Algoritmo rápido para recuperar fase de un solo interferograma con franjas abiertas y cerradas.

por

Oscar S. Dalmau Cedeño

I.S.P. de Manzanillo (1989)

Sometida a revisión al Departamento de Ciencias de la Computación
en el cumplimiento parcial de los requisitos para obtener el grado de

Maestro en Ciencias de la Computación y Matemáticas Industriales

en el

CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS A.C.,

Agosto 2006

Firma del autor .....
Departamento de Ciencias de la Computación
Agosto de 2006

Certificado por .....
Dr. Mariano J.J. Rivera Meraz
Grupo de Computación, Investigador Titular A
Director de Tesis

Aceptado por .....
Dr. Mariano J.J. Rivera Meraz
Coordinador de la Maestría en Ciencias de la Computación

II

1

Algoritmo rápido para recuperar fase de un solo interferograma con franjas abiertas y cerradas.

por

Oscar S. Dalmau Cedeño

Sometida a revisión al Departamento de Ciencias de la Computación
en Agosto de 2006, en el cumplimiento parcial de los
requisitos para obtener el grado de
Maestro en Ciencias de la Computación y Matemáticas Industriales

Resumen

El presente trabajo se desarrolla dentro del área del Tratamiento Digital de Imágenes, y está relacionado con el problema de recuperación de la fase de un patrón de franjas.

En el mismo se propone un algoritmo rápido para la recuperación de fase de un solo interferograma con franjas abiertas y cerradas. El algoritmo propuesto consta de dos etapas, en la primera se construye una fase inicial estimada a partir de filtros de cuadratura simple y haciendo uso de la orientación de las franjas. Aquí se define una estructura que guía el proceso de construcción de la fase inicial. La segunda etapa consiste en un refinamiento de la fase inicial, para lo cual se usa un proceso basado en pirámides gaussianas.

Director de Tesis: Dr. Mariano J.J. Rivera Meraz
Título: Grupo de Computación, Investigador Titular A

Dedicatorias

A mi familia,
A mis queridos hijos,
A mis padres y hermanos.

Agradecimientos

Quiero agradecer al Dr. Salvador Botello Rionda por su apoyo, desde mi largo y pesaroso camino a CIMAT y durante mi estancia aquí.

También quiero agradecer al Dr José Luis Marroquín Zaleta por su apoyo moral, su confianza y su ejemplo como maestro e investigador.

Un agradecimiento especial a mi Tutor de Tesis Dr. Mariano J.J Rivera Meraz por su ingenio, talento y amistad que me guiaron y permitió que saliera a la luz este trabajo.

A todo el colectivo de profesores del CIMAT y en especial al colectivo de cómputo.

Al personal de la biblioteca por su atención y amabilidad, a Manuel y a Ricardo.

A Lourdes Navarro y su equipo de trabajo de quienes siempre recibí su apoyo, al personal de Eventos a José Castro, a Luis Omar Avalos y a Angel Carrillo, a Berenice y a Lolita, y a todo el personal administrativo de CIMAT.

A mis compañeros de estudio, a Teresa E. Alarcón, Humberto Esqueda, Jabneel Maldonado, Patricia Bautista, Pedro Pablo Mayorga, Leonel Ramírez, Mauricio Ruiz, Omar Ocegueda, Lorena Avendaño y Dan-El Vila, con los que compartí momentos agradables que me hicieron sentir bien.

Finalmente agradezco el apoyo económico recibido por parte del CIMAT y CONACYT, a través de los proyectos Conacyt No. 46270 y No. 40721-Y.

Índice general

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción y alcance del trabajo

Cuando dos haces de luz coherentes se cruzan interfieren y se afecta a la distribución de intensidades resultante, la coherencia de un haz de luz indica que los rayos (fotones) de luz que la conforman están oscilando en fase. Por otro lado, la coherencia de dos haces expresa hasta qué punto están en fase sus ondas. Si la relación de fase cambia de forma rápida y aleatoria, los haces son incoherentes. Si dos trenes de ondas son coherentes y el máximo de una onda coincide con el máximo de otra, ambas ondas se suman de forma constructiva produciendo en ese punto una intensidad mayor que si los dos haces no fueran coherentes. Si son coherentes y el máximo de una onda coincide con el mínimo de la otra, ambas ondas se suman de forma destructiva anulándose entre sí, parcial o totalmente, con lo que la intensidad disminuirá. A la interacción constructiva y destructiva entre ondas coherentes es a lo que se llama *fenómeno de interferencia*.

Cuando las ondas son coherentes, puede formarse un diagrama de interferencia formado por franjas oscuras y claras. Dicho diagrama recibe el nombre de *interferograma*.

Al versátil médico inglés Thomas Young (1773-1829), quien realizó varios aportes al área de la mecánica, se debe el primer experimento donde muestra un diagrama de interferencias en 1801. A él se deben varios aportes en el campo de la teoría ondulatoria, en particular estableció por primera vez la fórmula para la velocidad de propagación de la onda, también trabajó en elasticidad e introdujo el concepto de energía en su significado actual, etc.

Los *interferogramas* o las *imágenes con patrones de franjas* han encontrado muchas aplicaciones, especialmente en el campo de la óptica, la interferometría, en las imágenes acústicas y de resonancia magnética, en el estudio de procesos sísmicos etc, de ahí su importancia.

Una imagen con patrones de franjas se puede modelar a través de la ecuación siguiente:

$$I(r) \quad = \quad a(r) + b(r)\cos[\phi(r)] + \eta(r), \quad r \in \mathcal{L}; \tag{1.1}$$

donde $\mathcal{L}$ define la retícula de la imagen, r denota la posición de un pixel en la retícula $\mathcal{L}$, $a(r)$ y $b(r)$ son funciones que representan la intensidad del fondo y el contraste local del patrón de franjas respectivamente, $\phi(r)$ es la fase y $\eta(r)$ es cierto ruido, que en general se asume que es ruido blanco.

Uno de los elementos fundamentales de un interferograma es la fase. En relación con la importancia de la fase en una señal se han publicado algunos trabajos, ver [1] y [2]. La determinación la fase se ha convertido en un campo de aplicación del procesamiento digital de señales (o imágenes) que en los últimos años ha atraído la atención de investigadores en esta rama de la ciencia.

Al ser un problema del tratamiento digital de imágenes, le son inherentes las dificultades propias de esta rama, es decir, recuperar la fase es un problema mal planteado, por lo que para hallar una solución única será necesario establecer algunas consideraciones. Además se incluyen las siguientes dificultades:

- Ambigüedad de la fase en 2π , producto a la periodicidad de coseno.
- Ambigüedad del signo, por la paridad del coseno.
- En casos reales, la presencia de algún tipo de ruido.

Muchas son las técnicas que se han empleado para la recuperación de fase, ver Capítulo 2, y aun hoy en día se siguen desarrollando otras, desde el análisis de Fourier hasta técnicas de regularización.

Alcance del trabajo

El objetivo de este trabajo es presentar un algoritmo para recuperar la fase de un solo patrón de franjas de forma aproximada, cuyo modelo de observaciones se corresponde con la ecuación 1.1. En particular se supondrá que el patrón de franjas está normalizado, entonces el modelo que se tiene es el siguiente:

$$I(r) \approx \cos[\phi(r)] + \eta(r), \quad r \in \mathcal{L}; \tag{1.2}$$

El algoritmo que se propone consta de 2 etapas: en la primera se estima una fase inicial usando un método que se basa en el uso de filtros de cuadratura, además se usa la información de orientación local y se define un nuevo ente en la imagen que guiará el proceso de formación de la fase inicial, la segunda etapa consiste en un refinamiento para lo que se propone usar pirámides gaussianas.

Dentro de las características principales del método propuesto está su eficiencia computacional, logrando recuperar una fase aproximada en menos de 1 minuto, y en muchos casos en alrededor de 10 segundos.

Estructura del trabajo

El trabajo consta de 6 capítulos: El capítulo 2 se dedica al estudio de algunas técnicas de obtención de la fase entre las que se encuentran la interferometría de fase escalonada, un método de recuperación de fase usando filtros de cuadratura, propuesto originalmente por Takeda et al [3], también se incluye una técnica para el análisis de un solo patrón de franjas a través de un método de regularización (Regularized Phase Tracking), desarrollada por Servin et al [4] (1997) , otros trabajos relacionados se pueden ver en Servin et al [5], [6] y en Marroquín et al [7] y finalmente se explica una técnica de refinamiento de la fase, Rivera [8]. En el capítulo 3 se ven algunas técnicas de desenvolvimiento de fase, se comienza con un método de desenvolvimiento en 1D, luego se da la formulación del problema de desenvolvimiento en 2D, se explican algunas técnicas de desenvolvimiento haciendo hincapié en los métodos de regularización entre ellos el Half Quadratic, Rivera y Marroquín [9], en el capítulo 4 se dan elementos sobre orientación local, algunas de las técnicas que se han empleado en la determinación de la orientación local en una imagen entre las

que se encuentra el Tensor de Inercia [10], [11], [12], [13], también se introduce un ente en la imagen que permitirá guiar la construcción de una fase estimada y que será el objetivo principal de este trabajo, en el capítulo 5 se propone un algoritmo para el análisis de un solo interferograma, en este capítulo se hace una explicación de las partes principales del algoritmo, la exposición se realiza a través de un ejemplo y en cada paso se usan los elementos teóricos ya vistos en los capítulos anteriores, o se hace una precisión en los elementos propios del algoritmo que no han sido vistos con anterioridad, al final de este capítulo se muestran otros experimentos con el algoritmo tanto en patrones con franjas abiertas como con franjas cerradas, en el último capítulo se muestran cuales son limitaciones del algoritmo propuesto así como algunas de las estrategias que se pueden seguir para perfeccionarlo.

Capítulo 2

Métodos para la obtención de la Fase

2.1. Interferometría de fase escalonada (Phase Step Interferometry, PSI)

La **Interferometría de Fase Escalonada (PSI)** consiste en una serie de interferogramas (imágenes con patrones de franjas), de los cuales se toma uno como referencia [14] y a partir de este se obtienen los demás. Dicha Fase es codificada mediante la variación de las intensidades de los patrones, de este modo, la fase puede ser recuperada a través de una operación punto a punto.

El desarrollo de esta técnica se remonta a los trabajos de Carré (1966); trabajos posteriores aparecieron con Crane(1969), Brunning et al. (1974), Hardy et al. (1977), Brunning (1978), Massie et al. (1979), Malacara (1990), Greivenkamp y Brunning (1991). Esta técnica ha recibido muchos nombres entre los cuales se encuentran: Interferometría para medición de la fase, interferometría en tiempo real, etc.

Una de las técnicas más populares para la obtención de la serie de patrones de franjas es la introducción de un paso de fase, así podemos encontrar algoritmos de 3 pasos, 4 pasos, etc. La característica común en estos algoritmos es que se requiere de una serie de interferogramas, los cuales se obtienen a partir de una fase de referencia. A partir de la serie de interferogramas, es posible obtener la fase modulo $2 * \pi$ (*Fase*

envuelta), la cual presenta algunas discontinuidades. El paso final consiste en eliminar las discontinuidades, para de este modo obtener la *Fase Desenvuelta*.
En este trabajo se tendrá el modelo de observación siguiente:

$$I_i(r) = a(r) + b(r)\cos[\phi(r) + \delta_i]; \quad r = (x, y), \tag{2.1}$$

Con

$$\delta_i = \beta + (i-1)\alpha; \quad i = 1, 2, ..., n. \tag{2.2}$$

Donde β es el offset y α es el tamaño del paso. El valor del offset es con respecto a un marco de referencia global, dado que en interferometría solo es relevante la fase relativa, el valor del offset no es relevante y se deja solo por para facilitar la manipulación algebraica.

2.1.1. PSI de 3 pasos

Para el algoritmo de 3 pasos, se tienen 3 imágenes con patrones de franjas. Este caso puede ser resuelto considerando pasos de igual tamaño, además, sin pérdida de generalidad se puede asumir que $\beta = -\alpha$. Entonces $\delta_i = -\alpha, 0, \alpha$ para $i = 1, 2, 3$; tenemos:

$$I_1(r) = a(r) + b(r)\cos[\phi(r) + \delta_1], \tag{2.3}$$
$$I_2(r) = a(r) + b(r)\cos[\phi(r) + \delta_2], \tag{2.4}$$
$$I_3(r) = a(r) + b(r)\cos[\phi(r) + \delta_3]. \tag{2.5}$$

Se tiene un sistema de 3 ecuaciones con 3 incógnitas $a(r), b(r)$ y $\phi(r)$, de las cuales sólo interesa determinar $\phi(r)$. Sustituyendo δ_i tenemos que:

$$I_1(r) = a(r) + b(r)\cos[\phi(r) - \alpha], \tag{2.6}$$
$$I_2(r) = a(r) + b(r)\cos[\phi(r)], \tag{2.7}$$
$$I_3(r) = a(r) + b(r)\cos[\phi(r) + \alpha]. \tag{2.8}$$

Luego aplicando identidades para el coseno de la suma y la diferencia en las ecuaciones

2.6 y 2.8:

$$I_1(r) = a(r) + b(r)\cos[\phi(r)]\cos(\alpha) + b(r)\sin[\phi(r)]\sin(\alpha), \quad (2.9)$$
$$I_2(r) = a(r) + b(r)\cos[\phi(r)], \quad (2.10)$$
$$I_3(r) = a(r) + b(r)\cos[\phi(r)]\cos(\alpha) - b(r)\sin[\phi(r)]\sin(\alpha). \quad (2.11)$$

Ahora se suman y restan las ecuaciones 2.9 y 2.11, y se multiplica 2.10 por 2, para obtener:

$$I_1(r) + I_3(r) = 2a(r) + 2b(r)\cos[\phi(r)]\cos(\alpha), \quad (2.12)$$
$$I_1(r) - I_3(r) = 2b(r)\sin[\phi(r)]\sin(\alpha), \quad (2.13)$$
$$2I_2(r) = 2a(r) + 2b(r)\cos[\phi(r)]. \quad (2.14)$$

Se resta la ecuación 2.14 de 2.12:

$$2I_2(r) - I_1(r) - I_3(r) = 2b(r)\cos[\phi(r)][1 - \cos(\alpha)], \quad (2.15)$$
$$I_1(r) - I_3(r) = 2b(r)\sin[\phi(r)]\sin(\alpha). \quad (2.16)$$

Finalmente se dividen miembro a miembro las ecuaciones anteriores quedando determinado $\phi(r)$ por la ecuación:

$$\phi(r) = \arctan\left[\frac{[1-\cos(\alpha)]}{\sin(\alpha)}\frac{I_1(r) - I_3(r)}{2I_2(r) - I_1(r) - I_3(r)}\right]. \quad (2.17)$$

A través de este método también es posible hallar $a(r)$ y $b(r)$:

$$b(r) = \frac{I_1(r) - I_3(r)}{2\sin[\phi(r)]\sin(\alpha)}, \quad (2.18)$$
$$a(r) = I_2(r) - b(r)\cos[\phi(r)]. \quad (2.19)$$

Para el caso particular en que $\alpha = \frac{\pi}{4}$, Ver Figuras 2.1, 2.2 y 2.3, obtenemos la fórmula.

$$\phi(r) = \arctan\left[(\sqrt{2}-1)\frac{I_1(r) - I_3(r)}{2I_2(r) - I_1(r) - I_3(r)}\right]. \quad (2.20)$$

Hay otros casos interesantes, que se pueden obtener tomando un offset y tamaño de

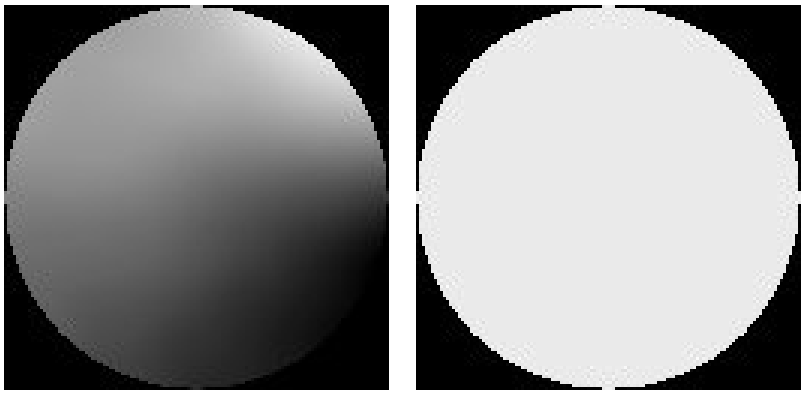


Figura 2.1. Fase Desevuelta y máscara

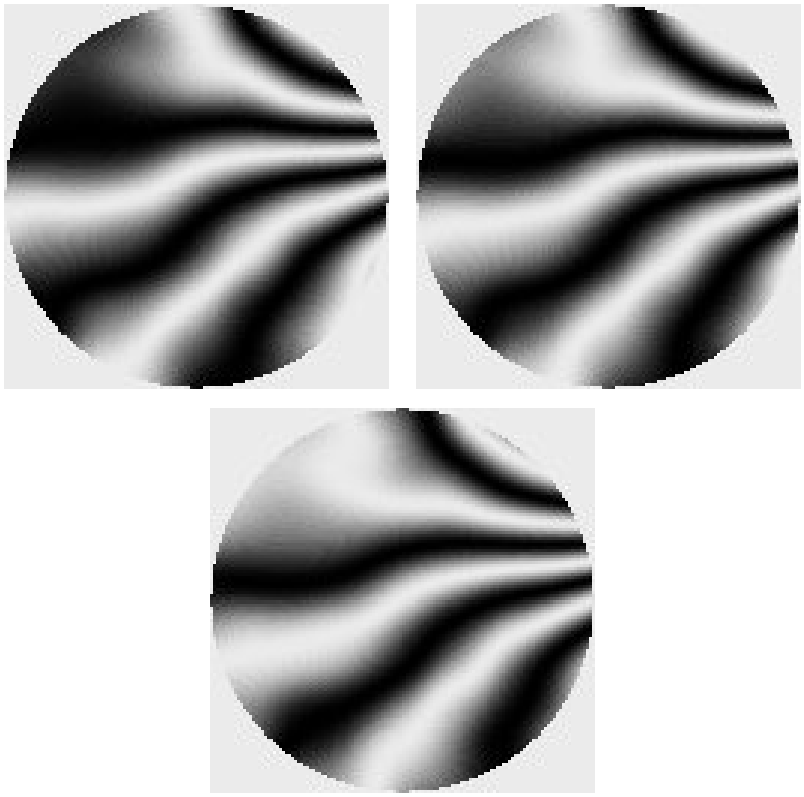


Figura 2.2. Interferograma de 3 pasos

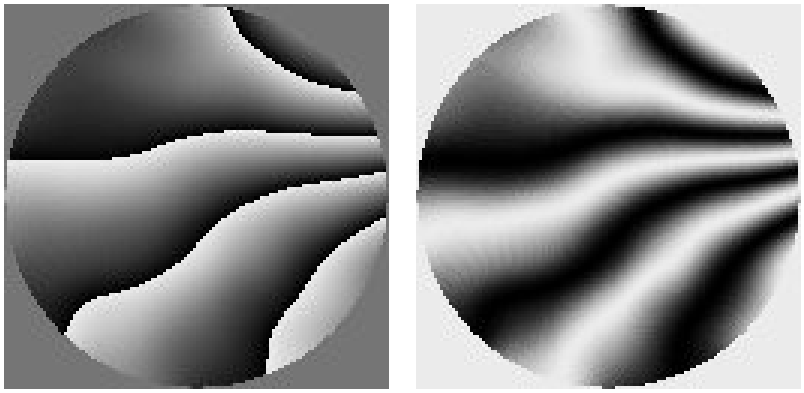


Figura 2.3. Fase envuelta y su coseno

paso conveniente. Por ejemplo, si $\beta = \frac{\pi}{4}$ y $\alpha = \frac{\pi}{2}$ (Wyant et al 1984), se tiene que:

$$\phi(r) = \arctan\left[\frac{I_3(r) - I_2(r)}{I_1(r) - I_2(r)}\right]. \tag{2.21}$$

Ahora, si en la ecuación 2.17 hacemos $\alpha = \frac{\pi}{2}$ obtenemos la ecuación.

$$\phi(r) = \arctan\left[\frac{I_1(r) - I_3(r)}{2I_2(r) - I_1(r) - I_3(r)}\right]. \tag{2.22}$$

Aunque el PSI de 3 pasos es la más simple de las técnicas para la obtención de la fase mediante interferometría de fase escalonada, uno de las dificultades que tiene es su sensibilidad al ruido, es por ello que se han realizado otros intentos que en general consisten en aumentar el número de patrones de franjas.

2.1.2. PSI de 4 pasos

Al igual que el caso anterior, hay varias ideas mediante las cuales se puede obtener un algoritmo PSI de 4 pasos. Ahora en lugar de tener 3 imágenes con patrones de franjas, se cuentan con 4 imágenes con patrones de franjas.

A continuación se verán 2 ejemplos

Algoritmo de 4 pasos

De forma similar a como se hizo en el caso anterior, se toman pasos de igual tamaño, es decir, $\delta_i = -\alpha, 0, \alpha, 2\alpha$ para $i = 1, 2, 3, 4$. En este caso la secuencia de pasos es conocida y tienen valores constantes. Evidentemente, si se tiene en cuenta que son solo 3 las incógnitas y 4 el número de ecuaciones, entonces se está en presencia de un problema sobredeterminado. Las ecuaciones para los modelos de 4 patrones de franjas son:

$$I_1(r) = a(r) + b(r)\cos[\phi(r) + \delta_1], \tag{2.23}$$
$$I_2(r) = a(r) + b(r)\cos[\phi(r) + \delta_2], \tag{2.24}$$
$$I_3(r) = a(r) + b(r)\cos[\phi(r) + \delta_3], \tag{2.25}$$
$$I_4(r) = a(r) + b(r)\cos[\phi(r) + \delta_3]. \tag{2.26}$$

Ahora considere el caso simple, donde $\alpha = \frac{\pi}{2}$. Para este valor de α, los cálculos se simplifican mucho, obteniéndose:

$$I_1(r) = a(r) + b(r)\sin[\phi(r)], \tag{2.27}$$
$$I_2(r) = a(r) + b(r)\cos[\phi(r)], \tag{2.28}$$
$$I_3(r) = a(r) - b(r)\sin[\phi(r)], \tag{2.29}$$
$$I_4(r) = a(r) - b(r)\cos[\phi(r)]. \tag{2.30}$$

Restando las ecuaciones 2.27 y 2.29 y las ecuaciones 2.28 y 2.30 obtenemos:

$$I_1(r) - I_3(r) = 2b(r)\sin[\phi(r)], \tag{2.31}$$
$$I_2(r) - I_4(r) = 2b(r)\cos[\phi(r)]. \tag{2.32}$$

De lo anterior se tiene que:

$$\phi(r) = \arctan\left[\frac{I_1(r) - I_3(r)}{I_2(r) - I_4(r)}\right]. \tag{2.33}$$

Que es una fórmula muy cómoda, no obstante, es fácil ver que si sumamos las ecuaciones 2.27 y 2.29, y las ecuaciones 2.28 y 2.30:

$$I_1(r) + I_3(r) = I_2(r) + I_4(r). \tag{2.34}$$

Sustituyendo $I_4(r)$ en la ecuación 2.33 obtenemos exactamente la ecuación 2.17 que corresponde al PSI de 3 pasos.

Algoritmo de Carré

Una expresión menos sensible al ruido fue propuesta por Carré en 1966. Este es un algoritmo de 4 pasos, pero a diferencia del caso anterior ahora no se necesita conocer el tamaño del paso. Basta que el tamaño del paso se mantenga constante entre los interferogramas.
Asumimos que $\delta_i = -3\alpha(r), -\alpha(r), \alpha(r), 3\alpha(r)$ para $i = 1, 2, 3, 4$:

$$I_1(r) = a(r) + b(r)\cos[\phi(r) - 3\alpha(r)], \tag{2.35}$$
$$I_2(r) = a(r) + b(r)\cos[\phi(r) - \alpha(r)], \tag{2.36}$$
$$I_3(r) = a(r) + b(r)\cos[\phi(r) + \alpha(r)], \tag{2.37}$$
$$I_4(r) = a(r) + b(r)\cos[\phi(r) + 3\alpha(r)]. \tag{2.38}$$

Otra diferencia del algoritmo de Carré con relación al anterior, es que ahora contamos con 4 variables en lugar de 3, por lo que el problema queda bien determinado. Observe que $\alpha(r)$ no es constante, y depende de cada posición, por lo que se podrían tener diferentes pasos de fase en cada punto.
La idea ahora es determinar primero $\alpha(r)$. Resolviendo el sistema anterior para $\alpha(r)$ se tiene:

$$\alpha(r) = \arctan\left\{\frac{3[I_2(r) - I_3(r)] - [I_1(r) - I_4(r)]}{[I_1(r) - I_4(r)] + [I_2(r) - I_3(r)]}\right\}^{\frac{1}{2}}. \tag{2.39}$$

Resolviendo ahora el sistema de ecuaciones para $\phi(r)$, se tiene que la solución para el frente de ondas (fase) viene expresado por:

$$\phi(r) = \arctan\left\{\tan[\alpha(r)]\frac{[I_1(r) - I_4(r)] + [I_2(r) - I_3(r)]}{[I_2(r) + I_3(r)] - [I_1(r) + I_4(r)]}\right\}. \tag{2.40}$$

o Sustituyendo $\alpha(r)$:

$$\phi(r) = \arctan\left\{\frac{\left\{\left[3[I_2(r)-I_3(r)]-[I_1(r)-I_4(r)]\right]\left[[I_1(r)-I_4(r)]+[I_2(r)-I_3(r)]\right]\right\}^{\frac{1}{2}}}{[I_2(r)+I_3(r)]-[I_1(r)+I_4(r)]}\right\}. \tag{2.41}$$

2.1.3. PSI de n pasos

Generalización PSI de n pasos

El desarrollo de esta teoría para n-pasos de fase se puede encontrar en [15] y [16], no obstante aquí se verán algunos detalles. En esta sección se usará 't' como subíndice para evitar confusión con la unidad imaginaria $i = \sqrt{-1}$.

Considérese la siguiente secuencia de interferogramas igualmente espaciados:

$$I_t(r) \quad ; \; t = -N, ..., 0, ..., N; \tag{2.42}$$

donde la fase viene dada mediante la siguiente expresión:

$$\phi_t(r) = \phi(r) + t\alpha. \tag{2.43}$$

Nótese que se está considerando una secuencia de imágenes en el 'tiempo' t. Entonces

$$I_t(r) = a(r) + b(r)\cos[\phi_t(r)] \tag{2.44}$$

$$= a(r) + b(r)\cos[\phi(r) + t\alpha] \tag{2.45}$$

$$= \sum_{j=-N}^{N} \{a(r) + b(r)\cos[\phi(r) + t\alpha]\}\delta(t-j). \tag{2.46}$$

Sea la familia de filtros de cuadratura, centrados en $\omega_t = t\alpha$.

$$\begin{aligned} h_t &= h_t^{re} + ih_t^{im}, && (2.47)\\ &= \sum_{j=-N}^{N} (h_t^{re} + ih_t^{im})\delta(t-j), && (2.48)\\ h_t^{re} &= \Re(h_t), && (2.49)\\ h_t^{im} &= \Im(h_t). && (2.50) \end{aligned}$$

Para el cálculo de los coeficientes h_t^{re} y h_t^{re} ver Servin et al [16]. Observe que los pasos de fase son simétricos, es decir, se cumple que $\omega_t = \omega_{-t}$. Considérese además las siguientes relaciones:

$$\begin{aligned} h_t^{re} &= h_{-t}^{re}, && (2.51)\\ h_t^{im} &= -h_{-t}^{im}. && (2.52) \end{aligned}$$

Sea también, la siguiente función definida a través de la ecuación 2.53

$$g_t(r) = I_t(r) * h_t \quad (2.53)$$

Aquí la convolución es respecto a t. Entonces la fase es posible estimarla, a partir de:

$$\phi_t(r) = \arctan\left[\frac{\Im[g_t(r)]}{\Re[g_t(r)]}\right] \quad (2.54)$$

Estamos interesados en hallar $\phi(r) = \phi_0(r)$.

Por otro lado.

$$\begin{aligned} g_t(r) &= I_t(r) * h_t && (2.55)\\ &= \sum_{j=-N}^{N} I_t(r)h_{t-j} && (2.56)\\ &= \sum_{j=-N}^{N} I_t(r)h_{t-j}^{re} + i\sum_{j=-N}^{N} I_t(r)h_{t-j}^{im} && (2.57) \end{aligned}$$

$$\Re[g_t(r)] = \sum_{j=-N}^{N} I_t(r) h_{t-j}^{re} \tag{2.58}$$

$$\Im[g_t(r)] = \sum_{j=-N}^{N} I_t(r) h_{t-j}^{im} \tag{2.59}$$

$$\phi(r) = \arctan\left[\frac{\Im[g_0(r)]}{\Re[g_0(r)]}\right] \tag{2.60}$$

$$= -\arctan\left[\frac{\sum_{j=-N}^{N} I_j(r) h_j^{im}}{\sum_{j=-N}^{N} I_j(r) h_j^{re}}\right] \tag{2.61}$$

Ejemplo: Para el PSI de 3 pasos por ejemplo se puede verificar que

$$h(t) = \sin(\alpha)[2\delta(t) - \delta(t-\alpha) - \delta(t+\alpha)] + i[1-\cos(\alpha)][\delta(t-\alpha) - \delta(t+\alpha)]$$

Pasos de fase mediante Mínimos Cuadrados

El método de los mínimos cuadrados para el análisis de patrones de franjas fue propuesto inicialmente por Brunning et al. (1974) [17], luego reformulado por Greivenkamp (1984) [18]. Su versión mas completa se puede encontrar en Greivenkamp y Brunning (1992) [14], [19].

El método de los mínimos cuadrados permite reconstruir el frente de ondas (fase) siempre y cuando sea conocida la secuencia de pasos, cada paso puede ser un valor arbitrario, pero conocido. En este caso lo primero que se hace es definir una secuencia $\{\delta_i\}$ de N pasos; donde $i = 1, 2, ..., N$, de modo que se tienen N imágenes con patrones de franjas.

$$I_i(r) = a(r) + b(r)\cos(\phi(r) + \delta_i) \tag{2.62}$$

Aplicando la fórmula del coseno de la suma

$$I_i(r) = a(r) + b(r)\cos(\phi(r))\cos(\delta_i) - b(r)\sin(\phi(r))\sin(\delta_i) \tag{2.63}$$

$$= a_0(r) + a_1(r)\cos(\delta_i) + a_2(r)\sin(\delta_i) \tag{2.64}$$

La idea es serparar el paso δ_i de la fase $\phi(r)$. En la última fórmula se definen $a_0(r)$, $a_1(r)$ y $a_2(r)$ mediante.

$$\begin{aligned} a_0(r) &= a(r) \\ a_1(r) &= b(r)\cos(\phi(r)) \\ a_2(r) &= -b(r)\sin(\phi(r)) \end{aligned}$$

Para estimar la fase, es suficiente determinar $a_1(r)$ y $a_2(r)$, pues la fase se puede calcular por la siguiente expresión.

$$\phi(r) = -\arctan\left[\frac{a_2(r)}{a_1(r)}\right].$$

Para poder emplear el método de los mínimos cuadrados es necesario definir el error. Denotemos el error cuadrático total como E, y definamóslo de la siguiente forma

$$E(\mathbf{a}(r)) = \|\mathbf{I}(r) - \mathbf{B}\mathbf{a}(r)\|^2. \tag{2.65}$$

Donde

$$\mathbf{I}(r) = [I_1(r), I_2(r), ..., I_N(r)]^T, \tag{2.66}$$

$$\mathbf{a}(r) = [a_0(r), a_1(r), a_2(r)]^T, \tag{2.67}$$

$$\mathbf{b}_j = [b_{1j}, b_{2j}, ..., b_{Nj}]^T;\ j = 1, 2, 3, \tag{2.68}$$

$$\mathbf{B} = \begin{bmatrix} b_0 & b_1 & b_2 \end{bmatrix}, \tag{2.69}$$

$$= \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ \vdots & \vdots & \vdots \\ b_{N1} & b_{N2} & b_{N3} \end{bmatrix}, \tag{2.70}$$

$$b_{i1} = 1, \tag{2.71}$$

$$b_{i2} = \cos(\delta_i), \tag{2.72}$$

$$b_{i3} = \sin(\delta_i). \tag{2.73}$$

Con el objetivo de hacer más clara la exposición, se eliminará la dependencia de

"r", por lo que se tiene.

$$\mathbf{E}(\mathbf{a}) = \|\mathbf{I} - \mathbf{B}\mathbf{a}\|^2. \tag{2.74}$$

Se tienen que determinar los valores de los coeficientes a_0, a_1 y a_2, o las componentes del vector $\mathbf{a}$, de forma que la cantidad $\mathbf{E}$ tome un valor mínimo.

Hallemos el gradiente de $\mathbf{E}$.

$$\nabla\mathbf{E}(\mathbf{a}) = 2[\nabla(\mathbf{I} - \mathbf{B}\mathbf{a})]^T(\mathbf{I} - \mathbf{B}\mathbf{a}), \tag{2.75}$$

$$= -2\mathbf{B}^T(\mathbf{I} - \mathbf{B}\mathbf{a}). \tag{2.76}$$

Igualando a cero el gradiente anterior:

$$-2\mathbf{B}^T(\mathbf{I} - \mathbf{B}\mathbf{a}) = 0, \tag{2.77}$$

$$\mathbf{B}^T\mathbf{B}\mathbf{a} = \mathbf{B}^T\mathbf{I}. \tag{2.78}$$

Lo anterior es un sistema de 3 ecuaciones con 3 variables. A continuación se hallarán la matriz del sistema $\mathbf{B}^T\mathbf{B}$ y el vector de términos independientes $\mathbf{B}^T\mathbf{I}$

$$\mathbf{B}^T\mathbf{B} = \begin{bmatrix} N & \sum\cos(\delta_i) & \sum\sin(\delta_i) \\ \sum\cos(\delta_i) & \sum\cos^2(\delta_i) & \sum\cos(\delta_i)\sin(\delta_i) \\ \sum\sin(\delta_i) & \sum\cos(\delta_i)\sin(\delta_i) & \sum\sin^2(\delta_i) \end{bmatrix}, \tag{2.79}$$

$$\mathbf{B}^T\mathbf{I} = \begin{bmatrix} \sum I_i \\ \sum I_i\cos(\delta_i) \\ \sum I_i\sin(\delta_i) \end{bmatrix}. \tag{2.80}$$

Sean $\mathbf{A}$ la matriz del sistema y $\mathbf{b}$ el vector de términos independiente, entonces el sistema puede ser resuelto fácilmente mediante:

$$\mathbf{A}\mathbf{a} = \mathbf{b}, \tag{2.81}$$

$$\mathbf{a} = \mathbf{A}^{-1}\mathbf{b}. \tag{2.82}$$

La forma común de expresar el sistema anterior es.

$$\begin{bmatrix} N & \sum\cos(\delta_i) & \sum\sin(\delta_i) \\ \sum\cos(\delta_i) & \sum\cos^2(\delta_i) & \sum\cos(\delta_i)\sin(\delta_i) \\ \sum\sin(\delta_i) & \sum\cos(\delta_i)\sin(\delta_i) & \sum\sin^2(\delta_i) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum I_i \\ \sum I_i\cos(\delta_i) \\ \sum I_i\sin(\delta_i) \end{bmatrix}.$$

Los algoritmos PSI se pueden usar tanto para franjas abiertas como franjas cerradas y su funcionamiento es muy rápido desde el punto de vista computacional.

Dentro de las dificultades principales están.

- La cantidad de patrones de franjas. Se requiere de al menos 3 imágenes con patrones de franjas
- Conocer con cierta precisión la secuencia de pasos, en caso contrario, estos algoritmos pueden causar problemas. Por lo que en muchos casos se requiere de una calibración de la secuencia de pasos; para ajustar los pasos de fase, ver Rivera [20].
- La sensibilidad ante ruido.

2.2. Método de Takeda

En método de Takeda es una de las técnicas de estimación de la fase en el dominio de las frecuencias. Fue propuesto por Takeda et al (1982) en 1D y luego se extendió a 2D por Macy (1983).

Considere el siguiente modelo, con frecuencia portadora ω_0.

$$I(r) = a(r) + b(r)\cos(\phi(r) + \omega_0 r).$$

Transformando la expresión anterior a su forma compleja

$$I(r) = a(r) + c(r)e^{i\omega_o r} + c^*(r)e^{-i\omega_o r},$$

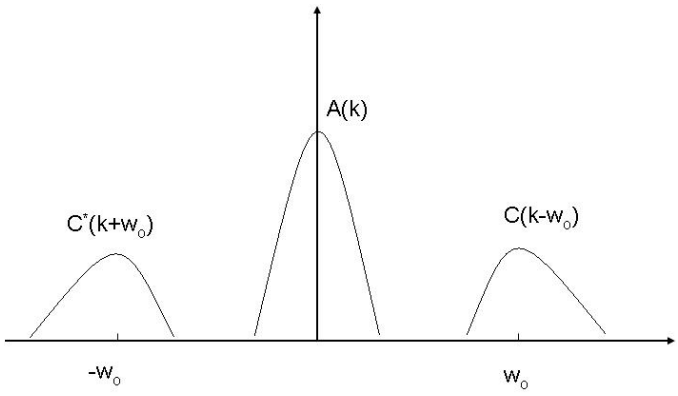


Figura 2.4. Espectro del interferograma (1D)

donde

$$c(r) = \frac{b(r)}{2} e^{i\phi(r)}.$$

Y $c^*(r)$ es la compleja conjugada de $c(r)$

Aplicando la transformada de Fourier, y las propiedades de linealidad y traslación:

$$\mathfrak{I}(k) = \mathfrak{A}(k) + \mathfrak{C}(k - \omega_o) + \mathfrak{C}^*(k + \omega_o), \tag{2.83}$$

Donde

$$\mathfrak{I} = \mathfrak{F}(I), \tag{2.84}$$
$$\mathfrak{A} = \mathfrak{F}(a), \tag{2.85}$$
$$\mathfrak{C} = \mathfrak{F}(c), \tag{2.86}$$
$$\mathfrak{C}^* = \mathfrak{F}(c^*). \tag{2.87}$$

En la Figura 2.4 se puede ver el espectro del interferograma.

Ahora se aplica un filtro de cuadratura $\mathfrak{Q}_{\omega_0}(k)$, centrado en la frecuencia ω_0 y queda

$$\mathfrak{Q}_{\omega_0}(k)\mathfrak{I}(k) = \mathfrak{Q}_{\omega_0}(k)\mathfrak{C}(k - \omega_o). \tag{2.88}$$

Aplicando una traslación y la transformada inversa:

$$\hat{\mathfrak{I}}(k) = \mathfrak{Q}_{\omega_0}(k)\mathfrak{C}(k), \tag{2.89}$$

$$\hat{I}(r) = q(r) * c(r), \tag{2.90}$$

$$= q(\omega_0)c(r). \tag{2.91}$$

Con lo anterior se puede calcular la fase, mediante

$$\phi(r) = \arctan \frac{\Im[\hat{I}(r)]}{\Re[\hat{I}(r)]}. \tag{2.92}$$

El Algoritmo de Takeda se puede resumir como sigue:

Algoritmo

1. Se aplica la *Transformada de Fourier* al interferograma.
2. Aplicar un Filtro de cuadratura de forma conveniente (en un semiplano, eliminar las bajas y las altas frecuencias).
3. Trasladar el resultado al origen para eliminar la frecuencia ω_o.
4. Aplicar la *Transformada Inversa de Fourier*.
5. Calcular la fase mediante $\phi(r) = \arctan \frac{\Im[c(r)]}{\Re[c(r)]}$

El método de Takeda es un método rápido desde el punto de vista computacional y también es muy útil pues se aplica a un solo patrón de franjas.

Comúnmente se usa la Trasformada Rápida de Fourier, lo cual hace más eficiente el algoritmo, pero incorpora una limitante, pues hay que trabajar con imágenes cuyas dimensiones sean potencias de 2.

Vemos algunos ejemplos:

1. En las Figuras 2.5, 2.6 y 2.7 se muestra un interferograma con franjas abiertas y las Fase Horizontal y Vertical.
   Observe que las franjas son verticales y al mismo tiempo el filtro vertical genera

una mejora en la salida de la fase.

2. En las Figuras 2.8, 2.9 y 2.10 se muestra un interferograma con franjas cerradas y las Fases Horizontal y Vertical.
   Observe que en este caso las imágenes de fase presentan algunos problemas en las franjas cerradas.

Como conclusión, el método de Takeda obtiene una fase buena cuando las franjas son abiertas y se filtran en la dirección de las franjas. Esto es cuando los modos del espectro, esquematizados en la Figura 2.4 son perfectamente separables. En el caso de franjas cerradas $\mathfrak{C}(k - \omega_o)$ se entiende desde las frecuencias positivas a las negativas y el filtrado de cuadratura no está definido. Por ello cuando hay franjas cerradas, o hay zonas con franjas cerradas, no funciona bien.

2.3. Regularización. *Regularized Phase Tracking*

Una de las ventajas fundamentales de las técnicas de regularización para la obtención de la fase es que permiten demodular un sólo patrón de franjas y se pueden usar tanto para franjas abiertas como para franjas cerradas.

En particular se hablará de la técnica RPT (Regularized Phase Tracking) que fue desarrollada por Servin et al (1997) [4].

Aquí se asume que localmente los patrones de franjas son monocromáticos, por lo que la fase se puede ser modulada localmente como un plano. Dicho plano es adaptado en cada región si tenemos en cuenta que la frecuencia local cambia continuamente en el patrón de franjas.

La función de costo o Energía consta de dos términos: Término de Datos y Término de regularización.

El primer término tiene que ver con la fidelidad de los datos observados con la

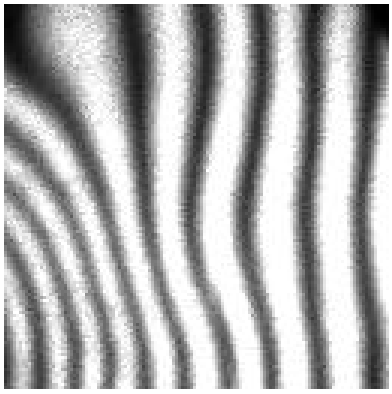


Figura 2.5. Interferograma con franjas abiertas

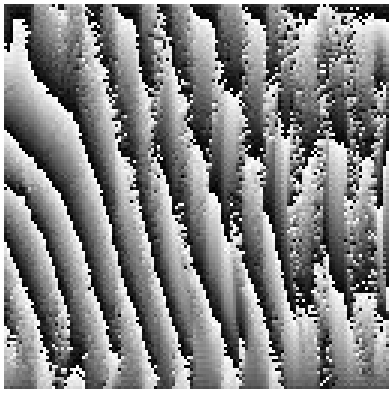


Figura 2.6. Fase, aplicando el filtro en el semiplano horizontal

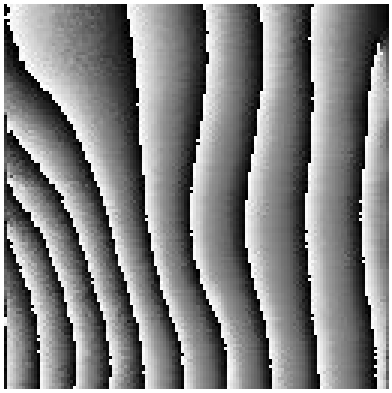


Figura 2.7. Fase, aplicando el filtro en el semiplano vertical

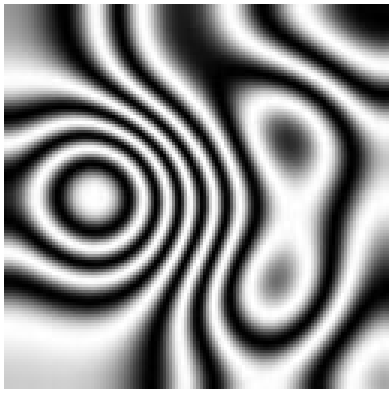


Figura 2.8. Interferograma con franjas cerradas

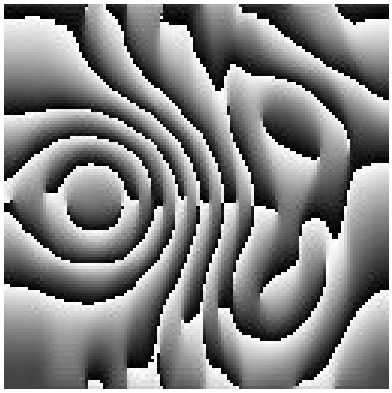


Figura 2.9. Fase, aplicando el filtro en el semiplano horizontal

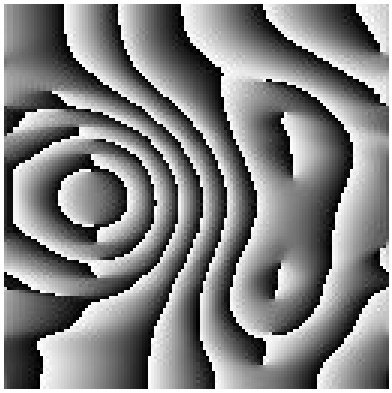


Figura 2.10. Fase, aplicando el filtro en el semiplano vertical

estimación realizada y el segundo con la suavidad del frente de ondas demodulado.

$$U_T(\phi) = \sum_{r\in\mathcal{L}} U_r(\phi_0,\omega), \tag{2.93}$$

donde

$$\begin{aligned} U_r(\phi_0,\omega) &= \sum_{s\in(\mathcal{N}_r \bigcap L)} \{[I^{'}(s)-\cos p(r,s)]^2 + \lambda[\phi_0(s)-p(r,s)]^2 m(s)\}, \quad (2.94)\\ p(r,s) &= \phi_0(r)+\omega(r)^T(r-s), \quad (2.95)\\ r &= (x,y),\\ s &= (\xi,\eta),\\ \omega(r) &= (\omega_x,\omega_y). \end{aligned}$$

Donde $\mathcal{L}$ es la retícula, $\mathcal{N}_r$ es la vecindad del pixel r, U_r es la función de energía que corresponde al pixel r, m(r) es una variable indicadora que es 1 si el sitio r ya ha sido estimado y 0 en cualquier otro caso, $\omega(r)$ representa un vector de la frecuencia local en ambas direcciones. $I^{'}$ es la imagen que se obtiene a partir del patrón de franjas original después de pasar un filtro pasa altas.

Dado que el funcional de costo puede tener varios mínimos locales, el algoritmo consiste en dos partes, en la primera se estima una fase inicial en toda la imagen y en la segunda se mejora la fase estimada en la primera parte.

Algoritmo RPT

Iniciar la función indicadora $m(r) = 0\ \forall r \in \mathcal{L}$.
Se selecciona una semilla o un punto inicial $r_0 \in \mathcal{L}$, se optimiza el funcional $U_{r_0}(\phi_0,\omega)$ y se marca r_0 como estimado, es decir, $m(r_0) = 1$.

Fase A, Estimar Fase Inicial

1. Seleccionar un pixel $r \in \mathcal{L}$ (aleatoriamente o en un orden determinado)
2. **(a)** Si m(r) = 1 ir al paso 1
   (b) Si m(r) = 0, verificar si exite $r' \in \mathcal{N}_r$ tal que m(r') = 1, es decir, que halla sido estimado.
      (b.1) Si no existe $r' \in \mathcal{N}_r$ que halla sido estimado ($m(r') \neq 1$) ir al paso 1

(b.2) Si existe algún $r' \in \mathcal{N}_r$ que halla sido estimado ($m(r') = 1$),
Tomar $[\phi_0(r'), \omega(r')]$ como condición inicial para minimizar $U(r)$ con respecto a $[\phi_0(r), \omega(r)]$

3. Hacer m(r) = 1

4. Ir al paso hasta que todos los pixeles hallan sido estimados

Fase B, Refinar Fase inicial

En esta parte se minimiza el potencial U_T, para lo que se propone usar un algoritmo ICM (Iterated Conditional Mode, propuesto por Besag), que fue diseñado para encontrar el *estimador máximo a posteriori* de una imagen en forma vectorial modelada como un campo aleatorio con distribución Gibbsiana.

Como el gradiente $\nabla_{\mathbf{b}} U(r)$ resulta de un sistema no lineal, para minimizar el funcional 2.94 se propone usar gradiente descendente, es decir, se desea determinar el vector $\mathbf{b}(r) = (\phi_0(r), \omega_x(r), \omega_y(r))^T$, o lo que es lo mismo, la fase y la frecuencia local en cada pixel r, de acuerdo a:

$$\mathbf{b}^{k+1}(r) \;=\; \mathbf{b}^{k}(r) - \mu \nabla_{\mathbf{b}} U(r), \tag{2.96}$$

Donde $\nabla_{\mathbf{b}} U(r)$ representa el gradiente del funcional $U(r)$, que viene expresado por:

$$\nabla_{\mathbf{b}} U(r) \;=\; \left[\frac{\partial U(r)}{\partial \phi_0(r)}, \frac{\partial U(r)}{\partial \omega_x(r)}, \frac{\partial U(r)}{\partial \omega_y(r)} \right]^T, \tag{2.97}$$

Como condición inicial se sugiere $\mathbf{b}^{(0)}(r_0) = (0, 0, 0)^T$.

Dentro de los problemas fundamentales de este algoritmo está su bajo rendimiento computacional.

Luego este algoritmo fue mejorado en [5] y así apareció el **FFRPT** (Fringe-Follower Regularised Phase Tracker). A la nueva propuesta se le incluye una estrategia de búsqueda para la estimación de la fase inicial, es decir, ahora se define una secuencia de pixeles en el interferograma a ser demodulado, a diferencia del RPT que se hacia de forma independiente. En este caso la estrategia está basada en la máxima irradianza

o el máximo nivel de intensidad. Una vez que se ha estimado la fase en un pixel, el próximo que se toma, es el pixel vecino con mayor irradianza. Para imágenes que tengan mucho ruido se sugiere realizar un preprocesamiento, aplicando un filtro pasa bajas, para obtener un patrón de franjas más suave. Por otro lado con el objetivo de guiar la búsqueda y por tanto simplificar el algoritmo, se propone el siguiente patrón en dos niveles.

$$I_b(r) \quad = \quad Umbral[I(r) * LPF(r)]. \tag{2.98}$$

Donde LPF es una matriz de convolución pasa bajas, $I(r)$ es la imagen del patrón de franjas y $Umbral(.)$ es un operador de binarización.
El nuevo algoritmo FFRPT incluye como primer paso, hallar el patrón de franjas binario I_b, el cual sigue como referencia, a partir de una semilla para estimar la fase inicial en toda la imagen y así completar la primera parte del algoritmo. Como segunda parte se toma de forma similar al RPT.

Por otro lado, también se le realiza una modificación al potencial 2.94 incluyendo un término mas, y adquiriendo la forma

$$\begin{aligned} U_r(\phi_0, \omega) \quad = \quad & \sum_{s \in (\mathcal{N}_r \bigcap L)} \{[I^{'}(s) - \cos p(r,s)]^2 + [I^{'}(s) - \cos(p(r,s) + \alpha)]^2 + \\ & \lambda[\phi_0(s) - p(r,s)]^2 m(s)\}. \end{aligned} \tag{2.99}$$

La constante de traslación α que aparece en el segundo término, es un valor entre 0.1π y 0.3π; frecuentemente se usa $\frac{\pi}{4}$.

En el 2003 apareció el **RQPT** (Regularized quadrature and Phase Tracking) [6], [7]; este algoritmo mejora las versiones anteriores del RPT haciéndolo más robusto.

Aquí se proponen dos potenciales diferentes, a partir de los cuales se obtienen además de la fase estimada, la imagen en cuadratura.

$$\begin{aligned} U_r(\phi_0, \omega) \quad = \quad & \sum_{s \in (\mathcal{N}_r \bigcap L)} \{[I(s) - \cos p(r,s)]^2 + [I_x(s) + \hat{\omega}_x \sin p(r,s)]^2 \\ & + [I_y(s) + \hat{\omega}_y \sin p(r,s)]^2\}, \end{aligned} \tag{2.100}$$

$$\begin{aligned} U_r(\phi_0,\omega) &= \sum_{s\in(\mathcal{N}_r \bigcap L)} \{[I^{'}(s)-\cos p(r,s)]^2+[I_x(s)+\hat{\omega}_x \sin p(r,s)]^2 \\ &\quad +[I_y(s)+\hat{\omega}_y \sin p(r,s)]^2+\lambda[\phi_0(s)-p(r,s)]^2 m(s)\}. \end{aligned} \tag{2.101}$$

Para este caso también se propone usar el método de gradiente descendente, y como condición inicial.

$$\hat{\mathbf{b}}(r_i)^0 = \hat{\mathbf{b}}(r_{i-1})^\infty. \tag{2.102}$$

donde $\hat{\mathbf{b}}(r) = (\hat{\phi_0}(r), \hat{\omega}_x(r), \hat{\omega}_y(r))^T$, es decir, como condición inicial $\hat{\mathbf{b}}(r_i)^0$ se toma el valor estable $\hat{\mathbf{b}}(r_{i-1})^\infty$ encontrado para r_{i-1}.

Como *mapa de calidad* para el seguimiento se toma simplemente.

$$\begin{aligned} si\ I(x) &> 0;\ buen\ dato \\ si\ I(x) &\leq 0;\ dato\ malo \end{aligned}$$

Basado en el artículo de Rivera [8] (2005), donde usa la idea del desarrollo en series de Taylor, para extraer la fase estimada del argumento del coseno, Legarda y Rivera [21] proponen un nuevo algoritmo **FHQRPT** (Fast Half-quadratic regularized phase tracking) basado en el **RPT**. En [8] se asume que la fase se puede expresar $\phi = \hat{\phi} + \tilde{\phi}$, donde $\hat{\phi}$ es una aproximación conocida de la fase y $\tilde{\phi}$ es un valor residual, por lo que, desarrollando la función *coseno* en series de Taylor

$$\cos(\hat{\phi}+\tilde{\phi}) \approx \cos(\hat{\phi}) - \tilde{\phi}\sin(\hat{\phi}). \tag{2.103}$$

Por otro lado

$$\hat{p}(r,s) = \hat{\phi}(r) + \hat{\omega}(r)^T(r-s), \tag{2.104}$$

$$\tilde{p}(r,s) = \tilde{\phi}(r) + \tilde{\omega}(r)^T(r-s). \tag{2.105}$$

Ahora la idea es determinar los términos residuales $\tilde{\phi}(r)$ y $\tilde{\omega}(r)$. Al potencial 2.99 también se le incorpora un término para la detección de *outliers* $l(s) \in [0,1]$ quedando

en la forma

$$\begin{aligned} U(\tilde{p}(r,s)) &= \frac{1}{2}\sum_{s\in\mathcal{N}_r} l(s)^2[I(s) - b(s)\cos\hat{p}(s) + b(s)\tilde{p}(s)\sin\hat{p}(s)]^2 + \\ & l(s)^2[I(s) - b(s)\cos(\hat{p}(s)+\alpha) + b(s)\tilde{p}(s)\sin(\hat{p}(s)+\alpha)]^2 + \\ & \mu(1-l(s))^2 + \lambda[\phi - \hat{p}(r,s) - \tilde{p}(r,s)]^2 m(s) \end{aligned} \tag{2.106}$$

donde, al igual que en [8], $b(s)$ es la amplitud estimada.

En esta técnica, primero se estima la *semilla* usando la ecuación 2.99, luego cada pixel es refinado a partir de la expresión 2.106, la ventaja de formulación 2.106 con relación a los anteriores **RPT** es que el proceso de minimización conduce a la solución de un sistema lineal de ecuaciones, que es posible resolver usando Gauss-Seidel y que el proceso de rechazo de datos atípicos hacen más robusto al algoritmo.

2.4. Refinamiento de la fase

En [8] Rivera presenta un algoritmo robusto para la demodulación de un solo interferograma con franjas abiertas y cerradas. Dicho algoritmo está compuesto por dos etapas: una de propagación y otra de refinamiento. Esta última no está restringida a este algoritmo, y como se verá más adelante, podrá ser usada para perfeccionar o mejorar la fase que se obtiene a partir de cualquier otro algoritmo, usando su salida como una estimación inicial de la fase.

El algoritmo asume que son conocidos de forma aproximada el contraste $\hat{b}$ y la fase ψ en la región de interés $R \subseteq \mathcal{L}$, donde $\mathcal{L}$ corresponde al dominio de toda la imagen y considera que el modelo de observaciones es el siguiente:

$$\hat{g}_r = \hat{b}_r \cos f_r, \quad r \in \mathcal{L}.$$

donde la verdadera fase se puede escribir como $f = \psi + \phi$, con $|\phi| \ll |\psi|$, es decir, se expresaría a través de una fase aproximada y un residual. Con lo anterior es posible usar la aproximación de primer order del desarrollo de Taylor,

$$\hat{g}_r \approx \hat{b}_r(\cos\psi_r - \phi_r \sin\psi_r).$$

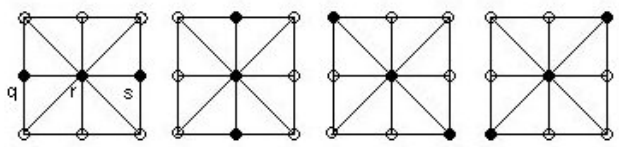


Figura 2.11. Cliques con las triadas $< q, r, s >$.

por lo que es posible definir el siguiente error:

$$E(\phi_r; \psi_r) \;=\; \hat{g}_r - \hat{b}_r(\cos\psi_r - \phi_r \sin\psi_r) \approx 0.$$

El potencial a minimizar sería:

$$\begin{aligned} U(\phi, \omega; \psi) \;=\; & \sum_{r \in R} [\omega_r^2 E(\phi_r, \psi_r) + \mu(1-\omega_r)^2] + \\ & \lambda \sum_{<q,r,s> \in R} [\psi_q + \phi_q - 2(\psi_r + \phi_r) + \psi_s + \phi_s]. \end{aligned}$$

donde las $\omega_r \in [0, 1]$ se interpretan como pesos y μ y λ son los parámetros que controlan la detección de datos atípicos y la suavidad de la solución.

En el caso del término de regularización usa cliques de tamaño 3 ,ver Figura 2.11; y penaliza los cambios en las segundas derivadas.

La minimización de U se hace en dos pasos hasta la convergencia: Dados ψ y $\hat{b}$ aproximadamente, se inicializan a 0 las ϕ_r.

- Se minimiza Urespecto a ω_r dejando ϕ_r constante y luego respecto a ϕ_r dejando ω_r constante.

- Se actualiza ψ.

Por las características de U, es posible encontrar una fórmula cerrada para ω_r, para lo cual se calcula la derivada parcial respecto ω_r:

$$\frac{\partial U}{\partial \omega_r} \;=\; 2\omega_r E^2(\phi_r; \psi_r) - 2\mu(1-\omega_r). \tag{2.107}$$

Igualando a cero y despejando ω_r, se obtiene:

$$\omega_r = \frac{\mu}{\mu + E^2(\phi_r; \psi_r)}. \tag{2.108}$$

Ahora para minimizar con respecto a ω_r, simplemente se calcula por la fórmula anterior.

Algoritmo de Refinamiento: Para refinar una fase inicial ψ en una región $R \subseteq \mathcal{L}$.
Dado $\epsilon > 0$;
mientras $\|\hat{g} - \hat{b}\cos\psi\| > \epsilon$ **hacer**
$\phi_r \leftarrow 0$;
calcular ω_r usando la fórmula 2.108; es lo mismo que: $\omega_r \leftarrow \arg\min_{\omega_r} U(\phi, \omega; \psi)$;
$\phi_r \leftarrow \arg\min_{\phi_r} U(\phi, \omega; \psi)$;
$\psi_r \leftarrow \psi_r + \phi_r$;
fin mientras

La Figura 2.12 muestra la entrada al algoritmo de refinamiento y en la Figura 2.13 se puede ver que los resultados son muy buenos. En el panel d) se puede ver que las franjas en el centro forman una especie de cuadrados, y este efecto es eliminado después del refinamiento. Este algoritmo ha sido generalizado para trabajar con varios patrones de franjas del tipo de corrimiento de fase Rivera et al [20].

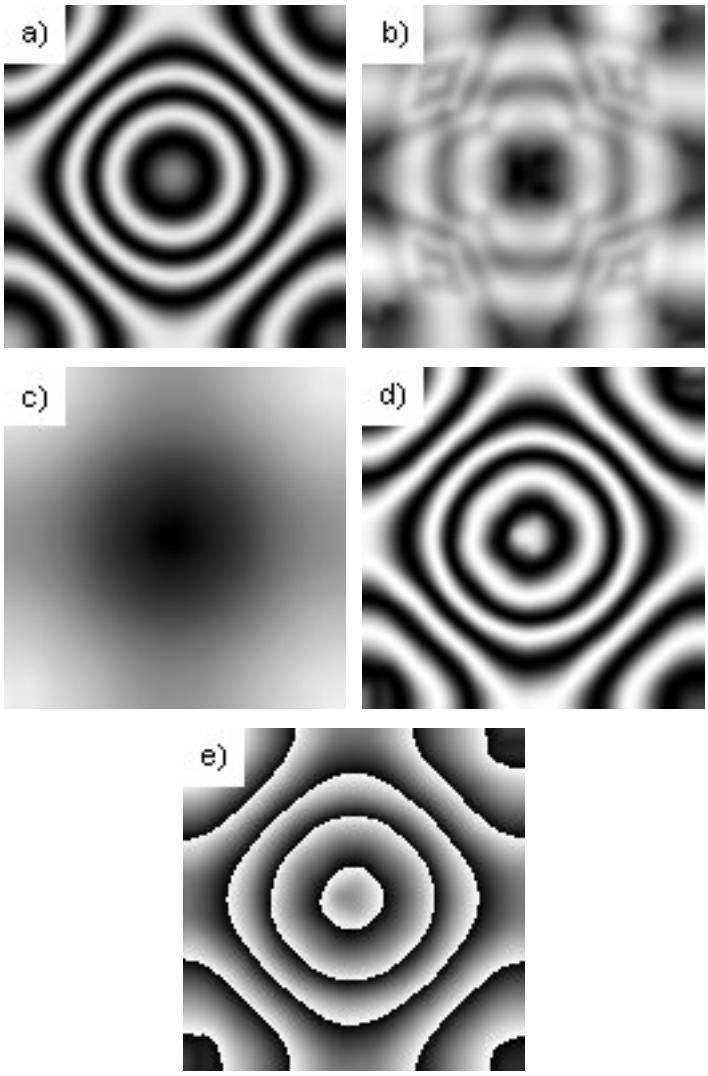


Figura 2.12. Entrada al algoritmo de refinamiento a) Imagen original b) Magnitud aproximada, c) Fase aproximada, d) Coseno de la fase, e) Fase envuelta.

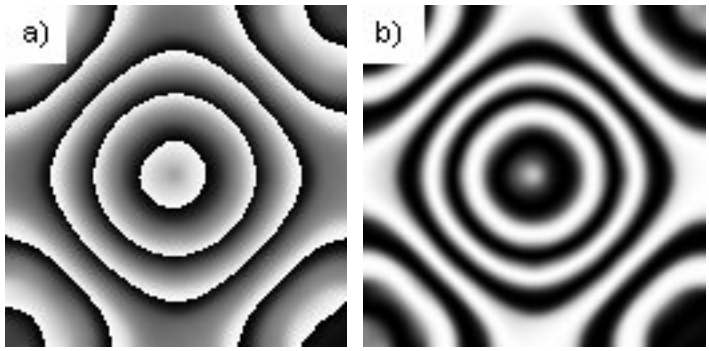


Figura 2.13. Salida del algoritmo de refinamiento a) Fase. b) Coseno de la fase.

Capítulo 3

Desenvolvimiento de Fase

En el capítulo anterior se vieron algunas técnicas para la obtención de la fase envuelta. En este capítulo se darán algunas nociones para desenvolver la fase.

3.1. Desenvolvimiento en 1D (Método de Itoh)

El Método de Itoh apareció en 1982 y la idea consiste en hallar la fase desenvuelta como un proceso de integración. Sea $\phi(n)$ la fase desenvuelta con $n = 0, 1, ..., N-1$ con la condición

$$-\pi < \quad \Delta\{\phi(n)\} \quad \leq \pi. \tag{3.1}$$

Donde Δ es el operador de diferencias

$$\Delta\phi(n) \quad = \quad \phi(n+1) - \phi(n), \quad n = 0, 1, ..., N-2. \tag{3.2}$$

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

$$\psi(n) = \mathcal{W}\{\phi(n)\} \quad = \quad \phi(n) + 2\pi k_1(n), \quad n = 0, 1, ..., N-1, \tag{3.3}$$

donde $k_1(n) \in \mathbb{Z}$.

Aquí $\psi(n)$ representa la *fase envuelta*, mientras que $\phi(n)$ es la *fase desenvuelta*. En

realidad el interés es recuperar de forma aproximada la fase desenvuelta, a partir de la fase envuelta. Por otro lado, otra forma de definir el operador $\mathcal{W}$, es mediante funciones trigonométricas, de la forma siguiente:

$$\mathcal{W}\{\phi(n)\} = \arctan\left[\frac{\sin\phi(n)}{\cos\phi(n)}\right]. \tag{3.4}$$

Aplicando el operador diferencia a la ecuación 3.3:

$$\Delta\{\mathcal{W}\{\phi(n)\}\} = \Delta\{\phi(n)\} + 2\pi\Delta\{k_1(n)\}; \tag{3.5}$$

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

$$\begin{aligned}\mathcal{W}\{\Delta\{\mathcal{W}\{\phi(n)\}\}\} &= \Delta\{\phi(n)\} + 2\pi\Delta\{k_1(n)\} + 2\pi k_2(n), &(3.6)\\ &= \Delta\{\phi(n)\} + 2\pi[\Delta\{k_1(n)\} + k_2(n)]. &(3.7)\end{aligned}$$

Como

$$-\pi < \mathcal{W}\{\Delta\{\mathcal{W}\{\phi(n)\}\}\} \leq \pi \tag{3.8}$$

y además

$$-\pi < \Delta\{\phi(n)\} \leq \pi. \tag{3.9}$$

Entonces necesariamente

$$2\pi[\Delta\{k_1(n)\} + k_2(n)] = 0. \tag{3.10}$$

Por lo que se tiene que:

$$\begin{aligned}\mathcal{W}\{\Delta\{\mathcal{W}\{\phi(n)\}\}\} &= \Delta\{\phi(n)\}, &(3.11)\\ \mathcal{W}\{\Delta\{\psi(n)\}\} &= \Delta\{\phi(n)\}. &(3.12)\end{aligned}$$

Es decir que podemos recuperar $\Delta\{\phi(n)\}$ a partir de la fase envuelta.

Pero podemos expresar $\phi(k)$ como una serie telescópica, de la forma siguiente.

$$\begin{aligned}
\phi(k) &= \phi(0) + [\phi(1) - \phi(0)] + [\phi(1) - \phi(2)] + ... + [\phi(k) - \phi(k-1)] && (3.13)\\
&= \phi(0) + \sum_{i=0}^{k-1} [\phi(i+1) - \phi(k)] && (3.14)\\
&= \phi(0) + \sum_{i=0}^{k-1} \Delta\{\phi(i)\} && (3.15)\\
&= \phi(0) + \sum_{i=0}^{k-1} \mathcal{W}\{\Delta\{\psi(i)\} && (3.16)\\
&= \phi(k-1) + \mathcal{W}\{\Delta\{\psi(k-1)\} && (3.17)
\end{aligned}$$

Con lo anterior ya se está en condiciones de presentar el ***Algoritmo de Itoh*** para estimar la fase ϕ.

Algoritmo de Itoh

1. Calcular las diferencias $\Delta\{\psi(k)\}$ para $k = 0, 1, ..., N-2$.

2. Envolver las diferencias anteriores, aplicando el operador $\mathcal{W}$, es decir, hallar $\mathcal{W}\{\Delta\{\psi(k)\}\} = \arctan\left[\frac{\sin\psi(k)}{\cos\psi(k)}\right]$.

3. Inicializar $\phi(0) = \psi(0)$.

4. Hallar la fase desenvuelta $\phi(k)$ para $k = 0, 1, ..., N-1$ mediante la ecuación 3.16 o mejor aún 3.17.

En las Figuras 3.1 y 3.2 se tienen la fase envuelta y la fase desenvuelta correspondiente después de aplicar el algoritmo de Itoh, en este caso la fase envuelta no tiene ruido, por lo que el resultado del desenvolvimiento es muy bueno. Por otro lado en las Figuras 3.3 y 3.4 se puede observar que la fase envuelta tiene cierto ruido (esta señal se obtuvo a partir de la Figura 3.1 añadiendo ruido blanco), por lo que el resultado del desenvolvimiento aplicando Itoh no es bueno, y se nota incluso una reducción del rango dinámico, con relación a la Figura 3.1.

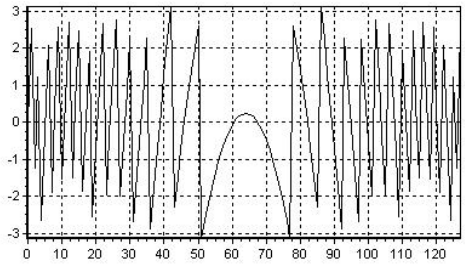


Figura 3.1. Fase envuelta sin ruido

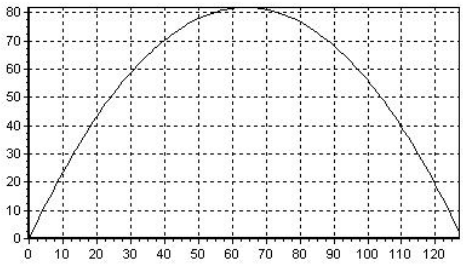


Figura 3.2. Fase desenvuelta sin ruido aplicando el algoritmo de Itoh

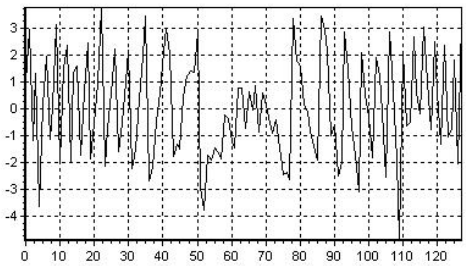


Figura 3.3. Fase envuelta con ruido

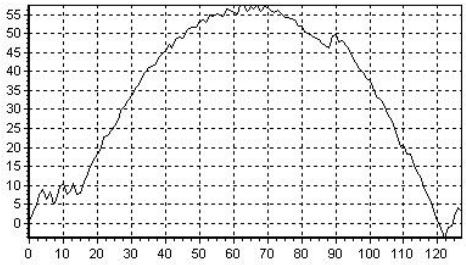


Figura 3.4. Fase envuelta con ruido aplicando el algoritmo de Itoh

3.2. Formulación del desenvolvimiento de fase en 2D

Algunas de las ideas para el desenvolvimiento de la fase en 1D son posibles extenderlas a 2D, otras no. En particular, el proceso de integración para la obtención de la fase, que es la base del Método de Itoh, no es posible extenderlo a 2D, y la dificultad fundamental está dada en que el proceso de integración en 2D no es independiente del camino de integración [2]. Por otro lado, el desenvolvimiento de fase, es en general, un problema mal condicionado. No obstante, se pueden establecer algunas restricciones bajo las cuales sea posible encontrar una solución. 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 la fase sean menores que π en magnitud.

De esta forma, el problema de desenvolvimiento de fase en 2D se puede plantear de la forma siguiente: Conocida la fase envuelta ψ en una retícula $\mathcal{L}$, hallar aproximadamente la fase desenvuelta ϕ en toda la retícula, bajo la restricción de que las derivadas de la fase estén en el intervalo $(-\pi, \pi]$, es decir, si

$$\psi_{i,j} = \phi_{i,j} + 2k\pi, \quad k \in \mathbb{Z}, \ (i,j) \in \mathcal{L}, \tag{3.18}$$

donde $-\pi < \psi_{i,j} \leq \pi$, $\forall (i,j) \in \mathcal{L}$, con la condición:

$$-\pi < \Delta_x\{\phi_{i,j}\} \leq \pi, \tag{3.19}$$

$$-\pi < \Delta_y\{\phi_{i,j}\} \leq \pi. \tag{3.20}$$

Para

$$\Delta_x\{\phi_{i,j}\} = \phi_{i+1,j} - \phi_{i,j}, \tag{3.21}$$

$$\Delta_y\{\phi_{i,j}\} = \phi_{i,j+1} - \phi_{i,j}. \tag{3.22}$$

De forma similar a como se hizo en el caso 1D y bajo el supuesto 3.19 y 3.20, es posible estimar $\Delta_x\{\phi_{i,j}\}$ y $\Delta_y\{\phi_{i,j}\}$ mediante.

$$\Delta_x\{\phi_{i,j}\} = \begin{cases} \mathcal{W}\{\Delta_x\{\psi_{i,j}\}\}, & \text{i = 0, 1,..., M-2; j = 0,1,...,N-1;} \\ 0, & \text{otro caso.} \end{cases} \tag{3.23}$$

$$\Delta_y\{\phi_{i,j}\} = \begin{cases} \mathcal{W}\{\Delta_y\{\psi_{i,j}\}\}, & \text{i = 0, 1,..., M-1; j = 0,1,...,N-2;} \\ 0, & \text{otro caso.} \end{cases} \tag{3.24}$$

Donde la retícula $\mathcal{L}$ tiene dimensiones $M \times N$. El resultado anterior, que se obtuvo como una generalización del caso unidimensional, tiene mucha importancia y es muy usado en diferentes algoritmos de desenvolvimiento.

3.3. Algoritmo de desenvolvimiento de fase

Bajo las consideraciones realizadas anteriormente, se han propuesto muchos algoritmos para el desenvolvimiento de la fase, así podemos encontrar tanto técnicas de *seguimiento de camino* como de *regularización*.

Seguimiento de Camino

Algunos de los algoritmos son:

- Corte de rama (Branch Cut).
- Seguimiento de camino guiado por mapas de calidad.
- Acercamiento de discontinuidad mínima de Flynn.

Corte de rama: Estos algoritmos tratan de seguir la idea del método de Itoh en 1D, es decir, la obtención de la fase desenvuelta a través de un proceso de integración.

$$\phi(r) = \int_C \nabla\phi.d\mathbf{r} + \phi(r_0) \tag{3.25}$$

Donde C es cualquier camino que une r con r_0.

Como se puede ver en [2], en el caso bidimensional, el problema de desenvolvimiento

puede ser dependiente del camino de integración, es decir, para algunos puntos se puede tener que.

$$\nabla \times \nabla\phi \neq 0. \tag{3.26}$$

o lo que es lo mismo

$$\frac{\partial^2\phi}{\partial x \partial y} \neq \frac{\partial^2\phi}{\partial y \partial x}. \tag{3.27}$$

Los puntos donde se cumple lo anterior, producen inconsistencias en el proceso de integración, inicialmente fueron detectados por Ghiglia (1987), más tarde Goldstein (1988) denominó a estos puntos *residuos*, por la relación que tienen los residuos de las funciones de variable compleja con los residuos asociados a la fase.

Para detectar los residuos, se calcula

$$q = \sum_{i=1}^{4} \Delta_i, \tag{3.28}$$

donde

$$\Delta_1 = \mathcal{W}\{\psi_{i,j+1} - \psi_{i,j}\}, \tag{3.29}$$
$$\Delta_2 = \mathcal{W}\{\psi_{i+1,j+1} - \psi_{i,j+1}\}, \tag{3.30}$$
$$\Delta_3 = \mathcal{W}\{\psi_{i+1,j} - \psi_{i+1,j+1}\}, \tag{3.31}$$
$$\Delta_4 = \mathcal{W}\{\psi_{i,j} - \psi_{i+1,j}\}. \tag{3.32}$$

Observe que lo que se está calculando en realidad es

$$q = \Delta_{yx}\{\phi_{i,j}\} - \Delta_{yx}\{\phi_{i,j}\} \tag{3.33}$$
$$= \frac{\partial^2\phi}{\partial y \partial x} - \frac{\partial^2\phi}{\partial x \partial y} \tag{3.34}$$
$$= \phi_{yx} - \phi_{yx}. \tag{3.35}$$

Es decir, q es la diferencia de las derivadas cruzadas. Luego si $q \neq 0$ entonces el pixel (i, j) es un residuo.

Como se puede verificar en [2], $q \in \{-2\pi, 2\pi\}$. Al signo de q se le da el nombre de *polaridad o carga del residuo*. Entonces se habla de residuos con polaridad positiva o negativa, según sea el signo de q.

Los algoritmos de *corte de rama* se basan en la detección de los residuos en la imagen. Si no se detentan residuos, entonces es posible integrar en toda la imagen, pues la integración sería independiente del camino. Si hay residuos, se crean los *cortes de ramas* que unen residuos con diferentes polaridad. Una vez que se han balanceado todos los residuos, y se han creado los *cortes de ramas*, se puede integrar eligiendo cualquier camino que no cruce por las ramas.

La diferencia entre los algoritmos entonces está, en como elegir de forma conveniente los cortes de ramas.

Regularización

- *Mínima norma L^p*: Este método es una generalización de mínimos cuadrados, en el sentido de la norma L^p.

$$\begin{aligned} \mathbf{E}^p(\phi) &= \sum_{r\in\mathcal{L}}\sum_{s\in\mathcal{N}_r} |\phi_r - \phi_s - \mathcal{W}(\psi_r - \psi_s)|^p \qquad (3.36) \\ &= \sum_{i=0}^{M-2}\sum_{j=0}^{N-1} |\phi_{i+1,j} - \phi_{i,j} - \mathcal{W}(\psi_{i+1,j} - \psi_{i,j})|^p + \\ &\quad \sum_{i=0}^{M-1}\sum_{j=0}^{N-2} |\phi_{i,j+1} - \phi_{i,j} - \mathcal{W}(\psi_{i,j+1} - \psi_{i,j})|^p \qquad (3.37) \end{aligned}$$

  Derivando e igualando a cero, conduce a resolver el sistema de ecuaciones

$$\begin{aligned} &[\Delta_x\{\phi_{q,r}\} - \mathcal{W}(\Delta_x\{\psi_{q,r}\})]U_{q,r} \\ +[\Delta_y\{\phi_{q,r}\} - \mathcal{W}(\Delta_y\{\psi_{q,r}\})]V_{q,r} &- [\Delta_x\{\phi_{q-1,r}\} - \mathcal{W}(\Delta_x\{\psi_{q-1,r}\})]U_{q-1,r} \qquad (3.38) \\ &-[\Delta_y\{\phi_{q,r-1}\} - \mathcal{W}(\Delta_y\{\psi_{q,r-1}\})]V_{q,r-1} = 0 \end{aligned}$$

donde

$$\mathbf{U}_{q,r} = |\phi_{q+1,r} - \phi_{q,r} - \mathcal{W}(\psi_{q+1,r} - \psi_{q,r})|^{p-2} \quad (3.39)$$

$$\mathbf{V}_{q,r} = |\phi_{q,r+1} - \phi_{q,r} - \mathcal{W}(\psi_{q,r+1} - \psi_{q,r})|^{p-2} \quad (3.40)$$

A las funciones U y V se interpretan como pesos

- *Mínimos cuadrados.* Es un caso particular de la norma L^p, para $p = 2$. Aquí se tiene el error cuadrático y como se puede observar, para este caso, los pesos U y V son ambos iguales a 1, Fried [22] y Hudgin [23].

La ecuación a que conduce es a la siguiente.

$$\begin{aligned}[\Delta_x\{\phi_{q,r}\} - \mathcal{W}(\Delta_x\{\psi_{q,r}\})] + [\Delta_y\{\phi_{q,r}\} - \mathcal{W}(\Delta_y\{\psi_{q,r}\})] \\ -[\Delta_x\{\phi_{q-1,r}\} - \mathcal{W}(\Delta_x\{\psi_{q-1,r}\})] - [\Delta_y\{\phi_{q,r-1}\} - \mathcal{W}(\Delta_y\{\psi_{q,r-1}\})] \\ = 0 \quad (3.41)\end{aligned}$$

de otra forma

$$[\Delta_x\{\phi_{q,r}\} - \Delta_x\{\phi_{q-1,r}\}] + [\Delta_y\{\phi_{q,r}\} - \Delta_y\{\phi_{q,r-1}\}] = \rho \quad (3.42)$$

$$\phi_{xx} + \phi_{yy} = \rho \quad (3.43)$$

donde

$$\rho = \mathcal{W}(\Delta_x\{\psi_{q,r}\}) + \mathcal{W}(\Delta_x\{\psi_{q-1,r}\}) + \mathcal{W}(\Delta_y\{\psi_{q,r}\}) + \mathcal{W}(\Delta_y\{\psi_{q,r-1}\})$$

La ecuación 3.43 es la conocida ecuación de Poisson para desenvolvimiento de franjas, y que se obtiene a partir de minimizar el error cuadrático.

- *Función de costo Half-quadratic.*

Primero redefinamos el operador Δ, aunque, básicamente este operador seguirá siendo un operador de diferencias, pero ahora se definirá en 2D.

$$\Delta h_{rs} \stackrel{def}{=} h_r - h_s \quad (3.44)$$

Este operador expresa la diferencia de una función h en dos pixeles vecinos

(horizontal o vertical), entonces las ecuaciones 3.23 y 3.24 se pueden escribir como

$$\Delta\hat{\phi}_{rs} = \mathcal{W}\{\Delta\psi_{rs}\} \tag{3.45}$$

O mejor, en el marco de la estimación Bayesiana, supongamos que.

$$\Delta\phi_{rs} = \Delta\hat{\phi}_{rs} + \Delta\eta_{rs} \tag{3.46}$$

Donde η_{rs} es cierto ruido aditivo. Dado que el problema de desenvolvimiento es mal condicionado, hay que imponer algunas restricciones. Entonces la fase $\hat{\phi}$ se obtiene de minimizar el funcional.

$$U(\phi) = D(\phi) + \lambda R(\phi) \tag{3.47}$$

donde el *término de datos*

$$D(\phi) = \sum_{<r,s>} [\Delta\phi_{rs} - \mathcal{W}\{\Delta\psi_{rs}\}]^2 \tag{3.48}$$

dice que $\hat{\phi}$ debe ser consistente con las observaciones ψ. El *termino de regularización*

$$R(\phi) = \sum_{<r,s>} [\Delta\phi_{rs}]^2 \tag{3.49}$$

está relacionado con las suposiciones sobre la variación espacial de la fase y tiene que ver con el conocimiento a priori de la fase a estimar, e impone una penalización por el incumplimiento de tales suposiciones. Aquí se asumen cambios suaves en la solución.

El *parámetro* λ controla la importancia relativa de ambos términos, Marroquín y Rivera [24].

$$U(\phi) = \sum_{<r,s>} \left\{[\Delta\phi_{rs} - \mathcal{W}\{\Delta\psi_{rs}\}]^2 + \lambda[\Delta\phi_{rs}]^2\right\} \tag{3.50}$$

El funcional es el resultado de la suposición de ruido gaussiano [9] y por otro

lado, el ruido puede generar inconsistencia (outliers), que es necesario tratar. Por lo que Rivera y Marroquín (2004) proponen el siguiente funcional, desde el punto de vista del *proceso de outliers*.

$$U_{hq}(\phi, l) = \sum_{<r,s>} \left\{ l_{rs}^2(\alpha_{rs}^2 + \lambda\beta_{rs}^2) + \Psi(l_{rs}) \right\} \tag{3.51}$$

donde $\alpha_{rs} \stackrel{def}{=} \Delta\phi_{rs} - \mathcal{W}\{\Delta\psi_{rs}\}$, $\beta_{rs} \stackrel{def}{=} \Delta\phi_{rs}$, $l_{rs}^2 \in [0, 1]$ que actúa como detector de outlier y Ψ es un potencial convexo.

El funcional 3.51 se puede expresar en términos de un potencial robusto [25] mediante

$$U_{hq}(\phi) = \sum_{<r,s>} \rho_{hq}(\alpha_{rs}) \tag{3.52}$$

para $\lambda = 0$ se tiene que.

$$l_{rs}^2 = \frac{\rho_{hq}^{'}(\alpha_{rs})}{2\alpha_{rs}} \tag{3.53}$$

donde l_{rs}^2 se interpretan como pesos. La expresión 3.53 da una fórmula cerrada para los pesos, lo cual facilita el proceso de minimización. Si $l_{rs}^2 \approx 0$, entonces existe una discontinuidad entre los pixeles r y s, y ocurre lo contrario cuando $l_{rs}^2 \approx 1$. La función Ψ se puede pensar como una *penalización* por la aparición de una discontinuidad entre los pixeles r y s. Esta función generalmente tiende a 1 cuando l_{rs}^2 tiende a 0, y visceversa, $\Psi(l_{rs}) \to 0$ cuando $l_{rs}^2 \to 1$, es decir, cuando no hay discontinuidad.

No obstante, lo anterior no es una regla, Black y Rangarajan en [25] muestran un catálogo de estimadores robustos y las funciones correspondientes del proceso de outlier, alguno de los cuales no cumplen con el supuesto anterior.

Si $l_{rs}^2 \to 1$ entonces no hay discontinuidad y por tanto $\Psi(l_{rs}) \to 0$, por lo que el funcional se reduce a la forma original de *mínimos cuadrados*, en caso contrario, el término que predomina es $\Psi(l_{rs})$.

El término Half-Quadratic proviene del hecho de que el funcional 3.51 es cuadrático si se fija l y es convexo para ϕ fijo.

Si el potencial robusto ρ es convexo, entonces se propone un algoritmo en dos pasos AM (Alternated Minimization).

Algoritmo

- Iniciar $l_{rs} = 1$
- Repetir
  - Determinar ϕ como el minimizador de $U_{hq}(\phi, l)$, con l fijo.
  - Determinar l como el minimizador de $U_{hq}(\phi, l)$, con ϕ fijo.
- hasta convergencia

El algoritmo anterior garantiza obtener el único mínimo que tiene el funcional.

En caso de ρ sea no convexo, se llega en general, a un mínimo local por lo que se propone seguir el método de *continuación de parámetros* GNC (Graduated Non-Convexity), propuesto por Blake and Zisserman (1987), en el cual los parámetros se ajustan para construir una aproximación convexa de la función objetivo original. La estrategia propuesta por Rivera y Marroquín en [9] fue la siguiente, $\lambda = \lambda(t) = \lambda_0 \times 0{,}5^{c_1 t}$ y $\mu = \mu(t) = \mu_0 \times 0{,}5^{c_2 t}$, donde c_1 y c_2 son constantes reales positivas, en particular ellos experimentaron con $c_1 = c_2 = \frac{1}{20}$, $\lambda_0 = 10$ y $\mu = 1$ y se obtuvieron muy buenos resultados.

Una de las funciones de penalización más populares, es $\Psi(z) = \mu(1 - z)^2$, la cual se corresponde con el potencial $\rho(x) = \frac{\mu x^2}{\mu + x^2}$ propuesto por Geman and McClure (1987). Entonces

$$z = \frac{\rho'(x)}{2x} \tag{3.54}$$

$$= \frac{\mu^2}{(\mu + x^2)^2} \tag{3.55}$$

donde $z = l_{rs}^2$ y para $\lambda \neq 0$, $x = \sqrt{\alpha_{rs}^2 + \lambda\beta_{rs}^2}$, por lo que.

$$l_{rs}^2 = \frac{\mu^2}{(\mu + \alpha_{rs}^2 + \lambda\beta_{rs}^2)^2} \tag{3.56}$$

Para este caso, los pasos del algoritmo serían.

Paso 1 Resolver el sistema de ecuaciones lineales, para ϕ_r

$$\sum_{s \in \mathcal{N}_r} [\alpha_{rs} + \lambda\beta_{rs}] l_{rs}^2 = 0$$

Paso 2 Calcular los pesos, mediante la fórmula

$$l_{rs}^2 = \frac{\mu^2}{(\mu + \alpha_{rs}^2 + \lambda\beta_{rs}^2)^2}$$

Paso 3 Ajustar λ y μ usando: $\lambda = \lambda(t) = \lambda_0 \times 0{,}5^{c_1 t}$ y $\mu = \mu(t) = \mu_0 \times 0{,}5^{c_2 t}$, donde c_1 y c_2 son constantes reales positivas

Como el potencial anterior es no-convexo hay que aplicar el método de continuación de parámetros, por lo que se sugiere comenzar con valores grandes para μ e irlo ajustando, haciéndolo decrecer.

En las parejas de Figuras 3.5 y 3.6, 3.7 y 3.8; y en 3.9 y 3.10 que representan la fase envuelta y su correspondiente fase desenvuelta después de aplicar el algoritmo Half-quadratic, se puede observar el desempeño de este algoritmo tanto en imágenes sintéticas como reales, con el cual se obtienen muy buenos resultados, destacando además su eficiencia computacional.

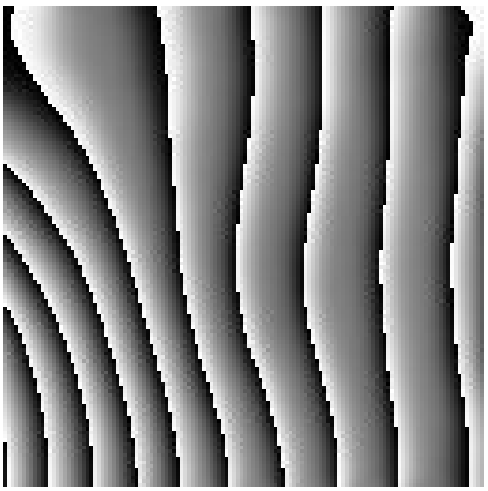


Figura 3.5. Fase envuelta con patrones de franjas abiertas, (imagen real)

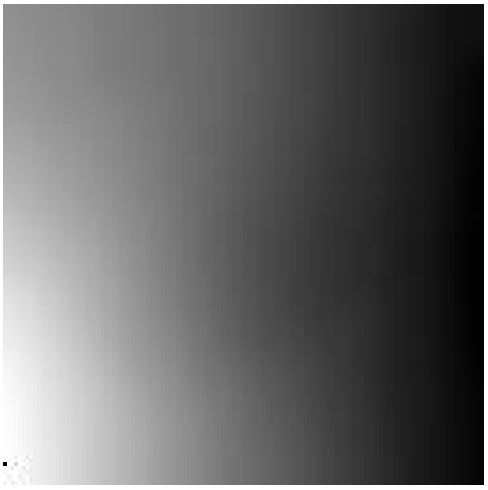


Figura 3.6. Fase desenvuelta con patrones de franjas abiertas aplicando Half-quadratic, (imagen real)

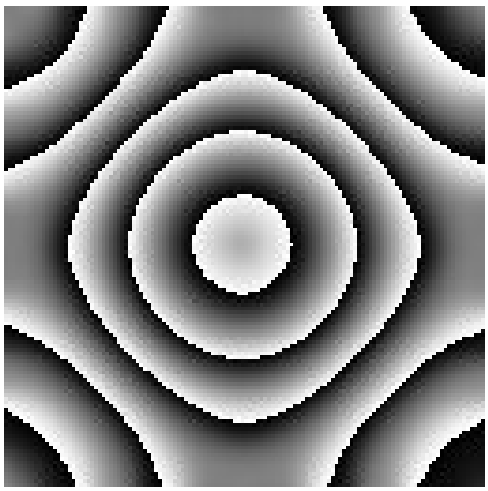


Figura 3.7. Fase envuelta con patrones de franjas cerradas (imagen sintética)

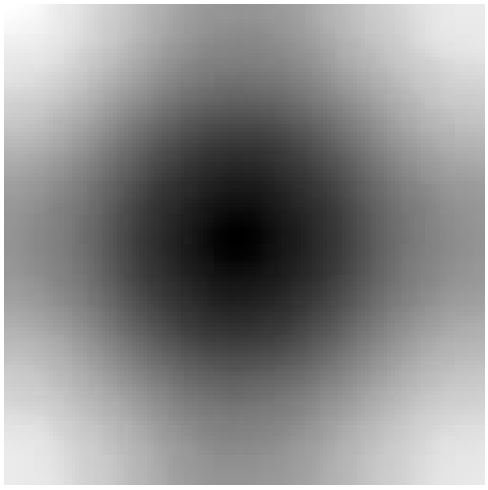


Figura 3.8. Fase desenvuelta con patrones de franjas cerradas aplicando Half-quadratic (imagen sintética)

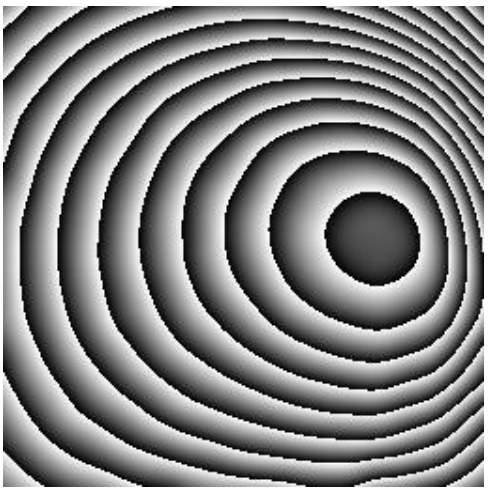


Figura 3.9. Fase envuelta con patrones de franjas cerradas, (imagen real)

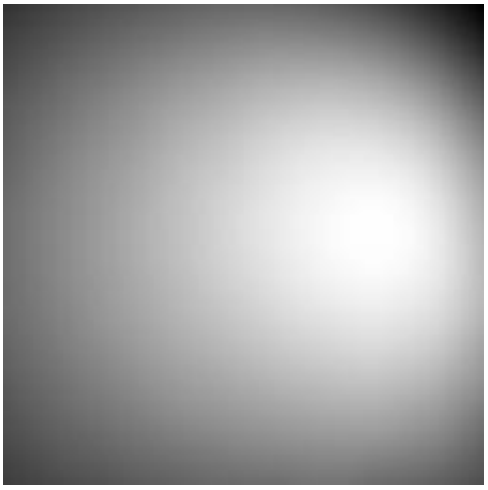


Figura 3.10. Fase desenvuelta con patrones de franjas cerradas aplicando Half-quadratic, (imagen real)

Capítulo 4

Orientación local

En buena medida el procesamiento digital de imágenes, se dedica al estudio y determinación de determinadas características o estructuras locales que son importantes para poder obtener información acerca de una imagen, por ejemplo: bordes, vértices, formas, orientación, etc.

La *orientación local* es precisamente una de estas características. Según Jähne en [10], la *orientación local ideal de una vecindad* se caracteriza por el cambio de los niveles de grises en una dirección mientras que en la otra dirección permanece constante. Para la estimación de la orientación local se han empleado varios métodos: Funciones de base radial [26] [27], filtros direccionales [26],[27], análisis de componentes principales [28].

Otros métodos para estimar la orientación local están relacionados con información de gradiente, o de derivadas en general, a estos últimos es a los que se le prestará atención en este trabajo, y aunque en este caso, el trabajo se centra en imágenes con patrones de franjas, estas ideas siguen siendo válidas para cualquier tipo de imagen, en particular, un importante campo de uso es en el *realce de huellas digitales*.

Según Larkin [29] los métodos que usan información de derivadas se pueden dividir en

- Primer orden (lineales), primer grado de derivada
- Segundo orden (cuadráticos), primer orden de derivada

- Primer orden, segundo grado de derivadas
- Segundo orden, segundo grado de derivadas

4.1. Primer Orden, primer grado derivación

Las técnicas basadas en gradientes para el análisis de la orientación fueron propuestas por Kass y Witkin (1987).

El problema fundamental de estas técnicas está en que las componentes del gradiente de la imagen son cercanas a cero en los máximos y mínimos (valles y crestas). En este caso la orientación pierde su dependencia de la fase [29] y queda en función del contraste local.

Otro de los problemas que tiene este estimador está relacionado con que es demasiado local [10], esto hace que sea muy sensible al ruido, por lo que se sugiere usar alguna medida de promedio, con el objetivo de que el estimador de la orientación local sea más robusto ante ruido.

Si denotamos la imagen por I, entonces el gradiente de la imagen es

$$\nabla I = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} \tag{4.1}$$

$$= \begin{bmatrix} I_x \\ I_y \end{bmatrix} \tag{4.2}$$

La información de orientación viene dada por

$$\alpha = \arctan \frac{I_y}{I_x} \tag{4.3}$$

Es decir, el ángulo $\alpha \in (-90^0, 90^0]$, tendría un rango de 180^0, y según Jähne [10], dado que la orientación está ligada a un ángulo, sería bueno caracterizar la orientación como una medida circular. Entonces define la *orientación* como un vector cuya *dirección* es $\beta = 2\alpha$ y su *magnitud* es cierta medida de certidumbre de la orientación, ej: la media. La definición anterior es asumida por varios investigadores, entre ellos Knutsson,

Granlund, Felsberg, Larkin etc.

4.2. Segundo Orden, primer grado derivación

4.2.1. Tensor de Estructura y Tensor de Inercia

El *tensor de estructura* [10], [11], [12], [13] es uno de los métodos para estimar la orientación local. Las primeras referencias aparecen en trabajos de en al menos dos grupos Försnert y Gülch [11], y por otro lado en los trabajos de Bigün y Granlund [30]. En el año 1989 Knutsson generaliza el uso del Tensor para representar la orientación a dimensiones mayores a 2.

La idea de los tensores se basa en encontrar una trasformación que cumpla con tres propiedades básicas: *Unicidad, Uniformidad y que sea polarmente separable* [30] [31]

- Unicidad: Este requisito tiene que ver con la eliminación de la discontinuidad producto al modulo 180^0 de la orientación, es decir, la transformación cumple con la propiedad

$$\mathbf{T}(\mathbf{x}) \quad = \quad \mathbf{T}(-\mathbf{x}) \tag{4.4}$$

  Lo que significa que $\mathbf{T}$ mapea a los vectores $\mathbf{x}$ y $-\mathbf{x}$ a un mismo vector $\mathbf{T}(\mathbf{x})$

- Uniformidad: La representación debe ser igualmente sensible a cambios de orientación en todas las orientaciones.

- Polarmente separable: Es decir la norma de $\mathbf{T}$ es independiente de la dirección de $\mathbf{x}$, y por tanto la norma del tensor es invariante bajo rotación de la señal.

$$||\mathbf{T}|| \quad = \quad f(||\mathbf{x}||) \tag{4.5}$$

Ejemplo: Se puede verificar que $\mathbf{T} \equiv \frac{1}{r}\mathbf{x} \otimes \mathbf{x} = \frac{1}{r}\mathbf{x} \bullet \mathbf{x}^T$, es decir, el tensor que se obtiene a través del producto diádico o producto tensorial, define una aplicación que cumple con las propiedades antes mencionadas.

Precisamente una de las formas de definir el tensor es usando la dirección del gradiente $\nabla I = (I_x, I_y)^T$, y el producto diádico.

$$\begin{aligned} \mathbf{T} &= \nabla I \otimes \nabla I & (4.6)\\ &= \begin{bmatrix} I_x I_x & I_x I_y \\ I_y I_x & I_y I_y \end{bmatrix} & (4.7) \end{aligned}$$

En la práctica el tensor se calcula sobre una ventana y se le aplica un suavizado, con el objetivo de hacer $\mathbf{T}$ no singular, quedando finalmente

$$\mathbf{T}_{\mathcal{E}} = \begin{bmatrix} \widetilde{I_x^2} & \widetilde{I_x I_y} \\ \widetilde{I_y I_x} & \widetilde{I_y^2} \end{bmatrix}. \quad (4.8)$$

A el tensor anterior se le da el nombre de *Tensor de Estructura*, otra forma de obtener la orientación local y que esta muy relacionada con la anterior es a través del *Tensor de Inercia*, en en libro de Jähne [10] se encuentran detalles de la deducción, en este caso la idea consiste en realizar todo el análisis en el dominio espectral y luego regresar al dominio espacial aplicando propiedades de la Transformada de Fourier, por lo que lo primero que se hace es aplicar la transformada de Fourier a toda la imagen, y luego en el dominio espectral, se ajusta una recta a la distribución de densidad que se tiene, minimizando el error cuadrático.

$$\hat{T} = \min_{\bar{k}} \int_{-\infty}^{\infty} d^2(k, \bar{k}) |\mathfrak{I}(k)|^2 dk; \quad (4.9)$$

donde $\bar{k}$ es un vector unitario y además

$$\begin{aligned} d(k, \bar{k}) &= k - (k^T \bar{k})\bar{k}, & (4.10)\\ d^2(k, \bar{k}) &= \bar{k}^T [I(k^T k) - k \otimes k]\bar{k}, & (4.11)\\ \mathfrak{I}(k) &= \mathfrak{F}(I). & (4.12) \end{aligned}$$

Por lo que se tiene

$$\hat{T} = \bar{k}^T \hat{\mathbf{T}} \bar{k}. \quad (4.13)$$

Para el caso de interés en 2D:

$$\hat{T}_{i,i} = \int_{-\infty}^{\infty} k_j^2 |\mathfrak{I}(k)|^2 dk, \tag{4.14}$$

$$\hat{T}_{i,j} = -\int_{-\infty}^{\infty} k_i k_j |\mathfrak{I}(k)|^2 dk, \;\; i,j \in \{1,2\} \;\; j \neq i. \tag{4.15}$$

Como $\bar{k}$ es unitario

$$\hat{T} = R(\bar{k}) = \bar{k}^T \hat{\mathbf{T}} \bar{k}, \tag{4.16}$$

se puede aplicar el *Principio de Rayleigh* [32], entonces $\hat{T}$ alcanza su valor máximo y su valor mínimo en los eigenvectores que se corresponden con los eigenvalores máximo y mínimo de $\hat{\mathbf{T}}$ respectivamente. Por lo que para hallar el mínimo basta determinar el eigenvalor más pequeño de la matriz $\hat{\mathbf{T}}$.

Observe que $\hat{\mathbf{T}}$ es una matriz simétrica y se corresponde con una forma cuadrática cuyo eje principal está en la dirección del eigenvector que minimiza $\hat{T}$.

El resultado anterior es posible expresarlo en el dominio espacial. Si se aplica la transformada inversa de fourier a la ecuación 4.13 sobre una ventana, que actuaría como una máscara de suavizamiento se obtiene la expresión.

$$\mathbf{T}_{\mathcal{I}} = \begin{bmatrix} \widetilde{I_y^2} & -\widetilde{I_x I_y} \\ -\widetilde{I_y I_x} & \widetilde{I_x^2} \end{bmatrix}; \tag{4.17}$$

que es muy similar a la ecuación 4.8, salvo en el signo de la diagonal no principal de la matriz, y en la diagonal principal están intercambiadas las derivadas parciales.

Al tensor $\mathbf{T}_{\mathcal{I}}$ se le conoce como *Tensor de Inercia*, y está relacionado con el tensor de estructura mediante las igualdades siguientes:

$$\mathbf{T}_{\mathcal{I}} = traza(\mathbf{T}_{\mathcal{E}})I - \mathbf{T}_{\mathcal{E}}, \tag{4.18}$$

$$\mathbf{T}_{\mathcal{E}} = traza(\mathbf{T}_{\mathcal{I}})I - \mathbf{T}_{\mathcal{I}}. \tag{4.19}$$

Es fácil ver que $traza(\mathbf{T}_{\mathcal{E}}) = traza(\mathbf{T}_{\mathcal{I}})$, es más $\mathbf{T}_{\mathcal{E}}$ y $\mathbf{T}_{\mathcal{I}}$ tienen los mismos eigenvalores y los mismos eigenvectores.

En efecto, sea $x_i^{\mathcal{I}}$ el eigenvector correspondiente al eigenvalor $\lambda_i^{\mathcal{I}}$ del tensor de inercia, donde $i \in \{1, 2\}$.

De la segunda igualdad se tiene que:

$$\begin{aligned} \mathbf{T}_{\mathcal{E}} x_i^{\mathcal{I}} &= [traza(\mathbf{T}_{\mathcal{I}})I - \mathbf{T}_{\mathcal{I}}]x_i^{\mathcal{I}} && (4.20) \\ &= traza(\mathbf{T}_{\mathcal{I}})x_i^{\mathcal{I}} - \lambda_i^{\mathcal{I}} x_i^{\mathcal{I}} && (4.21) \\ &= [traza(\mathbf{T}_{\mathcal{I}}) - \lambda_i^{\mathcal{I}}]x_i^{\mathcal{I}}. && (4.22) \end{aligned}$$

Por lo que se concluye que ambos tensores tienen los mismos eigenvectores. Como se puede hallar la traza de una matriz en función de los eigenvalores

$$\begin{aligned} traza(\mathbf{T}_{\mathcal{I}}) &= \sum_{i=1}^{2} \lambda_i^{\mathcal{I}} && (4.23) \\ &= \lambda_1^{\mathcal{I}} + \lambda_2^{\mathcal{I}}. && (4.24) \end{aligned}$$

También se concluye que tienen los mismos eigenvalores, esto nos dice que se pueden usar de forma indistinta tanto el tensor de inercia como el tensor de estructura para estimar la orientación local.

Otra vía interesante y que resulta equivalente a la anterior puede ser vista en [33], en este caso todo el desarrollo se realiza en el dominio espacial. De forma similar, la idea es hallar el minimizador del error cuadrático a lo largo de una dirección unitaria $\bar{k}$.

$$\begin{aligned} \varepsilon(\bar{k}) &= \int_{\mathcal{W}} (\bar{k}\nabla I)^2 d\mathcal{W} && (4.25) \\ &= \bar{k}^T \mathbf{T} \bar{k}, \;\; s.a.\; \bar{k}^T \bar{k} = 1. && (4.26) \end{aligned}$$

donde $\mathbf{T}$ es el tensor

$$\begin{aligned} \mathbf{T} &= \int_{\mathcal{W}} \nabla I \nabla I^T d\mathcal{W} && (4.27) \\ &= \int_{\mathcal{W}} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} d\mathcal{W}. && (4.28) \end{aligned}$$

que coincide con 4.8. Para resolver el problema de optimización 4.26, es decir, para minimizar el error cuadrático se pudiera aplicar el principio de Rayleigh, o también se puede buscar el mínimo de la siguiente función, definida a través del Lagrangiano

$$E(\bar{k}) \;=\; \bar{k}^T\mathbf{T}\bar{k} + \lambda(1 - \bar{k}^T\bar{k}); \tag{4.29}$$

lo que es equivalente a resolver el problema de eigenvalores

$$\mathbf{T}\bar{k} \;=\; \lambda\bar{k}, \;\; s.a \;\; \bar{k}^T\bar{k} = 1. \tag{4.30}$$

y se llega al mismo resultado, es decir, que el minimizador es el eigenvector correspondiente al menor eigenvalor.

Para resolver el problema de los eigenvalores, se usa una manera muy fácil que es a través de rotaciones,

$$\Lambda \;=\; R_{-\beta}\mathbf{T}R_{\beta}. \tag{4.31}$$

donde $\Lambda = diag(\lambda_1, \lambda_2)$ es una matriz diagonal y R_α es una matriz de rotación de ángulo α.

En particular para el caso $\mathfrak{R}^2$, los eigenvalores están dados por:

$$\lambda_{1,2} \;=\; \frac{1}{2}[T_{11} + T_{22} \pm \sqrt{(T_{11} - T_{22})^2 + T_{12}^2}]. \tag{4.32}$$

Desarrollando la expresión anterior, comparando y resolviendo el sistema de ecuaciones para la incógnita β, se llega a que el ángulo de rotación se calcula mediante la expresión siguiente, en término de las componentes del tensor

$$\tan 2\beta \;=\; \frac{2T_{12}}{T_{11} - T_{22}}. \tag{4.33}$$

Una expresión para medir la orientación local es la *medida de coherencia*. La medida de coherencia es un valor en el intervalo $[0, 1]$ y se calcula a partir de las componentes

del tensor por la fórmula

$$\mathcal{C} = \frac{(T_{11} - T_{22})^2 + 4T_{12}^2}{(T_{11} + T_{22})^2} \tag{4.34}$$

$$= \left[\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right]^2. \tag{4.35}$$

Si $\lambda_1 = 0$ y $\lambda_2 \neq 0$ entonces $\mathcal{C} = 1$ se dice que hay una buena orientación, por otro lado, si $\lambda_1 = \lambda_2 > 0$ entonces $\mathcal{C} = 0$, no existe una dirección de preferencia y por tanto se dice que la orientación es isotrópica (en realidad se pudieran definir otras medidas que digan cuan orientado está un pixel).

4.3. Definiciones

A partir de ahora y a los efectos de este trabajo, se le llamará *pixelón* a la terna $(\mathcal{W}, \mathcal{O}, \mathcal{B})$ donde $\mathcal{W}$ es cierta ventana, por lo regular cuadrada, $\mathcal{O}$ es la orientación media global de la ventana y la medida de certidumbre $\mathcal{B}$ de dicha orientación.

Para obtener la orientación global se propone calcular una media sobre toda la ventana, así se tiene lo siguiente

Tensor de inercia sobre un pixelón

Se llama *Tensor de inercia sobre un pixelón* a la matriz cuyas entradas son los valores medios de las entradas de cada Tensor de inercia asociado a cada pixel perteneciente al pixelón, es decir.

$$\mathbf{T}^{\mathcal{P}} = \frac{1}{|\mathcal{P}|} \sum_{r \in \mathcal{P}} \mathbf{T}^r \tag{4.36}$$

donde $\mathbf{T}^r$ se calcula usando la ecuación 4.17.

Orientación de un pixelón

Se llamará *orientación local* de un pixelón al eigenvector que minimiza

$$T^{\mathcal{P}} = \mathbf{u}^T \mathbf{T}^{\mathcal{P}} \mathbf{u}, \; s.a. \; \mathbf{u}^T \mathbf{u} = 1 \tag{4.37}$$

es decir, sería el vector $\mathcal{O}_{\mathcal{P}}$ cuyas coordenadas están dadas por

$$\mathcal{O}_{\mathcal{P}} = \begin{bmatrix} \mathbf{T}^{\mathcal{P}}_{11} - \mathbf{T}^{\mathcal{P}}_{22} \\ 2\mathbf{T}^{\mathcal{P}}_{12} \end{bmatrix} \tag{4.38}$$

donde el ángulo de orientación $\boldsymbol{\beta}^{\mathcal{P}}$ se determina aplicando una fórmula similar a 4.33, por lo que

$$\boldsymbol{\beta}^{\mathcal{P}} = \frac{1}{2} \arctan \left[\frac{2\mathbf{T}^{\mathcal{P}}_{12}}{\mathbf{T}^{\mathcal{P}}_{11} - \mathbf{T}^{\mathcal{P}}_{22}} \right] \tag{4.39}$$

Las Figuras 4.1, 4.2 y 4.3 muestran imágenes reales, sus mapas de coherencia y finalmente el mapa de orientación por pixelones, en cada caso se obtuvieron un total de $16 * 16 = 256$ pixelones, se puede verificar que se logra recuperar la orientación correcta, es decir, la que una persona esperaría visualmente. Lo anterior nos indica, al menos experimentalmente, que podemos usar esta definición de orientación.

Los experimentos anteriores fueron sobre imágenes reales sin patrones de franja, no obstante, el interés es trabajar con imágenes con patrones de franjas. En la Figura 4.4 se muestra una imagen real con franjas abiertas y se logra estimar la orientación por pixelones, de forma similar en la Figura 4.5 se muestra un experimento sintético con un patrón de franjas con franjas cerradas y también se obtiene muy bien la orientación.

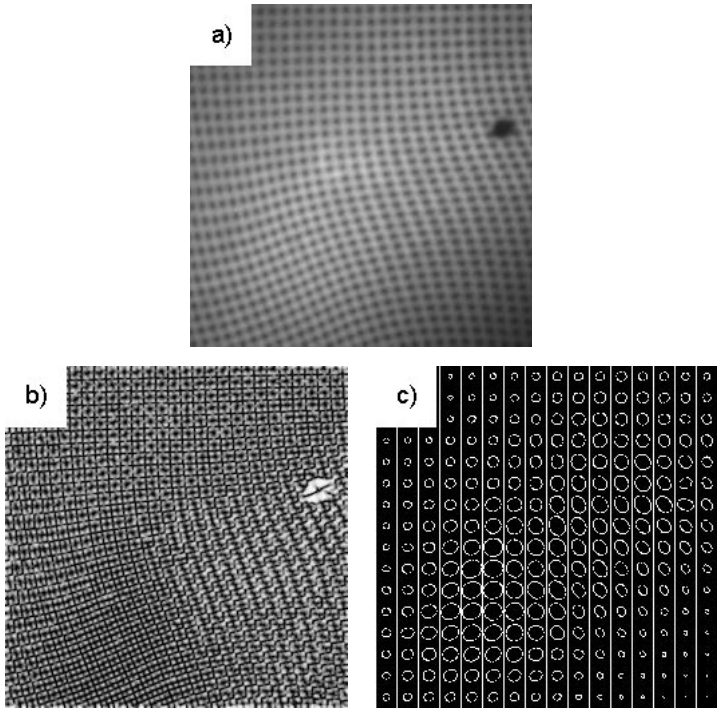


Figura 4.1. a) Imagen real b) Mapa de coherencia c)Mapa de orientación por pixelones.

Medida de coherencia de un pixelón

Se llamará *Medida de coherencia* de un pixelón a la medida tomada sobre el tensor de inercia del pixelón.

$$\mathcal{C}^{\mathcal{P}} = \frac{(T_{11}^{\mathcal{P}} - T_{22}^{\mathcal{P}})^2 + 4(T_{12}^{\mathcal{P}})^2}{(T_{11}^{\mathcal{P}} + T_{22}^{\mathcal{P}})^2} \tag{4.40}$$

$$= \left[\frac{\lambda_1^{\mathcal{P}} - \lambda_2^{\mathcal{P}}}{\lambda_1^{\mathcal{P}} + \lambda_2^{\mathcal{P}}}\right]^2 \tag{4.41}$$

donde $\lambda_1^{\mathcal{P}}$ y $\lambda_2^{\mathcal{P}}$ son los eigenvalores de $\mathbf{T}^{\mathcal{P}}$.

La *medida de coherencia* $\mathcal{C}^{\mathcal{P}}$ nos dice cuan orientado esta el pixelón $\mathcal{P}$, en la práctica es una medida que permite identificar si un pixelón está *orientado* o no.

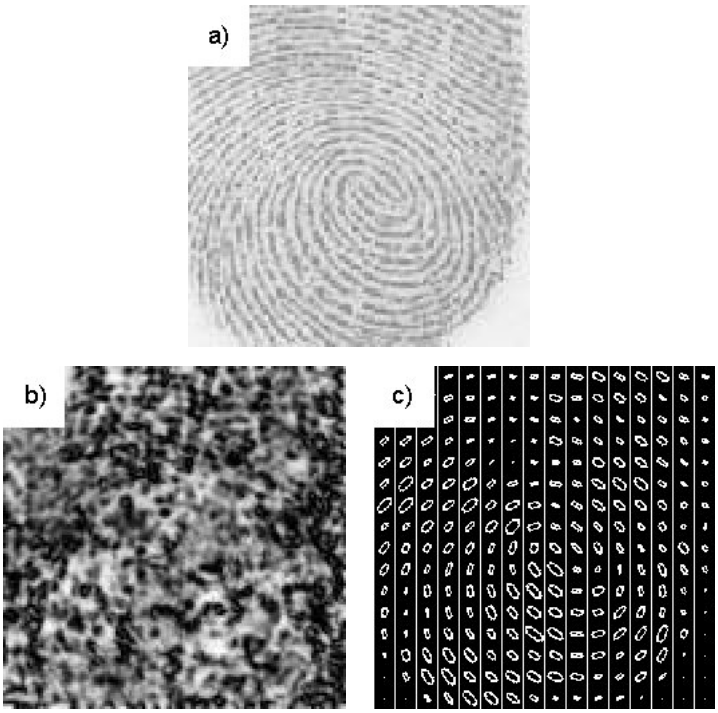


Figura 4.2. a) Imagen real de huella digital b) Mapa de coherencia c)Mapa de orientación por pixelones.

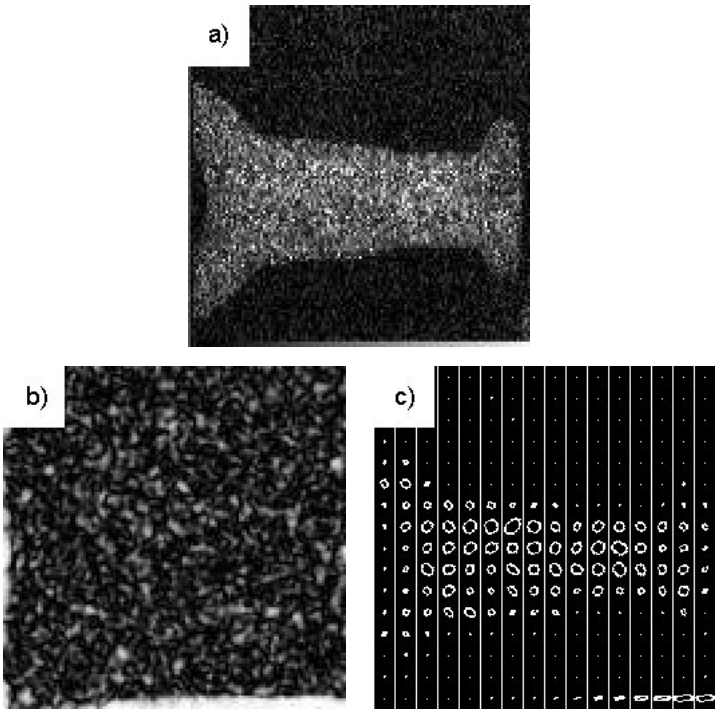


Figura 4.3. a) Imagen real con moteado speckle b) Mapa de coherencia c)Mapa de orientación por pixelones.

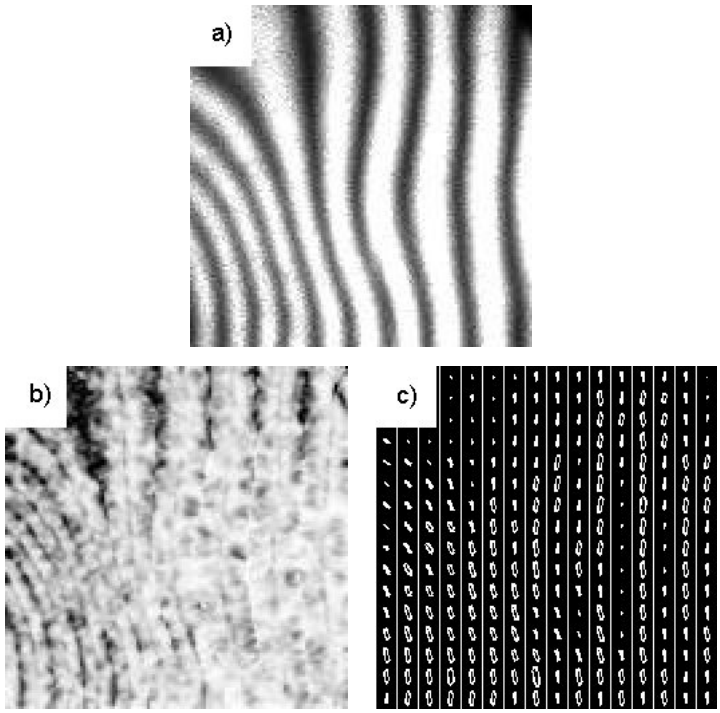


Figura 4.4. a) Imagen real con Franjas abiertas b) Mapa de coherencia c)Mapa de orientación por pixelones.

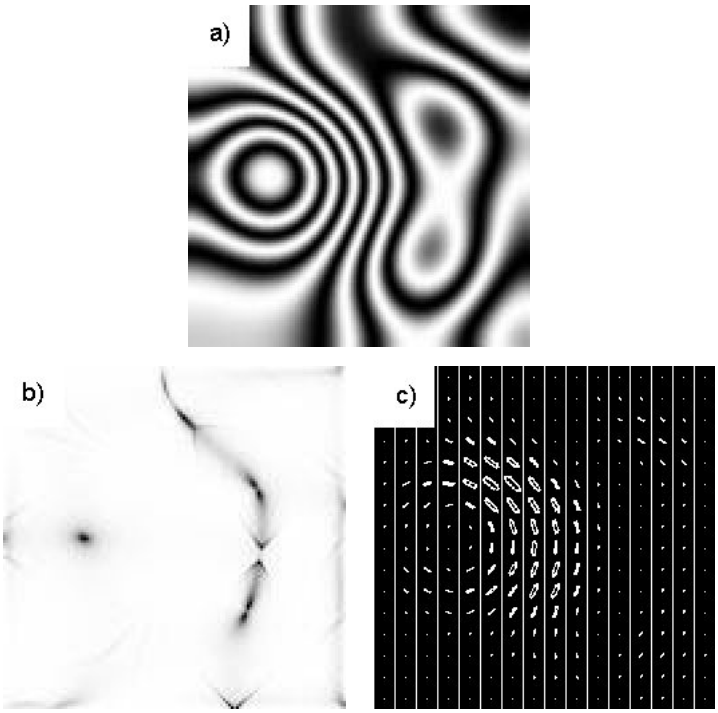


Figura 4.5. a) Imagen sintética con Franjas cerradas b) Mapa de coherencia c)Mapa de orientación por pixelones.

Medida de bondad

La *Medida de bondad relativa* es una medida de comparación que establece una relación de orden total con la relación de orden $\leq$, para el conjunto de *pixelones* $\mathfrak{P}$ de una imagen.

La *Medida de bondad* se define mediante

$$\mathcal{B}^{\mathcal{P}} = \lambda_M^{\mathcal{P}} \mathcal{C}^{\mathcal{P}} \tag{4.42}$$

donde $\lambda_M^{\mathcal{P}} = \text{máx}\{\lambda_1^{\mathcal{P}}, \lambda_2^{\mathcal{P}}\}$. Por convenio se asume que $\lambda_2^{\mathcal{P}}$ es el mayor eigenvalor, por lo que $\lambda_M^{\mathcal{P}} = \lambda_2^{\mathcal{P}}$.

De este modo decimos que $\mathcal{P}_1 \leq \mathcal{P}_2$ si y solo si $\mathcal{B}^{\mathcal{P}_1} \leq \mathcal{B}^{\mathcal{P}_2}$. O lo que es lo mismo decimos que un pixelón $\mathcal{P}_2$ está mejor orientado que $\mathcal{P}_1$ si su medida de bondad $\mathcal{B}^{\mathcal{P}_2}$ es mayor que la medida de bondad $\mathcal{B}^{\mathcal{P}_1}$, si $\mathcal{B}^{\mathcal{P}_2}$ es menor que $\mathcal{B}^{\mathcal{P}_1}$ entonces $\mathcal{P}_2$ está peor orientado que $\mathcal{P}_1$, en caso contrario, la calidad de la orientación es la misma, aunque por supuesto, esto no dice nada en relación con la orientación local en si misma de cada pixelón, solo es una medida de comparación que nos permite diferenciar la calidad de orientación entre dos pixelones de igual coherencia, considerando 'mejor' orientado al que tenga mayor eigenvalor principal.

Las medidas de coherencia y de bondad de un pixelón permiten construir *mapas de calidad* relacionados con la orientación local de los pixelones.

En la Figura 4.6 se pueden observar algunos de estos mapas, los mismos serán de mucha utilidad en el capítulo 5, cuando se estime la fase inicial de un patrón de franjas.

La Figura 4.7 muestra una imagen sintética de patrones de franja con moteado speckle, el mapa de orientación por pixelones logra recuperar la orientación del patrón de franjas en cada una de las ventanas, a pesar de que la imagen tiene mucho ruido.

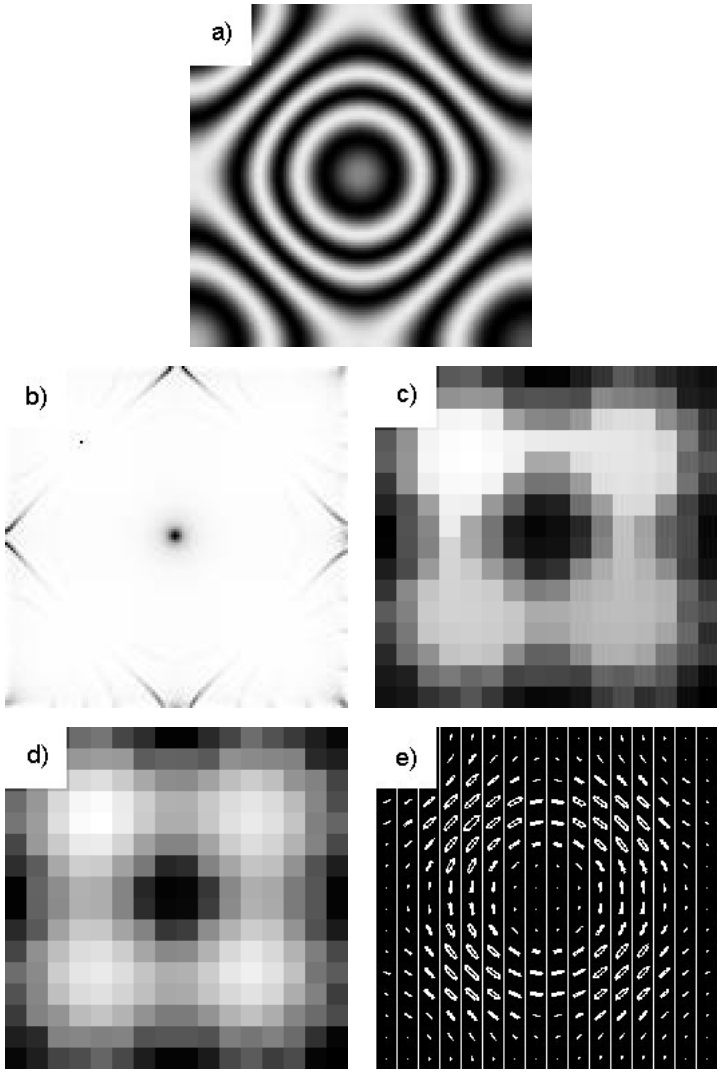


Figura 4.6. a) Imagen sintética con patrones de franjas cerradas b) Mapa de coherencia c) Mapa de medida bondad relativa por pixelón d) Mapa de coherencia por pixelón e)Mapa de orientación por pixelón.

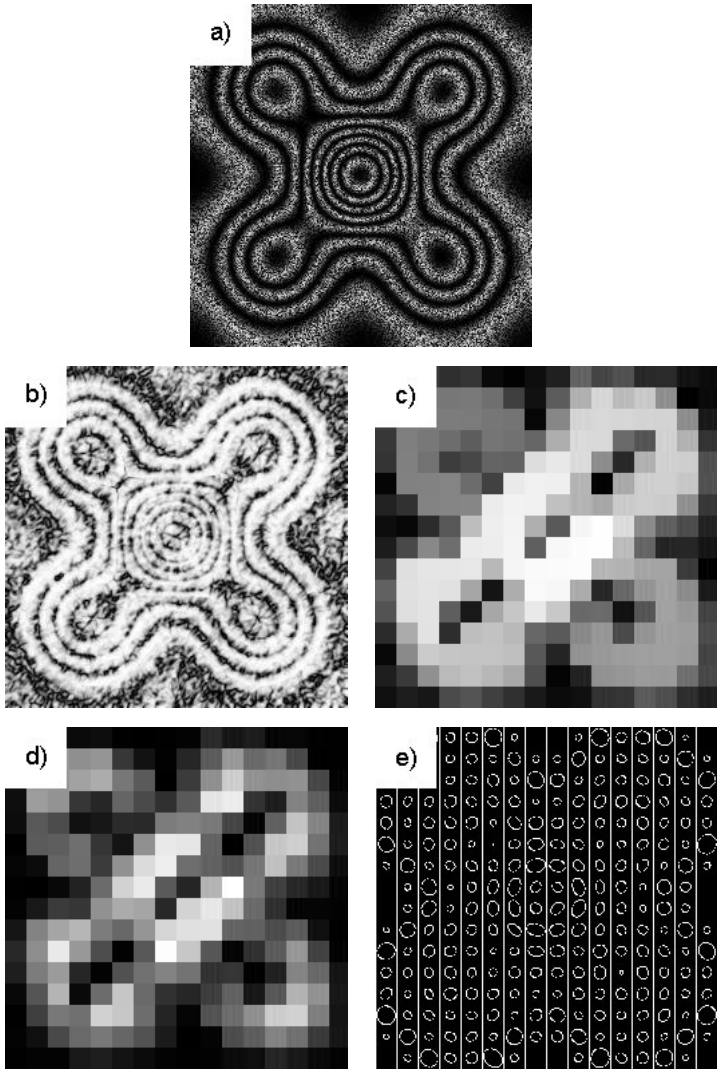


Figura 4.7. a) Imagen sintética de patrones de franjas con moteado speckle b) Mapa de coherencia c) Mapa de medida bondad relativa por pixelón d) Mapa de coherencia por pixelón e)Mapa de orientación por pixelón.

Capítulo 5

Trabajo realizado

En este capítulo se mostrará un algoritmo para la obtención de la fase usando un solo patrón de franjas. El algoritmo propuesto consta de dos etapas.

- **Etapa A.** Se estima una *fase inicial* para toda la imagen y constituye la parte principal del algoritmo.
- **Etapa B.** Refinamiento de la *fase inicial estimada* para obtener la *fase aproximada*. En esta etapa lo que se desea mostrar es que es posible obtener una fase aproximada a partir de la *fase inicial estimada* en la Etapa A. Aquí se propondrá una estrategia basada en pirámides gaussianas que se obtienen como resultado de aplicar un suavizado y un submuestreo, aunque se pudieran proponer otros algoritmos o estrategias de refinamiento.

Como se verá a través de un ejemplo, la **Etapa B** no siempre es necesaria. Si el interferograma no es muy complicado, en el sentido de que esté normalizado y tenga poco ruido, entonces prácticamente al aplicar la **Etapa A** se obtienen buenos resultados. A pesar de esto, se sugiere en este último caso aplicar un ligero refinamiento.

5.1. Algoritmo para la Etapa A

La idea general consiste en obtener una fase inicial, sin necesidad de usar muchos recursos y que sea eficiente desde el punto de vista computacional, tratando de

explotar al máximo la información que brindan los filtros de cuadratura (Método de Takeda) y por otro lado usar la información de orientación que está presente en el inteferograma.

De este modo lo primero que se hace es obtener la información de fase y de orientación local, a partir de lo cual se concentra dicha información en los llamados *pixelones* que guiarán el proceso de formación de la imagen de *fase inicial*.

Dentro de los parámetros que hay que definir están las dimensiones de los *pixelones*. Para los ejemplos que se muestran, se han tomado las dimensiones de modo que el número de pixelones en todos los casos sea $16 \bullet 16 = 256$. En cuanto a las dimensiones de las imágenes de prueba, se trabajará con aquellas cuyas dimensiones sean potencias de 2, puesto que al aplicar el Método de Takeda se usará la Transformada Rápida de Fourier(FFT), en particular serán imágenes de $n \times n$, por lo que las dimensiones de los pixelones serían $\frac{n}{16} \times \frac{n}{16}$. Es importante notar que el número de pixelones es un parámetro dado por el usuario. Aunque en este trabajo se implementó el filtro de cuadratura en el dominio de las frecuencias, también podría hacerse en el dominio espacial, y para el caso de la FFT existen algoritmos que no imponen la restricción de usar imágenes cuyas dimensiones sean potencia de 2, como por ejemplo: Fastest Fourier Transform in the West (FFTW) [34].

Otro elemento a tener en cuenta es el de la coherencia de los pixelones; esta información se usará en el proceso de acoplamiento de los pixelones, como una medida de orden en el momento de formar la fase inicial. Esa característica se pudiera usar también para discriminar los pixelones con "*mala*" orientación, es decir, aquellos pixelones cuya coherencia no rebase cierto umbral, y entonces para estos casos lo mejor sería realizar un proceso de interpolación que podría estar presente en la etapa de refinamiento.

Algoritmo (Etapa A)

1. Aplicar el Método de Takeda en la dirección vertical y horizontal.
2. Determinar el Tensor de Inercia para cada pixel en toda la imagen
3. Establecer los pixelones asociados a cada ventana en la imagen, es decir,

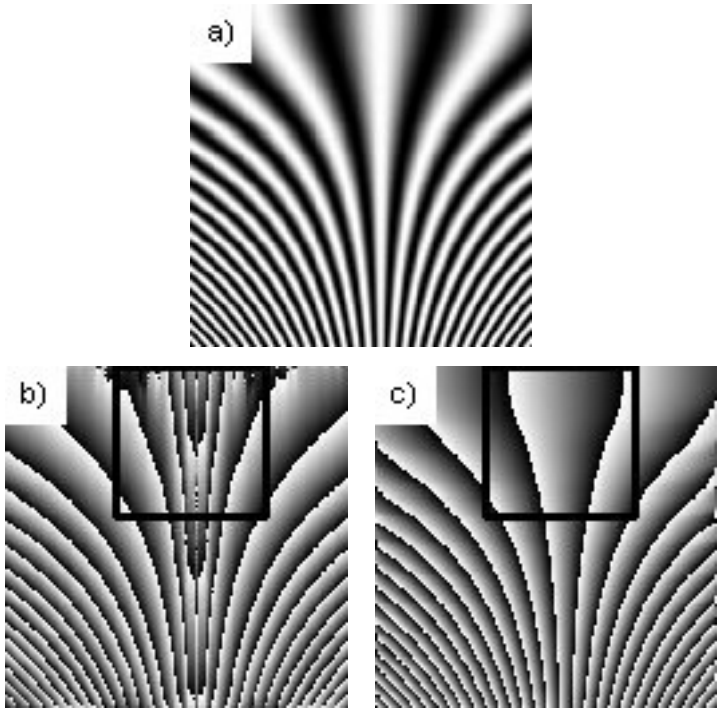


Figura 5.1. a)Interferograma con franjas abiertas b) Fase resultante después de aplicar el filtro horizontal c) Fase resultante después de aplicar el filtro vertical.

determinar la coherencia, la orientación, y la medida de bondad de cada pixelón.

4. Ordenar los pixelones, usando la medida de calidad de orientación propuesta.

5. Formar imagen de magnitud y de fase, a partir de los resultados que brinda el método de Takeda, la coherencia y la orientación por pixelón.

6. Desenvolver cada ventana asociada a un pixelón y acoplarlas para formar una *fase inicial*.

7. Aplicar postprocesamiento (suavizado) para obtener la *fase inicial estimada*.

Algunos de los elementos que intervienen en esta etapa han sido vistos en capítulos anteriores, por lo que no se entrará en detalles en este caso y se mostrarán los resultados de aplicar dichos pasos. Solo los pasos que no se hallan detallado con anterioridad se discutirán con más precisión. En principio se mostrará a través de un ejemplo la necesidad del primer paso, al menos en el caso que se realice un proceso automático de recuperación de la fase. En la Figura 5.1(a) se muestra una imagen

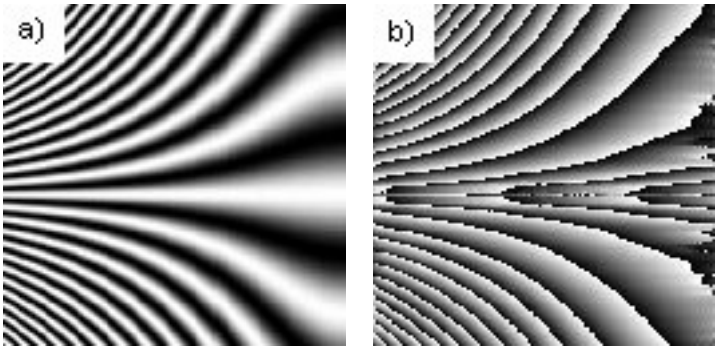


Figura 5.2. a) Imagen rotada 90^0 en sentido horario, b) Fase resultante al aplicar el filtro vertical.

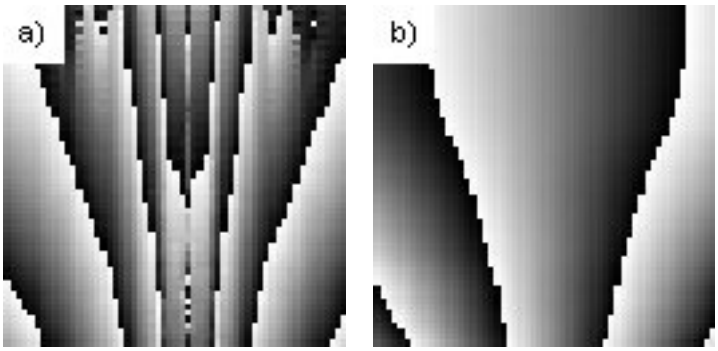


Figura 5.3. Ampliación de las zonas seleccionadas en la Figura 5.1 a) Corresponde a la Figura 5.1(b), b) Corresponde a la Figura 5.1(c).

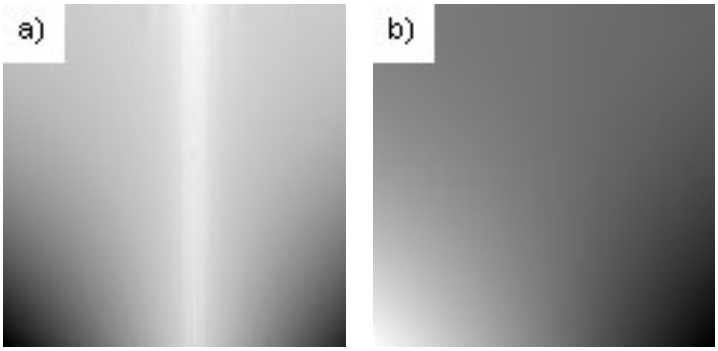


Figura 5.4. Fases desenvueltas aplicando Half Quadratic a) A la fase envuelta de la Figura 5.1(b) , b) A la fase envuelta de la Figura 5.1(c).

sintética con franjas abiertas, y las imágenes de fase envuelta, Figuras 5.1(b) y 5.1(c), que se obtienen después de aplicar el método de Takeda en las direcciones horizontal y vertical respectivamente. Es claro que si se aplicara el método de Takeda en una sola dirección de forma automática no siempre daría buenos resultados, por ejemplo: si el filtro se aplica por defecto en la dirección vertical, la salida hubiera sido el inciso (c) de la Figura 5.1. Por otro lado, si en lugar de entrar la imagen original, esta se hubiera rotado 90^0 grados en sentido horario, entonces la salida del algoritmo sería el inciso (b). En la Figura 5.2 podrá ver el resultado. También se muestra en la Figura 5.3 una ampliación de las zonas marcadas en la Figura 5.1, donde se ve con más claridad el efecto de no aplicar el filtro en la dirección de las franjas, así como el resultado de aplicar al algoritmo Half-Quadratic para desenvolver las fases que se obtienen al aplicar los filtros en ambas direcciones, ver Figura 5.5.

Después de aplicar el primer paso, le sigue hallar el tensor de inercia para cada pixel en toda la imagen, y con el resultado anterior se establecen los pixelones en toda la imagen, es decir, se determina la coherencia y la orientación por pixelón, Figura 5.4.

Los pasos 4 y 5 se pudieran intercambiar, pues estos no tienen relación directa, y su objetivo principal es tributar información al sexto paso.

Ahora se construye la *medida de bondad*, que además de indicar la orientación de los pixelones, nos dice entre dos pixelones que tengan igual coherencia, a cual vamos a considerar como “mejor” orientado. Para tal fin, usamos como criterio, el seleccionar,

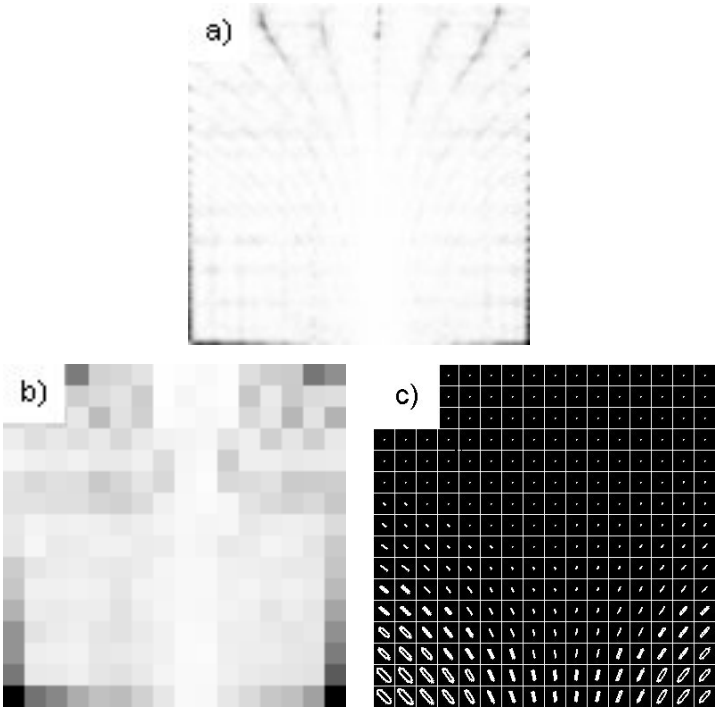


Figura 5.5. a) Mapa de coherencia, b) Mapa de coherencia por pixelón, c) Mapa de orientación por pixelón.

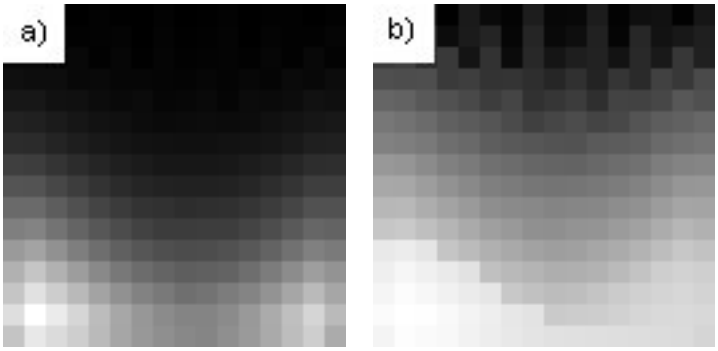


Figura 5.6. a) Mapa de medida de bondad, b) Mapa de propagación.

de los dos, al que tenga el eigenvalor principal más grande en su tensor de inercia. Con esto se construye un *mapa de propagación*, que será usado para formar la imagen de fase inicial, Figura 5.6. Para el ejemplo que se está analizando podrá ver el orden que se seguirá en la Tabla 5.1. El mapa de propagación expresa el orden en que se realizará el acoplamiento, el mismo se pudiera implementar a través de una lista, donde inicialmente se ubica el pixelón de mayor medida de bondad, luego se añade a lista el vecino que tenga mayor medida de bondad, lo anterior se repite hasta que el número de elementos de la lista coincida con el número de pixelones. A continuación se muestra en seudocódigo.

```
Procedure PropagationMap (MB, m).
   // MB(1:m) contiene las m entradas, MB -Medida de Bondad.
    mapa   ← ∅ // Inicializa la solución en vacío.
    x      ←  selec_mejor (MB)
    mapa   ←  añade (mapa, x)
    MB     ←  borra (MB, x)
    for i= 2 to m do
       x      ←  selec_mejor_vecino (MB, mapa)
       mapa   ←  añade (mapa, x)
       MB     ←  borra (MB, x)
    repeat
    return (mapa)
end PropagationMap
```

256	240	233	255	230	246	252	227	249	248	228	253	241	239	254	234
211	243	213	244	229	220	250	219	247	242	223	251	217	238	232	225
197	193	212	199	236	206	245	216	235	226	222	237	215	231	201	224
177	179	178	198	194	205	204	210	218	208	221	203	214	191	200	182
155	158	168	175	183	195	187	207	202	196	209	189	190	185	169	176
137	142	149	160	165	171	181	192	186	180	188	184	172	163	159	153
127	130	135	146	151	162	167	170	173	174	166	164	161	150	145	141
104	107	117	128	134	144	148	154	156	157	152	147	143	131	125	123
88	89	99	110	119	129	133	138	140	139	136	132	126	116	106	105
73	72	83	93	102	114	118	120	124	122	121	115	111	100	91	92
58	55	65	75	86	97	103	112	113	109	108	101	95	84	74	79
23	20	27	59	69	80	87	94	98	96	90	85	77	67	57	62
17	10	16	22	30	64	71	76	82	81	78	70	61	50	46	49
11	4	9	15	21	26	56	63	68	66	60	53	48	44	41	45
5	1	2	7	13	18	25	29	54	52	51	47	42	39	38	40
12	3	6	8	14	19	24	28	31	32	33	34	35	36	37	43

Tabla 5.1. Tabla sobre el mapa de propagación

También se forma una imagen de fase y otra de magnitud, a partir de la información de coherencia y orientación por pixelón, seleccionando el pixelón a partir de las imágenes de fase y magnitud que aportan los filtros horizontal y vertical, y se toma aquel que tenga una mayor coherencia. A estas imágenes intermedias se le aplica un proceso de desenvolvimiento por cada pixelón para obtener la Figura 5.7. En este ejemplo se usó Half Quadratic para desenvolver los pixelones, en muchos casos es suficiente desenvolver con Poisson.

El paso que continúa es el de formar la *fase inicial*, esta se formará a partir de un proceso de acoplamiento, siguiendo el mapa de propagación de modo que dos pixelones contiguos se acoplen ajustando una posible diferencia de DC, así como un posible cambio de signo. El proceso de acoplamiento comienza ubicando el pixelón de mayor medida de bondad, para el próximo se pueden dar como máximo 4 casos, que el pixelón a acoplar lo haga por arriba, por abajo, a la derecha o a la izquierda de alguno de los que ya están acoplado. En la Figura 5.8 se ilustra un ejemplo, donde el pixelón A es que se toma como origen y el B es el que se va acoplar, para ello, se forma una pequeña banda de acoplamiento de m filas y 6 columnas, donde m representa la dimensión de los pixelones, por supuesto, lo anterior es para este ejemplo, pues en caso de que el acoplamiento sea por arriba o por abajo, entonces la banda tendría 6 filas y m columnas . Se crearía entonces una imagen s de dimensiones $m \times 6$ ($s_{m\times 6}$)

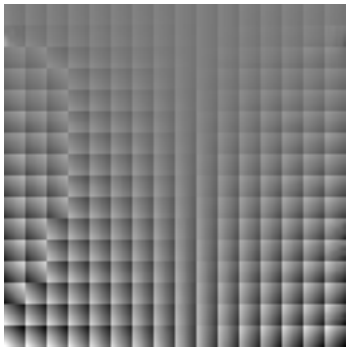


Figura 5.7. Imagen de fase por pixelón.

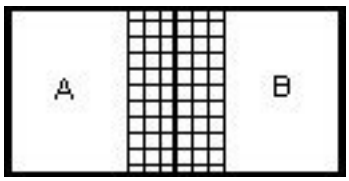


Figura 5.8. Banda de acoplamiento.

la cual servirá para determinar el dc^* y el $signo^*$ óptimos, que se pueden calcular a través de las siguientes expresiones:

$$\begin{aligned} d_1^* &= \arg\min_{d\in\Re} S_1(d), \\ d_2^* &= \arg\min_{d\in\Re} S_2(d), \end{aligned}$$

donde

$$\begin{aligned} S_1(d) &= \sum_{i=0}^{m-1} \Big\{ \{s[i][0] - 3s[i][1] + 3s[i][2] - (s[i][3] + d)\}^2 + \\ &\quad \{s[i][1] - 3s[i][2] + 3(s[i][3] + d) - (s[i][4] + d)\}^2 + \\ &\quad \{s[i][2] - 3(s[i][3] + d) + 3(s[i][4] + d) - (s[i][5] + d)\}^2 \Big\}, \end{aligned} \tag{5.1}$$

$$\begin{aligned} S_2(d) &= \sum_{i=0}^{m-1} \Big\{ \{s[i][0] - 3s[i][1] + 3s[i][2] + (s[i][3] - d)\}^2 + \\ &\quad \{s[i][1] - 3s[i][2] - 3(s[i][3] - d) + (s[i][4] - d)\}^2 + \\ &\quad \{s[i][2] + 3(s[i][3] - d) - 3(s[i][4] - d) + (s[i][5] - d)\}^2 \Big\}. \end{aligned} \tag{5.2}$$

$S_1(d)$ puede interpretarse como una discretización de:

$$\mathcal{S}^{(3)}(d) = \int_{r\in\Omega} |\frac{\partial^3}{\partial r^3}\hat{s}(r,d)|^2 dr, \tag{5.3}$$

donde Ω es la banda de acoplamiento, y $\hat{s}(r, d(r))$ se obtiene mediante la siguiente expresión:

$$\hat{s}(r, d(r)) = s(r) + d(r),$$

$s(r)$ es la imagen definida en la banda de acoplamiento y $d(r)$ representa una variación de intensidad que se define como:

$$d(r) = \begin{cases} d, & r \in B, \\ 0, & r \notin B. \end{cases} \tag{5.4}$$

La ecuación 5.3 penaliza las terceras derivadas para la fase en la región de acoplamiento, o lo que es lo mismo, penaliza las segundas derivadas de la frecuencia, por lo tanto, cuando se minimice dicha ecuación respecto a d, lo que se está pidiendo son cambios suaves en la frecuencia.

Para $S_2(d)$ se tiene una expresión similar a 5.3, pero ahora hay que redefinir $\hat{s}(r, d(r))$, de modo que además incluya una modificación del signo, obteniéndose de esta manera:

$$\hat{s}(r, d(r)) = sng(r)s(r) + d(r),$$

donde $d(r)$ se define como antes y $sng(r)$ de la forma siguiente:

$$sng(r) = \begin{cases} -1, & r \in B, \\ 1, & r \notin B. \end{cases} \tag{5.5}$$

En la práctica, $S_1(d)$ se obtiene al sumar un DC = d a la parte derecha de $s_{m\times 6}$ perteneciente al pixelón B a acoplar y $S_2(d)$ resulta de cambiarle el signo a la parte derecha de $s_{m\times 6}$ perteneciente al pixelón B y luego sumarle un DC = d. Como se dijo anteriormente, la idea es determinar el DC óptimo y el signo óptimo para realizar el acoplamiento del pixelón B. Entonces el dc^* se obtiene mediante:

$$dc^* = d^*_{i^*},$$

donde el índice óptimo i^* se calcula a través de:

$$i^* = \arg \min_{i\in\{1,2\}} S_i(d^*_i),$$

y para el signo, se toma:

$$\begin{aligned} signo^* &= sgn\big\{S_2(d^*_2) - S_1(d^*_1)\big\} \\ &= (-1)^{i^*+1}. \end{aligned}$$

Después de derivar las ecuaciones 5.1 y 5.2 e igualar a 0 ; y realizar algunos cálculos, se llega a que los valores de d^*_1 y d^*_2 se pueden obtener mediante:

$$d^*_1 = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{5}(-1)^j \begin{pmatrix} 5 \\ j \end{pmatrix} s[i][j]}{6m}, \tag{5.6}$$

$$d^*_2 = \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{5}(-1)^{j+\delta_B} \begin{pmatrix} 5 \\ j \end{pmatrix} s[i][j]}{6m}, \tag{5.7}$$

donde δ_B se define como:

$$\delta_B = \begin{cases} 1, & r \in B, \\ 0, & r \notin B. \end{cases} \tag{5.8}$$

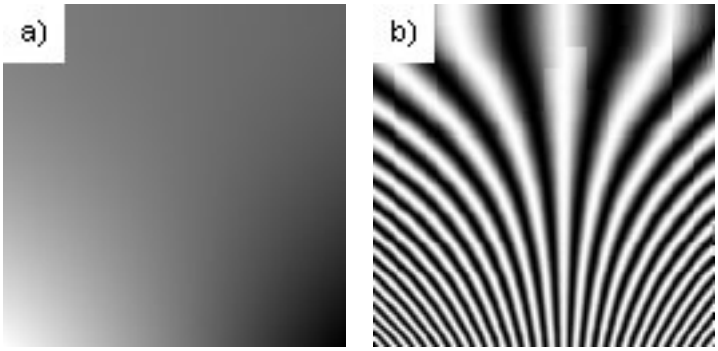


Figura 5.9. a) Fase después del acoplamiento, b) Coseno de la fase. Tiempo de cómputo: 0.751 segundos

con $r = (i, j)$, o más claramente

$$\delta_B = \begin{cases} 1, & j > 2, \\ 0, & j \leq 2. \end{cases} \tag{5.9}$$

La Figura 5.9 muestra el resultado del acoplamiento, tanto para la fase desenvuelta, como para el coseno de dicha fase. Observe que el resultado es bastante bueno, no obstante, se aprecian algunos detalles producto del acoplamiento, por lo que será necesario un postprocesamiento para mejorar el resultado, y de este modo obtener la *fase inicial estimada*. Para ello se empleará un proceso de suavizado, que consiste en aplicar una difusión homogénea a la imagen, por lo general pocas iteraciones, más adelante se verá que también es posible usar el refinamiento propuesto por Rivera en [8], ver Figura 5.10, con esto fue suficiente para eliminar las discontinuidades del acoplamiento. Es importante señalar que, en el acoplamiento, se emplearon otros potenciales similares a 5.3, que penalizaran las segundas derivadas o primeras derivadas, en lugar de penalizar las terceras derivadas; otra idea que se trabajó fue penalizar las primeras derivadas si los pixelones a acoplar tenían igual orientación y las terceras derivadas en caso contrario. Los experimentos realizados con las diferentes variantes mostraron resultados parecidos, por lo que de decidió usar la variante que se explicó en el presente trabajo.

En algunos experimentos que se muestran en las próximas secciones fue necesario realizar un preprocesamiento, sobre todo cuando las imágenes tienen ruido. El preprocesamiento, así como el postprocesamiento, consistió en suavizar la imagen

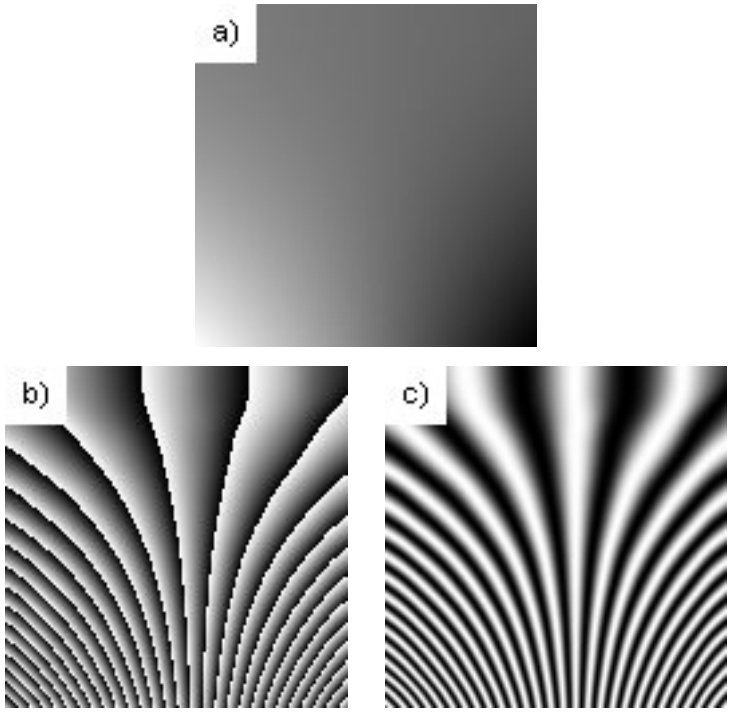


Figura 5.10. Resultados después del postprocesamiento 11 a) Fase, b) Fase envuelta, c) Coseno de la fase.

para lo cual se aplicó difusión homogénea.

5.1.1. Experimentos (Etapa A)

El ejemplo que se escogió para explicar la Etapa A fue un interferograma sintético con franjas abiertas, ahora se mostrarán algunos ejemplos para ver el comportamiento de esta etapa en imágenes con patrones de franjas sintéticas y reales que tengan franjas cerradas sin y con ruido.

Franjas cerradas sintéticas, sin ruido

En este ejemplo se verá el comportamiento de la Etapa A ante un interferograma sintético con franjas cerradas. La Figura 5.11 muestra los resultados hasta formar la *fase inicial*, aquí se puede ver que la fase que se obtiene está bastante bien, aunque tiene ligeros detalles, que mejoran al hacerle el postprocesamiento, Figura 5.12, con lo cual se tiene una *fase inicial estimada* con bastante calidad.

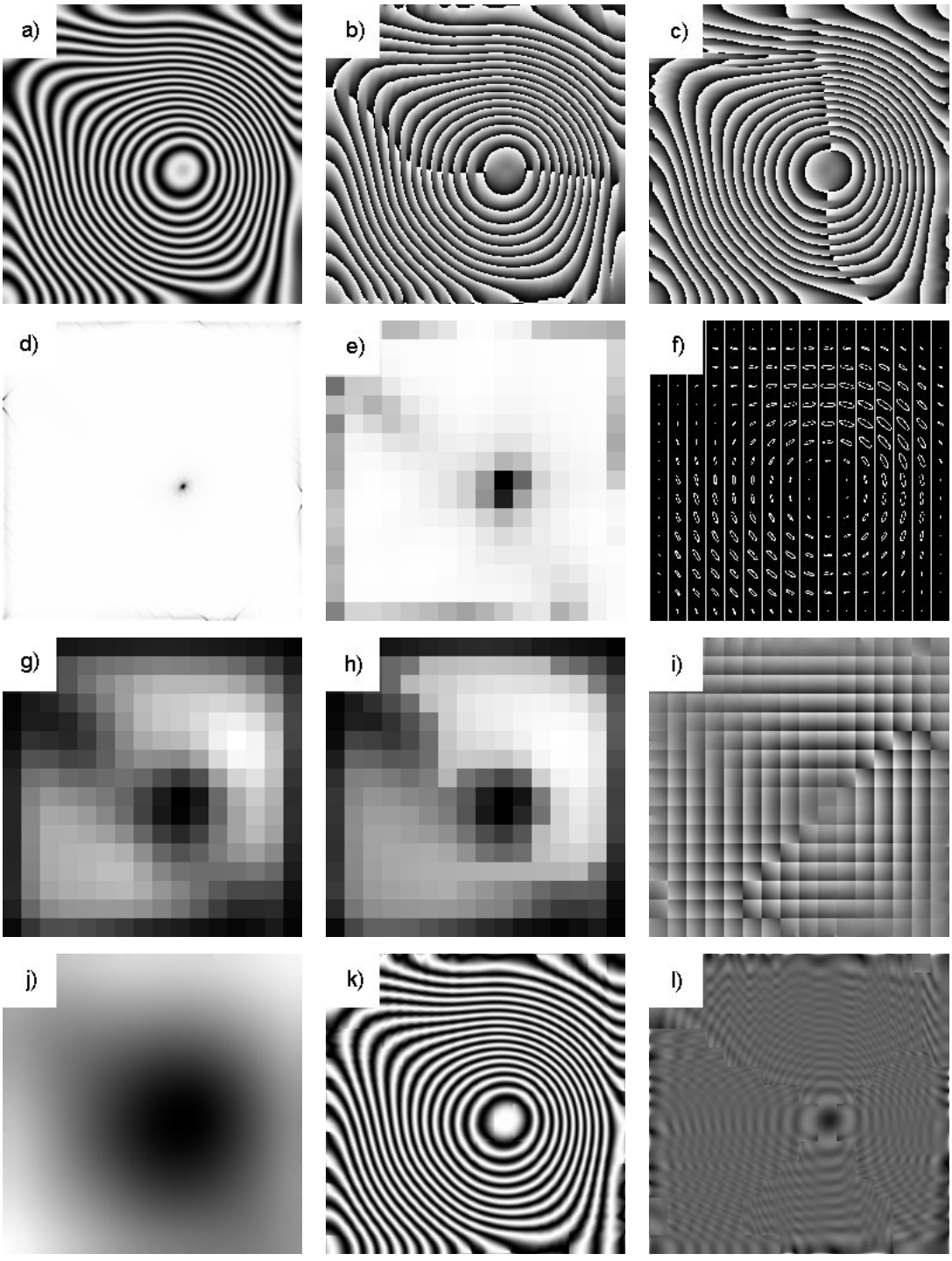


Figura 5.11. a) Imagen original, b) Fase al aplicar Takeda en la dirección horizontal, c) Fase al aplicar Takeda en la dirección vertical, d) Coherencia de la imagen original, e) Coherencia por pixelón, f) Mapa de orientación, g) Medida de bondad, h) Mapa de propagación, i) Imagen de fase desenvuelta por pixelón, j) Fase estimada, k) Coseno de la fase estimada, l) Magnitud estimada. Tiempo de cómputo: 3.592 segundos.

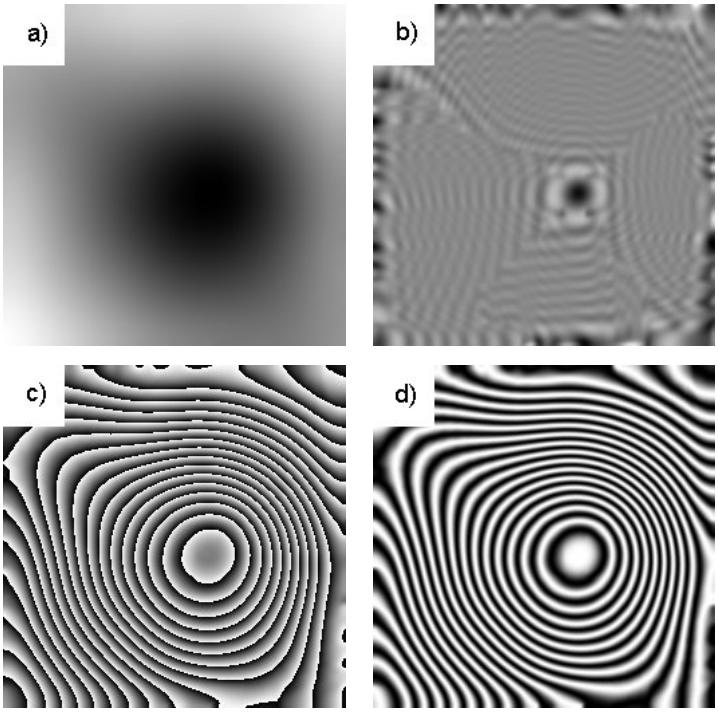


Figura 5.12. Resultados después del suavizado a) Fase , b) Magnitud, c) Fase envuelta, d) Coseno de la fase.

Franjas cerradas sintéticas, con ruido

Aquí se presenta el mismo interferograma del ejemplo anterior al que se le ha añadido ruido gaussiano. Como es conocido, el rango dinámico de un patrón de franjas normalizado está entre [-1,1], en este caso, la desviación estándar del ruido gaussiano aplicado fue de 0.2 y el rango dinámico para la imagen de ruido puro que se obtuvo fue de [-0.669985 , 0.7833369], lo que es bastante significativo, aún así se llega a buenos resultados. En la Figura 5.13 se describe todo el proceso. La Figura 5.13(k) muestra el coseno de la fase estimada, y se nota que todavía esta parte del ruido, aunque muy leve. Al comparar visualmente la Figura 5.13 con la Figura 5.11 del ejemplo anterior, se puede ver que las medidas y mapas por pixelón son muy similares, lo que nos dice que el camino que sigue el algoritmo es prácticamente el mismo. Sin embargo, no ocurre igual con los resultados que se obtienen de aplicar el método de Takeda en las dos direcciones, pues las imágenes de fase ahora se ven más ruidosas, y en caso de la coherencia, en el ejemplo anterior se puede decir que salvo en los bordes y un pequeño círculo central, el resto de la imagen está bien orientada; sin embargo al añadir el ruido gaussiano, la situación cambió totalmente, y la coherencia nos dice que la orientación es casi aleatoria de pixel a pixel. Después de aplicar el postprocesamiento los resultados mejoran bastante, Figura 5.14.

Franjas cerradas reales

En este punto se muestra un ejemplo real, es decir, se muestra un interferograma con patrones de franjas cerradas real. En la Figura 5.15 se pueden ver los resultados de estimación de la fase, aquí se nota un poco más el efecto del acoplamiento, Figura 5.15(k). Después del suavizado la fase mejora un poco, aunque sigue presentando algunos efectos del acoplamiento.

En muchos casos es suficiente aplicar la Etapa A del algoritmo y con ello se obtienen buenos resultados, no obstante, cuando el nivel de ruido es muy grande, entonces no es suficiente, por lo que se propone otra etapa del algoritmo para estos casos.

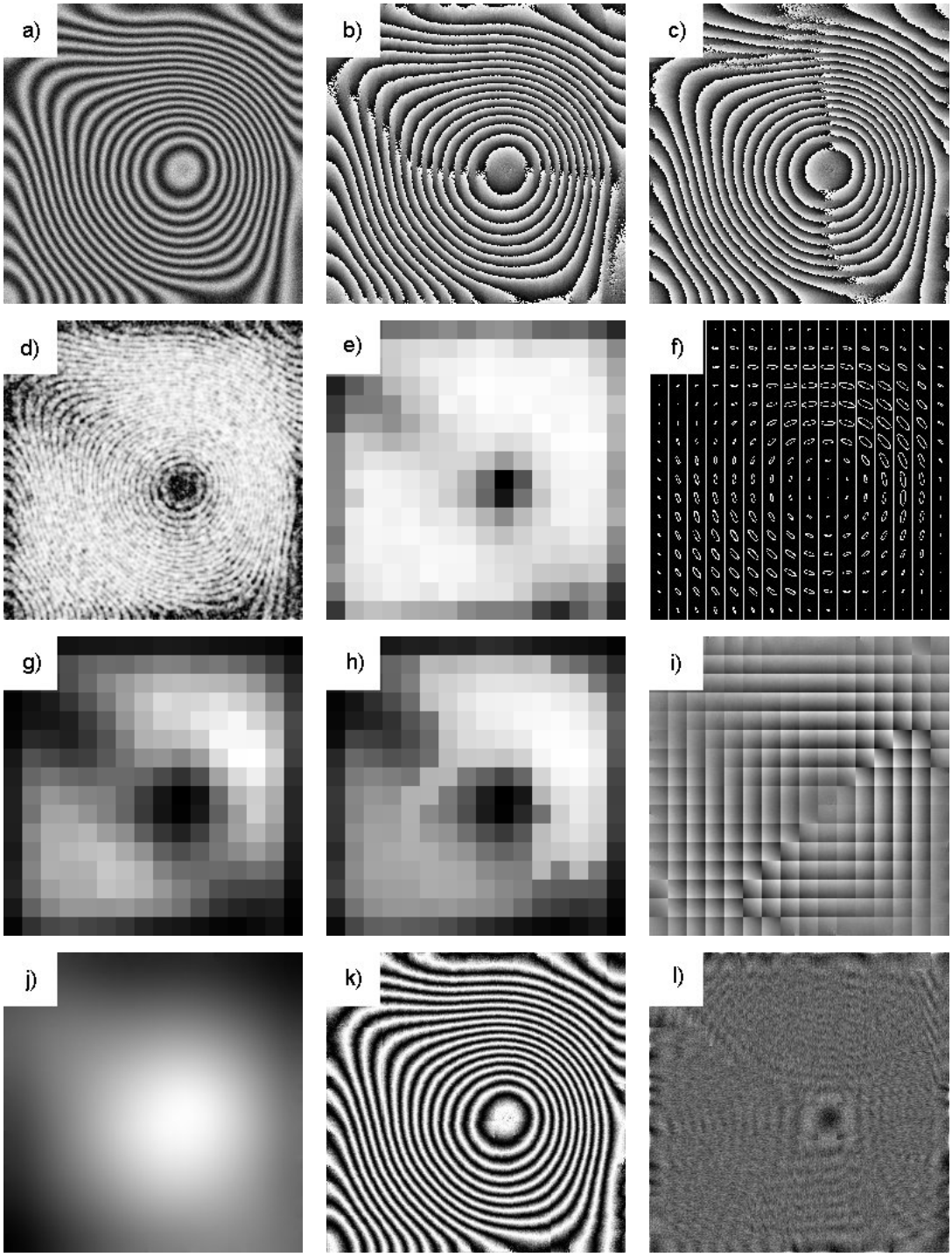


Figura 5.13. a) Imagen original (observaciones), b) Fase al aplicar Takeda en la dirección horizontal, c) Fase al aplicar Takeda en la dirección vertical, d) Coherencia de la imagen original, e) Coherencia por pixelón f) Mapa de orientación, g) Medida de bondad h) Mapa de propagación, i) Imagen de fase desenvuelta por pixelón, j) Fase estimada, k) Coseno de la fase estimada, l) Magnitud estimada. Tiempo de cómputo: 3.730 segundos.

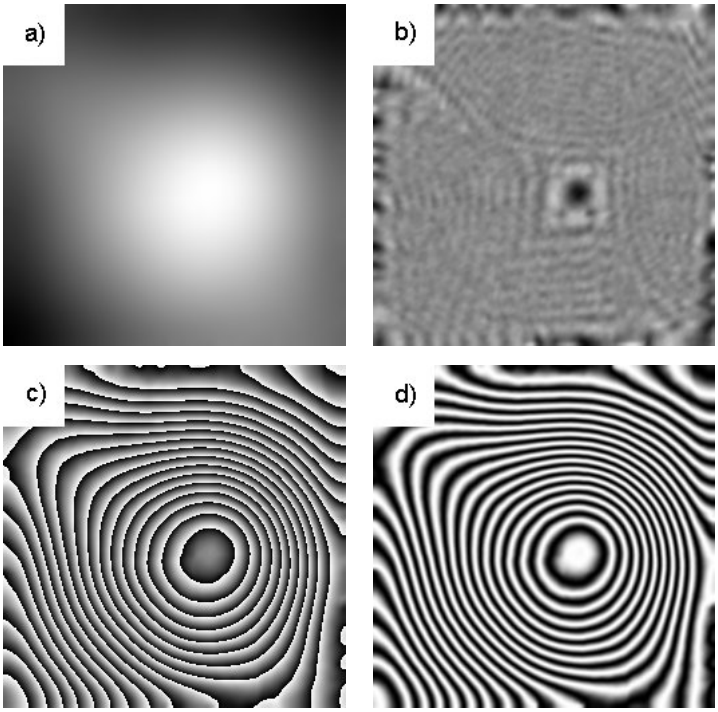


Figura 5.14. Resultados después del suavizado a) Fase b) Magnitud c) Fase envuelta d) Coseno de la fase desenvuelta.

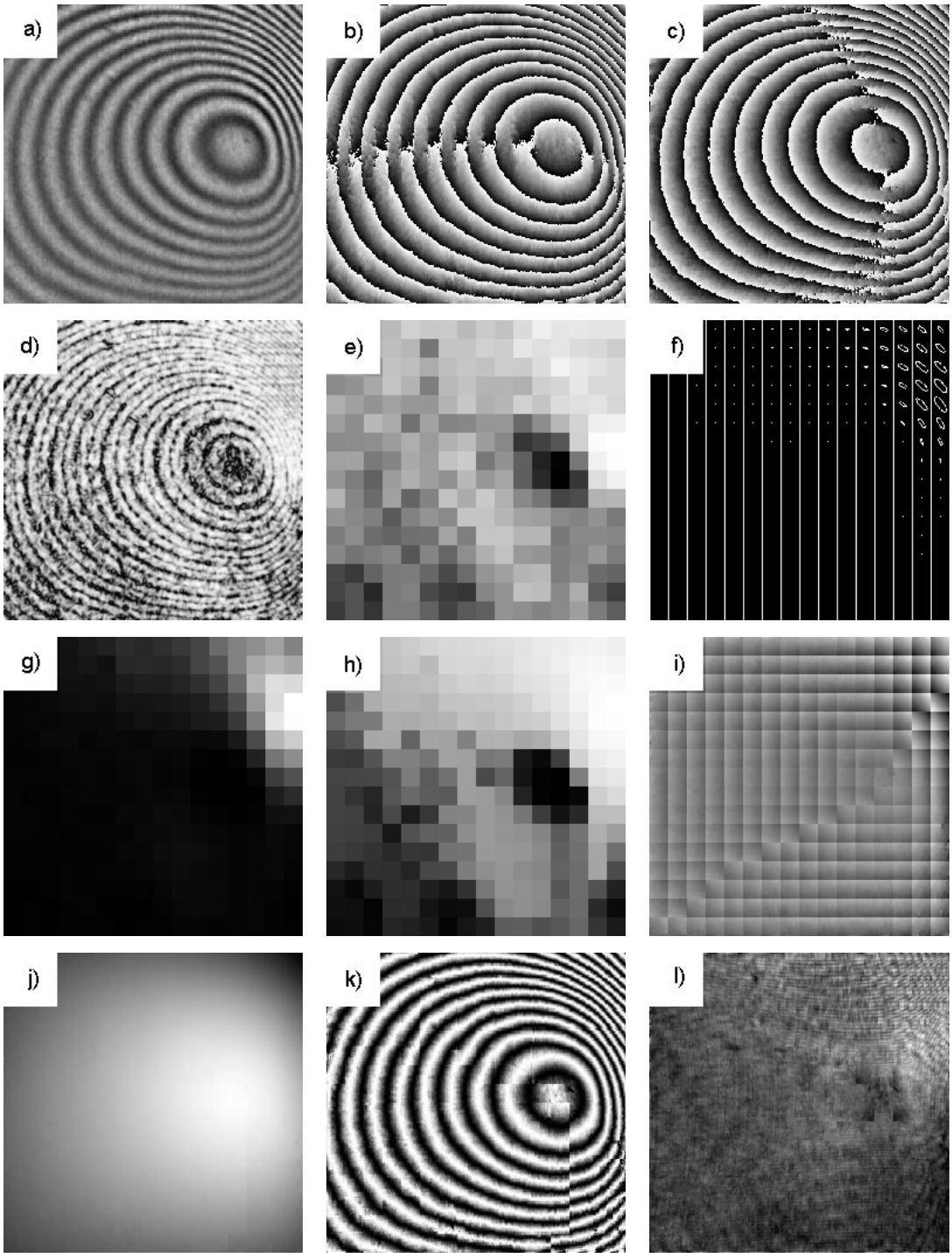


Figura 5.15. a) Imagen original, b) Fase al aplicar Takeda en la dirección horizontal, c) Fase al aplicar Takeda en la dirección vertical, d) Coherencia de la imagen original, e) Coherencia por pixelón, f) Mapa de orientación, g) Medida de bondad, h) Mapa de propagación, i) Imagen de fase desenvuelta por pixelón, j) Fase estimada, k) Coseno de la fase estimada, l) Magnitud estimada. Tiempo de cómputo: 3.531 segundos.

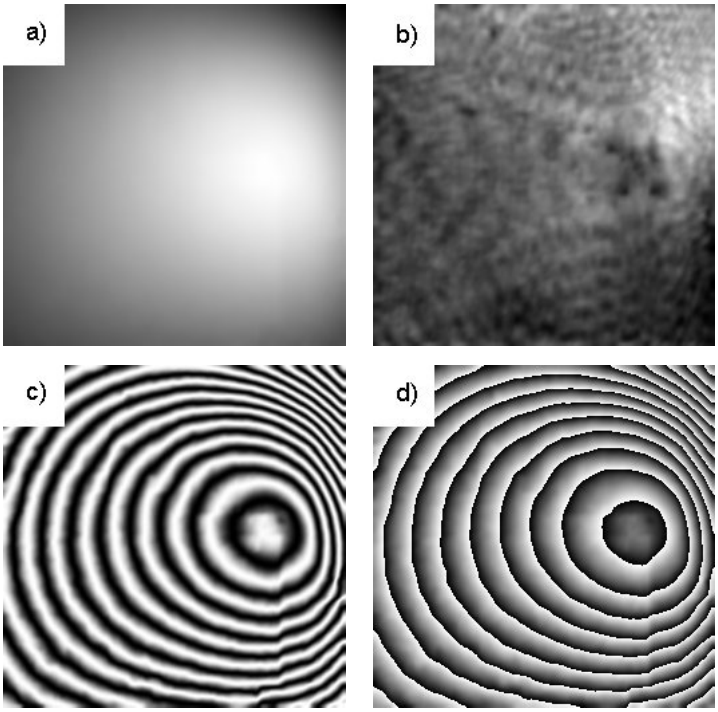


Figura 5.16. Resultados después del suavizado a) Fase, b) Magnitud, c) Fase envuelta, d) Coseno de la fase.

5.2. Algoritmo para la Etapa B

Con el objetivo de eliminar algunas discontinuidades producto del acoplamiento de los pixelones, al final de la Etapa A se realizaba un postprocesamiento; esto mejora los resultados, pero no justifica su calidad, por lo que se propone realizar un refinamiento de la fase, y para ello se usa el algoritmo de refinamiento propuesto por Rivera en [8] y que se explicó en capítulo 3.

Podemos diferenciar dos casos:

Caso # 1 Si el resultado que se obtiene a partir de la Etapa A es bastante bueno, entonces lo que se recomienda es realizar solamente un refinamiento, es decir, aplicar algunas iteraciones del algoritmo de refinamiento.

Caso # 2 En el caso en que el resultado no sea tan bueno, entonces lo que se propone es realizar un proceso basado en pirámides gaussianas que se obtienen a partir de suavizar y submuestrear la imagen, y en cada nivel realizar un proceso de refinamiento. Es aconsejable realizar este último proceso fundamentalmente en imágenes con mucho ruido

Antes de explicar el caso # 2, se aplicará el proceso de refinamiento a alguno de los ejemplos vistos en la sección anterior, lo cual nos servirá de referencia para ver cuando aplicar directamente el refinamiento a la *fase inicial estimada* o el proceso basado en pirámides.

En la Figura 5.17 se muestra el resultado que se obtiene de aplicar el refinamiento a la fase estimada del ejemplo de la Figura 5.11. Si se realiza una comparación entre la Figura 5.12, después del postprocesamiento, con el resultado que se tiene después del refinamiento se notarán muy pocas diferencias. Algo similar ocurre en el ejemplo de la Figura 5.13, es decir, al aplicar el proceso de refinamiento a la fase inicial estimada prácticamente se obtienen los mismos resultados, Figuras 5.14 y 5.18 después del suavizado y del refinamiento respectivamente. Sin embargo, si comparamos las Figuras 5.16 y 5.19 observaremos mayores cambios y una mejoría notable después del refinamiento.

Como se ha dicho, cuando el nivel de ruido es grande, hay altos contrastes o diferencias

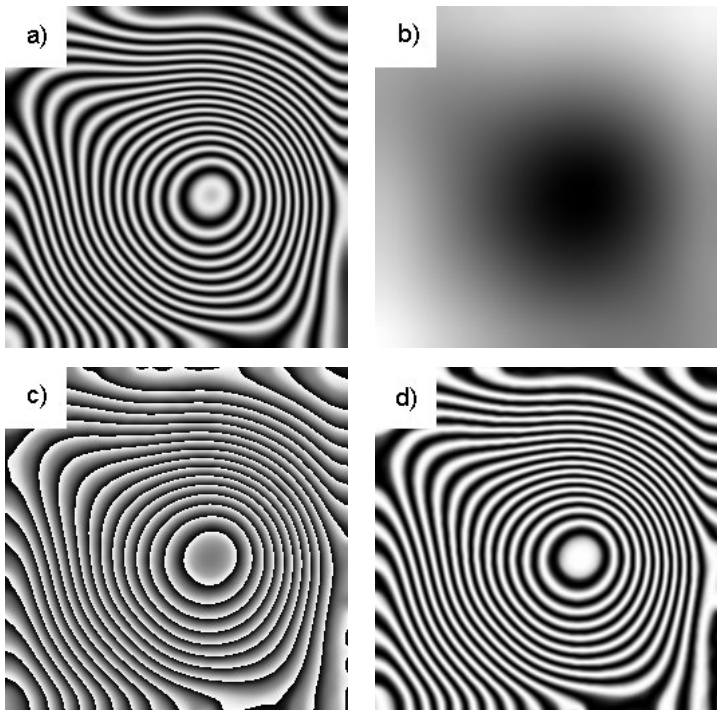


Figura 5.17. Resultados después del refinamiento a) Imagen original, b) Fase, c) Fase envuelta, d) Coseno de la fase. Tiempo de cómputo total: 3.992 segundos. Tiempo de cómputo de refinamiento: 0.313 segundos (4 iteraciones).

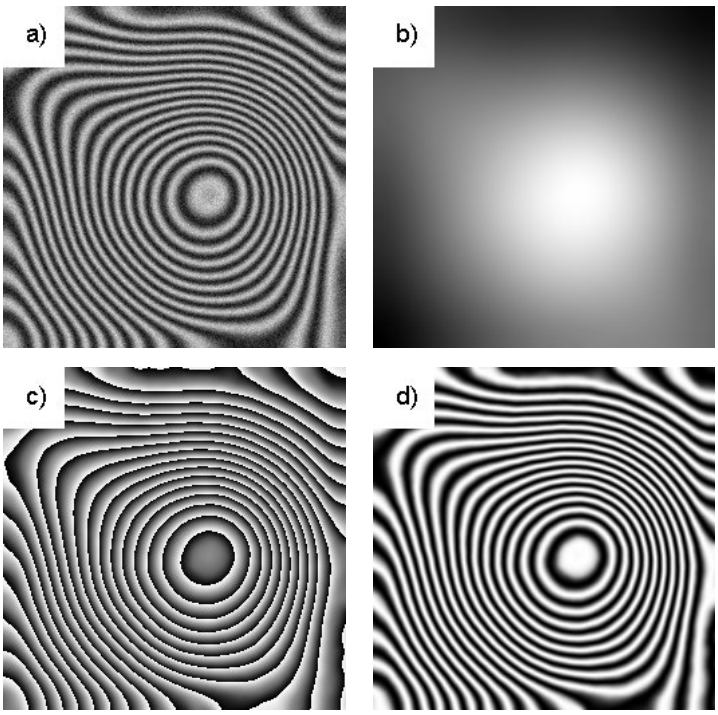


Figura 5.18. Resultados después del refinamiento a) Imagen original, b) Fase, c) Fase envuelta, d) Coseno de la fase. Tiempo de cómputo total: 4.059 segundos. Tiempo de cómputo de refinamiento: 0.313 segundos (4 iteraciones).

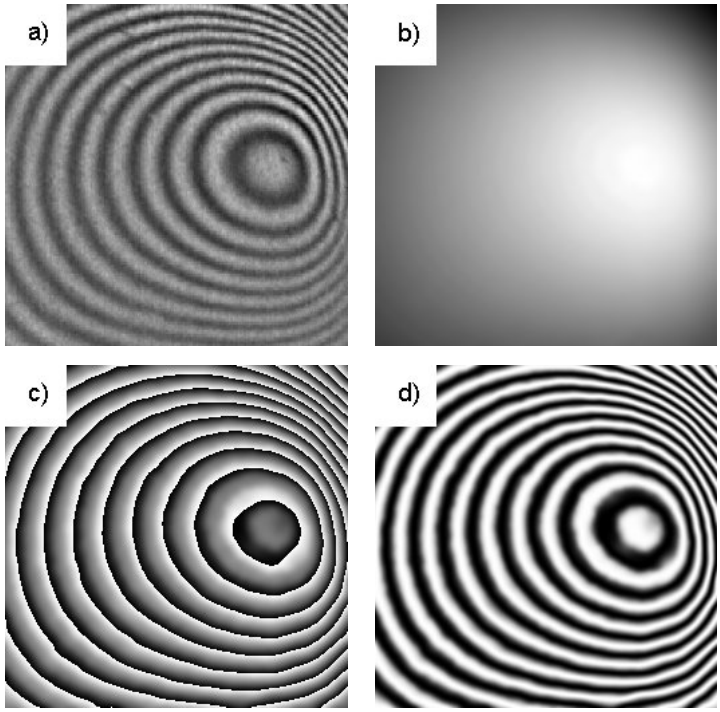


Figura 5.19. Resultados después del refinamiento a) Imagen original, b) Fase, c) Fase envuelta, d) Coseno de la fase.

de iluminación significativas, entonces los resultados no son tan buenos y se hace necesario perfeccionar la segunda etapa, por lo que se propone el siguiente algoritmo.

Sean los siguientes operadores: $\mathcal{R}_{\mu,\lambda}$ el operador de refinamiento propuesto por Rivera, $\mathcal{B}_2$ de suavizado y submuestreo o reducción, $\mathcal{E}_2$ de expansión e interpolación; donde el subíndice indica la razón de submuestreo o expansión. En el algoritmo que se muestra a continuación, los superíndices indican el nivel de la pirámide, y L es el laplaciano, el cual es usado en el proceso de reconstrucción.

Algoritmo (Etapa B)

Dada la fase estimada $\hat{\psi}^{(1)}$. (Nivel 1)

1. Suavizado y reducción. (Nivel 2)

$$\begin{aligned} \hat{\psi}^{(2)} &\leftarrow \mathcal{B}_2\hat{\psi}^{(1)}, \\ \mathrm{L}^{(1)} &\leftarrow \hat{\psi}^{(1)} - \mathcal{E}_2\hat{\psi}^{(2)}. \end{aligned}$$

2. Suavizado y reducción. (Nivel 3)

$$\begin{aligned} \hat{\psi}^{(3)} &\leftarrow \mathcal{B}_2\hat{\psi}^{(2)}, \\ \mathrm{L}^{(2)} &\leftarrow \hat{\psi}^{(2)} - \mathcal{E}_2\hat{\psi}^{(3)}. \end{aligned}$$

3. Refinamiento y reconstrucción (ó expansión e interpolación). (Nivel 2)

$$\tilde{\psi}^{(2)} \leftarrow \mathrm{L}^{(2)} + \mathcal{E}_2\mathcal{R}_{\mu,\lambda}\hat{\psi}^{(3)}.$$

4. Refinamiento y reconstrucción (ó expansión e interpolación). (Nivel 1)

$$\tilde{\psi}^{(1)} \leftarrow \mathrm{L}^{(1)} + \mathcal{E}_2\mathcal{R}_{\mu,\lambda}\tilde{\psi}^{(2)}.$$

5. Refinamiento .

$$\tilde{\psi} \leftarrow \mathcal{R}_{\mu,\lambda}\tilde{\psi}^{(1)}.$$

En la Figura 5.20 se muestra un esquema del algoritmo anterior.

Para el ejemplo el submuestreo se realizó tomando los pixeles de posición par en cada renglón, también se realizaron experimentos sin emplear pirámides gaussianas, es decir, usando solo submuestreos y los resultados fueron similares, el inconveniente de este último método es que no se garantiza que se cumpla el teorema del muestreo, ver [10] para más detalles. En el caso de la interpolación se usó interpolación bilineal, aunque se pudieran usar otras estrategias. La ventaja de la interpolación bilineal, para este caso, es su sencillez, de hecho se pueden dar fórmulas cerradas, ecuaciones 5.10 - 5.14, para hallar los valores intermedios. En la Figura 5.21 se supone que son conocidos los valores en los pixeles que están en negro y se quiere hallar los que están en blanco.

$$\psi(i,j+1) = \frac{\psi(i,j)+\psi(i,j+2)}{2}, \tag{5.10}$$

$$\psi(i+1,j) = \frac{\psi(i,j)+\psi(i+2,j)}{2}, \tag{5.11}$$

$$\psi(i+1,j+1) = \frac{\psi(i,j)+\psi(i+2,j)+\psi(i,j+2)+\psi(i+2,j+2)}{4}, \tag{5.12}$$

$$\psi(i+1,j+2) = \frac{\psi(i,j+2)+\psi(i+2,j+2)}{2}, \tag{5.13}$$

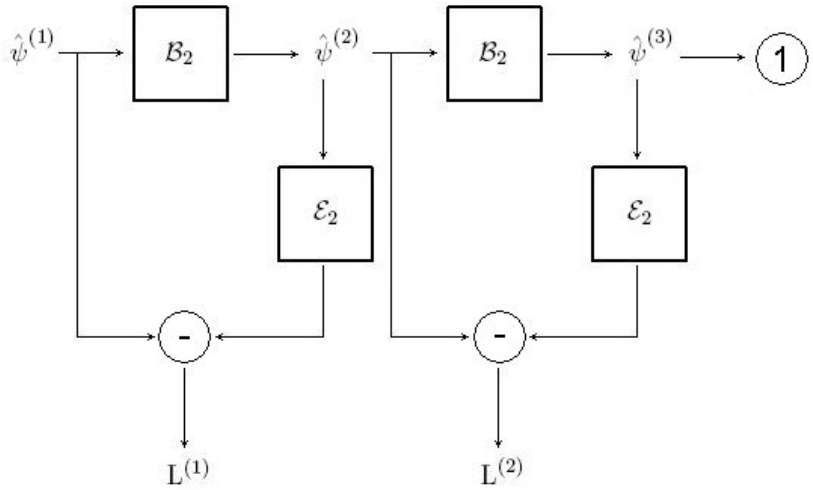
$$\psi(i+2,j+1) = \frac{\psi(i+2,j)+\psi(i+2,j+2)}{2}. \tag{5.14}$$

El operador de refinamiento depende de los parámetros μ y λ, en todos los casos se usaron los valores propuestos por el autor en su artículo, es decir: $\mu = 0.01$, $\lambda = 0.2$.

En la Figura 5.22 se muestra el proceso correspondiente a la Etapa B, aplicado al ejemplo de la Figura 5.15. Si se compara este resultado con los que se obtuvieron en las Figuras 5.16 después del suavizamiento y 5.19 después del refinamiento de podrá ver que se mejoró bastante.

5.2.1. Experimentos

En esta sección se pondrán otros ejemplos en interferogramas con franjas abiertas y cerradas. Se mostrará todo el proceso del algoritmo usando las dos etapas hasta obtener la fase aproximada. En cuanto a los parámetros del proceso de refinamiento,



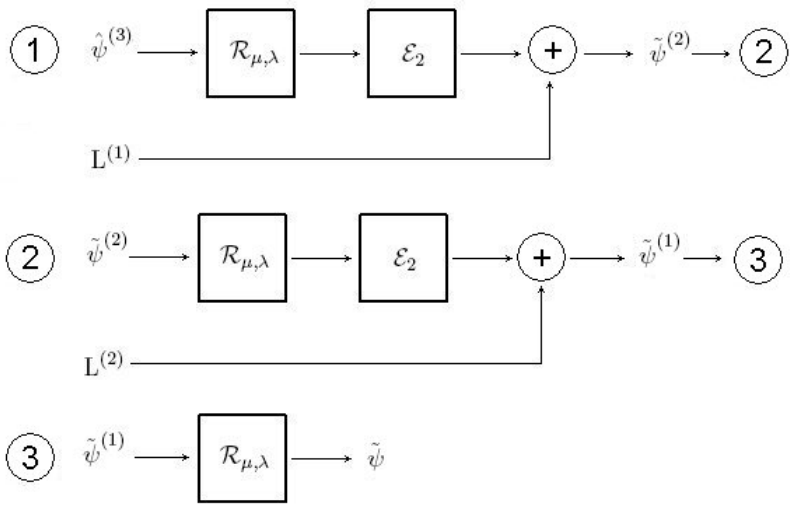


Figura 5.20. Esquema de la Etapa B

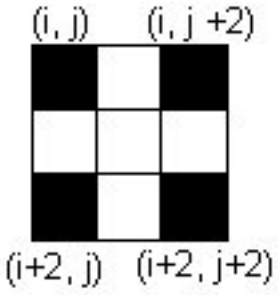


Figura 5.21. Interpolación

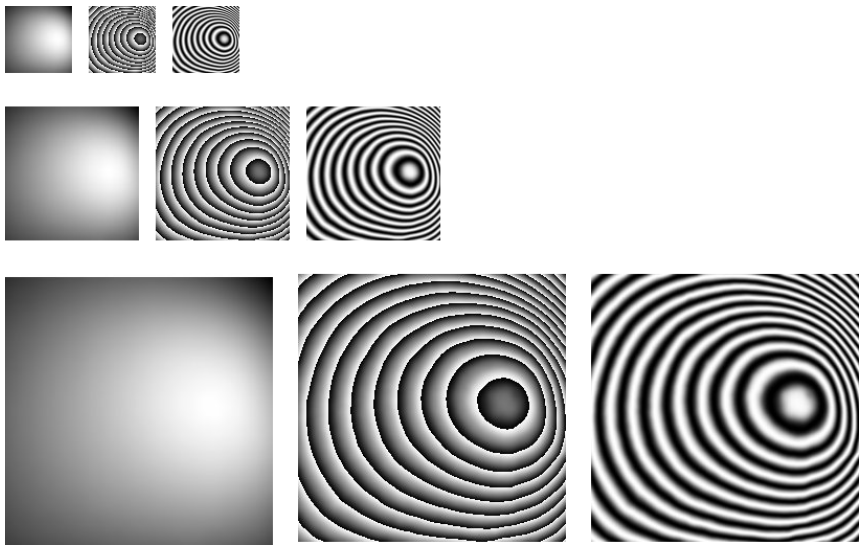


Figura 5.22. Etapa B, proceso de refinamiento usando pirámides gaussianas con 3 niveles. De arriba hacia abajo se muestra el proceso de refinamiento y reconstrucción del nivel 3 al 1 . De izquierda a derecha: Fase aproximada, fase envuelta y coseno de la fase.

ya se ha dicho que se están utilizando los propuestos por el autor, solo quedaría por definir el número de iteraciones en cada nivel de la pirámide, en este caso lo que se hace es fijar el número de iteraciones en 2 para los niveles intermedios; el número de iteraciones en el nivel 1 se deja como un parámetro y por defecto se tiene en 10 iteraciones. Hasta ahora no se ha dicho nada del tiempo de ejecución, para imágenes sintéticas con dimensiones de 128×128 el tiempo oscila entre 1 y 2 segundos, por supuesto esto dependerá también del número de iteraciones que se apliquen en el proceso de refinamiento del nivel 1, pero en los experimentos realizados se ha podido ver que cuando hay poco ruido son suficientes de 4 a 6 iteraciones en el nivel 1. En imágenes de 256×256 el tiempo total es de 10 segundos aproximadamente.

Interferograma sintético con franjas cerradas

En este ejemplo se tiene una imagen cuyas dimensiones son 128×128, ver Figuras 5.23, 5.24 y 5.25. Este es un interferograma con franjas cerradas y sin ruido, el resultado que se obtiene es muy bueno y el tiempo de duración de todo el proceso fue de 2 segundos aproximadamente.

Interferograma real con franjas abiertas

Aquí se muestra un interferograma real de 128×128, Figuras 5.26, 5.27 y 5.28. En esta imagen con franjas abiertas y con ruido, se dieron 4 iteraciones en el nivel 1 y el tiempo de duración fue de 2 segundos.

5.3. Experimentos

En todos los experimentos realizados hasta ahora se han usado los mismos parámetros, tanto en la etapa A como en la B. En esta sección se mostrarán otros ejemplos que son más complejos porque aumentan el número de franjas cerradas, aparecen bajas frecuencias o en algunos casos se le añade ruido gaussiano. Es por ello que el postprocesamiento de la etapa A y el refinamiento tendrán que ser más intensos. Para lograr buenos resultados se aumenta el número de iteraciones en el suavizamiento que se realiza antes del refinamiento, y en el propio proceso de refinamiento. Para las imágenes que tienen mucho ruido se propone realizar un preprocesamiento, que

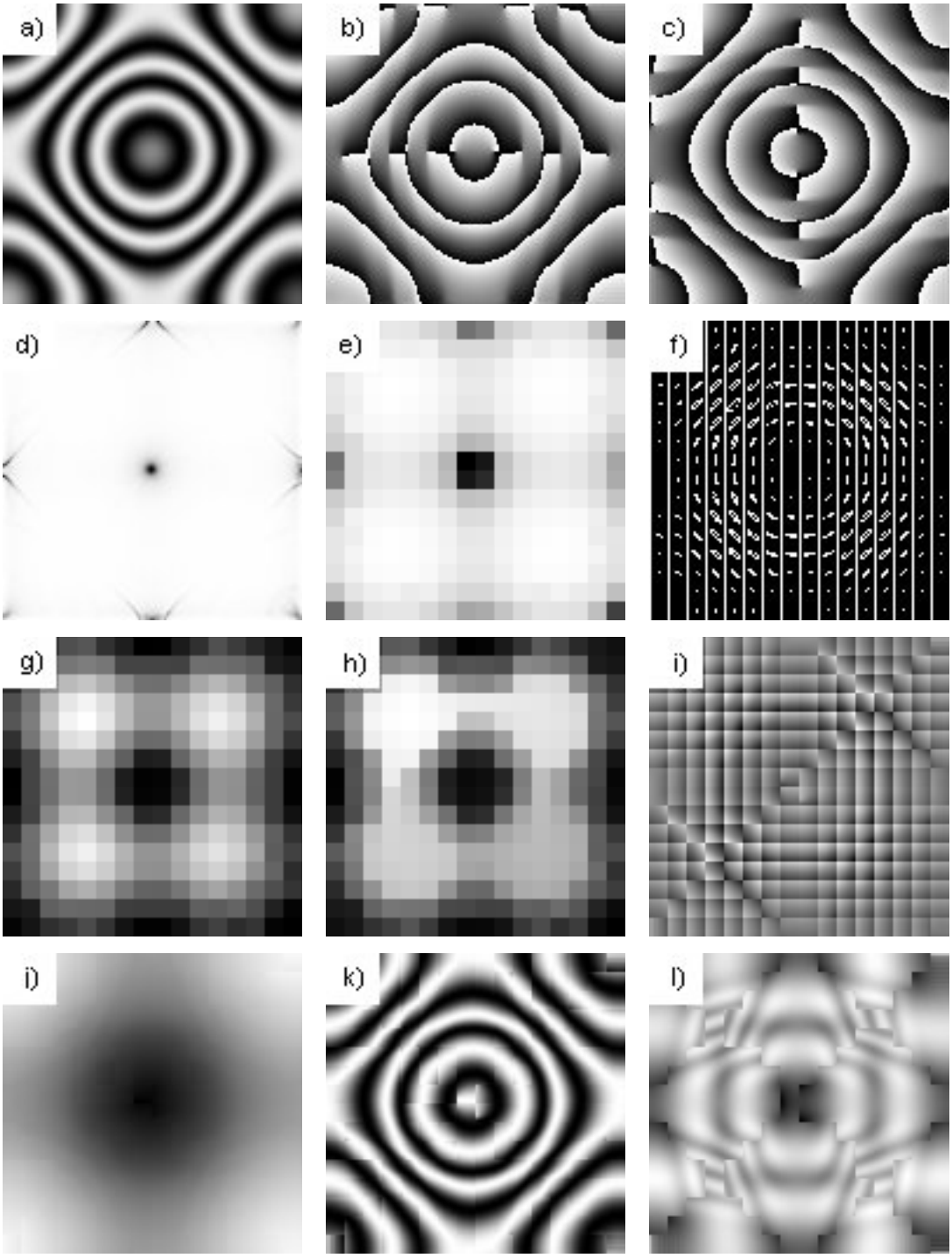


Figura 5.23. a) Imagen original, b) Fase al aplicar Takeda en la dirección horizontal, c) Fase al aplicar Takeda en la dirección vertical, d) Coherencia de la imagen original, e) Coherencia por pixelón, f) Mapa de orientación, g) Medida de bondad, h) Mapa de propagación, i) Imagen de fase desenvuelta por pixelón, j) Fase estimada, k) Coseno de la fase estimada, l) Magnitud estimada. Tiempo de cómputo: 0.719 segundos (Imagen de 128×128).

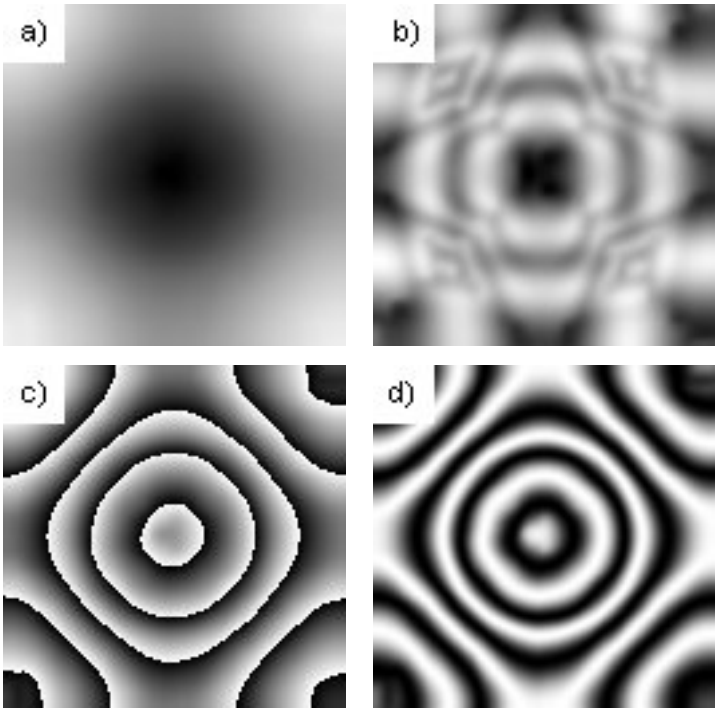


Figura 5.24. Resultados después del suavizado a) Fase desenvuelta b) Magnitud c) Fase envuelta d) Coseno de la fase desenvuelta

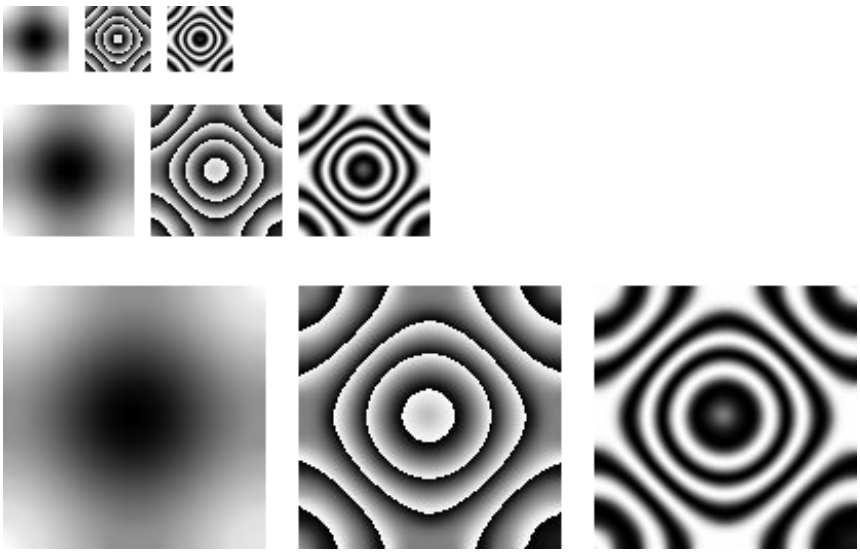


Figura 5.25. Etapa B, proceso de refinamiento usando pirámides gaussianas con 3 niveles. De arriba hacia abajo se muestra el proceso de refinamiento y reconstrucción del nivel 3 al 1 . De izquierda a derecha: Fase aproximada, fase envuelta y coseno de la fase. Tiempo de cómputo total: 1.375 segundos.

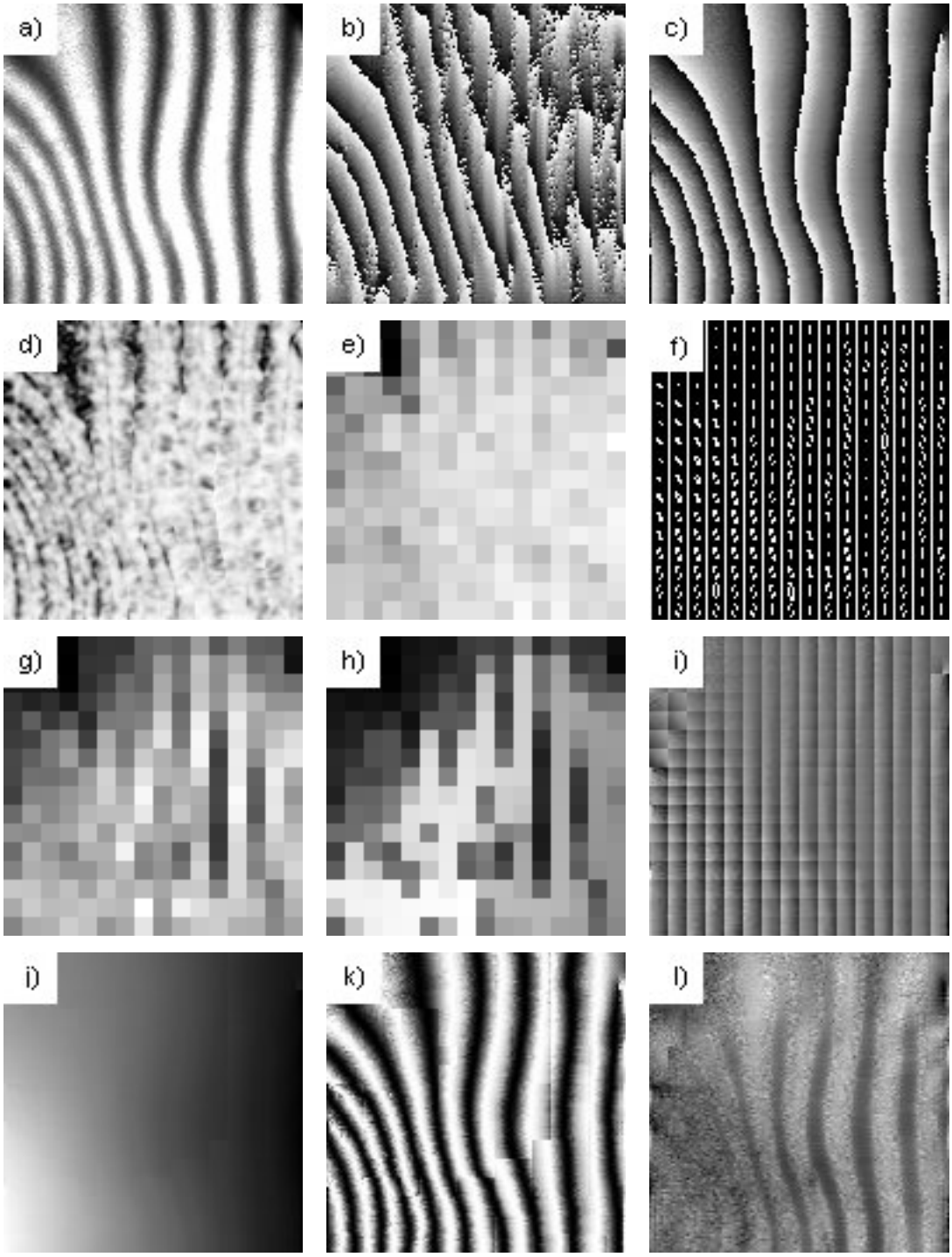


Figura 5.26. a) Imagen original, b) Fase al aplicar Takeda en la dirección horizontal, c) Fase al aplicar Takeda en la dirección vertical, d) Coherencia de la imagen original, e) Coherencia por pixelón, f) Mapa de orientación, g) Medida de bondad, h) Mapa de propagación, i) Imagen de fase desenvuelta por pixelón, j) Fase estimada, k) Coseno de la fase estimada, l) Magnitud estimada. Tiempo de cómputo: 0.775 segundos (Imagen de 128×128).

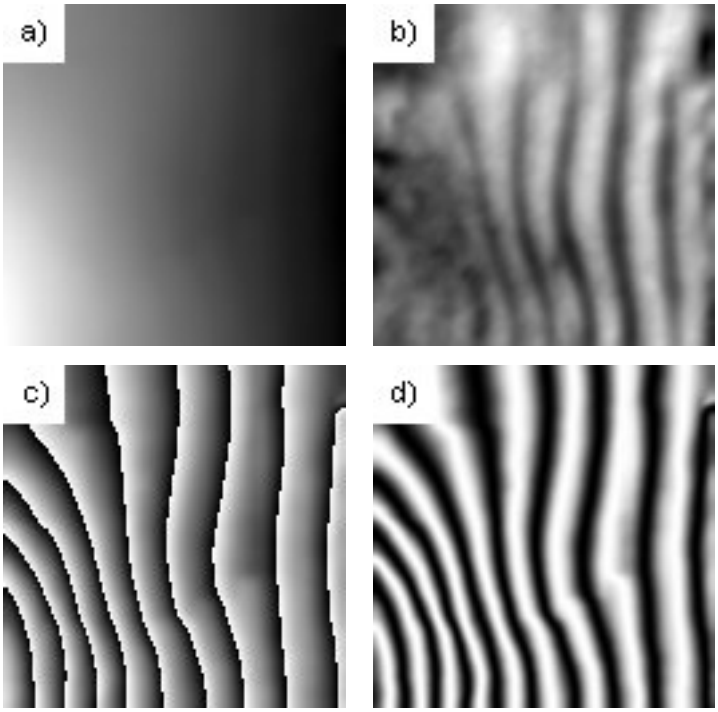


Figura 5.27. Resultados después del suavizado a) Fase, b) Magnitud, c) Fase envuelta, d) Coseno de la fase desenvuelta.

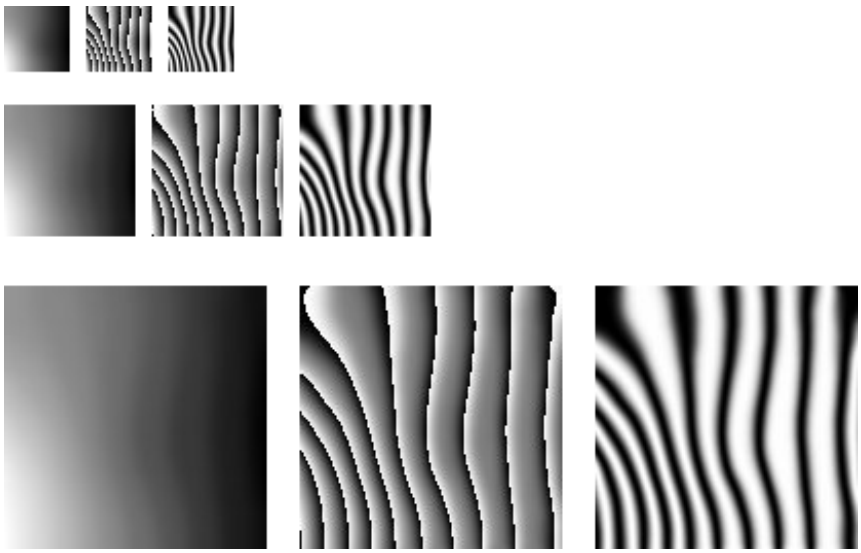


Figura 5.28. Etapa B, proceso de refinamiento usando pirámides gaussianas con 3 niveles. De arriba hacia abajo se muestra el proceso de refinamiento y reconstrucción del nivel 3 al 1 . De izquierda a derecha: Fase aproximada, fase envuelta y coseno de la fase. Tiempo de cómputo total: 1.422 segundos

consiste básicamente en realizar una difusión homogénea en toda la imagen.

Inicialmente se ilustrarán 3 experimentos para una misma imagen a la cual se le añade ruido blanco con diferentes varianzas. En estos casos solo se incluye el refinamiento, es decir, no se emplean las pirámides gaussianas. En el primero de ellos se tiene la imagen original, ver Figura 5.29, cuyas dimensiones son de 128×128. Los parámetros usados fueron:

- Iteraciones de suavizamiento: 8.
- Iteraciones de refinamiento: 4.

El tiempo de cómputo fue de 0.827 segundos.

La imagen del segundo experimento se obtuvo a partir de la imagen anterior después de añadir ruido blanco con 0,2 de desviación estándar, ver Figura 5.30. La imagen de ruido resultante tuvo un rango dinámico de $[-0.7220131, 0.7922179]$. Los parámetros usados fueron:

- Iteraciones de preprocesamiento: 2.
- Iteraciones de suavizamiento: 8
- Iteraciones de refinamiento: 4.

El tiempo de cómputo fue de 0.828 segundos. El preprocesamiento consistió en aplicar difusión homogénea a la imagen.

En el tercer ejemplo, ver Figura 5.31, la imagen se obtuvo a partir de la imagen del primer ejemplo, después de añadir ruido blanco con 0,4 de desviación estándar. La imagen de ruido resultante tuvo un rango dinámico entre $[-1.573998, 1.483773]$. Los parámetros usados fueron:

- Iteraciones de preprocesamiento: 2
- Iteraciones de suavizamiento: 8
- Iteraciones de refinamiento: 4

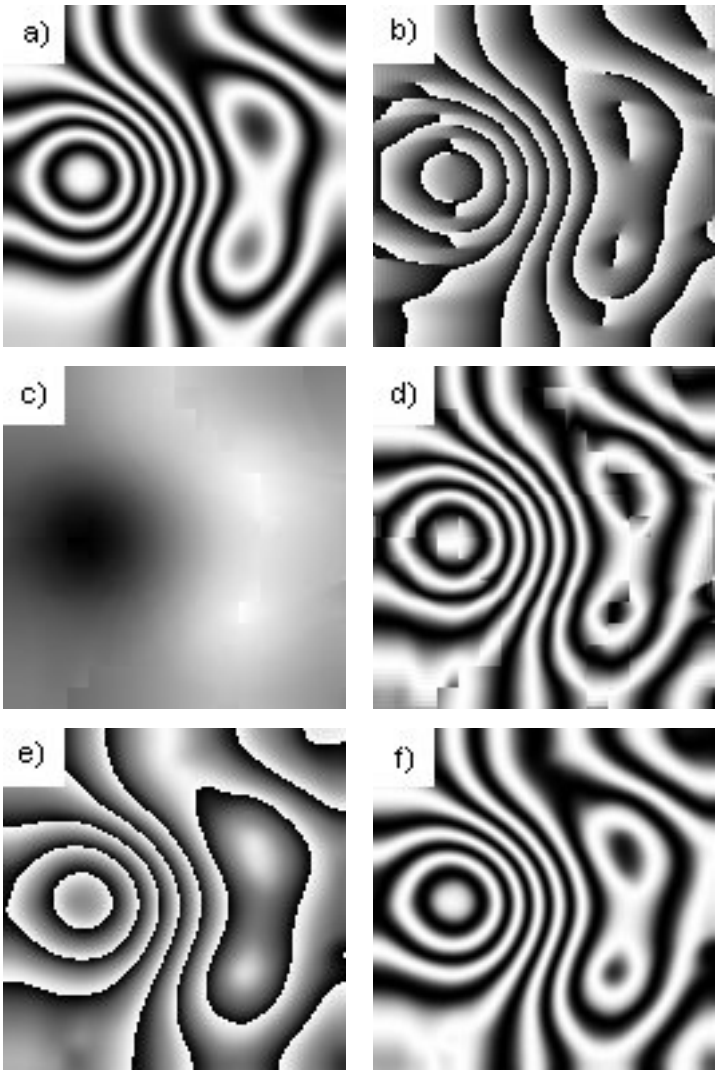


Figura 5.29. a) Patrón de franjas, b) Filtro vertical, c) Fase inicial, d) Patrón de franjas inicial e) Fase aproximada f) Patrón de franjas aproximado. Tiempo de cómputo: 0.827 segundos.

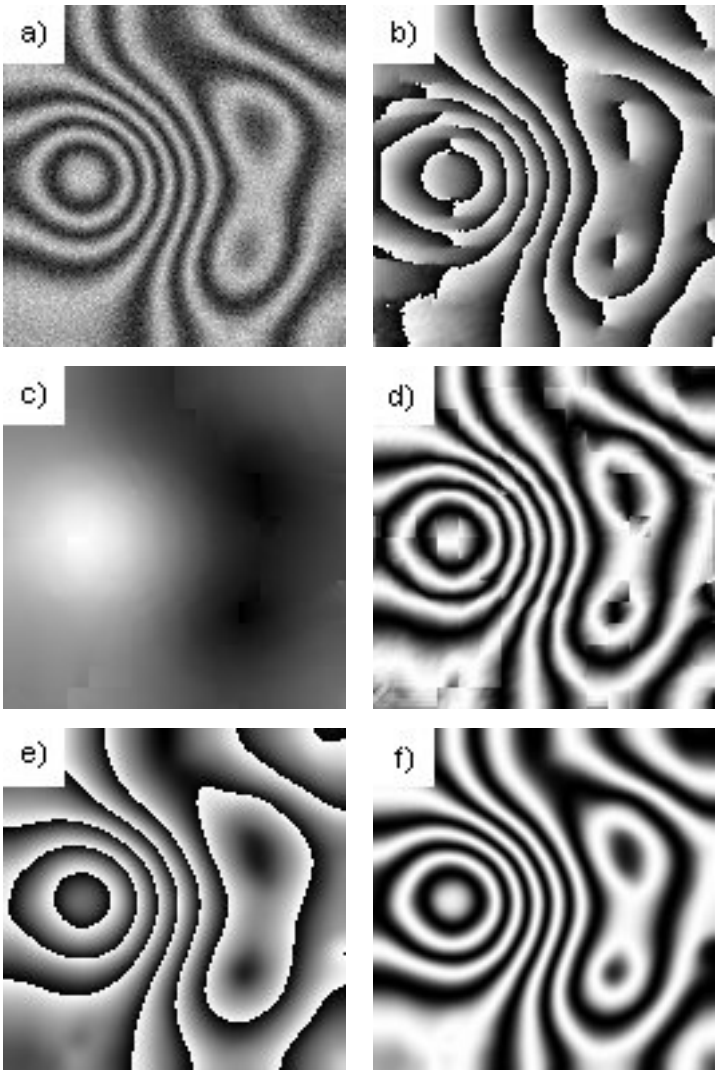


Figura 5.30. a) Patrón de franjas, b) Filtro vertical, c) Fase inicial, d) Patrón de franjas inicial e) Fase aproximada f) Patrón de franjas aproximado. Tiempo de cómputo: 0.828 segundos.

El tiempo de cómputo fue de 0.844 segundos.

En los 3 ejemplos que se mostraron, los resultados que se obtuvieron fueron satisfactorios, y el tiempo de cómputo promedio estuvo por debajo de 0.9 segundos.

En los próximos ejemplos se comenzarán a usar las pirámides en la Etapa B. Las dimensiones de la figuras son de 256×256.

La Figura 5.32 muestra una imagen sintética sin ruido. Los parámetros que se usaron fueron:

- Iteraciones de suavizamiento: 4
- Iteraciones de refinamiento: 10

El tiempo de cómputo fue de 10.469 segundos.

El ejemplo de la Figura 5.33 es muy similar al de la Figura 5.17, no obstante tiene una pequeña modificación. Los parámetros que se usaron fueron:

- Iteraciones de suavizamiento: 4
- Iteraciones de refinamiento: 4

El tiempo de cómputo fue de 6.594 segundos.

En la Figura 5.34, se muestra un ejemplo que es difícil para este algoritmo, se puede ver que la imagen de fase que se obtiene con el filtro vertical no es buena, aún así los resultados que se tienen son satisfactorios, no obstante se observa un ligero detalle que puede ser resuelto aumentado las iteraciones de refinamiento. Los parámetros que se usaron fueron:

- Iteraciones de suavizamiento: 4
- Iteraciones de refinamiento: 10

El tiempo de cómputo fue de 10.265 segundos.

En la Figura 5.35 se ilustra un ejemplo real. Los parámetros que se usaron fueron:

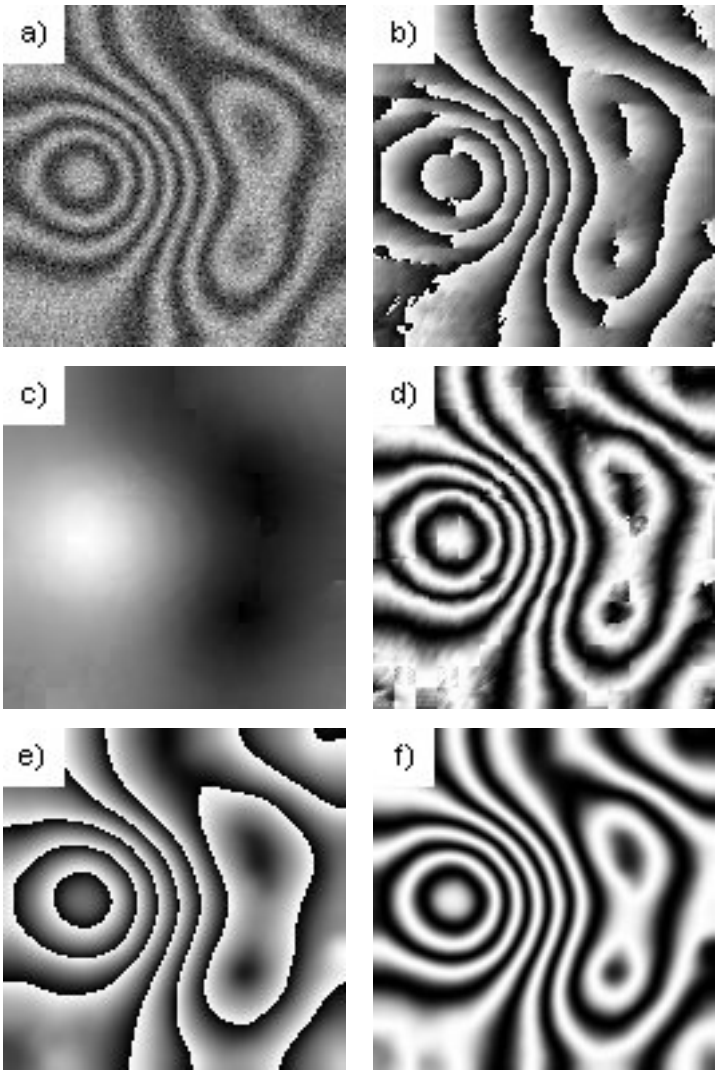


Figura 5.31. a) Patrón de franjas, b) Filtro vertical, c) Fase inicial, d) Patrón de franjas inicial e) Fase aproximada f) Patrón de franjas aproximado. Tiempo de cómputo: 0.844 segundos.

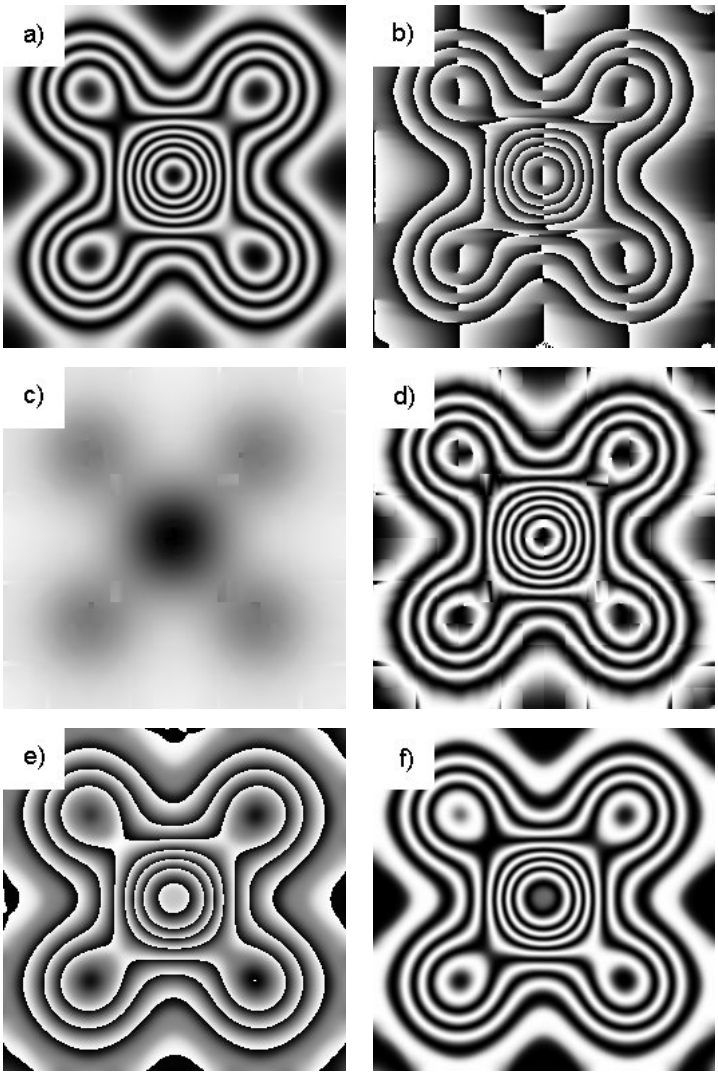


Figura 5.32. a) Patrón de franjas, b) Filtro vertical, c) Fase inicial, d) Patrón de franjas inicial e) Fase aproximada f) Patrón de franjas aproximado.

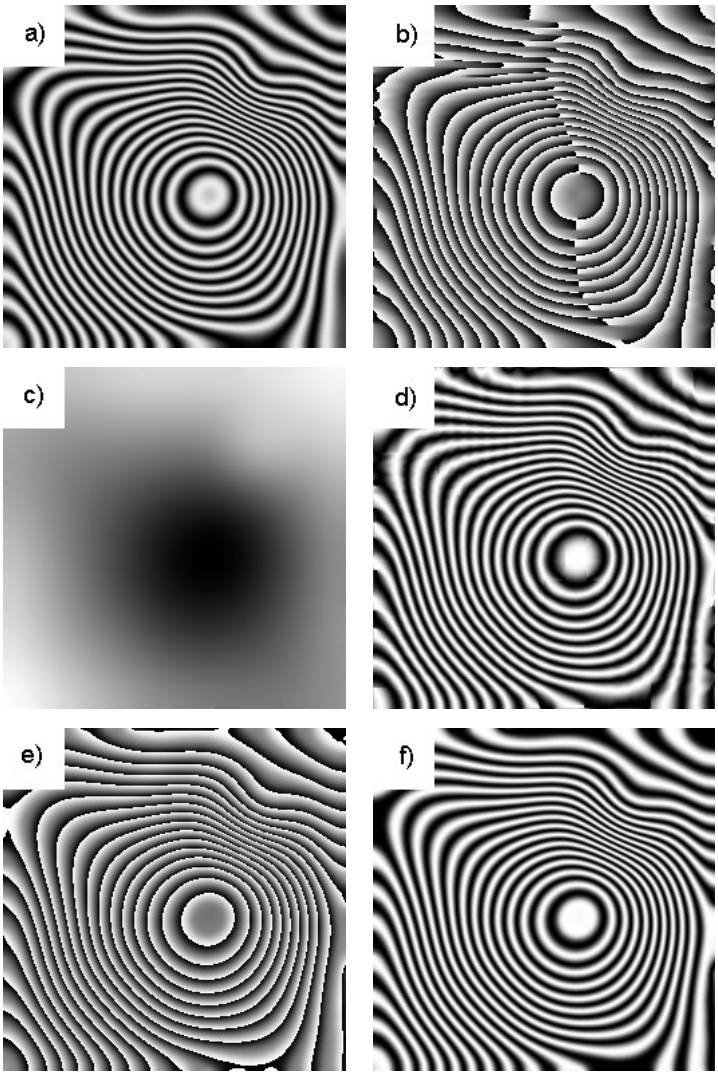


Figura 5.33. a) Patrón de franjas, b) Filtro vertical, c) Fase inicial, d) Patrón de franjas inicial e) Fase aproximada f) Patrón de franjas aproximado.

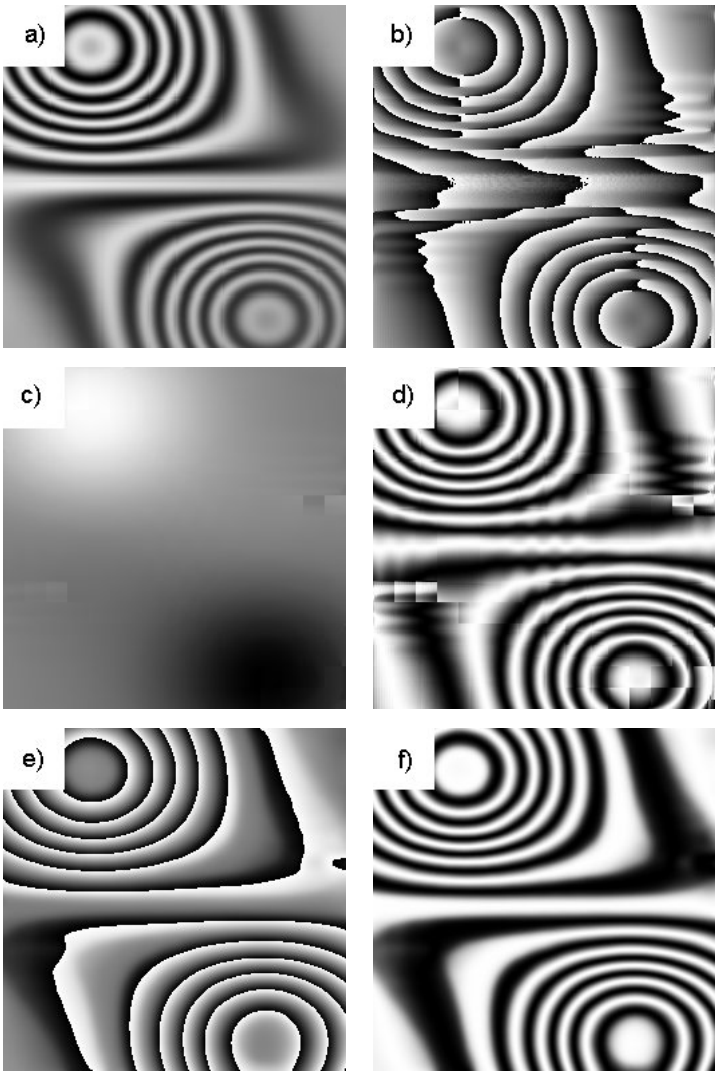


Figura 5.34. a) Patrón de franjas, b) Filtro vertical, c) Fase inicial, d) Patrón de franjas inicial e) Fase aproximada f) Patrón de franjas aproximado.

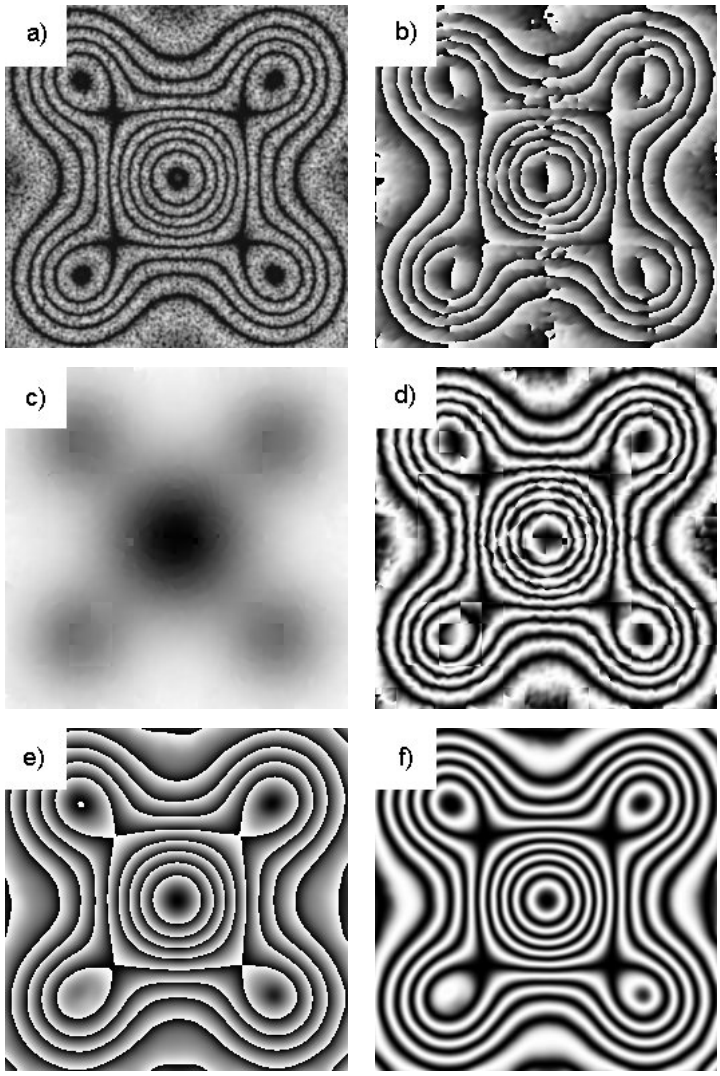


Figura 5.35. a) Patrón de franjas, b) Filtro vertical, c) Fase inicial, d) Patrón de franjas inicial e) Fase aproximada f) Patrón de franjas aproximado.

- Iteraciones de suavizamiento: 4
- Iteraciones de refinamiento: 10

El tiempo de cómputo fue de 10.672 segundos.

Finalmente se muestra un ejemplo más donde no funciona bien el algoritmo , ver Figura 5.36, esta es una imagen sintética con ruido multiplicativo. El problema fundamental se da en el centro de la imagen, y como se puede ver el acoplamiento en esta zona no es bueno, sin embargo en el resto de la imagen la fase estima es aceptable.

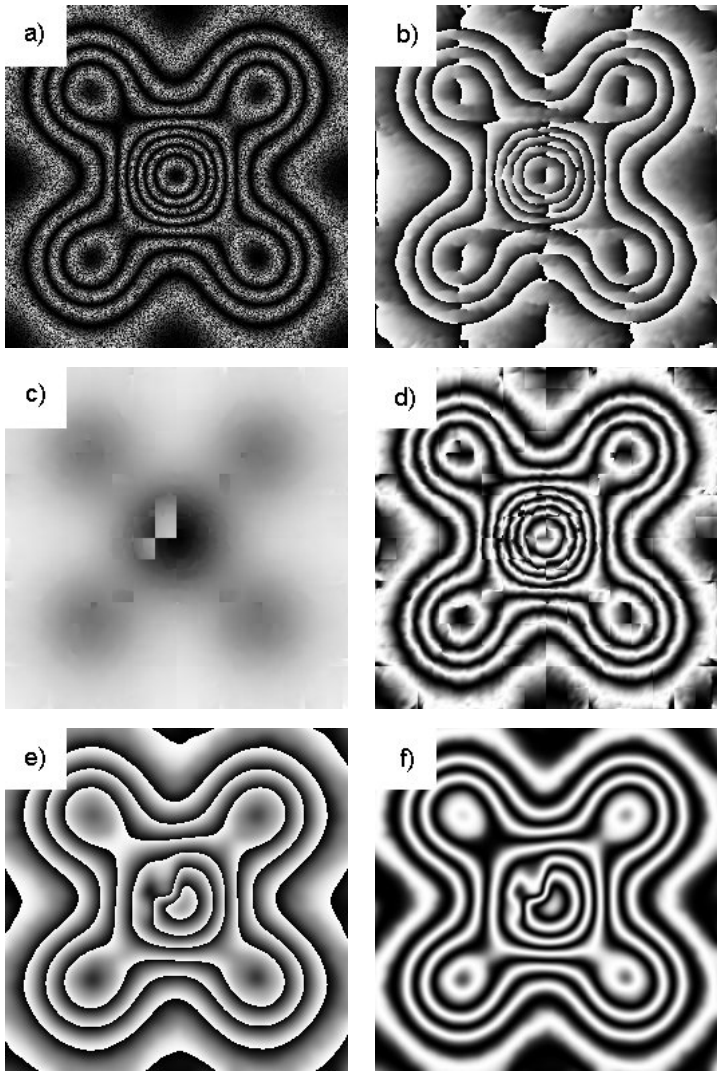


Figura 5.36. a) Patrón de franjas, b) Filtro vertical, c) Fase inicial, d) Patrón de franjas inicial e) Fase aproximada f) Patrón de franjas aproximado.

Capítulo 6

Conclusiones y trabajo futuro

En este trabajo se presentó un algoritmo para la recuperación de la fase de un solo patrón de franjas. Dentro de las características principales de este método están: no usa muchos recursos para estimar la fase y se aprovecha la información de orientación local presente en el interferograma, el mismo da bueno resultados tanto en imágenes con patrones de franjas abiertas, como con patrones de franjas cerradas, con niveles de ruido moderados y con gradientes suaves del contraste de iluminación. Es un algoritmo muy eficiente desde el punto de vista computacional lográndose tiempos, para la obtención de la fase aproximada, que mejoran considerablemente otros algoritmos del mismo tipo.

Entre los elementos a destacar de este trabajo están los siguientes:

- Se definió un nuevo elemento, el pixelón, que fue consistente con el algoritmo propuesto para estimar una fase inicial.
- El algoritmo propuesto tiene buen desempeño tanto en imágenes sintéticas como en imágenes reales con cierto nivel ruido
- Se logró verificar que es posible crear algoritmos eficientes de refinamiento, para mejorar la fase a partir de la fase inicial estimada en la etapa A.
- Otra de las ventajas del algoritmo es, la posibilidad de paralelizar la etapa A, puesto que el trabajo de desenvolvimiento es independiente en cada pixelón por lo que la imagen se puede dividir en diferentes zonas, en cada una de las cuales se aplicaría el algoritmo y entonces la fase se recuperaría a través de un proceso

de acoplamiento en cascada, esto permitiría obtener resultados casi en tiempo real, lo que podría ser muy útil en algunas aplicaciones.

La parte más sensible del algoritmo está en el paso del acoplamiento de los pixelones. Cuando hay mucho ruido o en zonas con frecuencias muy bajas el acoplamiento puede no ser bueno y entonces se pueden generar errores de acoplamiento que se propagarían a los pixelones vecinos. Una de las causas por las que este problema se genera, es por el modo en que se construye el mapa de propagación. En el algoritmo que se presenta en este trabajo se construye una sola región por lo que el mapa de propagación no expresa totalmente la medida de bondad de los pixelones. La propuesta para trabajo futuro está precisamente en resolver la situación anterior y entre las posibles soluciones están:

- Lograr que el mapa de propagación se ajuste al mapa de medida de bondad, creando tantas regiones conexas como sean necesarias de acuerdo a cierto umbral en la medida de bondad, de este modo estas regiones pueden crecer aumentando el umbral hasta que se logren conectar.
- Determinar un criterio para el acoplamiento de las regiones que se obtienen en el punto anterior.
- Otra idea que se puede incorporar, o que incluso, se podría hacer de forma independiente a lo anterior es realizar un proceso de propagación similar al RPT pero ahora usando los pixelones, de modo que se acoplen en la dirección de las franjas, y en cada paso realizar un proceso de refinamiento.

Bibliografía

[1] A. V. Oppenheim and J. S. Lim, "The importance of phase in signals," *Proceeding of the IEEE*, vol. 69, May 1981.

[2] D. C. Ghiglia and M. D. Pritt, *Two-Dimensional Phase Unwrapping*. New York, US: John Wiley & Sons, 1998.

[3] M. Takeda, H. Ina, and S. Kobayashi, "Fourier-transform method of fringe-pattern analysis for computer-based topography and interferometry," *Journal of the Optical Society of America*, vol. 72, pp. 156–, January 1982.

[4] M. Servin, J. Marroquin, and F. Cuevas, "Demodulation of a single interferogram by use of a two-dimesional regularized phase-tracking technique," *Appl. Opt.*, vol. 36, July 1997.

[5] ——, "Fringe-follower regularized phase tracker for demodulation of closed-fringe interferograms," *Appl. Opt.*, vol. 18, March 2001.

[6] M. Servin, J. Marroquin, and J. A. Quiroga, "Regularized quadrature and phase tracking from a single closed-fringe interferogram," *Journal of the Optical Society of America*, vol. 21, March 2004.

[7] ——, "Regularized quadrature and phase tracking (rqpt) from a single closed-fringe interferogram," Centro de Investigación en Matemáticas A.C (CIMAT), Guanajuato, México, Tech. Rep. I-03-13, Agosto 2003.

[8] M. Rivera, "Robust phase demodulation of interferograms with open and closed fringes," *Journal of the Optical Society of America*, vol. 22, pp. 1170–1175, June 2005.

[9] M. Rivera and J. L. Marroquín, "Half-quadratic cost functions for phase unwrapping," *Optics Letters*, vol. 29, March 2004.

[10] B. Jähne, *Digital Image Processing; Concepts, Algorithms, and Scientific Applications*, 2nd ed. Berlin: Springer-Verlag, 1991.

[11] W. Förstner and E. Gülch, "A fast operator for detection and precise location of distinct points, corners and centres of circular features," in *In ISPRS Intercommission Workshop*, Interlaken, June 1987, pp. 149–155.

[12] H. Knutsson, "A tensor representation of 3-d structure," in *5th IEEE-ASSP and EURASIP Workshop on Multidimensional Signal Processing*, Noordwijkerhout, The Netherlands, September 1987.

[13] M. Felsberg, "Low-level image processing with the structure multivector," Ph.D. dissertation, Institut fur Informatik und Praktische Mathematik der Christian Albrechts Universitat zu Kiel, Olshausenstr 40 D 24098 Kiel, March 2002.

[14] D. Malacara, *Optical Shop Testing*, 2nd ed., J. Goodman, Ed. New York, US: John Wiley & Sons, 1991.

[15] J. C. Estrada, "Patrones de franjas," Tesis de Maestría, Centro de Investigación en Matemáticas A.C, Guanajuato, México, Agosto 2003.

[16] M. Servin, D. Malacara, F. Cuevas, and J. Marroquin, "Complex linear filters for phase shifting with very low detuning sensitivity," *Journal of Modern Optics*, vol. 44, January 1997.

[17] J. Bruning, D. Herriot, J. Gallagher, D. Rosenfeld, A. White, and D. Brangaccio, "Digital wavefront measuring interferometer for testing surfaces and lenses," *Appl. Opt.*, vol. 13, pp. 2693–2703, 1974.

[18] J. Grievenkamp, "Generalized data reduction for heterodyne interferometry," *Opt. Eng.*, vol. 23, pp. 350–352, January 1984.

[19] K. A. Goldberg and J. Bokor, "Fourier-transform method of phase-shift determination," *Appl. Opt.*, vol. 17, June 2001.

[20] M. Rivera, R. Bizuet, A. Martínez, and J. A. Rayas, "Half-quadratic cost for computing arbitrary phase shifts and phase: Adaptive out of step phase shifting," *Optics Express*, vol. 14, pp. 3204–3213, April 2006.

[21] R. Legarda and M. Rivera, "Fast half-quadratic regularized phase tracking for non normalized fringe," *Accepted in Journal of the Optical Society of America*, 2006.

[22] D. Fried, "Least-squares fitting af a wave-front distortion estimate to an array of phase-difference measurements," *Journal of the Optical Society of America*, vol. 67, pp. 370–375, 1977.

[23] R. Hudgin, "Wave-front reconstruction for compensated imaging," *Journal of the Optical Society of America*, vol. 67, pp. 375–378, 1977.

[24] J. L. Marroquín and M. R. Meraz, "Quadratic regularization functionals for phase unwrapping," *Journal of the Optical Society of America*, vol. 12, pp. 2393–2400, 1995.

[25] M. J. Black and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statitics with applications in early vision," *Computer Vision*, March 1995.

[26] E. P. Simoncelli and H. Farid, "Steerable wedge filters for local orientation analysis," *IEEE Transactions on Image Processing*, September 1996.

[27] K. D. Weichan Yu and G. Sommer, "Approximate orientation steerability based on angular gaussians," *IEEE Transactions on Image Processing*, vol. 22, February 2001.

[28] X. Feng and P. Milanfar, "Multiscale principal components analysis for image local orientation estimation," *IEEE Transactions on Image Processing*, 2002.

[29] K. G. Larkin, "Uniform estimation of orientation using local and nonlocal 2-d energy operator," *Optics Espress*, September 2005.

[30] L. Haglund, H. Knutsson, and G. H. Granlund, "Scale and orientation adaptive filtering," in *Proceedings of the 8th Scandinavian Conference on Image Analysis.* Tromsö, Norway: NOBIM, May 1993, report LiTH–ISY–I–1527, Linköping University.

[31] H. Knutsson, "Representing local structure using tensors," in *In The 6th Scandinavian Conference in Image Analysis*, Oulu, Finland, June 1989, pp. 244–251.

[32] G. Strang, *Algebra Lineal y sus Aplicaciones*, 2nd ed. Wilminton Delaware, U.S.A: Addison-Wesley, 1986.

[33] T. A. I. Stuke and C. M. E. Barth, "Analysing superimposed oriented patterns," *6th IEEE Southwest on Image Analysis and Interpretation*, March 2004.

[34] M. Frigo and S. G. Johnson, "The design and implementation of fftw3," in *Proceedings of the IEEE*, vol. 93, no. 2, 2005, invited paper, pp. 149–155.