A mi madre

A Aliatar

A Darian

—No ..., no veo a nadie —dijo Alicia.

—Lo que yo daría por tener unos ojos así —dijo el Rey con un tono de tristeza—. ¡Eso es maravilloso! ¡Ver a Nadie! ¡Y a esa distancia además! ... Tengo que contentarme con ver a las personas reales a esa distancia. ¿No le parece?.

Lewis Carroll
A través del Espejo

— Es que a lo mejor no se trata de encontrar una moraleja ... —se aventuró a observar Alicia.

— Calla, criatura —dijo la Duquesa—. ¡Qué corta es tu experiencia! ¡Todas las cosas tienen su moraleja!

Lewis Carroll.
Alicia en el país de las maravillas.

¡Si no tiene nada que hacer, no lo venga a hacer aquí!

Dicho popular

Deseo agradecer a las diferentes personas que de una u otra forma hicieron posible el desarrollo de este trabajo. A mi familia: mi Madre, mis hermanos Leonardo, Ariadna y Aliata y mis sobrinos Aliatar y Darian por haberme apoyado y brindarme la tranquilidad de saber que están ahí.

A mis compañeros de clase por su amistad. A Sergio Limones y Alfonso Valdez por el ameno cohabitar estos dos años. A Isidro Cornejo y Jaime Almaguer por esa forma que tienen de ver la vida. A José Luis Gómez y Esteban Zarate por tener el comentario preciso en el momento necesario. A Ana Rosa Boyain y Goitia y Martín Ortiz por compartir su tesón. A Luis Roberto Sahagún por ser como es. A Arturo Olivares por su sinceridad. A Luis A. Domínguez por su magia. A Noé Alcalá por su sagacidad. A José Luis Juárez por su camaradería y el haber compartido su conocimiento. En particular a Eunice V. Jongitud por haber llegado en el momento justo.

A mis maestros por el haberme permitido conocer un poco más mi ignorancia. En particular a mi director de tesis, el Dr. Daniel Malacara, por compartir sus conocimientos y haber sugerido el tema desarrollado en este trabajo. A mis sinodales: el M.en C. Abundio Dávila por su amistad y consejo en lo referente a computación sin lo cual este trabajo no hubiera sido posible, y al Ing ESO C. Salvador Cuevas por haberme iniciado en el camino de la óptica y las sugerencias hechas al trabajo.

Muy en especial al CONACyT por el apoyo económico durante el desarrollo de la Maestría en Ciencia (Optica), sin el cual no hubiera conservado los pocos kilos que me quedan ni haber sostenido mi más preciado vicio.

Deseo extender mi más profundo agradecimiento al Centro de Investigaciones en Optica A.C. y a todas las personas con buen corazón que en el laborar por haberme permitido compartir con ellos dos años de sus vidas, en los cuales con la diaria convivencia me permitieron crecer como persona.

INDICE

INTRODUCCION

La interferometría tiene como principal objetivo la medición de algún parámetro físico mediante la aplicación del fenómeno de interferencia. Sin duda una de las principales aplicaciones es la prueba de la calidad de los sistemas y superficies ópticas. Estas pruebas de calidad en sistemas ópticos vienen haciendoce desde hace casi cien años, pero la precisión requerida ha ido en continuo aumento conforme la ciencia y la tecnología avanzan. Por otro lado, las computadoras, los láseres, y la nueva instrumentación electrónica han permitido este avance.

Hace tan sólo diez años que la nueva forma de hacer las pruebas era mediante una nueva técnica de interferometría heterodina llamada de desplazamiento de fase. Esta técnica ha tenido un gran auge y éxitos notables. Desgraciadamente, esta técnica de desplazamiento de fase también ha tenido serias limitaciones. Una de ellas ha sido su gran sensibilidad a vibraciones del sistema. Recientemente se han propuesto alternativas a este método. Una de éstas es la llamada interferometría de Fourier, que es también una técnica de interferometría heterodina.

Esta técnica basada en el algoritmo de la transformada rápida de Fourier, permite conocer sin ambigüedades la forma del frente de onda asociado a un interferograma debido a la codificación espacial de múltiples interferogramas en uno, además de que el interferograma que se analiza es estático, lo que representa una ventaja sobre las técnicas de desplazamiento de fase que requieren alta estabilidad y de dispositivos para correr la fase al hacer las pruebas. Esto hace que esta técnica de análisis de interferograma sea económica.

La idea básica de la interferometría de Fourier es introducir una inclinación entre los frentes de onda que se hacen interferir lo suficientemente grande para que las franjas del interferograma sean abiertas. El interferograma se digitaliza después por medio de una cámara de televisión conectada a una computadora. Se obtiene la transformada de Fourier del interferograma en el que se tienen tres órdenes de difracción, donde la información del frente de onda está en el orden uno. El siguiente paso es filtrar la transformada a manera de que quede sólo el orden uno de difracción, el cual se centra en el campo de la transformada. Se calcula la transformada de Fourier inversa del resultado del filtraje y en la fase de la ésta está el frente de onda que se quiere obtener.

En el presente trabajo se elaboraron las herramientas para desarrollar este tipo de análisis de interferogramas usando técnicas de procesamiento digital de imágenes, en particular los algoritmos para filtrar el orden uno de difracción y realizar el desdoblamiento de fase, que junto con el algoritmo de la transformada rápida de Fourier constituyen el fundamento del método. Este análisis se desarrolla a partir de un interferograma digitalizado en forma independiente de una tarjeta digitalizadora lo que le permite ser usado en cualquier tipo de computadora, con la única limitante de que tenga una tarjeta de video VGA.

Finalmente, se desarrolló un método para calcular los coeficientes de Zernike asociados al frente de onda que se obtiene al analizar un interferograma, a partir de éste y del polinomio de Zernike correspondiente sin necesidad de realizar interpolaciones. Este es el resultado más importante del presente trabajo, el cual es una opción diferente a los métodos actuales de realizar este cálculo.

CAPITULO 1

CONCEPTOS BASICOS DE LA INTERFEROMETRIA DE FOURIER

1.1. FUNDAMENTOS DE INTERFEROMETRIA

La interferometría tiene como principal objetivo la medición de algún parámetro físico mediante la aplicación del fenómeno de interferencia entre dos o más ondas, donde al menos una contiene información del parámetro que se desea conocer. La principal aplicación de ésta es en la prueba de la calidad de superficies y sistemas ópticos, donde se pueden medir deformaciones hasta de $\lambda/100$ ($\approx$50 Å).

Existen una gran diversidad de instrumentos con los que se puede producir interferencia entre dos o más ondas, llamados **interferómetros**. Podemos encontrar un buen número de trabajos publicados acerca de estos instrumentos (Malacara 1978, Françon 1966, Steel 1983).

Los fenómenos que provienen de la interferencia óptica son descritos en términos de la teoría ondulatoria debido a la naturaleza electromagnética de la luz, regida por las ecuaciones de Maxwell. Estas ondas obedecen al principio de superposición, por lo tanto, la inestabilidad del campo eléctrico resultante **E**, en un punto del espacio donde dos o más ondas de luz se superponen, es igual a la suma vectorial de las perturbaciones constitutivas independientes. Entonces brevemente: *la interferencia óptica se puede decir que es una interacción de dos o más ondas de luz que producen una irradiancia resultante, la cual se desvía de la suma de las irradiancias componentes.*

En el régimen de la interferometría tradicional se supone que solo existen variaciones espaciales debido a la información que nos interesa determinar, es decir, se está trabajando en el régimen estacionario.

El patrón de interferencia resultante depende de la diferencia de fase relativa entre las ondas constitutivas, las cuales si provienen de emisores separados el patrón resultante no se sostendrá el tiempo suficiente para ser fácilmente observable (una fuente típica contiene un gran número de átomos excitados, cada una capaz de radiar un tren de onda aproximadamente por 10^{-8} seg). Así en el mejor de los casos se podrá sostener un patrón de interferencia en el espacio por 10^{-8} seg para luego cambiar sucesivamente, por lo que sería inútil tratar de observar o fotografiar este patrón.

Para tener un patrón de interferencia por un tiempo razonable y que pueda ser observable, las dos ondas deben ser mutuamente coherentes, donde la coherencia entre éstas se logra solamente en dos casos:

1) Cuando tienen su origen en una misma fuente (coherencia espacial).
2) Cuando son monocromáticas, como es el caso de los láseres (coherencia temporal).

Los patrones más definidos existirán cuando las ondas que interfirieron tengan amplitudes iguales o casi iguales, lo cual equivale a una visibilidad de uno. En este caso las regiones centrales de las franjas claras y obscuras corresponden a interferencia totalmente constructiva o destructiva.

Matemáticamente representamos al fenómeno de interferencia al sumar dos ondas (Ec.1.1) con valores de fase, $\mathbf{k}(x,y)$, diferentes (recordemos que estamos en el caso estacionario, entonces la componente temporal ω es constante).

$$E(x,y;t) = E_o(x,y)e^{i(k\cdot r+\omega t+\epsilon)} \qquad (1.1)$$

Si sumamos dos ondas de la forma de la Ec.1.1, con diferente fase (pero con la misma frecuencia temporal ω) y se calcula su intensidad, se encuentra la siguiente expresión (Ec.1.2) (Hecht, 1986):

$$I(x,y) = |1 + V(x,y)\cos(\phi(x,y))| \qquad (1.2)$$

donde $\phi(x,y) = [(\mathbf{k}-\mathbf{k}')\cdot \mathbf{r}]$, la *diferencia de camino óptico* (DCO), contiene las variaciones entre la fase de las dos ondas constitutivas y V(x,y) se define como la *visibilidad de las franjas*.

En la práctica, además de considerar a las ondas que interfieren debemos tener en cuenta que el sistema de prueba introduce perturbaciones en el continuo de la imagen y que el campo del interferograma es finito. Estos efectos se representan de una manera más general al introducir en la Ec.1.2 componentes que representen estos efectos (Roddier, 1987):

$$I(x,y) = A(x,y)\ D(x,y)[1 + V(x,y)\cos[\phi(x,y)]] \qquad (1.3)$$

donde A(x,y) es el continuo el cual se identifica con las componentes del interferograma debidas a errores del sistema óptico de prueba (componentes ópticas y mecánicas defectuosas) y

$$D(r) = \begin{cases} 1 & \textit{Dentro del interferograma} \\ 0 & \textit{Fuera del interferograma} \end{cases} \qquad (1.4)$$

Entonces, el problema básico con el que se encuentra la interferometría es: ***conocer explícitamente la forma de la fase ($\phi(x,y)$) a partir de la información en un interferograma.***

1.2. FUNDAMENTOS DE INTERFEROMETRIA HETERODINA

Dentro de las diversas formas de analizar un interferograma, podemos resaltar a la **interferometría heterodina** la cual consiste en analizar simultáneamente varios interferogramas, en vez de uno sólo como en las técnicas convencionales (lo cual comúnmente se hace localizando los máximos o mínimos de interferencia) con el fin de determinar la fase del frente de onda bajo prueba.

Esta puede dividirse principalmente en dos grandes grupos, que han sido desarrollados en los últimos años (Takeda, 1989). La primera consiste en usar una frecuencia portadora que varía en el tiempo (frecuencai portadora temporal, FTP) en la cual se tiene un interferograma que varía en el tiempo de la forma (Fig.1.1):

$$I(x,y;t) = A(x,y)\ D(x,y)\ \{1 + V(x,y)\cos[2\pi f_0 t + \phi(x,y)]\} \qquad (1.5)$$

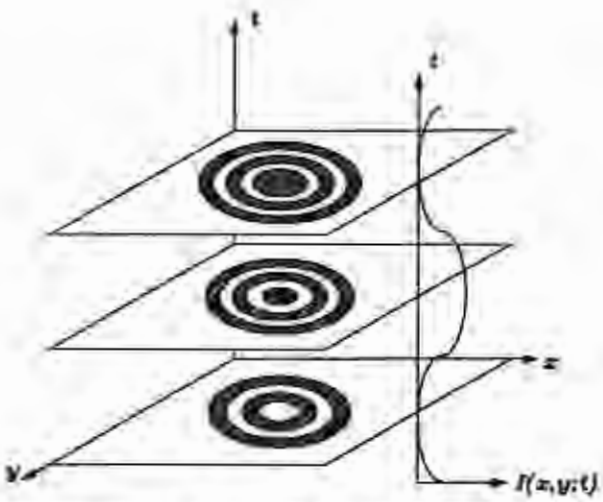


Figura 1.1 Principio básico de la interferometría heterodina basada en la técnica de Frecuencia Temporal Portadora (FTP).

donde nuevamente $\phi(x,y)$ es la incógnita a conocer; para lograr determinar esta variable se introduce un corrimiento temporal de frecuencia f_o, usando algún dispositivo. La interferometría de corrimiento de fase se puede considerar como la versión discreta de estas técnicas, donde el interferograma se muestrea en un periodo en intervalos de tiempo de $\Delta t = 1/Nf_o$:

$$I(x,y;n\Delta t) = A(x,y)\ D(x,y)\ \{1 + V(x,y)\cos[2\pi\frac{n}{N} + \phi(x,y)]\} \qquad (1.6)$$

La segunda técnica consiste en usar una frecuencia portadora que varía espacialmente (frecuencia portadora espacial, FEP), en la cual se analiza un sólo interferograma (Fig.1.2):

$$I(x,y) = A(x,y)\ D(x,y)\ \{1 + V(x,y)\cos[2\pi\vec{f}_0\cdot\vec{r} + \phi(x,y)]\} \qquad (1.7)$$

donde f_o $(=(f_{ox},f_{oy}))$ es la frecuencia portadora que se logra inclinando el frente de onda de referencia respecto al de prueba.

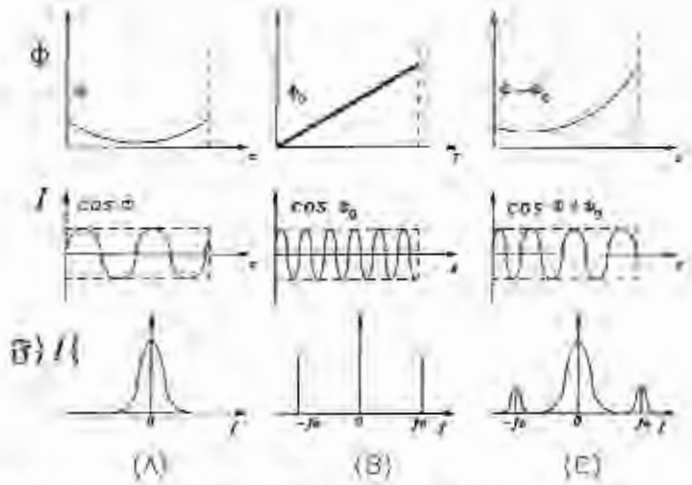


Figure 1.2 Principio básico de la interferometría heterodina mediante la técnica de FEP. (A) Frente de onda original. (B) Frecuencia portadora. (C) Combinación de (A) y (B).

De los trabajos publicados en interferometría heterodina, la mayoría trata el problema de FTP y los interferómetros comerciales más precisos con que se cuenta en la actualidad se basan en esta técnica. Sin embargo, las técnicas de FEP (que no han sido muy estudiadas pese a las ventajas que presenta en varias áreas) ofrece una alternativa más económica a los métodos comerciales actualmente en uso.

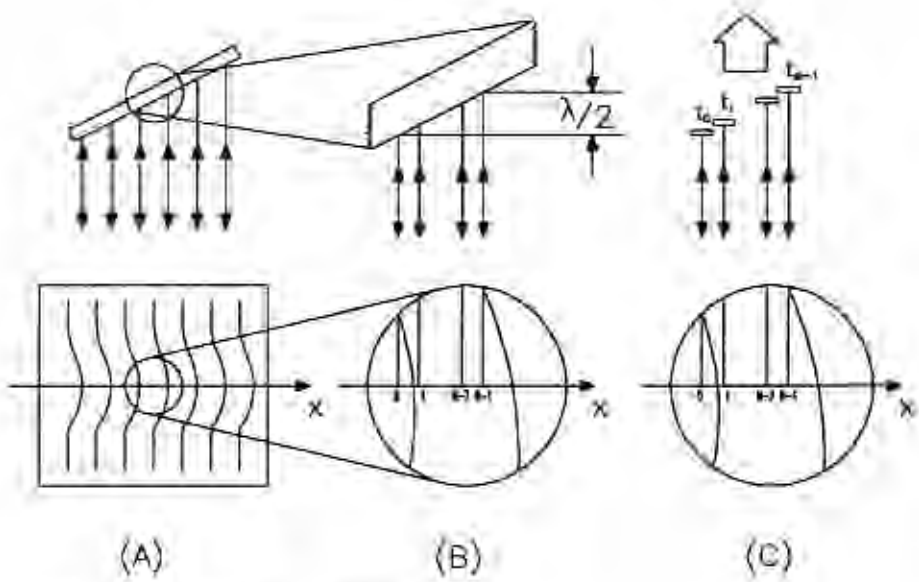


Figura 1.3 Relación entre las técnicas de FTP y FEP.

En este trabajo nos vamos a enfocar precisamente en la segunda de las técnicas de análisis heterodino de interferogramas: las técnicas de frecuencia espacial portadora.

La forma más común de generar una FEP es el introducir una inclinación (en inglés TILT) en el espejo de referencia (Fig.1.3A). Podemos interpretar las técnicas de FEP de la siguiente manera: Si muestreamos una de las franjas del interferograma en N puntos (Fig.1.3B), y consideramos la DCO en cada punto a partir del espejo que hemos inclinado, observamos que esta misma deformación se obtiene con N espejos desplazados en $\lambda/2N$ entre ellos (Fig.1.3C).

Tomando en cuenta que la fase y la amplitud de la DCO a conocer no cambia mucho de un intervalo a otro en un periodo de la frecuencia portadora, entonces el espejo inclinado funciona como un sistema multicanal de interferómetros de desplazamiento de fase paralelos, en donde todos los corrimientos de fase se encuentran simultáneamente codificados espacialmente en un mismo interferograma (en vez de varios muestreados en el tiempo con algún dispositivo de corrimiento de fase como las tecnicas de FTP).

Es precisamente este paralelismo espacial lo que constituye la base de las técnicas de FEP, lo cual hace posible las mediciones interferométricas heterodinas en un sólo interferograma. Podemos encontrar las siguientes ventajas y desventajas de las técnicas de FEP frente a las de FTP:

Ventajas:

1) Se requiere de un sólo interferograma en las técnicas de FEP en vez de los múltiples interferogramas a diferentes tiempos requeridos en las técnicas de FTP.
2) En FEP no se requiere de ningún dispositivo especial para generar la frecuencia portadora, mientras que en FTP se necesita algún elemento de corrimiento o desplazamiento de fase.

Desventajas:

1) En FEP se requiere un detector con una resolución espacial mayor que en FTP.
2) La respuesta del detector debe ser completamente uniforme.

Claramente los requerimientos sobre el detector constituyen las principales limitantes en las técnicas de FEP, estudiadas por Nugent (1985). Estas técnicas no eran posibles sino hasta hace poco tiempo, en el cual el desarrollo de técnicas de fabricación de dispositivos de alta resolución ha inclinado la balanza en su favor.

1.3. INTERFEROMETRIA DE FOURIER

Como ya se mencionó, el problema básico de la interferometría es conocer la información de la fase $\phi(x,y)$, a partir de la información en un interferograma. Los algoritmos en las técnicas de FTP son relativamente simples debido a que emplean la información de un parámetro adicional (el tiempo) de manera que el problema se reduce a leer la fase de una señal sinusoidal en el tiempo con la información espacial fija en el punto de prueba. En las técnicas de FEP los algoritmos son un poco más complicados debido a que toda la información está mezclada en las coordenadas espaciales. Debido a que hay que separar la información que se encuentra mezclada en las coordenadas espaciales, ésto se hace en el dominio de frecuencias directa o indirectamente.

Los métodos indirectos fueron los primeros que aparecieron, realizados en forma digital (Ichioka, 1972) o analógica (Mertz, 1983, Macy, 1983 y Womack, 1984) en los cuales se realiza básicamente una convolución sinusoidal del interferograma para conocer el frente de onda. Los métodos directos aparecieron poco más tarde con los trabajos de Takeda (1982) en el caso unidimensional y después Bone (1986) para el caso bidimensional; poco después Roddier y Roddier (1987) dieron una interpretación holográfica a este análisis con el cual se dió un gran paso en la comprensión de las técnicas de FEP.

En este trabajo se trata el caso del análisis de interferogramas mediante técnicas de FEP directa con interpretación holográfica, para lo cual reescribimos la Ec.1.7 de la siguiente forma:

$$\begin{aligned} I(x,y) &= A(x,y)\, D(x,y)\, \{\, 1+V(x,y)\cos[2\pi f_0 \cdot r - \phi(x,y)]\} \\ &= A(x,y)\, D(x,y)\, \{\, 1+ReC(r)e^{2i\pi f_0 \cdot r}\,\} \\ &= A(x,y)\, D(x,y)\, \{1+\tfrac{1}{2}C(r)e^{2i\pi f_0 \cdot r}+\tfrac{1}{2}C^*(r)e^{2i\pi f_0 \cdot r}\} \end{aligned} \qquad (1.8)$$

donde C(**r**), la **visibilidad compleja**, se define como:

$$C(\mathbf{r}) = V(\mathbf{r})e^{-i\phi(\mathbf{r})} \tag{1.9}$$

Entonces el problema se ha convertido en determinar C(**r**) a partir de I(**r**). Calculando la transformada de Fourier bidimensional de I(**r**) se obtiene (Ec.1.10):

$$\begin{aligned}\mathcal{F}\{I(x,y)\} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)e^{-i2\pi(ux+vy)}dx\,dy \\ &= \hat{D} * \hat{A} * \{\delta(f) + \tfrac{1}{2}\hat{C}(f-f_0) + \tfrac{1}{2}\hat{C}^*(-f-f_0)\}\end{aligned} \tag{1.10}$$

donde el asterisco (*) denota producto de convolución, el asterisco como superíndice (*) denota complejo conjugado y la $\delta(\mathbf{f})$ representa la distribución delta de Dirac.

Considerando el caso ideal de un interferograma infinito con distribución uniforme [A(**r**)=D(**r**)=1], la transformada de Fourier del interferograma se reduce al último término de convolución de la Ec.1.10, es decir, consta sólo de tres términos: Una distribución impulsiva en el origen $\delta(\mathbf{f})$, y dos términos desplazados del origen centrados en las frecuencias $+\mathbf{f}_o[=(f_{ox},f_{oy})]$ y $-\mathbf{f}_o$, la transformada de Fourier de la visibilidad compleja y su compleja conjugada.

Notemos que este es el mismo patrón que presenta un holograma, de aquí que se llame a ésta la *interpretación holográfica*. Este análisis muestra que la visibilidad compleja puede ser determinada sin ambigüedad siempre que C(**r**) sea una función con frecuencias menores a una frecuencia de corte f_c, y $|\mathbf{f}_o|$ sea mayor a f_c; esta condición se cumple siempre que las franjas en el interferograma sean abiertas.

La visibilidad compleja se obtiene si se realiza un corrimiento $-\mathbf{f}_o$ en el espacio de frecuencias y se aplica un filtro pasa bajas con radio f_c, con ésto se ha centrado en el origen el orden uno de difracción (Fig.1.4) y finalmente se obtiene la transformada de Fourier inversa. De la Ec.1.10, la amplitud de esta transformada inversa es la visibilidad del frente de onda y en la fase se tendrá el frente de onda buscado.

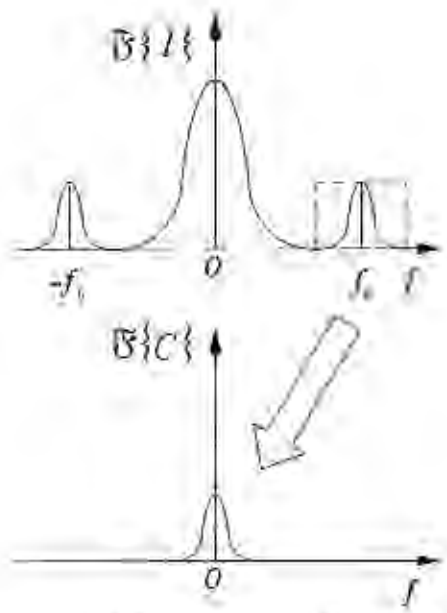


Figure 1.4 Proceso fundamental en el método de la interferometría de Fourier.

Notemos que esta técnica es válida sólo para interferogramas que dan información del frente de onda (v.g. Michelson, Twymann-Green), no de su derivada (v.g. Burch, desplazamiento lateral). Para realizar el análisis sobre el segundo tipo de interferogramas es necesario operaciones adicionales tanto en el espacio de coordenadas y en el espacio de frecuencias (Roddier y Roddier, 1991) .

Entonces, mediante las técnicas de FEP se ha llegado a conocer **explícitamente** la forma del frente de onda $\phi_s(x,y)$, relacionado con la DCO mediante un escalamiento ($W'(x,y) = \lambda/2\pi\ \phi_s(x,y)$), en forma sencilla y directa.

1.4.PRINCIPALES PROBLEMAS EN INTERFEROMETRIA DE FOURIER

Las dificultades en la interferometría de Fourier provienen principalmente debido a que ningún interferograma es infinito y que en general el continuo es no uniforme. Como D(r) tiene soporte finito, su transformada de Fourier es analítica y entonces su espectro es infinito. Después de convolucionar en la Ec.1.5 las transformadas de D(r) y A(r), los tres términos descritos arriba se empalmarán produciendo errores en la determinación de la visibilidad compleja. A continuación se discuten posibles formas de resolver los principales errores en el método de la interferometría de Fourier.

1.4.1.ESTIMACION DEL CONTINUO

Se encuentra que las fluctuaciones en el continuo A(r) debidas a imperfecciones en el sistema óptico de prueba afectan seriamente a la determinación del frente de onda, entonces si se quiere tener un buen resultado es necesaria una medición independiente del continuo. Con ayuda del patrón complementario en un interferograma, el cual tiene la fase complementaria, el continuo es fácil de determinar por adición de ambos (si se supone que los errores debidos al sistema óptico son aditivos). Dividiendo I(r) por A(r) se eliminan los efectos debidos a no uniformidades del continuo, teniendo una mejor estimación del frente de onda[1].

$$I'(r)=\frac{I(r)}{A(r)} \qquad (1.11)$$

El considerar estos efectos puede llevar a obtener resultados entre $\lambda/200$ y $\lambda/1000$ (Frankowsky et al, 1989). Si éstos no son considerados el resultado obtenido será bueno en el orden de $\lambda/50$ aproximadamente, que es el límite de resolución obtenido con la prueba de Foucault (Malacara, 1978).

---

[1] En las pruebas típicas se usa como fuente un laser, por lo que el ruido es gaussiano con frecuencia alta.

1.4.2.SOPORTE ANALITICO DEL INTERFEROGRAMA

El soporte finito de los interferogramas es una de las fuentes de error en el análisis de interferogramas mediante la técnica de interferometría de Fourier. Estos son debidos a las discontinuidades en el borde del interferograma, que producen ondulaciones en el plano de Fourier. Se han propuesto dos formas de resolver el problema: Extrapolar las franjas a cubrir el campo del espacio de coordenadas (Roddier et al, 1987) o multiplicar por un filtro que suavica la transformada de Fourier (Takeda, 1982). El primero basado en el algoritmo de extrapolación de Gerchberg (Gerchberg, 1974) es un método elegante que elimina el problema, pero requiere de tiempos de cálculo largos. El segundo método es más conveniente por la velocidad de cálculo, donde se pueden usar tipos de diferentes filtros (Fig.1.5): rectangular, Hanning o poligonal. Se puede pensar en una gran variedad de filtros buscando suavizar el espectro del interferograma para evitar máximos secundarios que generan empalmes, por lo que no es conveniente usar filtros del tipo rectangular (que tienen brincos abruptos).

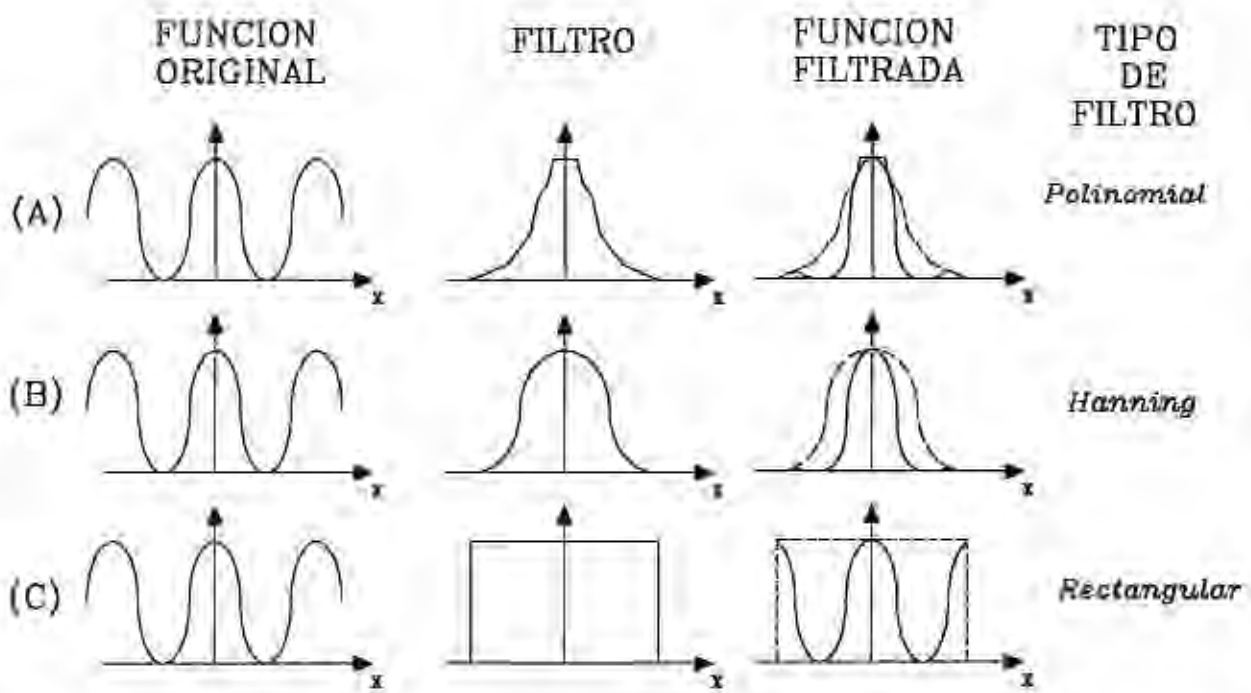


Figure 1.5 Diferentes tipos de filtros usados al procesar una imagen.(A) Filtro polinomial. (B) Filtro de Hanning. (C) Filtro rectangular.

Se han estudiado diferentes filtros apodizadores (Frankowsky *et all*, 1989), encontrando numéricamente que el filtro de Hanning (Fig.1.5B), usado originalmente por Takeda, es el que mejor elimina las ondulaciones. Analíticamente, para el radio del interferograma normalizado, el filtro tiene la forma:

$$H(|r|)=\begin{cases}\frac{1}{2}(1+\cos\pi|r|) & |r|\le 1\\ 0 & |r|>1\end{cases} \qquad (1.12)$$

Numéricamente este filtro es fácil de implementar, aunque su transformada de Fourier analítica es muy compleja (Fig.1.6) que no se encontró explícitamente en la literatura, solo aproximaciones (sumas infinitas de funciones de Bessel), sin embargo su comportamiento numérico es el más adecuado.

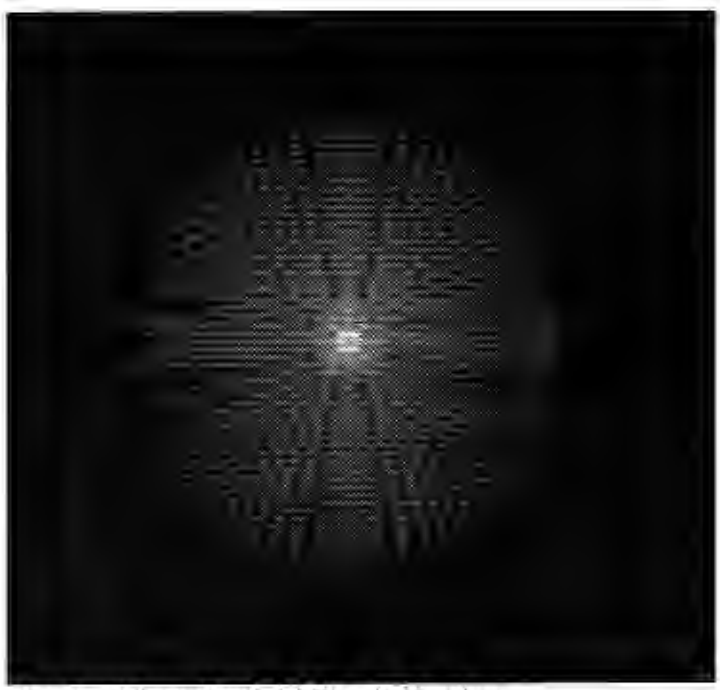


Figure 1.6 Transformada de Fourier bidimensional del filtro de Hanning.

1.4.3.DESDOBLAMIENTO DE FASE

La fase de la visibilidad compleja se calcula a partir de la transformada de Fourier inversa de la imagen filtrada en el plano de frecuencias. Esta fase es calculada módulo 2π, es decir, en la rama principal de la función arcotangente. Para conocer la fase en forma completa hay que realizar el proceso llamado **desdoblamiento de fase**. Matemáticamente el desdoblamiento de fase se define como la integral de la derivada de la fase comprimida (o sin desdoblar). En la práctica, debido a que se tiene un muestreo finito de la fase, este proceso se realiza sumando diferencias de fase (Fig.1.7). Para corregir las discontinuidades debidas al cálculo en la rama principal de la función arcotangente, se debe determinar una corrección a la fase calculada ($\phi_c(x,y)$), que llamaremos $\phi_c(x,y)$, de manera de reconstruir la fase real ($\phi_d(x,y)$):

$$\phi_d(x,y) = \phi_o(x,y) + \phi_c(x,y) \qquad (1.13)$$

El primer paso es calcular las diferencias de fase entre el i-ésimo punto muestreado y el punto precedente (i -1), para la fase calculada a lo largo de un renglón (o columna):

$$\Delta\phi_d(x_i,y) = \phi_d(x_i,y) - \phi_d(x_{i-1},y) \qquad (1.14)$$

donde el subíndice i corre de 1 al último punto muestreado (N).

Debido a que se tiene un número grande de puntos, las variaciones en la fase son lentas comparadas con el intervalo de muestreo, de manera que el valor absoluto de las diferencias de fase ($|\Delta\phi_c(x,y)|$) es mucho menor a 2π en los puntos donde la fase es continua, pero tiene un valor casi igual a 2π en la DCO para los puntos muestreados donde existe una discontinuidad de 2π. Entonces con un criterio apropiado para el valor absoluto de la diferencia de fase (v.g. $0.7\times2\pi$), se pueden especificar los puntos donde existen discontinuidades, y con el signo de la diferencia de fase podemos conocer la dirección de las correciones. Esto es válido ya que debido a la hipotesis de envolvente lenta se cumple el teorema del muestreo aún en el caso de que se sobrepase el límite de Nyquist con la información a priori de que la superficie bajo prueba es suave y entonces tiene derivadas continuas (Greivenkamp. 1987).

Ya conocidos los puntos donde se tienen las discontinuidades, el segundo paso es determinar la correción de fase para cada punto. Debido a que sólo nos interesan diferencias de fase relativas, sin pérdida de generalidad podemos escoger que el primer punto tenga una fase relativa de 0 ($\phi_o(x_0,y)=-\pi$), luego asignamos la misma fase para los puntos x, con i=0,1,...,k-1, hasta una discontinuidad en el punto k en donde asignamos $\phi_o(x_k,y)= \phi_o(x_0,y)\pm 2\pi$; se sigue el mismo proceso para cada discontinuidad.

De esta forma se construye para cada renglón (o columna) la corrección a la fase necesaria para desdoblar la fase (Fig.1.7B). Finalmente sumamos ésta a la fase calculada $\phi_c(x,y)$ y obtenemos la **fase reconstruida** (Ec.1.13) para un renglón (o columna) (Fig.1.7C).

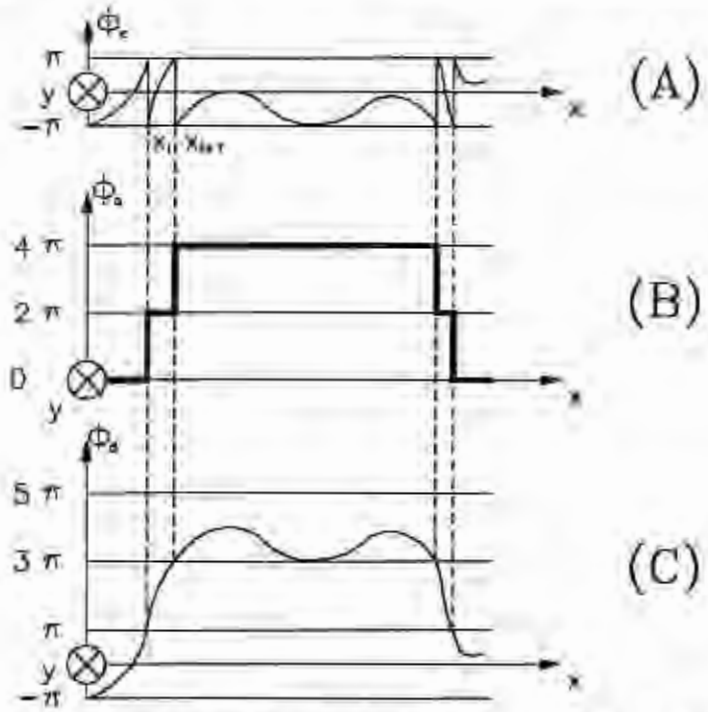


Figure 1.7 *(A)* DCO con discontinuidades debidas al cálculo en la rama principal de la función arcotangente. *(B) Fase* usada para corregir las discontinuidades en (A). *(C)* Fase continua.

En el caso de imágenes bidimensionales, se requiere además realizar el desdoblamiento de fase a lo largo de una columna (o renglón) para un punto x arbitrario, de manera de referir la reconstrucción de los renglones (o columnas) a esta reconstrucción adicional.

Según Itoh (1982) esto es posible sólo si los brincos que se dan en la fase comprimida son menores a 2π, es decir, está en el intervalo entre $-\pi$ y π. Si esta condición no se cumple, esto da lugar a *dislocaciones* en la reconstrucción de la fase, que son saltos en la fase mayores a 2π (Fig.1.8).

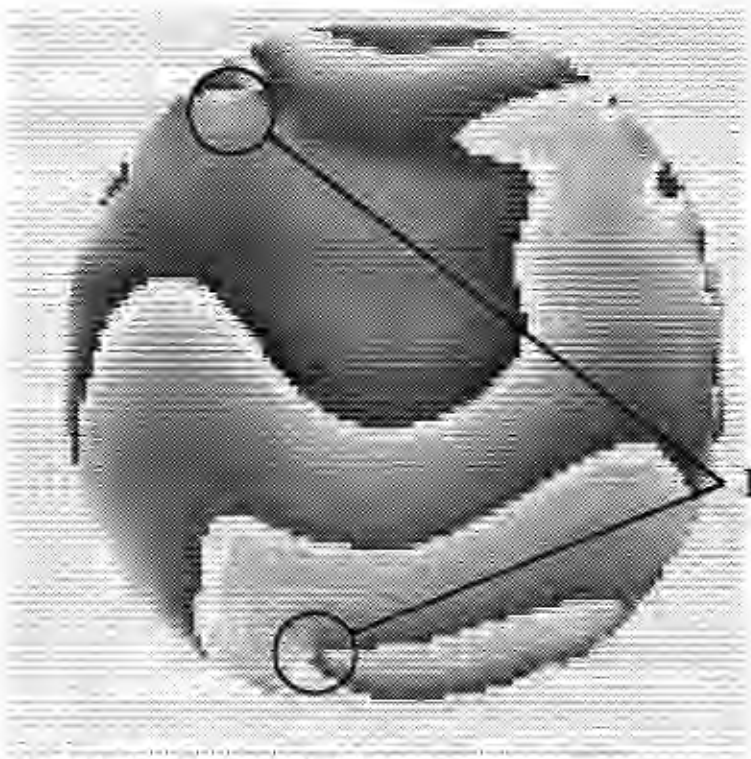


Figure 1.8 Fase desdoblada que muestra 2 dislocaciones.

Debido a este efecto, que ocurre sólo en el desdoblamiento de fase bidimensional, se tienen puntos donde la fase es indeterminada lo cual constituye la **principal fuente de error** en la interferometría de Fourier y técnicas relacionadas, siendo éste un problema todavía abierto.

En la literatura encontramos diferentes propuestas que no han dado una respuesta satisfactoria al problema: usando técnicas estadísticas (Itoh 1982, Itoh 1983), mínimos cuadrados (Hudgin 1977, Hunt 1979) o incluso autómatas celulares (Ghilia et al, 1987). La mejor aproximación ha sido dada por Barr (et al, 1991) en la cual asociando dislocaciones residuales, se eliminan prácticamente todas las dislocaciones.

1.5. ANALISIS DEL FRENTE DE ONDA

Como resultado del análisis de interferogramas usando la técnica de FEP se obtiene la DCO relacionada con el sistema óptico bajo prueba, que contiene información de las deformaciones de ésta respecto a un frente de onda ideal. Podemos analizar la DCO en forma cuantitativa o cualitativa. Cualitativamente se realiza un estudio topográfico de la DCO (W'(x,y)), que introduce el sistema óptico bajo prueba, a manera de visualizar las zonas "deformes". Cuantitativamente se desea conocer el grado de deformación que tiene ésta, lo cual se hace a través de parámetros como la deformación máxima de la DCO (deformación pico-valle) o el valor medio de la deformación (RMS). Se puede hacer un análisis más profundo hasta conocer la función de modulación (MTF) o el patrón de difracción que se esperarían del sistema.

Los parámetros cuantitativos más usados para analizar la DCO son los coeficientes de Zernike (o alternativamente los coeficientes de Seidel) los cuales proporcionan información del grado de deformación de la DCO en términos de diferentes aberraciones conocidas. Dichos coeficientes surgen de expander a la DCO en término de los polinomios de Zernike (Malacara, 1978), los cuales son polinomios ortogonales definidos sobre el círculo unitario. Estos pueden ser expresados como el producto de dos funciones: una que depende sólo del radio y otra con dependencia angular (Ec.1.15):

$$U_p(\rho,\theta) = R_n^{|l|}(\rho)\begin{Bmatrix}\cos(l\theta)_{l\le 0}\\ sen(l\theta)_{l>0}\end{Bmatrix} \qquad (1.15)$$

donde l es el parámetro de dependencia angular, n es el grado del polinomio, ρ es la distancia radial normalizada y θ es el ángulo medio desde el eje *y*. Se puede demostrar que $|l|$ es el mínimo coeficiente del polinomio R^l_n y que n y l son ambos pares o ambos impares, de manera que n-l es siempre par. Con estas propiedades tenemos entonces ½(n+1)(n+2) polinomios linearmente independientes U^l_n con grado menor o igual a n. Los polinomios radiales y angulares cumplen con las reglas de ortogonalidad (Ec.1.16 y Ec.1.17):

$$\int_0^1 R_n^l(\rho)R_{n'}^{l'}(\rho)\rho\,d\rho = \frac{1}{2(n+1)}\delta_{nn'} \tag{1.16}$$

$$\int_0^{2\pi} sen(l\theta)\cos(l'\theta)d\theta = 0 \qquad \forall\ l,l'$$

$$\int_0^{2\pi} sen(l\theta)sen(l'\theta)d\theta = \begin{cases} 0 & l\neq l' \\ 0 & l=l'=0 \\ \pi & l=l'\neq 0 \end{cases} \qquad \int_0^{2\pi} \cos(l\theta)\cos(l'\theta)dT = \begin{cases} 0 & l\neq l' \\ 2\pi & l=l'=0 \\ \pi & l=l'\neq 0 \end{cases} \tag{1.17}$$

de manera que se tiene la condición de ortogonalidad para los polinomios de Zernike (Ec.1.18):

$$\int_0^{2\pi}\int_0^1 U_n^l(\rho,\theta)U_{n'}^{l'}\rho\,d\rho\,d\theta = \begin{cases} \frac{\pi}{n+1}\delta_{n'n}\delta_{l'l} & l=0 \\ \frac{\pi}{2(n+1)}\delta_{n'n}\delta_{l'l} & l\neq 0 \end{cases} \tag{1.18}$$

La expansión de la DCO en términos de estos polinomios de Zernike a grado k es de la forma (Ec.1.19):

$$W'(\rho,\theta) = \sum_{n=0}^{k}\sum_{l=0}^{n} A_{nl}U_n^l(\rho,\theta) = \sum_{p=0}^{t} A_p U_p(\rho,\theta) \tag{1.19}$$

donde la doble sumatoria se reduce a una sola con ayuda del cambio de variable (Ec.1.20):

$$p = \frac{n(n+1)}{2} + l + \varepsilon_l \qquad \varepsilon_l = \begin{cases} 0 & l\geq 0 \\ 1 & l<0 \end{cases} \tag{1.20}$$

El máximo valor de p es el número total de polinomios de Zernike usados en el ajuste (⅛(k+1)(k+2)). Una lista de los primeros polinomios de Zernike la encontramos en la Tabla 1.1 (Wyant, 1988). Los coeficientes A_p se pueden conocer a partir de la varianza de ajuste σ^2, definida por (Malacara, 1992):

$$\sigma_f^2=\frac{1}{\pi}\int_0^{2\pi}\int_0^1[W'(\rho,\theta)-W(\rho,\theta)]^2\rho\,d\rho d\theta \qquad (1.21)$$

donde W'(x,y) es el frente de onda que ha sido determinado usando la técnica de interferometría de Fourier (DCO experimental) y W(x,y) es la DCO evaluada (DCO teórica). Usando la condición de ortogonalidad (Ec.1.18) y pedir que la variación de la varianza de ajuste sea cero, se obtiene:

$$A_p=\frac{\varepsilon_l(n+1)}{\pi}\int_0^{2\pi}\int_0^1 W_0'(x,y)U_p(x,y)\rho\,d\rho\,d\theta \qquad \varepsilon_l=\begin{cases}2 & l\neq 0\\ 1 & l=0\end{cases} \qquad (1.22)$$

con p = 0,1,...,L.

Con éste resultado es posible conocer el coeficiente de Zernike A_p solamente con la DCO experimental y polinomio de Zernike correspondiente, **sin necesidad de realizar ningún tipo de ajuste**, es decir realizando una integración numérica directamente.

Este resultado es de suma importancia ya que a diferencia del análisis clásico de interferogramas usado hasta ahora, donde el valor de los coeficientes de Zernike se hace seleccionando un número pequeño de puntos e interpolando (v.g. Forbes 1988, Malacara 1992) en este análisis se usa todo el frente de onda obtenido mediante técnicas de FEP para la obtención de los coeficientes de Zernike.

La interpretación de la DCO en términos de polinomios de Zernike es en ocasiones confusa debido a la naturaleza propia de éstos, los cuales contienen términos de polinomios de menor orden para compensar las aberraciones, de manera de que la varianza se minimice y al mismo tiempo sean ortogonales entre ellos. Una interpretación más sencilla es en término de los polinomios de Seidel, en la cual se hace un desarrollo del frente de onda a tercer orden. Este desarrollo, propuesto por Kingslake en 1925 (Malacara, 1978) tiene las forma (Ec.1.23):

TABLA 1.1. POLINOMIOS DE ZERNIKE

n	l	U_p	POLINOMIO	SIGNIFICADO
0	0	0	1	Pistón
1	1	1	$\rho \cos \theta$	Tilt x
PRIMER ORDEN		2	ρ sen θ	Tilt y
	0	3	$2\rho^2 - 1$	Defoco
2	2	4	$\rho^2 \cos 2\theta$	Astigmatismo 0°
		5	ρ^2 sen 2θ	Astigmatismo 45°
TERCER ORDEN	1	6	$(3\rho^2 - 2)\rho \cos \theta$	Coma x
		7	$(3\rho^2 - 2)\rho$ sen θ	Coma y
	0	8	$(6\rho^4 - 6\rho^2 + 1)$	Esférica
3	3	9	$\rho^3 \cos 3\theta$	Astigmatismo 0°
		10	$\rho^3 \cos 3\theta$	Astigmatismo 30°
	2	11	$(4\rho^2 - 3)\ \rho^2 \cos 2\theta$	Coma x
QUINTO ORDEN		12	$(4\rho^2 - 3)\ \rho^2$ sen 2θ	Coma y
	1	13	$(10\ \rho^4 - 12\rho^2 + 3)\rho \cos \theta$	
		14	$(10\ \rho^4 - 12\rho^2 + 3)\rho$ sen θ	
	0	15	$20\rho^6 - 30\rho^4 + 12\rho^2 - 1$	Esférica
4	4	16	$\rho^4 \cos 4\theta$	Astigmatismo 0°
		17	$\rho^4 \cos 4\theta$	Astigmatismo 22°
	3	18	$(5\rho^2 - 4)\rho^3 \cos 3\theta$	Coma x
		19	$(5\rho^2 - 4)\rho^3$ sen 3θ	Coma y
SEPTIMO ORDEN	2	20	$(15\rho^4 - 20\rho^2 + 6)\ \rho^2 \cos 2\theta$	
		21	$(15\rho^4 - 20\rho^2 + 6)\ \rho^2$ sen 2θ	
	1	22	$(35\rho^6 - 60\rho^4 + 30\rho^2 - 4)\ \rho \cos \theta$	
		23	$(35\rho^6 - 60\rho^4 + 30\rho^2 - 4)\ \rho$ sen θ	
	0	24	$70\rho^8 - 140\rho^6 + 90\rho^4 - 20\rho^2 + 1$	Esférica

$$W(\rho,\theta) = A\rho^4 + B\rho^3\cos\theta + C\rho^2(1+\cos^2\theta) + D\rho^2 + E\rho\cos\theta + F\rho\,sen\,\theta \qquad (1.23)$$

donde A es el coeficiente de la aberración esférica,

B es el coeficiente de la coma,

C es el coeficiente del astigmatismo,

D es el coeficiente del defoco,

E es el coeficiente del tilt alrededor del eje x,

F es el coeficiente del tilt alrededor del eje y,

G es el coeficiente del término pistón.

Podemos conocer estos coeficientes a partir de los coeficientes de Zernike, realizando una expansión a tercer orden (Ec.1.24)

$$\begin{aligned} W(\rho,\theta) = {} & U_0 + U_1(\rho\cos\theta) + U_2(\rho\,sen\,\theta) + U_3(2\rho^2-1) + \\ & + U_4(\rho^2\cos\theta) + U_5(\rho^2 sen\,\theta) + U_6(3\rho^2-2)\rho\cos\theta) + \\ & + U_7((3\rho^2-2)\rho\,sen\,\theta) + U_8(6\rho^4-2\rho^2+1) \end{aligned} \qquad (1.24)$$

reescribimos esta ecuación, asociando los términos en forma similar que en la Ec.1.23, notando que los términos de coma y astigmatismo tendrán en general una dirección arbitraria (θ_0 y θ, respectivamente), donde si los pensamos como vectores en el espacio, se cumplen las siguientes igualdades:

Para el astigmatismo:

$$\begin{cases} U_4 = \sqrt{U_4^2+U_5^2}\cos\theta_0 \\ U_5 = \sqrt{U_4^2+U_5^2}\,sen\,\theta_0 \end{cases} \quad \theta_0 = 2\theta_0' \qquad (1.25)$$

$$
\begin{aligned}
U_4\cos\theta + U_5 sen\theta &= \sqrt{U_4^2+U_5^2}\,(\cos 2\theta\cos 2\theta_0 + sen2\theta sen2\theta_0) \\
&= \sqrt{U_4^2+U_5^2}\,\cos 2(\theta-\theta_0) \\
&= \sqrt{U_4^2+U_5^2}\,(2\cos^2(\theta-\theta_0)-1)
\end{aligned}
\tag{1.26}
$$

Para la coma:

$$
\begin{cases}
U_6 = \sqrt{U_6^2+U_7^2}\cos\theta_1 \\
U_7 = \sqrt{U_6^2+U_7^2}\,sen\theta_1
\end{cases}
\tag{1.27}
$$

$$
\begin{aligned}
U_6\cos\theta + U_7 sen\theta &= \sqrt{U_6^2+U_7^2}(\cos\theta\cos\theta_1 + sen\theta sen\theta_1) \\
&= \sqrt{U_6^2+U_7^2}\cos(\theta-\theta_1)
\end{aligned}
\tag{1.28}
$$

Entonces reescribiendo la Ec.1.24. asociando los diferentes términos, tenemos:

$$
\begin{aligned}
W(\rho,\theta) = \quad & \rho^4(6U_8) && + \rho^3\cos(\theta-\theta_1)(3\sqrt{U_6^2+U_7^2}) \\
& + \rho^2(2U_3 - 6U_8 \pm \sqrt{U_4^2+U_5^2}) && + \rho^2\cos^2(\theta-\theta_0)(\mp 2\sqrt{U_4^2+U_5^2}) \\
& + \rho\cos\theta\,(U_1 - 2U_6) && + \rho\, sen\theta\,(U_2 - 2U_7) \\
& + (U_0 + U_8 - U_3)
\end{aligned}
\tag{1.29}
$$

Notemos que ésta es la misma expansión que en Ec.1.23. Relacionando los coeficientes de Zernike y de Seidel tal como se muestra en la tabla 1.2. (Wyant, 1988)., encontramos una relación aproximada para calcular los coeficientes de Seidel a partir de los coeficientes de Zernike de tercer orden. La relación es aproximada debido a la presencia de monomios de ρ en polinomios de Zernike de orden mayor que nó se toman en cuenta en este análisis (ya que normalmente los coeficientes asociados son mucho menores que los de tercer orden).

TABLA 1.2. RELACION ENTRE LOS COEFICIENTES DE SEIDEL Y LOS DE ZERNIKE

ESFERICA

$$A = 6\,U_8$$

COMA

$$B = 3\,(U_7^2 + U_6^2)^{1/2}$$

dirección = $\text{atan}(U_7 / U_6 + c)$

ASTIGMATISMO

$$C = \pm 2\,(U_5^2 + U_4^2)^{1/2}$$

dirección = $\text{atan}(U_5 / U_4 + c)$

NOTA: El signo es contrario al del cálculo del defoco.

DEFOCO

$$D = 2\,U_3 - 6\,U_8 \pm (U_5^2 + U_4^2)^{1/2}$$

NOTA: El signo se selecciona para minimizar el valor absoluto de la magnitud.

TILT

Componente x: $E = U_1 - 2\,U_6$

Componente y: $F = U_2 - 2\,U_7$

Magnitud tilt = $(E^2 + F^2)^{1/2}$ dirección = $\text{atan}\,(F/E + c)$

PISTON

$$G = U_0 - U_3 + U_8$$

En todos los casos:

$c = 0$ si numerador > 0

$c = -\pi$ si numerador < 0 y denominador > 0

$c = \pi$ si numerador < 0 y denominador < 0

CAPITULO 2

ASPECTOS PRACTICOS INVOLUCRADOS EN LA INTERFEROMETRIA DE FOURIER

2.1 PROCESO DE ANALISIS DE INTERFEROGRAMAS

Con el desarrollo del algoritmo de la transformada rápida de Fourier (FFT) (Cooley y Tukey, 1965) y el perfeccionamiento de las técnicas de procesamiento digital de imágenes, desde sus comienzos el análisis de interferogramas mediante la técnica de **interferometría de Fourier** fué elaborado usando métodos digitales (Takeda, 1981), es por ésto que el trabajo que desarrollamos consistió en elaborar los diferentes algoritmos involucrados en el proceso, listados en el apéndice I.

De acuerdo a lo que se discutió en la sección 1.3, el análisis para interferogramas con información del frente de onda se realiza siguiendo los siguientes pasos:

- Captura de un interferograma.
- Detección del borde del interferograma.
- Filtro de Hanning del interferograma (con radio normalizado).
- Transformada de Fourier del interferograma.
- Selección, filtraje y desplazamiento del primer orden de difracción.
- Transformada inversa de Fourier de la transformada filtrada.
- Cálculo de fase (en la rama principal de la función arcotangente).
- Desdoblamiento de fase.
- Evaluación del frente de onda.

Para llevar a cabo el proceso completo se debe realizar en forma secuencial cada uno de estos pasos. A continuación detallamos cada uno de éstos, relacionándolos con los listados de los algoritmos en lenguaje de programación "C" en el apéndice I.

2.2 CAPTURA DE UN INTERFEROGRAMA

El primer paso en el proceso de análisis de interferogramas es tener uno de estos patrones, el cual puede ser real o simulado. La forma de trabajar con un interferograma real es capturar a través de una cámara las franjas de interferencia, las que se transfieren a la memoria de una computadora usando una **tarjeta digitalizadora** (Fig. 2.1). En la memoria de la computadora la imagen se almacena en una matriz de pixeles, donde cada uno representa en forma numérica el valor correspondiente al nivel de gris de la imagen en ese punto. La resolución y niveles de gris de la imagen dependen del diseño de la tarjeta, las resoluciones más comunes son de 256 por 240 o 512 por 480 pixeles, con 64 o 256 niveles de gris. El el análisis del interferograma es independiente de la forma en que fué capturado, pero por razones que veremos al estudiar la transformada rápida de Fourier es conveniente guardar la imagen en arreglos cuadrados de múltiplos de 2 (64 por 64, 128 por 128, ...), con 64 niveles de gris.

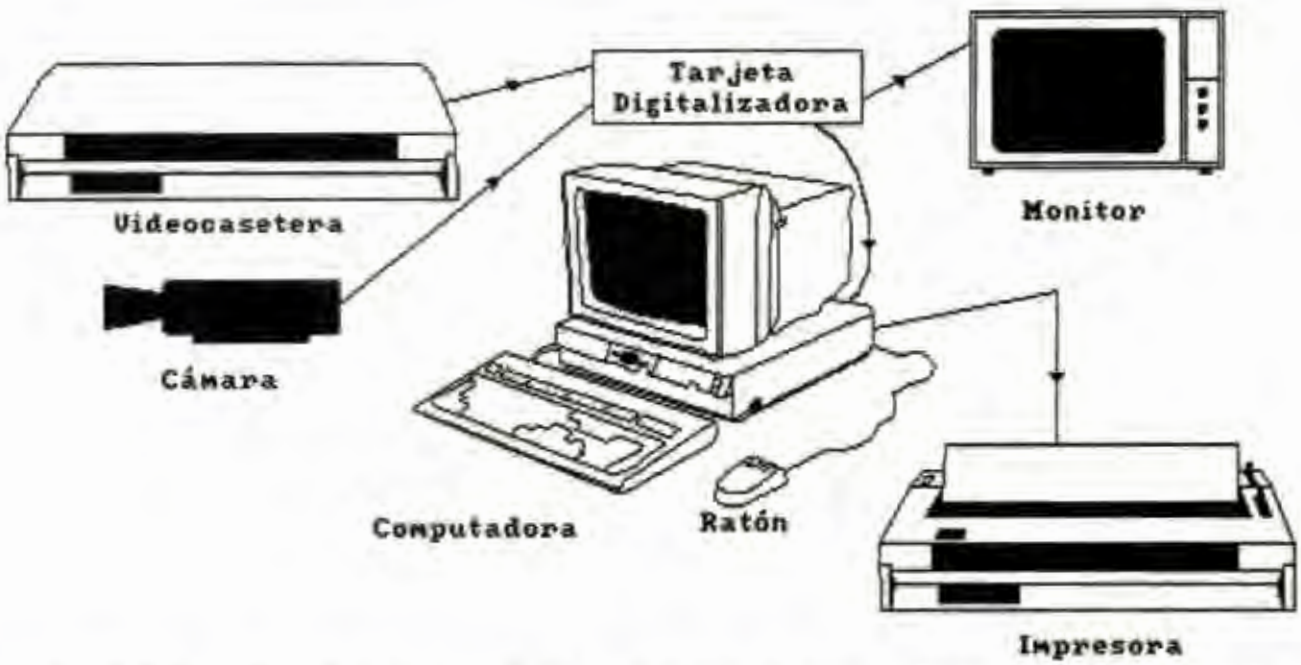


Figura 2.1 Equipo usado para digitalizar imágenes: Computadora con tarjeta digitalizadora, cámara o videocasetera a la entrada y un monitor o una impresora a la salida.

El hecho de que el trabajo desarrollado sea independiente de una tarjeta digitalizadora y se requieran 64 niveles de gris, es debido a que el despliegue de las imágenes lo realizamos directamente en el monitor de la computadora, usando el modo de despliegue 19 de los monitores VGA (Virtual Graphic Array) que tiene una resolución de 320 por 200 pixeles con 256 colores, de los cuales los primeros 64 corresponden a niveles de gris. Se seleccionó este tipo de monitores debido a que actualmente son los más comunes en el mercado y además permiten este tipo de exhibición. Si bien con este método la resolución de la imagen es relativamente baja[1], se tiene la ventaja de poder analizar interferogramas, una vez digitalizados, en cualquier computadora con monitor VGA.

La idea básica de esta exhibición es activar el modo 19 del monitor VGA y direccionar a éste para pintar cada pixel (a partir de la dirección hexadecimal en memoria A000:0000). Como normalmente las imágenes que se obtienen de una tarjeta digitalizadora son de un byte por pixel, esto equivale a usar una variable de tipo caracter para almacenar cada pixel. En la sección I.5.3 se lista el programa usado para este propósito, haciendo uso de la rutina Agraficas.h (sección I.4)[2]. En este programa se designa espacio de memoria para cada renglón de la imagen (usando la función de **C** farcalloc). El tamaño de las imágenes usadas es de 256 (NMAX = 256), pero se puede cambiar a alguna de las otras resoluciones (64, 128 o 512).

Para trabajar con interferogramas simulados, se generó un programa que numéricamente calcula un interferograma de tercer orden usando polinomios de Seidel (Ec.1.23). El listado de este programa se encuentra en la sección I.12. En este programa el control de los coeficientes se hace usando las teclas: <Flecha arriba>, <Flecha abajo>, <Flecha izquierda>, <Flecha derecha>, <Gris +> (despliega el interferograma), <Insert> (graba la imagen previamente desplegada) y <Esc> (salir del programa). Este es un ejemplo de como leer teclas que están fuera del código ASCII (American Standard Code for Information Interchange).

---

[1] Se puede aumentar la resolución de la imágenes trabajando en monitores Super Vga que permiten una resolución de 640 por 480 con 64 niveles de gris.

[2] Solo se requieren las funciones agraficas() y modeset() si se desea exhibir sin usar ningún letrero.

2.3 DETECCION DE BORDE Y FILTRO DE HANNING

Con el interferograma capturado se procede a determinar el contorno de éste, lo cual se hace usando el ratón, para lo que se usaron las funciones de la interrupción 51 del sistema operativo[3] (sección I.6) que ayudan a mover un circulo generado para determinar el contorno (Hearn, 1988). Ya con el borde seleccionado a partir de un punto central (x_c,y_c) y un radio r_c, se aplica el filtro de Hanning (Ec.1.12) en el círculo y el resultado se salva en un archivo, para después continuar con el proceso. Los efectos que se observan al no considerar el filtro de Hanning se discuten en la sección de resultados. El listado de este programa se encuentra en la sección I.3. La primera parte del programa muestra como generar un circulo y a trabajar con el ratón. La segunda parte muestra como realizar filtrajes en imágenes. El ambiente que genera este programa se muestra en la Fig.2.2.

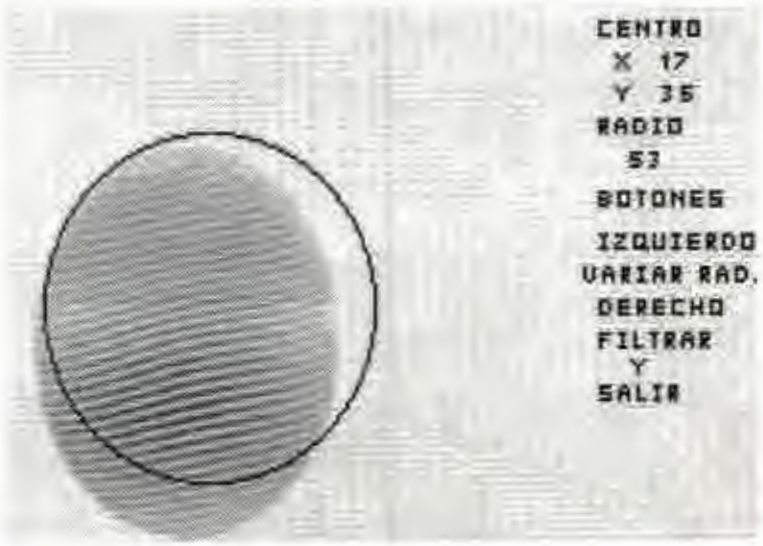


Figura 2.2 Ambiente del programa para determinar el borde de un interferograma y aplicar el filtro de Hanning (secc. I.3). En este ejemplo se está probando la superficie de un espejo plano.

[3] Las interrupciones del sistema operativo son señales de control que detienen momentáneamente la ejecución del programa principal para dar lugar a un evento en una dirección de memoria determinada.

2.4 TRANSFORMADA RAPIDA DE FOURIER

En este trabajo la transformada de Fourier es la herramienta fundamental para llevar a cabo el proceso de la **Interferometría de Fourier**. Esta transformación entre los espacios de coordenadas y de frecuencias se realiza aplicando el algoritmo de FFT (Cooley y Tukey, 1965) sobre la imagen, que a continuación describimos. Para un estudio completo de éste el libro de Brighman (1974) es una buena referencia. La transformada de Fourier de una función bidimensional f(x,y) se define:

$$F(u,v)=\mathcal{F}\{f(x,y)\}=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x)e^{-i2\pi(xu+yv)}dx\,dy \qquad (1.10)$$

Debido a que trabajamos con funciones discretas, nos interesa desarrollar la **transformada de Fourier discreta**. Para fines de analizar ésta nos restringiremos al caso unidimensional (Fig. 2.3). Anteriormente al desarrollo del algoritmo de FFT el cálculo de la transformada de Fourier se hacía realizando explícitamente la integración numérica expresada en Ec.1.10, que aún en el caso discreto involucra un número grande de operaciones, ya que se requiere realizar operaciones complejas del orden del cuadrado del número de puntos muestrados, que involucran básicamente la multiplicación matricial del vector de puntos muestreados por una matriz simétrica (ver Ec.2.13), cuyas entradas son exponenciales complejas con argumento igual a fracciones compuestas de 2π.

Debido a que las exponenciales complejas son periódicas en el argumento módulo 2π, en el caso de que el número de puntos muestreados sea no-primo permite simplificar las entradas de la matriz. Esta simplificación permite escribir a la matriz de exponenciales complejas como el producto de varias matrices donde cada una involucra solo un número de operaciones del orden del punto de números muestreados. Estas simetrías fueron usadas por Cooley y Tukey para elaborar el algoritmo de FFT, originalmente para el caso de punto muestreados igual a un múltiplo de 2, permitiendo reducir el número de operaciones complejas al orden del número de puntos muestreados.

La factorización de la matriz porduce una ligera discrapancia el cual es inherente al proceso de factorización de la matriz de exponenciales complejas. Este problema consiste en que el vector resultante con la transformada de Fourier discreta de los puntos muestreados está "revuelta", esto es, para el caso de números muestreados igual a un multiplo de 2 las entradas del vector se encuentran en orden inverso en su representación binaria (bit-reversal), el cual consiste en intercambiar el orden de las entradas de acuerdo a la representación binaria del número correspondiente a la entrada (v.g con 3 bits: entrada 6 = 110 intercambiada con la entrada 3 = 011). Actualmente ya se conoce la solución a este problema y es parte del algoritmo de FFT.

La descripción matemática del algoritmo parte de la discretización de un par transformado de Fourier (Fig.2.3a) que se realiza a partir del muestreo de la función f(x) (Ec.1.10), con una función Shah(x) con periodo **T** (Fig.2.3b y Fig.2.3c):

$$\begin{aligned} f(x)\Delta_0(x) &= f(x)\sum_{k=-\infty}^{\infty}\delta(x-kT) \\ &= \sum_{k=-\infty}^{\infty} f(kT)\ \delta(x-kT) \end{aligned} \qquad \textbf{(2.1)}$$

Notemos el fenómeno de empalme debido a la elección de **T** que ocurre en el plano de Fourier. Luego truncamos la función por medio de una función rectángulo, H(x) (Ec.2.2), cuyo resultado se muestra en la figura 2.3d.

$$H(x)=\begin{cases} 1 & -\frac{T}{2} < x \le T_0 - \frac{T}{2} \\ 0 & \textit{de otra forma} \end{cases} \qquad \textbf{(2.2)}$$

Al truncar se tienen **N** puntos equidistantes muestreados de la función f(x) (Ec.2.3). Notemos que debido a la duración finita de la función se produce un fenómeno de **ondulación** (o *rippling*) de la función en el dominio de frecuencias (Fig.2.3e):

$$f(x)\Delta_0(x)H(x) = \left[\sum_{k=-\infty}^{\infty} f(kT)\,\delta(x-kT)\right]H(x) = \sum_{k=0}^{N-1} f(kT)\,\delta(x-kT) \tag{2.3}$$

El paso final es muestrear la transformada de Fourier de esta última ecuación (Fig.2.3f), convolucionando con una función similar a Ec.2.1:

$$\Delta(x) = T_0 \sum_{r=0}^{\infty} \delta(x - rT_0) \tag{2.4}$$

El resultado final es un par transformado periódico con N puntos muestreados por periodo en ambos espacios. En el espacio de coordenadas el resultado es (Fig.2.3g):

$$\tilde{f}(x) = T_0 \sum_{r=0}^{\infty} \left[\sum_{k=0}^{N-1} f(kT)\,\delta(x - kT - rT_0)\right] \tag{2.5}$$

Aquí se indica que el resultado es aproximadamente igual a f(x). Para obtener el par transformado de esta ecuación, usamos un resultado de las series de Fourier para funciones periódicas, el cual dice que la transformada de Fourier de una función periódica está dada por:

$$\tilde{F}\left[\frac{n}{T_0}\right] = \sum_{n=-\infty}^{\infty} \alpha_n \delta(f - nf_0) \qquad f_0 = \frac{1}{T_0} \tag{2.6}$$

$$\alpha_n = \frac{1}{T_0} \int_{-\frac{T}{2}}^{T_0 - \frac{T}{2}} T_0\, \tilde{f}(x)\, e^{-i2\pi n \frac{x}{T_0}}\, dx = \sum_{l=0}^{\infty} f(lT)\, e^{-i2\pi ln\frac{T}{T_0}} \tag{2.7}$$

De manera que finalmente, la transformada de Fourier discreta de una función f(x) es:

$$\tilde{F}\left[\frac{n}{NT}\right] = \sum_{n=-\infty}^{\infty} \sum_{l=0}^{N-1} f(lT)\, e^{-i2\pi l\frac{n}{N}} \tag{2.8}$$

aquí nuevamente estamos usando el hecho de que se trata de una aproximación a la transformada de Fourier exacta. Esta Ec.2.8 relaciona N puntos en el espacio de coordenadas con N puntos en el espacio de frecuencias, a través de la transformada de Fourier continua. Si suponemos que los N puntos muestreados en la función original f(x) están en un periodo de una función periódica g(x), entonces Ec.2.8 es exacta, en cuyo caso la transformada de Fourier de f(x) (muestreada en N puntos) está dada por los N puntos calculados por Ec.2.8. Normalmente la transformada de Fourier (Ec.2.8) se escribe:

$$G\left(\frac{n}{NT}\right)=\sum_{k=0}^{N-1} g(kT)\,e^{-i2\pi\frac{nk}{N}} \qquad n=0,1,\ldots,N-1 \tag{2.9}$$

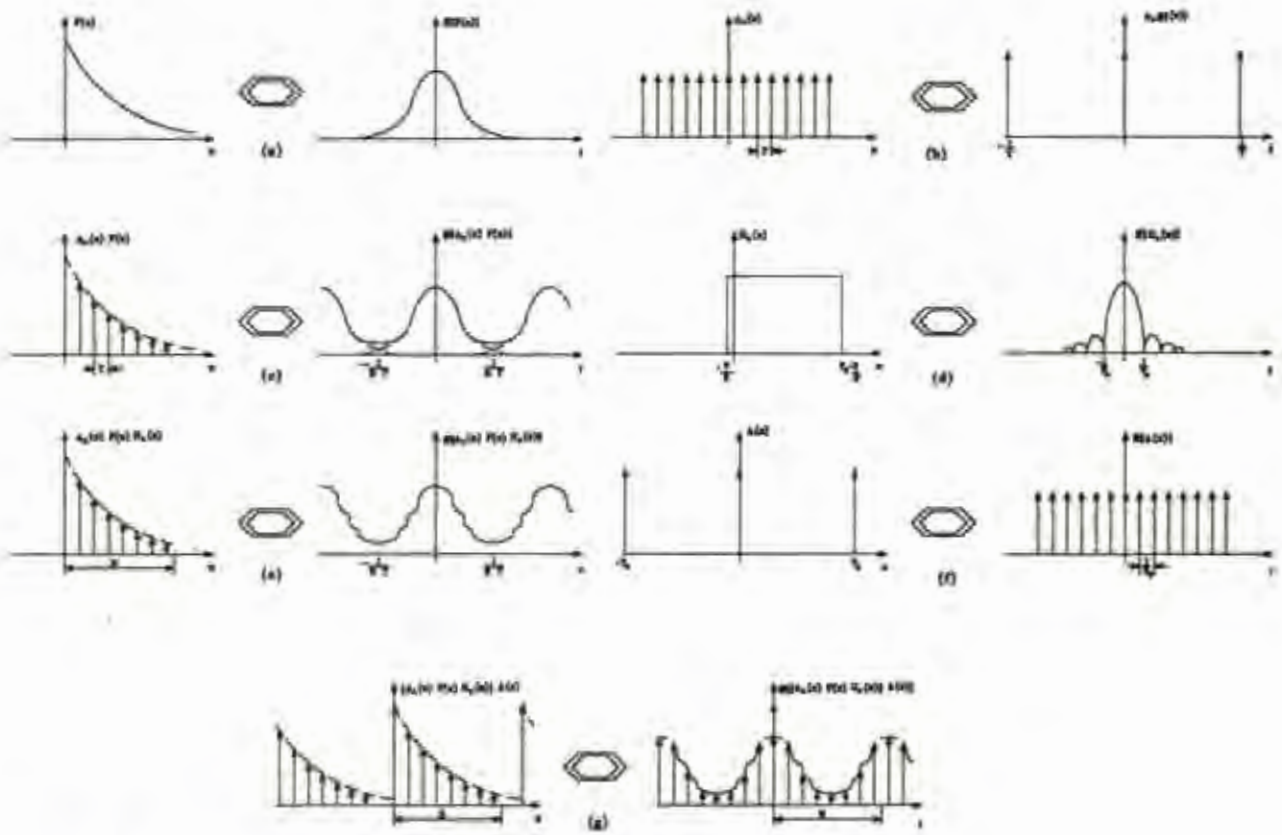


Figura 2.3 Derivación gráfica de la transformada de Fourier discreta

Calculada en forma similar a la transformada de Fourier, la **transformada discreta de Fourier inversa** está dada por:

$$g(kT)=\frac{1}{N}\sum_{n=0}^{N-1}G\left(\frac{n}{NT}\right)e^{i2\pi\frac{nk}{N}} \qquad k=0,1,\ldots,N-1 \tag{2.10}$$

Considerando la Ec.2.9, en la que por simplicidad consideramos el cambio de notación kT por k y n/NT por n, se tiene el sistema de ecuaciones:

$$X(n)=\sum_{k=0}^{N-1}x_0(k)W^{nk} \qquad n=0,1,\ldots,N-1 \tag{2.11}$$

$$W^n=e^{-i2\pi\frac{n}{N}} \tag{2.12}$$

Podemos escribir este sistema de ecuaciones en forma matricial de la forma:

$$X(n)=W^{nk}x_0(k) \tag{2.13}$$

Notemos que esta operación involucra N^2 multiplicaciones complejas y N(N-1) sumas complejas. El algoritmo de FFT permite reducir este número de operaciones. Este algoritmo deducido para arreglos con $N = 2^\gamma$ elementos, con γ entero (los cuales son los números **naturales** en computación), parte de expresar a **n** y **k** de la Ec.2.13 en forma binaria, es decir, como la suma de ceros y unos por múltiplos de 2.

$$\begin{aligned} n&=2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+\ldots+n_0 \\ k&=2^{\gamma-1}k_{\gamma-1}+2^{\gamma-2}k_{\gamma-2}+\ldots+k_0 \end{aligned} \tag{2.14}$$

Usando la Ec.2.14 reescribimos la Ec.2.13 de la forma:

$$X(n_{\gamma-1},n_{\gamma-2},\ldots,n_0)=\sum_{k_0=0}^{1}\sum_{k_1=0}^{1}\cdots\sum_{k_{\gamma-1}=0}^{1}x(k_{\gamma-1},k_{\gamma-2},\ldots,k_0)\;W^p \tag{2.15}$$

$$p=(2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+\ldots+n_0)(2^{\gamma-1}k_{\gamma-1}+2^{\gamma-2}k_{\gamma-2}+\ldots+k_0) \tag{2.16}$$

Como se cumple que $W^{a+b}=W^aW^b$, se puede reescribir W^p de la forma:

$$W^{P}=W^{(2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+\cdots+n_0)(2^{\gamma-1}k_{\gamma-1})}XW^{(2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+\cdots+n_0)(2^{\gamma-2}k_{\gamma-2})} \quad (2.17)$$
$$X\ldots W^{(2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+\cdots+n_0)(k_0)}$$

Notemos que como las representaciónes de k y n son binarias, se cumple:

$$W^{2\gamma}=W^{N}=\left[e^{-i2\frac{\pi}{N}}\right]^{N}=1 \quad (2.18)$$

entonces para el primer término de Ec.2.17 se tiene:

$$W^{(2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+\cdots+n_0)(2^{\gamma-1}k_{\gamma-1})} =\left[W^{2^{\gamma}(2^{\gamma-2}n_{\gamma-1}k_{\gamma-1})}\right]\left[W^{2^{\gamma}(2^{\gamma-3}n_{\gamma-2}k_{\gamma-1})}\right]\ldots\left[W^{2^{\gamma}(N_1k_{\gamma-1})}\right]W^{2^{\gamma}(n_0k_{\gamma-1})} \quad (2.19)$$
$$=W^{2^{\gamma}(n_0k_{\gamma-1})}$$

Este mismo efecto ocurre para los demás N-1 términos, donde solo quedan algunos elementos en cada término, entonces podemos escribir la Ec.2.15 de la forma:

$$\begin{aligned} X(n_{\gamma-1},n_{\gamma-2},\ldots,n_0) &=\sum_{k_0=1}^{1}\sum_{k_1=1}^{1}\cdots\sum_{k_{\gamma-1}=1}^{1}x(k_{\gamma-1},k_{\gamma-2},\ldots,k_0) \\ &X\ W^{2^{\gamma-1}(n_0k_{\gamma-1})}\ W^{(2n_1+n_0)2^{\gamma-2}k_{\gamma-2}} \\ &X\ W^{(2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+\cdots+n_0)k_0} \end{aligned} \quad (2.20)$$

El haber realizado esta factorización nos permite realizar las sumatorias por separado:

$$x_1(n_0,k_{\gamma-2},\ldots,k_0)=\sum_{k_{\gamma-1}=0}^{1}x_0(k_{\gamma-1},k_{\gamma-2},\ldots,k_0)W^{2^{\gamma-1}(n_0k_{\gamma-1})}$$

$$x_2(n_0,n_1,k_{\gamma-3},\ldots,k_0)=\sum_{k_{\gamma-2}=0}^{1}x_1(n_0,k_{\gamma-2},k_{\gamma-2},\ldots,k_0)W^{(2n_1+n_0)2^{\gamma-1}k_{\gamma-1}} \quad (2.21)$$

$$\vdots$$

$$x_{\gamma}(n_0,n_1,n_2,\ldots,n_{\gamma-1})=\sum_{k_0=0}^{1}x_{\gamma-1}(n_0,n_1,\ldots,k_0)W^{(2^{\gamma-1}n_{\gamma-1}+2^{\gamma-2}n_{\gamma-2}+\cdots+n_0)k_0}$$

$$X(n_{\gamma-1},n_{\gamma-2},\ldots,n_0)=x_{\gamma}(n_0,n_1,n_2,\ldots,n_{\gamma-1})$$

Este conjunto de ecuaciones constituyen el **fundamento del algoritmo de FFT** para N = 2^{τ}. Con éste el número de cálculos se reduce de N^2 a $N\gamma/2$ multiplicaciones y de N(N+1) a $N\gamma$ sumas[4]. Además se tiene la ventaja de que los cálculos se pueden realizar en el mismo espacio en memoria que el arreglo original. La última expresión de la Ec.2.21 es la clave del algoritmo, donde se produce la operación de **bit-reversal**.

Por ejemplo para $N=2^3= 8$, tomando n y k de forma binaria:

$$\begin{aligned} n &= 4n_2 + 2n_1 + n_0 \qquad n_i = 0,1 \\ k &= 4k_2 + 2k_1 + k_0 \qquad k_i = 0,1 \end{aligned} \tag{2.22}$$

con esto, la Ec.2.11 se escribe explícitamente:

$$X(n_2, n_1, n_o) = \sum_{k_2=0}^{1} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} x_0(k_2, k_1, k_o) W^{(4n_2 + 2n_1 + n_o)(4k_2 + 2k_1 + k_o)} \tag{2.23}$$

Reescribimos W^p para tener que:

$$\begin{aligned} W^{(4n_2 + 2n_1 + n_o)(4k_2 + 2k_1 + k_o)} &= \left[W^{4n_2 (4k_2 + 2k_1 + k_o) + 2n_1 (4k_2 + 2k_1 + k_o) + n_o (4k_2 + 2k_1 + k_o)}\right] \left[W^{(4n_2 + 2n_1 + n_o)(2k_1)}\right] \left[W^{(4n_2 + 2n_1 + n_o)(k_o)}\right] \\ &= \left[W^{8(2n_2 k_2)} W^{8(n_1 k_2)} W^{n_o k_2}\right] \left[W^{8(2n_2 k_1)} W^{(2n_1 + n_0)(2k_2)}\right] \left[W^{(4n_2 + 2n_1 + n_o)(k_o)}\right] \end{aligned} \tag{2.24}$$

Con lo que hemos generado un sistema de ecuaciones que representa la factorización de la matriz Ec.2.11, o equivalentemente el diagrama de flujo para N = 8 (Fig.2.4):

$$\begin{aligned} x_1(n_o, k_1, k_o) &= \sum_{k_2=0}^{1} x_0(k_2, k_1, k_o) W^{4n_o k_2} \\ x_2(n_o, n_1, k_o) &= \sum_{k_1=0}^{1} x_1(n_o, k_1, k_o) W^{(2n_1 + n_o)(2k_1)} \\ x_3(n_o, n_1, n_2) &= \sum_{k_0=0}^{1} x_1(n_o, n_1, k_o) W^{(4n_2 + 2n_1 + n_o)(k_o)} \\ X(n_2, n_1, n_o) &= x_3(n_o, n_1, n_2) \end{aligned} \tag{2.25}$$

[4]Para N = 256, el número de operaciones van de 65,536 a 384 (0.6%)

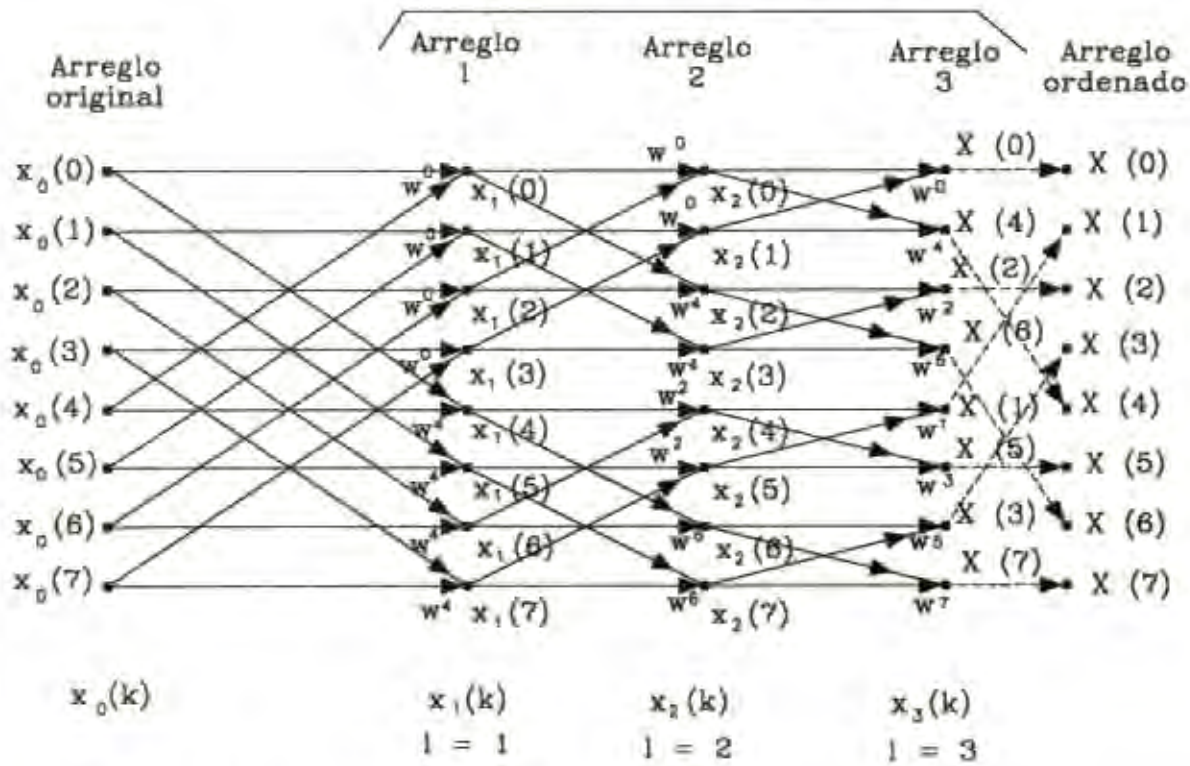


Figura 2.4 Diagrama de flujo del algoritmo de la FFT para el caso N = 8

La transformada de Fourier bidimensional de una imagen se efectua realizando la transformación que decribimos a cada renglón para después realizarla a cada (pensando a la imagen como una matriz de valores). Los algoritmos para realizar la FFT de una imagen directa e inversa se encuentran listados en las secciónes I.1 y I.2 respectivamente, donde usan como parámetros los Archivos de entrada y salida, tamaño de la imagen ($N = 2^r$) y el número de bits correspondiente r[5]. Al calcular la transformada de Fourier de un interferograma (el cual no tiene parte compleja) se incluye éste en el campo real y el campo complejo se inicializa a ceros. En la transformada inversa se requiere por cuestiones de escalamiento dividir la imagen por N^2.

[5] Debido a que en estas imágenes se trabajan dos campos (real e imaginario), el tamaño de los archivos con datos complejos es ocho veces más grande que el de las imágenes con variables de tipo caracter usadas previamente.

2.5 FILTRAJE EN EL PLANO DE FOURIER

Como se discutió en la sección 1.3, ésta es precisamente la parte esencial dentro del proceso de la interferometría de Fourier. La selección que se haga del orden uno de difracción es crítica para tener una buena reconstrucción de la DCO (Barr *et all*, 1991).

El primer problema con el que se enfrenta el filtraje en el plano de Fourier es el despliegue de la transformada, ya que el proceso de la FFT ubica el cero de la transformada en los extremos de la imagen (ver Fig.2.3g). Una exhibición típica de la imagen que se obtiene antes de la corrección se muestra en la Fig.2.5. En este despliegue se encuentran intercambiados los cuadrantes de la imagen (el primero y el tercero, y el segundo y cuarto).

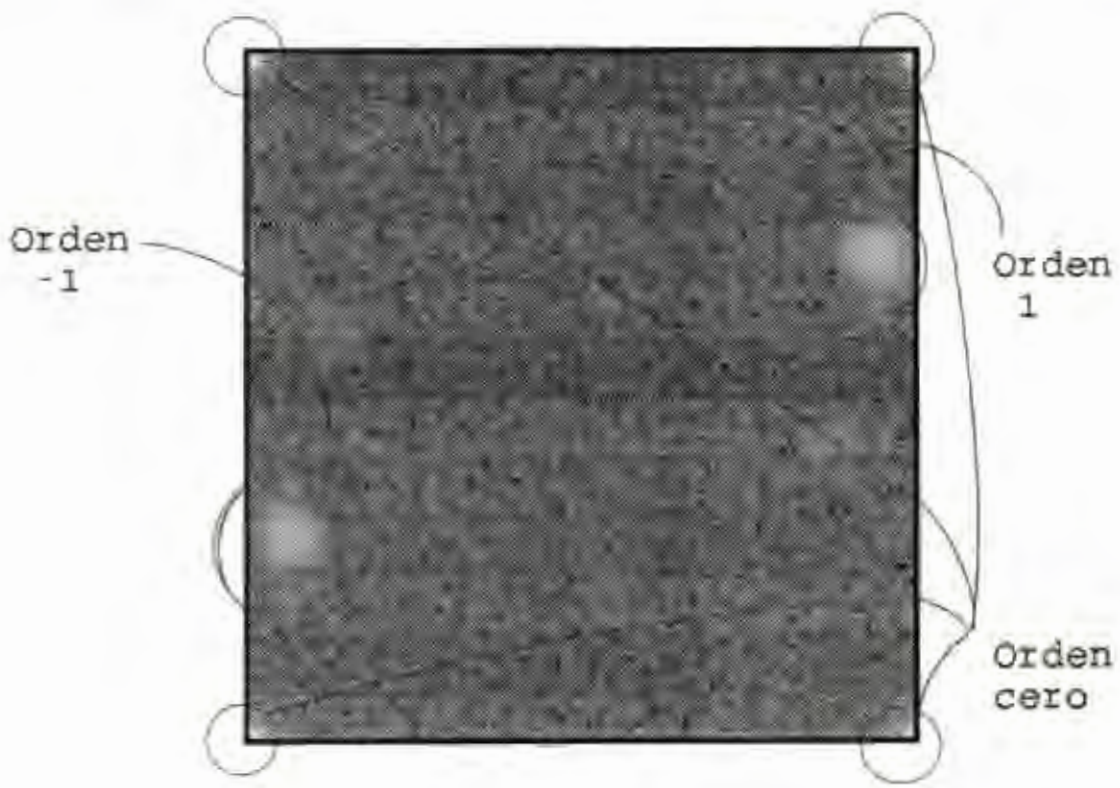


Figura 2.5 Despliegue de la imagen de una transformada de Fourier tal como se encuentra almacenada en un archivo a la salida del algoritmo de la FFT.

Para reconstruir la imagen correcta, entonces se deben intercambiar estos cuadrantes, a manera de reconstruirla correctamente. En este proceso hay que tener mucho cuidado, ya que como la imagen es simétrica (al ser la función original real) es muy fácil confundir los cuadrantes, lo cual provoca que estemos trabajando con el orden -1 en vez del orden 1.

Una imagen de la amplitud de la transformada de Fourier correctamente reconstruida se muestra en la Fig.2.6, donde se tienen perfectamente determinados los órdenes cero y uno de difracción. Notemos que el orden cero no es una función impulsiva, como se predijo en la teoría (secc. 1.3), sino que tiene un cierto ancho debido a haber multiplicado por un filtro de Hanning. La rutina usada para exhibir este tipo de imágenes se encuentra listada en la sección I.5.3, donde se puede encontrar la receta para hacer una reconstrucción adecuada.

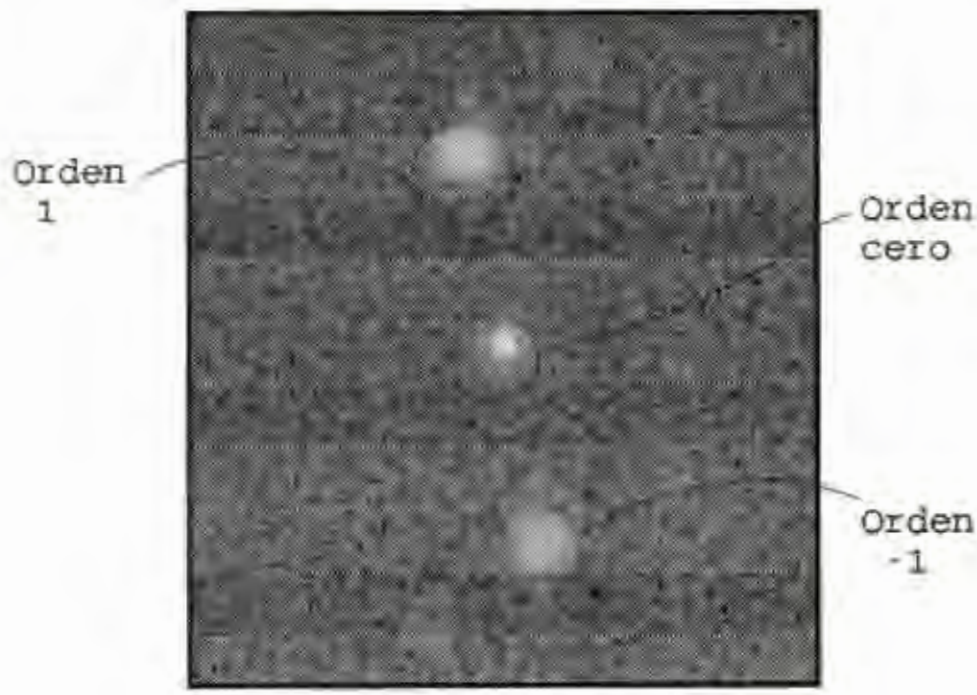


Figura 2.6 Transformada de Fourier correctamente exhibida. Notemos que debido a que se trata de una función real, la imagen es completamente simétrica.

Una vez que se tiene la imagen de la transformada de Fourier exhibida correctamente, se procede a seleccionar el orden **uno** de difracción, el cual debido a que no existen frecuencias negativas en la naturaleza, se encuentra necesariamente en la mitad superior derecha del plano de Fourier. Esto se puede hacer usando diferentes criterios. Una técnica muy usada es seleccionar automáticamente el orden uno a partir de la frecuencia de las franjas, conociendo así la frecuencia portadora (Kujawinska, 1989). Un segundo método automático consiste en determinar un umbral a partir del máximo secundario en el orden uno, estableciendo un contorno y el centroide de la mancha que queda por encima de este umbral (Frankowsky, 1989). Estos métodos automáticos son muy dependientes de tener una frecuencia portadora lo suficientemente grande para separar los órdenes cero y uno.

Debido a que en general el comportamiento de los interferogramas no es regular, se decidió hacer una selección manual del orden uno de difracción. Para esto se usó el ratón para mover un círculo que selecciona el orden uno de difracción, luego se realiza una traslación de ésta al centro del plano de frecuencias y el resto del plano se inicializa a cero (campo obscuro).

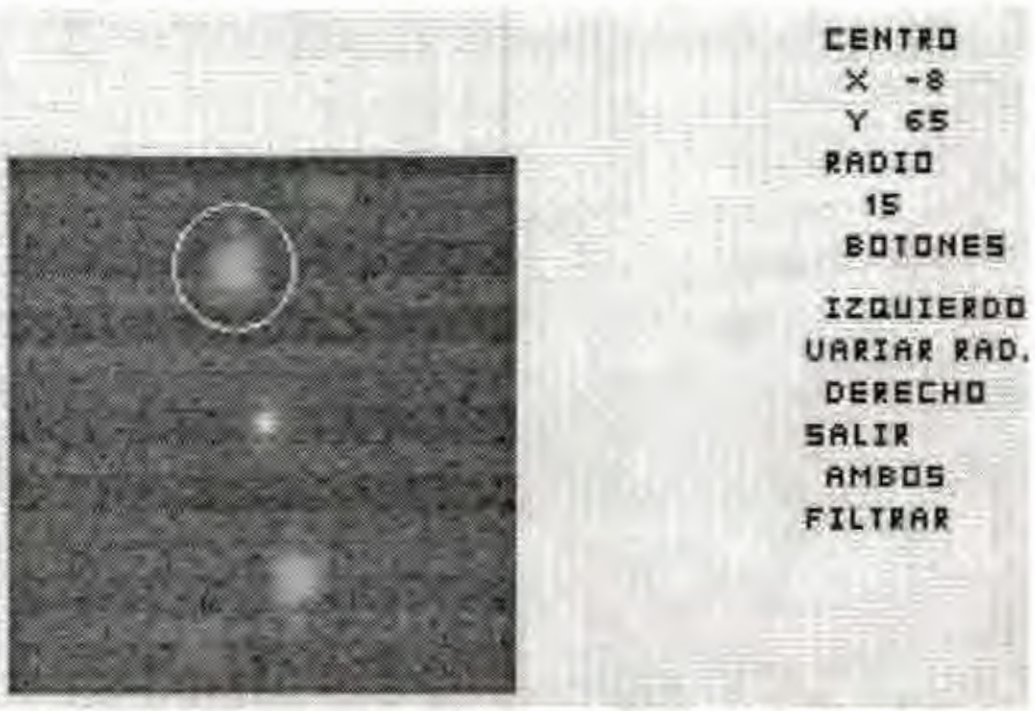


Figura 2.7 Ambiente del programa usado para seleccionar el orden uno de difracción en el plano de Fourier. En este caso se está probando la misma superficie que en la Fig.2.2.

La segunda parte de este problema es más complicada, en la cual debemos **desdoblar la fase** de la OPD, para la cual en principio sólo se conocen los valores de $-\pi$ a π. Si se realizó un buen filtraje en el paso anterior, la fase "doblada" casi no presentará problemas de dislocaciones (ver Fig.1.7), pero si este no fué el caso, el problema de dislocaciones es prácticamente imposible de resolver, que como se mencionó, es la **principal** fuente de error en la técnica de interferometría de Fourier.

Para desdoblar la fase se tomó como base la técnica desarrollada por Takeda (1982). Se calcula el desdoblamiento de fase para la columna central de la imagen con la técnica descrita en la sección 1.3, y usando esta información como base se realiza el desdoblamiento de fase para todos los renglones de la imagen. Repetimos lo mismo tomando como base el renglón central de la imagen y usamos esta información para reconstruir la información por columnas, de esta forma se han generado dos matrices con la información de la fase desdoblada. En el caso de tener imágenes sin el problema de dislocación, ambas matrices deben ser iguales, en caso contrario existe una dislocación en la reconstrucción de la fase.

En el algoritmo usado para la reconstrucción de la fase (sección I.8) se escoge que en el pixel central la corrección a la DCO sea cero y a partir de este punto se haga la reconstrucción de todo el frente de onda. Si las matrices de reconstrucción son iguales por columnas que por renglones, se considera que se tiene una reconstrucción correcta, en caso contrario se asigna un valor de cero a ese punto de la matriz. Notemos que de esta forma la información en ese pixel se pierde[8].

Con la combinación de algoritmos de filtraje en el plano de Fourier y de reconstrucción que desarrollamos, si la DCO reconstruida tiene varias dislocaciones el resultado final no es aceptable, por lo que conviene mejor realizar un nuevo filtraje del orden uno de difracción y una nueva reconstrucción de fase, hasta lograr una DCO correctamente reconstruida.

---

[8]Para una mejor reconstrucción habría que calcular el valor en el pixel de manera que la suma de cualesquiera cuatro pixeles adyacentes fuera cero (Barr, 1991).

La traslación de la zona de interés se realiza de manera que el centro del círculo coincide con el centro del plano de frecuencias, moviendo la información dentro del círculo sin alterar e información adicional en un pequeño anillo alrededor del círculo el cual se apodiza rápidamente con una función coseno (Barr, 1991) para evitar fenómenos de empalme.

En esta traslación se debe tener cuidado, ya que como se mencionó arriba, el centro del plano de frecuencias se encuentra distribuido en los extremos del archivo con la información del plano de Fourier. Cabe hacer notar que la selección que aquí se realiza es muy crítica ya que un error de unos cuantos pixeles puede dar lugar a un número grande de dislocaciones en la fase de la transformada inversa. El algoritmo usado para realizar este filtraje se encuentra en la sección I.7, que genera el ambiente mostrado en la figura 2.7, muestra cómo se realizan filtrajes en el plano de Fourier cuando existe un movimiento de la información (no solo se filtra), con lo cual se pierden las simetrías relacionadas con imágenes reales puras[6].

2.6 CALCULO Y DESDOBLAMIENTO DE FASE

Una vez que se ha filtrado la transformada de Fourier de un interferograma, se calcula su transformada de Fourier inversa, que como se vió en la sección 1.3., contiene en la amplitud la información de la visibilidad de las franjas y en la fase la DCO que buscamos (sección 1.4.3).

Para la primera parte de este problema se usa el archivo que contiene la transformada inversa del filtraje del orden uno. Calculando el arcotangente (con el comando ATAN2 del lenguaje **C**) del cociente de la parte imaginaria entre la parte real para un pixel, lo que da valores en la rama principal (de $-\pi$ a π) que son guardados en un archivo[7] (secc. I.8). Podemos exhibir este tipo de imágenes usando el algoritmo que se encuentra en la sección I.5.1.

---

[6] Los filtrajes estáticos en el plano de Fourier son los más comunes en el procesado digital de imágenes.

[7] Los archivos que se trabajan en esta parte usan números flotantes, lo cual genera archivos cuatro veces más grandes que los usados en las primeras etapas del proceso.

El resultado de la reconstrucción de fase tiene la forma que se observa en la Fig.2.8. Este es el tipo de imágenes que se obtienen al analizar interferogramas estáticos mediante la técnica de interferometria de Fourier.

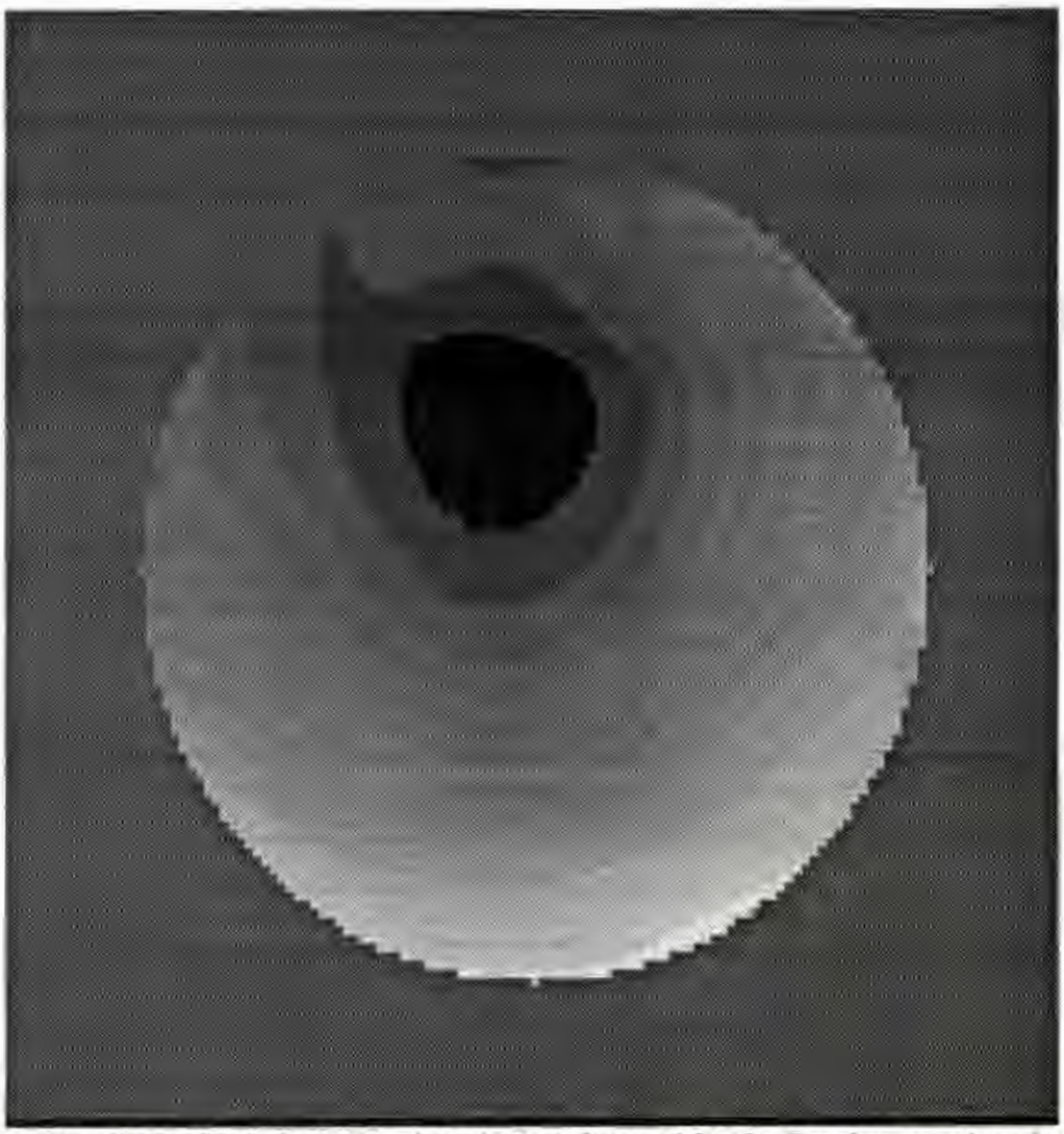


Figura 2.8 Resultado final de la técnica de **interferometría de Fourier**, mostrando la DCO obtenida. Este ejemplo es el resultado de analizar el interferograma mostrado en la Fig.2.2.

2.7 CALCULO DE LOS COEFICIENTES DE ZERNIKE Y SEIDEL

Una vez que se ha obtenido la DCO asociada a un interferograma estático mediante la técnica de interferometría de Fourier, el paso final es evaluar el frente de onda. Como se mencionó en la sección 1.5, esta evaluación se puede realizar en forma cualitativa o cuantitativa.

En forma cualitativa se realiza el estudio topográfico de la DCO, observando en el resultado final las deformaciones que introduce la superficie o sistema óptico bajo prueba (ver Fig.2.8). Esta forma de análisis es la más fácil de interpretar debido a que es un resultado gráfico que muestra los picos y valles de la superficie o sistema óptico bajo prueba, y así mismo es útil al realizar pruebas en un taller óptico, ya que el mismo mapa topográfico que se obtiene sirve como mapa de trabajo de la superficie o sistema óptico que se esté generando (Fig.2.8).

De los diferentes parámetros cuantitativos que se pueden trabajar, el más simple de éstos es el valor pico-valle (P-V), el cual es la diferencia máxima en el frente de onda obtenido. Este parámetro, junto con la información cualitativa de la DCO son la información mínima necesaria para tener una estimación del frente de onda bajo prueba.

De entre los parámetros cuantitativos que se pueden obtener además del valor P-V, los coeficientes de Zernike o alternativamente los coeficientes de Seidel son los parámetros cuantitativos más comúnmente usados debido a las características típicas en los sistemas ópticos que se encuentran asociadas a las diferentes aberraciones.

Los coeficientes de Zernike pueden ser determinados a partir de la información de la imagen que se obtiene como resultado del análisis mediante la técnica de interferometría de Fourier, usando las herramientas desarrolladas en la sección 1.5. Para llevar a cabo ésto se realizan dos pasos.

El primero de éstos consiste en efectuar un cambio de coordenadas de cartesianas a polares, ya que la integral a evaluar (Ec.1.22) es separable en coordenadas polares. Esto se realiza usando el algoritmo mostrado en la sección I.10, donde para tener un menor error en la transformación de coordenadas, se busca el pixel correspondiente en coordenadas cartesianas a uno en coordenadas polares, cubriendo así todo el plano polar.

Ya con la imagen en coordenadas polares se realiza la integración numérica de la DCO multiplicada por el polinomio de Zernike del coeficiente que se desea conocer. Esta se realiza sumando para cada renglón (el cual contiene la información en la coordenada radial) el producto del valor de la DCO en cada pixel por la mitad de la suma del polinomio de Zernike correspondiente evaluado en los extremos del pixel. Ya con la integración angular, se realiza en forma similar la integración radial de la columna con la información de la integración angular. Finalmente el valor de esta integral doble se normaliza de acuerdo a la Ec.1.22, teniendo de esta forma el coeficiente de Zernike correspondiente al polinomio de Zernike que se usó en el cálculo.

El algoritmo para realizar esta integración se muestra en la sección I.11, donde se usa la noción de programación por objetos OOP (Object Oriented Programing) al usar como variables las funciones radiales y angulares de los polinomios de Zernike (Tabla 1.1). Se calculan los primeros ocho coeficientes de Zernike y finalmente usando la información de la Tabla 1.2 se calculan los coeficientes de Seidel asociados al frente de onda bajo prueba.

Este tipo de análisis representa una nueva alternativa para el estudio de sistemas ópticos, válida en general para las técnicas de interferometría heterodina, en particular para el de análisis de interferogramas estáticos mediante la técnica de frecuencia espacial portadora: la interferometría de Fourier.

CAPITULO 3

RESULTADOS DE LA INTERFEROMETRIA DE FOURIER

3.1 OBSERVACIONES A LA TECNICA

La técnica de interferometría de Fourier que hemos estudiado en los capítulos anteriores presenta características que la hacen una alternativa a los métodos tradicionales de análisis de interferogramas. La ventaja principal que presenta ésta técnica es que recupera con una precisión alta la información de la DCO introducida por el sistema óptico bajo prueba a partir de un *interferograma estático*, eliminando el problema clásico de la interferometría: el dilema cóncavo-convexo.

Usando cualquier técnicas de interferometría heterodina se descarta este dilema, pero mientras que las técnicas de FTP requieren alta estabilidad al hacer las pruebas y de dispositivos para correr la fase, las técnicas de FEP, en particular la interferometría de Fourier, solo requiere de un interfereograma estático[1]. Este es el resultado parece contradictorio con los conceptos de la interferometriá tradicional, ya que en ésta de un interferograma estático es imposible saber el signo de la DCO si no se cuenta con algúna información adicional (v.g. el orden de franja), pero recalquemos que existe información adicional que se esta usando en el análisis mediante la interferometría de Fourier: una codificación espacial de varios interferogramas al introducir una frecuencia espacial portadora. Esto lo podemos observar en la Fig.3.1, donde se tienen interferogramas simulado con aberración de coma de Seidel (B = ± 3, Ec.1.23). En la Fig.3.1A se muestra el interferograma sin frecuencia portadora, en el cual no se conoce el signo de la DCO (ambos interferogramas son iguales). En las Fig.3.1B y Fig.3.1C se incluyo una frecuencia portadora (E=F=6, ambos positivos), que permite diferenciar entre coma positiva (Fig.3.1B) y coma negativa (Fig.3.1C) (ver Malacara, 1991, p 80).

---

[1] Esto es actualmente posible gracias al desarrollo de detectores con respuesta uniforme y resolución espacial alta.

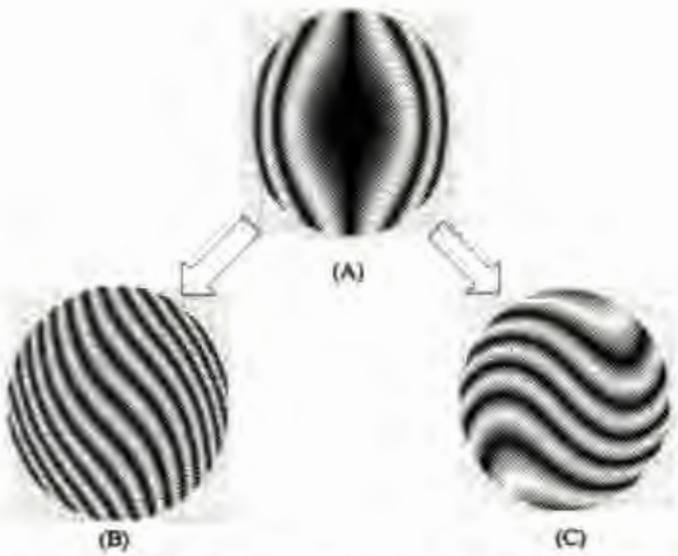


Figura 3.1 (A) Interferograma con indeterminación del signo de la DCO (B) Interferograma con DCO positiva. (C) Interferograma con DCO negativa.

Al introducir la frecuencia portadora se debe cuidar que ésta sea lo suficientemente grande para diferenciar entre los ordenes cero y uno, pero manteniendola suficientemente pequeña para no llegar al límite de Nyquist. Si se sobrepasa el límite de Nyquist aparecen franjas fantasmas en el interferograma y se genera un *efecto de cilindro* en la transformada de Fourier (los ordenes de difracción parecen girar en el plano frecuencias como si éste fuera un cilindro). Un ejemplo de este efecto lo vemos en la Fig.3.2B, donde al interferograma simulado de la Fig.3.1A se le dió una frecuencia portadora grande (E= 6, F= 36).

El valor óptimo de la frecuencia portadora que se debe introducir para realizar el análisis del interferograma depende principalmente del detector que use para capturar las imágenes, sin embargo se debe tomar en cuenta que si se realiza una reducción de la imagen el valor de las deformaciones de la DCO se mantienen constante, pero el límite de Nyquist se reduce en la razón que se haga la reducción[2].

[2]Este efecto se debe considerar si se varia la resolución de la imágenes al trabajar con los algoritmos desarrollados en este trabajo.

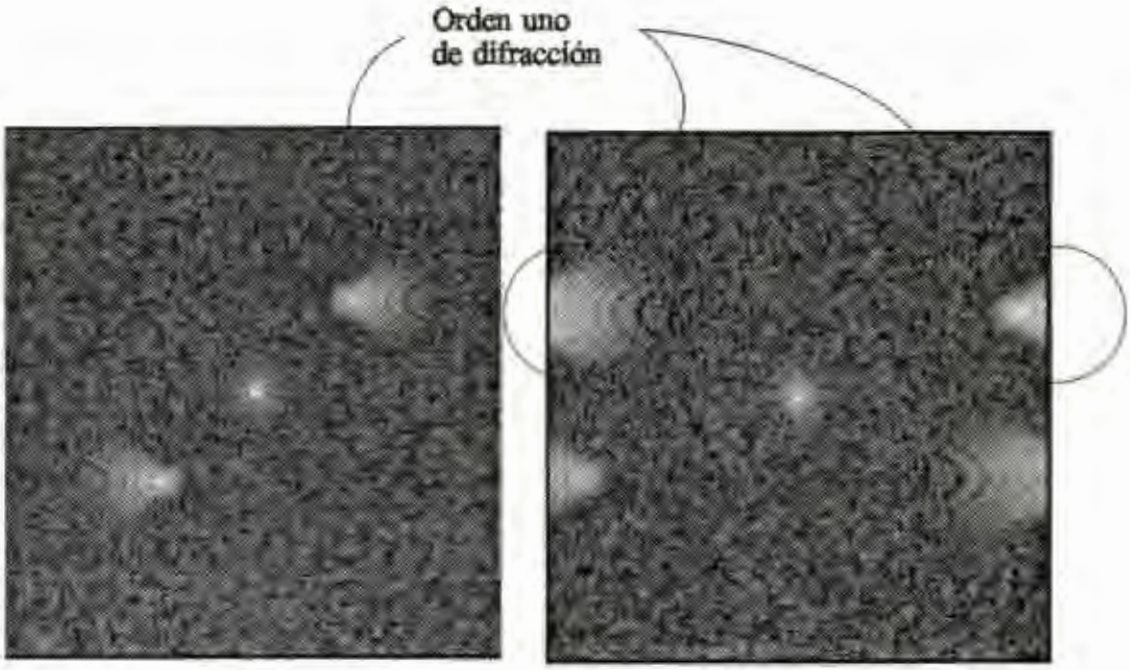


Figura 3.2 Efecto de introducir una frecuencia portadora por encima del límite de Nyquist. (A) Interferograma normal. (B) Interferograma con frecuencia superior al límite de Nyquist.

El segundo parámetro del cual depende el límite de Nyquist es la resolución de la tarjeta digitalizadora con la que se capturen las imágenes a una computadora. En nustro caso trabajamos con una tarjeta digitalizadora **Data traslation** DT2851, con una resolución de 512 por 480 pixeles con 64 niveles de gris, donde el límite de Nyquist es de aproximadamente 240 franjas en todo el campo.

Existe un último parámetro que se debe tomar en cuenta para estimar el límite de Nyquist de los interferogramas a trabajar. En los algoritmos que se desarrollaron para llevar a cabo la técnica de interferometría de Fourier (apéndice 1) se trabajan diferentes resoluciones, las cuales son de 64 por 64, 128 por 128 o 256 por 256 pixeles[3]

[3] Por limitaciones de memoria en las computadoras 256 x 256 pixeles es la resolución máxima posible. La resolución mínima para obtener información coherente de los interferogramas es de 64 x 64 pixeles.

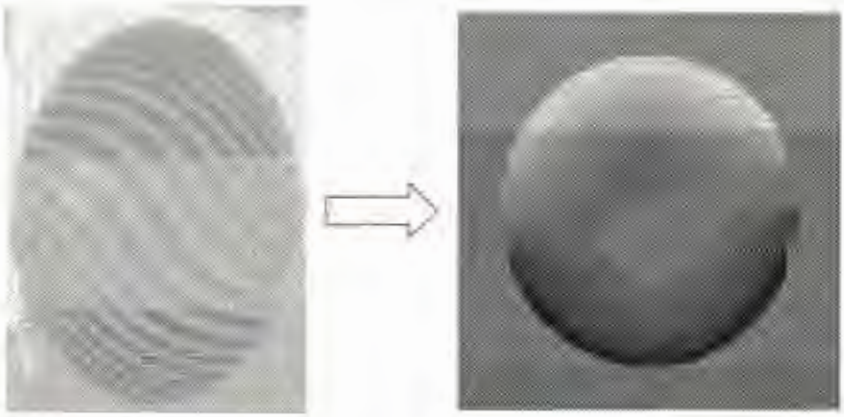
Considerando esto el límite de Nyquist efectivo de los interferogramas con los que trabajamos varía entre 32 y 128 franjas en todo el campo, lo cual está en relación directa con la frecuencia portadora que se requiere para realizar el análisis de interferogramas.

Observemos que la combinación de filtraje en el plano de Fourier (sección 2.5) y desdoblamineto de fase (sección 2.6) desarrollados en este trabajo funcionan solo para interferogramas presentando problemas al desdoblar la fase en presencia de dislocaciones, donde se requieren en ocasiones varios intentos de esta combinación de operaciones para obtener un buen resultado. Para mejorar el proceso hay que modificar estos algoritmos a manera de realizar un filtraje en el plano de frecuencias mediante la selección de un umbral del máximo del orden uno y eliminar las dislocaciones residuales pidiendo que la suma de la fase de cualesquiera cuatro pixeles adyacentes sea cero.

3.2 RESULTADOS CUALITATIVOS

De de los resultados cualitativos que se obtienen del análisis de interferogramas mediante la técnica de interferometría de Fourier, la reconstrucción de la DCO es el que ofrece mayor información en el análisis de un sistema óptico bajo prueba, ya que geométricamente muestra las deformaciones de la DCO. Un ejemplo de las diferentes etapas por las que se pasa en al analizar un interferograma se muestra en el capítulo 2, cuyo resultado final (Fig.2.8) muestra la DCO resultante de analizar el interferograma relacionado (Fig.2.2).

Otro ejemplo de interferogramas analizados con esta técnica se muestra en la Fig.3.3, en el cual se capturó por medio de la tarjeta digitalizadora un interferogramas de un libro (Malacara, 1991, p 80) con el patrón típico de la aberración de coma. Este interferograma simulado fué generados en forma similar a lo descrito en la sección 2.2, con la diferencia de que en este caso se analiza una fotografía. Los resultados al analizar los interferogramas muestran el patrón típico de la DCO de la aberración de coma (Malacara, 1991, p 457).



(A) (B)

Figura 3.3 Analisis de un interferograma simulado con coma de Seidel (A). (B) DCO resultante.

Este tipo de resultados presentan una información gráfica clara de la DCO, la cual está directamente relacionada con la forma de las superficies de la superficie o sistema óptico bajo prueba, constituyen una herramienta de trabajo en los lugares donde se generan las superficies ópticas (un taller óptico). Existen otro tipo de pruebas que se pueden llevar a cabo en estos talleres (pruebas por contacto, Ronchigramas) las cuales son simples de implementar, pero su precisión es baja. Es en las etapas finales del trabajo, donde se requiere una precisión mayor, donde conviene usar el trabajo que hemos desarrollado, el cual presenta resultados en tiempos cortos[4]. Para usar esta técnica se requiere de una computadora con una tarjeta digitalizadora, la cual no necesita estar en el área de pruebas (donde en general las condiciones de trabajo no son adecuadas para sistemas de cómpto) ya que se puede transportar la imagen de las franjas de interferencia grabadas en un videocassete. Debido a que los algoritmos desarrollados son independientes de una tarjeta digitalizadora, se puede realizar la capturar en una computadora y realizar el análisis en otra (que tenga una tarjeta de video VGA).

---

[4] En una computadora 386 con procesador matemático, los tiempos de cálculo varían de aproximadamente 1 min. para la resolución de 64 por 64 pixeles a 4 min. para la resolución de 256 por 256 pixeles.

3.3 COEFICIENTES DE ZERNIKE

Según se mencionó en la sección 2.7, es posible calcular directamente los coeficientes de los polinomios radiales de Zernike a partir de la DCO obtenida a partir de un interferograma usando la técnica de interferometría de Fourier. Este resultado abre una nueva posibildad de análisis de interferogramas, ya que se calculan los coeficientes de Zernike usanda toda la información de la DCO en vez de los métodos usados hasta ahora donde se realiza una interpolación de dichos coeficientes a partir de puntos particularmente escogidos.

Se calcularon los coeficientes de Zernike y a partir de éstos los coeficientes de Seidel, para 144 distintos interferogramas simulados a fin de observar la precisión que se obtiene en el cálculo de los coeficientes. La comparación se hizo entre el valor que se dió de entrada para cada caso (al simular el interferograma) y el valor obtenido al final del análisis de cada interferograma.

El proceso consistió en simular los interferogramas con valores fijos de las aberraciones de Seidel primarias (solas y combinadas), variando el valor de la frecuencia portadora y realizando diferentes filtrajes del orden uno de difracción. Este cálculo se realizó para las tres diferentes resoluciones que se usaron en el desarrollo de los algoritmos: 64 por 64, 128 por 128 y 256 por 256 pixeles. En estos cálculos se obtuvo un valor para la precisión en la estimación de una aberración particular (esférica, coma o astigmatismo de tercer orden) y otro para el valor residuales de la aberraciones (valor que aparece pese a que no se incluyó la aberración en la simulación). Notemos que el cálculo de los coeficientes de Seidel a partir de los coeficientes de los de Zernike (Tabla 1.2) es por sí una aproximación ya que se desprecian las contribuciones de polinomios de orden mayor, que en general tienen valores pequeños. Los resultados promedio de estas observaciones se muestran en la Tabla 3.1.

TABLA 3.1

PRECISION EN EL CALCULO DE LOS COEFICIENTES DE SEIDEL

RESOLUCION (pixeles)	64 por 64	128 por 128	256 por 256
Esférica	31.8 %	10.7 %	4.8 %
Coma	20.4 %	10.8 %	4.0 %
Astigmatismo	11.8 %	4.8 %	2.5 %
Esférica residual	7.7 %	4.0 %	3.5 %
Coma residual	11.7 %	2.8 %	2.4 %
Astigmatismo residual	9.3 %	4.1 %	3.1 %

De esta tabla y junto con el comentario de los tiempos de cálculo de la técnica interferometría de Fourier (pie de página 4), observamos que a mayor precisión se requiere de un mayor tiempo de cálculo y viceversa. La resolución con la que se desee trabajar el análisis depende principalmente de la comunión entre precisión y el tiempo que se esté dispuesto a invertir. En cualquiera de los casos, la información cualitativa que se obtiene es suficientemente buena.

Cabe mencionar que en el caso del análisis mostrado en la Fig.3.6, donde se trata de aberración de coma de Seidel, el análisis que se hizo de este interferograma reportó solamente esta aberración (C = 3.40), con aberraciones residuales de esférica y astigmatismo pequeñas(A = .20, B = .15). Este valor para el coeficiente de coma no corresponde al usado para generarla (C = 5), pero se debe considerar que existe un factor de amplificación en la fotografía.

Para interferogramas reales se estudiaron dos superficies, cuyos resultados se comparan con los obtenidos por un interferómetro Fizeau laser comercial de la compañia Wyko. Los interferogramas estáticos que analizamos fueron capturados en este interferómetro. Las superficies bajo prueba son planos generados en la compañia Zeiss de México. El primer caso fué el ejemplo mostrado a lo largo del capítulo 2. Esta compraración se muestra en la Tabla 3.2.

TABLA 3.2

COMPARACION DE VALORES DE COEFICIENTES DE SEIDEL

	64 por 64	128 por 128	256 por 256	Wyko
Primera superficie				
Esférica	-1.07 λ	- 0.76 λ	- 1.36 λ	- 1.137 λ
Coma	0.46 λ	0.60 λ	0.57 λ	0.402 λ
Astigmatismo	0.26 λ	0.25 λ	0.28 λ	0.231 λ
Segunda superfice				
Esférica	-0.80 λ	- 0.80 λ	- 0.95 λ	- 0.888 λ
Coma	0.77 λ	0.68 λ	0.40 λ	0.266 λ
Astigmatismo	0.12 λ	0.12 λ	0.14 λ	0.176 λ

En esta tabla podemos observar la similitud que existe entre los resultados usando el método desarrollado en este trabajo y uno de los instrumentos más precisos que se consigue en el mercado.

El trabajo que aquí desarrollamos constituye la primera aproximación a este tipo de cálculos, obteniendose resultados alentadores. Para mejorar el cálculo de los coeficientes se debe trabajar a futuro principalmente en los algoritmos de transformación de coordenadas cartesianas a polares y en la intergración numérica de la DCO discreta que obtenemos como resultado del análisis de un interferograma.

3.4 CONCLUSIONES

Gracias a la alta precisión que se logra en la determinación de la forma de una superficie, la interferometría heterodina es el futuro de las pruebas interferométricas comerciales. Entre las técnicas de interferometría heterodina, la de frecuencia espacial portadora promete mayores alcances, gracias al desarrollo de dispositivos con respuesta uniforme y gran resolución. En particular la interferometría de Fourier contiene los elementos que le permitirá ser una de las técnicas más usadas en el futuro.

Esta técnica basada en el algoritmo de la transformada rápida de Fourier, permite conocer sin ambigüedades la forma del frente de onda asociado a un interferograma debido a la codificación espacial de múltiples interferogramas en uno, además de que el interferograma que se analiza es estático lo que representa una ventaja sobre las técnicas de frecuencia temporal portadora que requieren alta estabilidad y de dispositivos para correr la fase al hacer las pruebas. Esto hace que ésta técnica de análisis de interferograma sea económica.

En el presente trabajo se elaboraron las herramientas para desarrollar este tipo de análisis de interferogramas usando técnicas de procesamiento digital de imágenes, en particular los algoritmos para filtrar el orden uno de difracción y realizar el desdoblamiento de fase, que junto con el algoritmo de FFT constituyen el fundamento del método estudiado. Este análisis se desarrolla a partir de un interferograma digitalizado en forma independiente de una tarjeta digitalizadora lo que le permite ser usado en cualquier tipo de computadora, con la única limitante de que tenga una tarjeta de video VGA.

El cálculo de los coeficientes de Zernike a partir del frente de onda y el polinomio de Zernike correspondiente sin necesidad de realizar interpolaciones. Este es el resultado más importante del presente trabajo, el cual es una opción diferente a los métodos actuales de realizar este cálculo.

BIBLIOGRAFIA

Barr L.D., Coúde du Foresto V., Fox V., Poczulp G.A., Roddier C. y Roddier F., "Large-mirror testing facility at the National Optical Astronomy Observatories", *Opt.Eng.*, **30,** 1405 (1991).

Bone D.J., Bachor H.A. y Sandeman R.J., "Fringe-pattern analysis using a 2-D Fourier transform", *Appl.Opt.*, **25**, 1653 (1986).

Brigham E.O. *The Fast Fourier Transform*, Prentice-Hall (1974).

Cooley J.W. y Tukey J.W., "An algoritm for machine calculation of complex Fourier transform", *Math.Computation*, **19**, 297 (1965).

Forbes G.W., "Optical system assessment for design: numerical ray tracing in the Gaussian pupil", *J.Opt.Soc.Am.A*, **5**, 1943 (1988).

Françon M., *Optical interferometry*, Academic Press (1966).

Frankowsky G., Stobbe I., Tischer W. y Schillke F., "Investigation of surface shapes using carrier frecuency based analysing system", *SPIE*, **1121**, 89 (1989).

Ghilia D.C., Mastin G.A. y Romero L.A., "Cellular automata method for phase ungrapping", *J.Opt.Soc.Am.A*, **4**, 267 (1987).

Gerchberg R.W., "Super-resolution through error energy reduction", *Opt.Acta*, **21**, 709 (1974).

Greivenkamp, J.E., "Sub-Nyquist Interferometry", *Appl.Opt.*, **26**, 5245, (1987).

Hecht E. y Zajac A., *Optica*, Adison Wesley (1986).

Hern d. y Baker P., *Gráficas por computadora*, Prentice Hall, pp 73-75, (1988).

Hudgin R.H., "Wave-front reconstruction for compensated imagining", *J.Opt.Soc.Am.*, **67**, 375 (1977).

Hunt B.R., "Matrix formulation of the reconstruction of phase values from phase differences", *J.Opt.Soc.Am.*, **69**, 393 (1979).

Ichioka Y. y Inuiya M., "Direct phase detecting system", *Appl.Opt.*, **11**, 1507 (1972).

Itho K., "Analysis of phase ungrapping algorithm", *Appl.Opt.*, **21**, 2470 (1982).

Itoh K. y Ohtsuka Y., "Phase estimation based on the maximum likehood criterion", *Appl.Opt.*, **22**, 3054 (1983).

Kujawinska M., Spik A. y Wójciak J., "Fringe pattern analysis using Fourier transform technics", *SPIE*, **1121**, 130 (1989).

Macy W.W.Jr., "Two-dimensional fringe-pattern analysis", *Appl.Opt.*, **22**, 3898 (1983)

Malacara D., *Optical Shop Testing*, Wiley, 1978.

Mertz L., "Real-time fringe-pattern analysis", *Appl.Opt.*, **22**, 1535 (1983).

Nugent K.A., "Interferogram analysis using accurate fully automatic algorithm", *Appl.Opt.*, **24**, 3101 (1985).

Roddier C. y Roddier F., "Interferogram analysis using Fourier transform techniques", *Appl.Opt.*, **26**, 1668 (1987).

Roddier F. y Roddier C., "Wavefront reconstruction using iterative Fourier transforms", *Appl.Opt*, **11**, 1325 (1991).

Steel W.H., *Interferometry*, Cambridge Univesity Press (1983).

Takeda M., Ina H. y Kobayashi S., "Fourier-transform method of fringe-pattern analysis for computed-based topography and interferometry", *J.Opt.Soc.Am.*, **72**, 156 (1982).

Takeda M., "Spatial carrier heterodyne techniques for precision interferometry and profilometry: An overview", *SPIE*, **1121**, 73 (1989).

Womack K.H., "Interferometric phase measurement using spatial synchronous detection", *Opt.Eng.*, **23**, 391 (1984).

Wyant J.C. y Creath K. *Aberrations*, Notas curso corto, WYCO Corp. (1988).

APENDICE I

En este apéndice se listan los algoritmos usados en el desarrollo de la tesis. En todos los casos se usa NMAX=256, que se puede cambiar a cualquier múltiplo de 2 (64, 128, ...).

I.1 TRANSFORMADA RAPIDA DE FOURIER (Four.c)

Para usar con imagenes cuadradas de tamaño 2^m cambiar BIT=m y NMAX=2^m (64,128,256,...)

```
#include <alloc.h>
#include <dos.h>
#include <stdio.h>
#include <math.h>
#define NMAX  256
#define BIT   8
struct compleja
 {float re;
  float im;}
void tabla(struct compleja *p[],int m,int n)
 {register int i;
  for (i=0;i<m;i++)
   if((p[i]=(struct compleja*) farcalloc
              (n,sizeof(struct compleja)))==0)
  {fprintf(stderr,"No hay espacio en memoria \n");
   exit(1);}}

void main(int argc, char *argv[])
{ struct compleja a[NMAX],*block[NMAX];
  unsigned char datos[NMAX];
  register int i,j;
  FILE *fp,*fp1;
 if(argc!=3)
  {fprintf(stderr,"USO:FourArch-imagenArch-trans\n");
    exit(1);}
 if((fp=fopen(argv[argc-2],"rb"))==NULL)
 {fprintf(stderr,"No existe %s \n",  argv[argc-2]);
   exit(1);}
  tabla(block,NMAX,NMAX);
  fp1=fopen(argv[argc-1],"wb+");
  printf("0\n");  /* Transformada por renglones */
  for (j=0;j<NMAX;j++)
  {fread(datos,1,NMAX,fp);
   for (i=0;i<NMAX;i++)
    {block[j][i].re=datos[i];
     block[j][i].im=0.0;}
   fft(block[j],BIT,NMAX);}
  printf("1\n"); /* Transformada por columnas */
  for (j=0;j<NMAX;j++)
  {for(i=0;i<NMAX;i++)
     a[i]=block[i][j];
   fft(a,BIT,NMAX);
    for(i=0;i<NMAX;i++)
     block[i][j]=a[i];}
  for(j=0;j<NMAX;j++)
   fwrite(block[j],8,NMAX,fp1);
  fclose(fp);fclose(fp1);
  }

void fft(a,m,n)      /* Algortimo de Turkey - Sandey */
struct compleja *a;
int m,n;
{struct compleja u,w,t;
 double arg,aux;
 int i,j=0,k,l,p;
 p=(int)pow(2,(m+1)/2);
 for(i=0;i<n-p;i++)  /* Bit-reversal */
  {if (i<j)
    {t=a[j];a[j]=a[i];a[i]=t;}
    k=n/2;
    while (k<=j)
     {j-=k;k/=2;}
    j+=k;}
 for (k=0;k<m;k++) /* FFT */
  {l=(int)(pow(2,k));
   u.re=1.;u.im=0.;
   arg=M_PI/l;
   w.re=cos(arg);w.im=sin(arg);
   for(j=0;j<l;j++)
   { for(i=j;i<n;i+=l*2)
    {p=i+l;
     t.re=a[p].re*u.re-a[p].im*u.im;
     t.im=a[p].re*u.im+a[p].im*u.re;
     a[p].re=a[i].re-t.re;
     a[p].im=a[i].im-t.im;
     a[i].re+=t.re;
     a[i].im+=t.im;}
    aux=u.re;
    u.re=w.re*aux-w.im*u.im;
    u.im=w.re*u.im+w.im*aux;}}}
```

I.2 ANTITRANSFORMADA RAPIDA DE FOURIER (Afour.c)

Para usar con imagenes cuadradas de tamaño 2^m cambiar BIT=m y NMAX=2^m

```
#include <alloc.h>
#include <dos.h>
#include <stdio.h>
#include <math.h>
#define NMAX  256
#define BIT   8

struct compleja
{float re;
 float im;}

void tabla(struct compleja *p[],int m,int n)
{register int i;
 for (i=0;i<m;i++)
 if((p[i]=(struct compleja*)farcalloc
                  (n,sizeof(struct compleja)))==0)
  {fprintf(stderr,"No hay espacio en memoria \n");
   exit(1);}
 };

main(int argc,char *argv[])
{struct compleja a[NMAX],*block[NMAX];
 register int i,j;
 float nmax=(float)NMAX;
 FILE *fp,*fp1;
 if(argc!=3)
{fprintf(stderr,"USO:AfourArch-transfArch-inversa\n");
   exit(1);}
 if((fp=fopen(argv[argc-2],"rb"))==NULL)
   {fprintf(stderr," No existe %s \n",argv[argc-2]);
     exit(1);}
 tabla(block,NMAX,NMAX);
 fp1=fopen(argv[argc-1],"wb");
 printf("Transformada inversa por columnas\n");
 for(j=0;j<NMAX;j++)
  {fread(block[j],8,NMAX,fp);
   afft(block[j],BIT,NMAX);}
 printf("Transformada inversa por renglones\n");
 for(j=0;j<NMAX;j++)
  {
   for(i=0;i<NMAX;i++)
    a[i]=block[i][j];
   afft(a,BIT,NMAX);
   for(i=0;i<NMAX;i++)
    {block[i][j].re=a[i].re/(nmax*nmax);
     block[i][j].im=a[i].im/(nmax*nmax);}
  }
 for(j=0;j<NMAX;j++)
  fwrite(block[j],8,NMAX,fp1);
 fcloseall();
 sound(1000);
 delay(300);
 nosound();
 return 0;
}

void afft(a,m,n)
struct compleja *a;
int m,n;
{
 struct compleja u,w,t;
 double arg,aux;
 int i,j=0,k,l,p;
 p=(int)pow(2,(m+1)/2);
 for(i=0;i<n-p;j++) /* Bit-reversal */
  {if (i<j)
    {t=a[j];a[j]=a[i];a[i]=t;}
    k=n/2;
    while (k<=j)
     {j-=k;k/=2;}
    j+=k;}
 for (k=0;k<m;k++)  /* FFT */
  {l=(int)(pow(2,k));
   u.re=1.;u.im=0.;
   arg=-M_PI/l;
   w.re=cos(arg);w.im=sin(arg);
   for(j=0;j<l;j++)
   { for(i=j;i<n;i+=l*2)
    {p=i+l;
     t.re=a[p].re*u.re-a[p].im*u.im;
     t.im=a[p].re*u.im+a[p].im*u.re;
     a[p].re=a[i].re-t.re;
     a[p].im=a[i].im-t.im;
     a[i].re+=t.re;
     a[i].im+=t.im;
    }
    aux=u.re;
    u.re=w.re*aux-w.im*u.im;
    u.im=w.re*u.im+w.im*aux;
   }
  }
 }
```

I.3 FILTRO DE HANNING (Hann.c)

Este programa suaviza una imagen con la función 1 + cos(r), para apodizar la transformada de Fourier de una imagen. El movimiento del circulo lo hace a través del uso del ratón.

Parámetros: 1) Archivo de la imagen original
2) Archivo de la imagen filtrada

```
#include <alloc.h>
#include <math.h>
#include "agraficas.h"
#include "mouse.h"
#define NMAX  256
#define HOR   320
#define VER   200
/* Variables para usar el ratón */
int xc=NMAX/2,yc=NMAX/2-1,estado=1,l;

void tabla(unsigned char *p[],int m,int n)
{register int i;

 for (i=0;i<m;i++)
if((p[i]=(unsigned char*)farcalloc
                (n,sizeof(unsigned char)))==0)
  {fprintf(stderr,"No hay espacio en  memoria \n");
  exit(1);}
}

void pix(int x,int y)
{int vec[900];
 l++;
 if(estado==0)
  scrn[(yc-y)*HOR+(xc-x)]=vec[l];
 else
  {vec[l]=scrn[(yc-y)*HOR+(xc-x)];
   if(yc-y<VER)
     scrn[(yc-y)*HOR+(xc-x)]=63;}
}

void pcp(int x,int y)
{if(x==0)
  {pix( y,x);  pix(-y,x);}
 else
  if(y==0)
   {pix(y, x); pix(y,-x);}
  else
   {pix( y, x);pix(-y, x);
    pix( y,-x);pix(-y,-x);
    pix( x, y);pix(-x, y);
    pix( x,-y);pix(-x,-y);}}

void circulo(int r)
{int x,y,p;
 l=(x=0);
 y=r;
 p=3-2*r;
 while(x<y)
  {pcp(x,y);
   if(p<0)
    p=p+4*x+6;
   else
    {p=p+4*(x-y)+10;
     y=y-1;}
   x++;}
}

int redondea(double x)
{double ent,frac;
 return((modf(sqrt(x),&ent)<.5))?ent:ent+1);
 }

void exhibe(datos,max)
unsigned chra *datos[NMAX];
float max;
{register int i,j;

 for(j=VER;j>=0;j--)
   for(i=0;i<NMAX;i++)
   if(i<NMAX-1)
 scrn[j*HOR+i] = (int)(datos[NMAX-1-j][i]*63./max);
}

void main(int argc, char *argv[])
{ unsigned char *datos[NMAX];
  float max=0,aux;
  int c,ro,r=120,k=1,i,j;
  FILE *fp,*fp1;
  char xca[5],yca[5],ra[5];
  int x=xc,y=yc,boton=0,hormin=0;
  int hormax=NMAX,vermin=0,vermax=NMAX;
 if(argc!=3)
{fprintf(stderr,"USO:HANN Arch-orig  Arch-filtrado");
   exit(0);}
```

```
if((fp=fopen(argv[argc-2],"rb"))==NULL)
 {fprintf(stderr,"No existe el archivo %s.\n"
                              ,argv[argc-2]);
   exit(1);}
if((fp1=fopen(argv[argc-1],"wb"))==NULL)
 {fprintf(stderr,"No puedo abrir %s.\n",argv[argc-1]);
 exit(2);}
 tabla(datos,NMAX,NMAX);
 for (j=0;j<NMAX;j++)
  {fread(datos[j],1,NMAX,fp);
   for(i=0;i<NMAX;i++)
    max=(max<datos[j][i])?datos[j][i]: max;}
 agraficas();
 exhibe(datos,max);
 escribe("Centro",270,5,67);
 escribe("x",275,15,68);
 itoa(xc-128,xca,10);  escribe(xca,290,15,63);
 escribe("y",275,25,68);
 itoa(128-yc-1,yca,10);  escribe(yca,290,25,63);
 escribe("radio",270,35,67);
 itoa(r,ra,10);   escribe(ra,280,45,63);
 escribe("Botones",270,57,112);
 escribe("izquierdo",270,70,68);
 escribe("variar  rad.",265,80,63);
 escribe("derecho",270,90,68);
 escribe("filtrar",270,100,63);
 escribe("y",275,108,63);
 escribe("salir",270,116,63);
 initializemouse();
 sethorlim(hormin+r,hormax-r);
 setverlim(vermin+r,vermax-r);
 setmouseposition(xc,yc);
 while(k)
  {circulo(r);
   while(x==xc && y==yc && boton==0)
    getmouseposition(&x,&y,&boton);
   estado=0;
   circulo(r);/* borrado del circulo */
   estado=1;  /* nuevo circulo */
 if( boton == 1)
    {int xa,xb=x,ya;
     while(boton == 1)
      {
       circulo(r);
       xa=xb;
       while(xb==xa && boton==1)
         getmouseposition(&xb,&ya,&boton);
       estado=0;
       circulo(r);
       estado=1;
       if(r>=0)
        if(r<NMAX/2)
          r+=xb-xa;
        else
          r=NMAX/2-1;
       else
        r=0;
       escribe("@@@@@",280,45,0);
       itoa(r,ra,10);escribe(ra,280,45,63);
      }
     sethorlim(hormin+r,hormax-r);
     setverlim(vermin+r,vermax-r);
     setmouseposition(x,y);
    }
 if(boton == 2) /* filtro y salvo */
    {
     for(j=0;j<NMAX;j++)
      {
       for(i=0;i<NMAX;i++)
          {
          ro=redondea(pow(NMAX-yc-j,2)+pow(xc-i,2));
           aux=1+cos(0.5*M_PI*ro/r);
           if(ro<r)
            datos[j][i]=(int)datos[j][i]*aux;
           else
            datos[j][i]=0;
          }
      }
     exhibe(datos,max);
     k=0;
     for(j=0;j<NMAX;j++)
      fwrite(datos[j],1,NMAX,fp1);
    }
   xc=x;
   yc=y;
   boton=0;
   escribe("@@@@@",290,15,0);
   itoa(xc-128,xca,10);
   escribe(xca,290,15,63);
   escribe("@@@@@",290,25,0);
   itoa(128-yc-1,yca,10);
   escribe(yca,290,25,63);
 }
 fcloseall();
 modeset(3);
 resetmouse();
}
```

I.4 RUTINA AGRAFICAS.H

```
#include<stdio.h>
#include<dos.h>
#include<string.h>
#define MAX 320
/* Dirección del monitor VGA */
char far *scrn=(char far *) 0xA0000000;
escribe (palabra[],p_x,p_y,col)
unsigned char palabra[];
int p_x,p_y,col;
{int i,j,k,largo=strlen(palabra);
 unsigned char letra;
 for(j=0;j<largo;j++)
 {letra=palabra[j];
  if(65<=letra && letra<=90) letra+=32;
  if(letra==165)letra-=1;
  switch (letra)
   {case 32: /* Espacio */
           p_x+=2;break;
    case 92: /* Backslash \ */
           for(i=0;i<3;i++)
            scrn[(p_y+i+1)*MAX+p_x+i] = col;
           p_x+=5;break;
    case 58: /* Dos puntos : */
           scrn[(p_y+1)*MAX+p_x]=col;
           scrn[(p_y+3)*MAX+p_x]=col;
           p_x+=2;break;
    case 64: /* @ para limpiar */
           for(i=0;i<5;i++)
            for(k=0;k<10;k++)
             scrn[(p_y+i)*MAX+p_x+k]=0;
           p_x+=6;break;
    case 97: /* Letra A */
           for(i=0;i<4;i++)
           {scrn[(p_y+i+1)*MAX+p_x] = col;
            scrn[(p_y+i+1)*MAX+p_x+3]=col;
            if(i<2)
             {scrn[(p_y+2)*MAX+p_x+1+i]=col;
              scrn[(p_y)*MAX+p_x+1+i]=col;}}
           p_x+=6;break;
    case 98: /* Letra B */
           for(i=0;i<5;i++)
            {scrn[(p_y+i)*MAX+p_x] = col;
             if(i<3)
              {scrn[(p_y)*MAX+p_x+i] = col;
               scrn[(p_y+2)*MAX+p_x+i]= col;
               scrn[(p_y+4)*MAX+p_x+i]= col;
               if(i!=1)
          scrn[(p_y+1+i)*MAX+p_x+3]=col;
              }}
           p_x+=6;break;
case 99:  /* Letra C */
          for(i=0;i<4;i++)
           {scrn[(p_y+i)*MAX+p_x] = col;
            scrn[(p_y)*MAX+p_x+i] = col;
            scrn[(p_y+4)*MAX+p_x+i]=col;}
          scrn[(p_y+4)*MAX+p_x]=col;
          p_x+=6;break;
case 100: /* Letra D */
          for(i=0;i<5;i++)
           {if(i<3)
             {scrn[p_y*MAX+p_x+i]=col;
              scrn[(p_y+4)*MAX+p_x+i]=col;
              scrn[(p_y+i+1)*MAX+p_x+3]=col;}
            scrn[(p_y+i)*MAX+p_x]=col;}
          p_x+=6;break;
case 101: /* Letra E */
          for(i=0;i<4;i++)
           {scrn[(p_y+i)*MAX+p_x]=col;
            scrn[(p_y)*MAX+p_x+i]=col;
            if(i!=3)
             scrn[(p_y+2)*MAX+p_x+1+i]=col;
            scrn[(p_y+4)*MAX+p_x+i]=col;}
          scrn[(p_y+4)*MAX+p_x]=col;
          p_x+=6;break;
case 102: /* Letra F */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x]=col;
            if(i<4)
             scrn[(p_y)*MAX+p_x+i]=col;}
           scrn[(p_y+2)*MAX+p_x+1]=col;
           scrn[(p_y+2)*MAX+p_x+2]=col;
           p_x+=6; break;
case 103: /* Letra G */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x]=col;
            if(i<4)
             {scrn[(p_y)*MAX+p_x+i]=col;
              scrn[(p_y+4)*MAX+p_x+i]=col;}}
          scrn[(p_y+2)*MAX+p_x+2]=col;
          scrn[(p_y+2)*MAX+p_x+3]=col;
          scrn[(p_y+3)*MAX+p_x+3]=col;
          p_x+=6; break;
case 104: /* Letra H */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x]=col;
            scrn[(p_y+i)*MAX+p_x+3]=col;}
          scrn[(p_y+2)*MAX+p_x+1]=col;
          scrn[(p_y+2)*MAX+p_x+2]=col;
          p_x+=6;break;
```

```
case 105: /* Letra I */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x+1]=col;
            if(i<3)
             {scrn[p_y*MAX+p_x+i]=col;
              scrn[(p_y+4)*MAX+p_x+i]=col;}}
          p_x+=5;break;
case 106: /* Letra J */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x+2]=col;
            if(i<3)
             {scrn[p_y*MAX+p_x+1+i]=col;
              scrn[(p_y+4)*MAX+p_x+i]=col;}
            if(i<2)
             scrn[(p_y+3+i)*MAX+p_x]=col;}
          p_x+=6;break;
case 107: /* Letra K */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x]=col;
            if(i<3)
             {scrn[(p_y+2+i)*MAX+p_x+1+i]=col;
              scrn[(p_y+2-i)*MAX+p_x+1+i]=col;}}
          p_x+=6;break;
case 108: /* Letra L */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x]=col;
            if(i<3)
             scrn[(p_y+4)*MAX+p_x+i]=col;}
          p_x+=5;break;
case 109: /* Letra M */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x]=col;
            scrn[(p_y+i)*MAX+p_x+4]=col;}
          scrn[(p_y+1)*MAX+p_x+1]=col;
          scrn[(p_y+2)*MAX+p_x+2]=col;
          scrn[(p_y+1)*MAX+p_x+3]=col;
          p_x+=7;break;
case 110: /* Letra N */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x]=col;
            scrn[(p_y+i)*MAX+p_x+3]=col;}
          scrn[(p_y+1)*MAX+p_x+1]=col;
          scrn[(p_y+2)*MAX+p_x+2]=col;
          p_x+=6;break;
case 164: /* Letra Ñ */
          for(i=0;i<3;i++)
           {scrn[(p_y+i+2)*MAX+p_x]=col;
            scrn[(p_y+i+2)*MAX+p_x+3]=col;
            scrn[(p_y)*MAX+p_x+i]=col;}
          scrn[(p_y)*MAX+p_x+3]=col;
          scrn[(p_y+3)*MAX+p_x+1]=col;
          scrn[(p_y+4)*MAX+p_x+2]=col;
          p_x+=6;break;

case 111: /* Letras O */
          for(i=0;i<4;i++)
           {scrn[p_y*MAX+p_x+i]=col;
            scrn[(p_y+4)*MAX+p_x+i]=col;
            scrn[(p_y+1+i)*MAX+p_x]=col;
            scrn[(p_y+1+i)*MAX+p_x+3]=col;}
          p_x+=6;break;
case 112: /* Letra P */
          for(i=0;i<5;i++)
           scrn[(p_y+i)*MAX+p_x]=col;
          for(i=0;i<3;i++)
           scrn[(p_y+i)*MAX+p_x+2]=col;
          scrn[p_y*MAX+p_x+1]=col;
          scrn[(p_y+2)*MAX+p_x+1]=col;
          p_x+=5;break;
case 113: /* Letra Q */
          for(i=0;i<4;i++)
           {scrn[p_y*MAX+p_x+i]=col;
            scrn[(p_y+4)*MAX+p_x+i]=col;
            scrn[(p_y+1+i)*MAX+p_x]=col;
            scrn[(p_y+1+i)*MAX+p_x+3]=col;}
          scrn[(p_y+3)*MAX+p_x+2]=col;
          scrn[(p_y+4)*MAX+p_x+4]=col;
          p_x+=6;break;
case 114: /* Letra R */
          for(i=0;i<5;i++)
           scrn[(p_y+i)*MAX+p_x]=col;
          for(i=0;i<2;i++)
           {scrn[p_y*MAX+p_x+1+i]=col;
            scrn[(p_y+2)*MAX+p_x+1+i]=col;
            scrn[(p_y+i)*MAX+p_x+2]=col;
            scrn[(p_y+3+i)*MAX+p_x+1+i]=col;}
          p_x+=5;break;
case 115: /* Letra S */
          for(i=0;i<4;i++)
           {scrn[p_y*MAX+p_x+i]=col;
            scrn[(p_y+2)*MAX+p_x+i]=col;
            scrn[(p_y+4)*MAX+p_x+i]=col;
            scrn[(p_y+1)*MAX+p_x]=col;
            scrn[(p_y+3)*MAX+p_x+3]=col;}
          p_x+=6;break;
case 116: /* Letra T */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x+1]=col;
            if(i<3) scrn[p_y*MAX+p_x+i]=col;}
          p_x+=5;break;
case 117: /* Letra U */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x]=col;
            scrn[(p_y+i)*MAX+p_x+3]=col;}
          scrn[(p_y+4)*MAX+p_x+1]=col;
          scrn[(p_y+4)*MAX+p_x+2]=col;
          p_x+=6;break;
```

```
case 118: /* Letra V */
          for(i=0;i<4;i++)
           {scrn[(p_y+i)*MAX+p_x]=col;
            scrn[(p_y+i)*MAX+p_x+3]=col;}
          scrn[(p_y+4)*MAX+p_x+2]=col;
          scrn[(p_y+4)*MAX+p_x+1]=col;
          p_x+=6;break;
case 119: /* Letra W */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x]=col;
            scrn[(p_y+i)*MAX+p_x+4]=col;}
          scrn[(p_y+3)*MAX+p_x+1]=col;
          scrn[(p_y+2)*MAX+p_x+2]=col;
          scrn[(p_y+3)*MAX+p_x+3]=col;
          p_x+=7;break;
case 120: /* Letra X */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x+i]=col;
            scrn[(p_y+4-i)*MAX+p_x+i]=col;}
          p_x+=7;break;
case 121: /* Letra Y */
          for(i=0;i<3;i++)
           {scrn[(p_y+i)*MAX+p_x+i]=col;
            scrn[(p_y+i)*MAX+p_x+4-i]=col;
            scrn[(p_y+2+i)*MAX+p_x+2]=col;}
          p_x+=7;break;
case 122: /* Letra Z */
          for(i=0;i<4;i++)
           {scrn[(p_y)*MAX+p_x+i]=col;
            scrn[(p_y+i)*MAX+p_x+3-i]=col;
            scrn[(p_y+4)*MAX+p_x+i]=col;}
          scrn[(p_y+3)*MAX+p_x]=col;
          p_x+=6;break;
case 46:  /* Punto . */
          scrn[(p_y+4)*MAX+p_x]=col; p_x+=3;break;
case 45:  /* Signo - */
          for(i=0;i<3;i++)
           scrn[(p_y+2)*MAX+p_x+i]=col;
          p_x+=5;break;
case 43:  /* Signo + */
          for(i=0;i<3;i++)
           {scrn[(p_y+2)*MAX+p_x+i]=col;
            scrn[(p_y+1+i)*MAX+p_x+1]=col;}
          p_x+=5;break;
case 49:  /* Número 1 */
          for(i=0;i<5;i++)
           scrn[(p_y+i)*MAX+p_x+1]=col;
          scrn[(p_y+1)*MAX+p_x]=col; p_x+=4;break;
case 50:  /* Número 2 */
          for(i=0;i<3;i++)
           {if(i!=2) scrn[(p_y)*MAX+p_x+i+1]=col;
            scrn[(p_y+1+i)*MAX+p_x+3-i]=col;
            scrn[(p_y+4)*MAX+p_x+1+i]=col;}
          scrn[(p_y+1)*MAX+p_x]=col;
          scrn[(p_y+4)*MAX+p_x]=col; p_x+=6;break;
case 51:  /* Número 3 */
          for(i=0;i<3;i++)
           {scrn[(p_y)*MAX+p_x+i]=col;
            if(i!=2)
             scrn[(p_y+4)*MAX+p_x+i]=col;}
          scrn[(p_y+1)*MAX+p_x+2]=col;
          scrn[(p_y+2)*MAX+p_x+1]=col;
          scrn[(p_y+3)*MAX+p_x+2]=col;
          p_x+=6;break;
case 52:  /* Número 4 */
          for(i=0;i<5;i++)
           {scrn[(p_y+i)*MAX+p_x+2]=col;
            if(i<3)
             scrn[(p_y+i)*MAX+p_x]=col;
            if(i<4)
             scrn[(p_y+2)*MAX+p_x+i]=col;}
          p_x+=6;break;case 53:  /* Número 5 */
          for(i=0;i<4;i++)
           {scrn[p_y*MAX+p_x+i]=col;
            if(i<3)
             {scrn[(p_y+2)*MAX+p_x+i]=col;
              scrn[(p_y+4)*MAX+p_x+i]=col;}}
          scrn[(p_y+1)*MAX+p_x]=col;
          scrn[(p_y+3)*MAX+p_x+3]=col;
          p_x+=6;break;
case 54:  /* Número 6 */
          for(i=0;i<3;i++)
           {scrn[p_y*MAX+p_x+i+1]=col;
            scrn[(p_y+2)*MAX+p_x+i]=col;
            if(i!=0)
             scrn[(p_y+4)*MAX+p_x+i]=col;}
          scrn[(p_y+1)*MAX+p_x]=col;
          scrn[(p_y+3)*MAX+p_x+3]=col;
          scrn[(p_y+3)*MAX+p_x]=col;
          p_x+=6;break;
case 55:  /* Número 7 */
          for(i=0;i<4;i++)
           scrn[p_y*MAX+p_x+i]=col;
          scrn[(p_y+1)*MAX+p_x+3]=col;
          scrn[(p_y+2)*MAX+p_x+2]=col;
          scrn[(p_y+3)*MAX+p_x+1]=col;
          scrn[(p_y+4)*MAX+p_x]=col;
          p_x+=6;break;
case 56:  /* Número 8 */
          for(i=0;i<3;i++)
           {if(i!=1){scrn[(p_y+1+i)*MAX+p_x]=col;
            scrn[(p_y+1+i))*MAX+p_x+3]=col;}
            if(i!=2){scrn[(p_y)*MAX+p_x+1+i]=col;
             scrn[(p_y+2)*MAX+p_x+1+i]=col;
             scrn[(p_y+4)*MAX+p_x+1+i]=col;}}
          p_x+=6;break;
```

```
case 57:  /* Número 9 */
          for(i=0;i<3;i++)
           {if(i!=0)
             scrn[(p_y+4)*MAX+p_x+i]=col;
            scrn[(p_y+2)*MAX+p_x+i+1]=col;
            if(i!=0)
             scrn[(p_y)*MAX+p_x+i]=col;}
          scrn[(p_y+3)*MAX+p_x+3]=col;
          scrn[(p_y+1)*MAX+p_x+3]=col;
          scrn[(p_y+1)*MAX+p_x]=col;
          p_x+=6;break;
case 48: /* Número 0 */
          for(i=0;i<3;i++)
           {if(i<2)
            {scrn[p_y*MAX+p_x+i+1]=col;
             scrn[(p_y+4)*MAX+p_x+i+1]=col;}
            scrn[(p_y+1+i)*MAX+p_x]=col;
            scrn[(p_y+1+i)*MAX+p_x+3]=col;}
          p_x+=6;break;
         }
   }
return 0;
} /* Final de escribe */
```

```
void agraficas(void)
{int i ;
 union REGS iosregs;

 iosregs.h.ah=0;
 iosregs.h.al=0x13;
 int86(0x10,&iosregs,&iosregs);
 for (i=0; i<64; i++)
  {iosregs.h.ah=0x10;
   iosregs.h.al=0x10;
   iosregs.x.bx=i;          /* Color */
   iosregs.h.dh=i;          /* Rojo */
   iosregs.h.ch=i;          /* Verde */
   iosregs.h.cl=i;          /* Azul */
   int86(0x10,&iosregs,&iosregs);
  }
}

modeset(mode)
 int mode ;
{union REGS regs ;
 regs.h.al = mode ;
 regs.h.ah = 0 ;
 int86(0x10,&regs,&regs);
 return 0;
}
```

I.5 EXHIBICION DE IMAGENES

A continuación se listan los programas usados para la exhibición de imágenes. En todos los casos se usa como parámetro el archivo a desplegar.

I.5.1 Exhibición de imágenes almacenadas con números flotantes. (Exfase.c)

```
#include <alloc.h>
#include <math.h>
#include "vga.h"
#define NMAX  256

void tabla(float *p[],int m,int n)
{register int i;

 for (i=0;i<m;i++)
  if((p[i]=(float*)farcalloc(n,sizeof(float)))==0)
  {fprintf(stderr,"No hay espacio en memoria \n");
   exit(1);}
};

main(int argc, char *argv[])
{float *datos[NMAX],max=0,min=0;
 register int i,j;
 FILE *fp;

 if(argc!=2)
  {fprintf(stderr,"USO: Exfase Archivo-fase \n");
   exit(1);}
if((fp=fopen(argv[argc-1],"rb"))==NULL)
 {fprintf(stderr," No existe  %s. \n",argv[argc-1]);
  exit(1);}
 tabla(datos,NMAX,NMAX);
 for (j=0;j<NMAX;j++)
  {fread(datos[j],4,NMAX,fp);
   for(i=0;i<NMAX;i++)
    {max=(max<datos[j][i])? datos[j][i] : max;
     min=(min>datos[j][i])? datos[j][i] : min;}}
 agraficas();
 for(j=200;j>=0;j--)
  for(i=0;i<NMAX;i++)
 scrn[j*320+i]=
    (int((datos[NMAX-j-1][i]-min)*63./(max-min));
 fclose(fp);
 sound(1000); delay(300); nosound();
 getche();
 modeset(3);
 return 0;
 }
```

I.5.2 Exhibición de imágenes almacenadas con caracteres. (Exhibe.c)

```
#include <alloc.h>
#include "agraficas.h"
#define NMAX  256

void tabla(unsigned char *p[],int m,int n)
{register int i;
 for (i=0;i<m;i++)
  if((p[i]=(unsigned char*)farcalloc
                  (n,sizeof(unsigned char)))==0)
   {fprintf(stderr,"No hay espacio en memoria\n");
    exit(1);}}

void main(int argc, char *argv[])
{unsigned char *datos[NMAX];
 unsigned char  block[NMAX],buff[20];
 float max=0;
 register int i,j;
 FILE *fp;
if(argc!=2)
 {fprintf(stderr,"USO: Exhibe Archivo-imagen\n");
  exit(1);}
if((fp=fopen(argv[argc-1],"rb"))==NULL)
 {fprintf(stderr," No existe %s. \n", argv[argc-1]);
  exit(1);}
tabla(datos,NMAX,NMAX);
for (j=0;j<NMAX;j++)
 {fread(block,1,NMAX,fp);
  for(i=0;i<NMAX;i++)
  max=(max<(datos[j][i]=block[i]))?datos[j][i]:max;}
agraficas();
for(j=200;j>=0;j--)
 for(i=0;i<NMAX;i++)
 scrn[j*320+i]=(int)(datos[NMAX-j-1][i]*63./max);
fclose(fp);
sound(1000); delay(300); nosound();
getche();
modeset(3);}
```

I.5.3 Exhibición de imágenes almacenadas con números complejos (Exfour.c)

```
#include <alloc.h>
#include <dos.h>
#include <math.h>
#include "agraficas.h"
#define NMAX  256
struct compleja
{  float x;
   float y; };
void tabla(float *p[],int m,int n)
{register int i;
 for (i=0;i<m;i++)
  if((p[i]=(float*)farcalloc(n,sizeof(float)))==0)
  {fprintf(stderr,"No hay espacio en memoria \n");
   exit(1); }
};

void main(int argc, char *argv[])
{struct compleja block[NMAX];
  float *datos[NMAX],max=0;
  register int i,j;
  FILE *fp;
  if(argc!=2)
   {fprintf(stderr,"USO: Exfour Archivo \n");
    exit(1);}
  if((fp=fopen(argv[argc-1],"rb"))==NULL)
   { fprintf(stderr," No existe %s.\n", argv[argc-1]);
     exit(1); }
  tabla(datos,NMAX,NMAX);
  for (j=0;j<NMAX;j++)
   {fread(block,8,NMAX,fp); // Aqui se puede
    for(i=0;i<NMAX;i++)    // eliminar log para
max=(max<(datos[j][i]=     // tener exhibición lineal
log(cabs(block[i])+1)))?datos[j][i]:max;}
  agraficas();
  for(j=3*NMAX/2-1-200;j<NMAX;j++)
   for(i=0;i<NMAX;i++)
    if(i<NMAX/2)
     scrn[(3*NMAX/2-1-j)*320+i+NMAX/2]=
                    (int)(datos[j][i]*63./max);
    else
     scrn[(3*NMAX/2-1-j)*320+i-NMAX/2]=
                    (int)(datos[j][i]*63./max);
   for(j=0;j<NMAX/2;j++)
    for(i=0;i<NMAX;i++)
     if(i<NMAX/2)
      scrn[(NMAX/2-1-j)*320+i+NMAX/2]=
                    (int)(datos[j][i]*63./max);
     else
      scrn[(NMAX/2-1-j)*320+i-NMAX/2]=
                    (int)(datos[j][i]*63./max);
  fclose(fp);
  sound(1000);delay(300);nosound();
  getche();
  modeset(3);}
```

I.6 RUTINAS DE MANEJO DE RATON (MOUSE.H)

Estas rutinas son usadas para controlar el movimiento del ratón, haciendo uso de la interrupción 33h (51) del sistema opertivo.

```
union REGS iregs,oregs;
int initializemouse()
{iregs.x.ax =0;
 int86( 0x33, &iregs, &oregs );
 if( oregs.x.ax == 0xFFFF) return( oregs.x.bx );
 else return (0);}

void resetmouse()
{iregs.x.ax=0;
 int86( 0x33, &iregs, &oregs );}

void setmouseposition(int x, int y)
{iregs.x.ax=4;
 iregs.x.cx=x*8;
 iregs.x.dx=y*8;
 int86(0X33,&iregs,&oregs);}

void getmouseposition( int *x, int *y, int
        *mousebutton )
{iregs.x.ax=3;
 int86( 0x33, &iregs, &oregs );
 *x= oregs.x.cx /8;
 *y= oregs.x.dx /8;
 *mousebutton=oregs.x.bx;}

void sethorlim( int hormin, int hormax )
{iregs.x.ax=7;iregs.x.dx=hormin*8;
 iregs.x.cx=hormax*8;
 int86( 0x33, &iregs, &oregs );}

void setverlim( int vermin, int vermax )
{iregs.x.ax=8;iregs.x.dx=vermin*8;
 iregs.x.cx=vermax*8;
 int86( 0x33, &iregs, &oregs );}
```

I.7 FILTRAJE EN EL PLANO DE FOURIER (Cirfou.c)

Con este programa se muestra como realizar operaciones en el plano de Fourier. Parámetros: 1) Archivo de la imagen transformado 2) Archivo filtrado de la transformada.

```
#include <alloc.h>
#include <math.h>
#include "agraficas.h"
#include "mouse.h"
#define NMAX  256
#define HOR   320
#define VER   200
int xc=NMAX/2,yc=NMAX/2-1,estado=1,i;
union REGS iregs, oregs;
struct compleja
{float x; float y;};

void tabla(float *p[],int m,int n)
{ register int i;
 for (i=0;i<m;i++)
  if((p[i]=(float*)farcalloc(n,sizeof(float)))==0)
  {fprintf(stderr,"No hay espacio en memoria \n");
   exit(1);}}

void pix(int x,int y)
{int vec[700];
   if(estado==0)
    scrn[(yc-y)*HOR+(xc-x)]=vec[i++];
   else{vec[i]=scrn[(yc-y)*HOR+(xc-x)];
    scrn[(yc-y)*HOR+(xc-x)]=63;}}

void pcp(int x,int y)
{if(x==0)
  {pix( y,x);pix(-y,x);}
 else if(y==0)
   {pix(y, x);pix(y,-x);}
  else
   {pix( y, x);pix(-y, x);
    pix( y,-x);pix(-y,-x);
    pix( x, y);pix(-x, y);
    pix( x,-y);pix(-x,-y);}}

void circulo(int r)
{int x,y,p;
 i=(x=0); y=r; p=3-2*r;
 while(x<y)
  {pcp(x,y);
   if(p<0)  p=p+4*x+6;
   else
    {p=p+4*(x-y)+10;
     y=y-1;}
   x++;}}

int redondea(int x)
{double ent,frac;
  return((modf(sqrt(x)<.5))?(int)ent:(int)ent+1);}

void selecciona(int r, FILE *fp,FILE *fp1)
{int i,j,ycc,xcc,rr,r1,ro,dif,lec,banda;
 long pos,pos1;
 struct compleja cirblock[NMAX];
 ycc=NMAX/2-1-yc; xcc=xc-NMAX/2;
 banda = (int)(r*.25);
 if(r!=0)
 for(j=r-1+banda;j>-r-banda;j--)
  {if(j+ycc<0)
    {pos=(long)(j+ycc+NMAX);pos*=8*NMAX;}
   else
    {pos=(long)(j+ycc);pos*=8*NMAX;}
   if(j<0)
    {pos1=(long)(j+NMAX);pos1*=8*NMAX;}
   else
    {pos1=(long)j;pos1*=8*NMAX;}
   rr=redondea((r+banda)*(r+banda)-j*j)+1;
   if(j<r && j>-r) ro=redondea(r*r-j*j)+1;
   else ro=0;
   lec=2*rr-1; dif=xcc-rr;
   if(dif<0)
    {if(-dif<lec)
     {fseek(fp,pos,SEEK_SET);
      fread(&cirblock[-dif],8,lec+dif,fp);
      pos+=(NMAX+dif)*8;
      fseek(fp,pos,SEEK_SET);
      fread(&cirblock[0],8,-dif,fp);}
     else { pos+=(NMAX+dif+1)*8;
      fseek(fp,pos,SEEK_SET);
      fread(&cirblock[0],8,lec,fp);}
    else { pos+=dif*8;
     fseek(fp,pos,SEEK_SET);
     fread(&cirblock[0],8,lec,fp);}
   for(i=rr-1;i>-rr;i--)
    if(abs(i)>=ro) {r1=redondea(j*j+i*i);
cirblock[i+rr-1].x=cos((r1-r-1)*M_PI/(2*banda));
cirblock[i+rr-1].y=cos((r1-r-1)*M_PI/(2*banda));
     }
   fseek(fp1,pos1,SEEK_SET);
   fwrite(&cirblock[rr-1],8,rr,fp1);
   pos1+=(NMAX-rr+1)*8;
   fseek(fp1,pos1,SEEK_SET);
   fwrite(&cirblock[0],8,rr-1,fp1);}
```

```
void exhibe(float *datos[NMAX],float max)
{register int i,j;
 for(j=3*NMAX/2-1-VER;j<NMAX;j++)
  for(i=0;i<NMAX;i++)
   if(i<NMAX/2)
    scrn[(3*NMAX/2-1-j)*HOR+i+NMAX/2]=
                  (int)(datos[j][i]*63./max);
   else scrn[(3*NMAX/2-1-j)*HOR+i-NMAX/2]=
                  (int)(datos[j][i]*63./max);
 for(j=0;j<NMAX/2;j++)
  for(i=0;i<NMAX;i++)
   if(i<NMAX/2)
    scrn[(NMAX/2-1-j)*HOR+i+NMAX/2]=
                  (int)(datos[j][i]*63./max);
   else  scrn[(NMAX/2-1-j)*HOR+i-NMAX/2]=
                  (int)(datos[j][i]*63./max);}

void main(int argc, char *argv[])
{ struct compleja block[NMAX];
  float *datos[NMAX],max=0;
  int c,r=30,k=1,i,j,x=xc,y=yc,boton=0;
  FILE *fp,*fp1;
  char xca[5],yca[5],ra[5];
  int hormin=0,hormax=NMAX;
  int vermin=0,vermax=VER-1;
  if(argc!=3)
  {fprintf(stderr,"USO:Cirfou Arch-Fou Arch-filtr\n");
    exit(1);}
  if((fp=fopen(argv[argc-2],"rb"))==NULL)
   {fprintf(stderr," No existe %s.\n",argv[argc-2]);
     exit(1);}
  fp1=fopen(argv[argc-1],"wb");
  tabla(datos,NMAX,NMAX);
  for (j=0;j<NMAX;j++)
   {              /* Aqui se puede omitir log para */
    fread(block,8,NMAX,fp); /* tener exhibición */
    for(i=0;i<NMAX;i++)                  /* lineal */
    max=(max<(datos[j][i]=
          log(cabs(block[i])+1)))?datos[j][i]:max;}
  agraficas();exhibe(datos,max);
  escribe("Centro",270,5,67);
  escribe("x",275,15,68); escribe("y",275,25,68);
  itoa(xc-128,xca,10);  escribe(xca,290,15,63);
  itoa(128-yc-1,yca,10);escribe(yca,290,25,63);
  escribe("radio",270,35,67);
  itoa(r,ra,10);  escribe(ra,280,45,63);
  escribe("Botones",275,55,112);
  escribe("izquierdo",270,70,68);
  escribe("variar  rad.",265,80,63);
  escribe("derecho",270,90,68);
  escribe("salir",265,100,63);
  escribe("ambos",270,110,68);
  escribe("filtrar",265,120,63);
  initializemouse();
  sethorlim(hormin+r,hormax-r);
  setverlim(vermin+r,vermax-r);
  setmouseposition(xc,yc);
  while(k)
  {circulo(r);
   while(x==xc && y==yc && boton==0)
    getmouseposition(&x,&y,&boton);
   estado=0;circulo(r);estado=1;
   if( boton == 1)          /* varia del radio */
    {int xa,xb=x,ya;
     while(boton==1)
      {circulo(r);xa=xb;
         while(xb==xa && boton==1)
           getmouseposition(&xb,&ya,&boton);
         estado=0;circulo(r);estado=1;
         if(r>=0) r+=xb-xa;else r=0;
         escribe("@@@@@",280,45,0);
         itoa(r,ra,10);escribe(ra,280,45,63);}
     sethorlim(hormin+r,hormax-r);
     setverlim(vermin+r,vermax-r);
     setmouseposition(x,y);}
    if(boton==2)          /* salgo del programa */
    { int sal=0,xsal,ysal;
     sethorlim(275,285);setverlim(190,195);
     setmouseposition(270,193);
     escribe("Salir",270,175,67);
     while(sal==0)
      {getmouseposition(&xsal,&ysal,&boton);
         if(xsal<280) {escribe("Si",270,190,63);
                       escribe("No",290,190,40);}
         else          {escribe("Si",270,190,40);
                        escribe("No",290,190,63);}
         if(boton==1)
          if(xsal<280){k=0; sal=1;} else sal=1;}
     sethorlim(hormin+r,hormax-r);
     setverlim(vermin+r,vermax-r);
     escribe("@@@@@",270,175,0);
     escribe("@@@@@",270,190,0);
     setmouseposition(x,y);}
    if(boton==3)  /* salvo el archivo */
     {fseek(fp1,0,SEEK_SET);
      for(j=0;j<NMAX;j++)
       {block[j].x=0;block[j].y=0; }
      for(j=0;j<NMAX;j++)
       fwrite(block,8,NMAX,fp1);
       selecciona(r,fp,fp1);}
    xc=x;yc=y;boton=0;
    escribe("@@@",290,15,0);
    itoa(xc-128,xca,10);escribe(xca,290,15,63);
    escribe("@@@",290,25,0);
    itoa(128-yc-1,yca,10);escribe(yca,290,25,63);}
fcloseall(); modeset(3); resetmouse();}
```

I.8 CALCULO DE FASE (Grarg.c)

Este programa recupera la fase a partir de la transformada inversa del filtro del orden uno en el plano de frecuencia.

Parámetros: 1) Archivo de la imagen inversa
2) Archivo para guaradar la fase

```
#include <alloc.h>
#include <math.h>
#include <stdio.h>
#define NMAX  128

struct compleja
{float x;
 float y;};

void tabla(float *p[],int m,int n)
{register int i;

 for (i=0;i<m;i++)
  if(!(p[i]=(float *)farcalloc(n,sizeof(float))))
  {fprintf(stderr,"No hay espacio en memoria \n");
   exit(1);}
};

main(int argc, char *argv[])
{
 struct compleja block[NMAX];
 float *datos[NMAX];
 register int i,j;
 FILE *fp,*fp1;

 if(argc!=3)
  {fprintf(stderr,"USO: Grargcir Archivo-inverso Archivo-fase\n");
   exit(1);}
 if((fp=fopen(argv[argc-2],"rb"))==NULL)
  {fprintf(stderr," No existe el archivo %s.\n", argv[argc-2]);
   exit(1);}
 if((fp1=fopen(argv[argc-1],"wb"))==NULL)
  {fprintf(stderr," No puedo abrir el archivo %s. \n" ,argv[argc-1]);
   exit(1);}
 tabla(datos,NMAX,NMAX);
 for (j=0;j<NMAX;j++)
  {fread(block,8,NMAX,fp);
   for(i=0;i<NMAX;i++)
     datos[j][i]=(float)atan(block[i].y/block[i].x);
   fwrite(datos[j],4,NMAX,fp1);}
 fcloseall();
 sound(1000);delay(300); nosound();
 return 0;
}
```

I.9 DESDOBLAMIENTO DE FASE (Reco.c)

Este programa realiza la reconstrucción de fase en una imagen obtenida de la inversa de la transformada de Fourier bidimensional, de tamaño N X N, establecido un umbral para la diferencia de fase (en múltiplos de PI).

Parámetros: 1) Archivo con la fase
2) Archivo para guardar la fase reconstruida
3) Umbral de diferencia (en múltiplos de PI)

```
#include <alloc.h>
#include <math.h>
#include <stdio.h>
#include <dos.h>
#define NMAX  256

void tabla(float *p[],int m,int n)
{
 register int i;
 for (i=0;i<m;i++)
  if((p[i]=(float*)farcalloc(n,sizeof(float)))==0)
  {fprintf(stderr,"No hay espacio en memoria\n");
   exit(1);}
};

void tabla1(char *p[],int m,int n)
{register int i;
 for (i=0;i<m;i++)
  if((p[i]=(char*)farcalloc(n,sizeof(char)))==0)
   {fprintf(stderr,"No hay espacio en memoria\n");
    exit(1);}
};

void main(int argc, char *argv[])
{ float *datos[NMAX];
  char *ren[NMAX],*col[NMAX],altren=0,altcol=0;
  register int i,j;
  FILE *fp,*fp1;
  if(argc!=4)
   {fprintf(stderr,"USO: Reco Archivo-fase
Archivo-reconstruido Umbral (multiplos de PI)\n");
    exit(1);}
  if((fp=fopen(argv[argc-3],"rb"))==NULL)
   {
    fprintf(stderr," No existe %s.\n",argv[argc-3]);
    exit(1);}
  fp1=fopen(argv[argc-2],"wb");
  if((umb=atof(argv[argc-1]))>=1.)
   {fprintf(stderr,"Se espera a lo más un 1\n");
    exit(1);}
  tabla(datos,NMAX,NMAX);
  tabla1(ren,NMAX,NMAX);
  tabla1(col,NMAX,NMAX);
  for (j=0;j<NMAX;j++)
   fread(datos[j],4,NMAX,fp);
  umb*=M_PI;
  ren[NMAX/2][0]=0;
/*Genero la matriz de escalones por renglones */
/* Identifico los escalones del renglon NMAX/2 */
  for(i=1;i<NMAX;i++)
   if(datos[NMAX/2][i]-datos[NMAX/2][i-1]>umb)
    {altren--;
     ren[NMAX/2][i]=altren;}
   else
    if(datos[NMAX/2][i-1]-datos[NMAX/2][i]>umb)
     {altren++;
      ren[NMAX/2][i]=altren;}
    else
     ren[NMAX/2][i]=altren;
//Identifico los escalones del renglon NMAX/2 -1
  if(datos[NMAX/2][0]-datos[NMAX/2-1][0]>umb)
   ren[NMAX/2-1][0]=ren[NMAX/2][0]-1;
  else
  if(datos[NMAX/2-1][0]-datos[NMAX/2-1][0]>umb)
    ren[NMAX/2-1][0]=ren[NMAX/2][0]+1;
   else
    ren[NMAX/2-1][0]=ren[NMAX/2][0];
  altren=0;
  for(i=1;i<NMAX;i++)
  if(datos[NMAX/2-1][i]-datos[NMAX/2-1][i-1]>umb)
   {altren--;
    ren[NMAX/2-1][i]=altren;}
   else
  if(datos[NMAX/2-1][i-1]-datos[NMAX/2-1][i]>umb)
    {altren++;
     ren[NMAX/2-1][i]=altren;}
    else
     ren[NMAX/2-1][i]=altren;
```

```
/*Identifico los escalones de la mitad inferior */
  for(j=0;j<NMAX;j++)
   for(i=1;i<NMAX/2;i++){
if(datos[i+NMAX/2][j]-datos[i-1+NMAX/2][j]>umb)
      ren[i+NMAX/2][j]=ren[i+NMAX/2-1][j]-1;
     else
if(datos[i-1+NMAX/2][j]-datos[i+NMAX/2][j]>umb)
      ren[i+NMAX/2][j]=ren[i+NMAX/2-1][j]+1;
     else
      ren[i+NMAX/2][j]=ren[i+NMAX/2-1][j];
/*Identifico los escalones de la mitad superior */
if(datos[NMAX/2-i][j]-datos[NMAX/2-i+1][j]>umb)
      ren[NMAX/2-i][j]=ren[NMAX/2-i+1][j]-1;
     else
if(datos[NMAX/2-i+1][j]-datos[NMAX/2-i][j]>umb)
      ren[NMAX/2-i][j]=ren[NMAX/2-i+1][j]+1;
     else
      ren[NMAX/2-i][j]=ren[NMAX/2-i+1][j];
   }
  col[0][NMAX/2]=0;
/*Genero la matriz de escalones por columnas */
  altcol=0;
/*Identifico los escalones de  columna NMAX/2 */
  for(i=1;i<NMAX;i++)
   if(datos[i][NMAX/2]-datos[i-1][NMAX/2]>umb)
    {altcol--;
     col[i][NMAX/2]=altcol;}
   else
    if(datos[i-1][NMAX/2]-datos[i][NMAX/2]>umb)
     {altcol++;
      col[i][NMAX/2]=altcol;}
    else
     col[i][NMAX/2]=altcol;
/*Identifico escalones de la columnaNMAX/2-1*/
  if(datos[0][NMAX/2]-datos[0][NMAX/2-1]>umb)
   col[0][NMAX/2-1]=col[0][NMAX/2]-1;
  else
  if(datos[0][NMAX/2-1]-datos[0][NMAX/2-1]>umb)
   col[0][NMAX/2-1]=col[0][NMAX/2]+1;
   else
    col[0][NMAX/2-1]=col[0][NMAX/2];
  altcol=0;
  for(i=1;i<NMAX;i++)
if(datos[i][NMAX/2-1]-datos[i-1][NMAX/2-1]>umb)
    {altcol--;
     col[i][NMAX/2-1]=altcol;}
   else
if(datos[i-1][NMAX/2-1]-datos[i][NMAX/2-1]>umb)
     {altcol++;
      col[i][NMAX/2-1]=altcol;}
    else
     col[i][NMAX/2-1]=altcol;
/* Identifico los escalones de la mitad derecha */
  for(j=0;j<NMAX;j++)
   for(i=1;i<NMAX/2;i++){
if(datos[j][i+NMAX/2]-datos[j][i-1+NMAX/2]>umb)
      col[j][i+NMAX/2]=col[j][i+NMAX/2-1]-1;
     else
if(datos[j][i-1+NMAX/2]-datos[j][i+NMAX/2]>umb)
      col[j][i+NMAX/2]=col[j][i+NMAX/2-1]+1;
     else
      col[j][i+NMAX/2]=col[j][i+NMAX/2-1];
/*Identifico los escalones de la mitad izquierda*/
if(datos[j][NMAX/2-i]-datos[j][NMAX/2-i+1]>umb)
      col[j][NMAX/2-i]=col[j][NMAX/2-i+1]-1;
     else
if(datos[j][NMAX/2-i+1]-datos[j][NMAX/2-i]>umb)
      col[j][NMAX/2-i]=col[j][NMAX/2-i+1]+1;
     else
      col[j][NMAX/2-i]=col[j][NMAX/2-i+1];
   }
  altren=ren[NMAX/2][NMAX/2];
  altcol=col[NMAX/2][NMAX/2];
  for(j=0;j<NMAX;j++)    /* Normalizo al pixel  */
   for(i=0;i<NMAX;i++)   /*NMAX/2,NMAX/2 */
    {ren[i][j]-=altren;
     col[i][j]-=altcol;}
 /* Comparo si las matrices son iguales por
columnas y renglones */
  for(j=0;j<NMAX;j++)
   {for(i=0;i<NMAX;i++)
     if(ren[j][i]==col[j][i])
      datos[j][i]+=col[j][i]*M_PI;
     else
      datos[j][i]=0;}
  for(j=0;j<NMAX;j++)
   fwrite(datos[j],4,NMAX,fp1);
  fcloseall();
  sound(1000);
  delay(300);
  nosound();
}
```

I.10 TRANSFORMACION A COORDENADAS POLARES (Polar.c)

Este programa convierte una imagen con datos en números flotantes de coordenadas cartesianas a coordenadas polares.

Parámetros: 1) Archivo en coordenadas cartesianas
2) Archivo en coordenadas polares

```
#include <alloc.h>
#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#define NMAX  128

void tabla(float *p[],int m,int n)
{
 register int i;

 for (i=0;i<m;i++)
  if((p[i]=(float*)farcalloc(n,sizeof(float)))==0)
  {fprintf(stderr,"No hay espacio en memoria \n");
   exit(1);}
};
void main(int argc, char *argv[])
{float *datos[NMAX],salida[NMAX],teta;
 int x,y,r,ii,jj,kk;
 register int j;
 FILE *fp,*fp1;
 char cual[10];
 if(argc!=3)
  {fprintf(stderr,"USO: Pol Arch-fase
Arch-polares\n");
   exit(1);}
 if((fp=fopen(argv[argc-2],"rb"))==NULL)
  {fprintf(stderr," No existe %s.\n",argv[argc-2]);
   exit(1);}
 fp1=fopen(argv[argc-1],"wb");
 tabla(datos,NMAX,NMAX);
 for (j=0;j<NMAX;j++)
  fread(datos[j],4,NMAX,fp);
 for(r=0;r<NMAX/2;r++)
  {for(j=0;j<NMAX;j++)
    {teta=j*2.*M_PI/NMAX;
     x=(int)(r*cos(teta)+NMAX/2.);
     y=(int)(r*sin(teta)+NMAX/2.);
     salida[j]=datos[x][y];}
   fwrite(salida,4,NMAX,fp1);}
 fwrite(salida,4,NMAX,fp1);
 fcloseall();
 sound(1000); delay(300);nosound();
 }
```

I.11 POLINOMIOS DE ZERNIKE (Aberr.c)

Este programa evalua los coeficientes de Zerenike tercer orden de un frente de onda a partir de una imagen con la fase reconstrida coordenadas polares.

Parámetros: 1) Archivo de la imagen inversa en polares

```
#include <alloc.h>
#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define NMAX  256
#define NMAX2 NMAX/2

void tabla(float *p[],int m,int n)
{register int i;
 for (i=0;i<m;i++)
  if((p[i]=(float*)farcalloc(n,sizeof(float)))==0)
  {fprintf(stderr,"No hay espacio en memoria \n");
   exit(1);}
}

float integral_r(fr,r)
float (*fr)(),*r;
{int i;
 float valor=0.,inter=2./NMAX;

 for(i=0;i<NMAX/2;i++)
  valor+=(*r++)*inter*((*fr)((i+1.)*inter)
                       +(*fr)(i*inter))/2.;
 return(valor);}

float integral_teta(fteta,linea)
float (*fteta)(),*linea;
 {int i;
  float valor=0.,inter=2.*M_PI/NMAX;

  for(i=0;i<NMAX;i++)
   valor+=inter*(*linea++)*((*fteta)((i+1)*inter)
                       +(*fteta)(i*inter))/2.;
  return(valor);}

float calcula(f_r,f_teta,datos,cte)
float (*f_r)(),(*f_teta)(),*datos[NMAX],cte;
 {float teta[NMAX/2];
  int j;
  for(j=0;j<NMAX/2;j++)
   teta[j]=integral_teta(f_teta,datos[j]);
  return(integral_r(f_r,teta)*cte/M_PI);
 }

void main(int argc, char *argv[])
{float f_r_0(),f_r_1(),f_r_3(),f_r_4(),f_r_6();
 float f_r_8();
 float f_t_0(),f_t_1(),f_t_2(),f_t_4(),f_t_5();
 float *datos[NMAX],Z[9],aberr[11],c,aux,sgn;
 register int j,i;
 FILE *fp;
 if(argc!=2)
  {fprintf(stderr,"USO: Aberr Arch-polares \n");
   exit(1);}
 if((fp=fopen(argv[argc-1],"rb"))==NULL)
  {fprintf(stderr," No existe %s\n",argv[argc-1]);
   exit(1);}
 tabla(datos,NMAX/2,NMAX);
 for (j=0;j<NMAX/2;j++)
  fread(datos[j],4,NMAX,fp);
 for(j=0;j<NMAX/2;j++)
  for(i=0;i<NMAX;i++)
   datos[j][i]=-datos[j][i];
 printf(" COEFICIENTES DE ZERNIKE \n"); j=0;
 Z[j++]=calcula(f_r_0,f_t_0,datos,1.);
 Z[j++]=calcula(f_r_1,f_t_1,datos,4.);
 Z[j++]=calcula(f_r_1,f_t_2,datos,4.);
 Z[j++]=calcula(f_r_3,f_t_0,datos,3.);
 Z[j++]=calcula(f_r_4,f_t_4,datos,6.);
 Z[j++]=calcula(f_r_4,f_t_5,datos,6.);
 Z[j++]=calcula(f_r_6,f_t_1,datos,8.);
 Z[j++]=calcula(f_r_6,f_t_2,datos,8.);
 Z[j++]=calcula(f_r_8,f_t_0,datos,5.);
 for(j=0;j<9;j++)
printf("El coeficiente Z[%d] es %7.3G lambdas\n",
                                  j,Z[j]/(2.*M_PI));
 printf("\n COEFICIENTES DE SEIDEL\n");
/* POTENCIA */
 aberr[0]=2.*Z[3];
printf("Coeficiente de potencia: %7.3f lambdas\n",
                                  aberr[0]/(2*M_PI));
/* DEFOCO */
 aux=sqrt(Z[4]*Z[4]+Z[5]*Z[5]);
 aberr[1]=2.*Z[3]-6.*Z[8];
 if(fabs(aberr[1]+aux)<fabs(aberr[1]-aux))
  {aberr[1]+=aux;sgn=-1.;}
  else{aberr[1]-=aux;sgn=1.;}
printf("Coeficiente de defoco   : %7.3f lambdas\n",
                                  aberr[1]/(2*M_PI));
```

```
/* TILT X */
 aberr[2]=Z[1]-2.*Z[6];
printf("Coeficiente tilt-x     : %7.3f lambdas\n",
                              aberr[2]/(2*M_PI));
/* TILT Y */
 aberr[3]=Z[2]-2.*Z[7];
printf("Coeficiente tilt-y     : %7.3f lambdas\n",
                              aberr[3]/(2*M_PI));
/* MAGNITUD DEL TILT*/
aberr[4]=sqrt(pow(aberr[3],2)+pow(aberr[2],2));
printf("Magnitud  tilt         : %7.3f lambdas\n",
                              aberr[4]/(2*M_PI));
/* DIRECCION DEL TILT */
 if(aberr[2]>0)  c=0;
 else
  if(aberr[3]<0)
   c=-M_PI_2;
  else
   c=M_PI_2;
aberr[5]=(atan2(aberr[3],aberr[2])+c)*180./M_PI;
printf("Dirección  tilt        : %7.3f\n",aberr[5]);
/* ESFERICA */
 aberr[6]=6.*Z[8];
printf("Coeficiente esférica   : %7.3f lambdas\n",
                              aberr[6]/(2*M_PI));
/* MAGNITUD DEL COMA */
 aberr[7]=sqrt(Z[6]*Z[6]+Z[7]*Z[7])*3.;
printf("Coeficiente coma       : %7.3f lambdas\n",
                              aberr[7]/(2*M_PI));
/* DIRECCION DEL COMA */
 if(Z[6]>0)  c=0;
 else
  if(Z[7]<0)
   c=-M_PI_2;
  else   c=M_PI_2;
 aberr[8]=(atan2(Z[7],Z[6])+c)*360./(2*M_PI);
printf("Dirección  coma        : %7.3f\n",aberr[8]);
/* MAGNITUD DEL ASTIGMATISMO */
 aberr[9]=2.*sgn*sqrt(Z[4]*Z[4]+Z[5]*Z[5]);
printf("Coef. astigmatismo     : %7.3f lambdas\n",
                              aberr[9]/(2*M_PI));
/* DIRECCION DEL ASTIGMATISMO */
 if(Z[4]>0)  c=0;
 else
  if(Z[5]<0)
   c=-M_PI_2;
  else   c=M_PI_2;
 aberr[10]=(atan2(Z[5],Z[4])+c)*360./(2*M_PI);
printf("Dirección  astigmatismo: %7.3f\n",
                                    aberr[10]);
fcloseall();
 sound(1000); delay(300); nosound();
 getche();}

float f_r_0(x)
float x;
{return (x);}

float f_r_1(x) /* f_r_2 es igual a este */
float x;
{return(x*x);}

float f_r_3(x)
float x;
{return((2.*x*x-1)*x);}

float f_r_4(x)          /* f_r_5 es igual a este */
float x;
{return(x*x*x);}

float f_r_6(x)
float x;
{return((3*x*x-2)*x)*x;}  /* f_r_7 es igual a este*/

float f_r_8(x)
float x;
{return((6.*x*x*x*x-6.*x*x+1)*x);}

float f_t_0()   /* f_t_3 y 8 son iguales a este */
{return(1.);}

float f_t_1(x)  /* f_t_6 es igual a este */
float x;
{return(cos(x));}

float f_t_2(x)  /* f_t_7 es igual a este */
float x;
{return(sin(x));}

float f_t_4(x)
float x;
{return(cos(2.*x));}

float f_t_5(x)
float x;
{return(sin(2.*x));}
```

I.12 GENERACION DE INTERFEROGRAMAS (Int.c)

Este programa es usado para generar un interferograma de Seidel de tercer orden

```
#include <dos.h>
#include <math.h>
#include <stdlib.h>
#include <conio.h>
#include "agraficas.h"
#define  NMAX 256
#define  NMAX2 320

int key()
{union REGS regs;
 int c;
 regs.h.ah=0x06;
 int86( 0x21, &regs, &regs );
 c = regs.h.al;
 return(c);}

void main(int argc,char *argv[])
{unsigned char nc[NMAX],pedazo[NMAX],ab[7][6];
 char st[17];
 int ck,k=1,i,j,n=0,by;
 float pi,w, co, x,y,r2,C[7],CC[7];
 int tono=1000,dura=500;
 FILE *fp,*fp1;
 if(argc!=2)
  {fprintf(stderr,"USO: Int Archivo-imagen\n");
   exit(1);}
 for(ck=0;ck<7;ck++)  C[ck]=0.0;
 agraficas();
 while(k)
  {for(i=0;i<6;i++)
    {CC[i]=C[i];
    if(fabs(C[i])<0.0001)
      CC[i]=0;
    gcvt(CC[i],4,ab[i]);
escribe("@@@@@@@@@@@",260,20+20*i,0);
    escribe(ab[i],270,20+20*i,67);
   }
  escribe("Esferica",260,10,63);
  escribe("Coma",260,30,63);
  escribe("Astigmat.",260,50,63);
  escribe("Defoco",260,70,63);
  escribe("Tilt x",260,90,63);
  escribe("Tilt y",260,110,63);
  ck = key(); switch(ck)
  {case 27: k=0;    /* Tecla <Esc> */
            break;
   case 80: C[n]-=.1; /* Tecla <Flecha arriba>
            break;
   case 72: C[n]+=.1; /* Tecla < Flecha abajo>
            break;
   case 77: n++;  /* Tecla <Flecha derecha> */
            if(n>5)  n=5;
            break;
case 75: n--;  /* Tecla <Flecha izquierda>
            if(n<0)  n=0;
            break;
   case 43:  /* Tecla <Gris +> */
            fp1=fopen("puente.dat","wb");
            for (i=0;i<NMAX;i++)
             {x=(i-NMAX)/(NMAX/2);
              for (j=0; j<NMAX; j++)
                 {y=(j-NMAX)/(NMAX/2);
                  r2=x*x+y*y;
                  w=C[0]*r2*r2+C[1]*y*r2;
                  w+=C[2]*(r2+2.*y*y);
                  w+=C[3]*r2+C[4]*y+C[5]*x;
                  co=cos(w/(2*M_PI));
                  if(i<200)
                   scrn[i*NMAX2+j]=63.*co*co;
                  else   pedazo[j]=63.*co*co;}
                   if(i>=200)
                   fwrite(pedazo,1,NMAX,fp1);
                  }
                  fclose(fp1);
                  sound(tono);delay(dura);
                  nosound();
                  break;
   case 82:  /* Tecla <Insert> */
                  fp=fopen(argv[argc-1],"wb");
                  fp1=fopen("puente.dat","rb");
                  for(i=0;i<NMAX;i++)
                   {if(i>=200)
                     fread(pedazo,1,NMAX,fp1);
                     for(j=0;j<NMAX;j++)
                      if(i<200)
                     nc[j]=(int)(scrn[i*NMAX2+j]);
                       else
                         nc[j]=(int)(pedazo[j]);
                     fwrite(nc,1,NMAX,fp);}
                  fcloseall();
                  remove("puente.dat");
                  sound(tono);delay(dura);
                  nosound(); break;
   default: break;}
 }
 modeset(3);}
```