UNIVERSIDAD NACIONAL DE COLOMBIA

Interferometría Holográfica Digital en Tiempo Real: Aplicación en la Cuantificación de Deformaciones Mecánicas

Natalia Múnera Ortiz

Universidad Nacional de Colombia
Facultad de Ciencias, Escuela de Física
Medellín, Colombia
2013

Interferometría Holográfica Digital en Tiempo Real: Aplicación en la Cuantificación de Deformaciones Mecánicas

Natalia Múnera Ortiz

Tesis presentada como requisito parcial para optar al título de:

Magister en Ciencias – Física

Director:

Dr. Jorge Iván García Sucerquia

Línea de Investigación:

Procesamiento Opto-Digital

Grupo de Investigación:

Grupo de Óptica, Grupo de Instrumentación Científica e Industrial

Universidad Nacional de Colombia

Facultad de Ciencias, Escuela de Física

Medellín, Colombia

2013

A mis padres Luis Javier Múnera y Luz Marina Ortiz. A mis hermanas Alejandra y Marcela. A Daniel E. López. A todas las personas que me apoyaron durante todo este tiempo para lograr cosas como esta.

Agradecimientos

Agradezco sinceramente al profesor Dr. Jorge Iván García Sucerquia, quien por sus acertadas asesorías, disponibilidad, acompañamiento y paciencia ha aportado enormemente en mi formación académica a través de la investigación.

Agradezco a Colciencias (Departamento Administrativo de Ciencia, Tecnología e Innovación) y a la Universidad Nacional de Colombia por el apoyo económico brindado bajo el programa de Jóvenes Investigadores e Innovadores 2011-2012.

Especial agradecimiento al profesor Dr. rer. nat. Román Castañeda, por su interés y sugerencias. Al profesor Ph.D Pedro Torres director del grupo de Fotónica y Opto-electrónica, por sus comentarios y sugerencias en el desarrollo del proyecto. De manera general, agradezco el apoyo e interés de todos los compañeros de trabajo, especialmente a Carlos A. Trujillo.

Por último agradezco a Daniel E. López por su apoyo incondicional. A mi familia por su constancia y apoyo.

Resumen

En esta tesis de Maestría se presenta la implementación de la técnica de interferometría holográfica digital (IHD) para la medición de micro-deformación con un algoritmo rápido que obtiene micro-deformaciones a tasas de video. La IHD consiste en el registro y reconstrucción de dos hologramas digitales del objeto en estudio para dos diferentes estados del mismo. Las diferencias de fase que cuantifican los cambios del objeto se obtienen por sustracción directa de los mapas de fase obtenida de la holografía digital. El registro de los hologramas, sus reconstrucciones, así como el cálculo de los mapas de fase, se llevan a cabo en el entorno GPGPU. El método propuesto puede procesar hologramas registrados de 1024x1024 pixeles en 48 milisegundos. Con el mejor rendimiento del método, puede procesar 21 hologramas por segundo. Esta implementación supera 133 veces el mejor rendimiento de la misma implementación en una CPU usando computación de multi-procesador.

Palabras clave: Interferometría holográfica digital, computación paralela, interferometría a tasa de video.

Abstract

This Master thesis presents the implementation of digital holographic interferometry technique (DHI) for micro-deformation measurement with a fast algorithm that obtains video rates micro-deformations. In DHI, two digital holograms of the object under study are recorded for the different states of it. The phase differences that quantify the changes of the object are obtained by direct subtraction of the phase maps obtained from the digital holograms. The recording of the holograms, their reconstructions as well as the production of the phase maps, are carried out in the GPGPU environment. The proposed method can process recorded holograms of 1024x1024 pixels in 48 milliseconds. At the best performance of the method, it processes 21 frames per second. This benchmark surpasses 133-times the best performance of the method on a regular CPU.

Keywords: Digital holographic interferometry, parallel computing, video rate interferometry.

Contenido

Pág.

Resumen ..... **IX**

Abstract ..... **XI**

Contenido ..... **XIII**

Lista de figuras ..... **XV**

Lista de tablas ..... **XVIII**

Introducción ..... **XIX**

1. Interferometría Holográfica Digital **1**
1.1 Holografía Digital 1
1.1.1 Registro 5
1.1.2 Reconstrucción 10
1.2 Interferometría holográfica digital 13
1.3 Interferometría holográfica digital por suma de intensidades 17
1.3.1 Algoritmo de tres pasos para el cálculo de la fase del frente de onda 18
1.4 Interferometría holográfica digital por resta de fases 19

2. Desenvolvimiento de Fase **23**
2.1 Desenvolvimiento de fase por dos longitudes de onda 26
2.2 Control del tamaño de pixeles en la imagen reconstruida 28

3. Fundamentos Del Cálculo Numérico Acelerado Con Tarjetas Gráficas **31**
3.1. Diferencias entre GPU y CPU 33
3.2. Modelo de programación en CUDA™ 34
3.2.1 Jerarquía de memorias y transferencia de datos 37
3.2.2 Uso de Librerías 40
3.2.3 Interoperabilidad con librerías de gráficos 42
3.2.4 Manejo de errores 44

4. Interferometría Holográfica Digital (IHD) en Tiempo Real **47**
4.1. Diseño experimental 47
4.1.1 Cálculo del vector sensitividad y factor geométrico 49
4.2. Reconstrucción de hologramas en paralelo 49

4.3. Resultados 55
4.3.1 Medición de la fase de interferencia 59
4.3.2 Medición de deformaciones mecánicas sin envolvimiento de fase 59
4.3.3 Desenvolvimiento de fase y medición de deformaciones mecánicas 64
4.4. Análisis de desempeño 69

5. Conclusiones y Perspectivas 71
5.1 Conclusiones 71
5.2 Perspectivas 72

Referencias 75

Lista de figuras

Pág.

Figura 1-1: Montaje característico de la holografía digital. DHV, Divisor de Haz Variable; EH, Expansor de Haz; FE, Filtro Espacial; L_1, Lente Colimadora; E, Espejo; CDH, Cubo Divisor de Haz; CMOS, Cámara; PC, Computador Personal. 3
Figura 1-2: Geometría en el sistema de registro de un holograma digital fuera de eje [20]. 6
Figura 1-3: Esquema geométrico de la reducción del ángulo de formación de imágenes. 8
Figura 1-4: Diagrama conceptual de los planos en holografía digital. 10
Figura 1-5: Geometría de reconstrucción en holografía digital..... 10
Figura 1-6: Esquema geométrico para medición de deformaciones de superficies opacas. 14
Figura 1-7: Alternativas en el procedimiento de interferometría holográfica digital. 20
Figura 2-1: a. Fase envuelta para una superficie esférica simulada de $4\mu m$ de altura máxima. b. Fase desenvuelta por el método de Goldstein. 25
Figura 2-2: Simulación de mapas de fase para dos longitudes de onda: a. objeto parabólico simulado, b. Mapa de fase φ_{λ_1} usando $\lambda_1 = 594nm$, c. Mapa de fase φ_{λ_2} usando $\lambda_2 = 633nm$, d. Mapa de fase obtenida luego de la resta de las fases $\varphi_{\lambda_{12}} = \varphi_{\lambda_1} - \varphi_{\lambda_2}$, e. Mapa de fase libre de discontinuidades ($\varphi_{\lambda_{12}} + 2\pi$ si $\varphi_{\lambda_{12}} < 0$)..... 27
Figura 3-1: Diagrama de distribución del área de silicio para una CPU y una GPU. En color rojo se presenta el área ocupado por las ALUs (Unidades Aritmético-Lógicas), en color morado la unidad de control de los núcleos y en naranja se presentan la memoria dentro de cada procesador..... 33
Figura 3-2: Ilustración de la declaración e invocación de un *kernel* en CUDA™ para la suma de dos vectores. 35
Figura 3-3: Código de llamada de un *kernel* con bloques de *hilos* de dos dimensiones. 35
Figura 3-4: Código de llamada de un *kernel* con bloques de *hilos* de dos dimensiones para un conjunto grande de datos. 36
Figura 3-5: Hilos, bloques y malla en la GPU. 37
Figura 3-6: Jerarquía de memorias en la GPU 38
Figura 3-7: Programación heterogénea entre la GPU y la CPU. 39
Figura 3-8: Código para la trasferencia de datos en CUDA. 40
Figura 3-9: Uso de la librería cuFFT para un arreglo bidimensional de datos. 41
Figura 3-10: Código para la modificación dinámica de una malla 2D por medio de un *kernel* en OpenGL 43

Figura 4-1: Esquema de un montaje experimental de IHD. **E_1**: Espejo con base rotatoria, **DHV**: Divisor de haz variable, **FE**: Filtro espacial, **L_1**: Lente colimadora, **E_2**: Espejo, **EH**: Expansor de Haz, **L_2**: Lente divergente, **CDH**: Cubo Divisor de Haz, **CMOS**: Cámara (del inglés Complementary metal-oxide-semiconductor), **PC**: Computador Personal. 48
Figura 4-2: Esquema geométrico para el cálculo del vector sensitividad. 49
Figura 4-3: Esquema del algoritmo empleado para el sistema de IHD a velocidades de video. 51
Figura 4-4: Diagrama ULM del flujo de datos entre la memoria de la CPU, GPU y el buffer de salida del OpenGL. En cada paso, una función de la CPU o una función de la GPU (*kernel*) es representada. 53
Figura 4-5: Filtrado espacial del holograma: supresión del orden cero y de la imagen virtual en la reconstrucción numérica del holograma digital: a. holograma digital registrado directamente de la cámara CMOS. b. Reconstrucción numérica por medio de la ecuación (1-28). c. Transformada de Fourier del holograma y d. Reconstrucción numérica de la porción de la imagen real mostrada en el rectángulo amarillo del panel c. 56
Figura 4-6: Curva de Contraste Vs. Relación de Intensidad Objeto-Referencia. 57
Figura 4-7: Cálculo del mapa de fase: a. Mapa de fase resultado del estado no deformado del objeto, b. Mapa de fase para un estado deformado del objeto y c. Fase de interferencia. Los valores de fase están en módulo 2π. 58
Figura 4-8: Resultados de la fase de interferencia respecto a una fase de referencia para diferentes desplazamientos del actuador. 60
Figura 4-9: Micro-deformación mecánica en una membrana de aluminio. 61
Figura 4-10: Curva de la deformación en un punto de referencia de la membrana y su comparación con el avance del actuador en cada una de las imágenes mostradas en la Figura 4-9: El punto 1 corresponde a la Figura 4-9a, el punto 2 corresponde a la Figura 4-9b y así sucesivamente 64
Figura 4-11: Desenvolvimiento de fase por medio del método de dos longitudes de onda. El color rojo indica un valor de *6,28rad*, que el color azul indica *0,00rad*. El punto rojo con amarillo representa el punto de referencia para comparación con el avance del actuador cuyos valores se presentan en la Tabla 4-2. 65
Figura 4-12: Curva de la deformación en un punto de referencia de la membrana y su comparación con el avance del actuador en cada una de las imágenes mostradas en la Figura 4-11: El punto 1 corresponde a la Figura 4-11a, el punto 2 corresponde a la Figura 4-11b y así sucesivamente 68
Figura 4-13: Esquema del rendimiento del procesamiento en GPU respecto a la CPU para la reconstrucción de hologramas a velocidades de video 70

Lista de tablas

Pág.

Tabla 4-1: Resultados obtenidos de la deformación de una membrana de aluminio por la técnica de IHD. 63
Tabla 4-2: Resultados obtenidos de la deformación de una membrana de aluminio por la técnica de IHD y el método de desenvolvimiento de fase por dos longitudes de onda. ... 68

Introducción

El análisis de las micro-deformaciones es usado en ingeniería para obtener información cuantitativa acerca del comportamiento mecánico de un material específico. Esta información es interesante para caracterizar las propiedades mecánicas y monitorear la integridad estructural de los componentes de un sistema [1]. Esta cuantificación puede ser hecha por medio de técnicas destructivas o no destructivas, de contacto o de no contacto. Las técnicas de no contacto son ideales debido a que preservan la integridad del material o componente analizado. Existen numerosas técnicas de no contacto para analizar las micro-deformaciones. Entre las más versátiles están las técnicas que usan la luz como sonda de medición. Las técnicas ópticas pueden dividirse en dos grandes categorías: interferométricas y no interferométricas [2,3]. Aunque las dos aproximaciones pueden generar una representación 3D del objeto, tienen un rango de aplicación y una sensitividad específica [4,5]. Debido a que nuestro interés es implementar una técnica de no contacto con una sensitividad en el rango de 0,00 a 4,80 micrómetros, esta tesis de maestría se centrará en las técnicas interferométricas.

Generalmente, los patrones de interferencia fueron registrados en películas fotosensibles con resolución espacial del orden de 10000 líneas por milímetro. Esta forma de hacer interferometría holográfica impuso la necesidad del desarrollo de materiales fotosensibles de mayor resolución. El proceso de revelado era llevado a cabo por medio de procesos químicos, lo que desalentó la aplicación de este método a un campo de aplicación más amplio.

La posibilidad de registrar patrones de interferencia con cámaras digitales y la recuperación de la información por medio de métodos numéricos abre una nueva puerta al campo de la interferometría [6]. La holografía digital (HD), llamado así al proceso del registro digital y reconstrucción de hologramas por métodos numéricos, evita el proceso químico de revelado para analizar los hologramas o el patrón de interferencia. Este

enfoque dispuso el camino para el desarrollo de la interferometría holográfica digital (IHD) [7].

Debido a que la HD permite calcular el frente de onda complejo dispersado por un objeto; la fase, necesaria para el análisis de interferogramas, puede ser calculada a partir de un holograma individual. Este enfoque compensa la necesidad de al menos tres patrones de interferencia para calcular la fase como en el método tradicional de corrimiento de fase [4,5].

En el método de IHD, dos hologramas provenientes del objeto bajo diferentes estados de deformación son registrados. Una vez los hologramas son procesados, la reconstrucción de los hologramas digitales son comparados por medio de sus mapas de fase. La IHD ha sido aplicada en diferentes campos de investigación y la industria [8–12]. A pesar de que la IHD es ideal para muchas otras aplicaciones en estos campos, su aplicación ha sido limitada debido al tiempo de procesamiento de los hologramas el cual está en el orden de los segundos. Una manera para lograr una aplicación de IHD a tasas de video, es la aceleración de la reconstrucción de los hologramas digitales y su análisis por medio de hardware. La tendencia actual del uso de la programación de propósito general en las tarjetas aceleradoras de gráficos GPU (de sus siglas en inglés Graphic Processing Unit), llamada GPGPU, para acelerar el cálculo complejo con datos de puntos flotantes, puede ser usada para lograr que la IHD opere a tasas de video.

El uso de la GPGPU en las ciencias ópticas ya ha sido propuesto. Trujillo implementó la reconstrucción de hologramas en tiempo-real basado en la transformada de Fresnel [13]; en esta aplicación, los autores lograron una aceleración de hasta 20 veces, respecto a la implementación en una CPU (de sus siglas en inglés Central Processing Unit) tradicional. Algunas implementaciones interferométricas hacen uso de la computación GPGPU: Baron y Kloppenborg [14] implementaron la técnica de interferometría en infrarrojo en la cual los autores reconstruyen la imagen por medio de la ubicación local de máximos con la técnica de gradiente conjugado no lineal, alcanzando un rendimiento de 5 veces más que en una CPU convencional, para una tarjeta NVIDIA® de 8600M GT, y hasta 200 veces más en una GPU ATI Radeon 5850. Wang y sus colaboradores [15], implementaron la técnica de interferometría de luz blanca en la cual la altura de objeto de estudio es encontrada, para cada uno de los pixeles, por medio del enfoque de las

franjas de correlación en 0,875 segundos en GPU, comparado con 9,343 segundos que tarda la CPU para una imagen de 768x576 pixeles. Jiang [16], usó la técnica de interferometría de luz blanca basado en el método de multiplexación de longitudes de onda (en ingles Wavelenght Division Multiplexing, WDM) además de una compensación automática para la longitud de camino óptico por medio de un SLED IR; el uso de una tarjeta NVIDIA® GeForce GTX 280 con 240 núcleos de CUDA™ (de sus siglas en ingles *Compute Unified Device Architecture*) proporcionó un rendimiento 20 veces mayor al de una CPU DualCore.

En esta tesis de maestría se reporta una implementación de la técnica IHD además de una mejora en el tiempo de ejecución de la medición de la fase de interferencia y micro-deformaciones basados en GPGPU. La arquitectura CUDA™ [17], desarrollada por la compañía NVIDIA®, para el procesamiento en paralelo, permite el cálculo simultáneo de los datos del holograma adquiridos directamente de la cámara. Luego del registro de los hologramas, la reconstrucción numérica es hecha por medio de la transformada de Fresnel y el cálculo de la fase de interferencia es hecho respecto a un holograma de referencia previamente registrado en un estado no deformado. Las diferencias de fase para cada nuevo holograma registrado son desplegadas en $48ms$ para imágenes de 1024x1024 pixeles. El cálculo de las micro-deformaciones son calculadas y visualizadas en $48ms$ para deformaciones en el rango de $0{,}00\pm0{,}09nm$ a $316{,}00\pm0{,}09nm$ y de $96ms$ en el rango de $316{,}00\pm0{,}09nm$ a $4800{,}00\pm0{,}09nm$, rango en el cual el mapa de fase de interferencia presenta envolvimiento. La completa paralelización del método en la GPU y el registro de los hologramas directamente en CUDA™, es un orden de magnitud más rápida que la misma implementación desarrollada en una CPU tradicional usando capacidad de multi-procesador. Este rendimiento de las GPU se debe a que estos dispositivos usan más núcleos de procesamiento en comparación con los núcleos usados en las unidades de procesamiento central de los computadores CPU que tradicionalmente son usados en forma serial.

Esta tesis de maestría es la primera propuesta presentada para obtener fases de interferencia y micro-deformaciones basados en la IHD operando a tasas de video. El alcance de este proyecto obedece al fortalecimiento de una de las iniciativas del grupo investigación de Procesamiento Opto-digital en el cálculo acelerado para aplicaciones ópticas en tiempo-real.

1. Interferometría Holográfica Digital

La interferometría holográfica (IH) es una técnica óptica usada para mapear deformaciones de superficies rugosas y/o variaciones en el índice de refracción de un medio traslucido. La IH consiste en la interferencia de dos campos ópticos coherentes esparcidos por la superficie de un objeto en diferentes estados. El primer registro corresponde a un estado inicial del objeto en análisis y el segundo registro corresponde a un estado deformado del objeto. Una vez obtenidos los dos registros se procede a la reconstrucción y al análisis de la deformación a través de la obtención de las diferencias de fase calculadas por medio de diversas técnicas. Cuando la diferencia de fase es obtenida a través de las fases calculadas directamente de la holografía digital, la interferometría holográfica se acuña con el nombre de interferometría holográfica digital (IHD). En este capítulo se presentará en detalle el proceso de registro y la reconstrucción de los hologramas; luego se presentarán dos aproximaciones a la IHD exponiendo a su vez el análisis de los interferogramas para cuantificar la deformación producida en el objeto.

1.1 Holografía Digital

La holografía nace a través de la propuesta hecha por Gabor para generar imágenes sin el uso de lentes [18,19]. El principio básico es formar las imágenes en dos pasos: i) grabar la información del frente de onda proveniente del objeto, a través del registro de su patrón de interferencia con una onda de referencia, dicho patrón se denomina *holograma*; ii) recuperación del frente de onda mediante la difracción de la onda de referencia en el holograma; si el holograma es iluminado con una réplica exacta de la onda de referencia un observador verá una imagen virtual la cual es indistinguible del objeto original pues posee toda la información de perspectiva y profundidad de foco. En la holografía digital, el segundo paso es realizado numéricamente en un computador. Este proceso puede ser llevado a cabo por diferentes metodologías, por ejemplo, holografía de Fraunhofer, holografía de Fourier y holografía de Fresnel. En este caso nos

centraremos en la reconstrucción numérica por medio de la transformada discreta de Fresnel ya que brinda el mayor número de posibilidades experimentales. Este método se describirá en sección 1.1.2.

Si se considera una onda plana incidiendo en la superficie del objeto, el proceso de registro del holograma se puede detallar a partir de considerar que el objeto esparce la onda de la forma:

$$O(x,y) = o(x,y)e^{i\varphi_o(x,y)}, \tag{1-1}$$

donde $o(x,y)$ es la amplitud real de la onda y $\varphi_o(x,y)$ es la fase de la onda objeto. La ecuación (1-1) representa la amplitud compleja del objeto.

La amplitud compleja de la onda de referencia

$$R(x,y) = r(x,y)e^{i\varphi_R(x,y)}, \tag{1-2}$$

con $r(x,y)$ la amplitud real y $\varphi_R(x,y)$ la fase de la onda de referencia.

Ambas ondas interfieren en la superficie de registro y su intensidad está dada por

$$I(x,y) = \left|O(x,y) + R(x,y)\right|^2. \tag{1-3}$$

La amplitud de transmisión $h(x,y)$ del medio de registro digital es proporcional a $I(x,y)$

$$h(x,y) = \tau I(x,y), \tag{1-4}$$

donde τ es el tiempo de exposición en el medio de registro. La ecuación (1-4) también es conocida como función holograma.

La recuperación del frente de onda objeto mediante la difracción de la onda de referencia al incidir en el holograma registrado $h(x,y)$ estará dada por

$$R(x,y)h(x,y) = \beta\tau(r^2 + o^2)R(x,y) + \beta\tau r^2 O(x,y) + \beta\tau R^2(x,y)O^*(x,y) \tag{1-5}$$

donde el primer término de la derecha se denomina el orden cero de la difracción, el segundo término es la imagen virtual y el tercer término es la imagen real que produce la

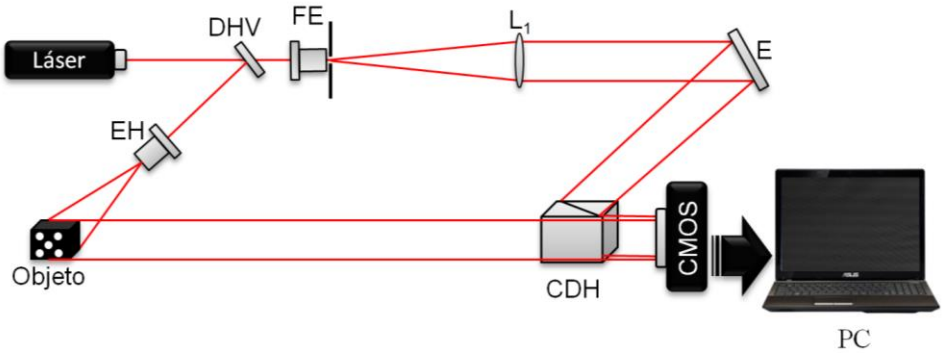
reconstrucción del holograma. Estos dos últimos términos son llamados también imágenes gemelas.

La holografía digital es entendida como el registro digital del holograma por medio de un arreglo CCD (del inglés Charge-Coupled Device) o CMOS (del inglés Complementary metal-oxide-semiconductor) y su posterior reconstrucción numérica por medio del computador.

Un montaje característico de la holografía digital se muestra en la Figura 1-1. Allí se ilustra un montaje óptico utilizado para registrar hologramas digitales. El divisor de haz variable (DHV) divide en dos el haz del láser, con intensidades relativas ajustables, lo cual permite optimizar el registro holográfico. Uno de los haces se dirige hacia el objeto ubicado a una distancia Z_0 de la cámara CCD o CMOS. El otro haz se colima para generar la onda plana de referencia.

El cubo divisor de haz (CDH) se utiliza para combinar la onda difractada por el objeto y la onda de referencia y dirigirlas hacia el sensor. El patrón resultante de la superposición es registrado por el sensor y almacenado en el sistema de procesamiento.

Figura 1-1: Montaje característico de la holografía digital. DHV, Divisor de Haz Variable; EH, Expansor de Haz; FE, Filtro Espacial; L_1, Lente Colimadora; E, Espejo; CDH, Cubo Divisor de Haz; CMOS, Cámara; PC, Computador Personal.



Este montaje permite intercambiar fácilmente los modos de registro holográfico en línea y fuera de línea [20], con sólo cambiar la alineación de los haces mediante el CDH o el espejo (E).

Una buena reconstrucción de las imágenes a partir del registro hecho con este montaje, o con cualquier otro, está íntimamente ligada al cumplimiento del teorema de muestreo [21,22] sobre toda la superficie del sensor CCD o CMOS, lo cual depende de los

parámetros del sensor y del montaje. Según el teorema del muestreo la frecuencia espacial máxima del patrón de interferencia entre la onda objeto y la de referencia registrado, $F\max$, debe satisfacer la condición

$$F\max \leq \frac{\mathrm{fs}}{2}, \tag{1-6}$$

donde fs es la frecuencia de muestreo, determinada por el inverso del tamaño del píxel del CCD o CMOS. Asumiendo un píxel cuadrado de lado ΔN , la ecuación (1-6) queda

$$F\max \leq \frac{1}{2\Delta N}. \tag{1-7}$$

El período de las franjas de interferencia de dos ondas planas con un ángulo θ entre sus direcciones de propagación está dado por

$$d = \frac{\lambda}{2\, sen\left(\frac{\theta}{2}\right)}, \tag{1-8}$$

así, la $F\max$ está dada por el inverso del período de las franjas de interferencia del patrón descrito por la ecuación (1-8), esto es

$$F\max = \frac{2sen\left(\frac{\delta\max}{2}\right)}{\lambda}. \tag{1-9}$$

Entonces, tomando la igualdad en la relación (1-7), se puede hallar el ángulo máximo permisible $\delta\max$ entre dos ondas que inciden sobre la cámara digital

$$\frac{1}{2\Delta N} = \frac{2sen\left(\frac{\delta\max}{2}\right)}{\lambda}$$
$$\delta\max = 2\mathrm{sen}^{-1}\left(\frac{\lambda}{4\Delta N}\right) \tag{1-10}$$

.

En aplicaciones prácticas, este ángulo es suficientemente pequeño para validar la aproximación

$$\delta \max = \frac{\lambda}{2\Delta N}. \tag{1-11}$$

Entonces, la distancia entre un objeto con un tamaño lateral determinado y la cámara deberá ser mayor que un valor mínimo $Zmin$, de manera que el ángulo bajo el cual las onditas esféricas provenientes de cada punto sobre el objeto interfieren con la onda plana de referencia en el plano de registro, no sobrepase el valor δmax. A continuación se detalla el registro de un sistema de holografía digital haciendo un análisis teórico de la distancia mínima de registro permisible en nuestro caso de interés de un arreglo fuera de eje [20].

1.1.1 Registro

Existen dos maneras de implementar un sistema de holografía digital: holografía digital en línea y holografía digital fuera de eje. Su principal diferencia radica en el ángulo del haz objeto y el haz de referencia. Debido a que la naturaleza del medio de registro obedece a un arreglo discreto de pixeles, es necesario establecer la distancia mínima a la cual puede formarse el holograma de tal manera que pueda resolverse sin pérdida de información en su posterior reconstrucción numérica.

A continuación se hará el análisis para holografía digital fuera de eje ya que permite diferenciar el orden cero de las imágenes gemelas (de acuerdo a la ecuación (1-5)), producto del método holográfico.

- **Distancia mínima de registro para un montaje fuera de eje**

Esta configuración permite la separación espacial de los términos mostrados en la ecuación (1-5) debido al ángulo de inclinación mínimo introducido en el montaje fuera de eje (ver Figura 1-2). Para la correcta separación de las componentes de las imágenes gemelas y las ondas que se propagan cercanas al eje óptico, debe cumplirse que este ángulo mínimo es [21]

$$\theta min = sen^{-1}\left(3\lambda\, fmax\right), \tag{1-12}$$

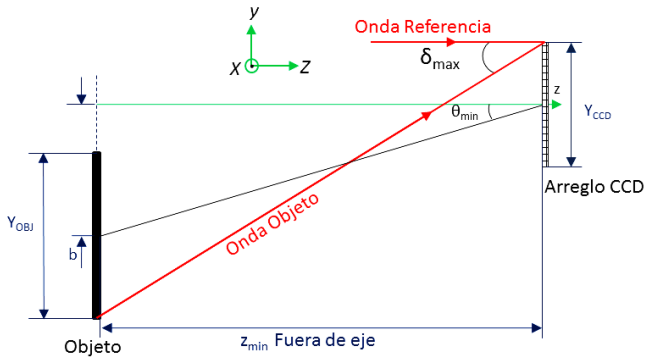
donde $fmax$ es la frecuencia espacial máxima del objeto.

El ancho de banda asociado a un objeto de tamaño $X_{OBJ} \times Y_{OBJ}$ está confinado dentro de un rectángulo de área $2\,f_X max \times 2\,f_Y max$, donde

$$2\,f_X max = \frac{X_{OBJ}}{\lambda Z} \quad . \qquad (1\text{-}13)$$

$$2\,f_Y max = \frac{Y_{OBJ}}{\lambda Z}$$

Figura 1-2: Geometría en el sistema de registro de un holograma digital fuera de eje [20].



Si el ángulo que forman el haz objeto con el haz de referencia (θmin) se introduce en la dirección ***y*** del plano coordenado entonces el ancho de banda de interés se restringe a esa sola dirección (ver Figura 1-2). Así, Tomando $2\,f_Y max$ como el ancho de banda de interés, el ángulo mínimo permitido, bajo la condición de ángulo pequeño es:

$$\theta min = \frac{3Y_{OBJ}}{2Z} \,. \qquad (1\text{-}14)$$

Para generar este ángulo, el objeto se ubica a una distancia b fuera del eje óptico del sistema, mientras que la onda de referencia incide perpendicularmente sobre el sensor CCD o CMOS, como se ilustra en la Figura 1-2.

El ángulo δmax para este montaje será

$$\delta\max = \frac{\frac{\mathrm{Y_{CCD}} + Y_{OBJ}}{2} + b}{\mathrm{Z_{min}\ Fuera\ de\ eje}}. \tag{1-15}$$

Tomando de la figura la distancia b como $b = Z_{min}\ Fuera\ de\ eje \times \theta\min$ y usando la relación (1-14), el ángulo δmax se puede presentar como:

$$\delta\max = \frac{\frac{\mathrm{Y_{CCD}} + Y_{OBJ}}{2} + \frac{3Y_{OBJ}}{2}}{\mathrm{Z_{min}\ Fuera\ de\ eje}}, \tag{1-16}$$

de lo que se puede concluir que la distancia mínima en el montaje fuera de eje es:

$$\mathrm{Z_{min}\ Fuera\ de\ eje} = \frac{\mathrm{Y_{CCD}} + 4Y_{OBJ}}{2\delta\max}. \tag{1-17}$$

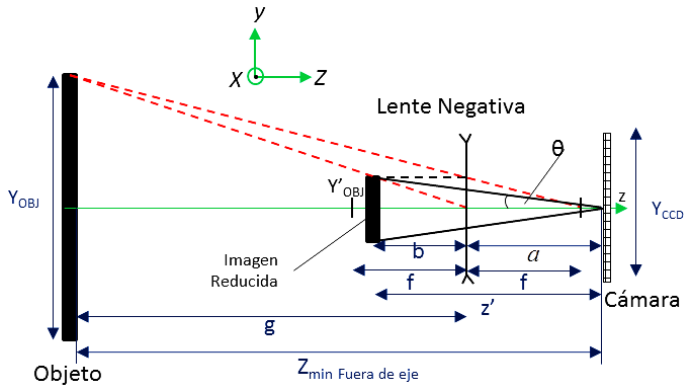
Es evidente que esta distancia mínima también es linealmente dependiente del tamaño del objeto, pero con una pendiente de $2/\delta\max$. Esta cantidad es cuatro veces mayor que la del montaje en línea [20].

Dentro de la restricción del teorema del muestreo, las relaciones entre el tamaño del objeto, su posición y los parámetros del CCD o CMOS deben satisfacer condiciones diferentes, en los dos tipos de configuración holográfica. El montaje en línea tiene requerimientos menos exigentes en cuanto a la resolución espacial del sensor CCD o CMOS [20,23,24], pero esta configuración tiene el problema de la superposición de las imágenes gemelas y del fondo coherente (orden cero de difracción). Este es un problema que tiene efectos devastadores al recuperar la información en las imágenes pues la información del objeto está mezclada con ruido producido por la superposición mencionada. Por otro lado, el montaje de holografía digital fuera de eje, si bien se pierde resolución espacial en el sensor, permite recuperar la información del objeto por medio de la aplicación de filtros espaciales [25,26]. Estos filtros espaciales, eliminan la imagen gemela y el orden cero aumentando el contraste de la imagen del objeto reconstruido. Esto será ilustrado con más detalle en el capítulo 3.

- **Reducción del ángulo de formación de la imagen**

En aplicaciones prácticas de metrología holográfica, usualmente se tienen objetos de interés con superficies grandes. Por ejemplo, un objeto con una superficie de interés de 30 *cm* de longitud lateral requiere una distancia de por lo menos 2,33 m entre el sensor CCD o CMOS y el objeto, para un sistema de registro con características como: longitud de onda $\lambda = 632{,}8nm$, número de pixeles 1024x1024 y un tamaño de pixel $\Delta x = 5{,}2\mu m$. Distancias mayores de $1m$ son difíciles de lograr en un laboratorio convencional, además de las posibles variaciones en el índice de refracción del aire o vibraciones durante el registro, hacen que la comparación entre interferogramas sea difícil. Una posible solución para tomar la onda esparcida por el objeto a distancias posibles en un laboratorio sin pérdida de campo visual es mediante la introducción de una lente [27,28] con el fin de aumentar el frente de onda que registra la cámara (ver Figura 1-3).

Figura 1-3: Esquema geométrico de la reducción del ángulo de formación de imágenes.



El campo de la onda que incide sobre el sensor CCD o CMOS luce como un objeto pequeño producto de la formación de una imagen virtual del objeto con magnificación menor. De acuerdo a la sección anterior, la ecuación (1-17) define la distancia mínima de formación de imagen sobre el sensor CCD o CMOS, esto a su vez define el ángulo θ que se muestra en la Figura 1-3.

Una lente cóncava o divergente, con distancia focal f negativa, puede usarse para la reducción del ángulo θ mostrado en la Figura 1-3. Con las características de la lente, es posible calcular la distancia a : distancia desde la CCD o CMOS a la cual se debe ubicar

la lente y la distancia g que indica la posición de la lente desde el objeto. Éste cálculo se basa en la fórmula de lentes:

$$\frac{1}{f}=\frac{1}{g}-\frac{1}{b}\ , \tag{1-18}$$

donde b es la distancia desde la lente hasta la imagen virtual. La magnificación transversal M_T del sistema estará dada entonces relacionando la longitud lateral de la imagen virtual Y'_{OBJ} y la longitud lateral del objeto Y_{OBJ} :

$$M_T=\frac{Y'_{OBJ}}{Y_{OBJ}}\,. \tag{1-19}$$

Con las relaciones $\tan\theta=Y'_{OBJ}/\big(2(a+b)\big)$, $Y'_{OBJ}=-Y_{OBJ}\,f/\big(g-f\big)$ y $b=gf/\big(f-g\big)$ podemos obtener una expresión para la distancia a :

$$a=\frac{-Y_{OBJ}\,f}{\big(g-f\big)2\tan\theta}+\frac{fg}{g-f}\,. \tag{1-20}$$

La distancia de reconstrucción de los hologramas registrados usando una lente divergente corresponde entonces a la distancia entre la CCD o CMOS y la imagen virtual del objeto:

$$z'=b+a\,. \tag{1-21}$$

El uso de una lente convergente también es posible. La ecuación (1-20) ahora cambia a:

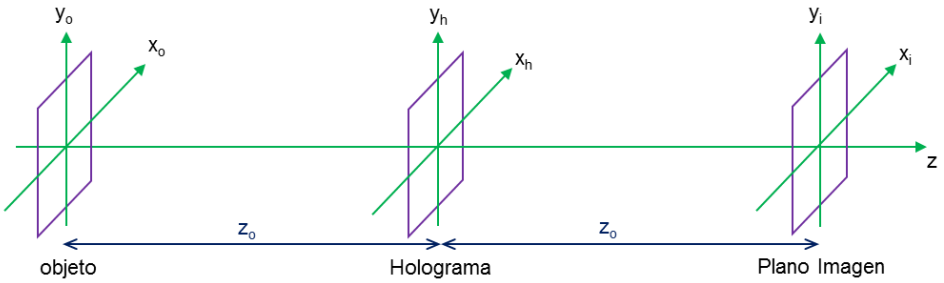
$$a=\frac{+Y_{OBJ}\,f}{\big(g-f\big)2\tan\theta}+\frac{fg}{g-f}\,. \tag{1-22}$$

Con el uso de lentes convergentes, la distancia $z'=b+a$ es menor respecto a la distancia de reconstrucción de las lentes divergentes. Esta implementación es útil si es necesaria la magnificación de un objeto pequeño, como es el caso de la microscopía holográfica, caso en el cual es recomendable usar una lente convergente para producir una imagen virtual magnificada [29].

1.1.2 Reconstrucción

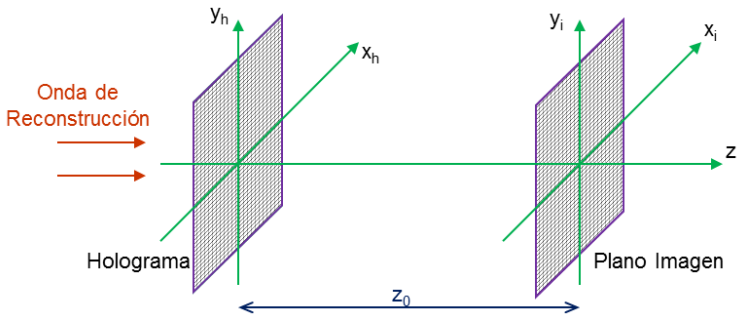
En el registro de hologramas de Fresnel, la onda objeto se propaga desde el plano del objeto (x_o,y_o), hasta el plano del holograma (x_h,y_h), ubicado a una distancia z_0 de éste como se muestra en la Figura 1-4. Dicha propagación deberá cumplir las condiciones experimentales [30] para que la onda objeto se difracte en el dominio de Fresnel.

Figura 1-4: Diagrama conceptual de los planos en holografía digital.



En el proceso de reconstrucción digital no existe frente de onda reconstruido sino que se calcula numéricamente la estructura de amplitud y fase que se obtendría en el plano imagen bajo la condición de onda plana de referencia la cual incide sobre el holograma registrado. Dicha estructura se representa en el sistema de visualización como una imagen, equivalente a la que proporcionaría el frente de onda reconstruido ópticamente, cuando se difracta la onda de referencia a través de una rejilla con transmitancia dada por el patrón de intensidades registrado en el holograma, $I(x_h, y_h)$ [6,31,32] (Figura 1-5).

Figura 1-5: Geometría de reconstrucción en holografía digital



La información de amplitud y fase proveniente del holograma registrado se obtiene a partir de la integral de Kirchhoff-Fresnel en la aproximación de Fraunhofer-Fresnel, la cual se escribe como [21,22,33]

$$E(x_i, y_i, z) = \frac{iE_0}{\lambda z} e^{-\frac{i2\pi z}{\lambda}} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x_h, y_h)\, e^{-\frac{i\pi}{\lambda z}\left[(x_h - x_i)^2 + (y_h - y_i)^2\right]} dx_h dy_h \quad , \tag{1-23}$$

donde $E(x_i, y_i, z)$ es el campo complejo de la imagen real reconstruida, E_0 representa la amplitud de la onda de referencia, λ la longitud de onda usada en el proceso de registro y los subíndices *h* e *i* especifican las coordenadas del plano holograma e imagen, respectivamente. Desarrollando los exponenciales del integrando, la ecuación , (1-23) toma la forma

$$E(x_i, y_i, z) = \frac{iE_0 e^{-\frac{i2\pi z}{\lambda}}}{\lambda z} e^{-\frac{i\pi}{\lambda z}\left(x_i^2 + y_i^2\right)} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left\{ I(x_h, y_h) e^{-\frac{i\pi}{\lambda z}\left(x_h^2 + y_h^2\right)} \right\} e^{\frac{i2\pi}{\lambda z}\left(x_h x_i + y_h y_i\right)} dx_h dy_h \tag{1-24}$$

De la ecuación anterior podemos observar que la información de amplitud y fase en el plano imagen es proporcional a la transformada de Fourier de una distribución de intensidades, modificada por un factor de fase cuadrático, conocido como la función *Chirp*. Esta función representa la fase de Fresnel en el plano de holograma [21,33] y por ese motivo, la ecuación (1-24) se ha denominado *Transformada de Fresnel* [21,29,33].

La ecuación (1-24) es una función continua aplicada a la distribución de intensidad continua. Como sabemos del registro digital por medio de una cámara CCD o CMOS, la distribución de intensidades se discretiza, esto hace que la Transformada de Fresnel (ecuación (1-24)) deba ser discretizada para su uso en la reconstrucción numérica de hologramas digitales.

- **Transformada Discreta de Fresnel**

Si tenemos un medio de registro con un arreglo rectangular de $N_x \times N_y$ muestras, este transforma el patrón continuo de intensidades a un arreglo discreto correspondiente al número de pixeles $N_x \times N_y$ y al tamaño de cada pixel $\Delta x_h \times \Delta y_h$. En consecuencia, el número de pixeles en el plano imagen estará determinado por el arreglo de registro

$(k\Delta x_h, l\Delta y_h)$ y se denotarán como un arreglo $(m\Delta x_i, n\Delta y_i)$, donde los enteros $k, m = 0,1,..., N_x - 1$ y $l, n = 0,1,..., N_y - 1$ determinan las direcciones del arreglo.

Teniendo en mente que debido a los efectos de la discretización causados por la naturaleza del sensor, las coordenadas continuas se transforman a coordenadas discretas de la siguiente manera: $(x_h, y_h) \to (k\Delta x_h, l\Delta y_h)$ y $(x_i, y_i) \to (m\Delta x_i, n\Delta y_i)$, la ecuación (1-24) se convierte en:

$$E(m,n,z) = \frac{iE_0}{\lambda z} e^{-\frac{i\pi}{\lambda z}\left[(m\Delta x_i)^2 + (n\Delta y_i)^2\right]} \sum_{k=0}^{N_x-1} \sum_{l=0}^{N_y-1} I(k,l)\, e^{-\frac{i\pi}{\lambda z}\left[(k\Delta x_h)^2 + (l\Delta y_h)^2\right]} e^{i\,2\pi\left(k\Delta x_h\, m\Delta x_i + l\Delta y_h\, n\Delta y_i\right)} \tag{1-25}$$

En la ecuación (1-25) se ha omitido el término exponencial $e^{-\frac{i2\pi z}{\lambda}}$ debido que afecta la reconstrucción en el plano imagen de manera uniforme.

Haciendo las siguientes relaciones:

$$\Delta x_i = \frac{\lambda z}{N_x \Delta x_h}, \tag{1-26}$$

$$\Delta y_i = \frac{\lambda z}{N_y \Delta y_h}, \tag{1-27}$$

la ecuación (1-25) puede ser escrita como

$$E(m,n,z) = \frac{iE_0}{\lambda z} e^{-i\pi\lambda z\left(\frac{m^2}{N_x^2\Delta x_h^2} + \frac{n^2}{N_y^2\Delta y_h^2}\right)} \sum_{k=0}^{N_x-1} \sum_{l=0}^{N_y-1} I(k,l)\, e^{-\frac{i\pi}{\lambda z}\left(k^2\Delta x_h^2 + l^2\Delta y_h^2\right)} e^{i\,2\pi\left(\frac{k\,m}{N_x} + \frac{l\,n}{N_y}\right)} \tag{1-28}$$

La ecuación (1-28) es denotada como la Transformada Discreta de Fresnel (TDF) [28,33,34] y corresponde a la amplitud compleja del campo óptico el cual permite la reconstrucción numérica tanto en intensidad como en fase del objeto de análisis. Esta es una ventaja de la holografía digital [6,35] respecto a la holografía óptica en la cual solo se obtiene la distribución de intensidad. En el caso digital, se puede tener acceso a la fase de módulo 2π la cual tiene una real ventaja en el caso de interferometría holográfica digital [29].

A partir de la TDF, es posible obtener la expresión para la distribución de intensidad

$$I(m,n,z) = |E(m,n,z)|^2 , \tag{1-29}$$

Y para el cálculo numérico de la distribución de fase tenemos la relación:

$$\varphi(m,n,z) = \arctan \frac{\mathrm{Im}(E(m,n,z))}{\mathrm{Re}(E(m,n,z))} . \tag{1-30}$$

Otra de las ventajas que presenta este método de reconstrucción es la rapidez en el cómputo ya que puede calcularse como una transformada de Fourier del holograma o distribución de intensidades, $I(k,l)$, multiplicado por el factor de fase $e^{-\frac{i\pi}{\lambda z}\left(k^2\Delta x_h^2 + l^2 \Delta y_h^2\right)}$. El factor que multiplica la doble sumatoria $e^{-i\pi\lambda z\left(\frac{m^2}{N_x^2\Delta x_h^2}+\frac{n^2}{N_y^2\Delta y_h^2}\right)}$ solo afecta a la fase total y puede ser omitida si solo es de interés la información de intensidad.

La transformada de Fourier se lleva a cabo por medio del algoritmo de transformada rápida de Fourier (FFT), con el fin de mejorar su efectividad computacional [33,36].

De acuerdo a lo anterior, la reconstrucción de campos ópticos en holografía digital no está limitada a la distancia de registro del holograma, sino que es posible determinar la información de fase e intensidad en cualquier plano antes o después del holograma (distancias z positivas o negativas). Esta posibilidad constituye otra de las características ventajosas de la reconstrucción numérica luego del registro digital del holograma.

1.2 Interferometría holográfica digital

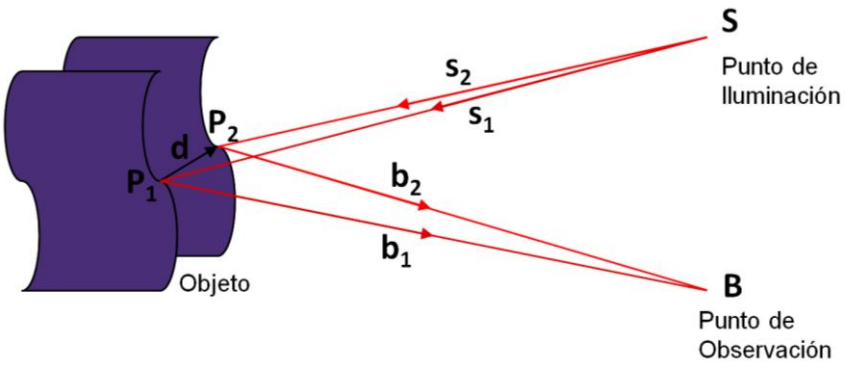
El uso de la holografía digital como instrumento de medida está fundamentado en su capacidad de reconstruir frentes de onda numéricamente. Esta característica permite comparar frentes de onda y establecer, tanto de manera cualitativa como cuantitativa, las diferencias y similitudes entre ellos. Esta idea, que es el fundamento de la interferometría en general, da lugar a la *interferometría holográfica digital* cuando al menos uno de los frentes de onda comparados es generado por métodos holográficos cuyo registro se hace digitalmente. El reconocimiento de las diferencias entre los frentes de onda se

establece a través del análisis de las franjas generadas por su interferencia o por sus diferencias de fase de los frentes de onda registrados.

La interferometría holográfica digital (IHD), registra en el sensor CCD o CMOS los frentes de onda luego de producir pequeñas deformaciones que sufre un cuerpo opaco o las variaciones en el índice de refracción de objetos traslucidos [22,29]. En nuestro caso de interés, objetos de superficies opacas para medición de deformaciones, el desplazamiento de cada punto en la superficie da lugar a diferencias en el camino óptico $\delta(P)$ para cualquier punto P. Esta es la diferencia de camino que recorre la luz desde la fuente **S**, un punto **P** del objeto y el punto **B** de observación antes y después de cambiar a un estado deformado como se puede ver en la Figura 1-6. La fase de interferencia $\Delta\varphi(P)$ está relacionada a la diferencia de camino óptico por medio de la expresión

$$\Delta\varphi(P)=\frac{2\pi}{\lambda}\delta(P). \tag{1-31}$$

Figura 1-6: Esquema geométrico para medición de deformaciones de superficies opacas.



Supongamos un vector de iluminación $S(x_s,y_s,z_s)$ y un vector de observación $B(x_B,y_B,z_B)$ dados en el sistema de coordenadas cartesianas. Cuando un objeto es deformado, el punto P de la superficie cambia de un punto $P_1(x_{P1},y_{P1},z_{P1})$ a una nueva posición $P_2(x_{P2},y_{P2},z_{P2})$. Así, podemos definir el vector desplazamiento como

$$\vec{d}(P)=(d_x(P),d_y(P),d_z(P))=P_2-P_1. \tag{1-32}$$

El punto P_1 y P_2 son diferentes microscópicamente, entonces la diferencia de camino óptico puede verse como

$$\delta(P) = \overline{SP}_1 + \overline{P_1B} - (\overline{SP}_2 + \overline{P_2B}) \qquad (1\text{-}33)$$
$$\delta(P) = \vec{s}_1 \cdot \overrightarrow{SP}_1 + \vec{b}_1 \cdot \overrightarrow{P_1B} - \vec{s}_2 \cdot \overrightarrow{SP}_2 - \vec{b}_2 \cdot \overrightarrow{P_2B}$$

donde $\overline{SP}_{1/2}$ y $\overline{P_{1/2}B}$ son del orden de metros mientras que $\left|\vec{d}(P)\right|$ está en el rango de micrómetros. Los vectores $\vec{s}_1$ y $\vec{s}_2$ pueden reemplazarse por un vector unitario puesto en la bisectriz del ángulo formado por $\vec{s}_1$ y $\vec{s}_2$.

$$\vec{s}_1 = \vec{s}_2 = \vec{s}\,. \qquad (1\text{-}34)$$

De igual forma, si reemplazamos los vectores $\vec{b}_1$ y $\vec{b}_2$ por un vector unitario $\vec{b}$

$$\vec{b}_1 = \vec{b}_2 = \vec{b} \qquad (1\text{-}35)$$

El vector desplazamiento estará dado por

$$\vec{d} = \overrightarrow{P_1B} - \overrightarrow{P_2B} \qquad (1\text{-}36)$$

Y por

$$\vec{d} = \overrightarrow{SP}_2 - \overrightarrow{SP}_1 \qquad (1\text{-}37)$$

Si reemplazamos la ecuación (1-36) y (1-37) en la ecuación (1-33) obtenemos

$$\delta(P) = \vec{d} \cdot \left(\vec{b} - \vec{s}\right). \qquad (1\text{-}38)$$

Así, reemplazando la ecuación (1-38) en la ecuación (1-31) podemos obtener entonces que la diferencia de fase o fase de interferencia obedece al término:

$$\Delta\varphi(P) = \vec{d} \cdot \frac{2\pi}{\lambda}\left(\vec{b} - \vec{s}\right) = \vec{d} \cdot \vec{S} \qquad (1\text{-}39)$$

donde se define el vector sensitividad como:

$$\vec{\mathbf{S}} = \frac{2\pi}{\lambda}\left(\vec{b} - \vec{s}\right). \tag{1-40}$$

El vector sensitividad sólo se define de acuerdo a la geometría del arreglo holográfico. Éste determina la dirección en la cual el montaje tiene mayor sensitividad. De acuerdo a la ecuación (1-39), en cada punto se está calculando la proyección del vector desplazamiento sobre el vector sensitividad. Para desplazamientos ortogonales al vector sensitividad la fase de interferencia siempre es cero independiente de la magnitud del desplazamiento.

El factor geométrico (GF) está relacionado con el vector sensitividad y se deriva de la ecuación (1-39):

$$\vec{d}(x,y,z) = \frac{\lambda}{2\pi} \cdot \frac{\Delta\varphi(x,y)}{\vec{b}(x,y,z) - \vec{s}(x,y,z)}. \tag{1-41}$$

De esta manera, el factor geométrico estará definido como:

$$GF = \frac{\lambda}{2\pi} \cdot \frac{1}{\vec{b}(x,y,z) - \vec{s}(x,y,z)}. \tag{1-42}$$

En general son necesarios tres interferogramas de la misma superficie con tres vectores de sensitividad diferentes para tener la información tridimensional de la deformación. Si solo la componente z del montaje es de interés entonces el montaje puede ser optimizado de manera tal que la dirección de iluminación sea paralela a la dirección de visualización, $\vec{b} = (0,0,1)$ y $\vec{s} = (0,0,-1)$; así el vector sensitividad estará dado por $\vec{\mathbf{S}} = 2\pi/\lambda\,(0,0,2)$. La componente z del vector desplazamiento estará dado por

$$d_z = \frac{\lambda}{4\pi}\Delta\varphi. \tag{1-43}$$

La ecuación (1-39) es la base de todas las medidas cuantitativas sobre superficies opacas por medio de la interferometría holográfica. La versatilidad de la holografía digital permite llevar a cabo los ensayos interferométricos de dos diferentes formas, como se muestra a continuación.

1.3 Interferometría holográfica digital por suma de intensidades

En la interferometría holográfica de doble exposición dos frentes de onda esparcidos por el mismo objeto, representando dos estados deformados diferentes, son registrados consecutivamente por medio del sensor CCD o CMOS. Estos dos registros son sumados simulando el método óptico que consiste en el registro de dos hologramas en la misma película fotográfica [37]. El nuevo holograma resultado de esta suma puede ser reconstruido con métodos numéricos como: transformada de Fresnel (descrito en la sección 1.1.2), el método de convolución o el método de espectro angular dependiendo de los parámetros del sistema holográfico [38,39].

Para la primera exposición se tiene una amplitud compleja para un punto P de la forma

$$E_1(P) = E_{01}(P)e^{i\varphi(P)}, \tag{1-44}$$

$E_{01}(P)$ es la amplitud compleja y $\varphi(P)$ es la distribución de fase. La variación del parámetro físico a ser medido, por ejemplo, variaciones en la forma de un objeto opaco o la variación en el índice de refracción para objetos traslúcidos, introduce variaciones en la distribución de fase del punto P de la forma $\Delta\varphi(P)$. Así, la amplitud compleja del segundo frente de onda registrado obedece a

$$E_2(P) = E_{02}(P)e^{i(\varphi(P)+\Delta\varphi(P))}, \tag{1-45}$$

siendo $E_{02}(P)$ la amplitud real y $\varphi(P)+\Delta\varphi(P)$ la nueva fase del estado deformado. Los dos frentes de onda son reconstruidos simultáneamente y dan una distribución de intensidades estacionaria de acuerdo a

$$\begin{aligned} I(P) &= \left|E_1(P)+E_2(P)\right|^2 \\ &= \left(E_{01}(P)e^{i\varphi(P)} + E_{02}(P)e^{i(\varphi(P)+\Delta\varphi(P))}\right)\left(E_{01}(P)e^{-i\varphi(P)} + E_{02}(P)e^{-i(\varphi(P)+\Delta\varphi(P))}\right) \\ &= I_1(P)+I_2(P)+\sqrt{I_1(P)I_2(P)}\left(e^{-i\Delta\varphi(P)}+e^{i\Delta\varphi(P)}\right) \\ &= I_1(P)+I_2(P)+2\sqrt{I_1(P)I_2(P)}\cos\left[\Delta\varphi(P)\right] \end{aligned} \tag{1-46}$$

Para amplitudes idénticas, $E_{01}(P) = E_{02}(P)$ obtenemos que

$$I(P) = 2I_1(P)\{1+\cos[\Delta\varphi(P)]\} \tag{1-47}$$

$\Delta\varphi(P)$ es llamada *diferencia de fase de interferencia* o de manera abreviada *fase de interferencia*. Si la variación espacial de la fase de interferencia es baja, la distribución de intensidad (1-46) representa la irradiancia del objeto modulada por un patrón de franjas cosenoidales. Los centros brillantes obedecen a múltiplos pares de π mientras que los centros oscuros corresponden a múltiplos impares de π.

Para la determinación de la fase de interferencia es necesaria la implementación de métodos de corrimientos de fase [4,40,41]. Este método necesita por lo menos tres imágenes de franjas de interferencia con diferentes corrimientos de fase externos conocidos. Una vez conocida la fase de interferencia se procede con el cálculo de la deformación inducida en la superficie de interés por medio de la ecuación (1-39). A continuación se hará una breve descripción del cálculo de la fase de interferencia por medio del método de corrimiento de fase.

1.3.1 Algoritmo de tres pasos para el cálculo de la fase del frente de onda

Si tenemos una distribución de intensidad de la forma

$$I_i(P) = I'_i(P) + I''_i(P)\cos[\varphi(P)+\gamma_i], \tag{1-48}$$

donde $I'_i(P)$ es la intensidad media, $I''_i(P)$ es la modulación de la amplitud, $\varphi(P)$ es la fase del frente de onda y γ_i es el corrimiento en la fase conocido. El subíndice i indica el número de registros, que en este caso será $i = 1,2,3.$ Si $\gamma_i = -\beta,0,+\beta$ tenemos

$$I_1(P) = I'_1(P) + I''_1(P)\cos[\varphi(P)-\beta] \tag{1-49}$$

$$I_2(P) = I'_2(P) + I''_2(P)\cos[\varphi(P)] \tag{1-50}$$

$$I_3(P) = I'_3(P) + I''_3(P)\cos[\varphi(P)+\beta] \tag{1-51}$$

Resolviendo el sistema de ecuaciones (1-49), (1-50) y (1-51) y despejando $\varphi(P)$ obtenemos

$$\varphi(P) = \arctan\left\{\left[\frac{1-\cos\beta}{\sin\beta}\right]\frac{I_1 - I_3}{2I_2 - I_1 - I_3}\right\} \tag{1-52}$$

Si $\beta = \pi/2$ entonces la ecuación (1-52) estará dada por

$$\varphi(P) = \arctan\left\{\frac{I_1 - I_3}{2I_2 - I_1 - I_3}\right\}. \tag{1-53}$$

Otros métodos para obtener la fase por medio del método de corrimiento de fase se detallan en las referencias [4,40,41].

1.4 Interferometría holográfica digital por resta de fases

La técnica de interferometría holográfica digital hace uso de las ventajas presentadas en la holografía digital como el cálculo directo de la fase del frente de onda esparcido por el objeto por medio de la ecuación (1-30). Esté método consiste en registrar el holograma de cada estado del objeto. Estos hologramas son reconstruidos independientemente por medio de la transformada de Fresnel (ecuación (1-28)).

Supongamos dos estados del objeto en el cual uno de ellos está deformado respecto al anterior. A partir de las reconstrucciones de las amplitudes complejas $E_1(m,n,z)$ y $E_2(m,n,z)$ puede obtenerse la distribución de fase de cada una de ellas a partir de la ecuación (1-30) de la siguiente manera

$$\varphi_1(m,n,z) = \arctan\frac{\mathrm{Im}(E_1(m,n,z))}{\mathrm{Re}(E_1(m,n,z))} \tag{1-54}$$

$$\varphi_2(m,n,z) = \arctan\frac{\mathrm{Im}(E_2(m,n,z))}{\mathrm{Re}(E_2(m,n,z))} \tag{1-55}$$

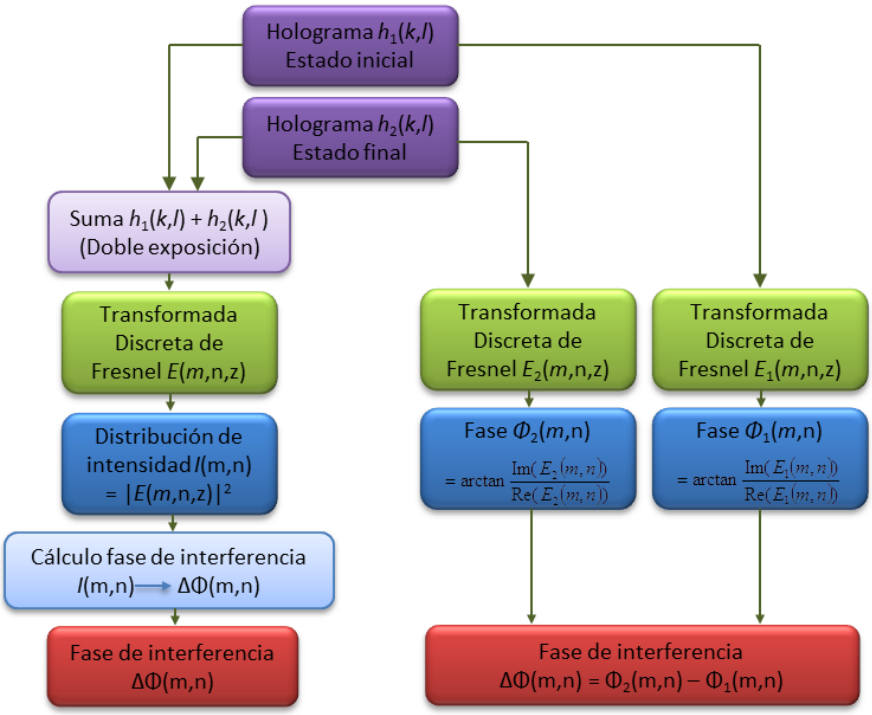
El subíndice 1 denota el primer estado (sin deformar) de la superficie de estudio y el subíndice 2 hace referencia al segundo estado (deformado). En (1-54) y (1-55) las fases toman valores entre $-\pi$ y π de acuerdo a los valores que toma la función $\arctan 2$. La fase de interferencia $\Delta\varphi$ es calculada por la resta directa de las ecuaciones (1-54) y (1-55) de la siguiente manera:

$$\Delta\varphi = \begin{cases} \varphi_1 - \varphi_2 & si\ \varphi_1 \geq \varphi_2 \\ \varphi_1 - \varphi_2 + 2\pi & si\ \varphi_1 < \varphi_2 \end{cases} \tag{1-56}$$

De esta manera no es necesaria la generación y el análisis del interferograma pues una vez se obtiene la fase de interferencia se procede con el cálculo del desplazamiento de la superficie de acuerdo con (1-43).

El esquema que se muestra en la Figura 1-7 resume las dos alternativas para realizar interferometría holográfica digital.

Figura 1-7: Alternativas en el procedimiento de interferometría holográfica digital.



De la figura anterior se puede observar que el método de interferometría holográfica digital por resta de fases es un método que permite tiempo de cómputo, además del esfuerzo mecánico en el montaje, mucho menor que el método de doble exposición, lo que nos acerca a una implementación de la técnica de IHD operando a frecuencias de video.

2. Desenvolvimiento de Fase

La transmisión o recepción de señales coherentes contiene la información espacial y temporal de fase y amplitud. Un ejemplo de procesamiento de señales coherentes es la holografía digital (HD). La HD ofrece ventajas tal como la habilidad de obtener la amplitud y fase del campo óptico y la versatilidad de técnicas de procesamiento que pueden aplicarse a datos de campo complejo [42].

En transmisión, cuando un objeto tiene mayor dimensión que la longitud de onda, el mapa de fase presenta discontinuidades de módulo 2π. Por esta razón, es necesario obtener un mapa de fase desenvuelto antes de obtener un perfil de deformación sin discontinuidades presentes. Existen numerosos algoritmos para detectar y remover discontinuidades de módulo 2π, entre algunos de ellos se encuentra el algoritmo de Goldstein [43], flynn [44], dependientes del mapa de calidad [45]; los cuales son robustos y contienen un costo computacional elevado además de presentar errores cuando el mapa de fase es ruidoso [46].

La fase es una propiedad relacionada con la parte espacial y/o temporal de la longitud de onda y es por esto que influencia la señal solo a través de sus valores principales, esto es, aquellos valores que están en el intervalo de ±π radianes [47]. Estos valores principales también son conocidos como valores de fase envuelta debido a que la fase absoluta es envuelta en el intervalo (-π,π].

En un sentido general, el desenvolvimiento de fase bidimensional es un problema imposible. Por ejemplo, una función de fase calculada φ deteriorada con ruido y envuelta en un intervalo (-π,π] es imposible recuperar sus valores de fase desenvuelta sin lugar a ambigüedades. Sin embargo, frente a ciertas suposiciones de las soluciones deseadas puede hacer el problema de desenvolvimiento un problema con solución. La suposición más común es que la fase desenvuelta deseada tiene diferencias locales que son menores a π radianes en magnitud en cualquiera de sus direcciones. También se asume

que la fase ϕ es de módulo 2π sobre una malla de puntos y la fase original obtenida mediante operaciones matemáticas es φ, entonces para un arreglo bidimensional tenemos que

$$\phi_{i,j} = \psi_{i,j} + 2\pi k, \quad k = entero \tag{2-1}$$

$$-\pi < \psi_{i,j} \leq \pi, \quad i = 0,...,M-1, \quad j = 0,...,N-1$$

Donde M y N son los tamaños totales de las filas y columnas del arreglo bidimensional, los subíndices i y j representan las direcciones en x y y respectivamente. La fase envuelta $\psi_{i,j}$ está dada y se desea determinar la fase desenvuelta $\phi_{i,j}$ en la misma ubicación en la malla. Se asume que las diferencias de fase de $\psi_{i,j}$ son menores que π radianes en magnitud en cualquier dirección con los pixeles cercanos [47]. En este caso $\psi_{i,j}$ incluye el efecto de ruido de la fase $\varphi_{i,j}$ calculada. Esta es la formulación del desenvolvimiento de fase bidimensional discreto. Generalmente se usa dos conjuntos de diferencias de fase para determinar el gradiente de la fase: aquellos respecto al subíndice i y respecto al subíndice j los cuales pueden ser calculados de las diferencias de fase envuelta $\psi_{i,j}$:

$$\Delta^x_{i,j} = W\{\psi_{i+1,j} - \psi_{i,j}\}, i = 0,...,M-2, \; j = 0,...,N-2,$$

$$\Delta^x_{i,j} = 0 \quad en\, otro\, caso \tag{2-2}$$

Y

$$\Delta^y_{i,j} = W\{\psi_{i,j+1} - \psi_{i,j}\}, i = 0,...,M-2, \; j = 0,...,N-2,$$

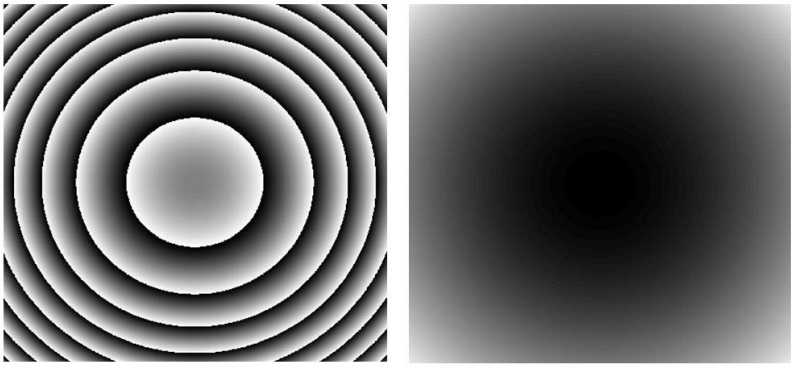
$$\Delta^y_{i,j} = 0 \quad en\ otro\ caso \tag{2-3}$$

Aquí se denota con los superíndices x y y las diferencias del envolvimiento de los valores correspondientes a los subíndices i y j respectivamente. El operador W define los valores de envolvimiento de su argumento en el rango (-π,π] por adición o sustracción de un múltiplo entero de 2π en su argumento que serán tenidos en cuenta en el momento de desenvolver la fase de acuerdo a la ecuación (2-1).

La ecuación (2-2) y (2-3) hace referencia al método de Itoh [48,49] en el cual se envuelve la fase envuelta $\varphi_{i,j}$ para evitar problemas con el ruido inherente a cantidades físicas reales. Una vez envuelta la fase y almacenada en la función $\psi_{i,j}$, puede usarse cualquier trayectoria donde se conectan los residuos resultantes marcados con sus diferentes polaridades. Dichos residuos corresponden a aquellos valores de fase que fueron envueltos y sus respectivos gradientes son hallados por medio de las ecuaciones (2-2) y (2-3). Luego se realiza un parche entre dos residuos de polaridades diferentes. Estos parches son evitados en la trayectoria del desenvolvimiento y solo son desenvueltos al final del proceso. El desenvolvimiento se determina por la correcta adición del múltiplo 2π para pasar de la función de fase envuelta $\psi_{i,j}$ a la fase desenvuelta $\phi_{i,j}$. Para más detalle de este método se recomienda al lector la referencia [45].

En la Figura 2-1a se ilustran los valores de fase envuelta mediante este procedimiento en una simulación de un frente de onda proveniente de una superficie esférica de altura máxima de $4\mu m$. En la Figura 2-1b se muestra el desenvolvimiento de fase para esta superficie por un método dependiente de trayectoria comúnmente empleado llamado el método de Goldstein [43].

Figura 2-1: a. Fase envuelta para una superficie esférica simulada de $4\mu m$ de altura máxima. b. Fase desenvuelta por el método de Goldstein.



a b

En la Figura 2-1a se ve claramente los saltos de fase de módulo 2π los cuales se observan como cambios en la intensidad de blanco a negro de acuerdo a la ecuación (2-1). La Figura 2-1b es desenvuelta de acuerdo al método de Itoh [48,49] desarrollado con Goldstein [43].

A continuación se presenta un método de desenvolvimiento de fase por medio del uso de dos longitudes de onda el cual puede remover discontinuidades en la fase dejando a un lado el alto costo computacional.

2.1 Desenvolvimiento de fase por dos longitudes de onda

El principio básico del desenvolvimiento de fase por medio del uso de dos longitudes de onda en el contexto de la holografía digital fue presentado por C. Wagner, S. Seebacher, W. Osten y W. Jüptner [50]. En este método, se incluyen dos mapas de fase registrados por medio de un sistema de holografía digital usando dos longitudes de onda independientemente.

Supongamos que se tiene una longitud de onda λ_1 con la cual se registra el holograma del objeto en un estado inicial, luego de la reconstrucción se obtiene la fase por medio de la ecuación (1-54). Un instante después, el objeto es deformado y el holograma registrado es reconstruido para calcular sus valores de fase (ecuación (1-55)). La diferencia de fase φ_{λ_1} es obtenida por medio de la ecuación (1-56). Este mismo proceso es llevado a cabo con la longitud de onda λ_2 con la cual será registrado el objeto en los mismos dos estados que los registrados con λ_1 y la diferencia de fase φ_{λ_2} es calculada. Una vez obtenidas las diferencias de fase φ_{λ_1} y φ_{λ_2}, se procede con la diferencia de estos dos mapas de fases como se muestra a continuación:

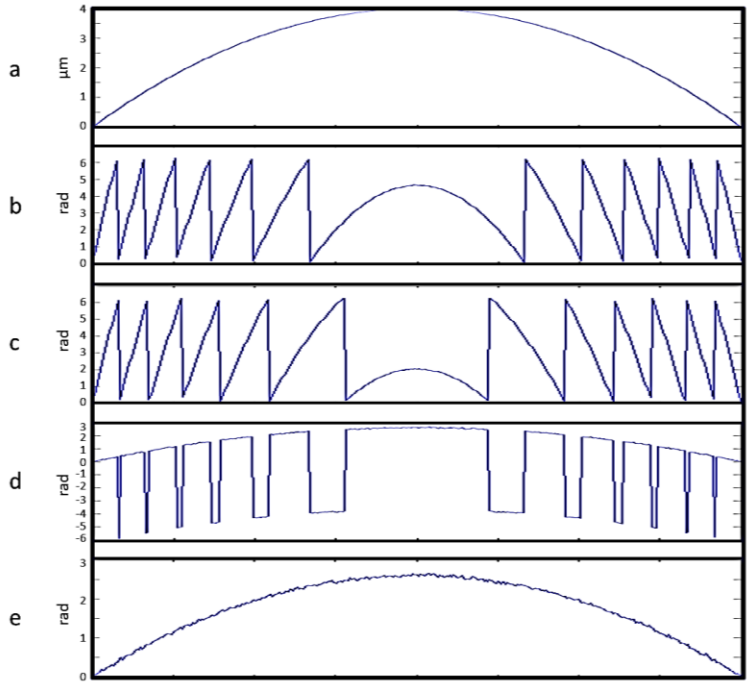
$$\Delta\varphi_{\Lambda} = \begin{cases} \varphi_{\lambda_1} - \varphi_{\lambda_2} & si\ \varphi_{\lambda_1} \geq \varphi_{\lambda_2} \\ \varphi_{\lambda_1} - \varphi_{\lambda_2} + 2\pi & si\ \varphi_{\lambda_1} < \varphi_{\lambda_2} \end{cases} \tag{2-4}$$

Este nuevo mapa de fase es equivalente a una distribución de fase de un holograma registrado con una longitud de onda Λ (ecuación (2-5)), llamada longitud de onda sintética o equivalente [51]. Un salto de fase corresponde entonces a un valor de $\Lambda/2$ para objetos opacos.

$$\Lambda = \frac{\lambda_1 \lambda_2}{|\lambda_1 - \lambda_2|} \tag{2-5}$$

En la Figura 2-2 se ilustra este proceso por medio de una simulación del perfil de un objeto esférico (como el mostrado en la Figura 2-1), cuya altura máxima es de $4\mu m$ (Figura 2-2.a). Las longitudes de onda implementadas fueron $\lambda_1 = 594nm$ y $\lambda_2 = 633nm$ lo que da como resultado una longitud de onda equivalente de $\Lambda = 9{,}641\mu m$. El cálculo del mapa de fase φ_{λ_1} y φ_{λ_2} de acuerdo a las ecuaciones (1-54) y (1-55) se muestra en la Figura 2-2.b y la Figura 2-2.c, respectivamente. El resultado de la diferencia de estos dos mapas de fase se muestra en la Figura 2-2.d y finalmente se obtiene el mapa de fase libre de discontinuidades por la adición de la cantidad 2π para los valores de fase negativos (Figura 2-2.e).

Figura 2-2: Simulación de mapas de fase para dos longitudes de onda: a. objeto parabólico simulado, b. Mapa de fase φ_{λ_1} usando $\lambda_1 = 594nm$, c. Mapa de fase φ_{λ_2} usando $\lambda_2 = 633nm$, d. Mapa de fase obtenida luego de la resta de las fases $\varphi_{\lambda_{12}} = \varphi_{\lambda_1} - \varphi_{\lambda_2}$, e. Mapa de fase libre de discontinuidades ($\varphi_{\lambda_{12}} + 2\pi$ si $\varphi_{\lambda_{12}} < 0$).



Como consecuencia del proceso de reconstrucción numérica de hologramas por medio de la transformada de fresnel (ecuación (1-28)) registrados con dos longitudes de onda

diferentes, el tamaño de pixel de la imagen reconstruida es diferente [52]. Este hecho impide la correcta visualización en la holografía digital a color donde debe haber una superposición precisa de los resultados de reconstrucción [53]. En nuestro caso de interés, la diferencia de tamaños de imágenes de reconstrucción impide la comparación de mapas de fase requeridas en la interferometría holográfica digital [54].

2.2 Control del tamaño de pixeles en la imagen reconstruida

Con el fin de evitar la diferencia de tamaños en la imagen reconstruida, es necesario usar métodos de re-escalamiento de imágenes al final del proceso de reconstrucción [53] o escalar el holograma registrado [54]. Otra manera de lograrlo, es por medio del uso del método de reconstrucción por convolución pues en este método los pixeles permanecen constantes e iguales al tamaño de pixel del sensor de registro a lo largo del proceso de reconstrucción; sin embargo, para largas distancias objeto-cámara, este método no funciona [51].

Ferraro y los demás [52] propusieron un método para controlar el tamaño de la imagen reconstruida por medio de la transformada de fresnel (ecuación (1-28)) en el cual dos imágenes del mismo objeto registrado a diferentes distancias con la misma longitud de onda se comparan correctamente. Este método también lo aplican al método de reconstrucción con diferentes longitudes de onda con el cual se obtiene una perfecta correspondencia.

El método de Ferraro consiste en mantener el tamaño de pixel de la imagen reconstruida igual con las dos longitudes de onda usadas. De acuerdo a la relación obtenida en la sección 1.1.2 entre el tamaño de pixel de la imagen reconstruida y el tamaño del pixel del holograma registrado, (ecuación (1-26) y (1-27)) tenemos que

$$\begin{aligned} \Delta x_{i\lambda_1} = \Delta x_{i\lambda_2} &\Rightarrow \frac{\lambda_1 z}{N_1 \Delta x_h} = \frac{\lambda_2 z}{N_2 \Delta x_h} \\ \Delta y_{i\lambda_1} = \Delta y_{i\lambda_2} &\Rightarrow \frac{\lambda_1 z}{N_1 \Delta y_h} = \frac{\lambda_2 z}{N_2 \Delta y_h} \end{aligned} \tag{2-6}$$

Siendo $\Delta x_{i\lambda_1}$ el tamaño de pixel de la imagen del objeto reconstruido en el sentido del eje x usando la longitud de onda λ_1, Δx_h el tamaño de pixel de la cámara de registro en el sentido del eje x, z la distancia de reconstrucción, N_1 el número de pixeles del holograma registrado con λ_1; el procedimiento para el eje y y el análisis para la longitud de onda λ_2 es equivalentemente.

Si igualamos los tamaños para el pixel imagen con la longitud de onda λ_1 y λ_2, de acuerdo a la ecuación (2-6) obtenemos la relación:

$$N_2 = \frac{N_1 \lambda_2}{\lambda_1} \tag{2-7}$$

La ecuación (2-7) indica que si $\lambda_1 < \lambda_2$ el número de pixeles del holograma registrado con λ_2 debe ser un poco mayor que el holograma registrado con λ_1.

En la sección 4.3.3 serán mostrados los resultados obtenidos en esta tesis de maestría con la aplicación de medición de deformaciones mecánicas usando el método de desenvolvimiento de fase por el uso de dos longitudes de onda.

En el siguiente capítulo se darán los fundamentos del cálculo numérico acelerado con las tarjetas gráficas por medio de la arquitectura CUDA™, la cual hace posible la implementación de los métodos anteriormente descritos.

3. Fundamentos Del Cálculo Numérico Acelerado Con Tarjetas Gráficas

La unidad de procesamiento gráfico (GPU) se ha convertido en una parte integral de los sistemas informáticos convencionales actuales. En la última década, ha habido un marcado aumento en el rendimiento y las capacidades de las GPU, permitiendo que la GPU moderna no sólo sea un potente motor de gráficos, sino también un procesador programable en una estructura altamente paralelizable y ancho de banda de memoria que supera sustancialmente su contraparte CPU (Unidad Central de Procesamiento) [55].

El rápido aumento de la capacidad de procesamiento y el mejoramiento de los ambientes de desarrollo de la GPU, han generado una comunidad de investigadores que se soportan en las GPU para enfrentar una amplia gama de exigentes problemas computacionalmente complejos. Este esfuerzo de la computación en la GPU, también conocido como computación GPU, ha posicionado a este dispositivo como una alternativa convincente frente a los microprocesadores tradicionales en los sistemas informáticos de alto rendimiento.

El incremento del rendimiento de estas tarjetas de procesamiento se ve enmarcado en la actual demanda en entornos científicos, industriales, comerciales e incluso en hogares. Usualmente, la capacidad de cómputo se basa en el procesador o CPU y su capacidad puede entenderse como una cantidad proporcional al número de transistores por unidad de área en el circuito integrado. Es por esta razón que las diferentes compañías han venido incrementando el número de estos transistores con el fin de aumentar la capacidad de cómputo pero el número de éstos por unidad de área tiene limitaciones físicas que impiden su fabricación [56].

Como resultado, se han buscado diferentes estrategias para superar estos inconvenientes por medio de herramientas de cálculo en paralelo. Entre éstas, el uso de las tarjetas aceleradoras de gráficos que permiten este tipo de programación. Por

consiguiente, la GPU se ha convertido en un potente procesador programable, con las interfaces de programación de aplicaciones (API) y el hardware desarrollado en dirección al desarrollo de dispositivos programables.

La popularidad de estas tarjetas radica en su técnica de programación SIMD (Single Instruction Multiple Data) la cual utiliza la arquitectura *multithread* para hacer computación en paralelo. En otras palabras, la GPU procesa muchos elementos en paralelo usando el mismo programa. Cada elemento es independiente de otro y la GPU divide el cómputo en subconjuntos iguales que se procesan de manera simultánea en un hardware especializado. Un enfoque bastante nuevo para aumentar la potencia de procesamiento por medio de computación paralela es el que se ejecuta en las unidades de procesamiento gráfico ya que estas tarjetas están diseñadas para el procesamiento de imágenes y de señales en general, desligando a las CPU de tareas como el post-procesamiento, codificación y decodificación de vídeo, escalado de imagen, visión 3D y reconocimiento de patrones por medio de la asignación de los valores de píxeles a *hilos* de procesamiento concurrente [55]. El concepto de un *hilo* en CUDA™ es un concepto abstracto que indica la tarea que se va a ejecutar en uno de los núcleos de los multiprocesadores de la GPU.

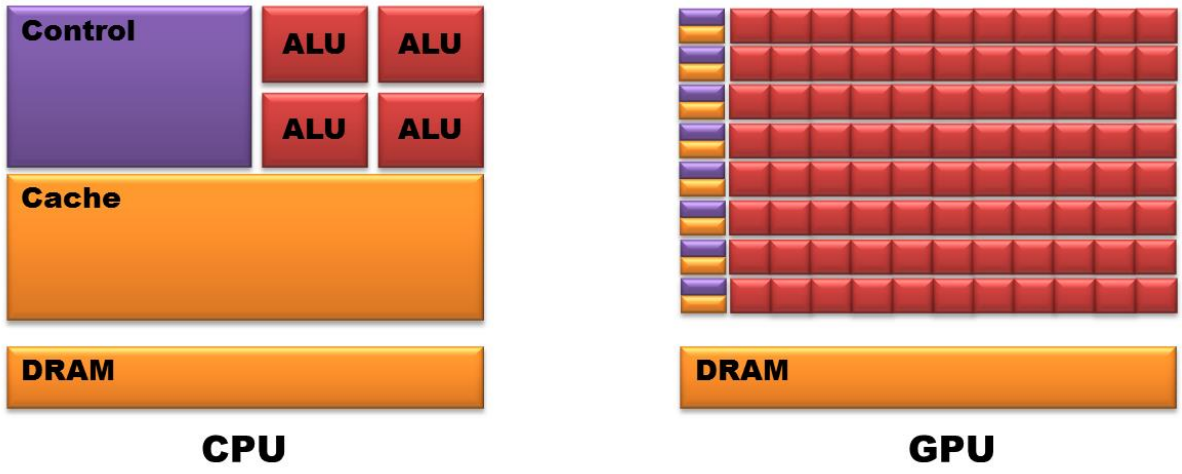
En la actualidad, se han implementado diversas aplicaciones en la investigación y la industria, por ejemplo: en el campo de química y biología computacional, análisis numérico, física, pronóstico meteorológico y climático, defensa e inteligencia, finanzas computacionales, fabricación: CAD (Computer Aided Design) y CAE (Computer Aided Engineering), medios de comunicación y entretenimiento, entre otros [57].

En este capítulo se presentan los fundamentos de la arquitectura de cómputo paralelo de propósito general hecha por NVIDIA® cuya herramienta para acceder a su arquitectura de programación CUDA™ (de las siglas en inglés *Computed Unified Device Architecture*) tiene un nuevo modelo de programación paralela y su propio conjunto de instrucciones para controlar el motor de cálculo paralelo de NVIDIA® [17]. Así mismo, se presentan las diferencias en hardware y software entre la CPU y la GPU, el uso de su memoria y de las funciones para ejecutar cálculos en paralelo denominadas *kernels*.

3.1. Diferencias entre GPU y CPU

Debido a su fabricación, la GPU tiene más transistores dedicados al procesamiento de datos o unidades ALU (Unidades Aritmético-Lógicas) que la CPU normales [17]. Este hecho implica que para grandes cantidades de operaciones de punto flotante, como la visualización de gráficos, la GPU gasta menos transistores en el control de flujo y en memoria caché que la CPU. Esta característica hace posible ejecutar algoritmos altamente paralelizados que manejan de manera muy eficiente miles de complejas operaciones de punto flotante. La Figura 3-1 ilustra la distribución del área de silicio en una CPU y una GPU.

Figura 3-1: Diagrama de distribución del área de silicio para una CPU y una GPU. En color rojo se presenta el área ocupado por las ALUs (Unidades Aritmético-Lógicas), en color morado la unidad de control de los núcleos y en naranja se presentan la memoria dentro de cada procesador.



Por su arquitectura, la CPU se compone de muy pocos núcleos y tiene mucha memoria caché que puede manejar pocos subprocesos de software a la vez. Por el contrario, una GPU se compone de cientos de núcleos que pueden manejar miles de *hilos* simultáneamente. La capacidad de una GPU con 100 núcleos para procesar miles de *hilos* puede acelerar algunos programas en proporción de 100x más que una CPU. Lo que es más, la GPU consigue aceleración mientras es más eficiente en potencia y costo que una CPU.

El uso prolongado de la memoria caché en la CPU se orienta a incrementar su rendimiento debido a las latencias bajas. La razón de la memoria compartida, es decir, la

memoria caché de GPU, en una tarjeta gráfica es aumentar el ancho de banda de memoria. En la CPU, entre más grande sea la memoria caché, más pequeñas serán las latencias. En la GPU, por el contrario, las latencias de acceso a memoria están ocultas por la ejecución simultánea de varios subprocesos.

Los circuitos integrados de NVIDIA® para los dispositivos aceleradores de gráficos o GPU, se construyen basados en multiprocesadores, los cuales tienen alrededor de diez núcleos, cientos de ALUs, varios miles de registros y decenas de kilobytes de memoria compartida. Además, la GPU tiene memoria global, a la cual pueden acceder todos los multiprocesadores, la memoria local de cada multiprocesador, y una zona de memoria especial para las constantes [17]. Esto se verá en detalle en la sección 3.2.1.

Las características anteriormente descritas del hardware de la GPU la hacen idónea para tareas como el procesamiento de gráficos pues estos dispositivos son capaces de recibir un grupo de datos, realizar todas las operaciones necesarias, y luego arrojar datos de salida de manera muy eficiente (en este caso, valores de píxeles). Estos datos se procesan de forma independiente y en simultáneo. El sistema de programación basado en GPU (GPGPU) va un poco más allá: Los datos son considerados como los hilos de procesamiento que pueden ser utilizados para mucho más que el renderizado de imágenes. Este último concepto es lo que permite llevar los dispositivos GPU a otros ambientes más allá del de procesamiento de gráficos.

3.2. Modelo de programación en CUDA™

A finales del año 2006, NVIDIA® creó una arquitectura de cómputo en paralelo de propósito general para acceder a las unidades de procesamiento de las GPU. Esta nueva arquitectura CUDA™ proporciona un nuevo modelo de programación que usa lenguajes estándares de programación conocido por los desarrolladores como C/C++, FORTRAN, OpenCL y Direct Compute [17], siendo de nuestro interés CUDA C. Éste extiende el lenguaje C por medio de su concepto fundamental: el *kernel.*

Un *kernel* es una función que se ejecuta *N* veces en paralelo invocando *N hilos* distintos. Se define usando la declaración **`__global__`** seguido del nombre que tendrá dicho *kernel*. Para cada llamado de un *kernel* se procede como al llamar una función en C pero

se agrega el número de subprocesos que ese *kernel* ejecutará en CUDA™ especificándolo entre los símbolos <<<...>>> y pueden ser de tipo **`int`** o **`dim3`**. Cada *hilo* que ejecuta un *kernel* se le asigna una única identificación *threadID* al cual se puede acceder mediante la variable **`threadIdx`**.

En la Figura 3-2 se muestra un ejemplo de la definición de un kernel para la suma de dos vectores **`A`** y **`B`** cuyo resultado se aloja en **`C`**. también se muestra la invocación de un *kernel*. El tamaño de cada uno de los vectores es *N*.

Figura 3-2: Ilustración de la declaración e invocación de un *kernel* en CUDA™ para la suma de dos vectores.

```
// Definición de un kernel
__global__ void VecAdd(float* A, float* B, float* C)
{
int i = threadIdx.x;
C[i] = A[i] + B[i];
}
int main()
{
...
// Invocación de un kernel con N hilos
VecAdd<<<1, N>>>(A, B, C);
}
```

En el ejemplo mostrado en la Figura 3-2 el vector **`VecAdd()`** se ejecuta *N* veces. El primer argumento entre llaves denota las dimensiones de los bloques de *hilos* de procesamiento. Si se establece en 1, una fila de *hilos* es llamada para ejecutar el *kernel*.

Si es el caso de un bloque de dos dimensiones, los *hilos* deben ser llamados de la forma como se ilustra en la Figura 3-3. Este ejemplo muestra la llamada de un *kernel* que representa la ejecución para el caso de una matriz, donde el subíndice **`i`** indica las filas y el subíndice **`j`** indica las columnas. En este caso, los bloques también pueden ser definidos en dos dimensiones. En la invocación del *kernel*, se define 1 bloque para ser ejecutados de acuerdo con la variable **`numBlocks`** y este bloque contiene ***`NxN`*** *hilos*.

Esta estructura es útil cuando se trabaja con matrices de datos de gran tamaño, ya que el tamaño de cada bloque es limitado: En una GPU típica sólo 512 *hilos* por bloque pueden ser definidos.

Figura 3-3: Código de llamada de un *kernel* con bloques de *hilos* de dos dimensiones.

```
// Kernel definition
```

```
__global__ void MatAdd(float A[N][N], float B[N][N],
float C[N][N])
{
int i = threadIdx.x;
int j = threadIdx.y;
C[i][j] = A[i][j] + B[i][j];
}
int main()
{
...
// Kernel invocation with one block of N * N * 1 threads
int numBlocks = 1;
dim3 threadsPerBlock(N, N);
MatAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
}
```

En el caso de tener un conjunto de datos mayor que 512, entonces es necesario usar más de un bloque para ejecutar la misma tarea en todo el conjunto. En la Figura 3-4 se muestra como se debe definir cada una de las filas y las columnas del arreglo con el fin de incluir todo el conjunto de datos.

Figura 3-4: Código de llamada de un *kernel* con bloques de *hilos* de dos dimensiones para un conjunto grande de datos.

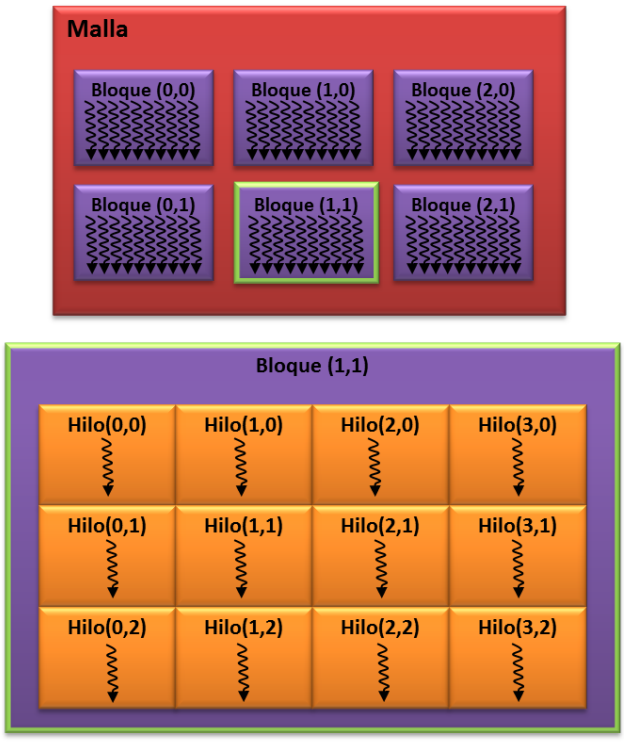
```
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N],
float C[N][N])
{
int i = blockIdx.x * blockDim.x + threadIdx.x;
int j = blockIdx.y * blockDim.y + threadIdx.y;
if (i < N && j < N)
C[i][j] = A[i][j] + B[i][j];
}
int main()
{
...
// Kernel invocation
dim3 threadsPerBlock(16, 16);
dim3 numBlocks(N / threadsPerBlock.x, N / threadsPerBlock.y);
MatAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
}
```

De la Figura 3-4 puede verse que se usó una malla de bloques de 16x16 (256) *hilos* cada uno. La malla es creada de tal forma que se usa un *hilo* por cada elemento de la matriz. Para este ejemplo se asume que el número de *hilos* por malla es divisible por el número de *hilos* de cada bloque.

La manera de acceder a cada uno de los *hilos* es mediante las instrucciones definidas en las filas **`i`** y las columnas **`j`** mostradas en la Figura 3-4: **`blockIdx.x:`** es la variable que

define el bloque al que se va a acceder, **`blockDim.x:`** define el tamaño del bloque o el número de *hilos* que contiene y por último **`threadIdx.x:`** es el identificador de cada uno de los *hilos* dentro del bloque. Esta manera de organizar los datos dentro de bloques y mallas se ilustra en la Figura 3-5.

Figura 3-5: Hilos, bloques y malla en la GPU.



En esta última figura se puede ver claramente el modelo de programación que usa CUDA™ para procesar un conjunto de datos. Los bloques están organizados en una malla unidimensional o bidimensional de bloques, que a su vez contiene *hilos* de tareas que serán ejecutados. El número de bloques de *hilos* en una cuadrícula es dictado por el tamaño de los datos que están siendo procesados y el número de procesadores en el sistema.

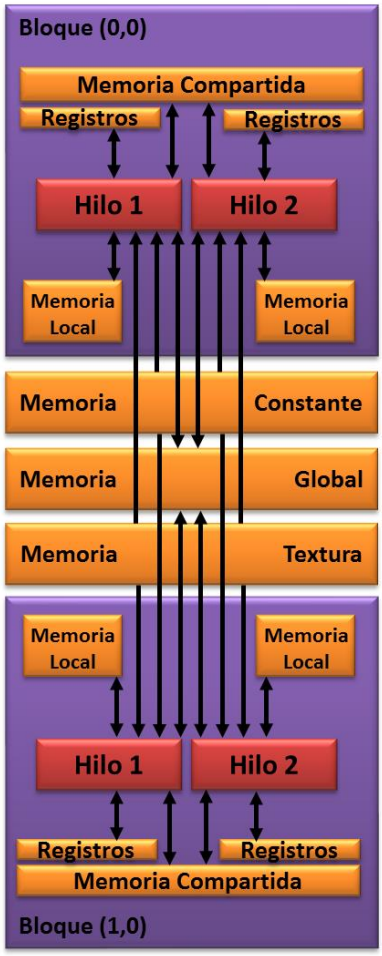
3.2.1 Jerarquía de memorias y transferencia de datos

Los *hilos* de CUDA pueden acceder a múltiples espacios de memoria durante su ejecución. Como puede observarse en la Figura 3-6 los *hilos* tienen una memoria local y pueden acceder a registros; además, cada uno de los bloques contiene una memoria

compartida visible y accesible a todos los *hilos* del mismo bloque. Esta memoria compartida tiene la misma duración del bloque. Adicionalmente, cada uno de los *hilos* puede acceder a la memoria global. También existen dos memorias de solo lectura que se denominan memoria de constantes y memoria de textura. Las memorias global, constante y de textura persisten en el *kernel* después de la ejecución de la misma aplicación y están optimizadas para ciertas aplicaciones. La memoria de textura también ofrece diferentes modos de direccionamiento por ejemplo, filtrado de datos para formatos específicos de datos [58].

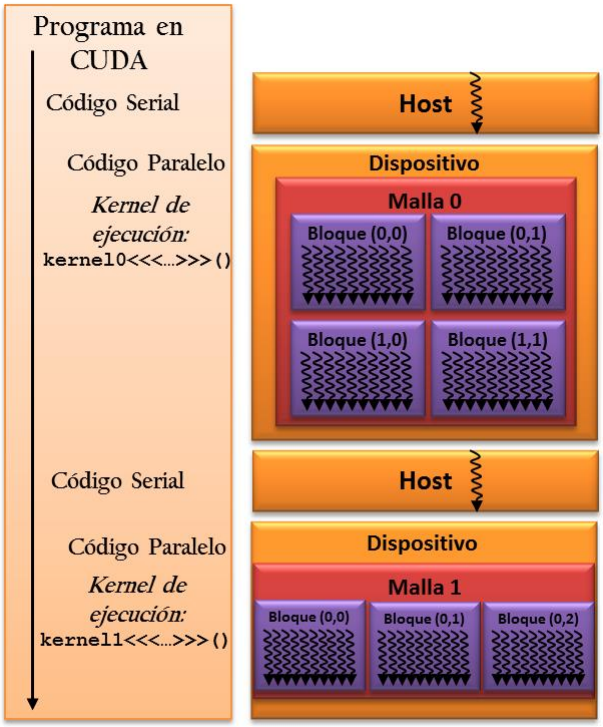
La memoria compartida es de más rápido acceso pues está ubicada directamente en la GPU y no hay necesidad de transferencia de datos mediante el bus de memoria como es el caso de las demás memorias.

Figura 3-6: Jerarquía de memorias en la GPU



Como se mencionó al principio de este capítulo, el modelo de programación de CUDA™ asume que los *hilos* se ejecutarán en un dispositivo físicamente operando como un co-procesador que corre en el *host* o procesador principal, en este caso, mediante el lenguaje C. En otras palabras, un *kernel* se ejecuta en la GPU dirigido por un programa de C ejecutado en la CPU (ver Figura 3-7).

Figura 3-7: Programación heterogénea entre la GPU y la CPU.



El modelo de programación CUDA™ también asume que tanto el *host* como el dispositivo mantienen sus propios espacios de memoria separados DRAM (de las siglas en inglés *Dinamic Random Access Memory*), estos son: la memoria principal y la memoria del dispositivo, respectivamente. Por lo tanto, un programa gestiona los espacios de memoria global, constante y textura visibles para los *kernels* a través de llamadas durante la ejecución de CUDA™. Esto incluye la asignación de memoria y des-asignación así como la transferencia de datos entre el *host* y la memoria del dispositivo. Antes de la ejecución de un *kernel*, las variables y los datos deben estar previamente definidos. Estos datos pueden ser definidos como cualquier tipo de dato en lenguaje C como **`int`**, **`char`**, **`float`**, **`float2`** o como variables propias de CUDA C como **`uchar`** o **`complex`**. Las funciones que la arquitectura CUDA™ ha incorporado para el manejo de

las memorias son: **`cudaMalloc()`**, una función que pone una variable en el espacio de memoria de la GPU, **`cudaFree()`**, una función que libera la memoria de la GPU de una variable y **`cudaMemcopy()`** una función que pasa los datos de una variable guardada en la memoria de la CPU a una variable en la memoria de la GPU o viceversa. En la Figura 3-8 se muestra un ejemplo de cómo usar estas funciones.

Figura 3-8: Código para la trasferencia de datos en CUDA.

```
// Códigodel Device
__global__ void VecAdd(float* A, float* B, float* C, int N) {
int i = blockDim.x * blockIdx.x + threadIdx.x;
if (i< N) C[i] = A[i] + B[i];
}

// Códigodel Host
int main() {
int N = ...;
size_t size = N * sizeof(float);

// Ubicación de las variables en el espacio de memoria del Host
float* h_A = (float*)malloc(size);
float* h_B = (float*)malloc(size);

// Ubicación de las variables en el espacio de memoria del Device
float* d_A;
cudaMalloc(&d_A, size);
float* d_B;
cudaMalloc(&d_B, size);
float* d_C;
cudaMalloc(&d_C, size);

// Trasferencia de datos desde el Host al Device
cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);

// Llamadadel Kernel
int threadsPerBlock = 256;
int blocksPerGrid = (N + threadsPerBlock - 1) / threadsPerBlock;

VecAdd<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N);

// Copia de los resultados del Device al Host
cudaMemcpy(h_C,d_C,size, cudaMemcpyDeviceToHost);

// Liberación de la memoria del Device
cudaFree(d_A);
cudaFree(d_B);
cudaFree(d_C);
}
```

3.2.2 Uso de Librerías

Además de las operaciones aritméticas sobre los datos de matrices realizados a través de *kernels*, existen librerías altamente utilizadas por la comunidad científica e ingenieril.

Dichas librerías hacen parte del toolkit de CUDA y son libres para el uso de la comunidad en general. Las librerías cuBLAS [59] y cuFFT [60], son las dos más famosas librerías de matemáticas de NVIDIA® aceleradas en GPU. cuBLAS es una implementación de BLAS (*Basic Linear Algebra Sub-programs*, por sus siglas en inglés). El modelo básico en el que la aplicación emplea la librería cuBLAS, es mediante la creación de objetos como matrices y vectores en el espacio de memoria de la GPU. Éstos se llenan con datos y son procesados con una secuencia de funciones cuBLAS, finalmente, se transfieren los resultados del espacio de memoria de la GPU de nuevo a la CPU. Para llevar a cabo esta tarea, cuBLAS proporciona funciones auxiliares para crear y destruir objetos en el espacio GPU, y para escribir y recuperar datos de estos objetos [17].

La librería cuFFT de NVIDIA® es uno de los algoritmos numéricos más importantes y ampliamente utilizado en física computacional y procesamiento de señales en general. La librería cuFFT proporciona una interfaz sencilla para el cálculo de FFT en paralelo en una GPU NVIDIA. Esta permite a los usuarios aprovechar rápidamente el poder de punto flotante y el paralelismo de la GPU en una librería altamente optimizada y probada.

La cuFFT sigue el modelo de implementación del más eficiente algoritmo de cálculo de la trasformada de Fourier, la FFTW [61]. Los parámetros utilizados son similares a los utilizados con la FFTW. Para realizar una transformada de Fourier 2D de un conjunto complejo de datos a un conjunto complejo datos en CUDA™, inicialmente se definen las dimensiones de la transformada (ver Figura 3-9). Se definen: el plan (configuración para el manejo en la librería cuFFT), la variable de entrada y la variable de salida de tipo **`complex`** en CUDA™ (**`float2`**). Se hace la asignación dinámica en la GPU de estas variables con la función **`cudaMalloc()`**. La función **`cufftplan2d()`** genera el plan de FFT para el uso de la librería cuFFT. El paso siguiente es ejecutar el plan con **`cufftExecC2C()`** (transformada de Fourier), en la que se determina si la trasformada es directa o inversa. En *odata* se asignan los datos de salida. Por último, se liberan las variables en la GPU, usadas por CUDA™, por medio de las instrucciones **`cufftdestroy()`** y **`cudaFree()`**.

Figura 3-9: Uso de la librería cuFFT para un arreglo bidimensional de datos.

```
#define NX 256
#define NY 128

cufftHandle plan;
```

```
cufftComplex *idata, *odata;

// Ubicación de la memoria del Device
cudaMalloc((void**)&idata, sizeof(cufftComplex)*NX*NY);
cudaMalloc((void**)&odata, sizeof(cufftComplex)*NX*NY);

// Declaración del plan
cufftPlan2d(&plan, NX, NY, CUFFT_C2C);

// Ejecución de la FFT
cufftExecC2C(plan, idata, odata, CUFFT_FORWARD);

//Ejecución de la FFT inversa
cufftExecC2C(plan, odata, odata, CUFFT_INVERSE);
cufftDestroy(plan);
cudaFree(idata); cudaFree(odata);
```

3.2.3 Interoperabilidad con librerías de gráficos

Una de las ventajas que presenta CUDA™ es la interoperabilidad con algunas de las librerías usadas ampliamente para manejo de gráficos. Entre estas librerías se encuentran: OpenGL [62] (de sus siglas en inglés: Open Graphics Library, desarrollada por Silicon Graphics Inc) y Direct3D (desarrollada por Microsoft®). Estas librerías consisten en una interfaz capaz de dibujar escenas complejas 2D y 3D a partir de puntos, líneas, triángulos, sombras y texturas.

En este caso, la librería OpenGL es la interfaz de gran uso por su facilidad de implementación. Esta proporciona una interfaz gráfica de usuario para CUDA™ a través de una completa interoperabilidad entre ellos. Algunos recursos de OpenGL se pueden asignar a los recursos de CUDA™, ya sea para permitir a CUDA™ leer los datos procesados por OpenGL o para permitir a CUDA™ transferir datos procesados para el consumo de OpenGL [17]. Esta interoperabilidad entre CUDA™ y OpenGL elimina la necesidad de transferir datos desde la memoria de la GPU a la memoria de la CPU cada vez que se necesita la visualización de los resultados de una operación realizada por un *kernel*, por ejemplo, sobre una imagen. Entre más pequeñas sean las trasferencias de datos más rápidas son las ejecuciones de los algoritmos, especialmente para grandes cantidades de datos a procesar.

La interoperabilidad de CUDA™ con OpenGL requiere la especificación **`cudaGLSetGLDevice()`** en el código desarrollado, antes de cualquier llamada en el tiempo de ejecución. Los espacios de memoria que deben ser usados por OpenGL en CUDA™ son el buffer, la textura y buffer de renderizado o *renderbuffer*. El buffer es

llamado mediante la instrucción **`cudaGraphicsGLRegisterBuffer()`**. En CUDA™ este buffer aparece como un puntero del dispositivo y puede ser leído y escrito por *kernels* a través de llamadas a **`cudaMemcpy()`**. Un objeto textura o *renderbuffer* se registra utilizando **`cudaGraphicsGLRegisterImage()`.** En CUDA™, este objeto aparece como un arreglo y por lo tanto puede estar sujeto a una textura de referencia y ser leído y escrito por llamadas a *kernels* o vía **`cudaMemcpy2D()`**. **`cudaGraphicsGLRegisterImage()`** soporta todos los formatos de textura con 1, 2 ó 4 componentes, un tipo *float* interno (por ejemplo **`GL_RGBA_FLOAT32`**) y enteros sin normalizar (por ejemplo **`GL_RGBA8UI`**). En la Figura 3-10 se muestra un ejemplo de un *kernel* que modificar dinámicamente una cuadrícula 2D de vértices almacenados de un tamaño **`width X height`** en un objeto búfer de vértices.

Figura 3-10: Código para la modificación dinámica de una malla 2D por medio de un *kernel* en OpenGL

```
GLuint positionsVBO;
struct cudaGraphicsResource* positionsVBO_CUDA;

int main()
{
// Establecer el dispositivo
cudaGLSetGLDevice(0);
// Inicializar OpenGL y GLUT
...
glutDisplayFunc(display);
// Crear el objeto buffer y registrarlo en CUDA
glGenBuffers(1, positionsVBO);
glBindBuffer(GL_ARRAY_BUFFER, &vbo);
unsigned int size = width * height * 4 * sizeof(float);
glBufferData(GL_ARRAY_BUFFER, size, 0, GL_DYNAMIC_DRAW);
glBindBuffer(GL_ARRAY_BUFFER, 0);
cudaGraphicsGLRegisterBuffer(&positionsVBO_CUDA,
positionsVBO,
cudaGraphicsMapFlagsWriteDiscard);
// Inicialización del loop de renderizado
glutMainLoop();
}

void display()
{
// Mapa del objeto buffer para escritura desde CUDA
float4* positions;
cudaGraphicsMapResources(1, &positionsVBO_CUDA, 0);
size_t num_bytes;
cudaGraphicsResourceGetMappedPointer((void**)&positions,
&num_bytes,
positionsVBO_CUDA));
// Ejecución del kernel
dim3 dimBlock(16, 16, 1);
dim3 dimGrid(width / dimBlock.x, height / dimBlock.y, 1);
createVertices<<<dimGrid, dimBlock>>>(positions, time,
width, height);
// Eliminar el mapa del objeto buffer
cudaGraphicsUnmapResources(1, &positionsVBO_CUDA, 0);
// Renderizado desde el objeto buffer
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
```

```
glBindBuffer(GL_ARRAY_BUFFER, positionsVBO);
glVertexPointer(4, GL_FLOAT, 0, 0);
glEnableClientState(GL_VERTEX_ARRAY);
glDrawArrays(GL_POINTS, 0, width * height);
glDisableClientState(GL_VERTEX_ARRAY);
// Intercambiar buffers
glutSwapBuffers();
glutPostRedisplay();
}

void deleteVBO()
{
cudaGraphicsUnregisterResource(positionsVBO_CUDA);
glDeleteBuffers(1, &positionsVBO);
}

__global__ void createVertices(float4* positions, float time,
unsigned int width, unsigned int height)
{
unsigned int x = blockIdx.x * blockDim.x + threadIdx.x;
unsigned int y = blockIdx.y * blockDim.y + threadIdx.y;
// Calcular coordenadas u,v
float u = x / (float)width;
float v = y / (float)height;
u = u * 2.0f - 1.0f;
v = v * 2.0f - 1.0f;
// calcular patrón de onda sinusoidal
float freq = 4.0f;
float w = sinf(u * freq + time)
* cosf(v * freq + time) * 0.5f;
// Escribir las posiciones
positions[y * width + x] = make_float4(u, w, v, 1.0f);
}
```

3.2.4 Manejo de errores

A fin de facilitar la ejecución concurrente entre el *host* y el dispositivo, algunas funciones son asíncronas: El control se devuelve al *hilo* del *host* antes de que el dispositivo ha completado la tarea solicitada.

Todas las funciones en CUDA™ devuelven un código de error, pero para una función asíncrona, este código de error no puede reportar ninguno de los errores asincrónicos que pueden ocurrir en el dispositivo ya que devuelve la función antes de que el dispositivo haya terminado la tarea; el código de error sólo informa de los errores que se producen en el sistema principal antes de ejecutar la tarea en el dispositivo, por lo general relacionados con la validación de parámetros. Si se produce un error asíncrono, será informado por alguna llamada a una función subsiguiente en el tiempo de ejecución [17].

Una manera de comprobar si hay errores asíncronos es sincronizando los *hilos* justo después de la llamada a una función asíncrona por medio de la instrucción

cudaThreadSynchronize() y comprobando el código de error devuelto por la misma instrucción [17]. Otra manera de hacerlo es deshabilitando la ejecución de *kernels* asíncronos para toda la aplicación de CUDA™ estableciendo la variable de entorno **CUDA_LAUNCH_BLOCKING** en 1. Esta característica de CUDA™ se proporciona sólo para fines de depuración y nunca debe ser usado como una manera de hacer programación [17].

En el tiempo de ejecución, CUDA™ mantiene una variable por cada *hilo* del *host* que es inicializado en **cudaSuccess** y se sobre-escribe por el código de error cada vez que este ocurre. La función **cudaPeekAtLastError()** devuelve una variable de error mientras que la función **cudaGetLastError()** retorna la variable de error además de restablecer el sistema en **cudaSuccess**.

La ejecución de *kernels* no devuelve ningún código de error por lo que las funciones **cudaPeekAtLastError()** y **cudaGetLastError()** deben ser llamadas justo después de la llamada del *kernel* para recuperar los errores. Para asegurarse que cualquier error devuelto por **cudaPeekAtLastError()** o **cudaGetLastError()** no se origina en las llamadas antes de la ejecución del *kernel*, hay que asegurarse de la variable de error en tiempo de ejecución se establece en **cudaSuccess** justo antes de la llamada del *kernel*, por ejemplo, llamando al **cudaGetLastError()** antes de la llamada del *kernel*. Las llamadas de los *kernel* son asíncronas, por lo que para comprobar si hay errores asincrónicos, la aplicación debe sincronizar entre la llamada del *kernel* y el llamado a **cudaPeekAtLastError()** o **cudaGetLastError()**.

En el capítulo 4 se mostrará en detalle una implementación de un sistema de interferometría holográfica digital cuyo cálculo acelerado con GPU fue desarrollado para operar a tasas de video. Para esta implementación en GPGPU, se desarrolló un algoritmo de reconstrucción de hologramas por medio de la transformada de Fresnel, cálculo de fases de interferencia y finalmente se implementó el cálculo de la deformación mecánica de una membrana circular de aluminio en CUDA™ C usando los elementos descritos en los capítulos anteriores.

4. Interferometría Holográfica Digital (IHD) en Tiempo Real

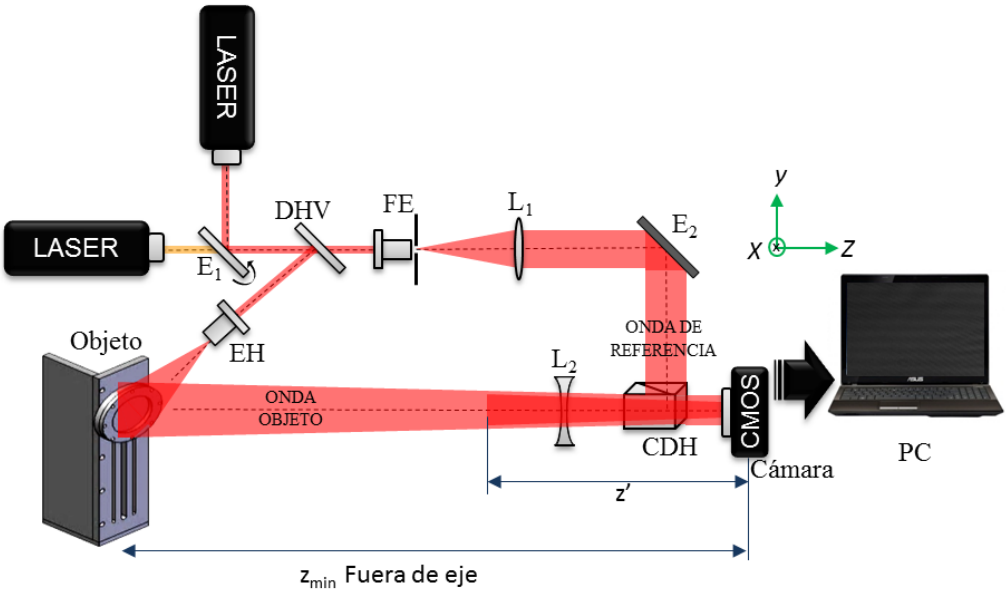
A lo largo de esta tesis de maestría se han presentado los fundamentos de holografía digital (HD), interferometría holográfica digital (IHD), desenvolvimiento de fase y programación GPGPU con el fin de desarrollar una herramienta basada en la técnica de IHD para medición de deformaciones a velocidad de video. En este capítulo se muestra una implementación de esta técnica: Se incluyen los cálculos realizados para el montaje experimental así como una descripción detallada del funcionamiento del algoritmo implementado en CUDA™ (computer unified device architecture), y finalmente se presentan los resultados y el rendimiento de la implementación.

4.1. Diseño experimental

El montaje experimental implementado para la aplicación de IHD se muestra en la Figura 4-1. Este es un montaje típico de holografía digital fuera de eje. La implementación consiste en un espejo E_1 que permite escoger entre un láser de H_eN_e centrado en una longitud de onda $\lambda_1 = 594nm$ o un láser centrado en $\lambda_2 = 632.8nm$, un divisor de haz variable (DHV) que produce una onda objeto y una onda de referencia a partir de un láser. La onda objeto es expandida luego de atravesar el DHV por medio de un objetivo de microscopio de $60X$ marca Newport. Este EH (expansor de haz) asegura que la porción del objeto de interés será iluminada. La onda esparcida por el objeto es recogida por una lente divergente (L_2) de distancia focal $-150mm$ que se encarga de producir un frente de onda reducido del objeto para visualizarlo en su totalidad. En la onda de referencia, un filtro espacial (FE) es el encargado de remover las fluctuaciones aleatorias del perfil de intensidad del láser debidas a partículas de polvo depositadas sobre la óptica empleada, por ejemplo. Una vez es filtrado el haz de referencia, se colima con una lente colimadora (L_1) de $25mm$ de distancia focal. El espejo (E_2) es el encargado de direccionar

la onda de referencia hacia el cubo divisor de haz (CDH) quien se encarga de combinar el haz objeto y el haz de referencia para formar el patrón de interferencia entre estas dos ondas sobre la superficie del sensor. En este caso, el frente de onda es registrado por una cámara CMOS marca Thorlabs con características como: número de pixeles 1280x1024, tamaño de pixel de $5{,}2\mu m$ x $5{,}2\mu m$, registros de 25 FPS (Frames por segundo). Para esta implementación se usó un arreglo cuadrado de 1024x1024 pixeles. El objeto analizado en esta aplicación consta de una membrana circular de aluminio de $8cm$ de diámetro y $1{,}5mm$ de espesor la cual está sujeta en todo su perímetro por medio de tornillos como se puede observar en la Figura 4-1. Un tornillo micrométrico motorizado con pasos de $33nm$, es localizado en el centro de la membrana con el fin de producir micro-deformaciones en ella.

Figura 4-1: Esquema de un montaje experimental de IHD. **E_1**: Espejo con base rotatoria, **DHV**: Divisor de haz variable, **FE**: Filtro espacial, **L_1**: Lente colimadora, **E_2**: Espejo, **EH**: Expansor de Haz, **L_2**: Lente divergente, **CDH**: Cubo Divisor de Haz, **CMOS**: Cámara (del inglés Complementary metal-oxide-semiconductor), **PC**: Computador Personal.

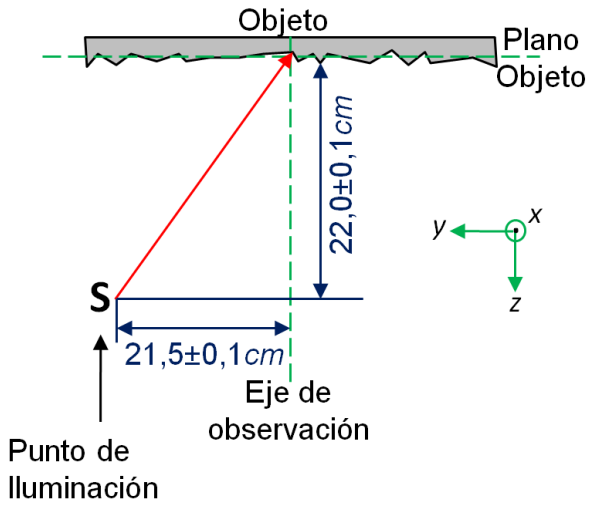


Para este montaje experimental tenemos una distancia mínima entre el objeto y la cámara CMOS, de acuerdo a la ecuación (1-17) $z_{min}\ \text{Fuera de eje} = 3{,}7m$. Esta distancia es poco práctica dentro del laboratorio, es por tal razón que se incluyó una lente divergente para disminuir el frente de onda esparcido por el objeto (ver sección 1.1.1). La nueva distancia de reconstrucción es $z' = 0{,}34m$ de acuerdo a la ecuación (1-21).

4.1.1 Cálculo del vector sensitividad y factor geométrico

El vector sensitividad fue calculado de acuerdo a la geometría mostrada en la Figura 4-2. La dirección de observación es perpendicular al plano del objeto. El vector unitario de iluminación $\vec{s}(0\ \ -0{,}699\ \ -0{,}715)$ y el vector de observación $\vec{b}(0\ \ 0\ \ 1)$ proporcionan una diferencia $(\vec{b}-\vec{s})=(0\ \ 0{,}699\ \ 1{,}715)$. De acuerdo a estos valores, el vector sensitividad está dado por $\vec{S}=(2\pi/\lambda)(0\ \ 0{,}699\ \ 1{,}715)$ según la ecuación (1-40). El factor geométrico fue calculado de acuerdo a la ecuación (1-42) siendo $\lambda=\lambda_2=632.8nm$. Dicho valor es $FG_{\lambda_2}=54{,}38\pm0{,}09nm$.

Figura 4-2: Esquema geométrico para el cálculo del vector sensitividad.



4.2. Reconstrucción de hologramas en paralelo

En nuestra implementación, debido a que el proceso de reconstrucción numérica es altamente paralelizable, esto quiere decir que la mayoría de sus operaciones numéricas se pueden hacer punto a punto, el cálculo fue llevado a cabo por medio de la arquitectura CUDA™ desarrollada por la compañía NVIDIA®. En esta aproximación se usó un hilo (ver capítulo 3, pag. 54) de la CPU para controlar directamente la adquisición de la cámara CMOS y transferir los datos del holograma digital a la memoria de la GPU.

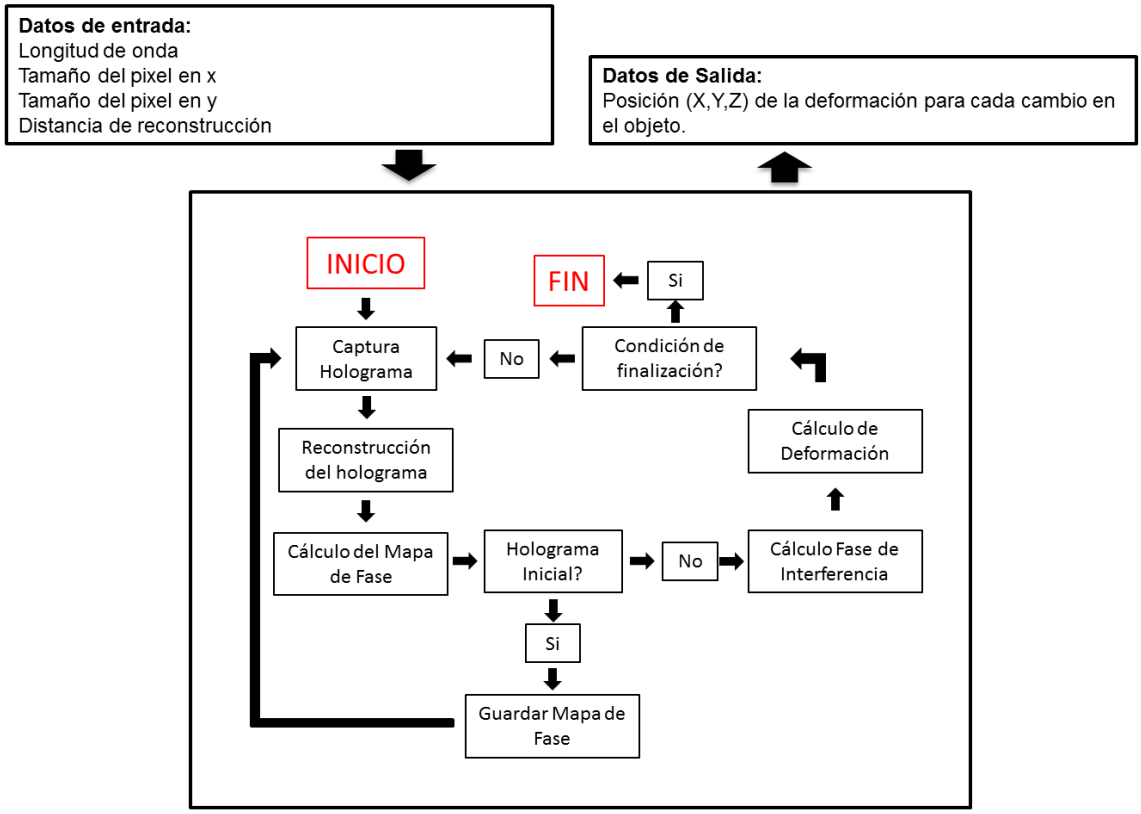
Una vez la GPU obtiene los datos del holograma, comienza el filtrado espacial del holograma para eliminar el orden cero y la imagen gemela: una matriz A de ceros de tamaño $N_x \text{ x } N_y$ es almacenada en la memoria de la GPU; los datos del holograma digital se pasan al dominio de las frecuencias donde solo se toman los valores de frecuencia de la imagen real, estos datos son centrados y superpuestos en la matriz A. Luego se vuelve al dominio espacial. Finalmente, esta matriz se reconstruye numéricamente por medio de la transformada de Fresnel (ecuación (1-28)) y la fase es obtenida de acuerdo a la ecuación (1-54). Esta matriz es visualizada a través de OpenGL y almacenada en la memoria global de la GPU. En la siguiente adquisición de la cámara, se hace el mismo proceso de filtrado y la fase es calculada por medio de la ecuación (1-55). La fase de interferencia es obtenida de acuerdo a la ecuación (1-56) respecto al primer mapa de fase calculado (ver Figura 4-3).

La obtención de la fase de interferencia y la deformación se repite hasta que se alcanza la condición de finalización. La sincronización entre los hilos de la GPU, luego de una instrucción en CUDA™, garantiza su correcta visualización a través de OpenGL para cada una de los *frames* adquiridos por la cámara.

En la Figura 4-3 se muestra un esquema del funcionamiento del algoritmo realizado en esta tesis de maestría. En esta implementación, se requieren algunos datos de entrada para el procesamiento del holograma: longitud de onda, tamaño del pixel en x, tamaño del pixel en y y la distancia de reconstrucción. El algoritmo retorna los valores (x, y, z) de la deformación para cada uno de los hologramas registrados en un estado particular del objeto. El cálculo de la deformación es obtenido luego de desenvolver la fase por medio del método de dos longitudes de onda (ver capítulo 2).

La mejora en la rapidez de cálculo de esta metodología es lograda gracias al proceso de reconstrucción. La reconstrucción numérica es paralelizada usando la potencia del hardware multi-procesador de la GPU accedida por medio de la arquitectura de programación CUDA™ [17].

Figura 4-3: Esquema del algoritmo empleado para el sistema de IHD a velocidades de video.



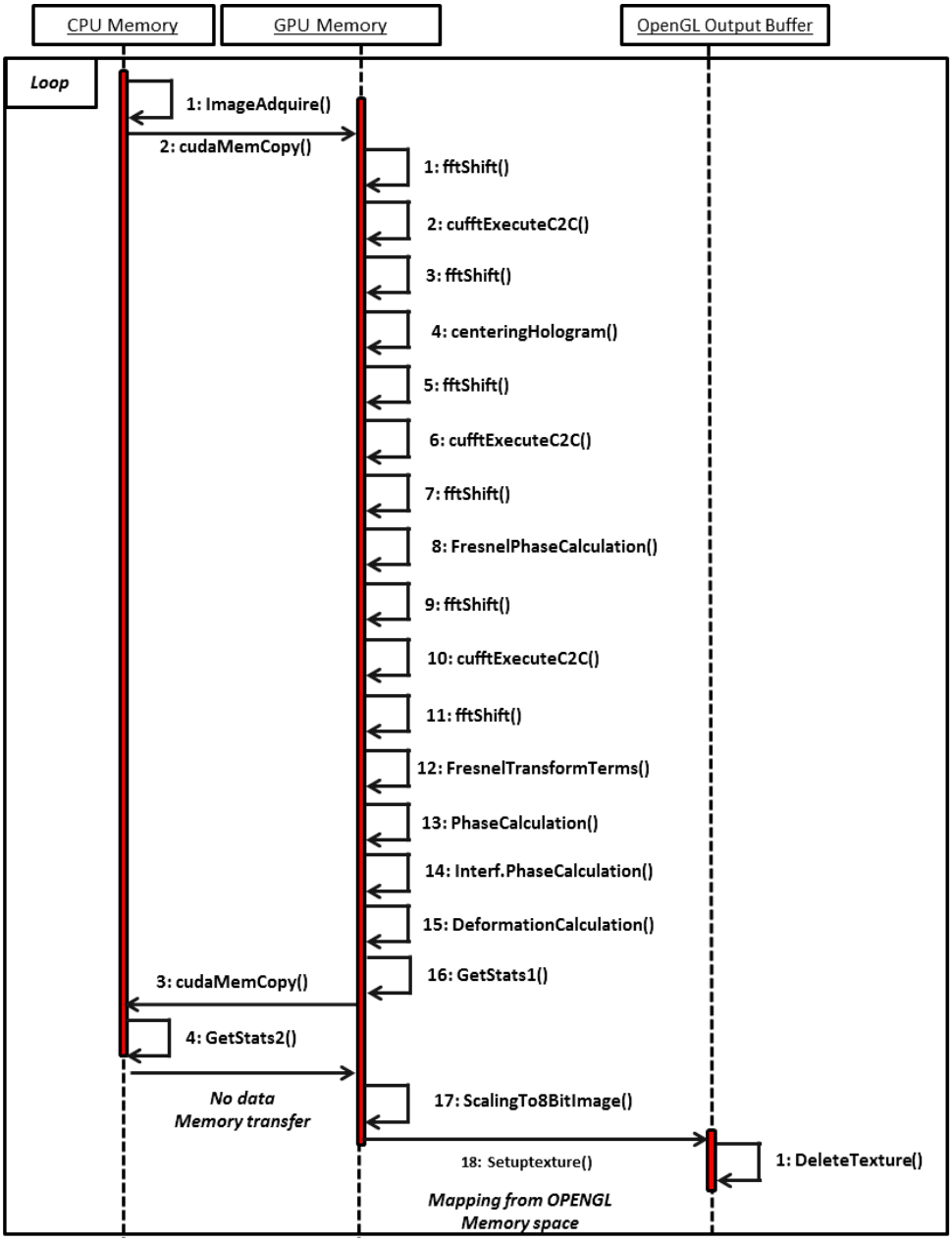
En esta tesis de maestría fue usada una GPU de la serie GeForce GTX 580 de NVIDIA® compuesta de 16 multi-procesadores cada uno de ellos con 32 núcleos disponibles, por lo tanto, hasta 512 *hilos* accesibles simultáneamente para el cómputo en paralelo. Por esta última característica, la programación en paralelo por medio de la GPU puede ser empleada para llevar a cabo una aplicación de complejidad computacional de O2(M x log M)^2 requerida para la reconstrucción de hologramas digitales y obtención de la fase de interferencia del sistema IHD. Debido a que la reconstrucción numérica de los hologramas digitales se hace por medio de la FFT (de sus siglas en inglés: Fast Fourier Transform), la implementación en CUDA™ usa la librería CUFFT [60] desarrollada por NVIDIA® (ver capítulo 3 sección 3.2.2 para más detalle). La librería CUFFT, como se mencionó en el capítulo anterior, es un algoritmo que calcula la FFT basado en la FFTW [63] que se optimizó para su cálculo en paralelo sobre la GPU.

En la Figura 4-4 se muestra un diagrama UML (Lenguaje Unificado de Modelado, de sus siglas en inglés: *Unified Modeling Lenguage*) del flujo de datos entre las memorias de la

CPU y GPU además del buffer de salida del OpenGL para la implementación hecha en esta tesis de maestría. El primer hilo de la CPU adquiere el holograma directamente de la cámara CMOS por medio de la función *ImageAcquire()* para la longitud de onda λ_2. Esta función retorna el holograma registrado a la memoria de la CPU. Luego, la función *cudaMemCopy()* transfiere los datos del holograma a la memoria de la GPU. Una vez finaliza esta operación, el orden cero y la imagen virtual es removida del holograma por medio de las instrucciones o *kernels* numerados del 1 al 7 en el diagrama UML. El kernel 8: *FresnelPhaseCalculation()* calcula la fase de fresnel ($e^{-\frac{i\pi}{\lambda z}\left(k^2\Delta x_h^2+l^2\Delta y_h^2\right)}$) en la ecuación (1-28) y multiplica su resultado con el holograma filtrado pixel por pixel. A los datos complejos, resultado de la anterior operación, se les aplica la transformada de Fourier para reconstruir el objeto, esto mediante los *kernels* 9 a 11 en la Figura 4-4. Una vez se hace la FFT, la transformada de fresnel (TF) se lleva a cabo por el kernel 12: *FresnelTransformTerm()* el cual se encarga de multiplicar los demás términos de fase de la TF (ecuación (1-28)). La fase del holograma reconstruido es calculada con el *kernel* 13: *PhaseCalculation()* y subsecuentemente, la fase de interferencia es calculada por el *kernel* 14: *Interf.PhaseCalculation().* El cálculo de las micro-deformaciones es hecho por el *kernel* 15: *DeformationCalculation()* (ecuación (1-42)).

Con el fin de visualizar los resultados es necesario escalar los valores del cálculo de la fase de interferencia o de micro-deformación y usar completamente el rango dinámico de visualización del OpenGL, el *kernel* 17: *ScalingTo8BitImage()* usa el valor máximo y el valor mínimo del resultado final producidos por el *kernel* 16: *GetStats1()* y *GetStats2()*; estos dos kernels retornan los resultados a la memoria global que es accesible por ambos dispositivos: la CPU y la GPU. El *kernel* 18: *Setuptexture()* configura la textura en la memoria global usada por OpenGL para visualizar los resultados del sistema de IHD: valores de micro-deformaciones o fase de interferencia. Luego de la visualización, el kernel *Deletetexture()* es el encargado de borrar los datos de la memoria con el fin de no desbordar la memoria del dispositivo.

Figura 4-4: Diagrama ULM del flujo de datos entre la memoria de la CPU, GPU y el buffer de salida del OpenGL. En cada paso, una función de la CPU o una función de la GPU (*kernel*) es representada.



La configuración de los *hilos* de programación para los *kernels CenteringHologram(), FresnelPhaseCalculation(), FresnelTransformTerm(), PhaseCalculation(), Interf.PhaseCalculation(), DeformationCalculation()* y *scalingTo8BitImage()*, siguen el esquema regular sugerido por la guía de programación de CUDA™ C [17]. La malla

cuadrada está compuesta de bloques cuadrados de $16\text{x}16$ *hilos*. El ancho y alto de la malla está determinado por el tamaño del holograma registrado digitalmente; el número de bloques está dado por $N/16$, con N el número de pixeles a lo largo de cada una de las direcciones del holograma. Para nuestro caso, hologramas de $1024\text{x}1024$ pixeles, fue usada una malla de 64x64 bloques, cada uno de ellos corriendo en $16\text{x}16$ *hilos*. De esta manera, cada uno de los pixeles del holograma digital es reconstruido simultáneamente asignando un *hilo* a cada pixel.

El *kernel fftshift()* intercambia los cuadrantes de los datos devueltos por el algoritmo FFT. Para ejecutar esta tarea, sólo es necesario un cuarto de la malla descrita. La función *cufftExecuteC2C()* de la librería CUFFT 3.1 calcula la configuración de cada *kernel* que la compone para disminuir el tiempo de procesamiento global [60]. Los *kernel* son configurados sobre una malla de $1024\text{ x }1$ bloques operando con $128\text{ x }1$ *hilos*.

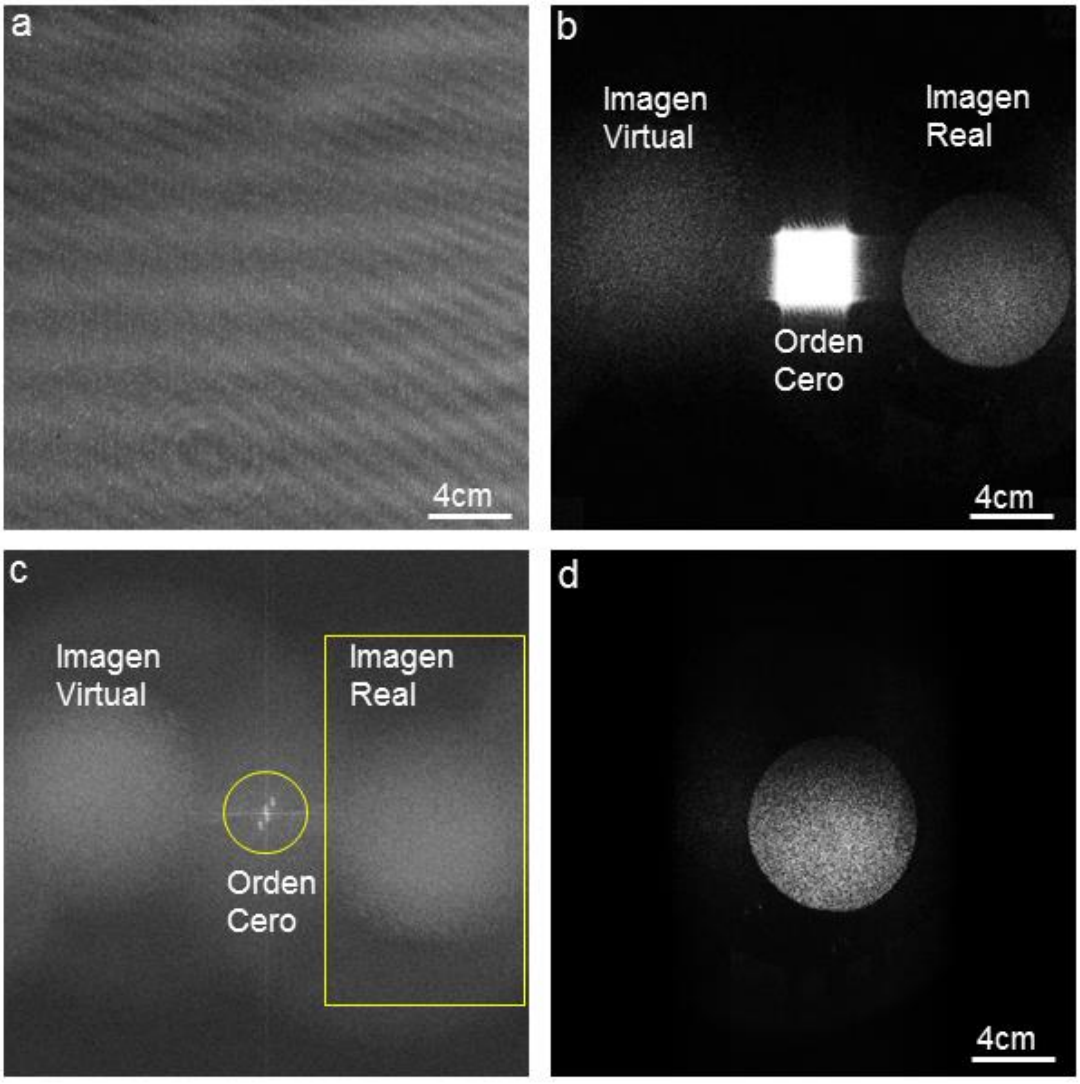
La reconstrucción numérica acelerada de hologramas digitales para obtener la fase de interferencia y las micro-deformaciones fue implementada en un computador personal que contiene un procesador Intel® Core™ i7-2600 corriendo a 3.40GHz y 8GB de memoria RAM. Este procesador soporta una tarjeta graficadora NVIDIA® de la serie Geforce GTX 580 que contiene 512 núcleos CUDA™ con frecuencia de 1544 MHz y 1536MB de memoria local. Los hologramas digitales fueron registrados con una cámara CMOS con una resolución de 1024x1204 pixeles conectada al computador por medio de la interfaz desarrollada por IDS®: USB2.0. La cámara adquiere a 25 FPS (frames per second) lo que se traduce a un tiempo de $40ms$ para el registro de un holograma en la memoria de CPU. El tiempo de transferencia entre memorias de la CPU y la GPU es de $1ms$ en ambas direcciones para un arreglo de 1024x1024. El proceso de reconstrucción numérica por cada longitud de onda toma $646{,}9ms$ en la CPU mientras que el mismo proceso en la GPU toma un tiempo de $4{,}9ms$. Este hecho significa que en el proceso de reconstrucción, desde el momento en el que el holograma es registrado en la memoria de la CPU, el método propuesto reduce el tiempo de procesamiento unas 133 veces por holograma reconstruido.

4.3. Resultados

En la Figura 4-5 se muestra el proceso de reconstrucción numérica y el filtrado espacial del holograma para suprimir el orden cero y la imagen virtual. En el panel b se muestra una imagen compuesta por *i)* la imagen virtual, *ii)* el orden cero y *iii)* la imagen real producto del proceso holográfico mostrado en la ecuación (1-5). Debido a que la holografía digital es un método de formación de imágenes basado en el fenómeno de difracción, la imagen virtual y la imagen real pueden ser entendidas como el orden de difracción +1 y -1 del holograma registrado. Las dos imágenes están simétricamente ubicadas respecto a la posición del holograma sobre la distancia de reconstrucción; esto significa que solo una de ellas puede estar enfocada a la distancia de reconstrucción dada por el sistema de holografía digital (HD) como se muestra en el esquema de la Figura 4-1. En la Figura 4-5 b. la distancia de reconstrucción ha sido elegida para obtener la imagen real enfocada. El rectángulo brillante ubicado en el centro de la imagen reconstruida es el orden cero de difracción producto de las ondas que no son esparcidas por el objeto. La presencia de la imagen virtual y del orden cero de difracción disminuyen el contraste de la imagen reconstruida como se puede ver claramente en la Figura 4-5 b. y d. Con el fin de eliminar el orden cero y la imagen virtual, se han implementado diferentes métodos [25,26]. Uno de estos métodos ampliamente usados es el filtrado de la imagen virtual y el orden cero en la transformada de Fourier del holograma digital. En la Figura 4-5 c. se muestra el módulo del espectro de Fourier del holograma registrado. El orden cero está encerrado en un círculo y está ubicado en el centro de la imagen. El área delimitada por el rectángulo amarillo corresponde a la imagen real cuyos datos se extraen para tener solo la información del objeto en estudio. Esta información es puesta en el centro de un arreglo de ceros que tiene dimensiones idénticas al espectro de Fourier. El espectro de Fourier modificado, es operado mediante su transformada inversa para obtener la reconstrucción del holograma adaptado, el cual no contiene información de la imagen virtual y del orden cero de difracción. La reconstrucción del holograma adaptado se muestra en la Figura 4-5d, la cual muestra un contraste mejorado pues elimina el ruido distractor en la imagen reconstruida y por lo tanto, más detalles del objeto son ahora visibles. El precio que debe ser pagado por esta mejora es el incremento de la complejidad computacional. Debido a que el método de supresión de la imagen virtual y el orden cero de difracción es realizado en el espectro de Fourier, es necesario ejecutar por lo menos dos transformadas de Fourier adicionales. Como resultado, la complejidad

computacional para la reconstrucción numérica de un holograma digital es por lo menos doble $O2(M \times \log M)^2$.

Figura 4-5: Filtrado espacial del holograma: supresión del orden cero y de la imagen virtual en la reconstrucción numérica del holograma digital: a. holograma digital registrado directamente de la cámara CMOS. b. Reconstrucción numérica por medio de la ecuación (1-28). c. Transformada de Fourier del holograma y d. Reconstrucción numérica de la porción de la imagen real mostrada en el rectángulo amarillo del panel c.



Con el fin de optimizar el montaje experimental del sistema de IHD se hizo un análisis del contraste del holograma reconstruido vs. La relación de intensidad objeto – referencia. La relación de intensidades está dada por

$$ri = \frac{i_{obj}}{i_{ref}} \tag{4-1}$$

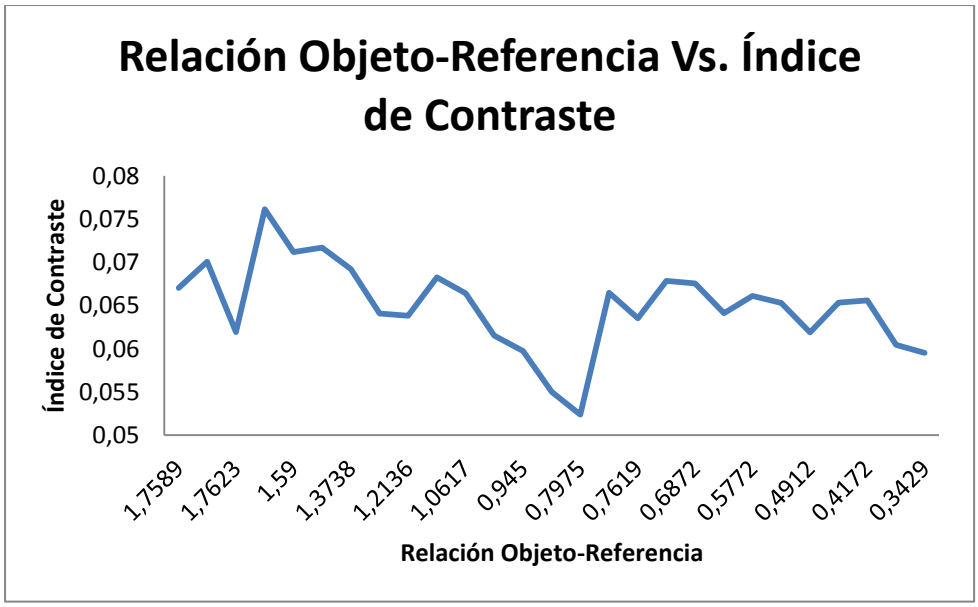
con i_{obj} intensidad onda objeto y i_{ref} intensidad onda de referencia.

El índice de contraste IC fue calculado por medio de un método ampliamente empleado en análisis de imágenes basado en el análisis de la desviación estándar del histograma de la imagen [64], en nuestro caso, la imagen de la reconstrucción del holograma por medio de la transformada de Fresnel (ecuación (1-28)).

$$IC = \frac{\sigma_{Pixeles}}{\sigma_{normalización}} \tag{4-2}$$

Donde $\sigma_{Pixeles}$ es la desviación estándar de los pixeles de la imagen y $\sigma_{normalización}$ es la desviación estándar de una imagen altamente contrastada que corresponde al valor de 128. Los resultados pueden observarse en la Figura 4-6.

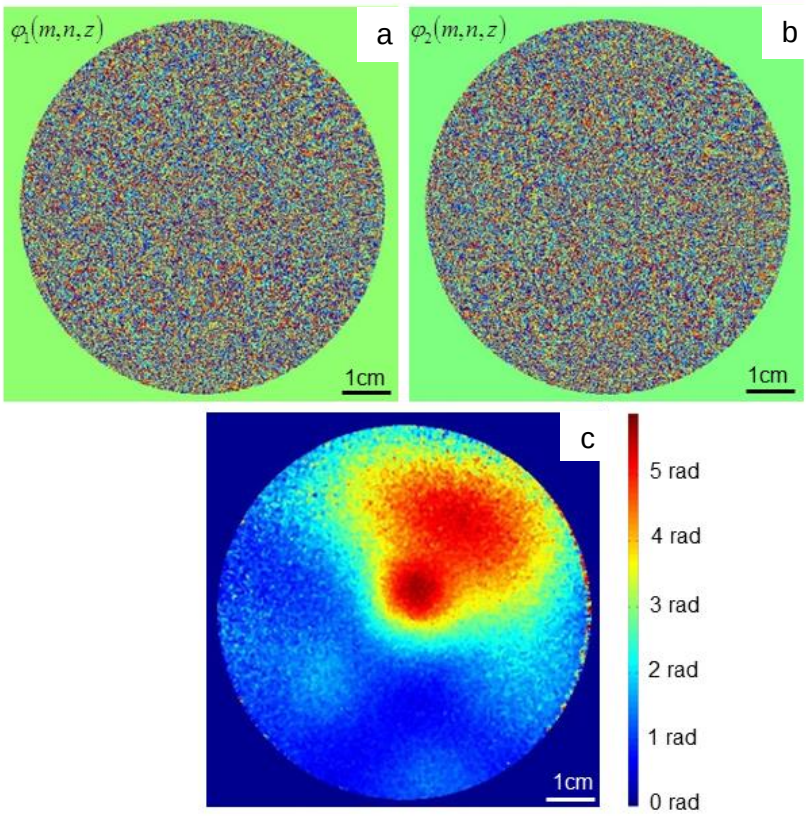
Figura 4-6: Curva de Contraste Vs. Relación de Intensidad Objeto-Referencia.



En la Figura 4-6 puede notarse que se alcanza un valor máximo de contraste de la reconstrucción del holograma, esto indica que para una relación de intensidad objeto – referencia de 1.59 el montaje experimental presentará el mayor rendimiento en la reconstrucción.

Una vez la reconstrucción numérica es calculada por medio de la ecuación (1-28), se procede al cálculo del mapa de fase del holograma por medio de la ecuación (1-54) (ver Figura 4-7a.). En la siguiente adquisición para un estado deformado del objeto, se procede de la misma forma y se obtiene el mapa de fase por medio de la ecuación (1-55) (Figura 4-7b.). Luego, la fase de interferencia es calculada por medio de la ecuación (1-56). Los resultados de estas operaciones se muestran en la Figura 4-7. El tiempo de procesamiento para obtener la fase de interferencia es de 21 FPS ($47{,}64\ ms$) para cada una de las longitudes de onda.

Figura 4-7: Cálculo del mapa de fase: a. Mapa de fase resultado del estado no deformado del objeto, b. Mapa de fase para un estado deformado del objeto y c. Fase de interferencia. Los valores de fase están en módulo 2π.



Luego de la obtención de la fase desenvuelta, el paso a seguir es el cálculo de las micro-deformaciones como se muestra en la siguiente sección. Allí, el rango de medición de deformaciones por medio de la técnica de IHD implementada en esta tesis de maestría es de $0{,}00\pm0{,}09nm$ hasta $4.800{,}00\pm0{,}09nm$.

4.3.1 Medición de la fase de interferencia

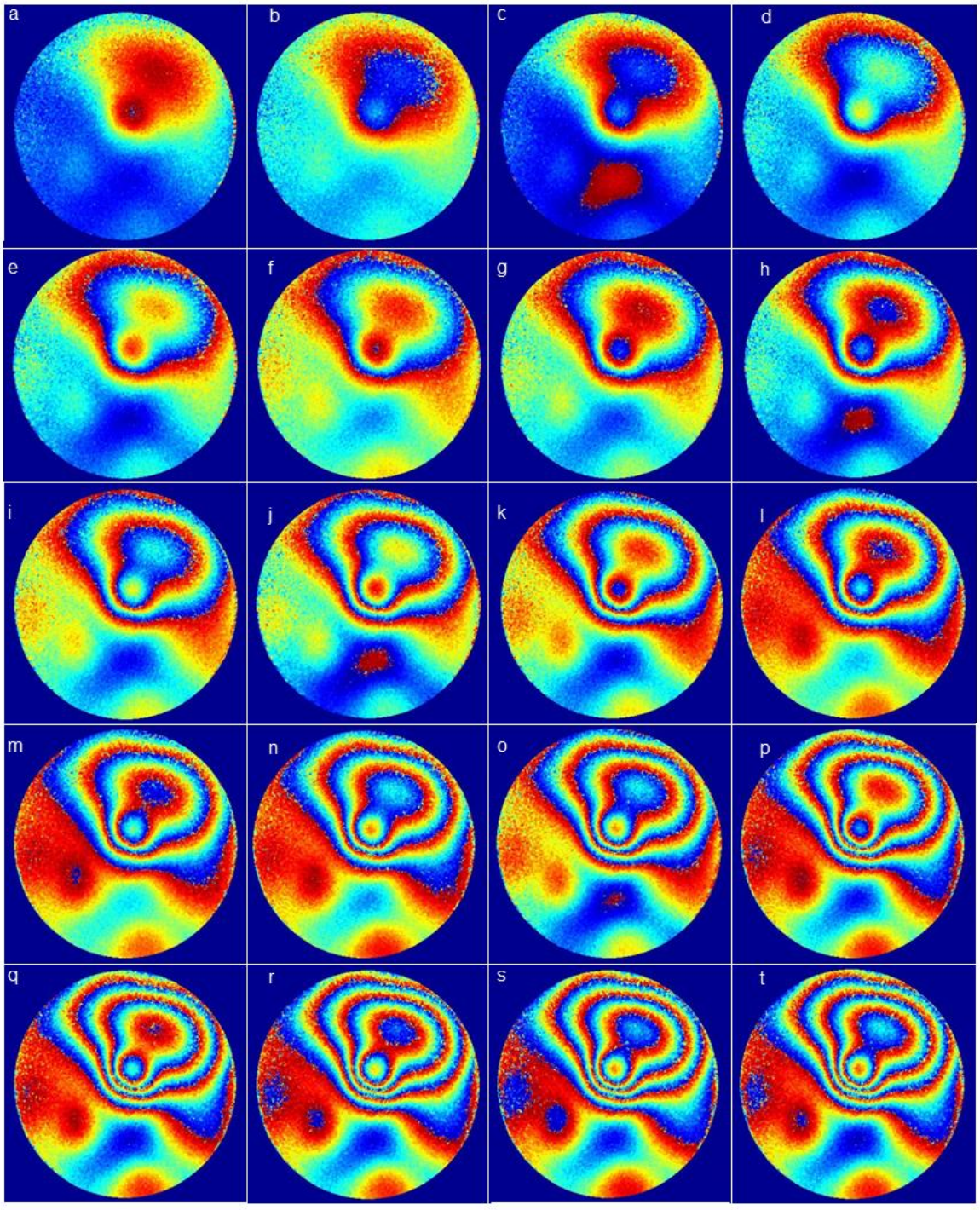
De acuerdo al montaje experimental implementado en esta tesis de maestría, para deformaciones mayores a $363{,}00 \pm 0{,}09nm$ introducidas por el tornillo micrométrico motorizado (actuador), la fase de interferencia registrada con una longitud de onda $\lambda_2 = 632.8nm$ se envuelve. Esto significa que hay saltos abruptos en los valores del mapa de fase de 0 a 2π como se pueden observar en la Figura 4-8 por su cambio contiguo del color rojo al color azul (ver capítulo 2). El avance del actuador es de $33nm$ por cada $0.16V$.

Como se puede observar en la Figura 4-8, a medida que se deforma la membrana, el grado de envolvimiento de la fase aumenta. Dicho envolvimiento obedece a ambigüedades o discontinuidades por el desfase de la onda incidente en la superficie de registro conforme avanza el frente de onda. El envolvimiento de la fase se da cuando los desfases sobre la superficie de registro de la onda incidente son mayores a 2π, los cuales con producto de deformaciones del objeto mayores que $\lambda/2$. Para el caso del registro con $\lambda_2 = 632.8nm$ la fase de interferencia se envuelve para deformaciones mayores a $316nm$ aproximadamente.

4.3.2 Medición de deformaciones mecánicas sin envolvimiento de fase

Luego de obtener el mapa de fase de interferencia, en el cual no se presenta envolvimiento con $\lambda_2 = 632.8nm$, esto es para deformaciones en la membrana que no excedan los $316nm$, el cálculo de la deformación es obtenido por medio de la ecuación (1-41). El factor geométrico calculado en la sección 4.1.1 es multiplicado por la fase de interferencia desenvuelta para obtener los valores de las micro-deformaciones introducidas en el objeto. El resultado de este proceso es mostrado en la Figura 4-9.

Figura 4-8: Resultados de la fase de interferencia respecto a una fase de referencia para diferentes desplazamientos del actuador.

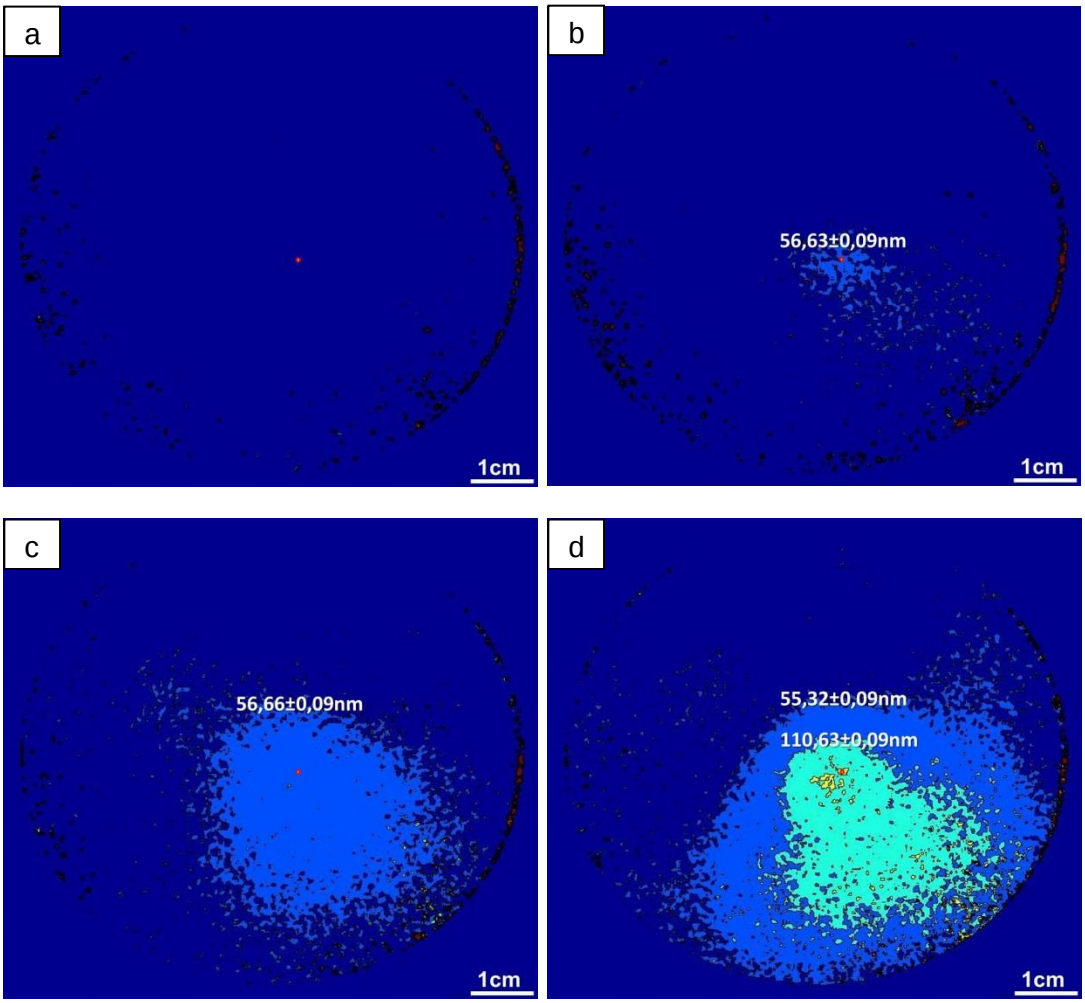


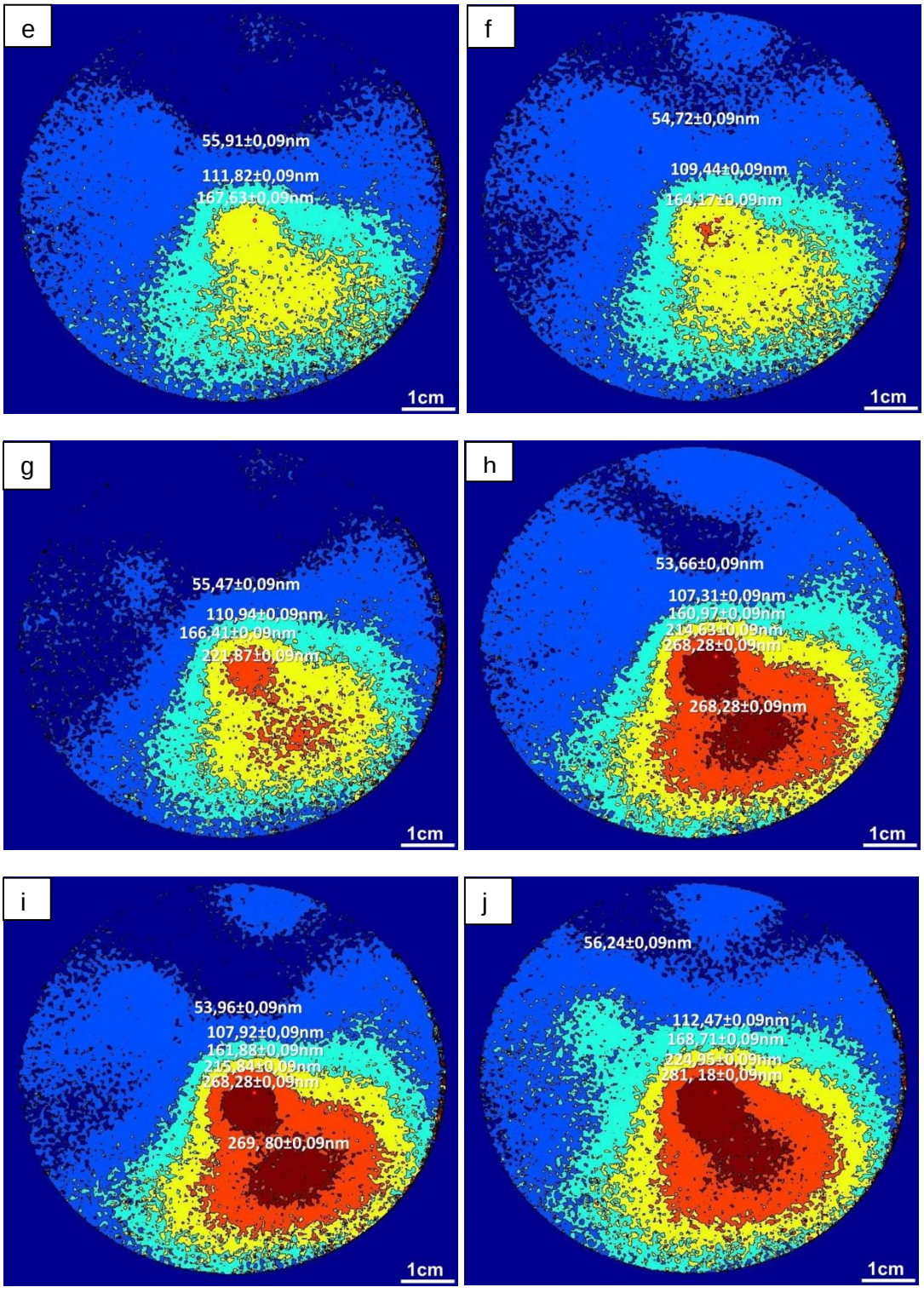
Nº Imagen	Avance Actuador [nm]
a	363
b	396
c	429
d	462
e	495
f	528
g	561
h	594
i	627
j	660
k	693
l	726
m	759
n	792
o	825
p	858
q	891
r	924
s	957
t	990

Las imágenes mostradas en la Figura 4-9 son divididas en áreas cuya delimitación posee el mismo valor de deformación. Cada una de estas imágenes contiene zonas marcadas

en un color específico y sus valores de deformación para el perímetro que encierra cada color se detalla en la Tabla 4-1. El punto delineado con color rojo y coloreado con el color amarillo es el punto de referencia elegido para la comparación con el avance del actuador ubicado en la parte posterior de la membrana para introducir deformaciones en ella.

Figura 4-9: Micro-deformación mecánica en una membrana de aluminio.





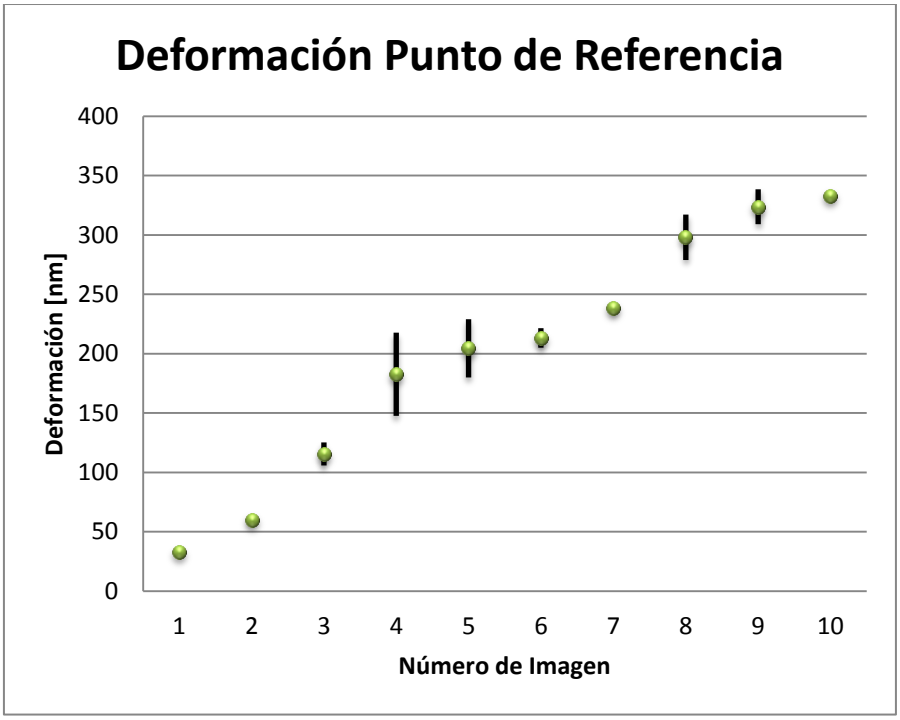
En la Figura 4-9 se puede notar que la delimitación de las diferentes zonas no es una línea suavemente continua, este hecho obedece a que el sistema de holografía digital presenta ruido de speckle el cual no fue tratado en esta tesis de maestría.

Tabla 4-1: Resultados obtenidos de la deformación de una membrana de aluminio por la técnica de IHD.

Figura Nº	Avance actuador [nm]	Deformación punto de referencia [nm]	N° de Divisiones	Deformación en la división	% ERROR punto de referencia
Figura 4-9a	33	33,09	1	0	0,27
Figura 4-9b	66	60,27	1	56,63	8,68
Figura 4-9c	99	115,62	1	56,66	16,79
Figura 4-9d	132	182,62	2	55,32 110,63	38,35
Figura 4-9e	165	204,5	3	55,91 110,82 167,63	23,94
Figura 4-9f	198	213,28	3	54,72 109,44 164,17	7,72
Figura 4-9g	231	238,9	5	55,47 110,94 166,41 221,87	3,42
Figura 4-9h	264	298,03	5	53,66 107,31 160,97 214,63 268,28	12,89
Figura 4-9i	297	323,76	5	53,96 107,92 161,88 215,84 268,28	9,01
Figura 4-9j	330	333,08	5	56,24 112,47 168,71 224,95 281,18	0,93

En la Figura 4-10 se muestra una curva que compara la deformación que sufre la membrana en un punto de referencia respecto al avance del actuador. Las barras negras corresponden al porcentaje de error detallado en la Tabla 4-1. Para cada una de las imágenes mostradas en la Figura 4-9, el subíndice **a** corresponde a la imagen **1**, **b** corresponde a la imagen **2** y así sucesivamente la **j** corresponde a la imagen **10**.

Figura 4-10: Curva de la deformación en un punto de referencia de la membrana y su comparación con el avance del actuador en cada una de las imágenes mostradas en la Figura 4-9: El punto 1 corresponde a la Figura 4-9a, el punto 2 corresponde a la Figura 4-9b y así sucesivamente.



En la Figura 4-10 puede observarse que los puntos 4, 5, 8 y 9 tienen una deformación diferente al avance que tiene el actuador lo que se traduce en un porcentaje de error alto como lo muestra su barra de error.

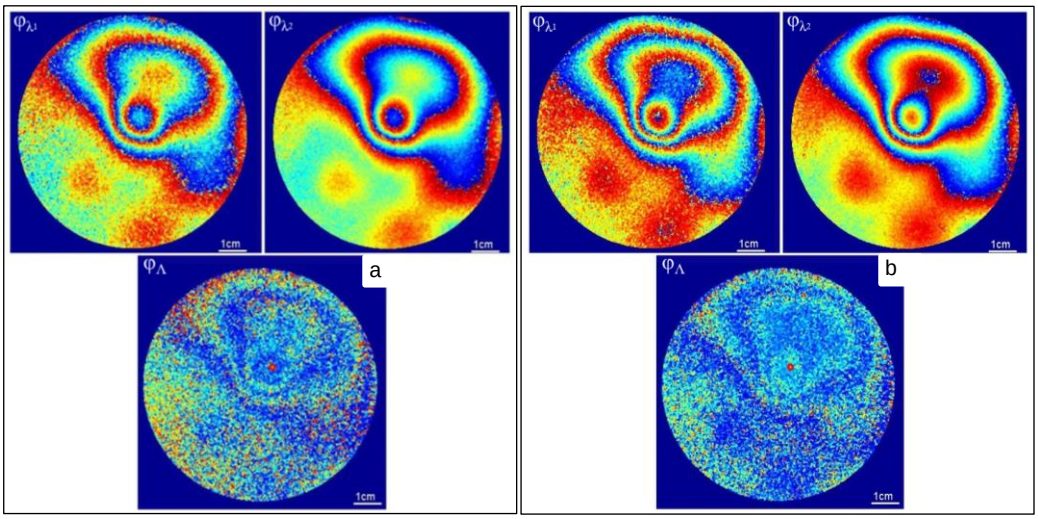
4.3.3 Desenvolvimiento de fase y medición de deformaciones mecánicas

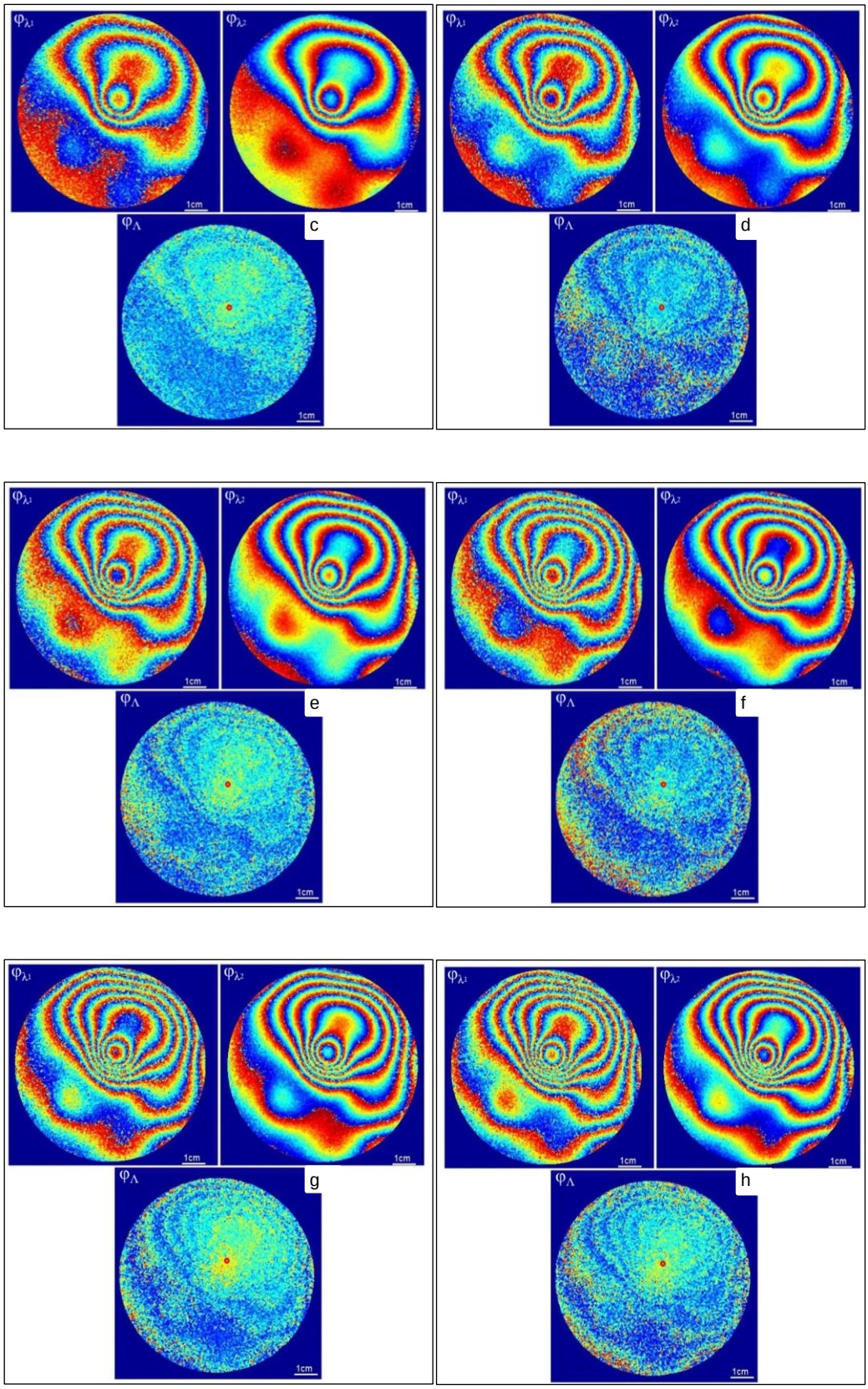
Como se mostró en la Figura 4-8, cuando las deformaciones son mayores a $363nm$ la fase se envuelve. Por tal motivo, es necesario implementar métodos que permitan la obtención de la fase desenvuelta o lo que es lo mismo, que no presenten discontinuidades. En el capítulo 2 se hizo una descripción detallada del método de desenvolvimiento de fase por medio del uso de dos longitudes de onda. Para dicho método es necesario hacer nuevos cálculos para el vector sensitividad y el factor geométrico.

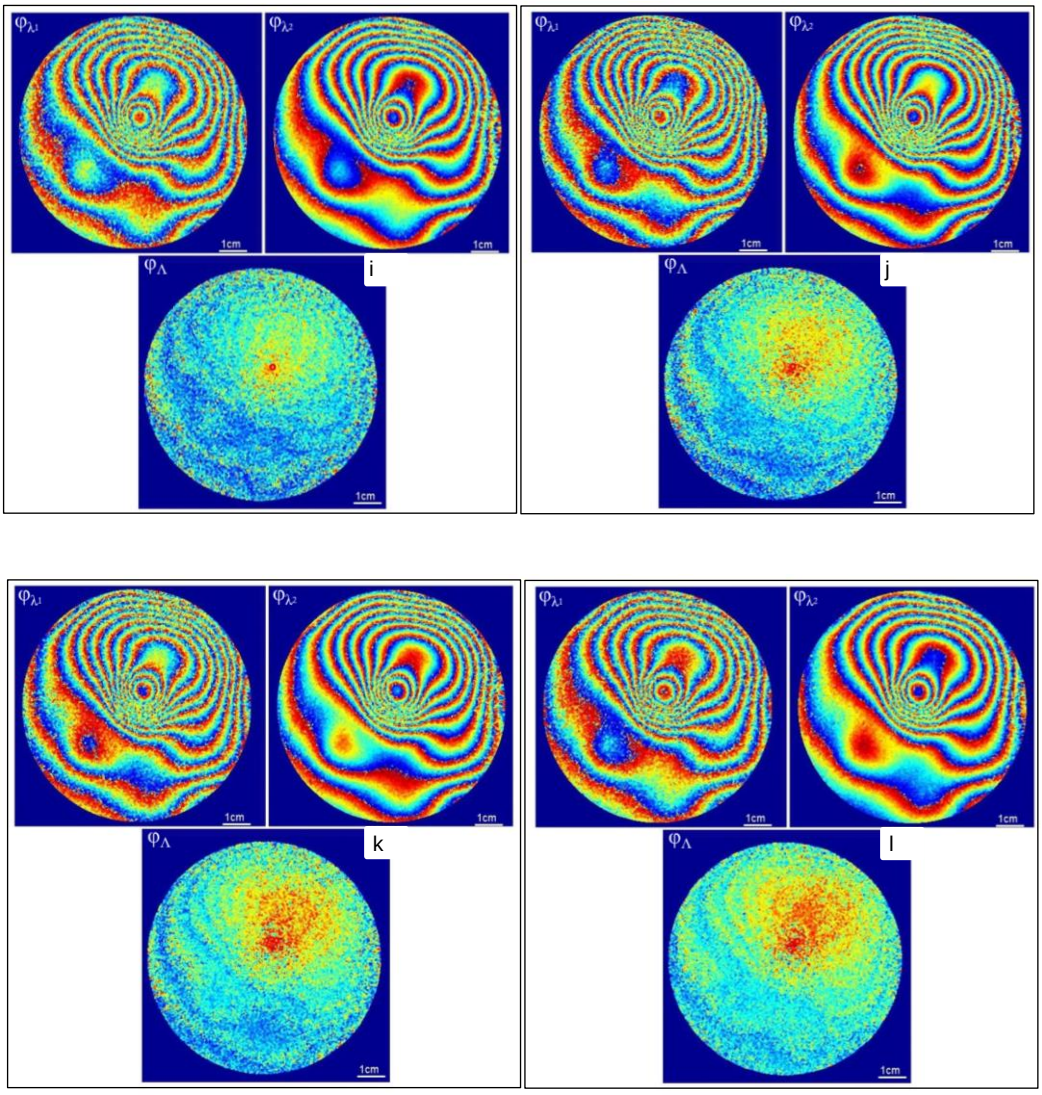
En este caso, se usó un láser de Helio-Neón centrado en una longitud de onda de $\lambda_1 = 594nm$ y un láser centrado en $\lambda_2 = 632.8nm$. De acuerdo a la ecuación (2-5), la longitud de onda efectiva estará dada por $\Lambda = 9687{,}7nm$.

Los resultados de desenvolvimiento de fase son mostrados en la Figura 4-11 donde se especifica el mapa de fase de interferencia registrado con λ_1 y λ_2. Según la ecuación (2-4) la fase es desenvuelta y se calcula el mapa de fase con la longitud de onda equivalente φ_Λ. Si se observa la Figura 4-11a, se muestra el mapa de fase desenvuelto a partir de los dos mapas de fase mostrados en las dos figuras superiores. Similarmente para los mapas de fase siguientes. Aquí, el color rojo indica un valor de $6{,}28rad$ mientras que el color azul indica $0{,}00rad$.

Figura 4-11: Desenvolvimiento de fase por medio del método de dos longitudes de onda. El color rojo indica un valor de $6{,}28rad$, que el color azul indica $0{,}00rad$. El punto rojo con amarillo representa el punto de referencia para comparación con el avance del actuador cuyos valores se presentan en la Tabla 4-2.




φλ1
φλ2
φΛ
1cm
c
d
e
f
g
h



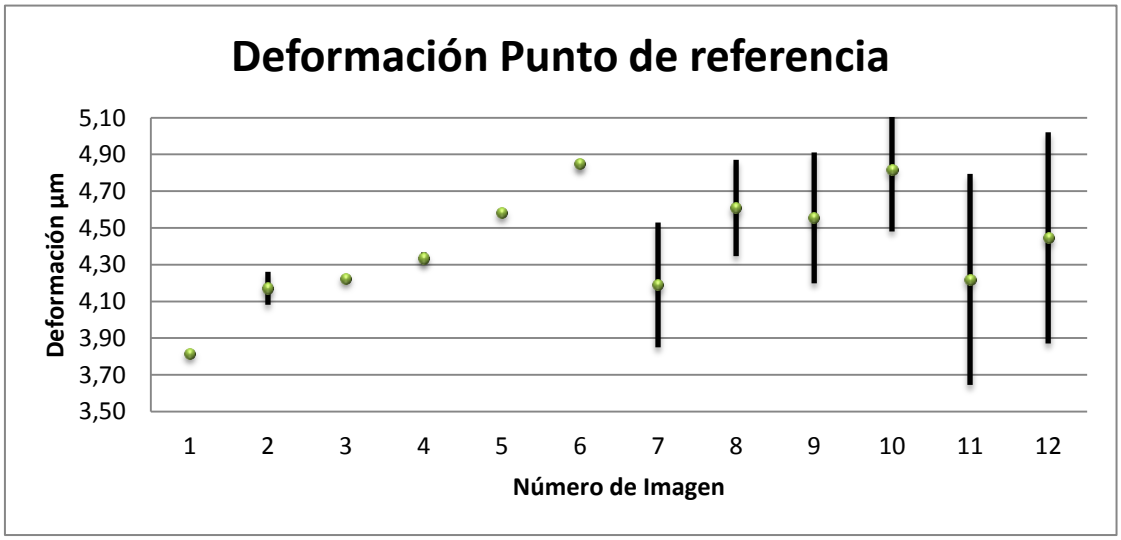
Una vez obtenido el mapa de fase desenvuelto, el factor geométrico es calculado de acuerdo a la ecuación (1-42) siendo λ reemplazada con el nuevo valor de la longitud de onda equivalente $\Lambda = 9687{,}7nm$ (ecuación (2-5)) como se mostró en la sección 2.1. Dicho valor corresponde al valor $FG_{\Lambda} = 832{,}47 \pm 0{,}09nm$.

Similarmente, el cálculo de la deformación es obtenido por medio de la ecuación (1-41) con la longitud de onda equivalente $\Lambda = 9687{,}7nm$. El factor geométrico es multiplicado por la fase de interferencia desenvuelta para obtener los valores de las micro-deformaciones introducidas en el objeto en estudio. Los resultados del cálculo de la deformación de la membrana en un punto de referencia se muestran en la Tabla 4-2 y en la Figura 4-12.

Tabla 4-2: Resultados obtenidos de la deformación de una membrana de aluminio por la técnica de IHD y el método de desenvolvimiento de fase por dos longitudes de onda.

N° IMAGEN	Avance actuador [µm]	Deformación punto de referencia [µm]	% ERROR
Figura 4-11a	3,800	3,816	0,41
Figura 4-11b	4,000	4,172	4,31
Figura 4-11c	4,200	4,226	0,62
Figura 4-11d	4,400	4,335	1,48
Figura 4-11e	4,600	4,583	0,37
Figura 4-11f	4,800	4,850	1,04
Figura 4-11g	5,000	4,190	16,21
Figura 4-11h	5,200	4,609	11,37
Figura 4-11i	5,400	4,555	15,66
Figura 4-11j	5,600	4,817	13,98
Figura 4-11k	5,800	4,220	27,24
Figura 4-11l	6,000	4,446	25,90

Figura 4-12: Curva de la deformación en un punto de referencia de la membrana y su comparación con el avance del actuador en cada una de las imágenes mostradas en la Figura 4-11: El punto 1 corresponde a la Figura 4-11a, el punto 2 corresponde a la Figura 4-11b y así sucesivamente.



Tanto en la Tabla 4-2 como en la Figura 4-12 anterior se puede notar que hay un porcentaje de error elevado para las imágenes 7, 8, 9, 10, 11 y 12. Esto puede ser debido a que la membrana, al ser de aluminio y por lo tanto no ser un material totalmente

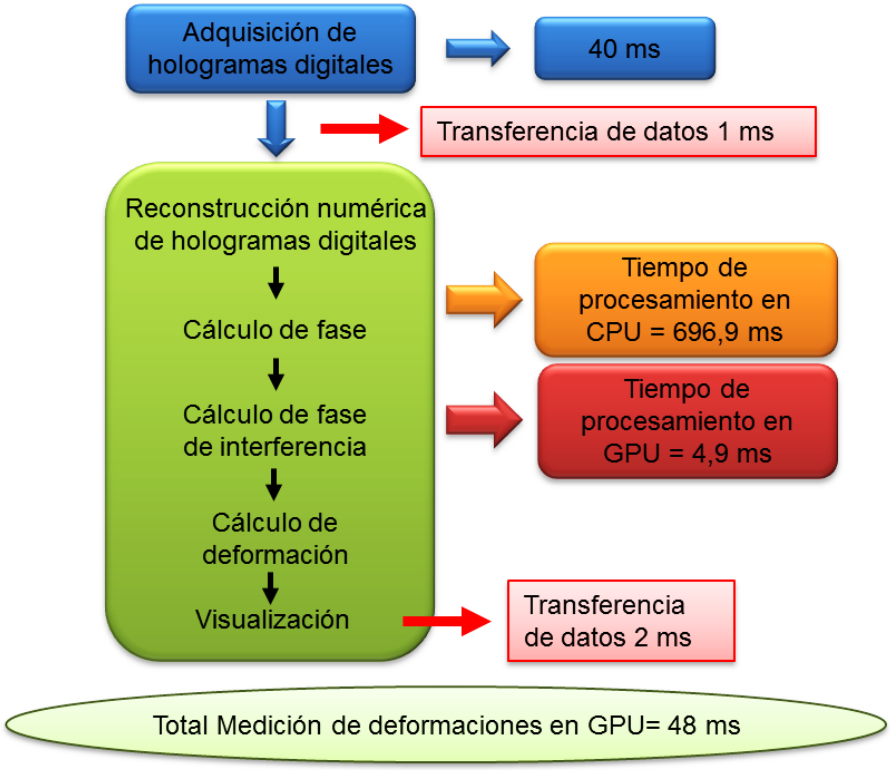
elástico, no se deforma en igual magnitud que el avance del actuador. Para la calibración real de esta herramienta, es necesario usar otro método, por ejemplo, el método de elementos finitos.

4.4. Análisis de desempeño

Como se mencionó en el capítulo 3, el hardware desarrollado para las tarjetas aceleradoras de gráficos (GPU) permite el cálculo simultáneo de operaciones lógicas. Esta ventaja que presentan las GPU respecto a las CPU hace que la programación GPGPU tenga un mejor rendimiento a la hora del cálculo de muchos datos como es el caso del procesamiento de imágenes.

En el caso particular de esta tesis de maestría, el tiempo de procesamiento del holograma digital, una vez capturado directamente de la cámara CMOS Thorlabs, la reconstrucción numérica, el cálculo de la fase de interferencia y el cómputo de la deformación toma $646{,}9ms$ en la CPU por medio de una implementación optimizada hecha en Matlab®; mientras que el mismo proceso en la GPU toma un tiempo de $4{,}9ms$ calculado con el algoritmo implementado en CUDA™. Este hecho significa que en el cálculo de la deformación o fase de interferencia, desde el momento en el que el holograma es registrado en la memoria de la CPU, el método propuesto reduce el tiempo de procesamiento unas 133 veces por holograma capturado. En la Figura 4-13 se muestra un esquema de los cálculos necesarios para el sistema de IHD y el tiempo de procesamiento ejecutado en la GPU y en la CPU.

Figura 4-13: Esquema del rendimiento del procesamiento en GPU respecto a la CPU para la reconstrucción de hologramas a velocidades de video.



5. Conclusiones y Perspectivas

5.1 Conclusiones

En esta tesis de Maestría en Ciencias - Física se ha implementado la técnica de interferometría holográfica digital (IHD) a velocidad de video. La técnica se soporta en la reconstrucción numérica de hologramas digitales y el cálculo de las diferencias de fase entre dos estados diferentes de un objeto por medio de la arquitectura CUDA™ de procesamiento en paralelo. Luego de la adquisición de los datos de los hologramas digitales, directamente de la cámara, la reconstrucción numérica es calculada por medio de la transformada discreta de Fresnel. El cómputo de las diferencias de fase es hecho con referencia a un holograma de referencia previamente registrado en un estado no deformado del objeto. Con el fin de registrar deformaciones mecánicas, se usó un objeto tipo membrana de $8cm$ de diámetro y $1{,}5mm$ de espesor. Las deformaciones sobre la membrana fueron inducidas por medio de un tornillo micrométrico motorizado (actuador) con una sensibilidad de $20nm$ operando a pasos de $33nm$.

En el rango de medición de $0{,}00nm$ - $3{,}16nm$, y con la longitud de onda dada por $\lambda_2 = 632.8nm$, las diferencias de fase son desplegadas a una razón de $48ms$ para una imagen de 1024x1024 lo que corresponde a 21 FPS. El factor geométrico del sistema interferométrico proporcionó como resultado: $54{,}64 \pm 0{,}09nm$, el cual fue calculado de acuerdo a la geometría del montaje experimental.

En el caso en el que el mapa de fase contenga discontinuidades, es decir, en el rango de medición de $3{,}16nm$ - $4{,}80\mu m$, se implementó el método de desenvolvimiento de fase por dos longitudes de onda, cuya longitud de onda equivalente está dada por $\Lambda = 9687{,}7nm$, formada a partir de $\lambda_1 = 594nm$ y $\lambda_2 = 632.8nm$. Las diferencias de fase son desplegadas a una razón de $96ms$ para una imagen de 1024x1024 lo que corresponde a

10 FPS. El factor geométrico del sistema interferométrico para la implementación con dos longitudes de onda proporcionó como resultado: $FG = 832{,}47 \pm 0{,}09nm$.

Los resultados obtenidos en esta tesis de maestría conducen a las siguientes conclusiones:

El sistema de interferometría holográfica digital permite la medición de deformaciones en el orden de nanómetros lo cual aporta información de las propiedades mecánicas de un material y permite además el monitoreo de su integridad estructural. Esta información es útil al momento de la caracterización de materiales.

La IHD es una técnica que requiere un gran esfuerzo computacional en arquitecturas tradicionales, de allí el interés en el uso de arquitecturas de cálculo en paralelo que permite el tratamiento en simultáneo de múltiple conjunto de datos. La implementación de un algoritmo propio en CUDA™ que permite la reconstrucción numérica de hologramas digitales, cálculo de fase de interferencia con su posterior cálculo de deformación a velocidades de video, hace que esta técnica sea atractiva para muchas aplicaciones que requieren de monitoreo instantáneo de parámetros físicos.

La cuantificación de las micro-deformaciones por medio del sistema de IHD alcanzan un amplio rango. En la metodología propuesta en esta tesis de maestría, la cuantificación de deformaciones, ocurren en un rango de $0{,}00 \pm 0{,}09nm$ a $4800{,}00 \pm 0{,}0nm$. La aplicación de esta metodología en un mayor rango requiere de una estrategia de desenvolvimiento de fase distinta a la implementada en esta tesis de maestría, lo cual aumentan el costo computacional.

La técnica de IHD a velocidades de video desarrollada en esta tesis de maestría es la primera propuesta presentada para obtener fases de interferencia y micro-deformaciones en tiempo real.

5.2 Perspectivas

Dentro del desarrollo de esta tesis de maestría se identificaron varias perspectivas encaminadas al mejoramiento de la implementación de la IHD a velocidades de video, entre ellas tenemos:

El desempeño del sistema interferométrico implementado puede ser mejorado por medio de algoritmos (operando a velocidad de video) que permitan optimizar el rendimiento de la reconstrucción de los hologramas digitales. Esta mejora puede hacerse por medio del tratamiento del ruido de speckle inherente a la técnica de holografía digital. Esta implementación aumentaría el contraste espacial del método de IHD para la cuantificación de deformaciones.

La rapidez del método implementado puede ser mejorada mediante la incorporación de una cámara que permita tener una adquisición a velocidad mayor de 25 FPS ($40ms$) puesto que el procesamiento de los hologramas en arquitectura paralela consume solo $4,9ms$.

La cuantificación de deformaciones mayores a $4,8\mu m$ puede efectuarse por medio de la implementación de algoritmos de desenvolvimiento de fase para la eliminación de ambigüedades. Esta implementación puede basarse en algoritmos conocidos en la literatura como Goldstein, guiados por el mapa de calidad, Flynn o L^p norm.

La implementación de un método de calibración, diferente al método implementado en esta tesis de maestría, es necesario para obtener una herramienta de medición más confiable.

Referencias

[1] S. Yoshida, Muchiar, I. Muhamad, R. Widiastuti, and A. Kusnowo, “Optical interferometry technique for deformation analysis,” *Opt. express* **2**(13), 516–530 (1998).

[2] F. Chen, G. M. Brown, and M. Song, “Overview of three-dimensional shape measurement with optical methods,” *Opt. eng.* **39**(1), 10–22 (2000).

[3] F. Blais, “Review of 20 years of range sensor development,” *J. Electron. Imaging* **13**(1), 231–240 (2004).

[4] D. Malacara, *Optical Shop Testing*, D. Malacara, Ed., p. 773, John Wiley & Sons, Inc, New Jersey (2007).

[5] K. J. Gasvik, *Optical Metrology*, Third Edit, p. 360, John Wiley & Sons, Inc, Trondheim, Norway (2002).

[6] U. Schnars and W. Jüepner, “Direct recording of holograms by a CCD-target and numerical reconstruction,” *Appl. Opt.* **33**(2), 179–181 (1994).

[7] U. Schnars, “Direct phase determination in hologram interferometry with use of digitally recorded holograms,” *J. Opt. Soc. Am. A* **11**(7), 2011–2015 (1994).

[8] C. Shakher, “Interferometric methods to measure temperature and temperature profile of gaseous flames,” in *International Conference on Optics and Photonics*, p. 7, Chandigarh (2009).

[9] D. L. Reuss and P. H. Schultz, “Interferometric temperature measurements of a flame in a cylindrical tube using holography,” *Appl. Opt.* **26**(9), 1661–1667 (1987).

[10] J.-M. Desse, P. Picart, and P. Tankam, “Digital three-color holographic interferometry for flow analysis,” *Opt. Express* **16**(8), 5471–5480 (2008).

[11] A. K. Stetson and R. L. Powell, “Interferometric Hologram Evaluation and Real-Time Vibration Analysis of Diffuse Objects,” *J. Opt. Soc. Am. A* **55**, 1694–1695 (1965) [doi:10.1364/JOSA.55.001694].

[12] D. W. Sweeney and C. M. Vest, “Measurement of three-dimensional temperature field above heated surfaces by holographic interferometry,” *Int. J. Heat Mass Transfer.* **17**, 1443–1454 (1974).

[13] C. A. Trujillo, J. F. Restrepo, and J. García-Sucerquia, "Real-time numerical reconstruction of digitally recorded holograms," *SPIE Proc.* **8011**(80116T-8), 80116T–1–80116T–9 (2011).

[14] F. Baron and B. Kloppenborg, "GPU-accelerated image reconstruction for optical and infrared interferometry," *Society of Photo-Optical …* **7734**, W. C. Danchi, F. Delplancke, and J. K. Rajagopal, Eds., 77344D–77344D–9 (2010) [doi:10.1117/12.856713].

[15] J. Wang, K. Wang, S. Zhao, L. Lin, and J. Bai, "White light interferometry for fast areal surface measurement based on GPGPU," *Proc. of SPIE Optical Design and Testing IV* **7849**(18), 1–8 (2010).

[16] X. Jiang, "In situ real-time measurement for micro-structured surfaces," *CIRP Annals - Manufacturing Technology* **60**(1), 563–566 (2011) [doi:10.1016/j.cirp.2011.03.074].

[17] NVIDIA, "NVIDIA CUDA™: NVIDIA CUDA C Programming Guide 3.1.1," 1–175 (2010).

[18] D. Gabor, "Microscopy by reconstructed wave-fronts," *P ROY SOC LOND A MAT* **197**(1051), 454–487 (1949).

[19] D. Gabor, "Microscopy by Reconstructed Wave Fronts: II," *Proc. Phys. Soc. B 64 449* **64**(6) (1951).

[20] L. Xu, J. Miao, and A. Asundi, "Properties of digital holography based on in-line configuration," *Opt. eng.* **39**(12), 3214–3219 (2000).

[21] J. W. Goodman, *Introduction to Fourier Optics*, Second Edi, L. Cox and J. M. Morriss, Eds., pp. 22–27 (441), San Francisco, CA (1996).

[22] T. Kreis, W. Jüepner, and J. Geldmacher, "Principles of digital holographic interferometry," *Proc. SPIE* **3478**, 45–54 (1998).

[23] L. Xu, X. Peng, Z. Guo, J. Miao, and A. Asundi, "Imaging analysis of digital holography," *Opt. Express* **13**(7), 2444–2452 (2005).

[24] N. Verrier and M. Atlan, "Off-axis digital hologram reconstruction: some practical considerations," *Appl. Opt.* **50**(34), H136–H146 (211AD).

[25] J. Wu, F. L. Lue, Y. C. Dong, M. Zheng, M. Huang, and Y. N. Wu, "Zero-order noise suppression with various space-shifting manipulations of reconstructed images in digital holography," *Appl. Opt.* **50**(34), 56–61 (2011).

[26] T. M. Kreis and W. P. O. Jüpner, “Suppression of the dc term in digital holography,” *Opt. eng.* **36**(8), 2357–2360 (1997).

[27] U. Schnars, T. M. Kreis, and W. P. O. Jüpner, “CCD recording and numerical reconstruction of holograms and holographic interferograms,” *Proc. interferometry VII* **2544**, 57–63 (1995).

[28] U. Schnars, T. M. Kreis, and W. P. O. Jüpner, “Digital recording and numerical reconstruction of holograms: Reduction of the spatial frequency spectrum,” *Opt. eng.* **35**(4), 977–982 (1996).

[29] T. Kreis, *Handbook of Holographic Interferometry. Optical and Digital Methods*, pp. 185–221, WILEY-VCH GmbH & Co., Weinheim (2005).

[30] J. I. García-Sucerquia, R. Castañeda, F. F. Medina, and M. Giorgio, “Distinguishing between Fraunhofer and Fresnel diffraction by the Young’s experiment,” *Opt. commun.* **200**, 15–22 (2001).

[31] M. A. Kronrod, N. S. Merzlyakov, and L. P. Yaroslavski, “Reconstruction of holograms with a computer,” *Sov. Phys. Tech. Phys.* **17**, 333–334 (1972).

[32] T. Kreis and W. Jüepner, “Principles of digital holography,” *Proc. 3rd. Workshop on Automatic Processing of Fringes Patterns*, 353–363 (1997).

[33] O. K. ERSOY, *Diffraction, Fourier Optics and Imaging*, First edit, p. 413, John Wiley & Sons, Inc, New Jersey (2007).

[34] J. W. Goodman and R. W. Lawrence, “Digital image formation from electronically detected holograms,” *Appl Phys Lett* **11**(3), 77–79 (1967).

[35] U. Schnars, “Direct phase determination in holograms interferometry with use of digitally recorded holograms,” *J. Opt. Soc. Am. A* **11**, 2011–2015 (1994).

[36] T. M. Kreis, M. Adams, and W. P. O. Jüeptner, “Methods of digital holography: a comparison,” *Proc. SPIE 3098, Optical Inspection and Micromeasurements II* **224**, 224–233 (1997) [doi:10.1117/12.281164].

[37] J. E. Sollid, “Holographic Interferometry Applied to Measurements of Small Static Displacements of Diffusely Reflecting Surfaces,” *Appl. Opt.* **8**(8), 1587–1595 (1969).

[38] G. J. Lora, N. Munera, and J. I. García-Sucerquia, “Modelling and reconstruction of gabor-type holograms,” *Dyna* **166**, 81–88 (2011).

[39] D. Mendlovic, Z. Zalevsky, and N. Konforti, “Computation considerations and fast algorithms for calculating the diffraction integral,” *J. mod. optic* **44**(2), 407–414 (1997).

[40] K. Creath, “Phase-measurement interferometric thechnique,” *Progress in Optics* **26**, 349–393 (1988) [doi:10.1016/S0079-6638(08)70178-1].

[41] K. Creath, “Phase-shifting speckle interferometry,” *Appl. Opt.* **24**(18), 3053–3058 (1985).

[42] P. Hariharan, *Optical Holography. Principles, techniques, and applications*, Second edi, P. L. Knight and A. Miller, Eds., p. 406, Cambridge University Press, New York (1996).

[43] R. M. Goldstein, H. A. Zebker, and C. L. Werner, “Satellite radar interferometry: Two-dimensional phase unwrapping,” *Radio Science* **23**(4), 713 (1988) [doi:10.1029/RS023i004p00713].

[44] T. J. Flynn, “Two-dimensional phase unwrapping with minimum weighted discontinuity,” *Journal of the Optical Society of America A* **14**(10), 2692, OSA (1997) [doi:10.1364/JOSAA.14.002692].

[45] D. C. Ghiglia and M. C. Pritt, *Two dimensional phase unwrapping. Theory, Algorithms, and Software*, p. 493, John Wiley & Sons, Inc, New York (1998).

[46] M. Servin, J. L. Marroquin, D. Malacara, and F. J. Cuevas, “Phase unwrapping with a regularized phase-tracking system.,” *Applied Optics* **37**(10), 1917–1923, OSA (1998).

[47] D. C. Ghiglia and M. C. Pritt, *Two-Dimensional Phase Unwrapping. Theory, Algorithms and Software*, primera, pp. 59–177, John Wiley & Sons, Inc, New York (1998).

[48] K. Itoh, “Analysis of the phase unwrapping algorithm,” *Applied Optics* **21**(14), 2470, OSA (1982) [doi:10.1364/AO.21.002470].

[49] J. M. Bioucas-Dias and G. Valadao, “Phase Unwrapping via Graph Cuts,” *IEEE Transactions on Image Processing* **16**(3), 698–709 (2007) [doi:10.1109/TIP.2006.888351].

[50] C. Wagner, S. Seebacher, W. Osten, and W. Jüptner, “Digital Recording and Numerical Reconstruction of Lensless Fourier Holograms in Optical Metrology,” *Applied Optics* **38**(22), 4812, OSA (1999) [doi:10.1364/AO.38.004812].

[51] U. Schnars and W. P. O. Jüeptner, “Digital recording and numerical reconstruction of holograms,” *Meas.Sci.Technol.* **13**, R85–R101 (2002).

[52] P. Ferraro, S. De Nicola, G. Coppola, A. Finizio, D. Alfieri, and G. Pierattini, “Controlling image size as a function of distance and wavelength in Fresnel-

transform reconstruction of digital holograms," *Optics Letters* **29**(8), 854, OSA (2004) [doi:10.1364/OL.29.000854].

[53] I. Yamaguchi, T. Matsumura, and J. Kato, "Phase-shifting color digital holography," *Optics Letters* **27**(13), 1108, OSA (2002) [doi:10.1364/OL.27.001108].

[54] N. Demoli, D. Vukicevic, and M. Torzynski, "Dynamic digital holographic interferometry with three wavelengths," *Optics Express* **11**(7), 767, OSA (2003) [doi:10.1364/OE.11.000767].

[55] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU Computing," *Proceedings of the IEEE* **96**(5), 879–899 (2008) [doi:10.1109/JPROC.2008.917757].

[56] L. R. Harriott, "Limits of lithography," *Proceedings of the IEEE* **89**(3), 366–374 (2001) [doi:10.1109/5.915379].

[57] NVIDIA, "Popular GPU-Accelerated applications," 2012, <http://www.nvidia.com/object/gpu-applications.html>.

[58] C. A. Trujillo, "Seguimiento 4D de microorganismos con microscopía holográfica digital sin lentes a velocidad de video por medio de GPGPU.," p. 86, Universidad Nacional de Colombia Sede Medellin (2012).

[59] NVIDIA, "cuBLAS Library User Guide," 2012, <http://docs.nvidia.com/cuda/pdf/CUDA_CUBLAS_Users_Guide.pdf>.

[60] NVIDIA, "CUDA FFT Library User Guide," 2010, <http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUFFT_Library.pdf>.

[61] M. Frigo and S. G. Johnson, "FFTW," 2012, <http://www.fftw.org/>.

[62] M. Segal and K. Akeley, "The OpenGL® Graphics System: A Specification," *The Khronos Group Inc*, 2008, <http://www.opengl.org/registry/doc/glspec30.20080811.pdf>.

[63] M. Frigo and S. G. Johnson, "FFTW: An adaptive software architecture for the FFT," *Proce. of the 1998 IEEE Inter. Confe. on Acous., Spe. and Sign. Proce.*, 1381–1384 (1998).

[64] S. Christian, "ANALYSIS OF CONTRAST ENHANCEMENT METHODS FOR INFRARED IMAGES," p. 99, California Polytechnic State University (2011).