UAEM | Universidad Autónoma del Estado de México

CENTRO UNIVERSITARIO XICO VALLE DE CHALCO

C.U. Valle de Chalco

OPTIMIZACIÓN DE LAS TÉCNICAS DE DESENVOLVIMIENTO DE FASE EN DOS DIMENSIONES

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

ABEL LÓPEZ OCAÑA

TUTOR ACADÉMICO
DR. WILLIAM DE LA CRUZ DE LOS SANTOS

TUTOR ADJUNTO
DR. JUVENAL RUEDA PAZ
TUTORA ADJUNTA
DRA. MARÍA DE LOURDES LÓPEZ GARCÍA

VALLE DE CHALCO SOLIDARIDAD, MÉXICO OCTUBRE 2017.

Universidad Autónoma del Estado de México
Centro Universitario Valle de Chalco

Valle de Chalco Solidaridad, Edo de Méx. a lunes, 23 de octubre de 2017

DR. EN C. JUVENAL RUEDA PAZ
COORDINADOR DE LA MAESTRÍA CIENCIAS DE LA COMPUTACIÓN
DEL CENTRO UNIVERSITARIO *UAEM* VALLE DE CHALCO.

P R E S E N T E.

Por este medio le comunico a usted que la comisión revisora designada para realizar la tesis denominada: **"Optimización de las técnicas de desenvolvimiento de fase en dos dimensiones"**, como parte de los requisitos para obtener el grado académico de Maestría en **Ciencias de la Computación** presenta **Abel López Ocaña**, con número de cuenta **0824818** para sustentar el acto de evaluación de grado, ha dictaminado que dicho trabajo reúne las características de contenido para proceder a la impresión del mismo

A T E N T A M E N T E

Tutor adjunto	Tutor Académico	Tutor Adjunto
Dr. Juvenal Rueda Paz	**Dr. William de la Cruz de los Santos**	**Dra. María de Lourdes López García**

CENTRO UNIVERSITARIO XICO VALLE DE CHALCO

Av. Hermenegildo Galeana Num 3
Col. María Isabel CP.56615
Tel. 55) 59714940 Ext.115
http://titulacioncuvalledechalco.weebly.com/

CUVCH

Oficio CoordMACSCO 029/2017
Valle de Chalco Solidaridad, Edo. de México a 23 de octubre del 2017.

Ing. en Comp. Abe Lòpez Ocaña.
Candidato al Grado de Maestría en Ciencias de la Computación
Centro Universitario UAEM Valle de Chalco
Presente

De acuerdo con el Reglamento de Estudios Avanzados de la Universidad Autónoma del Estado de México y habiendo cumplido con todas la indicaciones que la Comisión Revisora realizó con respecto a su trabajo de tesis titulado **" Optimización de las técnicas de desenvolvimiento de fase en dos dimensiones"**, la Coordinación de la Maestría en Ciencias de la Computación del Centro Universitario UAEM Valle de Chalco, concede la autorización para que proceda a la impresión de la misma.

Sin más por el momento, le reitero la seguridad de mi especial consideración y estima.

PATRIA, CIENCIA Y TRABAJO

"2017, Año del centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

CENTRO UNIVERSITARIO UAEM
VALLE DE CHALCO
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

Dr. Juvenal Rueda Paz
Coordinador de la Maestría
en Ciencias de la Computación
Centro Universitario UAEM Valle de Chalco

CARTA DE CESIÓN DE DERECHOS DE AUTOR

El que suscribe **Abel López Ocaña** Autor del trabajo escrito de evaluación profesional en la opción de Tesis con el título Optimización de las técnicas de desenvolvimiento de fase en dos dimensiones, por medio de la presente con fundamento en lo dispuesto en los artículos 5, 18, 24, 25, 27, 30, 32 y 148 de la Ley Federal de Derechos de Autor, así como los artículos 35 y 36 fracción II de la Ley de la Universidad Autónoma del Estado de México; manifiesto mi autoría y originalidad de la obra mencionada que se presentó en el **Centro Universitario UAEM Valle de Chalco** para ser evaluada con el fin de obtener el Grado de Maestro en Ciencias de la Computación. Así mismo expreso mi conformidad de ceder los derechos de reproducción, difusión y circulación de esta obra, en forma NO EXCLUSIVA, a la Universidad Autónoma del Estado de México; se podrá realizar a nivel nacional e internacional, de manera parcial o total a través de cualquier medio de información que sea susceptible para ello, en una o varias ocasiones, así como en cualquier soporte documental, todo ello siempre y cuando sus fines sean académicos, humanísticos, tecnológicos, históricos, artísticos, sociales, científicos u otra manifestación de la cultura. Entendiendo que dicha cesión no genera obligación alguna para la Universidad Autónoma del Estado de México y que podrá o no ejercer los derechos cedidos.

Por lo que el autor da su consentimiento para la publicación de su trabajo escrito de evaluación profesional.

Se firma la presente en la ciudad de Valle de Chalco, a los 23 días del mes de octubre de 2017.

Abel López Ocaña

Nombre y firma de conformidad

A mis padres y hermanos que me apoyaron en todos mis estudios y siempre me motivaron a seguir mis ideales.

Agradecimientos

Agradezco a:

Mis padres y hermanos que me brindaron su apoyo y afecto en mi proceso de formación profesional, sin ellos esto no hubiera sido posible.

La Universidad Aut0noma del Estado de México, Centro universitario UAEM valle de Chalco por la oportunidad de cursar la maestría y por facilitar el material e infraestructura necesaria, así, como a todo el personal docente, administrativo y de mantenimiento que labora en ella.

Consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca otorgada que hizo posible que no descuidara mis estudios.

En especial quiero agradecer a mi asesor el doctor William de la Cruz de los Santos quien me brindo su tiempo, conocimiento, experiencia y paciencia de manera incondicional.

A mis asesores adjuntos los doctores María de Lourdes López García y Juvenal Rueda Paz por el tiempo y apoyo brindado.

Por último pero no menos importante quiero agradecer a mis compañeros quienes compartieron conmigo esta experiencia.

Resumen

El desenvolvimiento de fase es un procedimiento clave presente en tecnologías como radar de apertura sintética (SAR), sonar de apertura sintética (SAS), máquinas de resonancia magnética (MRI) y proyección de patrones de franjas por mencionar algunos. De manera general, el problema de desenvolvimiento de fase consiste en eliminar las discontinuidades en el mapa de fase sumando múltiplos de 2π. La recuperación de la fase real puede verse afectada por factores como ruido, bajo muestreo, cambios abruptos de fase y errores de calibración en el equipo de medición. Debido a estos factores, la recuperación de la fase se convierte en un problema computacionalmente difícil por lo que sólo se puede obtener una aproximación a la fase real.

Existen varios algoritmos para el problema de desenvolvimiento de fase que se pueden clasificar como espaciales y temporales. Los algoritmos espaciales procesan pixel a pixel el mapa de fase siguiendo una ruta de integración y los algoritmos temporales utilizan una secuencia de mapas de fase obtenidos en diferentes tiempos para obtener la fase real. Entre los métodos espaciales más robustos y eficaces se tienen el algoritmo de Goldstein y el algoritmo guiado por calidad. En esta investigación se propone una estrategia de partición por bloques del mapa de fase para acelerar algoritmo de Goldstein usando programación paralela. Además, se propone hacer uso del concepto de residuos utilizado en el algoritmo de Goldstein para mejorar la eficacia del algoritmo de desenvolvimiento de fase guiado por calidad.

Abstract

The phase unwrapping is an important process into technologies such as Synthetic Aperture Radar (SAR), Sonar Aperture Radar, Magnetic Resonance Images (MRI) and Digital Projection Patterns (DPP), among others. In general, the phase unwrapping problem consists in removing the 2π discontinuities in a wrapped phase map. The problem of retrieve the unwrapped phase is affected by several factors such as the presence of noise, phase discontinuities, under sampling and calibration errors. These factors are the main reason that the unwrapping is a difficult problem and only approximation results can be obtained.

There exists several algorithms to the unwrapping problem that can be classified into spatial and temporal. The spatial algorithms process pixel-by-pixel the wrapped phase following an integration path and the temporal algorithms use a sequence of wrapped phase maps obtained in discrete times to recovery the unwrapped phase map. Among the most robust spatial algorithms are the Goldstein algorithm and the Quality guide algorithm. In this thesis, a partition strategy to speed up the Goldstein algorithm using parallel programming, it is proposed. Also, it is used the concept of residues into the quality guide algorithm to enhance the reliability of the quality maps based on maximum gradients.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se introduce al problema del desenvolvimiento de fase en una y dos dimensiones. También se establecen los objetivos, justificación de la investigación, contexto de la tesis y la organización de la tesis.

1.1. Planteamiento del problema

El problema de desenvolvimiento de fase consiste en recuperar los valores originales a partir de los valores de la fase envuelta [10, 12, 22], es decir, se obtiene un mapa de fase continua a partir de una discontinua. Este problema surge en las aplicaciones que utilizan la fase como un indicador indirecto de una cantidad física como el retardo de tiempo del cual se puede inferir otra información [4]. Dado que la fase es observable sólo en un espacio circular repetitivo, las mediciones derivadas de la fase se envuelven con respecto a algún módulo o ambigüedad que es el equivalente físico de 2π rad [12, 22, 25].

Formalmente el problema puede ser planteado como

$$\psi(t) = \varphi(t) + 2\pi k(t) \tag{1.1}$$

donde $\psi(t)$ es la fase envuelta (*wrapped*) conocida, $\varphi(t)$ es la fase ideal (*unwrap-*

ped) que se desea conocer y $k(t)$ es una función definida en los enteros.

Recuperar la fase consiste en sumar multiplos de 2π en el mapa de fase envuelto, según sea necesario. Sin embrago, el problema de desenvolvimiento de fase se vuelve una tarea difícil cuando el mapa envuelto se encuentra corrompido por las siguientes fuentes de error:

- Presencia de ruido no deseado.
- Una tasa de muestreo que se encuentre por debajo de la condición Nyquis, la cual asume que una señal analógica puede ser reconstruida sin error siempre que la razón de muestreo sea igual o mayor al doble de ancho de banda de la señal analógica.
- Cambios de fase abrubtos.
- Errores, al calibrar los aparatos de medición.

La presencia de cualquiera de las fuentes de error antes mencionadas hace del problema de desenvolvimiento de fase, un problema computacionalmente difícil [21].

1.2. Objetivos

1.2.1. Objetivo general

Mejorar la eficiencia en tiempo de ejecución y eficacia de los métodos de desenvolvimiento de fase espaciales como el algoritmo de Goldstein y el algoritmo guiado por calidad.

1.2.2. Objetivos específicos

A continuación se presentan los puntos importantes de esta investigación, los cuales ayudaran a llevar en orden el trabajo:

- Estudiar el problema del desenvolvimiento de fase en una y dos dimensiones.
- Investigar los algoritmos espaciales basados en rutas de integración existentes para resolver el problema de desenvolvimiento de fase.
- Diseñar e implementar una versión paralela del algoritmo de Goldstein usando un método de partición del mapa de fase envuelto.
- Realizar pruebas de desempeño y evaluación experimental del algoritmo desarrollado utilizando datos sintéticos y reales.
- Mejorar la eficacia del algoritmo de desenvolvimiento de fase guiado por calidad usando el concepto de residuos.

1.3. Justificación de la investigación

El desenvolvimiento de fase es utilizado en distintas aplicaciones como en el radar de apertura sintética (SAR), en el radar interferométrico de apertura sintética (InSAR), en las imágenes por resonancia magnética (MRI), en el sonar de apertura sintética (SAS), en la perfilometría de luz estructurada, en la medición de la distorsión del frente de onda en la óptica adaptativa, y en el perfilado de piezas mecánicas por rayos X, por mencionar algunas [10, 13, 12, 26, 21, 2, 4, 14]. Sin duda, el desenvolvimiento de fase es un problema importante que se encuentra en muchas áreas de estudio.

Es necesario mencionar que el desenvolvimiento de fase es un problema imposible de resolver, ya que dada una matriz de fase con valores desenvueltos contiene información que no está disponible en la matriz con valores de fase envueltos, es decir, dada una matriz de fase envuelta ambigua, no hay manera definitiva de determinar cuál de las muchas posibles soluciones no envueltas sin ambigüedades es correcta. La mayoría de los algoritmos de desenvolvimiento de fase se basan en la suposición de que la velocidad de muestreo espacial es lo suficien-

temente alta en la mayoría de las partes de la señal; evitando así, el efecto de *aliasing* que es un fenómeno propio de la conversión A/D, en el cual, la frecuencia de la señal reconstruida es menor que el de la señal original, esto ocurre cuando la frecuencia de muestreo es demasiado baja.

1.4. Contexto y delimitación de la tesis

En este trabajo de tesis nos limitaremos al estudio de algoritmos espaciales ya que son los métodos más robustos, mientras que los algoritmos temporales requieren de condiciones específicas para obtener los mapas de fase. Los algoritmos espaciales también se pueden agrupar en tres clases: métodos basados en rutas de integración (*Path-Following*), métodos de mínimos cuadrados (*Least-Squares*) y métodos de norma mínima (*Minimun-Norm*).

Los algoritmos basados en rutas de integración se pueden subclasificar en tres grupos: métodos dependientes de la trayectoria, métodos de compensación residual y métodos guiados por calidad [25, 13]. Los métodos dependientes de la trayectoria detectan la posición de los bordes o pasos abruptos de fases en una imagen y usan esta información para calcular los desplazamientos de fase en curso, los algoritmos de compensación residual buscan residuos en una imagen y generan cortes entre residuos positivos y negativos o traza cortes entre un residuo y borde de la imagen. Los cortes evitan que los errores que se encuentran en el mapa de fase envuelto se propaguen en el proceso de desenvolvimiento.

Los algoritmos guiados por calidad resuelven primero los píxeles de mayor calidad con los valores de mayor fiabilidad y después los píxeles de menor calidad que tengan los valores de fiabilidad más bajos para evitar la propagación de errores. La trayectoria de desenvolvimiento se determina mediante el uso de la fiabilidad de los píxeles. Aunque algunos errores pueden permanecer sin ser detectados y propagarse, de manera que depende de la ruta de desenvolvimiento. Los algoritmos de seguimiento de ruta son generalmente eficientes en tiempo de

ejecución. Entre los algoritmos basados en rutas de integración se encuentran el algoritmo Goldstein, algoritmos guiados por mapas de calidad y métodos basados en máscaras [25, 2].

El desenvolvimiento de fase por mínimos cuadrados es una de las técnicas más robustas para resolver el problema de desenvolvimiento de fase bidimensional. Este método obtiene una solución minimizando las diferencias de fase entre las derivadas parciales de los datos de fase envuelta y las aproximaciones de la fase desenvuelta. Los algoritmos basados en mínimos cuadrados se dividen en ponderados y no-ponderados. Los métodos más utilizados para el desenvolvimiento de fase de mínimos cuadrados son métodos iterativos y directos tales como el método de optimización real de Jacobi y el método de Gauss-Seidel [10].

Los métodos de la norma mínima formulan el problema de desenvolvimiento de fase en un sentido generalizado de norma mínima. Buscan una función de fase cuyos gradientes de fase no envueltos son independientes de la trayectoria y lo más cerca posible de los gradientes de fase envueltos medidos. Si se utiliza la norma L2 o euclidiana para medir el error de ajuste, el problema de optimización tiene una solución analítica que está dada por la ecuación de Poisson con las condiciones límite de Neumann [10, 26]. El método de los mínimos cuadrados no ponderados puede implementarse eficientemente mediante la transformada rápida de Fourier (FFT), la transformación discreta coseno (DCT) o los métodos multigrid. Cuando se dispone de un mapa de calidad, la función de error se puede ponderar píxel por píxel por el mapa de calidad para reducir los efectos del ruido de fondo y los píxeles poco fiables. El problema de los mínimos cuadrados ponderados se resuelve generalmente iterando el método no ponderado, y por lo tanto, es más lento que la contraparte no ponderada.

1.5. Organización de la tesis

En el capítulo 2 se estudia la importancia de la fase en una señal, así como el proceso de envolvimiento para comprender mejor el problema, posteriormente se estudia el desenvolvimiento de fase en una dimensión analizando el funcionamiento del algoritmo de Itoh como un ejemplo práctico; además se listan las tecnologías que generan señales de fase envuelta y se presentan las aportaciones de otros investigadores sobre el tema.

En el capítulo 3 se detalla el funcionamiento del algoritmo de Goldstein con el fin de presentar una versión paralela, y se concluye con la experimentación y resultados del algoritmo propuesto. La experimentación contempla mapas de fase reales y sintéticos.

El capítulo 4 se explica a detalle el algoritmo de desenvolvimiento de fase guiado por mapa de calidad con la finalidad de comprender su estructura y funcionamiento del algoritmo; por último se presenta una mejora en la construcción de mapas de calidad usando el concepto de residuo.

En el capítulo 5 se presentan las conclusiones y posibles lineas de investigación futuras derivadas de este trabajo.

Capítulo 2

Marco teórico y estado del arte

En este capítulo se explica la importancia de la fase en una señal de dos dimensiones, se describe el problema de desenvolvimiento de fase de una y dos dimensiones, y las principales fuentes de error que afectan el desenvolvimiento de fase. También se da una introducción de los conceptos básicos de la programación paralela y finalmente se presenta una revisión del estado del arte que proponen ideas similares a la propuesta en este trabajo de investigación.

2.1. La importancia de la fase

Para resaltar la importancia de la fase en una señal de dos dimensiones a continuación se realiza el siguiente experimento. Sean H_k y h_n las transformada discreta de Fourier e inversa dadas por

$$H_k = \sum_{n=0}^{N-1} h_n \exp[-j2\pi nk/N], \quad k = 0, \ldots, N-1$$

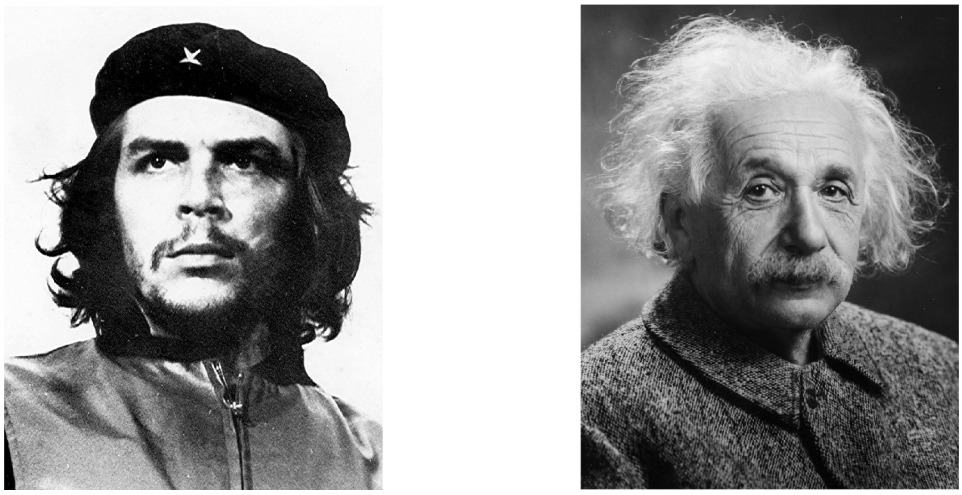
y

$$h_n = \frac{1}{N} \sum_{k=0}^{N-1} H_k \exp[j2\pi nk/N], \quad n = 0, \ldots, N-1.$$

donde las cantidades n y k denotan índices de muestras discretas en el dominio *tiempo* y *frecuencia*. Además, se tiene que

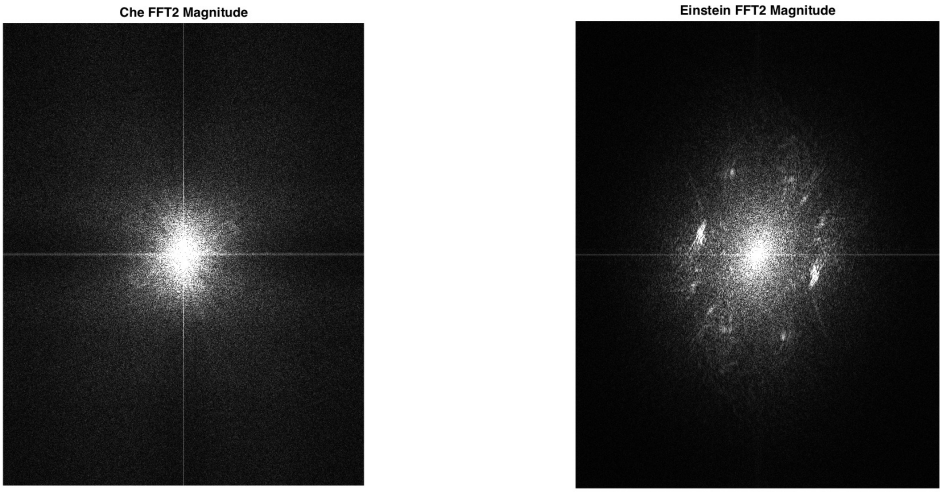
1. La cantidad h_n representa una muestra de una función continua $h(t)$ en el tiempo $t = nT$,
2. La cantidad H_k representa una aproximación del espectro continuo $H(\omega)$ en la frecuencia $\omega = \omega_k = 2\pi k/(NT)$,
3. La magnitud del espectro es $|H|$ y
4. La fase del espectro es $\psi = \arctan[\mathcal{I}(H), \mathcal{R}(H)]$.

Considere las imágenes que se muestran en la Figura 2.1 a las cuales se les extrae la magnitud del espectro y la fase como se muestran en la Figuras 2.2 y 2.3 respectivamente. Si se intercambian las fases de las imágenes y se calcula la transformada inversa de Fourier, se obtienen las imágenes de la Figura 2.4, en las que se puede observar que la información de fase contiene información muy importante en el dominio espacial de las imágenes.



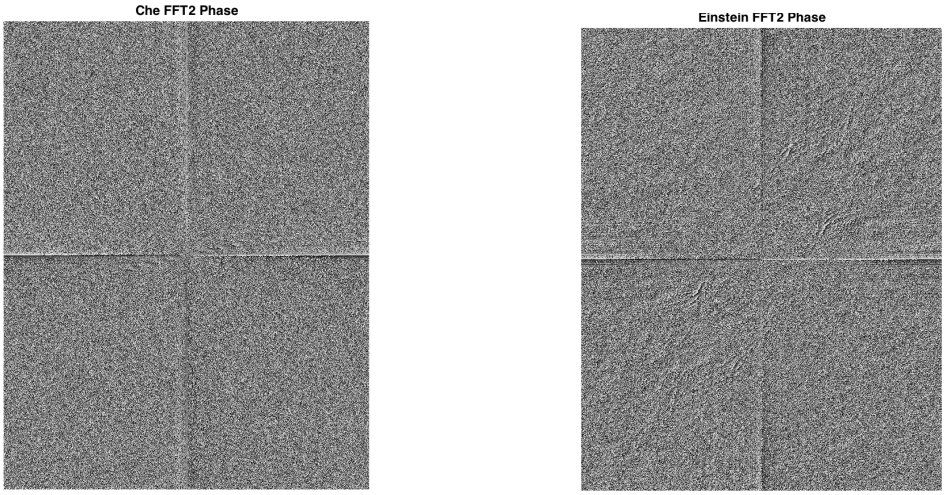
(a) (b)

Figura 2.1: Imágenes bidimencionales (a) Che Guevara y (b) Albert Einstein.



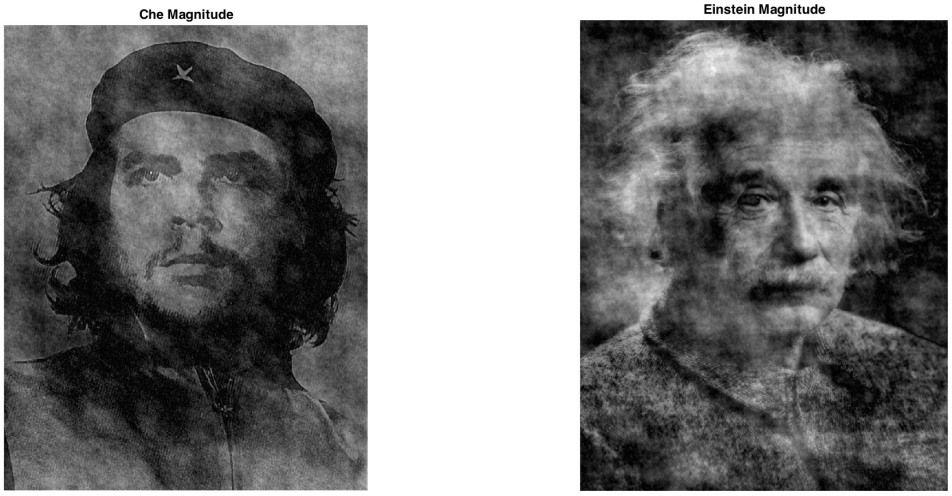
(a) (b)

Figura 2.2: Espectro bidimensional de la magnitud de la transformada de Fourier (a) Che Guevara y (b) Albert Einstein.



(a) (b)

Figura 2.3: Espectro de fase de la transformada de Fourier (a) Che Guevara y (b) Albert Einstein.



(a) (b)

Figura 2.4: Reconstrucción de imágenes por inversión de Fourier con cambios de espectros de fase.

2.2. Desenvolvimiento de fase en una dimensión

Sea $\varphi(n)$ la fase desenvuelta con $n = 0, 1, \ldots, N-1$ con la condición

$$-\pi < \Delta\{\varphi(n)\} \leq \pi. \tag{2.1}$$

donde Δ es el operador de diferencias

$$\Delta\{\varphi(n)\} = \varphi(n+1) - \varphi(n), \quad n = 0, 1, \ldots, N-2 \tag{2.2}$$

Sea también el operador $\mathcal{W}$ que permite envolver una señal en el intervalo$(-\pi, \pi]$. De este modo se tiene que

$$\psi(n) = \mathcal{W}\{\varphi(n)\} = \varphi(n) + 2\pi k(n), \quad n = 0, 1, \ldots, N-1, \tag{2.3}$$

donde $k(n) \in \mathbb{Z}$.

Aquí $\psi(n)$ representa la fase envuelta, mientras que $\varphi(n)$ es la fase desen-

vuelta. Al ser conocida la fase envuelta $\psi(n)$ el interés radica en recuperar la fase desenvuelta $\varphi(n)$. Otra forma de definir el operador de envoltura $\mathcal{W}$, es mediante funciónes trigonométricas de la forma siguiente:

$$\mathcal{W}\{\varphi(n)\} = \arctan\left[\frac{\sin\varphi(n)}{\cos\varphi(n)}\right] \tag{2.4}$$

Aplicando el operador de diferencias al operador $\mathcal{W}$ se tiene

$$\Delta\{\mathcal{W}\{\varphi(n)\}\} = \Delta\{\varphi(n)\} + 2\pi\Delta\{k_1(n)\}; \tag{2.5}$$

y ahora se aplica el operador $\mathcal{W}$ a la ecuación anterior

$$\begin{aligned}\mathcal{W}\{\Delta\{\mathcal{W}\{\varphi(n)\}\}\} &= \Delta\{\varphi(n)\} + 2\pi\Delta\{k_1(n)\} + 2\pi k_2(n)\\ &= \Delta\{\varphi(n)\} + 2\pi\left[\Delta\{k_1(n)\} + k_2(n)\right].\end{aligned} \tag{2.6}$$

Dado que

$$-\pi < \mathcal{W}\{\Delta\{\mathcal{W}\{\varphi(n)\}\}\} \leq \pi \tag{2.7}$$

y además

$$-\pi < \Delta\{\varphi(n)\} \leq \pi. \tag{2.8}$$

Entonces necesariamente

$$2\pi\left[\Delta\{k_1(n)\} + k_2(n)\right] = 0. \tag{2.9}$$

Por lo que se tiene que:

$$\mathcal{W}\{\Delta\{\mathcal{W}\{\varphi(n)\}\}\} = \Delta\{\varphi(n)\} \tag{2.10}$$

$$= \mathcal{W}\{\Delta\{\psi(n)\}\}. \tag{2.11}$$

Es decir, que se puede recuperar $\Delta\{\varphi(n)\}$ a partir de la fase envuelta. Pero se

puede expresar $\varphi(k)$ como la siguiente serie

$$\begin{aligned}\varphi(k) &= \varphi(0) + [\varphi(1) - \varphi(0)] + [\varphi(2) - \varphi(1)] + \cdots + [\varphi(k) - \varphi(k-1)] \\ &= \varphi(0) + \sum_{i=0}^{k-1} [\varphi(i+1) - \varphi(k)] \\ &= \varphi(0) + \sum_{i=0}^{k-1} \Delta\{\varphi(n)\} \\ &= \varphi(0) + \sum_{i=0}^{k-1} \mathcal{W}\{\Delta\{\psi(i)\}\}.\end{aligned} \tag{2.12}$$

2.2.1. El algoritmo de Itoh

En 1982, Itoh analizó el problema de desenvolvimiento de fase en una dimensión y demostró que la fase desenvuelta se puede obtener mediante la integración (sumando) de las diferencias de fase envueltas. Itoh desarrolló un algoritmo para envolver la fase de una señal unidimensional y recuperarla a partir de la sumatoria de las diferencias de fase envueltas, mismo que se presenta en el algoritmo 1.

Algorithm 1 Método de Itoh para el desenvolvimiento de fase de una dimensión

1: Calcular las diferencias de fase
$D(i) = \psi(i+1) - \psi(i)$
para $i = 0, \ldots, N-2$.
2: Envolver las diferencias anteriores, aplicando el operador $\mathcal{W}$, es decir:
$\Delta(i) = \arctan\{\sin D(i), \cos D(i)\}$
para $i = 0, \ldots, N-2$.
3: Inicializar el primer valor desenvuelto
$\varphi(0) = \psi(0)$.
4: Desenvolver mediante las sumas de las diferencias de la fase envuelta
$\varphi(i) = \varphi(i-1) + \Delta(i-1)$,
para $i = 1, \ldots, N-1$.

2.2.2. Fuentes de error

La mayoría de las aplicaciones del mundo real producen imágenes de fase envuelta que contienen errores. En el desenvolvimiento de fase de dos dimensiones

existen cuatro fuentes de error que complican el proceso, éstas fuentes son las siguientes:

1. *Ruido*: Se denomina ruido a toda señal no deseada que se mezcla con la señal útil que se quiere transmitir. Es el resultado de diversos tipos de perturbaciones que tiende a enmascarar la información cuando se presenta en la banda de frecuencias del espectro de la señal, es decir, dentro de su ancho de banda. El ruido se debe a múltiples causas: a los componentes electrónicos (amplificadores), al ruido térmico de los resistores, a las interferencias de señales externas, etc. Es imposible eliminar totalmente el ruido, ya que los componentes electrónicos no son perfectos, sin embargo, es posible limitar su valor de manera que la calidad de la comunicación resulte aceptable.

2. *Bajo muestreo*: El desenvolvimiento de fase de señales que se encuentran con bajo muestreo pueden ser difíciles de resolver, o en algunos casos imposibles. Esto ocurre cuando la diferencia entre dos muestras sucesivas es mayor que $+\pi$ o menor que $-\pi$. La diferencia entre estas muestras adyacentes están presentes debido al hecho de que la imagen de fase envuelta no contiene suficientes muestras y no existe una envoltura de fase real. Esta situación genera automáticamente un desenvolvimiento incorrecto.

3. *Cambios abruptos de fase*: El desenvolvimiento de fase de una señal de fase continua que contiene cambios repentinos en su fase pueden ser difíciles de desenvolver de manera correcta, incluso imposibles en la mayoría de los casos. Esto ocurre cuando los cambios de fase son mayores que $+\pi$ o menores que $-\pi$. Estos cambios de fase tan abruptos pueden parecer datos de fase envueltos reales, pero en realidad no existen, ya que estos cambios se deben a los cambios repentinos de fase.

2.2.3. Desenvolvimiento de fase en dos dimensiones

Para resolver el problema de desenvolvimiento de fase en dos dimensiones, es posible extender algunas ideas del desenvolvimiento en una dimensión. En particular, el proceso de integración para la obtención de la fase desenvuelta (base del método de Itoh) no es posible extenderlo a dos dimensiones, la razón radica en que el proceso de integración en dos dimensiones no es independiente del camino de integración. Por otro lado, el problema del desenvolvimiento de fase, es en general, un problema sin solución. No obstante, se pueden establecer algunas restricciones bajo las cuales sea posible encontrar una solución aproximada. Una de las restricciones más importantes que se establece, es que la frecuencia local en cada dirección esté en el intervalo $(-\pi, \pi]$, o lo que es lo mismo, que las derivadas (diferencias de fase envueltas) de la fase sean menores que π en magnitud.

El problema de desenvolvimiento de fase de dos dimensiones se puede plantear de la siguiente manera: se conoce la fase envuelta $\psi(x, y)$ en una retícula $\mathcal{L}$, hallar aproximadamente la fase desenvuelta $\varphi(x, y)$ en toda la retícula, bajo la restricción de que las derivadas de la fase esté en el intervalo $(-\pi, \pi]$, tal y como se plantea en la siguiente ecuación

$$\psi(x, y) = \varphi(x, y) + 2\pi k(x, y), \quad -\pi < \psi(x, y) \leq \pi \tag{2.13}$$

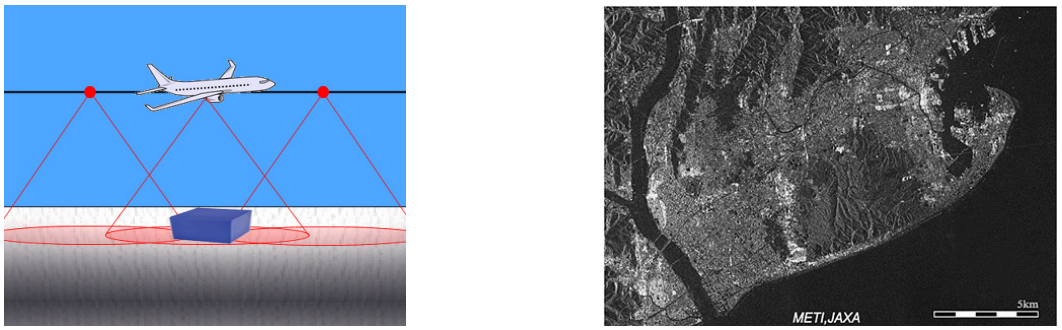
donde $k(x, y) \in \mathbb{Z}$ para todo punto $(x, y) \in \mathcal{L}$.

2.3. Tecnologías que generan mapas de fase

El desenvolvimiento de fase es un proceso que se encuentra presente en muchas tecnologías actuales; a continuación se explican las más representativas con el fin de visualizar la importancia de este proceso y la necesidad de mejorar los algoritmos de desenvolvimiento de fase.

2.3.1. Radar de apertura sintética

El radar de apertura sintética (SAR) es un sistema capaz de producir imágenes de la reflectividad a frecuencias de microondas. Las imágenes SAR se construyen a partir del procesado de la fase y la amplitud de la señal reflejada. Tradicionalmente se había explotado la información en forma de amplitud, pero en los últimos años se ha desarrollado enormemente la interferometría SAR, que es capaz de obtener información de la elevación del terreno gracias a la fase de las imágenes. La aplicación de la interferometría SAR para la generación de Modelos Digitales de Elevación (DEMs) tiene varias ventajas en comparación con la fotografía estereoscópica tradicional. La importancia y versatilidad de la técnica interferométrica se fundamenta, principalmente, en que el SAR trabaja bajo todo tipo de condiciones meteorológicas y en su autonomía respecto a las fuentes de iluminación naturales. La interferometría SAR puede generar DEMś en zonas donde todos los métodos estereoscópicos tradicionales fallan, debido a la falta de patrones identificables en la escena. Por otra parte, la obtención de estas imágenes desde el espacio se beneficia de una gran cobertura sobre la superficie terrestre y de la posibilidad de estudiar la topografía de zonas remotas de difícil acceso y observación por medios tradicionales. Además de la generación de DEMś, la interferometría SAR ha demostrado ser muy útil en el estudio de glaciares, terremotos, erosión, hidrología y vulcanología. En la Figura 2.5 se puede visualizar el esquema tradicional de un sistema SAR, además de una imagen obtenida con un sistema SAR.



(a) (b)

Figura 2.5: (a) Geometría de adquisición de datos para radar de apertura sintética (SAR), donde el objeto a visualizar es escaneado desde 3 ángulos distintos; (b) Vista nocturna alrededor de la ciudad de Shizuoka por el PALSAR.

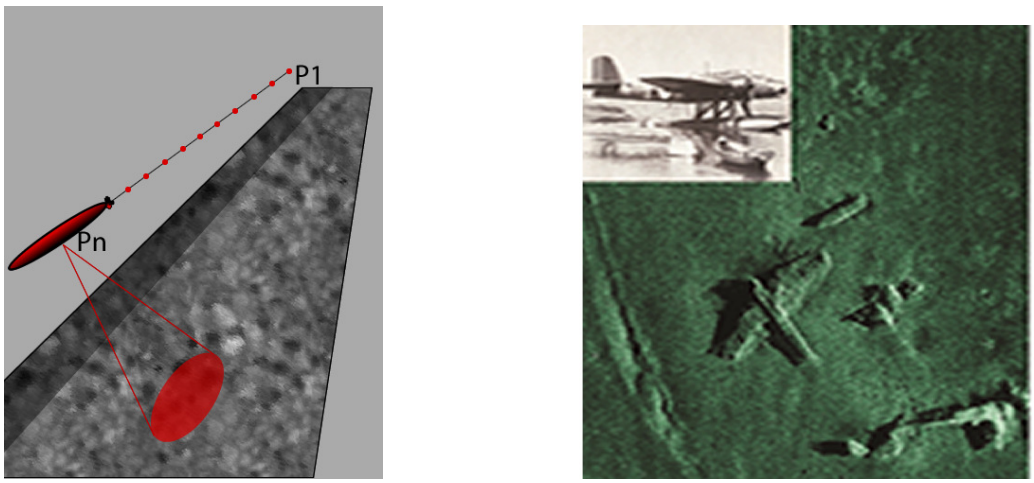
2.3.2. Sonar de apertura sintética

El principio del sonar de apertura sintética (SAS) es utilizar múltiples pulsos para crear una gran matriz sintética (o apertura). A partir de esto, se produce una imagen del lecho marino de tal manera que la información procedente de múltiples pulsos va en cada píxel del fondo marino, el esquema tradicional de un sistema SAS se puede ver en la Figura 2.6. SAS tiene el potencial de producir imágenes de alta calidad, con una resolución de centímetros en un rango de cientos de metros.

Esto convierte a SAS en una técnica adecuada para la obtención de imágenes del fondo marino para aplicaciones como la búsqueda de objetos pequeños, imágenes de naufragios, arqueología subacuática e inspección de tuberías. SAS tiene una semejanza muy cercana con el radar de apertura sintética (SAR). El sonar de apertura sintética utiliza transmisiones codificadas en fase, señales en las que la anchura de banda de señal está determinada por la codificación de la fase.

La razón para usar formas de onda codificadas en fase es aumentar la energía de la señal de transmisión mientras se mantiene un ancho de banda de señal grande. El procesamiento de la gama realizado sobre las señales codificadas en fase se denomina compresión de impulsos o filtro combinado. El objetivo de una

imagen obtenida por sonar es estimar la reflectividad acústica de la mejor manera posible, teniendo en cuenta el sensor y la geometría. SAS ofrece una excelente visión de conjunto de las propiedades de las imágenes SAR, relacionadas con el procesamiento de la señal y el contenido de la escena.



(a) (b)

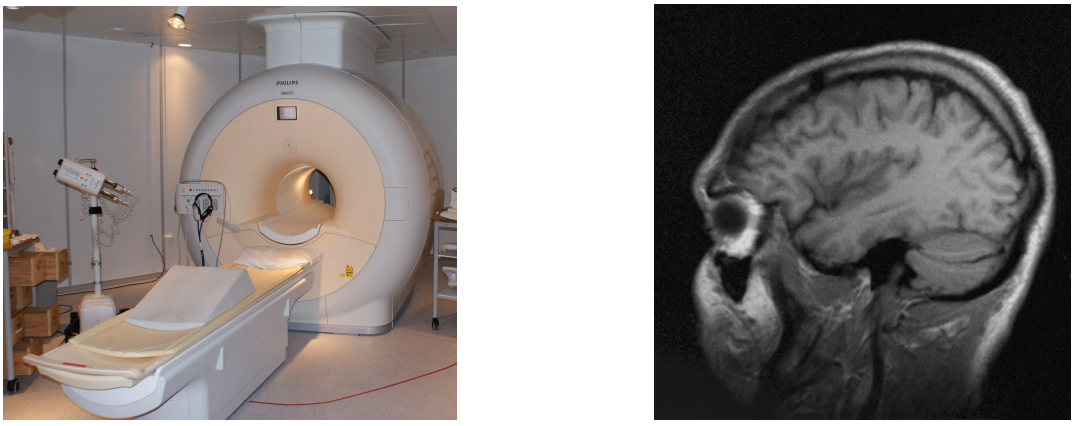
Figura 2.6: (a) Geometría de adquisición de datos para sonar de apertura sintética; (b) imagen del SAS de un hidroavión alemán Heinkel He 115 de WWII que se encuentra en el fondo marino.

2.3.3. Imagen de resonancia magnética

La imagen de resonancia magnética (MRI) es una técnica diagnóstica no invasiva que tiene el potencial de crear una imagen de algunos eventos a nivel celular o subcelular. Esta tecnología de imagen se basa en la interacción de protones entre sí y con moléculas circundantes en un tejido de interés. Cuando se colocan en un campo magnético fuerte, los protones giran a una frecuencia dada y son capaces de aceptar energía de una onda de radiofrecuencia aplicada a esta frecuencia de rotación o resonancia. El comportamiento de la energía insertada en el sistema se describe mediante dos constantes de relajación: el tiempo de relajación T_2 o transversal y el tiempo de relajación T_1 o longitudinal. Diferentes tejidos tienen diferentes tiempos de relajación, y esto puede ser utilizado para producir contraste

endógeno entre los diferentes tejidos. Los agentes de contraste exógenos pueden aumentar adicionalmente el contraste del tejido al acortar selectivamente T_1 o T_2 en un tejido de interés.

La MRI puede ponderarse para detectar diferencias en T_1 o T_2 ajustando los parámetros en la adquisición. La resonancia magnética ofrece varias ventajas sobre otras modalidades de imagen. En primer lugar, es no ionizante ya que detecta las señales magnéticas generadas por protones y otras moléculas. La técnica es también tomográfica, permitiendo cualquier plano tomográfico a través de un volumen tridimensional, se pueden obtener imágenes de alta resolución con excelente contraste de tejidos blandos entre diferentes tejidos. Por último, múltiples mecanismos de contraste son posibles mediante la resonancia magnética y la técnica se puede utilizar para proporcionar medidas anatómicas, así como lecturas fisiológicas. En la Figura 2.7 se puede ver una máquina de resonancia magnética.



(a) (b)

Figura 2.7: (a) Equipo de resonancia magnética;(b) imágene de una cabeza humana obtenida por resonancia magnética (vista sagital).

2.3.4. Proyección de patrones de franjas

La proyección de un patrón de franjas sobre el objeto es la técnica de iluminación más utilizada para la caracterización de información de las dimensiones de un

objeto y su forma tridimensional. Este método es usualmente utilizado en arreglos con una sola cámara, el patrón de franjas (franjas paralelas blancas y negras) que se proyecta sobre el objeto y su interacción es captada por una cámara, observe la Figura 2.8 . Con la obtención de la imagen de la escena, se puede procesar en una computadora, que permite determinar las dimensiones de la superficie. Cuando una rejilla o un patrón de franjas se proyecta sobre un objeto, las franjas sufren deformaciones de acuerdo a la topografía del objeto. Los patrones de franjas se pueden describir por la siguiente expresión

$$g(x, y) = a(x, y) + b(x, y) \cos [2\pi f_0 x + \varphi(x, y)] \tag{2.14}$$

donde $a(x, y)$ y $b(x, y)$ corresponden a la intensidad del fondo continuo y el contraste de las franjas, respectivamente; f_0 es la frecuencia espacial del patrón de franja, $\varphi(x, y)$ es la fase que representa la deformación del patrón de franjas proyectados sobre el objeto.

Existen varias técnicas que son utilizadas para obtener la fase de un patrón de franjas como por ejemplo: método de Fourier, método directo de detección de fase, método de detección de fase en pasos, y Contorneo por Moire. Las imágenes adquiridas están envueltas con discontinuidades de 2π. Por lo que es necesario aplicar un algoritmo de desenvolvimiento de fase para obtener la fase continua. Así, por medio del proceso de triangulación se obtiene la información tridimensional del objeto.

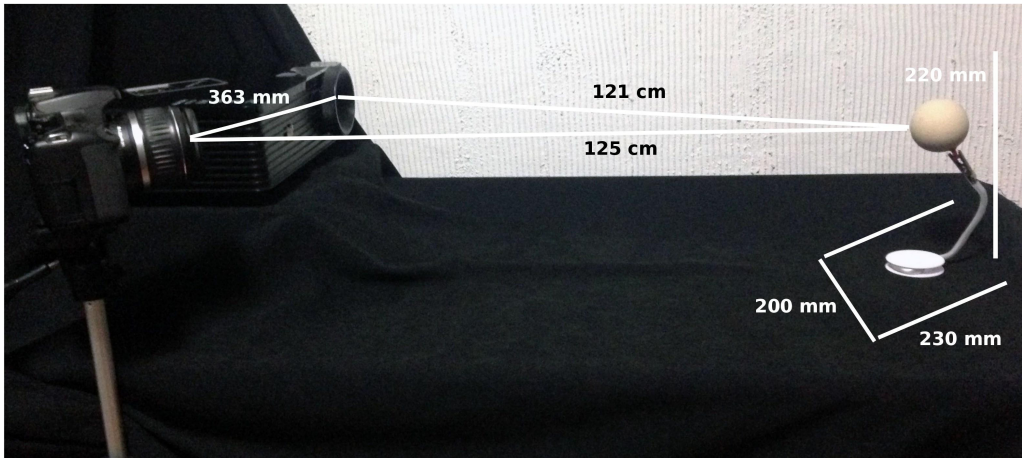


Figura 2.8: Sistema de proyección digital de patrones compuesto por una cámara y un proyector digital.

2.4. Programación paralela

La computación paralela es una forma de cómputo en la que muchas instrucciones se ejecutan simultáneamente, operando sobre el principio de que problemas grandes a menudo se pueden dividir en unos más pequeños, que luego son resueltos simultáneamente [24].

2.4.1. Arquitecturas paralelas

Las computadoras paralelas pueden clasificarse según el nivel de paralelismo que admite su hardware: equipos con procesadores multinúcleo y multi-procesador que tienen múltiples elementos de procesamiento dentro de una sola máquina y los clústeres, MPPS y grids que utilizan varios equipos para trabajar en la misma tarea. Muchas veces, para acelerar tareas específicas, se utilizan arquitecturas especializadas de computación en paralelo junto a procesadores tradicionales. La clasificación más extendida de arquitecturas hardware fue propuesta en 1972 por [8], y distingue cuatro tipos de arquitecturas:

- SISD (*Single Instruction Single Data*). Se corresponde con las arquitecturas monoprocesador clásicas.
- SIMD (*Single Instruction Multiple Data*). Una misma instrucción se ejecuta simultáneamente sobre multiples datos de entrada. El ejemplo típico de este tipo de arquitecturas son los procesadores vectoriales, que permiten aplicar en paralelo operaciones sobre arrays.
- MIMD (*Multiple Instruction Multiple Data*). Distintas instrucciones se ejecutan en paralelo, cada una de ellas operando sobre datos diferentes. Es el tipo de paralelismo más general, que viene representado por las arquitecturas multiprocesador.
- MISD (*Multiple Instruction Single Data*). Es el tipo de arquitecturas menos convencional. De hecho, no existen máquinas reales que se adapten completamente a este tipo, si bien algunos autores consideran que los procesadores segmentados se encuentran dentro de esta categoría, pues sobre cada dato de entrada se ejecutan distintas operaciones en cada una de las etapas.

La clasicación de Flynn no sólo ha servido para clasificar hardware, sino que ha incluido también algunas clasificaciones de programas paralelos, generando los conceptos de programación SPMD (*Single Program Multiple Data*) y MPMD (*Multiple Program Multiple Data*). En principio, los programas SPMD corresponden con los programas que sólo explotan paralelismo de datos, mientras que los MPMD cubren el tipo más general de paralelismo. En realidad, la única exigencia de SPMD es que el mismo código resida en todos los procesadores, por lo que también puede cubrir cualquier tipo de paralelismo sin más que utilizar identificadores de los procesadores para realizar bifurcaciones en el código del programa. También existe un equivalente a las arquitecturas MPSD (*Multiple Program Single Data*) que explota paralelismo de tipo tubería [24].

2.4.2. Modelos de programación paralela

Al igual que en la computación secuencial, existen varios modelos para la computación paralela, éstos son una mera abstracción entre el software y el hardware, sin embargo, no existe un modelo único como lo es el modelo de Von Neumann para la computación secuencial. A continuación se describen éstos modelos [20].

1. *Modelo de memoria compartida*: Los hilos de ejecución o tareas comparten un mismo espacio de direcciones, sin embargo, se puede simular en memoria distribuida, teniendo en cuenta la penalización por acceder a un espacio de memoria que no corresponda al procesador donde se ejecuta la tarea o hilo. Los hilos o tareas accederán a la memoria de forma concurrente. Por ello se tendrán que habilitar mecanismos de control de acceso a la memoria.

2. *Paso de mensajes*: Cuando varios procesos se están ejecutando y necesitan intercambiar información, una forma de hacerlo es mediante paso de mensajes. Esta comunicación puede ser tanto síncrona como asíncrona, y puede ser utilizado también como método de sincronización. En éste modelo, el programador es el encargado de generar el paralelismo a través del código de programación. Éste tipo de paralelismo se suele usar para arquitecturas de memoria distribuida.

3. *Modelo de paralelismo de datos*: Este modelo trabaja con un conjunto de datos muy grande, suele estar organizado de una forma conocida (por ejemplo, arreglos de dimensiones múltiples). Los procesos trabajan sobre zonas diferentes de los datos, normalmente se aplica la misma operación a las porciones de datos. En arquitecturas de memoria compartida, los procesos podrán tener acceso a todos los datos. Para memoria distribuida, habrá que partir la estructura de datos y repartirla. Este modelo se puede clasificar como SIMD.

4. *Modelo híbrido*: Es el resultado de combinar al menos dos de los modelos anteriores. Si vamos a un nivel de abstracción mayor, encontramos otros dos modelos para la computación paralela: SPMD y MPMD. SPMD trata de múltiples procesos ejecutando las mismas instrucciones sobre un conjunto de datos diferente. En MPMD cada proceso ejecuta un programa diferente operando sobre datos diferentes, es menos usado que SPMD. Tanto el modelo SPMD como MPMD pueden ser realizados por cualquier combinación de modelos anteriormente explicados.

2.5. Estado del arte

En los últimos 20 años se han propuesto diversos algoritmos para el desenvolvimiento de fase. Estos pueden clasificarse en: (a) algoritmos basados en ruta de integración, (b) algoritmos por regiones y (c) algoritmos globales.

Los algoritmos basados en ruta de integración desenvuelven el mapa de fase detectando saltos de la fase de 2π entre los pixeles vecinos a lo largo de una trayectoria. Estos algoritmos operan mediante el uso de recorridos lineales simples, utilizando sofisticados algoritmos de corte, o eligiendo un camino de integración basado en un criterio de calidad. Usualmente son rápidos pero incapaces de manejar imágenes de fase ruidosas. Estos a su vez, pueden ser clasificados en algoritmos de compensación residual y algoritmos guiados por calidad. Los algoritmos de compensación residual buscan residuos en una imagen de fase envuelta y colocan cortes o líneas entre residuos positivos y negativos. Debido a que este procedimiento asegura que no haya componente de rotación en el campo de fase, se puede obtener la fase desenvuelta independiente de la trayectoria cuando la ruta de integración no cruza los cortes. Sin embargo, la calidad y tiempo de ejecución dependen del criterio de generación de los cortes.

Los algoritmos globales trabajan minimizando una función de costo que tome todos los pixeles del mapa de fase envuelto en cuenta. Las soluciones se obtienen

a partir de los métodos de transformación, estimación bayesiana y optimización de redes. Los algoritmos basados en regiones funcionan dividiendo el mapa de fase en pequeñas regiones que se procesan por separado, para después fusionarlas en una sola.

El algoritmo de corte propuesta por Goldstein [9], es el primer trabajo de los basados en ruta de integración. En el algoritmo de Goldstein, las integrales no nulas de gradientes de fase envueltos alrededor de vecindades de 2×2 pixeles se denominan residuos o cargas, estos indican la presencia de discontinuidades con respecto a la suposición de que los gradientes de fase envueltos son menores de medio ciclo. Además, los valores de fase desenvuetos podrían obtenerse si un camino de integración siempre encierra un número igual de residuos positivos y negativos. Goldstein utiliza cortes para equilibrar los residuos. Por lo tanto, el problema de desenvolvimiento de fase se reduce a elegir un buen conjunto de cortes.

Recientemente han surgido trabajos que buscan mejorar el rendimiento del algoritmo de Goldstein, por ejemplo en [6] se presenta un método que aumenta el número de residuos en la fase envuelta, mediante una multiplicación en el dominio espacial para realizar cambio de frecuencia de dos dimensiones, aprovechando la propiedad de cambio de frecuencia de la transformada de Fourier. Con la creación de residuos sintéticos se busca equilibrar el mayor número de residuos y obtener cortes de menor longitud.

Algunas mejoras al algoritmo de Goldstein se centran en la modificacion del proceso de selección de cortes, muchos investigadores buscan aproximar la mejor combinación mediante distintas técnicas, tal es el caso de [5], que propone una forma de resolver el problema de la longitud de cortes basado en la teoría de residuos; se propone que los mapas de fase envueltos con saltos de fase desplazados se utilicen para equilibrar los residuos. Con este enfoque se busca minimizar el tiempo de procesamiento y la conexión de residuos, lo cual es esencial en el desarrollo de algoritmos de corte. Como resultado de los experimentos realiza-

dos se obtiene que un algoritmo de corte bajo este enfoque es capaz de resolver mapas de fase con un altas concentraciones de ruido y cambios abruptos de fase.

Otra manera de equilibrar los cortes es la propuesta por [32] donde se sugiere la implementación de un algoritmo para buscar la longitud mínima de un corte, convirtiendo el problema de búsqueda de cortes al problema del vendedor viajero o TSP. El rendimiento del algoritmo es probado en mapas de fase envueltas simulados y reales. Aunque el algoritmo por su naturaleza es lento al realizar los cortes tiene una alta probabilidad de encontrar la longitud de corte más corto o una aproximación del mismo.

Bajo el argumento de optimizar el tiempo de selección de cortes y obtener un mejor resultado, en [19] se propone que el proceso de colocación de cortes esté guiada por un mapa de calidad inicial, para limitar la existencia de los cortes en zonas donde se encuentran los pixeles de baja calidad, después asignar el valor de calidad mínimo a los pixeles que están marcados como corte para generar un mapa de calidad final, por último, los pixeles de mayor calidad con la máxima fiabilidad se integraran primero y los pixeles con una confiabilidad gradualmente decreciente se integran en secuencia. El algoritmo creado bajo este enfoque tiene la ventaja que cualquier mapa de calidad derivado directamente de los datos de fase envuelta se puede elegir como mapa de calidad original o inicial.

Como se ha estado mencionando en el texto, la configuración de cortes es muy importante en el algoritmo de Goldstein, y como es imposible probar todas las posibles configuraciones en un tiempo razonable, se necesitan métodos de selección de cortes. En [11] se explica que la orientación de los cortes y la dirección del mapa de fase envuelto ayudan a tomar criterios adicionales para optimizar la configuración de cortes, si las fuentes de discontinuidad se deben a procesos de filtrado, ruido, agujeros en el patrón de franjas y discontinuidades paralelas a las franjas, además, dichos criterios pueden ser implementados en una función que minimice el costo de la longitud de los cortes con la técnica de *Simulated Anneling*.

En otra modificación reciente del algoritmo de Goldstein [17] se explica un nue-

vo enfoque denominado vector de residuos, que es un vector generado por un residuo en un mapa de fase envuelto y tiene orientación hacia el residuo de equilibrio, este enfoque propone usar el vector de residuos como guía para colocar los cortes. Además, se descubrió que el vector de residuos proporciona toda la información necesaria para colocar el corte y equilibrar la discontinuidad, no solo encuentra el residuo ideal, si no también, es capaz de identificar los posibles residuos que podrían utilizarse para equilibrar el corte. A diferencia del algoritmo de flujo de costo mínimo que tiene que resolver los flujos de red hasta que identifica el residuo de equilibrio óptimo, es decir, todos los residuos en la red son posibles residuos de equilibrado, lo que aumenta la complejidad del problema.

En [7] se asegura que los residuos son de ayuda para encontrar discontinuidades en el mapa de fase envuelto, pero no son necesarios, por lo que un corte pequeño no asegura del todo un buen resultado, la razón es que los cortes eliminan errores locales, más no errores globales por lo que es necesario colocar cortes en base a un mapa de calidad, bajo este enfoque se propone un mapa de calidad mejorado y un algoritmo más fiable de colocación de cortes basado en la detección de borde en el mapa de calidad. Los experimentos y simulación muestran que el método modificado es efectivo en zonas donde existe discontinuidad regional, además resulta ser más práctico que el algoritmo de Goldstein.

Actualmente, existen aplicaciones que requieren procesar imágenes grandes en tiempo real, lo cual, no es posible con implementaciones seriales, por lo que es necesario utilizar otras estrategias para realizar el desenvolvimiento de fase. Las mejoras se pueden hacer desde hardware aumentando los recursos computacionales como tarjetas gráficas y estaciones de trabajo; o también desde software con estrategias de programación más eficientes como el paralelismo.

Varios autores proponen versiones más rápidas basadas en distintas técnicas, ejemplo de ello es la presentada por [15] con una implementación de una versión modificada del algoritmo de Goldstein que se ejecuta de manera paralela. La estrategia que se utiliza es partir la imagen y procesarla de manera simultánea. La

etapa de elección de cortes se realiza minimizando una función objetivo con la estrategia de Simulated Anneling con el fin de minimizar su longitud, el algoritmo es capaz de procesar mapas de fase grandes y los tiempos que se reportan es de 25 veces más rápido que la misma implementación serial, además que la memoria utilizada durante la ejecución es $50\,\%$ menos que la versión serial.

La mayoría de las versiones paralelas que se han propuesto se ejecutan en la GPU (*Graphics Processor Unit*), ya que es una de las mejores opciones por el número de núcleos que maneja, aunque la velocidad de una GPU no se puede comparar con la velocidad del procesador CPU (*Central Processing Unit*), se pueden crear muchas tareas simultáneamente.

En [1] proponen un algoritmo de desenvolvimiento de fase basado en mínimos cuadrados no ponderados para la extracción rápida de mapas de fase cuantitativos fuera del eje, implementada en lenguaje de C y CUDA (*Compute Unified Device Architecture*) que se ejecuta en la GPU; en [16] presentan un algoritmo de desenvolvimiento de fase basado en técnicas de Fourier de alta velocidad que es robusto con mapas de fase con alto nivel de ruido y residuos, el algoritmo se ejecuta de manera paralela en una tarjeta gráfica estándar, se obtuvo un tiempo de 5 ms en un mapa de fase de 640$\times$480 px la cual contiene un alto número de saltos de fase.

En [18] presenta un sistema de medición de forma tridimensional que puede lograr simultáneamente la adquisición, reconstrucción y visualización de forma tridimensional a 30 fotogramas por segundo (fps) con 480 puntos de medición por trama. Todo el procesamiento se realiza en una GPU sin necesidad de una potente CPU, la extracción de la fase se realiza mediante la técnica de escaneado de luz estructurado.

Finalmente, en [3] proponen un método de desenvolvimiento de fase multiescalar que desenvuelve muestras del mapa de fase envuelto a diferentes niveles dependiendo de sus diferencias de fase. Para determinar si el aliasing de fase se producirá después del muestreo descendente y, por tanto, si una muestra de paso

bajo es adecuada para desenvolver en algún nivel de descomposición debe determinarse el componente de gradiente de fase local correspondiente de la señal de paso bajo. Cuando un coeficiente de paso bajo no es adecuado para desenvolver debido al aliasing de fase.

Capítulo 3

Optimización del algoritmo de Goldstein

En este capítulo se explica el procedimiento de desenvolvimiento de fase utilizado por el algoritmo de Goldstein; se describen las tres etapas de identificación de residuos, generación de cortes e integración. Se describe la estrategia de partición del mapa de fase utilizada para obtener la paralelización del algoritmo de Goldstein. Se muestran ejemplos prácticos del funcionamiento del algoritmo en su versión paralela, tiempos de ejecución y desempeño, y por último se discuten los resultados obtenidos.

3.1. El algoritmo de Goldstein

De manera general el algoritmo propuesto por Goldstein consta de tres etapas importantes para obtener el mapa de fase desenvuelto las cuales son: identificación de residuos, generación de cortes e integración. A continuación se explica a detalle cada una de ellas.

3.1.1. Identificación de residuos

Un residuo es una inconsistencia en el mapa de fase y su identificación se puede realizar en orden o no, es decir, no es necesario iniciar en un píxel en específico, ni llevar una trayectoria fija. Para la identificación de los residuos es necesario contar con dos matrices, una que contenga los valores de fase envueltos y otra donde se guarden los valores que indican si un píxel es un residuo o no. Para determinar si un píxel es un residuo se realiza el siguiente procedimiento: se toma un píxel p con coordenadas (x, y) vea la Figura 3.1 y se calcula la suma de las diferencias de fase q que se definen en la ecuación 3.1.

Si $q > 0$, entonces se considera residuo positivo; mientras que si $q < 0$, entonces es un residuo negativo; de lo contrario si $q = 0$ el píxel no es residuo. Determinado el valor del píxel, éste se guarda en la segunda matriz usando las mismas coordenadas. Los valores posibles son los siguientes: 1 para residuos positivos, 2 para residuos negativos y 0 para pixeles que no son residuo. El proceso se repite píxel a píxel hasta que se recorra toda la matriz.

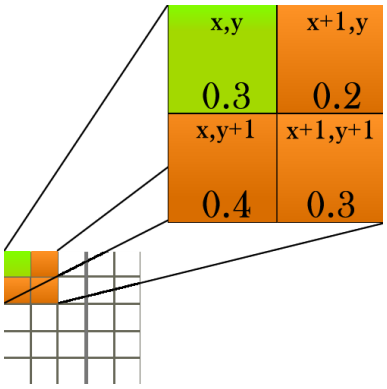


Figura 3.1: Vecindad de 2×2 del píxel con coordenadas (x, y).

$$q = \sum_{i=1}^{4} \Delta_i = \Delta_1 + \Delta_2 + \Delta_3 + \Delta_4 \tag{3.1}$$

donde

$$\begin{aligned}
\Delta_1 &= \psi(x,y) - \psi(x,y+1), \\
\Delta_2 &= \psi(x,y+1) - \psi(x+1,y+1), \\
\Delta_3 &= \psi(x+1,y+1) - \psi(x+1,y), \\
\Delta_4 &= \psi(x+1,y) - \psi(x,y).
\end{aligned}$$

3.1.2. Generación de cortes

Un corte es una línea que une dos residuos (uno positivo y otro negativo), o un residuo con un borde de la imagen, si es que no existe otro residuo de polaridad opuesta y si el residuo se encuentra cerca del borde de la imagen. La Figura 3.2 muestra ejemplos de cortes con pixeles de polaridad opuesta, y pixeles con el borde de la imagen. La Figura 3.3 muestra dos ejemplos de cortes representados como pixeles en una imagen.

La generación de cortes se realiza de la siguiente manera: se recorre la matriz en busca de un residuo, cuando es ubicado se toma como centro y se busca alrededor un residuo, si lo encuentra se traza un corte, si el residuo ubicado es de polaridad contraria se termina el procedimiento o de lo contrario se aumenta la búsqueda a un radio de 2 pixeles, y se repite el procedimiento. El radio de búsqueda se puede aumentar a 3 pixeles como máximo, si no lo encuentra se repite la búsqueda en el otro píxel que pertenece al corte, esto continúa hasta que exista el mismo número de residuos positivos y negativos en el corte. Para que un corte se considere de buena calidad debe cumplir con las siguientes características:

1. Los cortes deben tener el mismo número de residuos positivos y negativos.
2. Un corte debe ser lo más pequeño posible.
3. Los cortes no deben aislar áreas de la imagen.
4. El grosor de un corte debe ser de un píxel.

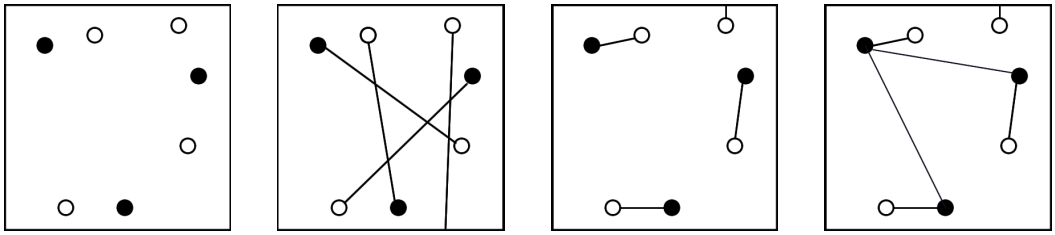


Figura 3.2: (a) Residuos positivos y negativos en una matriz de fase envuelta, (b) configuración de cortes donde los mismos generan zonas aisladas, (c) configuración ideal de cortes, en este ejemplo no se generan zonas aisladas y todos los residuos quedan equilibrados, (d) ejemplo de cortes equilibrados, sin ser de buena calidad por la longitud, lo que seguramente provocará malos resultados en el desenvolvimiento.

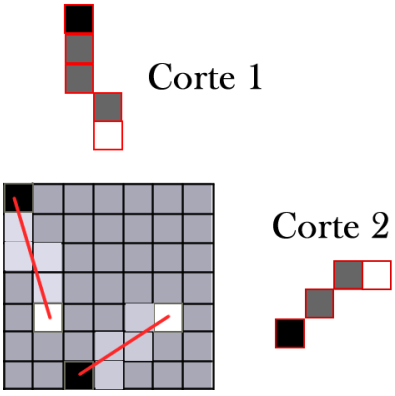


Figura 3.3: Ejemplo de cortes en una matriz de fase envuelta.

3.1.3. Integración

La integración es el proceso en el que se obtiene la fase desenvuelta $\varphi(x, y)$, el procedimiento comienza eligiendo un píxel inicial p con coordenadas (x, y) de la imagen, que no sea residuo ni este marcado como corte, la solución de ese píxel será su valor de fase envuelta $\psi(x, y)$, después se ingresan, a una estructura de datos de cola, los pixeles vecinos $\varphi(x + 1, y)$, $\varphi(x - 1, y)$, $\varphi(x, y + 1)$ y $\varphi(x, y - 1)$ siempre y cuando no estén marcados como corte o residuo; posteriormente se saca un valor de la cola y se repite la inserción de los vecinos, el procedimiento

termina hasta que la cola quede vacía; por último, se resuelven los pixeles que no eran residuo ni parte de un corte y se repite todo el procedimiento con los pixeles que están marcados como cortes. A continuación se muestran las ecuaciones con las que se calcula la fase desenvuelta de los vecinos de un píxel p con coordenadas (x, y).

$$\begin{aligned}
\varphi(x+1,y) &= \varphi(x,y) - [\psi(x,y) - \psi(x+1,y)] \\
\varphi(x-1,y) &= \varphi(x,y) + [\psi(x,y) - \psi(x-1,y)] \\
\varphi(x,y+1) &= \varphi(x,y) - [\psi(x,y) - \psi(x,y+1)] \\
\varphi(x,y-1) &= \varphi(x,y) + [\psi(x,y) - \psi(x,y-1)]
\end{aligned}$$

La Figura 3.4 muestra un ejemplo de integración en el caso que no hay residuos y cortes presentes. En la Figura 3.5 se muestra un ejemplo de integración con presencia de cortes sin zonas aisladas en el mapa de fase. Finalmente, en la Figura 3.6 se muestra un ejemplo de integración con presencia de cortes que generan zonas aisladas en el mapa de fase. En este último ejemplo, cada región aislada de fase se tiene que integrar por separado utilizando un pixel inicial por región. Cuando existen zonas aisladas, ya sea por la generación de cortes o cambios abruptos en la fase se pierde discontinuidad en el mapa de fase desenvuelto.

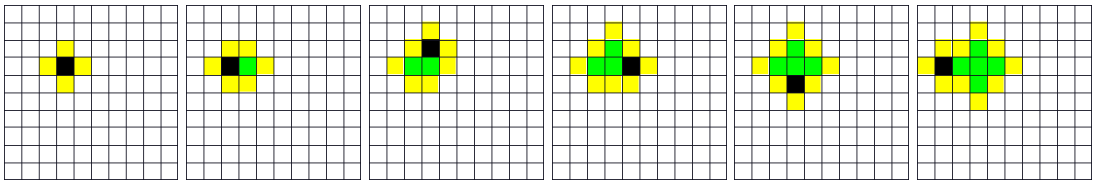
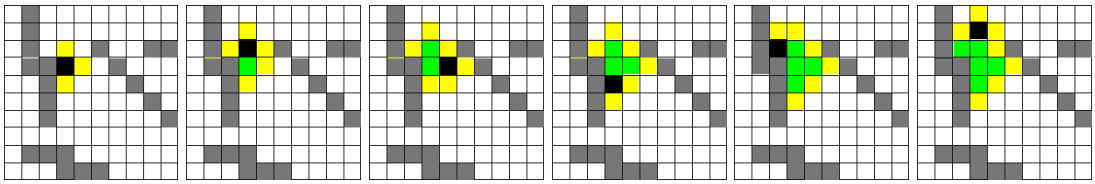
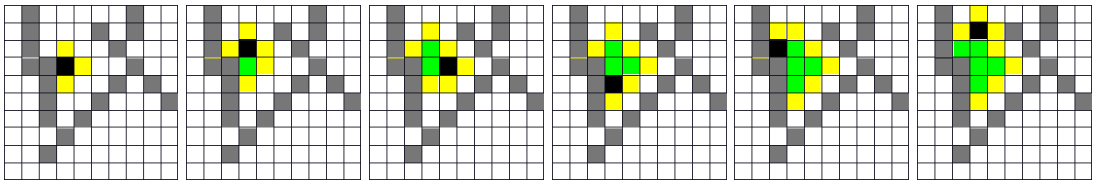


Figura 3.4: Proceso de integración en un mapa de fase que no cuenta con residuos. Los cuadros negros representan pixeles que se están procesando, amarillos a vecinos del píxel que se está resolviendo y verdes a pixeles resueltos.



(a) (b) (c) (d) (e) (f)

Figura 3.5: Proceso de integración en un mapa de fase con presencia de cortes. Los cuadros negros representan pixeles que se están procesando, amarillos a vecinos del píxel que se está resolviendo, verdes a pixeles resueltos y los de color gris a cortes.



(a) (b) (c) (d) (e) (f)

Figura 3.6: Proceso de integración cuando existen áreas aisladas a consecuencia de cruces entre cortes en el mapa de fase envuelto. Los cuadros negros representan pixeles que se están procesando, amarillos a vecinos del píxel que se está resolviendo, verdes a pixeles resueltos y los de color gris a cortes.

3.2. Paralelización del algoritmo de Goldstein

El algoritmo de Goldstein inicialmente se diseñó para implementarse y ejecutarse de manera secuencial, por lo tanto, para poder ser implementado de manera paralela es necesario verificar que no exista dependencia de datos y otros aspectos de hardware como memoria disponible, la arquitectura de las computadoras y el lenguaje de programación. En esta investigación se utilizó una arquitectura multi-núcleo con un modelo de memoria compartida, y el lenguaje de programación C con OpenMP para programación paralela. En las siguientes secciones se describen la estrategia de paralelización para las tres etapas del algoritmo de Goldstein.

3.2.1. Identificación de residuos

Para iniciar el proceso de identificación de residuos es necesario contar con dos matrices; una donde se guarden los valores que indican si un píxel es residuo o no, la cual se llamará `bitflags` y la otra donde se encuentran los valores de fase y que se llamará `phase`. Para saber si un píxel p con coordenadas (x, y) es residuo o no, se necesita consultar los valores de la matriz phase en las siguientes coordenadas $(x+1, y)$, $(x+1, y+1)$ y $(x, y+1)$, y sumar las diferencias de fase como lo indica la ecuación (3.1); si el resultado es mayor o menor que 0 se considera que es un residuo, el resultado se guarda en la matriz `bitflags` en el siguiente formato: 1 residuos positivos, 2 residuos negativos y 0 pixeles sin problemas; tome en cuenta que este procedimiento se realiza en todos los pixeles de la matriz `phase`.

Analizando el procedimiento se puede deducir que la identificación de un residuo solo ocupa una área de la matriz de 2×2 pixeles, por lo tanto, se asume que se pueden verificar dos pixeles al mismo tiempo, siempre y cuando no tengan un vecino en común como se muestra en la Figura 3.1, entonces, no es difícil deducir que se pueden verificar varios residuos al mismo tiempo, siempre y cuando se cumpla que no comparten un vecino entre todos los pixeles que se estén procesando. Tomando en cuenta que el número de *threads* o hilo de procesamiento que se pueden crear va de la mano con los núcleos del procesador, se procede a divididir la matriz fase en el doble del número de núcleos disponibles como se muestra en la Figura 3.7.

La idea de partir la matriz en franjas es para crear un hilo por cada núcleo disponible. Como la imagen se divide en el doble del número de núcleos, cada hilo dejará una franja sin procesar. Por lo tanto, la matriz se procesa en su totalidad en dos partes, con este tipo de partición se asegura que los hilos nunca consulten un píxel al mismo tiempo. Vea la Figura 3.8 donde se muestra el orden de ejecución.

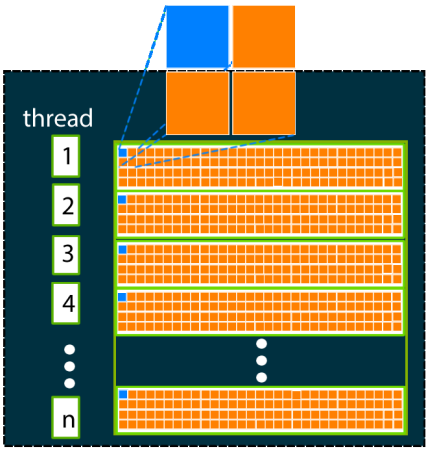


Figura 3.7: Estrategia para las búsqueda de residuos en paralelo donde cada franja (área delimitadas por bordes de color verde) es procesada por un hilo. Los cuadros de color azul son los pixeles que se desea saber si son residuos. Se puede notar que no hay forma en la que dos hilos utilicen una misma área de la imagen.

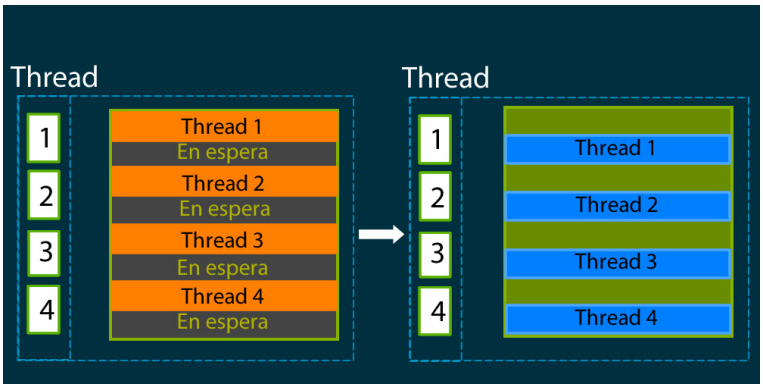
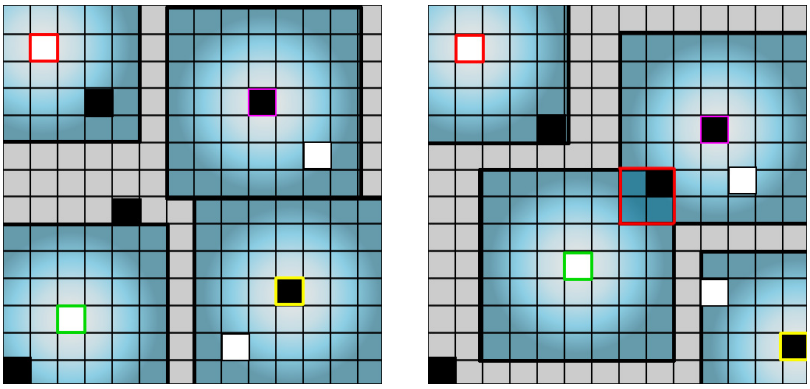


Figura 3.8: Estrategia de partición para la generación de cortes donde las franjas naranjas y azules representan hilos que realizan cortes en la matriz de manera paralela. El ejemplo que se muestra es el resultado de una ejecución con un procesador de cuatro núcleos.

3.2.2. Generación de cortes

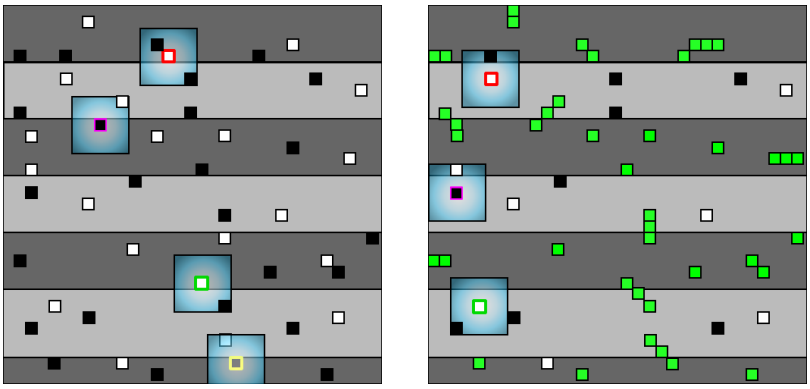
De manera general el proceso de generación de cortes consiste en trazar líneas (cortes) entre residuos de polaridad opuesta o residuo y borde de la imagen. Es importante tener en cuenta que la longitud de los cortes debe ser lo más corta posible y tratar de equilibrar todos los residuos para obtener un buen resultado. Pa-

ra trazar un corte primero se ubica un residuo ya sea positivo o negativo, después la búsqueda se realiza alrededor del residuo aumentando el radio de búsqueda hasta encontrar el residuo opuesto como se muestra en la Figura 3.9. Ya ubicados los dos residuos se traza el corte, se repite el mismo proceso hasta que no exista un solo residuo. De la misma manera que la identificación de residuos, el proceso solo depende del área de búsqueda, por lo tanto, se decidió utilizar la misma estrategia de partición utilizada en la identificación de residuos; la única diferencia es que el ancho de las bandas está limitado por el radio de búsqueda, la razón es que si el radio de búsqueda es mayor que el ancho de la banda puede tener problemas pues la búsqueda entraría en otra banda a la cual no se le asigno y por ende entrar en conflicto con el hilo que está asignado a ese espacio, véase la Figura 3.10. Bajo este enfoque de partición se pueden implementar muchos algoritmos para la generación de cortes.



(a) (b)

Figura 3.9: En esta imagen se muestra que cada residuo tiene un radio de búsqueda (cuadro de color azul) el cual delimita el área de búsqueda del residuo contrario; (a) se realizan 4 búsquedas en paralelo de residuos contrarios donde el radio de búsqueda de cada residuo está separado uno de otro, (b) se realizan 4 búsquedas en paralelo de residuos contrarios donde existe conflicto en dos hilos pues dos radios de búsqueda utilizan un área en común (cuadro representado con borde de color rojo).



(a) (b)

Figura 3.10: Estrategia paralela de generación de cortes. Los residuos positivos están representados por puntos de color blanco, residuos negativos de color negro y los pixeles que pertenecen a un corte de color verde; los cuadros de color azul representan el radio de búsqueda de cada residuo. El proceso de selección de cortes consiste en dos partes: en (a) se ilustra que se ejecuta un hilo por cada franja de color gris obscuro y en (b) se ejecuta un hilo por cada franja gris clara.

3.2.3. Integración

Para realizar la integración es necesario comenzar a partir de un píxel lo que causa que la fase desenvuelta sea relativa el pixel de inicio y dificulta la paralelización de la integración. Una alternativa para paralelizar la integración es resolver una zona de la imagen de manera lineal, y después aprovechar los pixeles que ya se encuentran desenvueltos para iniciar el proceso de desenvolvimiento de nuevas zonas pero ahora en paralelo, de esta manera al finalizar el total de la matriz el resultado será obtenido de un mismo punto de inicio. La estrategia para descomponer en partes el proceso de integración consiste en crear un bloque central $T1$ de tamaño $w \times h$ el cual se resolverá de manera serial, ya que el bloque $T1$ se resolvió se crea un segundo bloque central del doble de tamaño que el anterior dividido en bloques de tamaño $w \times h$. Por cada bloque creado se ejecutará un hilo que realizara el proceso de integración tomando como inicio todos los pixeles que ya se encuentren resueltos en el bloque anterior contiguo; ya que se resolvieron

en su totalidad, se repite el proceso, es decir, se crea un bloque del doble del tamaño y se divide en sub-bloques de tamaño $w \times h$ y el proceso continúa hasta que se haya resuelto el mapa de fase en su totalidad. En la Figura 3.11 se puede ver la estrategia de partición para la creación de bloques en el proceso de integración.

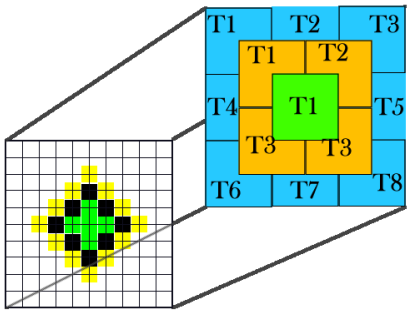


Figura 3.11: Estrategia de partición para la integración donde el cuadro de color verde $T1$ es el primer bloque de tamaño $w \times h$, posteriormente se crea una segunda zona del doble del tamaño de color naranja dividido en $T1$, $T2$, $T3$ y $T4$ de tamaño $w \times h$ y así sucesivamente hasta cubrir el total de la matriz. Cada bloque se resolverá por un hilo distinto y los colores representan las etapas necesarias para resolver toda la imagen.

3.3. Experimentos y discusión

Para conocer la eficiencia y robustez del algoritmo propuesto se usaron distintas fuentes de fase envueltas que presenten las principales fuentes de error (ruido, cambios de fase abruptos y bajo muestreo). Se usaron mapas de fase generados por computadora y datos reales provenientes de fuentes de fase como SAR, IFSAR y MRI. En la Tabla 3.1 se muestran los mapas utilizados para realizar las pruebas de eficiencia.

Se usó el algoritmo paralelo de Goldstein propuesto a cada uno de los mapas de fase en la Tabla 3.1. A continuación se discuten los resultados obtenidos y tiempos de ejecución del algoritmo paralelo.

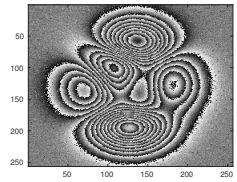
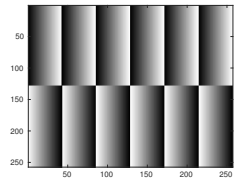
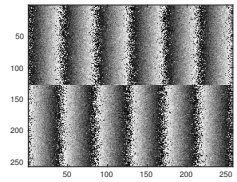
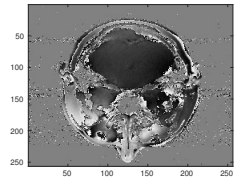
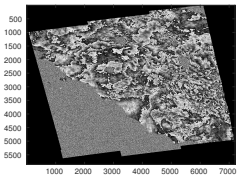
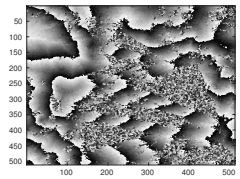
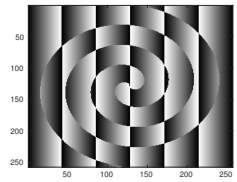
Mapa	Nombre	Dimensiones (px)	Tipo
	peaks.256×256.phase	256×256	float
	shear.257×257.phase	257×257	byte
	noise.257×257.phase	257×257	byte
	head.256×256.phase	256×256	byte
	ifsar.5854×7202.phase	5854×7202	float
	ifsar.512×512.phase	512×512	byte
	spiral.257×257.phase	257×257	byte

Tabla 3.1: Mapas de fase de distintas fuentes como: Sintéticos, MRI, SAR y IFSAR, los cuales presentan fuentes de error como cambios de fase abruptos, ruido y bajo muestreo.

El desempeño del algoritmo propuesto depende directamente de la arquitectura de la computadora donde se ejecute, por lo que, la velocidad varia de un equipo a otro. En este trabajo las pruebas se realizaron en un equipo con las siguientes características:

- Marca: Acer-Aspire-E5-523.
- CPU: Dual core AMD A9-9410 RADEON R5 5 COMPUTE CORES 2C+3G.
- Disco duro : SSD 450 GB.
- Memoria RAM: 8 GB.
- Sistema operativo: Linux Mint 18.2 cinnamon 64 bits.

A continuación se discuten los resultados obtenidos:

1. El mapa de fase sintético (peaks.256x256.phase) se generó con el software MATLAB que corresponde a la función de distribución de picos, este mapa contiene ruido el cual es considerado una de las principales fuentes de error en el proceso de desenvolvimiento de fase. La Figura 3.13 muestra ejemplos de las etapas del algoritmo de Goldstein paralelo. En la Figura 3.13(f) se puede ver que el mapa de fase fue desenvuelto correctamente.

2. El mapa de fase sintético (shear.257x257.phase) fue generado en MATLAB y no contiene ruido pero contiene cambios abruptos de fase, que es considerada también una fuente de error. La Figura 3.14 muestra ejemplos de las etapas del algoritmo de Goldstein paralelo. En la Figura 3.14(f) se puede visualizar el resultado obtenido y es posible decir que el algoritmo realiza de manera correcta el desenvolvimiento.

3. El mapa de fase sintético (noise.257x257.phase) se le agregó ruido al mapa anterior generando dos fuentes de error, ruido y cambios abruptos de fase. La Figura 3.15 muestra ejemplos de las etapas del algoritmo de Goldstein

paralelo. El resultado obtenido se puede ver en la Figura 3.15(f) y nuevamente el algoritmo obtiene buenos resultados. En esta prueba es posible notar que existen cortes entre bloques distintos, con lo que se puede determinar que las estrategias utilizadas para la partición de trabajo no intervienen en la generación de cortes y la integración se inicia desde el mismo punto de origen.

4. El mapa de fase obtenido de una fuente MRI (head.256x256.phase) contiene una excesiva cantidad de residuos, y por lo tanto un alto grado de dificultad. En la Figura 3.16 (a) es posible ver la cantidad de residuos que existen y a diferencia de los mapas anteriores estos se concentran en la mayor parte de la matriz. En la Figura 3.16 (d) se puede ver que los cortes son más largos y en ocasiones un corte cruza dos bloques distintos, lo que significa que se paralelizo de manera correcta. El mapa de fase desenvuelto se puede ver en la Figura 3.16(f).

5. El mapa de fase obtenido de un sistema IFSAR (ifsar.5854x7202.phase) que concentra una gran cantidad de residuos en algunas regiones de la matriz lo que hace difícil obtener el resultado adecuado. La Figura 3.17 muestra ejemplos de las etapas del algoritmo de Goldstein paralelo. En la figura 3.17 (e) se puede ver el mapa desenvuelto. En este caso el algoritmo que genera los cortes no alcanza a procesar todos los residuos, razón por la que se notan algunas inconsistencias en el resultado.

6. El mapa de fase obtenido de un sistema IFSAR (ifsar.512x512.phase) donde los residuos se concentran en áreas determinadas, aunque es una imagen pequeña los residuos ocupan gran parte del mapa de fase, lo que provoca que algunos no se unan a un corte. En la Figura 3.18 (d) se puede notar que algunos cortes son demasiado largos lo que provoca que se creen áreas cerradas y la solución no sea de un mismo punto de inicio. Al existir áreas cerradas el proceso de integración no se puede realizar de manera correcta

en la Figura 3.18 (f) se muestra el resultado.

7. El mapa de fase sintético (spiral.257x257.phase) creado en MATLAB el cual contiene cambios de fase abruptos en toda la matriz, normalmente este tipo de mapas son difíciles para la mayoría de los algoritmos de desenvolvimiento de fase. El resultado obtenido es malo como se puede ver en la Figura 3.19 (f), a pesar que la mayoría de los residuos fueron unidos a un corte, esto se puede verificar en la Figura 3.19 (d), por lo tanto, se puede determinar que el resultado es provocado por los cambios de fase al no existir cortes cruzados entre ellos.

La finalidad de realizar un algoritmo paralelo basado en el algoritmo de Goldstein es disminuir el tiempo de ejecución sin perder calidad con respecto al algoritmo serial. A continuación se presentan los tiempos de ejecución del algoritmo de Goldstein para su versión serial y paralela usando el mapa de fase peaks con diferentes dimensiones. La Tabla 3.2 muestra los tiempos de ejecución del algoritmo de Goldstein serial y paralelo.

Tabla 3.2: Tiempos de ejecución del algoritmo de Goldstein paralelo y serial con mapa de fase envuelto de la función de distribución peaks en distintas dimensiones. Los tiempos de ejecución están en milisegundos. La aceleración es la razón entre el tiempo serial y paralelo.

Dimensión (px)	Serial (ms)	Paralelo (ms)	Aceleración
128×128	2.57	0.98	2.60
256×256	9.90	2.35	4.19
512×512	50.75	7.70	6.58
1024×1024	248.55	28.29	8.78
2048×2048	1049.12	117.97	8.89
4096×4096	4734.43	472.43	10.02
8192×8192	20001.63	1962.20	10.19

3.4. Conclusiones

Utilizando programación paralela es posible multiplicar la velocidad del algoritmo para el desenvolvimiento de fase propuesto por Goldstein sin perder calidad en el resultado. En la Figura 3.12 se puede visualizar la mejora en tiempos de ejecución del algoritmo paralelo con respecto a la version original. La estrategia por bloques propuesta en las tres etapas es capaz de dar resultados similares a la versión original del algoritmo, sin embargo, la generación de cortes es un proceso que requiere de muchos recursos computacionales (memoria y velocidad del procesador). El uso de una estrategia distinta a la utilizada en este trabajo podría mejorar la calidad de la fase desenvuelta pero repercutiría directamente en la velocidad del proceso. En las pruebas donde el resultado fue desfavorable se debió a la longitud de los cortes, pues la estrategia utilizada para trazarlos está pensada en la optimización del tiempo, sin embargo, la estrategia de partición puede adaptarse a otros algoritmos propuestos por otros autores.

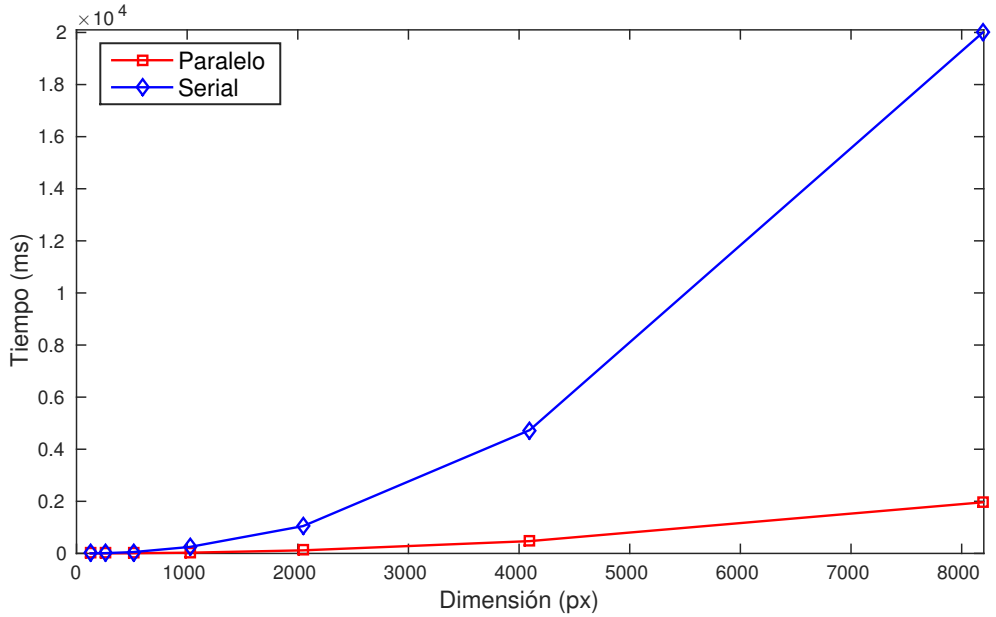
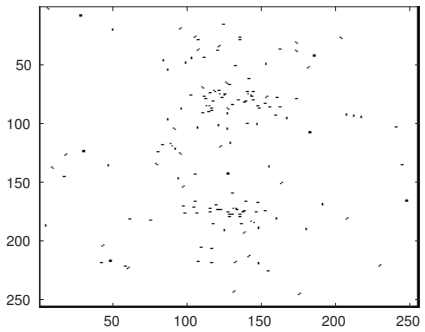
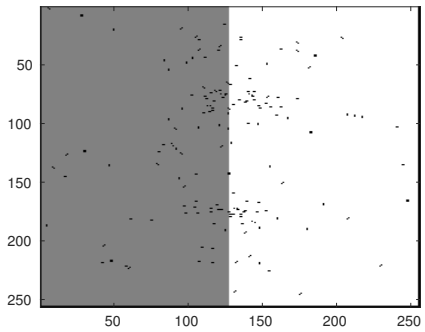


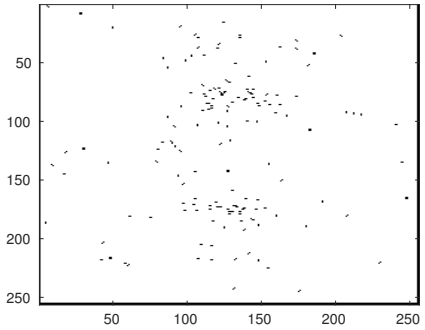
Figura 3.12: Comparación de tiempos de ejecución del Algoritmo de Goldstein serial y paralelo para el mapa de fase peaks con diferentes dimensiones.



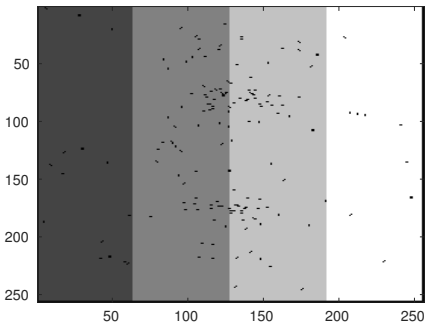
(a) Residuos



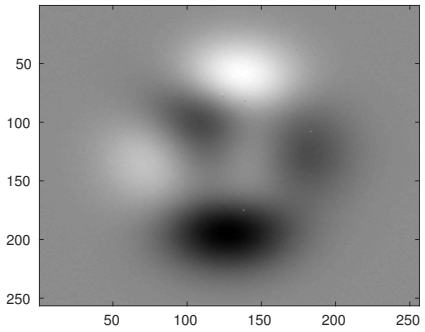
(b) Búsqueda de residuos por bloques



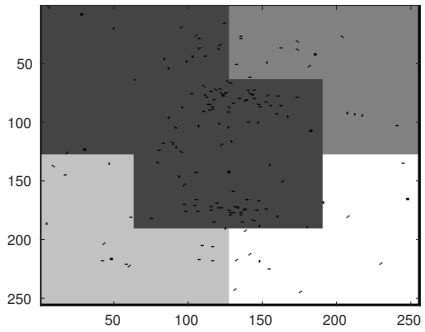
(c) Cortes



(d) Generación de cortes por bloques

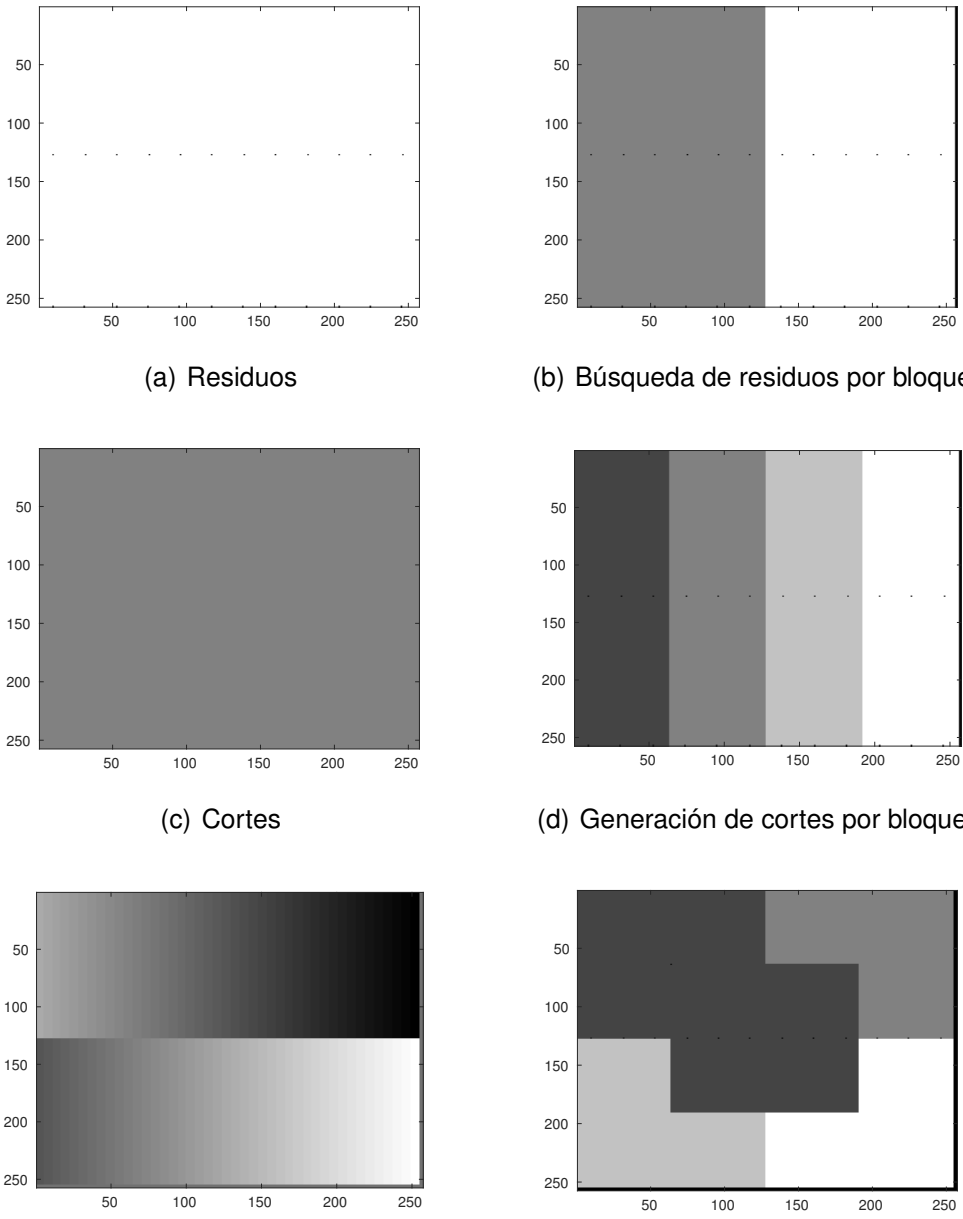


(e) Fase desenvuelta



(f) Integración por bloques

Figura 3.13: Desenvolvimiento de fase del mapa peaks.256x256 donde las sub-figuras a, c y e representan cada etapa del algoritmo de Goldstein (búsqueda de residuos, selección de cortes e integración). Las sub-figuras b, d, y f representan los hilos creados por etapa. Cada tono de gris es un hilo ejecutado en paralelo y los píxeles negros representan residuos o cortes.



(a) Residuos

(b) Búsqueda de residuos por bloques

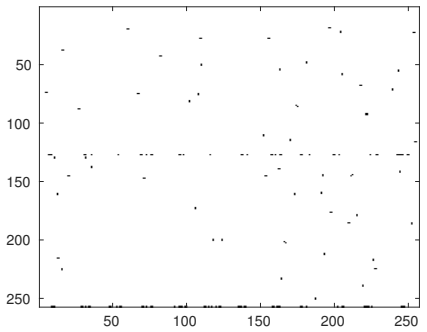
(c) Cortes

(d) Generación de cortes por bloques

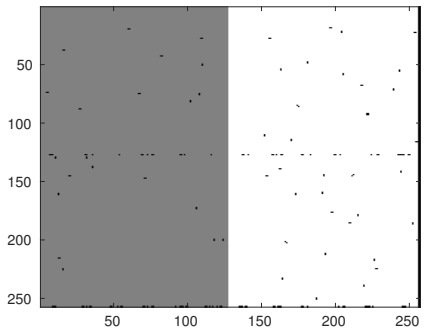
(e) Fase desenvuelta

(f) Integración por bloques

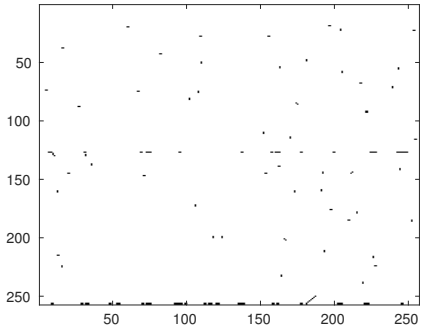
Figura 3.14: Desenvolvimiento de fase del mapa shear.257x257 donde las sub-figuras a, c y e representan cada etapa del algoritmo de Goldstein (búsqueda de residuos, selección de cortes e integración). Las sub-figuras b, d, y f representan los hilos creados por etapa. Cada tono de gris es un hilo ejecutado en paralelo, y los píxeles negros representan residuos o cortes.



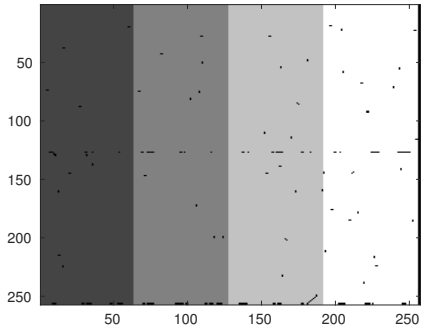
(a) Residuos



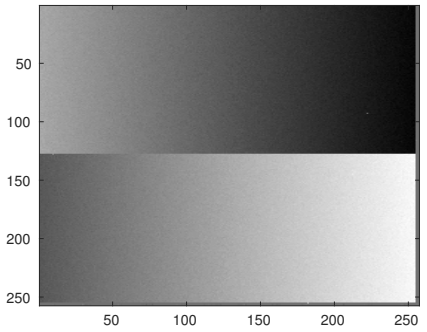
(b) Búsqueda de residuos por bloques



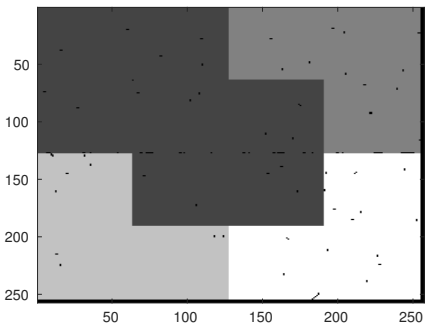
(c) Cortes



(d) Generación de cortes por bloques

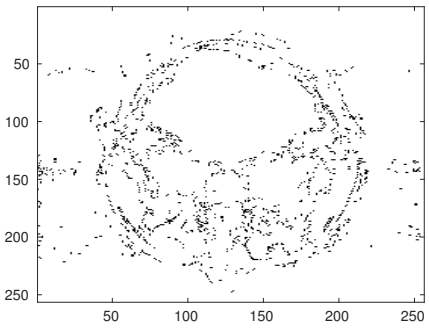


(e) Fapa desenvuelta

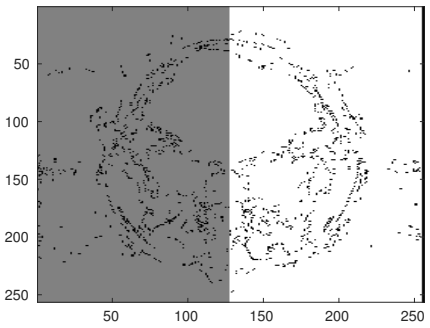


(f) Integración por bloques

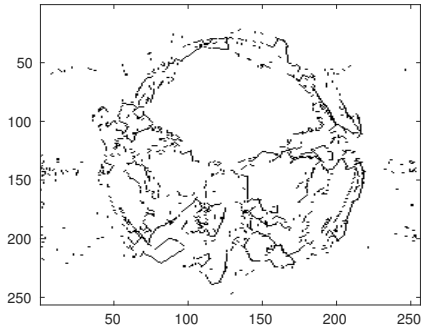
Figura 3.15: Desenvolvimiento de fase del mapa noise.256x256 donde las sub-figuras a, c y e representan cada etapa del algoritmo de Goldstein (búsqueda de residuos, selección de cortes e integración). Las sub-figuras b, d, y f representan los hilos creados por etapa. Cada tono de gris es un hilo ejecutado en paralelo, y los píxeles negros representan residuos o cortes.



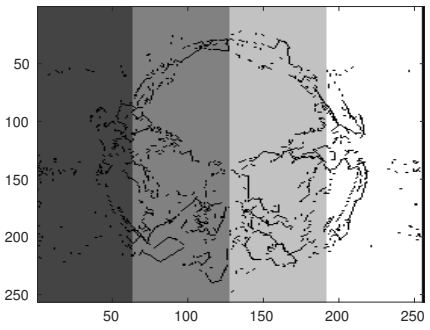
(a) Residuos



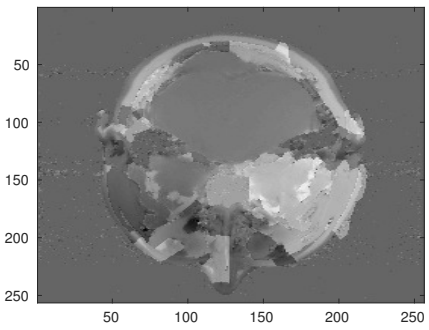
(b) Búsqueda de residuos por bloques



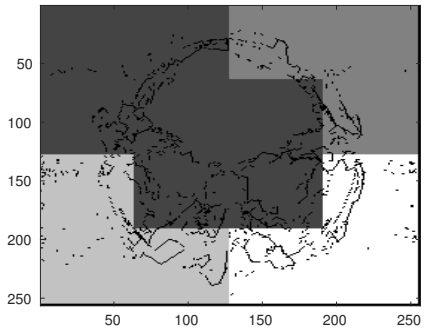
(c) Cortes



(d) Generación de cortes por bloques

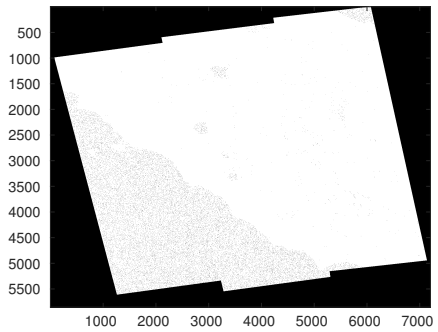


(e) Fase desenvuelta

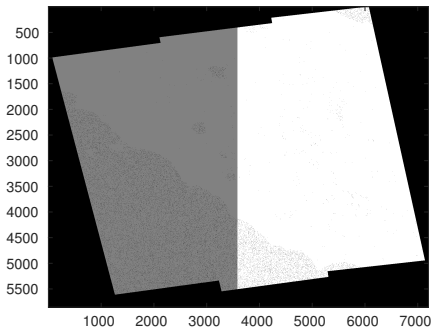


(f) Integración por bloques

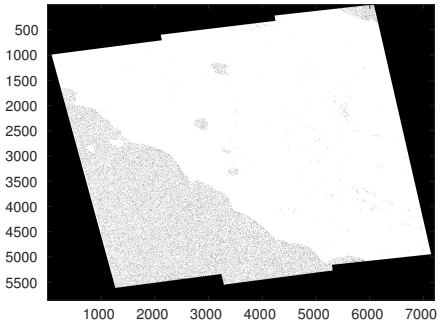
Figura 3.16: Desenvolvimiento de fase del mapa head.256x256 donde las sub-figuras a, c y e representan cada etapa del algoritmo de Goldstein (búsqueda de residuos, selección de cortes e integración). Las sub-figuras b, d, y f representan los hilos creados por etapa. Cada tono de gris es un hilo ejecutado en paralelo, y los píxeles negros representan residuos o cortes.



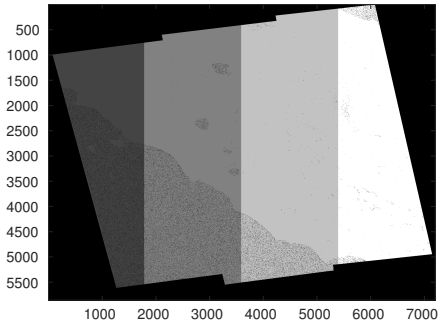
(a) Residuos



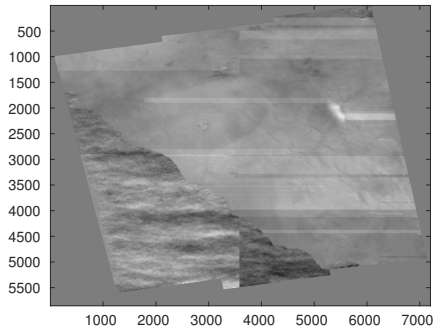
(b) Búsqueda de residuos por bloques



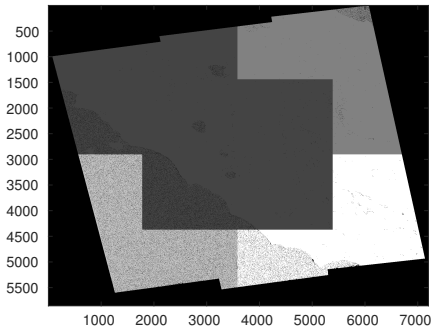
(c) Cortes



(d) Generación de cortes por bloques

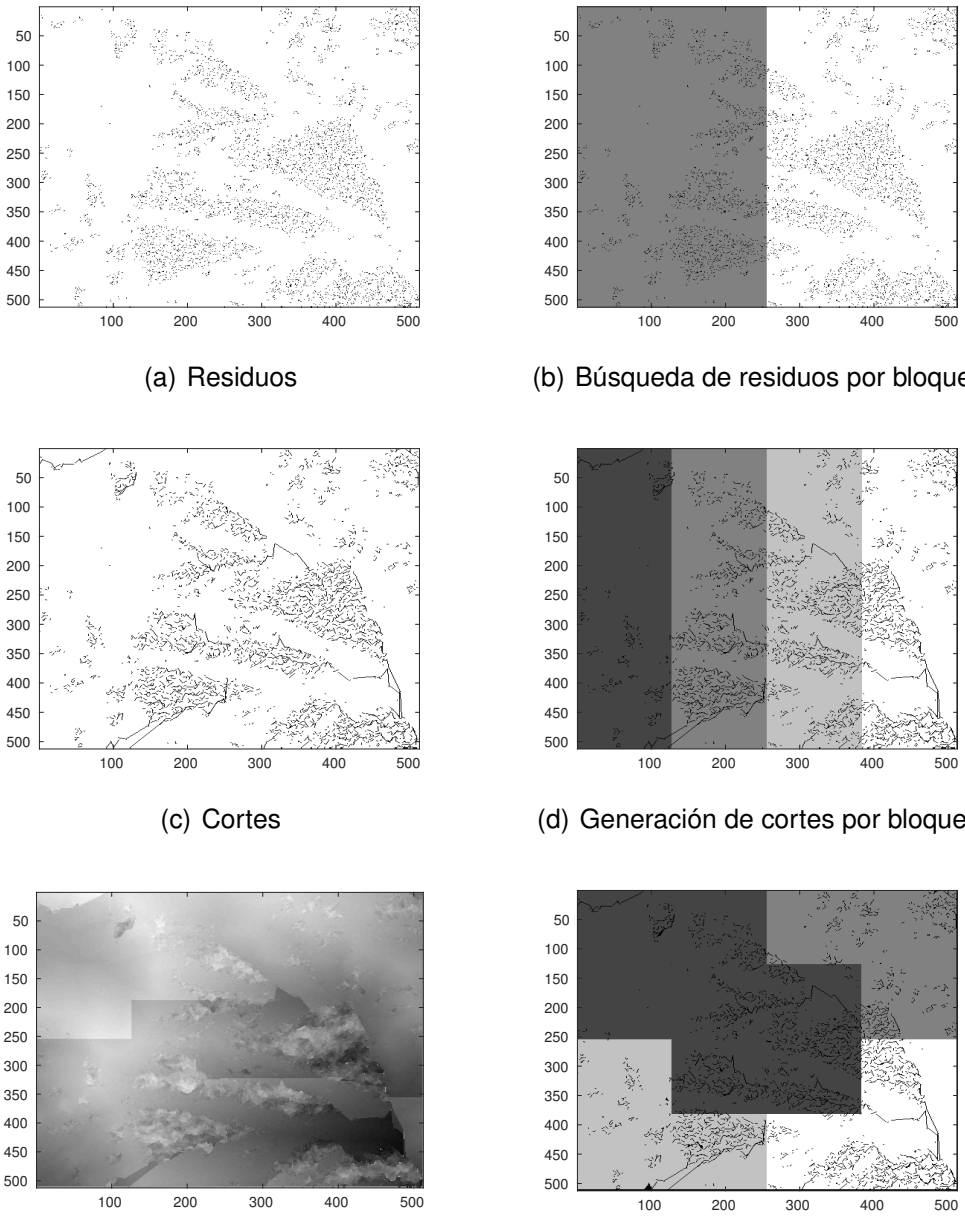


(e) Fase desenvuelta



(f) Integración por bloques

Figura 3.17: Desenvolvimiento de fase del mapa ifsar.5854x7202 donde las sub-figuras a, c y e representan cada etapa del algoritmo de Goldstein (búsqueda de residuos, selección de cortes e integración). Las sub-figuras b, d, y f representan los hilos creados por etapa. Cada tono de gris es un hilo ejecutado en paralelo, y los píxeles negros representan residuos o cortes.



(a) Residuos

(b) Búsqueda de residuos por bloques

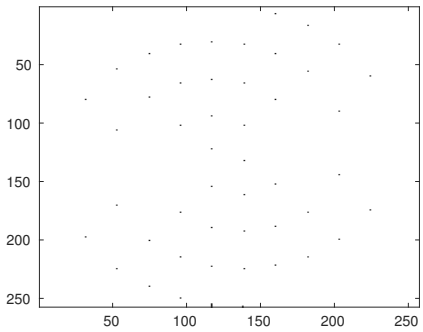
(c) Cortes

(d) Generación de cortes por bloques

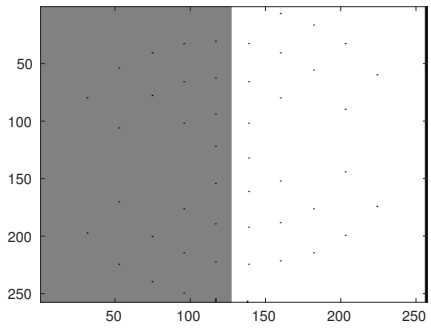
(e) Fase desenvuelta

(f) Integración por bloques

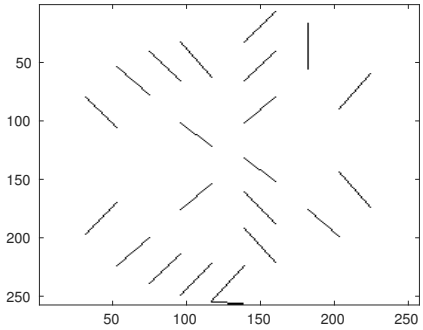
Figura 3.18: Desenvolvimiento de fase del mapa ifsar.512x512 donde las sub-figuras a, c y e representan cada etapa del algoritmo de Goldstein (búsqueda de residuos, selección de cortes e integración). Las sub-figuras b, d, y f representan los hilos creados por etapa. Cada tono de gris es un hilo ejecutado en paralelo, y los píxeles negros representan residuos o cortes.



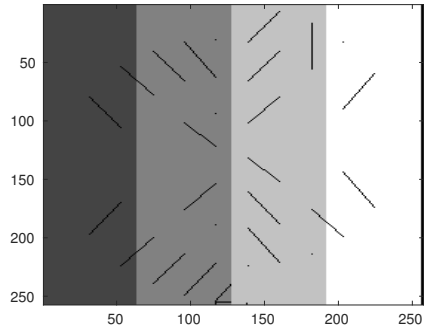
(a) Residuos



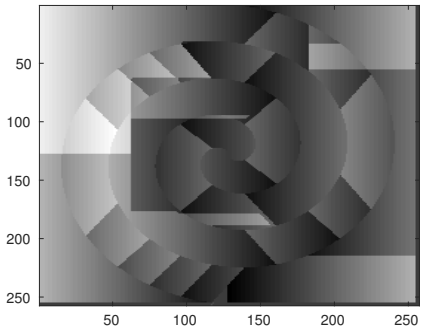
(b) Búsqueda de residuos por bloques



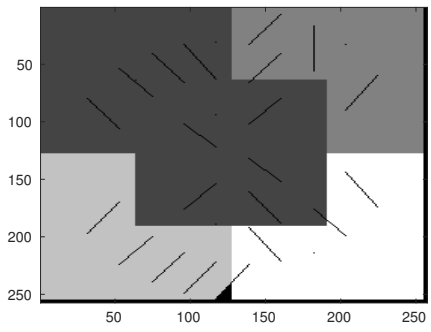
(c) Cortes



(d) Generación de cortes por bloques



(e) Fase desenvuelta



(f) Integración por bloques

Figura 3.19: Desenvolvimiento de fase del mapa spiral.257x257 donde las sub-figuras a, c y e representan cada etapa del algoritmo de Goldstein (búsqueda de residuos, selección de cortes e integración). Las sub-figuras b, d, y f representan los hilos creados por etapa. Cada tono de gris es un hilo ejecutado en paralelo, y los píxeles negros representan residuos o cortes.

Capítulo 4

Optimización del algoritmo guiado por calidad

En este capítulo se plantea una mejora para el algoritmo de desenvolvimiento de fase guiado por mapas de calidad QG (por sus siglas en inglés de *Quality Guided*), por ello se inicia con la descripción completa del algoritmo, se explica de manera detallada la mejora propuesta, y se realizan las pruebas para evidenciar su funcionamiento; por último se muestran los resultados y se presentan las conclusiones.

4.1. Algoritmo guiado por calidad

El algoritmo de desenvolvimiento de fase guiado por mapas de calidad es una técnica ampliamente usada por su sencillez y robustez, además es un método que se encuentra en la categoría de los algoritmos de desenvolvimiento de fase basados en rutas de integración. El algoritmo QG desenvuelve la fase utilizando un mapa de calidad para guiar la ruta de integración; el proceso comienza desde el punto de mayor calidad y continúa hasta los de calidad inferior [23, 29]. Un mapa de calidad es una matriz de las mismas dimensiones que la fase envuelta que asigna valores de calidad a cada uno de los pixeles en el mapa de fase envuelto.

En la actualidad se han propuesto muchos mapas de calidad y estos se cla-

sifican según los recursos utilizados para generarlos; existen dos enfoques para calcular los mapas de calidad [30, 28]:

1. Se puede calcular a partir de los datos de medición brutos tales como: mapa de modulación Q_{MOD}, mapa de fiabilidad Q_{REL}, mapa de amplitud de transformada de Fourier Q_{FT}, método de filtrado de Fourier en ventanas Q_{WFF}, método de crestas de Fourier con ventanas Q_{WFR} y el método de la transformada wavelet Q_{WT}.

2. Se puede calcular a partir de la fase envuelta tales como: mapa de coeficiente de pseudo correlación Q_{PCC}, mapa de variación derivada de fase Q_{PDV}, mapa de la segunda diferencia de fase Q_{SPD}, mapa de amplitud de transformada de Fourier Q_{FT}, método de filtrado de Fourier en ventanas Q_{WFF}, método de crestas de Fourier con ventanas Q_{WFR} y el método de la transformada wavelet Q_{WT}.

El algoritmo QG funciona de la siguiente manera: primero se generan los valores de calidad pixel a pixel y se guardan en una matriz (mapa de calidad) en la misma posición que en la matriz de fase envuelta; enseguida se busca en el mapa de calidad un píxel $p(x, y)$ con el valor más alto y se inserta en una cola de prioridad ordenada por calidad (*lista adjunta*); si la lista adjunta no esta vacía se saca el primer valor (siempre el de mayor calidad), se desenvuelve y se inserta a sus vecinos $p(x, y + 1)$, $p(x, y - 1)$, $p(x + 1, y)$, $p(x - 1, y)$ en la lista adjunta; se ordenan los valores y se repite el proceso hasta que la lista adjunta quede vacía.

El pseudo código algoritmo QG se muestra en el algoritmo 2. El resultado del algoritmo QG depende directamente del mapa de calidad, por lo tanto, el problema se reduce a la elección de un mapa que sea capaz de mitigar las principales causas de error.

Algorithm 2 QG

1: Generar el mapa de calidad de los datos sin procesar.
2: Buscar píxel con la calidad más alta e insertarlo en la lista adjunta.
3: Resolver el primer píxel en la lista adjunta.
4: Actualizar la lista adjunta.
   Sacar el primer píxel de la lista adjunta.
   Insertar los pixeles vecinos envueltos en la lista adjunta.
   Ordenar la lista adjunta según los valores de calidad.
5: Regresar al paso 3 hasta que la lista adjunta se encuentre vacía.
6: Regresar al paso 2 hasta que se procesen todos los pixeles.

En general, el algoritmo QG obtiene buenos resultados, sin embargo cuenta con dos desventajas importantes, la primera es que el tiempo de ejecución es alto y la segunda es que no es robusto con mapas de fase envueltos que contienen altos niveles de ruido. La razón por la cual es lento se debe a la constante necesidad de reordenar los datos de la lista adjunta, ya que al utilizar un mapa de calidad es necesario ordenar de mayor a menor calidad los pixeles que se encuentran en la lista adjunta. Así, cuando la lista adjunta contiene pocos datos el proceso es relativamente rápido, de lo contrario, cuando los datos aumentan el ordenamiento es tardado ya que por cada píxel desenvuelto se insertan cuatro pixeles en la lista adjunta y es necesario reordenar constantemente para asegurar que siempre se saca el píxel de mayor calidad.

En la siguiente sección se proponen dos mejoras para el algoritmo QG. En la primera se usan los residuos del mapa de fase envuelto para mejorar los mapas de calidad basados en gradientes. Con esta estrategia se obtuvo que el algoritmo QG sea robusto contra el ruido. Finalmente, se proponen una estructura de datos más eficiente para implementar la lista adjunta que se basa en listas ligadas.

4.2. Mapas de calidad con residuos

Los algoritmos para el desenvolvimiento de fase por rutas de integración funcionan de manera como el algoritmo de Goldstein y el algoritmo QG. En ambos casos se evitan los pixeles que se consideran problemáticos, con el fin de propagar errores en el proceso de integración. Se desenvuelven primero los pixeles más fiables y por último los pixeles problemáticos. En el algoritmo de Goldstein se calculan las sumas de las diferencias de fase de un píxel $p(x, y)$ con respecto a sus vecinos $p(x, y+1)$, $p(x, y-1)$, $p(x+1, y)$, $p(x-1, y)$ para detectar inconsistencias en el mapa de fase envuelto, a estas inconsistencias se les llama residuos. La búsqueda de residuos tiene como propósito trazar cortes que unan un residuo positivo a uno negativo para evitar que se propaguen las discontinuidades en el proceso de integración. De manera similar, al algoritmo QG asigna valores que determinan la calidad de cada píxel para primero desenvolver los de mayor calidad y así evitar que se desenvuelvan los pixeles de baja calidad [27]; en ambos algoritmos se busca evitar los pixeles que causan error en el proceso de integración, sin embargo, no existe relación directa entre los valores de calidad de un pixel y un residuo.

El mapa de calidad basado en gradientes máximos se define como

$$z_{i,j} = \mathrm{máx}\{\mathrm{máx}\{|\Delta^x_{i,j}|\}, \mathrm{máx}\{|\Delta^y_{i,j}|\}\} \tag{4.1}$$

donde $\Delta^x_{i,j}$ y $\Delta^y_{i,j}$ son la derivadas parciales de la fase en la dirección x, y y la función $\mathrm{máx}$ se calcula en una vecindad de $k \times k$ pixeles.

Se encontró experimentalmente que para un mapa de fase envuelto y un mapa de calidad basado en máximos gradientes, los pixeles de más baja calidad no siempre corresponden a residuos en el mapa de fase dado. Por el contrario, en muchas ocaciones corresponden a pixeles con calidad alta. Esto significa que en el proceso de integración del algoritmo QG se resolverán primero los pixeles que corresponden a residuos, propagando de este modo errores en el proceso de

integración. La Figura 4.1 muestra un mapa de fase envuelto con la distribución peaks, los residuos identificados y un mapa de calidad basado en gradientes.

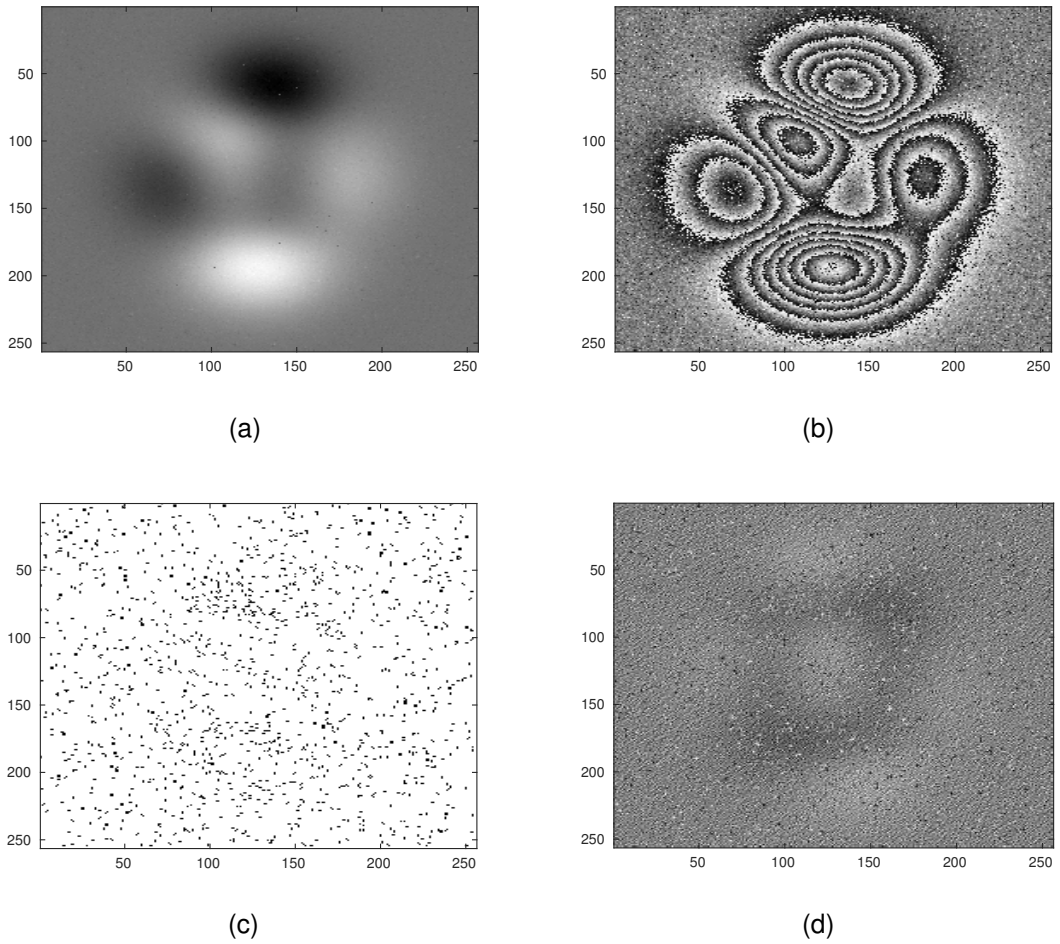


Figura 4.1: (a) Mapa de fase desenvuelto (fase ideal), (b) mapa de fase envuelta con ruido, (c) residuos identificados y (d) mapa de calidad de calculado por el método de gradiente de fase máxima donde los pixeles más obscuros representan los valores de menor calidad y lo pixeles claros representan mayor calidad.

En la Figura 4.1 se muestran los residuos y mapa de calidad de peaks.256×256 donde se puede ver que el número de residuos es menor a los pixeles de baja calidad, tomando en cuenta que la calidad máxima es 1 y 0 es la mínima. Es posible variar el umbral para determinar si todos los residuos están marcados con 0 en el mapa de calidad. En la Tabla 4.1 se pueden ver los resultados de variar el umbral

y es posible determinar que no existe relación entre residuos y valor de calidad mínimo, lo cual quiere decir que en el algoritmo QG se procesan residuos cuando la calidad se encuentra por debajo de la media, por lo tanto, si el mapa de fase contiene un alto nivel de ruido, se resolverán residuos al inicio del proceso de integración provocando malos resultados; por el contrario, cuando se ubican los residuos y se marcan con el valor de calidad mas bajo el resultado mejora de manera significativa, esto se puede ver en la Figura 4.2.

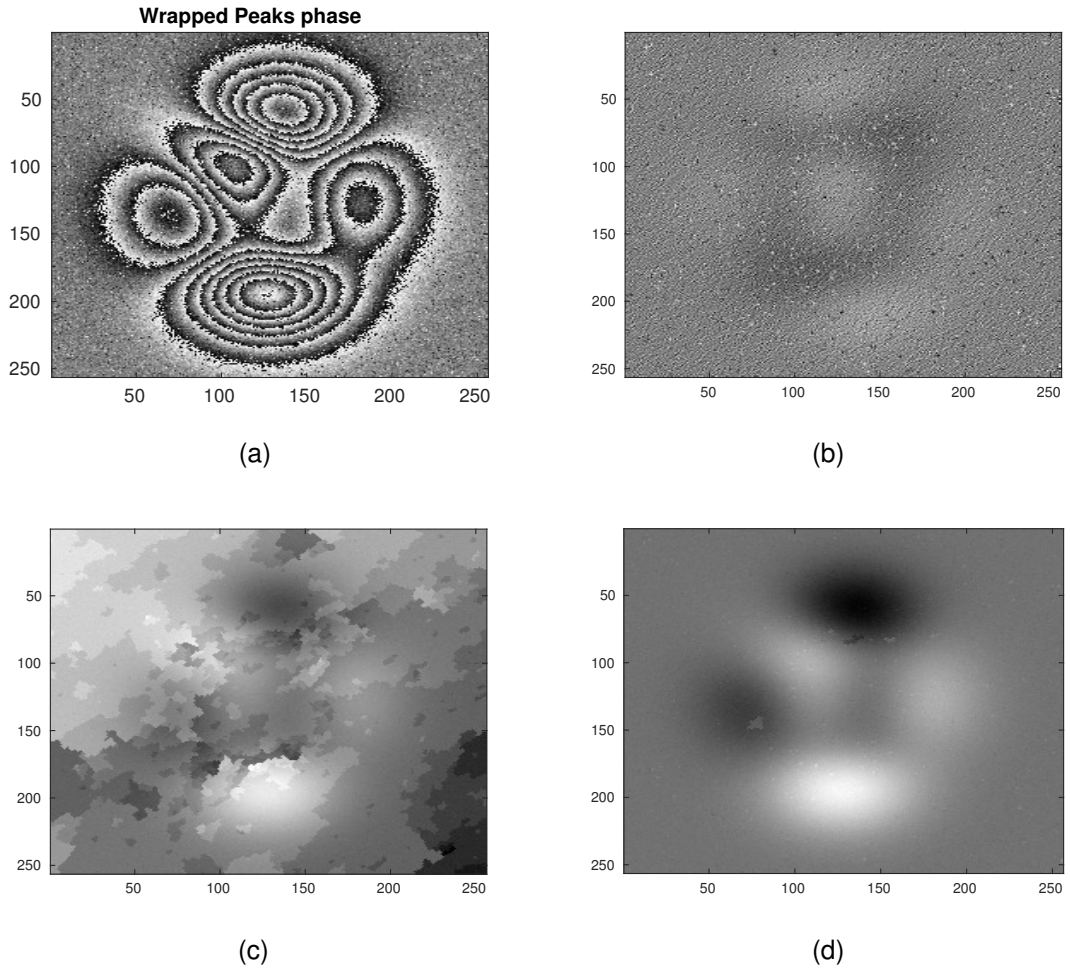


Figura 4.2: (a) Fase envuelta de la función peaks con un nivel de ruido SNR=2, (b) mapa de calidad + residuos, (c) mapa desenvuelto usando el algoritmo QG con el mapa de calidad de gradientes máximos y (d) mapa de fase desenvuelto usando el algoritmo QG con el mapa de calidad en (b).

Tabla 4.1: Ejemplo que muestra como cambia el número de residuos identificados y residuos ignorados usando un umbral para el mapa de fase envuelto peaks.256×256. Los valores de calidad son calculados por el método de máximo gradiente.

Umbral	Residuos de baja calidad	Residuos ignorados
0.25	110	2525
0.3	252	2383
0.35	443	2192
0.4	681	1954
0.45	999	1636
0.5	1309	1326

4.3. Lista adjunta usando listas ligadas

El desempeño del algoritmo QG depende principalmente de la inserción de pixeles en la lista adjunta. En términos teóricos la complejidad de la inserción en una lista ordenada con N elementos se lleva a cabo en a lo más N iteraciones. Es evidente que mientras menos elementos se encuentren insertados en la lista adjunta, menor será el tiempo de inserción. Esta observación fue tomada en cuenta en [31] donde se utiliza una estrategia de partición por intervalos de calidad. En la estrategia de partición, el rango de valores de calidad $q_r = q_{max} - q_{min}$ se divide en N intervalos donde q_{min} y q_{max} son los valores de calidad mínimo y máximo en el mapa de calidad, respectivamente. Si el rango q_r se divide equitativamente entonces el ancho de cada intervalo es $w = q_r/N$. De este modo, se creó una nueva estructura de datos, la cual, consiste en un arreglo de longitud N donde en cada celda desde $k = 1, \ldots, N$ existe un índice a una lista adjunta. Así, para cada $k = 1, \ldots, N$, solo los pixeles con valores de calidad en el intervalo $[w(k-1), wk]$ serán insertados en la k-ésima lista adjunta. De hecho, para cualquier valor de calidad q su índice k correspondiente se puede calcular por medio de la ecuación $k = \lfloor (q - q_{min})/w \rfloor + 1$.

La Figura 4.3 se muestra la estructura de datos utilizado por el método de partición por valores de calidad. Los autores en [31] proponen implementar una lista ligada con inserción ordenada para cada lista adjunta. Sin embargo, el desempeño

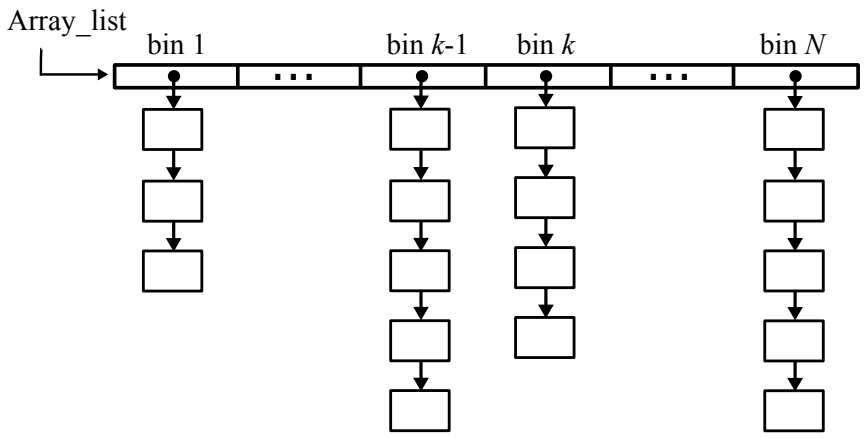


Figura 4.3: Estructura de datos utilizada por el método de partición de valores de calidad. En cada celda de la lista principal se tiene un índice a una lista adjunta.

de una lista ligada con inserción ordenada no mejora sustancialmente.

En este trabajo se propone utilizar nodos pivotes en cada lista ligada de la estrategia de partición de valores de calidad. Un pivote corresponde al nodo central en una lista ligada, este nodo ayuda a reducir a la mitad la complejidad de la inserción. Por ejemplo, si se desea insertar un pixel p con un valor de calidad q, entonces se compara el valor de calidad q con el nodo central q_c; en caso de que $q > q_c$ entonces el pixel p debe ser insertado en la segunda mitad de la lista adjunta, en otro caso, se debe insertar en la primera mitad de la lista. Este procedimiento reduce a la mitad la búsqueda para encontrar la posición adecuada de un pixel en la lista adjunta. La Figura 4.4 muestra la estructura de datos usada por la estrategia de partición de valores de calidad usando pivotes.

En la Tabla 4.2 se muestra una comparación de los tiempo de ejecución usando listas ligadas y listas ligadas con pivote. Como puede verse la listas ligadas con pivote tienen mejor desempeño que las listas ligadas tradicionales. En el primer caso del mapa de fase peaks de dimensión 256$\times$256, la aceleración es de 1.46 y para el mapa de fase de dimensión 2048$\times$2048 la aceleración es de 1.8 más rápidos. Cabe mencionar que la estructura de datos de lista ligada con pivote que se propone en esta investigación, hasta donde conocemos de la literatura, es la primera vez que se muestra una mejora a la inserción ordenada usando una lista

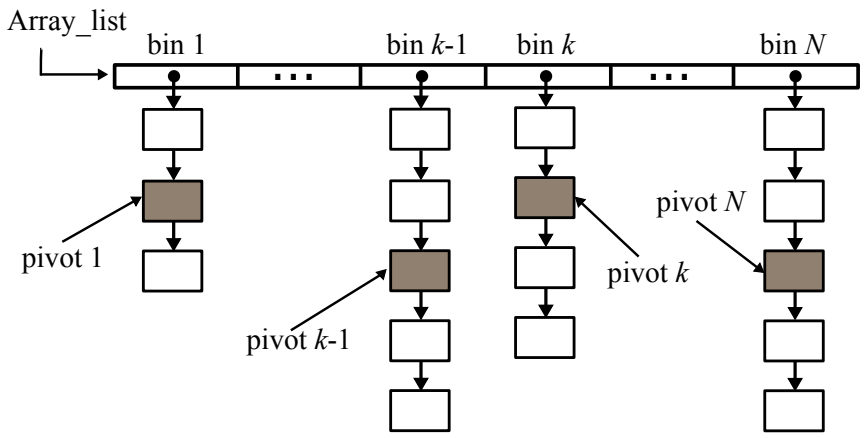


Figura 4.4: Estructura de datos utilizada por el método de partición de valores de calidad usando pivotes. En cada celda de la lista principal se tiene un índice a una lista adjunta.

ligada.

Tabla 4.2: Comparación de los tiempos de ejecución del algoritmo QG usando la estrategia de partición de valores de calidad usando listas ligadas y listas ligadas con pivote. Los tiempos de ejecución están dados en milisegundos.

Fase	Dimensión	Lista-Ligada	Lista-Ligada pivote
peaks	256×256	338.0	230.0
peaks	2048×2048	7523.0	4158.0

4.4. Conclusiones

Se demostró experimentalmente que los residuos ayudan a mejorar el mapa de calidad para obtener mejores resultados en el algoritmo QG. Los resultados obtenidos en este capítulo se comparan con los resultados obtenidos con el algoritmo de Goldstein. También se mejoró el desempeño en tiempo de ejecución de la estructura de datos de listas adjunta usando listas ligadas con pivote. En base a los resultados obtenidos con nuevas estructuras de datos se pueden estudiar métodos de paralelización para acelerar la inserción ordenada.

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo se presenta una versión paralela del algoritmo de Goldstein usando una arquitectura multi-núcleo, y con ello se obtuvieron resultados competitivos con respecto a la versión original. Además, se propuso una mejora en la construcción de los mapas de calidad para el algoritmo de desenvolvimiento guiado por calidad haciendo uso del concepto de residuos donde los resultados obtenidos son tan buenos en comparación del algoritmo de Goldstein.

El problema del desenvolvimiento de fase es un tema ampliamente estudiado y en la actualidad existen algoritmos que son capaces de resolver mapas de fase que presentan las principales causas de error (ruido, cambios abruptos de fase y bajo muestreo). Sin embargo, no existe un algoritmo que resuelva todas las causas de error, por lo que es necesario que sea desarrollado un algoritmo capaz de resolver cualquier mapa de fase envuelto, sin importar las causas de error con las que pueda contar. Un paso siguiente de esta investigación es la implementación de una versión paralela de un algoritmo robusto de alto desempeño que haga uso de las características de los algoritmos existentes y que aproveche las estrategias de partición propuestas en este trabajo.

Referencias

1. Backoach, O., Kariv, S., Girshovitz, P., y Shaked, N. T. (2016). Fast phase processing in off-axis holography by cuda including parallel phase unwrapping. *Optics express*, *24*(4), 3177–3188.

2. Bioucas-Dias, J. M., y Valadao, G. (2007). Phase unwrapping via graph cuts. *IEEE Transactions on Image processing*, *16*(3), 698–709.

3. Blinder, D., Ottevaere, H., Munteanu, A., y Schelkens, P. (2016). Efficient multiscale phase unwrapping methodology with modulo wavelet transform. *Optics express*, *24*(20), 23094–23108.

4. Chen, C. W. (2001). *Statistical-cost network-flow approaches to two-dimensional phase unwrapping for radar interferometry* (Tesis Doctoral no publicada). Stanford University.

5. De Souza, J., Oliveira, M., y dos Santos, P. (2015). Branch-cut algorithm for optical phase unwrapping. *Optics letters*, *40*(15), 3456–3459.

6. Du, G., Wang, M., Zhou, C., Si, S., Li, H., Lei, Z., y Li, Y. (2016). An algorithm to increase the residues of wrapped-phase in spatial domain. *arXiv preprint arXiv:1604.07231*.

7. Feng, X., Jicang, W., Lei, Z., y Xiaoling, L. (2007). A new method about placement of the branch cut in two-dimensional phase unwrapping. En *Synthetic aperture radar, 2007. apsar 2007. 1st asian and pacific conference on* (pp. 755–759).

8. Flynn, M. J. (1972). Some computer organizations and their effectiveness. *IEEE transactions on computers*, *100*(9), 948–960.

9. Goldstein, R. M., Zebker, H. A., y Werner, C. L. (1988). Satellite radar interferometry: Two-dimensional phase unwrapping. *Radio science*, *23*(4), 713–720.

10. Guo, Y., Chen, X., y Zhang, T. (2014). Robust phase unwrapping algorithm based on least squares. *Optics and Lasers in Engineering*, *63*, 25 - 29. Descargado de http://www.sciencedirect.com/science/article/pii/S0143816614001523 doi: http://dx.doi.org/10.1016/j.optlaseng.2014.06.007

11. Gutmann, B., y Weber, H. (2000). Phase unwrapping with the branch-cut method: role of phase-field direction. *Applied optics*, *39*(26), 4802–4816.

12. Hedley, M., y Rosenfeld, D. (1992). A new two-dimensional phase unwrapping algorithm for mri images. *Magnetic resonance in medicine*, *24*(1), 177–181.

13. Herráez, M. A., Burton, D. R., Lalor, M. J., y Gdeisat, M. A. (2002, Dec). Fast two-dimensional phase-unwrapping algorithm based on sorting by reliability following a noncontinuous path. *Appl. Opt.*, *41*(35), 7437–7444. Descargado de http://ao.osa.org/abstract.cfm?URI=ao-41-35-7437 doi: 10.1364/AO.41.007437

14. Huang, H. Y. H., Tian, L., Zhang, Z., Liu, Y., Chen, Z., y Barbastathis, G. (2012, Jun). Path-independent phase unwrapping using phase gradient and total-variation (tv) denoising. *Opt. Express*, *20*(13), 14075–14089. Descargado de http://www.opticsexpress.org/abstract.cfm?URI=oe-20-13-14075 doi: 10.1364/OE.20.014075

15. Huang, Q., Zhou, H., Dong, S., y Xu, S. (2015). Parallel branch-cut algorithm based on simulated annealing for large-scale phase unwrapping. *IEEE Transactions on Geoscience and Remote Sensing*, *53*(7), 3833–3846.

16. Jeught, S. V. d., Sijbers, J., y Dirckx, J. J. (2015). Fast fourier-based phase unwrapping on the graphics processing unit in real-time imaging applications. *Journal of Imaging*, *1*(1), 31–44.

17. Karout, S. A., Gdeisat, M. A., Burton, D. R., y Lalor, M. J. (2007). Residue vector, an approach to branch-cut placement in phase unwrapping: theoretical study. *Applied optics*, *46*(21), 4712–4727.

18. Karpinsky, N., Hoke, M., Chen, V., y Zhang, S. (2014). High-resolution, real-time three-dimensional shape measurement on graphics processing unit. *Optical Engineering*, *53*(2), 024105–024105.

19. Lu, Y., Wang, X., y He, G. (2005). Phase unwrapping based on branch cut placing and reliability ordering. *Optical Engineering*, *44*(5), 055601–055601.

20. Marcelino, M. H. (s.f.). Innovación y computación paralela.

21. Matias, G. R. V. (2006). *Radar interferometry: 2d phase unwrapping via graph cuts* (Tesis Doctoral no publicada). INSTITUTO SUPERIOR TÉCNICO.

22. Petković, T., Pribanić, T., y Đonlić, M. (2017). Temporal phase unwrapping using orthographic projection. *Optics and Lasers in Engineering*, *90*, 34–47.

23. Ramji, M., y Ramesh, K. (2010). Adaptive quality guided phase unwrapping algorithm for whole-field digital photoelastic parameter estimation of complex models. *Strain*, *46*(2), 184–194.

24. Rubio Díez, F. (2004). *Programación funcional paralela eficiente en eden* (Tesis Doctoral no publicada). Universidad Complutense de Madrid, Servicio de Publicaciones.

25. Shalem, I., y Yavneh, I. (2008). A multilevel graph algorithm for two dimensional phase unwrapping. *Computing and Visualization in Science*, *11*(2), 89–100.

26. Takajo, H., y Takahashi, T. (1988). Least-squares phase estimation from the phase difference. *JOSA A*, *5*(3), 416–425.

27. Wang, H., Weaver, J. B., Perreard, I. I., Doyley, M. M., y Paulsen, K. D. (2011). A three-dimensional quality-guided phase unwrapping method for mr elastography. *Physics in medicine and biology*, *56*(13), 3935.

28. Zhang, S., Li, X., y Yau, S.-T. (2007). Multilevel quality-guided phase unwrapping algorithm for real-time three-dimensional shape reconstruction. *Applied optics*, *46*(1), 50–57.

29. Zhang, Y., Wang, S., Ji, G., y Dong, Z. (2014). An improved quality guided phase unwrapping method and its applications to mri. *Progress In Electromagnetics Research*, *145*, 273–286.

30. Zhao, M., Huang, L., Zhang, Q., Su, X., Asundi, A., y Kemao, Q. (2011). Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies. *Applied optics*, *50*(33), 6214–6224.

31. Zhao, M., y Kemao, Q. (2014, Jun). Quality-guided phase unwrapping implementation: an improved indexedinterwoven linked list. *Appl. Opt.*, *53*(16), 3492–3500. Descargado de `http://ao.osa.org/abstract.cfm?URI=ao-53-16-3492` doi: 10.1364/AO.53.003492

32. Zheng, D., y Da, F. (2011). A novel algorithm for branch cut phase unwrapping. *Optics and Lasers in Engineering*, *49*(5), 609–617.