} = 2/\pi$ and $c_0 = 0$. The ± 1st terms carry 80.1% of the beam's energy.

The CGH Fourier transform, as indicated in Eq. (9.6), can be considered basically as the superposition of $g(\mathbf{x})$ weighted by c_1, its inverted complex conjugate version $g^*(-\mathbf{x})$ weighted by c_{-1}, and a delta function weighted by c_0. Note that if the hologram includes a linear phase (like in the examples included in Section 9.3.1) its binarized version results in the opposite direction of the lateral shift of the ± 1st orders. The experimental results in Fig. 9.15 show these effects. In Fig. 9.15(a) a phase-only RGB hologram $\psi_1(\mathbf{u}) = \psi(\mathbf{u})$ is designed to generate the lotus flower centred on axis. Figure 9.15(b) shows the result when the lotus flower object is shifted laterally. The RGB hologram is calculated as $\psi_2(\mathbf{u}) = \text{BIN}[\psi(\mathbf{u}) - \gamma u]$, where BIN stands for the binarization. The resulting hologram $\psi_2(\mathbf{u})$ exhibits the three terms in Eq. (9.6), and now the reconstruction shows two laterally shifted ± 1st orders, one with the direct version of the flower and another with the inverted version, and a central zero order with the shape of a bright spot corresponding to the delta function.

Finally, if the two holograms are combined by adding their phases (the complex function is multiplied), the resulting new hologram can be written as

$$\begin{aligned} e^{i\psi_1(\mathbf{u})} e^{i\psi_2(\mathbf{u})} &\cong e^{i\psi(\mathbf{u})} \{c_1 e^{+i\psi(\mathbf{u})} e^{-i\gamma u} + c_0 + c_{-1} e^{-i\psi(\mathbf{u})} e^{+i\gamma u}\} = \\ &= c_1 e^{+i2\psi(\mathbf{u})} e^{-i\gamma u} + c_0 e^{+i\psi(\mathbf{u})} + c_{-1} e^{+i\gamma u}. \end{aligned} \tag{9.7}$$

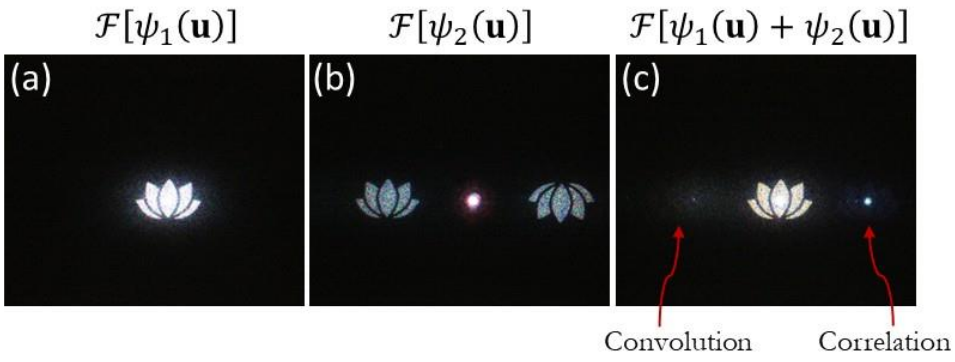

Fig. 9.15. Optical reconstruction for the $[R, G, B]$ wavelength compensated phase-only holograms designed as: (a) Continuous phase hologram providing optical reconstruction on-axis. (b) Binary phase hologram providing laterally shifted reconstruction. (c) Addition of the phases of the two holograms.

Therefore, its Fourier transform is given by:

$$\mathcal{F}\left\{e^{i[\psi_1(\mathbf{u})+\psi_2(\mathbf{u})]}\right\} = \; c_1[g * g](x + x_0, y) + c_0 g(x, y) + c_{-1}\delta(x - x_0, y). \tag{9.8}$$

The corresponding experiment is shown in Fig. 9.15(c), which illustrates the generation of the convolution term located on the left, weighted by the Fourier coefficient c_1, the reconstruction of the lotus flower on axis, with a weight given by c_0, and the generation of a bright spot on the right part, corresponding to the delta function autocorrelation, which is weighted by the Fourier coefficent c_{-1}.

The procedure can be extended to create a two-dimensional array of different terms by combining a horizontal and a vertical grating. This is illustrated in the experimental results presented in Figs. 9.16(a) and 9.16(b). In Fig 9.16(a) a linear phase proportional to the horizontal spatial frequency u is added to the hologram before binarizing the phase. Then each of the RGB components is rescaled to compensate for the wavelength dispersion. As a result, Fig. 9.16(a) shows two equally intense replicas, one shifted to the right corresponding to the direct lotus function $g(x - x_0, y)$, and one shifted to the left corresponding to the inverted complex-conjugate version, $g^*(-x - x_0, -y)$. An additional zero-order bright spot appears in the centre corresponding to the $\delta(\mathbf{x})$ function in Eq. (9.6). Note that this zero-order term shows up even in the case of the binary π-phase profile, since any defect of the SLM phase modulation leads to a DC term that is focused in the centre. When the linear phase is added to the hologram proportional to the vertical spatial frequency v before the phase binarization, the two patterns appear shifted vertically, as shown in Fig. 9.16(b), where now the lotus functions appear as $g(x, y - y_0)$ and $g^*(-x, -y - y_0)$ respectively.

Finally, when the two binarized phase holograms are added together, the result shows a pattern of different diffraction order terms. If we consider the approximation in Eq. (9.6), the combination yields an array of 3×3 terms. The diagonal orders correspond to the convolution terms, while the antidiagonal orders correspond to the correlation terms. The orders located along the x or the y direction, reproduce the lotus flower, as shown in Figs. 9.16(c) and 9.16(d). In the former case the phase levels of the binarized phase correspond to a π phase difference, so only the orders $(\pm 1, \pm 1)$ are visible. On the contrary, the phase level in Fig. 9.16(d) was selected to reproduce the triplicator. In this case, the four orders aligned horizontally and vertically reproduce the lotus. The central order located on the axis, reproduces the delta function.

Thus, these experiments illustrate the possibility of using the RGB SLM system to produce optical convolver/correlator phase diffraction gratings using the triplicator profiles.

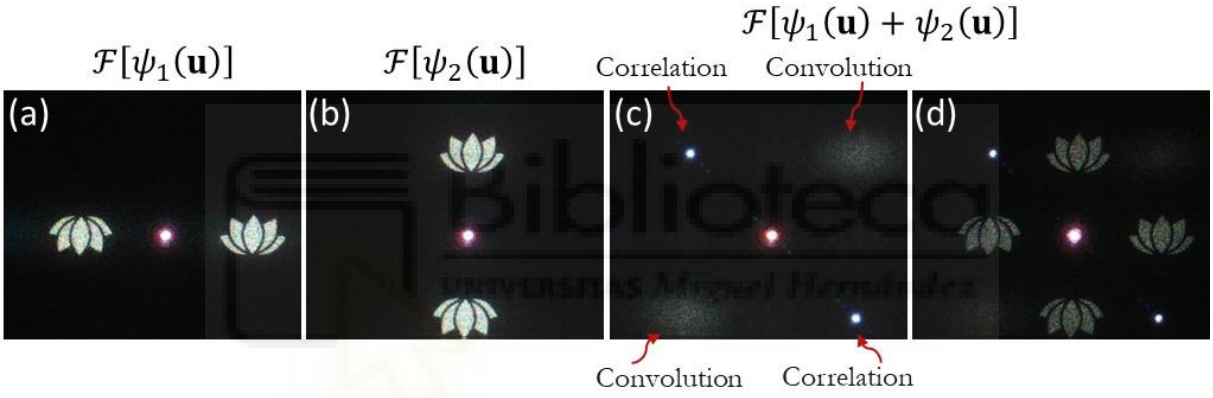

Fig. 9.16. Optical reconstruction for the $[R, G, B]$ holograms: (a) binary phase CGH in the horizontal direction; (b) binary phase CGH in the vertical direction; (c) combined CGH where the phase levels are adjusted to provide binary π phase response; (d) combined CGH where the phase levels are adjusted to provide a triplicator response.

Python Codes

Script 1: Hologram design and IFTA algorithms

This Python script is used to build the standard and windowed IFTA, the combined IFTA and the power control, both the simulation and for the generation of RGB phase-only CGHs to be displayed on the SLM for experiments.

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft2, fftshift,ifft2,ifftshift
import os
import cv2
import time
```

```
start_t=time.time()
os.chdir("C:/Users/Laboratorio/MakeHologram/Wirtingers_Holography")
filename="RGB_500"
im=plt.imread(f"{filename}.png")

height=im.shape[0]
width=im.shape[1]
""" Rescaling is only applied for generating the holograms for the experiment."""
# Define wavelengths (in meters, for rescale)
lambda_r = 0.662e-6  # Red wavelength
lambda_g = 0.518e-6  # Green wavelength (reference)
lambda_b = 0.449e-6  # Blue wavelength
# Calculate scaling factors with respect to green
scale_r =  lambda_r/lambda_g
scale_b =  lambda_b/lambda_g

h_r, w_r = int(height * scale_r), int(width * scale_r)
h_b, w_b = int(height * scale_b), int(width * scale_b)
print(h_r,int(h_r), w_r,int(w_r))
print(h_b,int(h_b), w_b,int(w_b))

x_offset_r = (int(w_r)-width) // 2
y_offset_r = (int(h_r)-height) // 2

x_offset_b = (width - int(w_b)) // 2
y_offset_b = (height - int(h_b)) // 2
# Step 1: Pad the red channel to the scaled size
padded_red = np.zeros((h_r, w_r))
padded_red[y_offset_r:y_offset_r + height, x_offset_r:x_offset_r + width] = im[:, :, 0]
# Step 2: Crop the blue channel to the scaled size
cropped_blue=np.zeros((h_b, w_b))
cropped_blue = im[:, :, 2][y_offset_b:y_offset_b + h_b, x_offset_b:x_offset_b + w_b]
# Step 3: Resize both channels back to the original dimensions
scaled_red = cv2.resize(padded_red, (width, height), interpolation=cv2.INTER_LINEAR)
scaled_blue = cv2.resize(cropped_blue, (width, height), interpolation=cv2.INTER_LINEAR)
#<<----------------------End of rescaling preparation------------------------------>>
power1=4 # Power control
# Power-controlled intensity for different channels. For the simulation, we only need to change the
rescaled red and blue channels to unrescaled channels.
#R
im_shift_r=fftshift(scaled_red**power1)
#G
im_shift_g=fftshift(im[:,:,1]**power1)
#B
im_shift_b=fftshift(scaled_blue**power1)
# Genaration of random phase noise.
rand=np.random.uniform(0, 1, (height, width))
rand_2pi=np.pi*rand
rand_ma=np.max(rand_2pi)
rand_mi=np.min(rand_2pi)
exp_rand=np.exp(1j*rand_2pi)
# Multiply noise with the field of the preprocessed original objects.
im_r_rand=exp_rand*np.sqrt(im_shift_r)
im_g_rand=exp_rand*np.sqrt(im_shift_g)
im_b_rand=exp_rand*np.sqrt(im_shift_b)
# Forward FFT of different channels
current_field_r =fftshift(fft2(im_r_rand ))
current_field_g =fftshift(fft2(im_g_rand))
current_field_b =fftshift(fft2(im_b_rand))

iterations1=20 # Iteration number for the IFTA without window.
iterations2=20 # Iteration number for the IFTA with window.
# Loop for the IFTA without window.
for j in range(iterations1):
    # Reconstrution field that is computed by Inverse FFT2 of the phase value from the previous
forward FFT2 computation, and the amplitudes are 1.
    current_field_r_i = ifft2(ifftshift(np.exp(1j * np.angle(current_field_r))))
    current_field_g_i = ifft2(ifftshift(np.exp(1j * np.angle(current_field_g))))
    current_field_b_i = ifft2(ifftshift(np.exp(1j * np.angle(current_field_b))))
    # Create new reconstruction fields, where the amplitudes are original object fields.
    current_field_r_n =np.sqrt(im_shift_r)*np.exp(1j * np.angle(current_field_r_i))
```

```
    current_field_g_n =np.sqrt(im_shift_g)*np.exp(1j * np.angle(current_field_g_i))
    current_field_b_n =np.sqrt(im_shift_b)*np.exp(1j * np.angle(current_field_b_i))
    # Compute Forward FFT2 to obtain target phase holograms.
    current_field_r = fftshift(fft2(current_field_r_n ))
    current_field_g = fftshift(fft2(current_field_g_n ))
    current_field_b = fftshift(fft2(current_field_b_n ))
# Loop for the IFTA with window.
for i in range(iterations2):
    # Inverse FFT2 computation that is the same as standard IFTA loop.
    current_field_r_i = ifft2(ifftshift(np.exp(1j * np.angle(current_field_r))))
    current_field_g_i = ifft2(ifftshift(np.exp(1j * np.angle(current_field_g))))
    current_field_b_i = ifft2(ifftshift(np.exp(1j * np.angle(current_field_b))))
    # Retrieve both amplitudes and phase of inverse FFT2 computation, and normalize amplitude part by
the maximum in each channel.
    current_field_r_i_t =np.sqrt(abs(current_field_r_i)**2/np.max(abs(current_field_r_i)**2))
*np.exp(1j * np.angle(current_field_r_i))
    current_field_g_i_t = np.sqrt(abs(current_field_g_i)**2/np.max(abs(current_field_g_i)**2))
*np.exp(1j * np.angle(current_field_g_i))
    current_field_b_i_t =np.sqrt(abs(current_field_b_i)**2/np.max(abs(current_field_b_i)**2))
*np.exp(1j * np.angle(current_field_b_i))
    # Create new object fields with original fields, and the phases are the same as the previous step.
    current_field_r_n =np.sqrt(im_shift_r)*np.exp(1j * np.angle(current_field_r_i))#*exp_rand
    current_field_g_n =np.sqrt(im_shift_g)*np.exp(1j * np.angle(current_field_g_i))#*exp_rand
    current_field_b_n =np.sqrt(im_shift_b)*np.exp(1j * np.angle(current_field_b_i))#*exp_rand

    l=620 # Define window size, which is the length from the window edge to the image edge.
    c_w,c_h=width//2,height//2
    # For the experiment, rescale the window size for red and blue channels.
    lr=int((height*(scale_r-1)+2*l)/(2*scale_r))
    lb=int((height*(scale_b-1)+2*l)/(2*scale_b))
    # In the window, the amplitudes are from the original. Otherwise, the amplitudes are from
retrieved amplitudes in inverse FFT2 computation.
    current_field_r_n[:,c_w-lr:c_w+lr]=current_field_r_i_t[:,c_w-lr:c_w+lr]
    current_field_g_n[:,c_w-l:c_w+l]=current_field_g_i_t[:,c_w-l:c_w+l]
    current_field_b_n[:,c_w-lb:c_w+lb]=current_field_b_i_t[:,c_w-lb:c_w+lb]

    current_field_r_n[c_h-lr:c_h+lr,0:c_w-lr]=current_field_r_i_t[c_h-lr:c_h+lr,0:c_w-lr]
    current_field_g_n[c_h-l:c_h+l,0:c_w-l]=current_field_g_i_t[c_h-l:c_h+l,0:c_w-l]
    current_field_b_n[c_h-lb:c_h+lb,0:c_w-lb]=current_field_b_i_t[c_h-lb:c_h+lb,0:c_w-lb]

    current_field_r_n[c_h-lr:c_h+lr,c_w+lr:width]=current_field_r_i_t[c_h-lr:c_h+lr,c_w+lr:width]
    current_field_g_n[c_h-l:c_h+l,c_w+l:width]=current_field_g_i_t[c_h-l:c_h+l,c_w+l:width]
    current_field_b_n[c_h-lb:c_h+lb,c_w+lb:width]=current_field_b_i_t[c_h-lb:c_h+lb,c_w+lb:width]
    # Compute Forward FFT2 to obtain phase holograms with window.
    current_field_r = fftshift(fft2(current_field_r_n ))
    current_field_g = fftshift(fft2(current_field_g_n ))
    current_field_b = fftshift(fft2(current_field_b_n ))
# Phase modulation depth for the experiment, and they are all 2 for simulation.
rdp,gdp,bdp=1.78,2.56,3.5
# Final optimized phase for encoding on SLM, turn angles to grayscales.
optimized_phase_r = np.angle(current_field_r)
phase_rr_modi=(optimized_phase_r/np.pi+1)*(255/rdp)

optimized_phase_g = np.angle(current_field_g)
phase_gr_modi=(optimized_phase_g/np.pi+1)*(255/gdp)

optimized_phase_b = np.angle(current_field_b)
phase_br_modi=(optimized_phase_b/np.pi+1)*(255/bdp)

"""Lens"""
lambda_r = 0.662e-6
lambda_g = 0.518e-6
lambda_b = 0.449e-6
arr_r=np.zeros((height, width))
arr_g=np.zeros((height, width))
arr_b=np.zeros((height, width))
pixel_size=4.5e-6
f=-2 # focal length (meters).
center_h=height//2
center_w=width//2
# Calculate the phase of lenses according to Eq. (9.1) for three wavelengths.
for i in range(height):
```

```
    for j in range(width):
        r = pixel_size * np.sqrt((i - center_h)**2 + (j - center_w)**2)
        arr_r[i, j] =  r**2 / (f * lambda_r)
        arr_g[i, j] =  r**2 / (f * lambda_g)
        arr_b[i, j] =  r**2 / (f * lambda_b)
# Change the range of the phase into [0,2].
arr_r_mod=np.mod(arr_r,2)
arr_g_mod=np.mod(arr_g,2)
arr_b_mod=np.mod(arr_b,2)
# Map the phase to grayscale by corresponding phase modulation.
arr_r_modified=arr_r_mod*(255/rdp)
arr_g_modified=arr_g_mod*(255/gdp)
arr_b_modified=arr_b_mod*(255/bdp)
# Save and crop the corresponding RGB CGHs with/without lens for experiment and simulation,
respectively.
im_modify_noL = np.zeros_like(im,shape=(im.shape[0], im.shape[1], 3))
im_modify_noL[:,:,0] = phase_rr_modi
im_modify_noL[:,:,1] = phase_gr_modi
im_modify_noL[:,:,2] = phase_br_modi

im_modify_c = np.zeros_like(im, shape=(im.shape[0], im.shape[1],3))
im_modify_c[:,:,0] = phase_rr_modi+arr_r_modified
im_modify_c[:,:,1] = phase_gr_modi+arr_g_modified
im_modify_c[:,:,2] = phase_br_modi+arr_b_modified

def crop(im_modify,name):
    y_offset=center_h-1080//2
    im_cropped=im_modify[y_offset:y_offset+1080,:]
    im_cropped = im_cropped.astype(np.uint8)
    im_modi = Image.fromarray(im_cropped)
    im_modi.save(f"{filename}_ IFTA _It1_{iterations1}_It2_{iterations2}_{name}_p {power1}.png")

C=crop(im_modify_c, "L")
C_noL=crop(im_modify_noL, "nL")

end_t=time.time()
print(f"Time consuming {end_t-start_t}s, iteration {iterations1+iterations2}")
```

Script 2: Hologram reconstruction

This second script applies the inverse FFT to rebuild the holograms' reconstruction based on the different IFTA methods. The parameters *SNR* and *DIF* used to evaluate the quality of the reconstruction compared to the original object are calculated as well.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import ifft2,ifftshift

""" This file can process multiple holograms together and automatically, where the SNR and Diff are
calculated for each one."""
# Read the path of the original image.
original_path2="C:/Users/Laboratorio/MakeHologram/FFT_CGH_thesis/SNR_Diff/fl_one.png"
# Save the iteration numbers in different arrays.
r1=[0]
r2=[1,40]
r3=[20]
power=[1, 2, 3, 4]
# n1 and n2 are iterations for the IFTA without and with the window.
for n1 in r2:

    for n2 in r1:
        # n2=40-n1
        # Use n1, n2, and p to complete the file path for reading different holograms.
        for p in power:
            file_path1 =f"C:/Users/Laboratorio/OneDrive/Documents/Microstar/Simulation of difference
of CGHs/SNR_Diff small RGB and fl_TF/fl/L620/fl_one_ GS _n1_{n1},n2_{n2}_p_{p},nl.png"
            # Read RGB CGHs.
            holo1 = plt.imread(file_path1)
```

```
            height=holo1.shape[0]
            width=holo1.shape[1]
            # Convert grayscales to angles.
            holo1_R_angle=holo1[:,:,0]*2*np.pi
            holo1_G_angle=holo1[:,:,1]*2*np.pi
            holo1_B_angle=holo1[:,:,2]*2*np.pi

            # Compute inverse FFT of phase holograms to get reconstructions.
            Reconstruction_holo1_r = ifftshift(ifft2(np.exp(1j * holo1_R_angle)))
            Reconstruction_holo1_g = ifftshift(ifft2(np.exp(1j * holo1_G_angle)))
            Reconstruction_holo1_b = ifftshift(ifft2(np.exp(1j * holo1_B_angle)))

            l=620
            c_w,c_h=width//2,height//2
            lh,lw=height-2*l,width-2*l # The size of the window.
            I1_r=np.abs(Reconstruction_holo1_r)**2
            I1_r_normalized = I1_r / np.sum(I1_r[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]) # All the
sum is 1.
            I1_g=np.abs(Reconstruction_holo1_g)**2
            I1_g_normalized = I1_g / np.sum(I1_g[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2])
            I1_b=np.abs(Reconstruction_holo1_b)**2
            I1_b_normalized = I1_b / np.sum(I1_b[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2])
            # Normalized Original
            original2=plt.imread(original_path2)
            original2_r=original2[:,:,0]/np.sum(original2[:,:,0][c_h-lh//2:c_h+lh//2, c_w-
lw//2:c_w+lw//2])
            original2_g=original2[:,:,1]/np.sum(original2[:,:,1][c_h-lh//2:c_h+lh//2, c_w-
lw//2:c_w+lw//2])
            original2_b=original2[:,:,2]/np.sum(original2[:,:,2][c_h-lh//2:c_h+lh//2, c_w-
lw//2:c_w+lw//2])

            # SNR in the window.
            SNR_r=np.sum(I1_r[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2])/(np.sum(I1_r)-
np.sum(I1_r[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]))
            SNR_g=np.sum(I1_g[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2])/(np.sum(I1_g)-
np.sum(I1_g[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]))
            SNR_b=np.sum(I1_b[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2])/(np.sum(I1_b)-
np.sum(I1_b[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]))
            SNR=(SNR_r+SNR_g+SNR_b)/3

            diff_r=original2_r[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]-I1_r_normalized[c_h-
lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]
            diff_g=original2_g[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]-I1_g_normalized[c_h-
lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]
            diff_b=original2_b[c_h-lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]-I1_b_normalized[c_h-
lh//2:c_h+lh//2, c_w-lw//2:c_w+lw//2]
            # Diff in the window with respect to the original.
            D_r=np.sqrt(np.sum((diff_r)**2))
            D_g=np.sqrt(np.sum((diff_g)**2))
            D_b=np.sqrt(np.sum((diff_b)**2))
            D=(D_r+D_g+D_b)/3
            print(f"n1 {n1} n2 {n2} p{ p}: SNR={SNR} Diff={D}")
            max_Diff=2.5e-06
            # Save the absolute of the difference distribution between the reconstruction and the
original, here the absolute calculation is applied.
            plt.figure()
            plt.imshow(abs(diff_r)+abs(diff_g)+abs(diff_b),vmax=max_Diff,cmap="YlGnBu")
            plt.colorbar()
            plt.axis("off")
            plt.savefig(f"Diff n1_{n1} n2_{n2}.png", dpi=300, bbox_inches='tight')
            plt.close()

            f=500**2 # Small value for seeing noise.
            I1_rgb=np.zeros_like(holo1)
            I1_rgb[:, :, 0] = I1_r_normalized*f
            I1_rgb[:, :, 1] = I1_g_normalized*f
            I1_rgb[:, :, 2] = I1_b_normalized*f
            # Save the reconstructed intensity.
            plt.figure()
            plt.imshow(I1_rgb)
            plt.axis("off")
```

```
            plt.savefig(f"Full Re I n1_{n1} n2_{n2} p_{p}.png", dpi=300, bbox_inches='tight')
            plt.close()
            # Save the reconstructed magnitude.
            M_rgb=np.zeros_like(holo1)
            M_rgb[:, :, 0] = np.abs(Reconstruction_holo1_r)*f
            M_rgb[:, :, 1] = np.abs(Reconstruction_holo1_g)*f
            M_rgb[:, :, 2] = np.abs(Reconstruction_holo1_b)*f

            plt.figure()
            plt.imshow(M_rgb)
            plt.axis("off")
            plt.savefig(f"Full Re Mag n1_{n1} n2_{n2} p_{p}.png", dpi=300, bbox_inches='tight')
            plt.close()
            # Save the intensity profile crossing the centre row.
            plt.figure()
            plt.plot(I1_r_normalized[c_h,:], label="R channel", color='red')
            plt.plot(I1_g_normalized[c_h,:], label="G channel", color='green')
            plt.plot(I1_b_normalized[c_h,:], label="B channel", color='blue')
            plt.ylim(0,2.5e-5)
            plt.legend()
            plt.gca().xaxis.set_visible(False)
            plt.savefig(f"I profile n1_{n1} n2_{n2} p_{p}.png", dpi=300, bbox_inches='tight')
            plt.close()
            # Save the magnitude profile crossing the centre row.
            plt.figure()
            plt.plot(np.abs(Reconstruction_holo1_r)[c_h,:], label="R channel", color='red')
            plt.plot(np.abs(Reconstruction_holo1_g)[c_h,:], label="G channel", color='green')
            plt.plot(np.abs(Reconstruction_holo1_b)[c_h,:], label="B channel", color='blue')
            plt.ylim(0,0.005)
            plt.legend()
            plt.gca().xaxis.set_visible(False)
            plt.savefig(f"Mag profile n1_{n1} n2_{n2} p_{p}.png", dpi=300, bbox_inches='tight')
            plt.close()
# Save the original object magnitude and profile crossing the centre row.
M_o=np.zeros_like(holo1)
M_o[:, :, 0] = np.sqrt(original2_r)*f
M_o[:, :, 1] = np.sqrt(original2_g)*f
M_o[:, :, 2] = np.sqrt(original2_b)*f

plt.figure()
plt.imshow(M_o)
plt.axis("off")
plt.savefig("Full Mag Or.png", dpi=300, bbox_inches='tight')
plt.close()

plt.figure()
plt.plot(original2_r[c_h,:], label="R channel", color='red')
plt.plot(original2_g[c_h,:], label="G channel", color='green')
plt.plot(original2_b[c_h,:], label="B channel", color='blue')
plt.ylim(0,2.5e-5)
plt.legend()
plt.gca().xaxis.set_visible(False)
plt.savefig("I profile Or.png", dpi=300, bbox_inches='tight')
plt.close()
```

Script 3. Convolution and Correlation

This script is used to simulate the convolution and correlation operations.

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft2, fftshift
import os
import cv2

os.chdir("C:/Users/Laboratorio/MakeHologram/Con_Cor")
filename="Lotus"
```

```
im_o=plt.imread(f"{filename}.png")
h,w=im_o[:,:,0].shape
# Genaration of random phase noise.
rand=np.random.uniform(0, 1, (h, w,3))
rand_2pi=2*np.pi*rand
rand_ma=np.max(rand_2pi)
rand_mi=np.min(rand_2pi)
exp_rand=np.exp(1j*rand_2pi)
exp_rand_conjugate=np.exp(-1j*rand_2pi)
# Multiply the phase noise with the original pattern.
im_noise=im_o[:,:,:3]*exp_rand
# Multiplied by a complex conjugate phase noise.
im_noise_conjugate=im_o[:,:,:3]*exp_rand_conjugate

height=1920
width=1920
center_h=height//2
center_w=width//2
length=-400 # shift distance.
# Lotus with noise in the centre.
im_c = np.zeros((height, width, 3), dtype=complex)
im_c[center_h-150:center_h+150,center_w-150:center_w+150]=im_noise
# Lotus with noise shifts along the y-axis.
im_v = np.zeros((height, width, 3), dtype=complex)
im_v[center_h-150+length:center_h+150+length,center_w-150:center_w+150]=im_noise
# Lotus with noise shifts along the x-axis to the left.
im_l = np.zeros((height, width, 3), dtype=complex)
im_l[center_h-150:center_h+150,center_w-150+length:center_w+150+length]=im_noise
# Inverted Lotus with complex conjugate noise shifts along the x-axis to the left.
im_noise_con_flipped = np.flip(im_noise_conjugate, axis=(0, 1))
im_l_fl = np.zeros((height, width, 3), dtype=complex)
im_l_fl[center_h-150:center_h+150,center_w-150+length:center_w+150+length]=im_noise_con_flipped

"""Rescaling"""
# Define wavelengths (in meters, for rescale)
lambda_r = 0.662e-6  # Red wavelength
lambda_g = 0.518e-6  # Green wavelength (reference)
lambda_b = 0.449e-6  # Blue wavelength
# Calculate scaling factors with respect to green
scale_r =  lambda_r/lambda_g
scale_b =  lambda_b/lambda_g

h_r, w_r = int(height * scale_r), int(width * scale_r)
h_b, w_b = int(height * scale_b), int(width * scale_b)

x_offset_r = (int(w_r)-width) // 2
y_offset_r = (int(h_r)-height) // 2

x_offset_b = (width - int(w_b)) // 2
y_offset_b = (height - int(h_b)) // 2
# Pass the target images, and then the rescaled channels are returned.
def rescale(im):
    # Step 1: Pad the red channel to the scaled size
    padded_red = np.zeros((h_r, w_r), dtype=complex)
    padded_red[y_offset_r:y_offset_r + height, x_offset_r:x_offset_r + width] = im[:, :, 0]
    # Step 2: Crop the blue channel to the scaled size
    cropped_blue=np.zeros((h_b, w_b), dtype=complex)
    cropped_blue = im[:, :, 2][y_offset_b:y_offset_b + h_b, x_offset_b:x_offset_b + w_b]
    # Step 3: Resize both channels back to the original dimensions
    real_partr = cv2.resize(np.real(padded_red), (width, height), interpolation=cv2.INTER_LINEAR)
    imag_partr = cv2.resize(np.imag(padded_red), (width, height), interpolation=cv2.INTER_LINEAR)
    scaled_red = real_partr + 1j * imag_partr

    real_partb = cv2.resize(np.real(cropped_blue), (width, height), interpolation=cv2.INTER_LINEAR)
    imag_partb = cv2.resize(np.imag(cropped_blue), (width, height), interpolation=cv2.INTER_LINEAR)
    scaled_blue = real_partb + 1j * imag_partb  # Recombine into complex format
    #R
    im_shift_r=fftshift(scaled_red)
    #G
    im_shift_g=fftshift(im[:,:,1])
    #B
    im_shift_b=fftshift(scaled_blue)
```

```
    return im_shift_r,im_shift_g,im_shift_b
# Rescaled channels for specific images.
im_v_re_r,im_v_re_g,im_v_re_b=rescale(im_v)
im_l_re_r,im_l_re_g,im_l_re_b=rescale(im_l)
im_l_re_r_fl,im_l_re_g_fl,im_l_re_b_fl=rescale(im_l_fl)

"""FFT calculation"""
# Phase modulation depth for R, G, and B lasers.
rdp,gdp,bdp=1.85,2.63,3.55
def FFT(im_shift_r,im_shift_g,im_shift_b):
    im_r_rand = np.abs(im_shift_r) ** 0.5 * np.exp(1j * np.angle(im_shift_r))
    im_g_rand = np.abs(im_shift_g) ** 0.5 * np.exp(1j * np.angle(im_shift_g))
    im_b_rand = np.abs(im_shift_b) ** 0.5 * np.exp(1j * np.angle(im_shift_b))
    # FFT of different channels
    current_field_r =fftshift(fft2(im_r_rand ))
    current_field_g =fftshift(fft2(im_g_rand))
    current_field_b =fftshift(fft2(im_b_rand))
    # Convert phase to grayscale.
    phase_r = np.angle(current_field_r)
    phase_g = np.angle(current_field_g)
    phase_b = np.angle(current_field_b)

    return phase_r,phase_g,phase_b
# Corresponding phases in units of radians when the channels are passed.
v_phase_r,v_phase_g,v_phase_b=FFT(im_v_re_r,im_v_re_g,im_v_re_b)
h_phase_r,h_phase_g,h_phase_b=FFT(im_l_re_r,im_l_re_g,im_l_re_b)
h_phase_r_fl,h_phase_g_fl,h_phase_b_fl=FFT(im_l_re_r_fl,im_l_re_g_fl,im_l_re_b_fl)

""" Convolution calculation"""
# Pass two phases that are from two images, and then the convoluted phase will be returned.
def con(phase_r1,phase_r2,phase_g1,phase_g2,phase_b1,phase_b2):
    con_r=np.exp(1j*phase_r1)*np.exp(1j*phase_r2)
    con_r=np.angle(con_r)+np.pi
    con_r=np.mod(con_r,2*np.pi)

    con_g=np.exp(1j*phase_g1)*np.exp(1j*phase_g2)
    con_g=np.angle(con_g)+np.pi
    con_g=np.mod(con_g,2*np.pi)

    con_b=np.exp(1j*phase_b1)*np.exp(1j*phase_b2)
    con_b=np.angle(con_b)+np.pi
    con_b=np.mod(con_b,2*np.pi)

    phase_r_conv=(con_r/np.pi)*(255/rdp)
    phase_g_conv=(con_g/np.pi)*(255/gdp)
    phase_b_conv=(con_b/np.pi)*(255/bdp)

    return phase_r_conv,phase_g_conv,phase_b_conv
# Convolution and correlation phases are obtained depending on the input phases.
con_r,con_g,con_b=con(v_phase_r, h_phase_r, v_phase_g, h_phase_g, v_phase_b, h_phase_b)
cor_r,cor_g,cor_b=con(v_phase_r, h_phase_r_fl, v_phase_g, h_phase_g_fl, v_phase_b, h_phase_b_fl)

"""Convert angles to grayscale"""
def convToGray(phase_r,phase_g,phase_b):
    r_gray=(phase_r/np.pi+1)*(255/rdp)
    g_gray=(phase_g/np.pi+1)*(255/gdp)
    b_gray=(phase_b/np.pi+1)*(255/bdp)

    return r_gray,g_gray,b_gray
# Phases in grayscale for the phases without convolution.
Sv_r,Sv_g,Sv_b=convToGray(v_phase_r,v_phase_g,v_phase_b)
Sh_r,Sh_g,Sh_b=convToGray(h_phase_r,h_phase_g,h_phase_b)
Shfl_r,Shfl_g,Shfl_b=convToGray(h_phase_r_fl,h_phase_g_fl,h_phase_b_fl)

"""Lens"""
lambda_r = 0.662e-6
lambda_g = 0.518e-6
lambda_b = 0.449e-6
arr_r=np.zeros((height, width))
arr_g=np.zeros((height, width))
arr_b=np.zeros((height, width))
```

```
pixel_size=4.5e-6
f=-2 # focal length (meters).
# Calculate the phase of lenses.
for i in range(height):
    for j in range(width):
        r = pixel_size * np.sqrt((i - center_h)**2 + (j - center_w)**2)
        arr_r[i, j] =  r**2 / (f * lambda_r)
        arr_g[i, j] =  r**2 / (f * lambda_g)
        arr_b[i, j] =  r**2 / (f * lambda_b)
# Change the range of the phase into [0,2].
arr_r_mod=np.mod(arr_r,2)
arr_g_mod=np.mod(arr_g,2)
arr_b_mod=np.mod(arr_b,2)
# Map the phase to grayscale by corresponding phase modulation.
arr_r_modified=arr_r_mod*(255/rdp)
arr_g_modified=arr_g_mod*(255/gdp)
arr_b_modified=arr_b_mod*(255/bdp)

# Crop the holograms to adapt to the size of SLM.
def crop(im_modify,name):
    y_offset=center_h-1080//2
    im_cropped=im_modify[y_offset:y_offset+1080,:]
    im_cropped = im_cropped.astype(np.uint8)
    im_modi = Image.fromarray(im_cropped)
    im_modi.save(f"{name}.png")
# converge RGB holograms.
def saveim(p_r,p_g,p_b):

    im_modify_c = np.zeros((height, width, 3), dtype=np.float32)
    im_modify_c[:,:,0] = p_r+arr_r_modified
    im_modify_c[:,:,1] = p_g+arr_g_modified
    im_modify_c[:,:,2] = p_b+arr_b_modified

    im_modify_c_r = np.zeros((height, width, 3), dtype=np.float32)
    im_modify_c_r[:,:,0] = p_r+arr_r_modified

    im_modify_c_g = np.zeros((height, width, 3), dtype=np.float32)
    im_modify_c_g[:,:,1] = p_g+arr_g_modified

    im_modify_c_b = np.zeros((height, width, 3), dtype=np.float32)
    im_modify_c_b[:,:,2] = p_b+arr_b_modified

    return im_modify_c,im_modify_c_r,im_modify_c_g,im_modify_c_b

saved_1=saveim(con_r,con_g,con_b)[0]
im_l_save=crop(saved_1,f"VH0 con L")
saved_1=saveim(con_r,Sv_g,Sv_b)[0]
im_l_save=crop(saved_1,f"VH0 con R_L")
saved_1=saveim(Sv_r,con_g,Sv_b)[0]
im_l_save=crop(saved_1,f"VH0 con G_L")
saved_1=saveim(Sv_r,Sv_g,con_b)[0]
im_l_save=crop(saved_1,f"VH0 con B_L")

saved_1=saveim(cor_r,cor_g,cor_b)[0]
im_l_save=crop(saved_1,f"VH180 cor L")

saved_1=saveim(cor_r,Sv_g,Sv_b)[0]
im_l_save=crop(saved_1,f"VH180 cor R_L")

saved_1=saveim(Sv_r,cor_g,Sv_b)[0]
im_l_save=crop(saved_1,f"VH180 cor G_L")

saved_1=saveim(Sv_r,Sv_g,cor_b)[0]
im_l_save=crop(saved_1,f"VH180 cor B_L")

end_t=time.time()
print(f"Time consuming {end_t-start_t}s")
```

Script 4. Convolution and Correlation for Binarized phase.

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft2, fftshift
import os
import cv2

os.chdir("C:/Users/Laboratorio/MakeHologram/Con_Cor")
filename="Lotus"
im_o=plt.imread(f"{filename}.png")
h,w=im_o[:,:,0].shape
# Genaration of random phase noise.
rand=np.random.uniform(0, 1, (h, w,3))
rand_2pi=2*np.pi*rand
rand_ma=np.max(rand_2pi)
rand_mi=np.min(rand_2pi)
exp_rand=np.exp(1j*rand_2pi)
exp_rand_conjugate=np.exp(-1j*rand_2pi)

im_noise=im_o[:,:,:3]*exp_rand
im_noise_conjugate=im_o[:,:,:3]*exp_rand_conjugate
im_noise_con_flipped = np.flip(im_noise_conjugate, axis=(0, 1))  # Flip both vertically & horizontally

height=1920
width=1920
center_h=height//2
center_w=width//2
length=-400

im_c = np.zeros((height, width, 3), dtype=complex)
im_c[center_h-150:center_h+150,center_w-150:center_w+150]=im_noise

im_v = np.zeros((height, width, 3), dtype=complex)
im_v[center_h-150+length:center_h+150+length,center_w-150:center_w+150]=im_noise

im_l = np.zeros((height, width, 3), dtype=complex)
im_l[center_h-150:center_h+150,center_w-150+length:center_w+150+length]=im_noise

im_l_fl = np.zeros((height, width, 3), dtype=complex)
im_l_fl[center_h-150:center_h+150,center_w-150+length:center_w+150+length]=im_noise_con_flipped

"""Rescaling """
# Define wavelengths (in meters, for rescale)
lambda_r = 0.662e-6  # Red wavelength
lambda_g = 0.518e-6  # Green wavelength (reference)
lambda_b = 0.449e-6  # Blue wavelength
# Calculate scaling factors with respect to green
scale_r =  lambda_r/lambda_g
scale_b =  lambda_b/lambda_g

h_r, w_r = int(height * scale_r), int(width * scale_r)
h_b, w_b = int(height * scale_b), int(width * scale_b)

x_offset_r = (int(w_r)-width) // 2
y_offset_r = (int(h_r)-height) // 2

x_offset_b = (width - int(w_b)) // 2
y_offset_b = (height - int(h_b)) // 2

def rescale(im):
    # Step 1: Pad the red channel to the scaled size
    padded_red = np.zeros((h_r, w_r), dtype=complex)
    padded_red[y_offset_r:y_offset_r + height, x_offset_r:x_offset_r + width] = im[:, :, 0]
    # Step 2: Crop the blue channel to the scaled size
    cropped_blue=np.zeros((h_b, w_b), dtype=complex)
    cropped_blue = im[:, :, 2][y_offset_b:y_offset_b + h_b, x_offset_b:x_offset_b + w_b]
    # Step 3: Resize both channels back to the original dimensions
    real_partr = cv2.resize(np.real(padded_red), (width, height), interpolation=cv2.INTER_LINEAR)
    imag_partr = cv2.resize(np.imag(padded_red), (width, height), interpolation=cv2.INTER_LINEAR)
```

```
    scaled_red = real_partr + 1j * imag_partr

    real_partb = cv2.resize(np.real(cropped_blue), (width, height), interpolation=cv2.INTER_LINEAR)
    imag_partb = cv2.resize(np.imag(cropped_blue), (width, height), interpolation=cv2.INTER_LINEAR)
    scaled_blue = real_partb + 1j * imag_partb  # Recombine into complex format
    #R
    im_shift_r=fftshift(scaled_red)
    #G
    im_shift_g=fftshift(im[:,:,1])
    #B
    im_shift_b=fftshift(scaled_blue)

    return im_shift_r,im_shift_g,im_shift_b

im_v_re_r,im_v_re_g,im_v_re_b=rescale(im_v)
im_l_re_r,im_l_re_g,im_l_re_b=rescale(im_l)
im_l_re_r_fl,im_l_re_g_fl,im_l_re_b_fl=rescale(im_l_fl)
im_l_re_r_c,im_l_re_g_c,im_l_re_b_c=rescale(im_c)

rdp,gdp,bdp=1.78,2.56,3.5

def FFT_nB(im_shift_r,im_shift_g,im_shift_b):
    im_r_rand = np.abs(im_shift_r) ** 0.5 * np.exp(1j * np.angle(im_shift_r))
    im_g_rand = np.abs(im_shift_g) ** 0.5 * np.exp(1j * np.angle(im_shift_g))
    im_b_rand = np.abs(im_shift_b) ** 0.5 * np.exp(1j * np.angle(im_shift_b))
    # FFT of different channels
    current_field_r =fftshift(fft2(im_r_rand ))
    current_field_g =fftshift(fft2(im_g_rand))
    current_field_b =fftshift(fft2(im_b_rand))
    # Convert phase to grayscale.
    optimized_phase_r = np.angle(current_field_r)
    optimized_phase_g = np.angle(current_field_g)
    optimized_phase_b = np.angle(current_field_b)

    return optimized_phase_r,optimized_phase_g,optimized_phase_b

def convToGray_nB(phase_r,phase_g,phase_b):
    r_gray=(phase_r/np.pi+1)*(255/rdp)
    g_gray=(phase_g/np.pi+1)*(255/gdp)
    b_gray=(phase_b/np.pi+1)*(255/bdp)

    return r_gray,g_gray,b_gray

def FFT(im_shift_r,im_shift_g,im_shift_b):
    im_r_rand = np.abs(im_shift_r) ** 0.5 * np.exp(1j * np.angle(im_shift_r))
    im_g_rand = np.abs(im_shift_g) ** 0.5 * np.exp(1j * np.angle(im_shift_g))
    im_b_rand = np.abs(im_shift_b) ** 0.5 * np.exp(1j * np.angle(im_shift_b))
    # FFT of different channels
    current_field_r =fftshift(fft2(im_r_rand ))
    current_field_g =fftshift(fft2(im_g_rand))
    current_field_b =fftshift(fft2(im_b_rand))
    # Convert phase to grayscale.
    optimized_phase_r = np.angle(current_field_r)
    optimized_phase_g = np.angle(current_field_g)
    optimized_phase_b = np.angle(current_field_b)
    """Binarize phase to 0 and 1."""
    optimized_phase_r_bi = np.where(optimized_phase_r < 0, 0, 1)
    optimized_phase_g_bi = np.where(optimized_phase_g < 0, 0, 1)
    optimized_phase_b_bi = np.where(optimized_phase_b < 0, 0, 1)

    return optimized_phase_r_bi,optimized_phase_g_bi,optimized_phase_b_bi

v_phase_r,v_phase_g,v_phase_b=FFT(im_v_re_r,im_v_re_g,im_v_re_b)
h_phase_r,h_phase_g,h_phase_b=FFT_nB(im_l_re_r,im_l_re_g,im_l_re_b)
h_phase_r_fl,h_phase_g_fl,h_phase_b_fl=FFT(im_l_re_r_fl,im_l_re_g_fl,im_l_re_b_fl)
h_phase_r_c,h_phase_g_c,h_phase_b_c=FFT_nB(im_l_re_r_c,im_l_re_g_c,im_l_re_b_c)

def convToGray(phase_r,phase_g,phase_b,type_):
    if type_=="tri":
            gr,gg,gb=185,185,185
    else:
            gr,gg,gb=136,136,136
```

```
    r_gray=phase_r*gr
    g_gray=phase_g*gg
    b_gray=phase_b*gb

    return r_gray,g_gray,b_gray
"""Convolution between two binarized phases"""
def con_cor_Bi1(phase_r1,phase_r2,phase_g1,phase_g2,phase_b1,phase_b2,type_):
    if type_=="tri":
            gr,gg,gb=185,185,185
    else:
            gr,gg,gb=136,136,136
    con_r=np.exp(1j*phase_r1)*np.exp(1j*phase_r2)
    con_r=np.angle(con_r)+np.pi
    con_r=np.mod(con_r,2*np.pi)

    con_g=np.exp(1j*phase_g1)*np.exp(1j*phase_g2)
    con_g=np.angle(con_g)+np.pi
    con_g=np.mod(con_g,2*np.pi)

    con_b=np.exp(1j*phase_b1)*np.exp(1j*phase_b2)
    con_b=np.angle(con_b)+np.pi
    con_b=np.mod(con_b,2*np.pi)

    con_r_gray=con_r*gr
    con_g_gray=con_g*gg
    con_b_gray=con_b*gb

    return con_r_gray,con_g_gray,con_b_gray
"""Convolution between with and without binarized phases"""
def con_cor_Bi2(phase_r1,phase_r2,phase_g1,phase_g2,phase_b1,phase_b2,type_):
    if type_=="tri":
            gr,gg,gb=185,185,185
    else:
            gr,gg,gb=136,136,136
    phase_r2_gray=phase_r2*0.64*np.pi
    phase_g2_gray=phase_g2*0.64*np.pi
    phase_b2_gray=phase_b2*0.64*np.pi

    phase_r1_gray=phase_r1+np.pi
    phase_g1_gray=phase_g1+np.pi
    phase_b1_gray=phase_b1+np.pi

    con_r=np.exp(1j*phase_r1)*np.exp(1j*phase_r2)
    con_r=np.angle(con_r)+np.pi
    con_r=np.mod(con_r,2*np.pi)

    con_g=np.exp(1j*phase_g1)*np.exp(1j*phase_g2)
    con_g=np.angle(con_g)+np.pi
    con_g=np.mod(con_g,2*np.pi)

    con_b=np.exp(1j*phase_b1)*np.exp(1j*phase_b2)
    con_b=np.angle(con_b)+np.pi
    con_b=np.mod(con_b,2*np.pi)

    con_r_gray,con_g_gray,con_b_gray=convToGray_nB(con_r, con_g, con_b)

    return con_r_gray,con_g_gray,con_b_gray

t="tri"
con_r,con_g,con_b=con_cor_Bi1(v_phase_r, h_phase_r, v_phase_g, h_phase_g, v_phase_b, h_phase_b,t)
cor_r,cor_g,cor_b=con_cor_Bi1(v_phase_r, h_phase_r_fl, v_phase_g, h_phase_g_fl, v_phase_b,
h_phase_b_fl,t)
cor_r2,cor_g2,cor_b2=con_cor_Bi2(h_phase_r_c, h_phase_r_fl, h_phase_g_c, h_phase_g_fl, h_phase_b_c,
h_phase_b_fl,t)
"""Lens"""
lambda_r = 0.662e-6
lambda_g = 0.518e-6
lambda_b = 0.449e-6
arr_r=np.zeros((height, width))
arr_g=np.zeros((height, width))
arr_b=np.zeros((height, width))
```

```
pixel_size=4.5e-6
f=-2 # focal length (meters).
# Calculate the phase of lenses.
for i in range(height):
    for j in range(width):
        r = pixel_size * np.sqrt((i - center_h)**2 + (j - center_w)**2)
        arr_r[i, j] =  r**2 / (f * lambda_r)
        arr_g[i, j] =  r**2 / (f * lambda_g)
        arr_b[i, j] =  r**2 / (f * lambda_b)
# Change the range of the phase into [0,2].
arr_r_mod=np.mod(arr_r,2)
arr_g_mod=np.mod(arr_g,2)
arr_b_mod=np.mod(arr_b,2)
# Map the phase to grayscale by corresponding phase modulation.
arr_r_modified=arr_r_mod*(255/rdp)
arr_g_modified=arr_g_mod*(255/gdp)
arr_b_modified=arr_b_mod*(255/bdp)

def crop(im_modify,name):
    y_offset=center_h-1080//2
    im_cropped=im_modify[y_offset:y_offset+1080,:]
    im_cropped = im_cropped.astype(np.uint8)
    im_modi = Image.fromarray(im_cropped)
    im_modi.save(f"{name}.png")

def saveim(p_r,p_g,p_b):
    im_modify_c = np.zeros((height, width, 3), dtype=np.float32)
    im_modify_c[:,:,0] = p_r+arr_r_modified
    im_modify_c[:,:,1] = p_g+arr_g_modified
    im_modify_c[:,:,2] = p_b+arr_b_modified

    im_modify_c_r = np.zeros((height, width, 3), dtype=np.float32)
    im_modify_c_r[:,:,0] = p_r+arr_r_modified
    im_modify_c_g = np.zeros((height, width, 3), dtype=np.float32)
    im_modify_c_g[:,:,1] = p_g+arr_g_modified
    im_modify_c_b = np.zeros((height, width, 3), dtype=np.float32)
    im_modify_c_b[:,:,2] = p_b+arr_b_modified

    return im_modify_c,im_modify_c_r,im_modify_c_g,im_modify_c_b

saved_1=saveim(con_r,con_g,con_b)[0]
im_l_save=crop(saved_1,f"Bi_VH0 {t} con L")

saved_1=saveim(cor_r,cor_g,cor_b)[0]
im_l_save=crop(saved_1,f"Bi_VH180 {t} cor L")

saved_1=saveim(con_r,con_g,con_b)[0]
im_l_save=crop(saved_1,f"Bi_H00 {t} con L")

saved_1=saveim(cor_r,cor_g,cor_b)[0]
im_l_save=crop(saved_1,f"Bi_H0_180 {t} cor L")

phase_r,phase_g,phase_b=convToGray(v_phase_r, v_phase_g, v_phase_b, t)
saved_1,saved_2,saved_3,saved_4=saveim(phase_r,phase_g,phase_b)
im_l_save=crop(saved_1,f"Bi_VH0 {t} Sv L")

phase_r,phase_g,phase_b=convToGray(h_phase_r,h_phase_g,h_phase_b, t)
saved_1,saved_2,saved_3,saved_4=saveim(phase_r,phase_g,phase_b)
im_l_save=crop(saved_1,f"Bi_H0 {t} Sh L")

saved_1=saveim(cor_r2,cor_g2,cor_b2)[0]
im_l_save=crop(saved_1,f"Bi_H0_180 {t} cor L")

phase_r,phase_g,phase_b=convToGray_nB(h_phase_r_c,h_phase_g_c,h_phase_b_c, t)
saved_1,saved_2,saved_3,saved_4=saveim(phase_r,phase_g,phase_b)
im_l_save=crop(saved_1,f"Bi_H0 {t} Sh L")
```

10. Conclusions

In conclusion, this Doctoral Thesis contributes to the advance in the good use of phase-only liquid-crystal SLMs for displaying diffraction gratings and computer-generated holograms. The specific goals that have been achieved along the realization of the Thesis are the following:

- A detailed study has been performed on different phase profiles of phase-only diffraction gratings, namely triplicator gratings, which generate three equally intense 0th and ±1st diffraction orders. Different designs have been compared, including the binary phase profile, the continuous sinusoidal phase profile, and the phase profile proposed by Gori *et al*, which shows the optimal efficiency. A quantized version, with a different number of allowed phase steps that adapts to the discrete pixel structure of SLMs, has been compared as well.
- A phase-only LCOS-SLM has been used to experimentally verify their properties and analyse the difficulties in their implementation. Besides providing the first experimental verification of Gori's intensity curves versus parameter a, we have shown that the binary grating is a feasible option, with a theoretical efficiency of 86.4%, not so far from the optimum value of 92.3% given by Gori's profile.
- A detailed study is performed on how the pixelated structure and the limited fill factor of a

phase-only SLM affects the efficient display of diffraction gratings with short periods, in particular at the Nyquist limit (gratings with a period of 2 pixels). Analytical expressions for the diffraction orders' intensities are obtained in terms of the pixel size, fill factor, and phase difference between the two phase levels in the binary grating. It has been shown, theoretically and experimentally, how this brings parasite higher diffraction orders and a sinc envelope function that affects any displayed diffractive element. The specific case of the Nyquist diffraction grating has been studied. By definition, this is a binary grating, and the conditions to obtain a binary phase triplicator with maximum diffraction angle and maximum diffraction efficiency have been examined.

- We illustrated this Fourier analysis by implementing binary phase triplicator gratings of different periods. When displaying large periods (64 pixels), the SLM pixelated structure has very little effect on the result. However, when displaying short period gratings, both the SLM pixelation as well as the pixel crosstalk caused by fringing cannot be ignored. Consequently, the experimental gray values meeting the triplicator condition do not agree with the expected theoretical values. We introduced a simple model that fits the experimental intensity data by calculating the smoothed phase profile that is actually implemented by the SLM when the binary grating is addressed. This model considers unbalanced and asymmetric phase profiles governed by four parameters.
- A simple strategy to mitigate the fringing effect was demonstrated, consisting of increasing the values of the addressed gray levels.
- A random multiplexing approach has also been demonstrated, where different pixels of the SLM are selected to display the phase function of different diffractive elements. This was applied to implement a random triplicator, where the phase function selectively takes the phase values among two blazed gratings and a constant phase. This interesting approach, which benefits from the large number of pixels available in modern SLMs, eliminates the unwanted diffraction orders, although at the cost of reducing the diffraction efficiency.
- A multiwavelength SLM-based system has been built to display time-multiplexing colour diffractive elements. It consists of an RGB diode laser source (each laser switching at 60 Hz) that is synchronized with a phase-only SLM operating at a high rate (180 Hz). The wavelength dispersion characteristic of diffraction has been compensated in the hologram design to achieve the correct scaling of the optical reconstruction in the RGB components, thus enabling the production of a colour or white response.
- Well-known iterative Fourier transform algorithms (IFTA) have been programmed and applied

to the design of such dispersion compensated colour computer-generated holograms. As a result, the experimental successful realization of different colour holograms and diffractive elements with polychromatic and black-and-white diffraction patterns has been demonstrated.

- Finally, by applying the triplicator grating profile to computer-generated holograms we can achieve convolution/correlation operations in terms of RGB signals. In particular, the triplicator grating concept has been extended to generate colour phase-only holograms, where each of the RGB components is designed with the phase profile giving equal weight to the ± 1st and 0th harmonic orders, which in this context correspond respectively to the direct and complex conjugated inverted version of the target pattern, and a delta function. By adding a linear phase, the three terms have been demonstrated to be spatially separated and by combining with a second related hologram, the diffracted field has been shown to exhibit regions where the convolution and the autocorrelation are obtained.

Therefore, to summarize, the results of this study are interesting for the accurate implementations of phase triplicator gratings and computer-generated holograms in programmable SLMs. Given the growing interest in using such devices in many optical techniques in bioengineering, telecommunications, or industrial applications, the different aspects that affect the ideal SLM performance must be properly evaluated to apply the best practice for their compensation. The different studies within this Thesis provide useful tools for their characterization and evaluation.

10. Conclusiones

En conclusión, esta Tesis Doctoral contribuye al avance en el buen uso de las SLMs de cristal líquido puros de fase para la realización de redes de difracción y hologramas generados por ordenador. Los objetivos específicos que se han alcanzado a lo largo de la realización de la Tesis son los siguientes:

- Se ha realizado un estudio detallado de diferentes perfiles de fase de redes de difracción puras de fase, concretamente redes triplicadoras, que generan tres órdenes de difracción, 0 y ± 1, con la misma intensidad. Se han comparado diferentes diseños, incluyendo el perfil de fase binario, y perfiles de fase continuos como el perfil sinusoidal y el perfil propuesto por Gori *et al*, que muestra la eficiencia óptima. También se ha comparado una versión cuantizada, con diferente número de valores discretos de fase permitidos, y que se adapta a la estructura de píxeles discretos de los SLM.
- Se ha utilizado un LCOS-SLM puro de fase para verificar experimentalmente sus propiedades y analizar las dificultades de su implementación. Además de proporcionar la primera verificación experimental de las curvas de intensidad de Gori frente al parámetro numérico a, hemos demostrado que la red binaria es una opción factible, con una eficiencia teórica del

86,4%, no muy alejada del valor óptimo del 92,3% dado por el perfil de Gori.

- Se realiza un estudio detallado sobre cómo la estructura pixelada y el factor de llenado del SLM afectan a la implementación eficiente de redes de difracción con periodos cortos, de unos pocos píxeles, y en particular en el límite de Nyquist (redes con un periodo de 2 píxeles). Se obtienen expresiones analíticas para las intensidades de los órdenes de difracción en función del tamaño del píxel, el factor de llenado y la diferencia de fase entre los dos niveles de fase de la red binaria. Se ha demostrado, teórica y experimentalmente, cómo esto trae consigo la generación de órdenes de difracción parásitos, y una función envolvente tipo sinc que afecta a cualquier elemento difractivo. Se ha estudiado el caso concreto de la red de difracción de Nyquist, ya que proporciona el máximo ángulo de difracción. Por definición, se trata de una rejilla binaria, y se han examinado las condiciones para obtener un triplicador de fase binario y la máxima eficiencia de difracción.
- Ilustramos este análisis de Fourier implementando redes triplicadoras de fase binaria de diferentes periodos. Cuando se visualizan periodos grandes (64 píxeles), la estructura pixelada del SLM tiene muy poco efecto en el resultado. Sin embargo, cuando se visualizan redes de periodos cortos, no se puede ignorar tanto el pixelado de la SLM como el efecto *fringing*. En consecuencia, los valores de gris experimentales que cumplen la condición de triplicador no coinciden con los valores teóricos esperados. Introducimos un modelo sencillo que se ajusta a los datos experimentales de intensidad calculando el perfil de fase suavizado que realmente implementa el SLM cuando se implementa la red binaria. Este modelo considera perfiles de fase suavizados y asimétricos, gobernados por cuatro parámetros numéricos de ajuste.
- Se ha demostrado una estrategia sencilla para mitigar el efecto *fringing*, consistente en aumentar los valores de los niveles de gris aplicados al SLM.
- También se ha demostrado una técnica de multiplexación espacial aleatoria, en el que se seleccionan diferentes píxeles de la SLM para mostrar la función de fase de diferentes elementos difractivos. Esto se aplicó para implementar un triplicador aleatorio, en el que la función de fase toma selectivamente los valores de fase entre dos redes de fase lineal y otra fase constante. Este interesante enfoque, que se beneficia del gran número de píxeles disponibles en los SLM modernos, elimina los órdenes de difracción armónicos no deseados, aunque a costa de reducir la eficiencia de difracción.
- Se ha montado un sistema basado en SLM con multiplexación temporal y una fuente láser de tres longitudes de onda para visualizar elementos difractivos en color. La fuente láser consta de tres láseres de diodo RGB (cada láser conmuta a 60 Hz) que se sincroniza con un SLM puro de

fase que funciona a alta frecuencia (180 Hz). La dispersión de longitud de onda característica de la difracción se ha compensado en el diseño del holograma para lograr el escalado correcto de la reconstrucción óptica en los componentes RGB, permitiendo así la producción de una respuesta en color o en blanco.

- Se han programado y aplicado algoritmos clásicos de transformada iterativa de Fourier (IFTA) al diseño de tales hologramas en color generados por ordenador y compensados por dispersión. Como resultado, se ha demostrado la realización experimental con éxito de diferentes hologramas en color y elementos difractivos con patrones de difracción policromáticos y en blanco y negro.
- Por último, aplicando el perfil de la red triplicadora a los hologramas generados por ordenador podemos lograr operaciones de convolución/correlación en términos de señales RGB. En particular, el concepto de red triplicadora se ha ampliado para generar hologramas en color puros de fase, en los que cada una de las componentes RGB se diseña con el perfil de fase dando el mismo peso a los órdenes armónicos ± 1 y 0, que en este contexto corresponden respectivamente a la versión directa y la versión invertida y complejo-conjugada del patrón objetivo, y a una función delta. Añadiendo una fase lineal, se ha demostrado que los tres términos están separados espacialmente y, combinándolos con un segundo holograma relacionado, se ha demostrado que el campo difractado presenta regiones en las que se obtienen la convolución y la autocorrelación.

Por tanto, en resumen, los resultados de este estudio son interesantes para la implementación precisa de redes triplicadoras de fase y hologramas generados por ordenador en SLM programables. Dado el creciente interés en el uso de este tipo de dispositivos en numerosas técnicas ópticas en bioingeniería, telecomunicaciones, o aplicaciones industriales, los diferentes aspectos que afectan al rendimiento ideal de los SLMs deben ser adecuadamente evaluados para aplicar la mejor práctica para su compensación. Los diferentes estudios de esta Tesis proporcionan herramientas útiles para su caracterización y evaluación.

References

[Ama-1995] J. Amako, H. Miura and T. Sonehara, "Speckle-noise reduction on kinoform reconstruction using a phase-only spatial light modulator," *Applied Optics* **34**(17), 3165–3171 (1995). https://doi.org/10.1364/AO.34.003165

[Arf-1995] G. B. Arfken and H. J. Weber, *Mathematical Methods for Physicists*, 4th Edition, Academic Press, San Diego (1995).

[Arr-1999] V. Arrizón, E. Carreón and M. Testorf, "Implementation of Fourier array illuminators using pixelated SLM: efficiency limitations", *Optics Communications* **160**(4-6), 207-213 (1999). https://doi.org/10.1016/S0030-4018(98)00670-1

[Bor-2001] R. Borghi, F. Frezza, L. Pajewski, M. Santarsiero, and G. Schettini, "Optimization of a four-level triplicator using genetic algorithms", *Journal of Electromagnetic Waves and Applications* **15**(9), 1161-1174 (2001). https://doi.org/10.1163/156939301X01084

[Bou-2000] M. Bouvier and T. Scharf, "Analysis of nematic-liquid-crystal binary gratings with high spatial frequency", *Optical Engineering* **39**(8), 2129-2137 (2000). https://doi.org/10.1117/1.1304857

[Che-2017] J.-S. Chen, J. Jia and D. Chu, "Minimizing the effects of unmodulated light and uneven intensity profile on the holographic images reconstructed by pixelated spatial light modulators", *Chinese Optics Letters* 15(10), 100901 (2017). Link.

[Col-2019] M. Collini, F. Radaelli, L. Sironi, N. G. Ceffa, L. D'Alfonso, M. Bouzin and G. Chirico, "Adaptive optics microspectrometer for cross-correlation measurement of microfluidic flows", *Journal of Biomedical Optics* **24**(2), 025004 (2019). https://doi.org/10.1117/1.JBO.24.2.025004

[Dav-1994] J. A. Davis and D. M. Cottrell, "Random mask encoding of multiplexed phase-only and binary phase-only filters," *Optics Letters* **19**(7), 496–498 (1994). https://doi.org/10.1364/OL.19.000496

[Dav-2002] J. A. Davis, J. Adachi and D. M. Cottrell, "Diffraction efficiency of nonsynchronously sampled diffraction gratings", *Optical Engineering* **41**(11) 2983–2986 (2002). https://doi.org/10.1117/1.1512302

[Dav-2006] J. A. Davis, B. A. Slovick, C. S. Tuvey and D. M. Cottrell "High diffraction efficiency from one- and two-dimensional Nyquist frequency binary-phase gratings", *Applied Optics* **47**(15), 2829-2834 (2006). https://doi.org/10.1364/AO.47.002829

[Dav-2023] J. A. Davis. I. Moreno, S. Gao, M. M. Sánchez-López and D. M. Cottrell, "Multiplexing onto a spatial light modulator using random binary patterns", *Optical Engineering* **62**(10), 103104 (2023). https://doi.org/10.1117/1.OE.62.10.103104

[Fet-1997] F. Fetthauer and O. Bryngdahl, "Image quantization by a windowed iterative Fourier transform algorithm with local object dependent control", *Optics Communications* **139**(4-6), 187-192 (1997) https://doi.org/10.1016/S0030-4018(97)00129-6

[Gao-2023] S. Gao, M. M. Sánchez-López and I. Moreno, "Triplicator phase grating implementation in pixelated displays", *Proceedings of SPIE* vol. **12768**, *Holography, Diffractive Optics, and Applications XIII*, 127681G (2023). https://doi.org/10.1117/12.2687312 (INVITED)

[Gao-2024a] S. Gao, M. M. Sánchez-López and I. Moreno, "Experimental implementation of phase triplicator gratings in a spatial light modulator", *Chinese Optics Letters* **22**(2), 020501 (2024). https://doi.org/10.3788/COL202422.020501

[Gao-2024b] S. Gao, M. M. Sánchez-López and I. Moreno, "Feasibility study of liquid-crystal spatial light modulators for displaying triplicator gratings at their spatial resolution

limit", *Proceedings of SPIE* vol. **13016**, *Liquid Crystals Optics and Photonic Devices*, 130160O (2024). https://doi.org/10.1117/12.3017454

[Gao-2025] S. Gao, M. M. Sánchez-López and I. Moreno, "Analysis of diffraction gratings displayed in spatial light modulators at the Nyquist limit: Application to the triplicator grating", *Optics & Lasers in Engineering* **184**, 108628 (2025). https://doi.org/10.1016/j.optlaseng.2024.108628

[Ger-1972] R. W. Gerchberg and W. O. Saxton, "A practical algorithm for the determination of phase from image and diffraction plane pictures", *Optik* **35**(2), 237-246 (1972).

[Goo-2005] J. W. Goodman, *Introduction to Fourier Optics*, 3rd Edition, Roberts & Company Publishers, 2005.

[Gor-1998] F. Gori, M. Santarsiero, S. Vicalvi, R. Borghi, G. Cincotti, E. Di Fabrizio, and M. Gentili, "Analytical derivation of the optimum triplicator", *Optics Communications* **157**, 13-16 (1998). https://doi.org/10.1016/S0030-4018(98)00518-5

[Gör-2018] F. Görlitz, S. Guldbrand, T. H. Runcorn, R. T. Murray, A. L. Jaso-Tamame, H. G. Sinclair, E. Martinez-Perez, J. R. Taylor, M. A. A. Neil, C. Dunsby and P. M. W. French, "easySLM-STED: Stimulated emission depletion microscopywith aberration correction, extended field of view and multiple-beam scanning", *Journal of Biophotonics***11**, e201800087 (2018). https://doi.org/10.1002/jbio.201800087

[Lam-2024] E. S. Lamb, Z. Shi, T. Kremp, D. J. DiGiovanni and P. S. Westbrook, "Shape sensing endoscope fiber", *Optica* **11**(10), 1462-1467 (2024). https://doi.org/10.1364/OPTICA.532250

[Lee-2024] R. M. Lees, B. Pichler and A. M. Packer, "Contribution of optical resolution to the spatial precision of two-photon optogenetic photostimulation in vivo", *Neurophotonics* **11**(1), 015006 (2024). https://doi.org/10.1117/1.NPh.11.1.015006

[Les-1969] L. B. Lesem, P. M. Hirsch and J. A. Jordan Jr, "The kinoform: a new wavefront reconstruction device", IBM Journal of Research and Development **13**(2), 150-155 (1969). https://doi.org/10.1147/rd.132.0150

[Lin-2013] C. Lingel, T. Haist, and W. Osten, "Optimizing the diffraction efficiency of SLM-based holography with respect to the fringing field effect", *Applied Optics* **52**(28), 6877-6883 (2013). http://dx.doi.org/10.1364/AO.52.006877

[Loh-2008] A. Lohmann, "A pre-history of computer-generated holography", *Optics and Photonics*

News **19**(2), 36-41 (2008) https://doi.org/10.1364/OPN.19.2.000036

[Lu-1994] G. Lu, Z. Zhang, S. Wu, and F. T. S Yu, "Simple method for measuring phase modulation in liquid crystal televisions", *Optical Engineering* **33**(9), 3018-3022 (1994). https://doi.org/10.1117/12.177518

[Mai-1990] J. N. Mait, "Design of binary-phase and multiphase Fourier gratings for array generation", *Journal of the Optical Society of America A* **7**(8), 1514-1518 (1990). https://doi.org/10.1364/JOSAA.7.001514

[Már-2005] A. Márquez, C. Iemmi, I. Moreno, J. Campos, and M. J. Yzuel, "Anamorphic and spatial frequency dependent phase modulation on liquid crystal displays. Optimization of the modulation diffraction efficiency", *Optics Express* **13**(6), 2111-2119 (2005). https://doi.org/10.1364/OPEX.13.002111

[Mar-2007] A. Martínez, M. M. Sánchez-López and I. Moreno, "Phasor analysis of binary diffraction gratings with different fill factors", *European Journal of Physics* **28**(5), 805-816 (2007). https://doi.org/10.1088/0143-0807/28/5/004

[Mar-2019] D. Marco, M. M. Sánchez-López, A. Cofré, A. Vargas and I. Moreno, "Optimal triplicator design applied to a geometric phase vortex grating", *Optics Express* **27**(10), 14472-14486 (2019). https://doi.org/10.1364/OE.27.014472

[Mor-1995] I. Moreno, J. Campos, C. Gorecki and M. J. Yzuel, "Effects of amplitude and phase mismatching errors in the generation of a kinoform for pattern recognition", *Japanese Journal of Applied Physics Part 1* **34**(12A), 6423-6432 (1995). https://doi.org/10.1143/JJAP.34.6423

[Mor-2005] I Moreno, M. M. Sánchez-López, C. Ferreira, J. A. Davis and F. Mateos, "Teaching Fourier optics through ray matrices", *European Journal of Physics* **26**, 261-271 (2005). https://doi.org/10.1088/0143-0807/26/2/005

[Mor-2010] I. Moreno and C. Ferreira, "Fractional Fourier Transforms and Geometrical Optics", Chap. 3 in *Advances in Imaging and Electron Physics* **161**, 89-146 (2010). https://doi.org/10.1016/S1076-5670(10)61003-8

[Mor-2021] I. Moreno, M. M. Sánchez-López, J. A. Davis and D. M. Cottrell, "Simple method to evaluate the pixel crosstalk caused by fringing field effect in liquid-crystal spatial light modulators", *Journal of the European Optical Society - Rapid Publications* **17**, 1 (2021). https://doi.org/10.1186/s41476-021-00174-7

[Mos-2019] S. Moser, M. Ritsch-Marte and G. Thalhammer, "Model-based compensation of pixel crosstalk in liquid crystal spatial light modulators", *Optics Express* **27**(18), 25046-25063 (2019). https://doi.org/10.1364/OE.27.025046

[Muñ-2025] J. D. Muñoz-Bolaños, P. Rajaeipour, K. Kummer, M. Kress, Ç. Ataman, M. Ritsch-Marte and A. Jesacher, "Confocal Raman microscopy with adaptive optics", *ACS Photonics* **12**(1), 176-184 (2025). https://doi.org/10.1021/acsphotonics.4c01432

[Ngu-2022] T. L. Nguyen, S. Pradeep, R. L. Judson-Torres, J. Reed, M. A. Teitell and T. A. Zangle, "Quantitative phase imaging: recent advances and expanding potential in biomedicine", *ACSnano* **16**(8), 11516-11544 (2022). https://doi.org/10.1021/acsnano.1c11507

[Nik-2008] V. Nikolenko, B. O. Watson, R. Araya, A. Woodruff, D. S. Peterka and R. Yuste, "SLM microscopy: scanless two-photon imaging and photostimulation with spatial light modulators", *Frontiers in Neural Circuits* **2**, 5 (2008). https://doi.org/10.3389/neuro.04.005.2008

[Nobel-2014] The Nobel Prize in Chemistry for 2014 press release, The Royal Swedish Academy of Sciences, "Super-resolved fluorescence microscopy". Link

[Pes-2020] G. Pesce, P. H. Jones, O. M. Maragò and G. Volpe, "Optical tweezers: theory and practice", *European Physical Journal Plus* **135**, 949 (2020). https://doi.org/10.1140/epjp/s13360-020-00843-5

[Pho-2019] "Europe's age of light! – How photonics will power growth and innovation", *Photonics21 Multiannual Strategic Roadmap 2021-2027* (2019). Link

[Rom-2010] L. A. Romero and F. M. Dickey, "The mathematical theory of laser beam-splitting gratings", *Progress in Optics* **54**(10), 319-386 (2010). https://doi.org/10.1016/S0079-6638(10)05411-9

[Sal-1991] B. E. A. Saleh and M. C. Teich, Fundamentals of Photonics, John Wiley & Sons (1991).

[Sch-2020] S. Scholes, R. Kara, J. Pinnell, V. Rodríguez-Fajardo and A. Forbes, "Structured light with digital micromirror devices: a guide to best practice", *Optical Engineering* **59**(4), 041202 (2020). https://doi.org/10.1117/1.OE.59.4.041202

[Sha-1999] J. Shamir, Optical Systems and Processes, SPIE (1999).

[Soo-2024] S. R. Soomro, S. Sager, A. M. Paniagua-Diaz, P. M. Prieto and P. Artal, "Head-mounted adaptive optics visual simulator", *Optics Express* **15**(2), 608-623 (2024). https://doi.org/10.1364/BOE.506858

[Stat-2021] J. Starobrat, S. Fiderkiewicz, A. Kolodziejczyk, M. Sypek, R. Beck, K. Pavlov, M. Słowikowski, A. Kowalczyk, J. Suszek and M. Makowski, "Suppression of spurious image duplicates in Fourier holograms by pixel apodization of a spatial light modulator", *Optics Express* **29**(24), 40259-40273 (2021). https://doi.org/10.1364/OE.441489

[Stat-2023] J. Starobrat, F. Wlodarczyk, M. Makowski, J. Suszek, M. Sypek, and A. Kolodziejczyk, "Pixel-level phase filters for off-axis shifting of sinc envelope in holographic projection", *Optics Express* **31**(18), 29526-29605 (2023). https://doi.org/10.1364/OE.499256

[Tuc-2010] V. V. Tuchin, Edt.., *Handbook of Photonics for Biomedical Science*, CRC Press (2010).

[Urr-2024] D. F. Urrego, G. J. Machado and J. P. Torres, "Non-mechanical steering of the optical beam in spectral-domain optical coherence tomography", *Scientific Reports* **14**, 14860 (2024). https://doi.org/10.1038/s41598-024-65125-x

[Var-2013] A. Vargas, R. Donoso, M. Ramírez, J. Carrión, M. M. Sánchez-López and I. Moreno, "Liquid crystal retarder spectral retardance characterization based on a Cauchy dispersion relation and a voltage transfer function". *Optical Review* **20**, 378-384 (2013). https://doi.org/10.1007/s10043-013-0068-4

[Wal-1990] S. J. Walker, and J. Jahns, "Array generation with multilevel phase gratings", *Journal of the Optical Society of America A* **7**(8), 1509 -1513 (1990). https://doi.org/10.1364/JOSAA.7.001509

[Wen-2022] K. Wen, X. Fang, Y. Ma, M. Liu, S. An, J. J. Zheng, T. Kozacki and P. Gao, "Large-field structured illumination microscopy based on 2D grating and a spatial light modulator", Optics Letters 47(11), 2666-2669 (2022). https://doi.org/10.1364/OL.460292

[Wyr-1990] F. Wyrowski, "Diffractive optical elements: iterative calculation of quantized, blazed phase structures", *Journal of the Optical Society of America* **7**(6), 961-969 (1990). https://doi.org/10.1364/JOSAA.7.000961

[Yam-2024] Y. Yamaguchi, M. Miura, R. Higashida, K. Aoshima and K. Machida, "Viewing zone

enlargement method for holographic displays based on the slanted pixel arrangement on a spatial light modulator", *Applied Optics* **63**(9), 2204-2211 (2024). https://doi.org/10.1364/AO.506449

[Yan-2023] Y. Yang, A. Forbes and L. C. Cao, "A review of liquid crystal spatial light modulators: devices and applications", *Opto-Electronic Science* **2**(8), (2023). https://www.oejournal.org/article/doi/10.29026/oes.2023.230026

[Zha-2014] Z. Zhang, Z. You and D. Chu, "Fundamentals of phase-only liquid crystal on silicon (LCOS) devices", *Light: Science & Applications* **3**, e213 (2024). https://doi.org/10.1038/lsa.2014.94