UNIVERSIDAD DE GUANAJUATO

Centro de Investigaciones en Optica, A.C.

"AUTOMATIZACION DE LA PRUEBA DE RONCHI"

TESIS

Que para obtener el grado de:

MAESTRO EN OPTICA - TECNOLOGIA OPTICA

Presenta :

Noe Alcala Ochoa

Leon, Gto. Marzo de 1990

... a mi madre.

... a mi padre Raúl
a mis hermanos Beatriz,
Judith, Claudio, Nahum y Adán.

...a mis compañeros Rosario,
Vicente, Gloria, Joel, Apolinar
y Julio, quienes con el tiempo
se convirtieron en inestimables
amigos.

AGRADECIMIENTOS

Quiero expresar mi agradecimiento a las autoridades académicas y administrativas del Centro de Investigaciones en Óptica, A.C. por las facilidades que me otorgaron para que este proyecto de tésis pudiera ser concluido. Quiero agradecer a todo el personal de Investigación y de tecnología, especialmente al de desarrollo tecnológico por su colaboración en la elaboración de este trabajo. Deseo agradecer tambien al Consejo NAcional de Ciencia y Tecnología por haberme otorgado una beca para realizar mis estudios de Postgrado y elaborar mi tésis.

Agradezco al M.C. Abundio Dávila por haber dirigido esta tésis y por su paciencia en la revisión y corrección de la misma.

Al Dr. Enrique Landgrave por sus atinadas sugerencias en el desarrollo de este trabajo y por sus consejos personales.

Agradezco de manera muy especial al Dr. J.J. Sánchez Mondragón por el apoyo moral y los consejos que me brindó durante esta etapa de mi formación profesional.

INDICE

	Pag.
DEDICATORIA	
AGRADECIMIENTOS	
LISTA DE FIGURAS	3
INTRODUCCION	5
CAPITULOS	
I.- LA PRUEBA DE RONCHI.	7
1.1.- Descripción de la prueba de Ronchi.	7
1.2.- La función de aberración de un frente de onda.	11
1.3.- Obtención de las aberraciones transversales a partir de dos ronchigramas.	14
II.- REDUCCIÓN DE DATOS EN LA PRUEBA DE RONCHI.	17
2.1.- Simulación de la prueba de Ronchi.	17
2.2.- Método de reducción de datos.	21
2.3.- Estimación del error introducido por el método de reducción de datos.	23
III - DESCRIPCIÓN DEL PROCESO AUTOMÁTICO Y RESULTADOS EXPERIMENTALES.	27
3.1.- Captura y digitalización de ronchigramas.	27
3.2.- Tratamiento de los ronchigramas por medio de las técnicas de visión por computadora.	33
3.2.1.- Suavizamiento del ronchigrama por promedios de vecindades.	33
3.2.2.- Esqueletizado.	35
3.2.3.- Adelgazado.	38
3.2.4.- Muestreo de datos.	39
3.3.- Resultados experimentales.	39
IV.- CONCLUSIONES	49

APENDICE 1 *50*

A.1.- Explicación de los programas. *50*

A.2.- Listado de los programas. *55*

BIBLIOGRAFIA *113*

LISTA DE FIGURAS

1.1 Arreglo usado en la prueba de Ronchi. 8

1.2 Aberraciones $w(x,y)$ de un frente de onda $\sum$ y aberraciones transversales $TA(x,y)$ de sus rayos.

1.3 Esquema de la rejilla de Ronchi mostrando el angulo de giro ϕ y el eje de referencia TAy. 13

1.4 Convenciones de signos que empleamos para muestrear los ronchigramas vertical y horizontal. 13

2.1 Esquema del trazo exacto de rayos en la prueba de Ronchi. 19

2.2 Simulación de ronchigramas de una superficie parabólica que posee deformaciones del tipo de la ec. 2.3 con los coeficientes de aberración $A=B=C=1\lambda$ y $D=5\lambda$. 25

2.3 Tabla de errores introducida por el método de reducción de datos. 26

3.1 Arreglo para capturar un ronchigrama. 29

3.2 Diagramas de dos configuraciones posibles en el probador de Ronchi. (a) para pruebas fuera de eje. (b) para pruebas en eje. 32

3.3 Vecindad utilizada en el proceso de suavizamiento. 34

3.4 Vecindad de 5x5 pixeles usada en el algoritmo de esqueletización. 34

3.5 Direcciones de detección en el algoritmo de esqueletización. 37

3.6 Vecindad utilizada en el algoritmo de adelgazamiento. 37

3.7 Ronchigramas reales capturados con el programa SEMPER. 42

3.8 Ronchigramas esqueletizados como resultado de aplicar el procedimiento de la secc. 3.3.2 (programas FILBAJ y ESKEL). 43

3.9 Ronchigramas adelgazados siguiendo el procedimiento de la secc. 3.3.3. (programa THIN). 44

3.10 Figura tridimensional de la superficie bajo prueba despues de aplicar el método de reducción de datos (secc. 2.2) (programa GRA_PRU). 46

3.11 Perfil de la superficie en el plano z-x. 47

3.12 Perfil de la superficie en el plano z-y. 47

INTRODUCCION

Este trabajo es el resultado de un estudio experimental sobre la automatización de la prueba de Ronchi. La motivación para desarrollarlo surgió de la necesidad de mejorar la calidad de los espejos primarios que se fabrican en nuestro taller óptico para su posterior incorporación en telescopios astronómicos.

No se pretendió obtener una calidad perfecta, ya que esto significaría el poseer la capacidad de generar superficies ópticas reales identicas a las ideales. En la practica se desea obtener sólo una calidad aceptable. Esta se logra al aproximar la superficie real hasta que el promedio de sus desviaciones (con respecto a la ideal) sea menor a cierto límite tolerable. La detección y cuantificación de estas desviaciones puede llevarse a cabo usando una gran diversidad de técnicas, que se seleccionan dependiendo del grado de precisión que se desee. Sin embargo en las primeras etapas del pulido de la superficie, una técnica muy utilizada por los expertos en fabricación optica ha sido la interpretación visual del patrón de franjas que resulta de la prueba de Ronchi (Ronchigrama). Esta prueba se utiliza en algunos casos hasta las ultimas etapas de pulido, en las que resulta ya difícil detectar visualmente pequeños cambios en el ronchigrama.

El proposito de este trabajo es el de evitar la interpretación visual de la prueba de Ronchi usando las tecnicas conocidas como "tecnicas de visión por computadora". Ellas nos permiten hacer mediciones sobre las imagenes de los ronchigramas, y a partir de estas obtener las desviaciones de la superficie real con respecto a la ideal.

Nuestro punto de partida fue la revisión de la teoría geométrica de la prueba de Ronchi que se usó posteriormente para la generación de ronchigramas ideales. Esto con el fin de utilizarlos como patrones de referencia en la detección de errores o desviaciones introducidas por el metodo numérico de ajuste. Para nuestros propósitos esta etapa aseguró la confiabilidad del proceso de ajuste.

Una vez probada la confiabilidad de este proceso, se procedio a capturar las imágenes de los Ronchigramas reales y enseguida aplicar las técnicas de visión por computadoras para detectar la posición de los máximos centrales de cada franja. Finalmente se consideraron los errores que pueden ser introducidos por la lente de la cámara y la geometría del arreglo sensor de imagenes, obteniéndose con esto una buena repetibilidad en la cuantificación de las desviaciones.

En el *capitulo 1* se presenta una revisión de la teoría geométrica de la prueba de Ronchi y su relación con el frente de onda. En el *capitulo 2* se hace una simulación de la prueba de Ronchi en la cual se generan digitalmente ronchigramas y se emplea un método de reducción de datos para recobrar la superficie inicial que los originó. Este proceso lo usamos para estimar la precisión de los algoritmos que utilizamos en la reducción de datos reales. En *el capitulo 3* se describen los metodos de procesamiento de imagenes que utilizamos para procesar los ronchigramas. En la sección de conclusiones discutimos las ventajas y limitaciones de este método, señalando algunas formas en que podríamos mejorarlo en el futuro. Finalmente en el *apendice 1* se explican brevemente los programas más importantes que utilizamos y se lista su contenido.

CAPITULO 1

LA PRUEBA DE RONCHI

Existen dos modelos teoricos para explicar la formación de las franjas en la prueba de Ronchi. El modelo geométrico, que las explica como el resultado de la proyección de las lineas claras y obscuras que forman una rejilla binaria, y el modelo físico, que explica la formación de las franjas como el resultado de la difracción en una rejilla binaria. Sin embargo, para rejillas cuya frecuencia sea del orden de *30 lin/cm*, ambos modelos llegan al mismo resultado. En este trabajo nos restringiremos exclusivamente al tratamiento geométrico de la prueba.

En la primera sección de este capítulo se presenta una breve descripción de la Prueba de Ronchi usando el arreglo típico para llevarla a cabo y mencionando los elementos que la componen. Tambien incluiremos aqui una relación de las ecuaciónes que ligan las aberraciones transversales con las aberraciones longitudinales, y planteremos a partir de ellas un sistema de ecuaciones diferenciales para calcular la función de aberración del frente de onda a partir de datos obtenidos de dos ronchigramas. Estas ecuaciones seran nuestro punto de partida en el *capitulo 2.*

1.1 DESCRIPCION DE LA PRUEBA RONCHI

La Prueba de Ronchi es una prueba de precisión moderada, que resulta apropiada para controlar la calidad de una superficie óptica en las primeras etapas de pulido. Esta prueba se puede hacer tan complicada como se desee, pero en una de sus versiones más simples

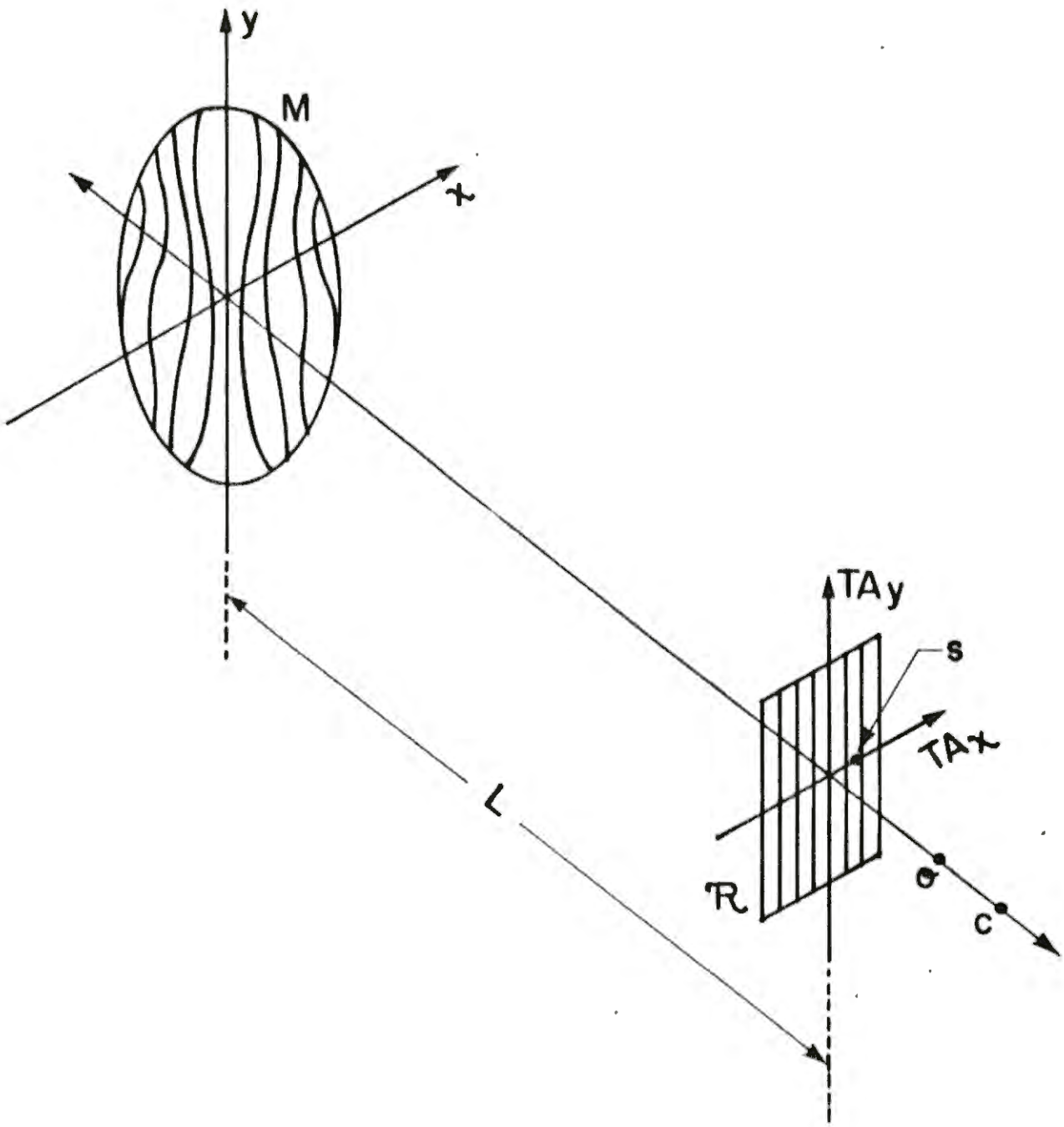


Fig. 1.1 Arreglo usado en la prueba de Ronchi.

requiere de muy pocas componentes: una fuente puntual de luz incoherente, una rejilla de lineas rectas de baja frecuencia (aproximadamente *30 lin/cm*), y desde luego la superficie bajo prueba.

La fig. 1.1 muestra como se pueden arreglar estas componentes. A una distancia *L* del vértice del espejo bajo prueba *M*, y cerca de su centro de curvatura c, colocamos la rejilla *R* iluminada por la fuente puntual *s*. Si observamos en la posición *O* veremos sobre el espejo un patrón de franjas claras y obscuras de cuya interpretación inferiremos la forma de la superficie.

Cornejo *(1978)* proporciona una gran cantidad de ronchigramas que se pueden utilizar como patrones para realizar una interpretación cualitativa de un ronchigrama real. Estos patrones de franjas son relativamente fáciles de interpretar visualmente si utilizamos una rejilla de lineas rectas y la superficie bajo prueba es esférica. Si sustituimos sin embargo la superficie esférica por una asférica, los patrones de franjas se tornan mas irregulares y su interpretación se dificulta.

Antes de deducir geométricamente la ecuación diferencial que explica la formación de las franjas en la prueba de Ronchi, y de plantear el sistema de ecuaciones diferenciales que resolveremos para cuantificar las deformaciones de una superficie a partir de su Ronchigrama, es conveniente hacer una revisión de los conceptos de aberración de un frente de onda, de las aberraciones transversales de los rayos y de la ecuación diferencial que los relaciona.

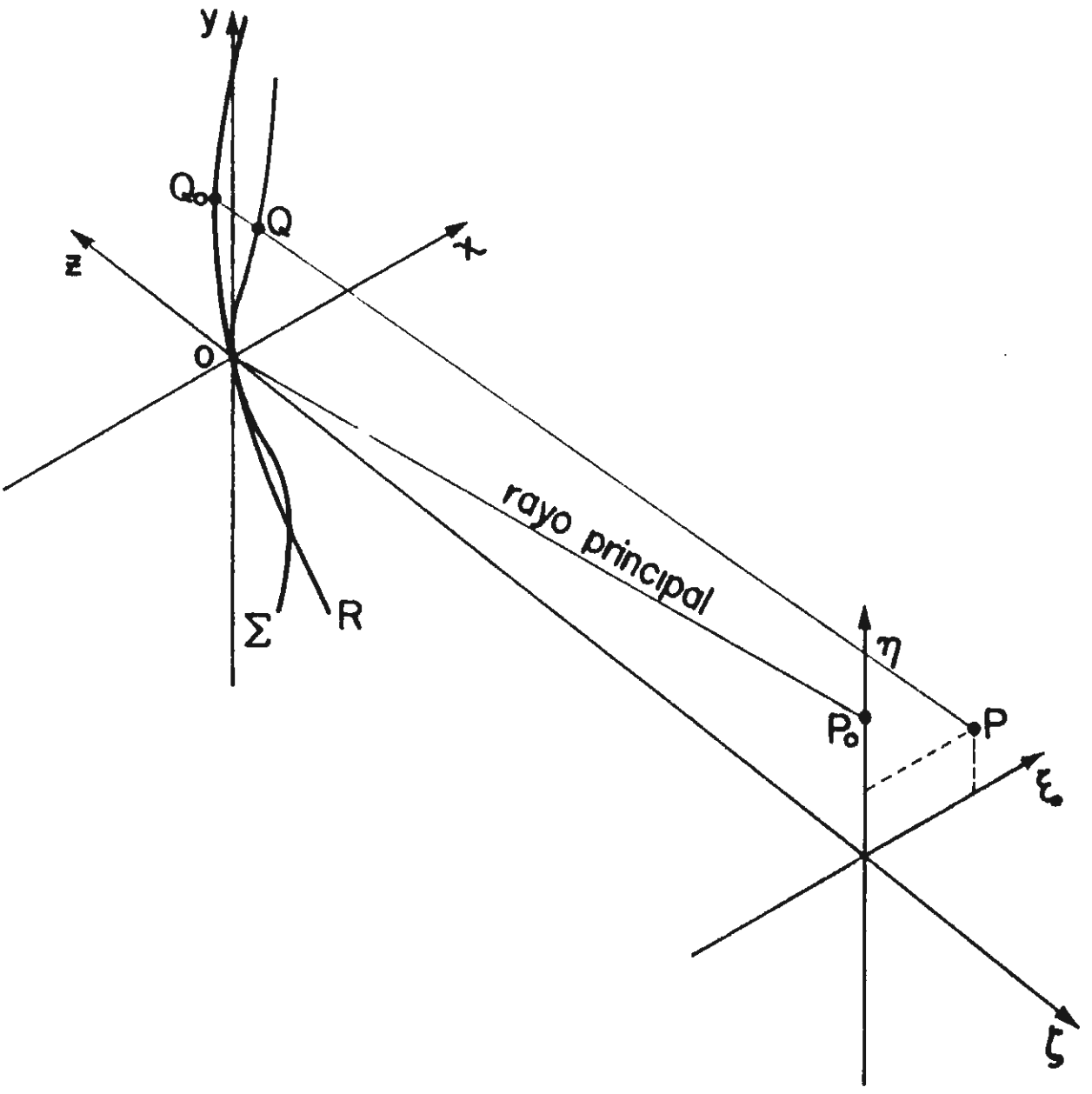


fig. 1-2. Aberraciones w(x,y) de un frente de onda $\sum$ y aberraciones transversales TA(x,y) de los rayos.

1.2 LA FUNCIÓN DE ABERRACIÓN DE UN FRENTE DE ONDA.

Consideremos un sistema óptico con simetría axial iluminado por una fuente puntual. La *fig. 1-2* muestra el frente de onda $\sum$.

Si suponemos que la fuente de iluminación se colocó en el plano $y'z'$ del espacio objeto, el rayo principal, que determina el origen de coordenas del sistema xyz en la pupila de salida, intersectará al plano de observación ξ-η en el punto P_0 sobre el eje η. Sean $(0, \eta_0)$ las coordenadas de este punto, y (ξ, η) las coordenadas del punto de intersección P de otro rayo cualquiera $\hat{r}$ con el plano ξ-η de observación. Definimos entonces las aberraciones transversales de este rayo como

$$TAx(x,y) = \delta\xi(x,y) = \xi \quad , \qquad (1.1)$$

$$TAy(x,y) = \delta\eta(x,y) = \eta - \eta_0 \quad , \qquad (1.2)$$

Para definir la aberración del frente de onda $w(x,y)$, consideremos una esfera R centrada en el punto P_0 y de radio $\overline{P_0O}$, donde O es el origen de coordenadas del plano x-y *(fig.1-2)*. Si el rayo $\hat{r}$ intersecta a R en el punto Q_0 y a $\sum$ en el punto Q, definimos la aberración del frente de onda $\sum$ como el camino óptico $\overline{QoQ}$, esto es

$$w(x,y) = \overline{Q_0Q} \qquad (1.3)$$

Notemos que la aberración $w(x,y)$ será positiva en el punto Q si la distancia $\overline{PQ}$ es menor que la distancia $\overline{PQ}_0$, es decir, si la esfera de referencia R se encuentra "detras" del frente de onda prueba Σ., tomando com referencia la dirección de propagación de la luz.

Podemos relacionar la aberración $w(x,y)$ con las aberraciones transversales $TAx(x,y)$ y $TAy(x,y)$ usando las ecuaciones de Rayces *(1964)*.

$$\frac{\partial w(x,y)}{\partial x} = - \frac{TAx}{\overline{P_0O} - w(x,y)} \quad , \qquad (1.4a)$$

$$\frac{\partial w(x,y)}{\partial y} = - \frac{TAy}{\overline{P_0O} - w(x,y)} \quad , \qquad (1.4b)$$

que podemos aproximarlas como

$$\frac{\partial w(x,y)}{\partial x} = - \frac{TAx}{L} \quad , \qquad (1.5a)$$

$$\frac{\partial w(x,y)}{\partial y} = - \frac{TAy}{L} \quad , \qquad (1.5b)$$

donde L es la proyección de $\overline{P_0O}$ sobre el eje optico Z.

Conviene señalar que se pueden definir las aberraciones de un frente de onda referidas no a una esfera, sino a una superficie arbitraria $s(x,y)$. Veamos como se modifican las *ecs. 1.5* cuando medimos las aberraciones del frente de onda $\sum$ respecto a la superficie $s(x,y)$.

Supondremos que $w_s(x,y)$, $TAx_s(x,y)$ y $TAy_s(x,y)$ son las aberraciones de la superficie $s(x,y)$ y que $w(x,y)$, $TAx(x,y)$ y $TAy(x,y)$ son las aberraciones del frente de onda $\sum$, tomando en ambos casos la esfera R como superficie de referencia. Si sustituimos estas expresiones en la *ec. 1.5* y restamos las ecuaciones correspondientes obtenemos las siguientes relaciones:

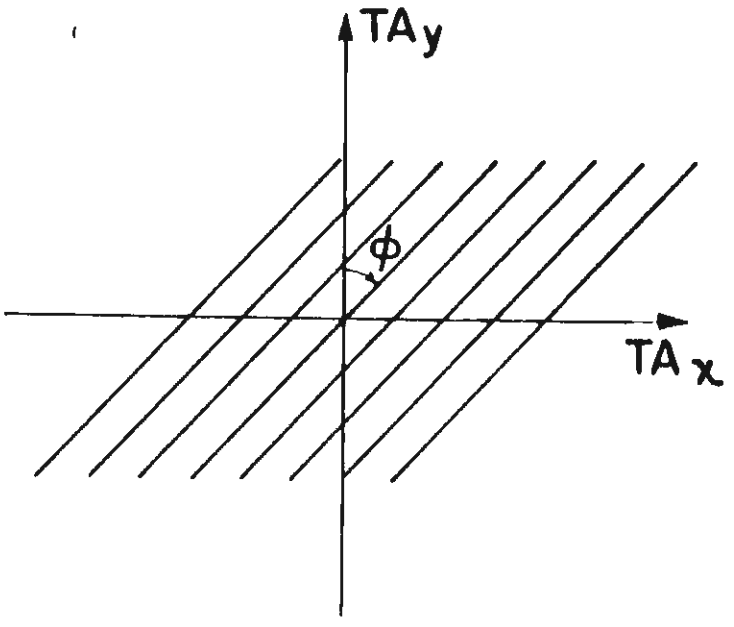
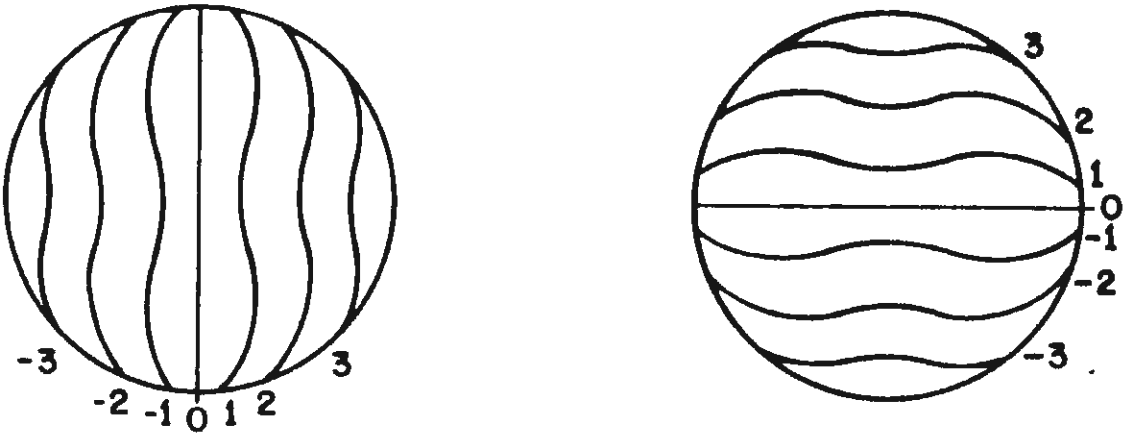


fig. 1-3 Esquema de la rejilla de Ronchi mostrando el angulo de giro ϕ y el eje de referencia TAy.



(a) *(b)*

Fig. 1.4 Convenciones de signos que empleamos para muestrear los ronchigramas. *(a)* vertical *(b)* horizontal.

$$\frac{TAx - TAx_s}{L} = - \frac{\partial}{\partial x} (w - w_s) , \qquad (1.6a)$$

$$\frac{TAy - TAy_s}{L} = - \frac{\partial}{\partial y} (w - w_s) . \qquad (1.6b)$$

Se puede observar que estas expresiones son equivalentes a las *ecs. 1.5* salvo que ahora las interpretaremos como las aberraciones del frente de onda $\sum$ pero referidas a la superficie arbitraria $s(x,y)$.

1.3 OBTENCIÓN DE LAS ABERRACIONES TRANSVERSALES A PARTIR DE DOS RONCHIGRAMAS.

En esta sección describiremos las ecuaciones básicas *(Malacara, 1965)*, que nos permiten cuantificar las aberraciones transversales que posee un sistema óptico a través de mediciones en las franjas de los ronchigramas. Las irregularidades en una franja se interpretan como la diferencia entre las aberraciones transversales correspondientes al frente de onda real $\sum$, y a una esfera de referencia con centro en la posición de la rejilla. Estas irregularidades dependen del angulo ϕ que posee la rejilla al momento en que se realiza la prueba *(fig. 1-3)*.

En el caso de colocar la rejilla a cero grados ($\phi=0^0$), podemos determinar unicamente las aberraciones transversales en la dirección de TAx, mediante la ecuación

$$TAx(x,y) = m_x d , \qquad (1.7a)$$

donde d es la frecuencia de la rejilla y m_x es un numero entero que

indica la *m-ésima* franja sobre el espejo que contiene al punto *(x,y)*. Este punto no es elegido arbitrariamnete sobre la franja, sino que es el punto de máxima intensidad en ella. En el *capitulo 3* explicamos un método para encontrar los puntos de máxima intensidad en cada franja.

Un resultado similar se obtiene al girar la rejilla a un ángulo de 90^0 para encontrar las aberraciones transversales en la dirección *TAy*.

$$TAy(x,y) = m_y d \ . \qquad (1.7b)$$

La elección de los signos de las franjas no es arbitraria y por ello conviene definir aqui la convencion que utilizamos para reducir los ronchigramas *(fig. 1.4)*.

Para un ronchigrama horizontal obtenido dentro de foco, las franjas que se encuentren situadas en la parte superior de la franja central, o franja cero, las consideraremos positivas; mientras que si se localizan en la parte inferior las consideraremos negativas. Para un ronchigrama vertical serán positivas si se encuentran a la derecha de la franja vertical cero, y negativas en caso contrario.

En general es difícil identificar la franja de orden cero, especialmente cuando el ronchigrama consta de muchas franjas. Para nuestro propósito resultó suficiente definir la franja de orden cero como la franja que mas se acerca a una recta y que constituye el centro de simetría del patrón de franjas.

En lo sucesivo supondremos que las aberraciones transversales *TAx y TAy* se pueden obtener de dos ronchigramas, correspondientes a dos posiciones de la rejilla, perpendiculares entre si *(ecs. 1.7)*.

Ahora es claro que podemos reescribir las ecuaciones *1.5* y *1.6* sustituyendo los valores de *TAx* y *TAy* proporcionados por las ecuaciones *1.7*. *De las ecs. 1.5*

$$\frac{\partial w_e}{\partial x} = - \frac{m_x d}{L} \quad , \qquad (1.8a)$$

$$\frac{\partial w_e}{\partial y} = - \frac{m_y d}{L} \quad , \qquad (1.8b)$$

y de las ecs. 1.6

$$\frac{\partial W(x,y)}{\partial x} = - \frac{(m_x d - TAx_s)}{L} \quad , \qquad (1.9a)$$

$$\frac{\partial W(x,y)}{\partial y} = - \frac{(m_y d - TAy_s)}{L} \quad , \qquad (1.9b)$$

donde además hemos sustituido $W(x,y)$ opr $w-w_s$. Conviene señalar que $TAx_s(x,y)$ y $TAy_s(x,y)$ no siempre se podran expresar como el producto de un numero entero por la frecuencia de la rejilla.

En el capitulo siguiente emplearemos estas ultimas expresiones para reducir los datos obtenidos de una simulación de la prueba de Ronchi.

CAPITULO 2

REDUCCION DE DATOS EN LA PRUEBA DE RONCHI

Como en caulquier procedimiento de reducción de datos, resulta necesario comprobar los resultados que se obtienen por los algoritmos que se proponen. En nuestro caso la comprobación se implementó en *2* etapas: La primera consitió en la generación de ronchigramas ideales, y la segunda en su utilización como patrones de prueba en la detección de errores que introduce el metodo de ajuste.

En las primeras dos secciones se describen las etapas mencionadas, partiendo de los resultados del capitulo anterior, y en la ultima sección se presentan las desviaciones o errores que introduce el metodo de ajuste utilizando superficies que poseen diferentes tipos de aberraciones y razones focales.

2.1 SIMULACIÓN DE LA PRUEBA DE RONCHI.

Simularemos la prueba de Ronchi a partir de un arreglo óptico como el de la *fig. 1-1,* y expresaremos la forma del espejo por la siguiente ecuación

$$Z(x,y) = Z_o(x,y) + D_o(x,y) \ , \qquad (2.1)$$

en donde $D_o(x,y)$ es una función de deformación y $Z_o(x,y)$ es una superficie de revolución. Por lo general Z_o se puede expresar en su forma más común por

$$z_o(x,y) = \frac{c\,s^2}{1 + [\ 1-(k+1)c^2s^2\ ]}\ , \qquad (2.2)$$

donde $s^2=x^2+y^2$, c es el inverso del radio paraxial de curvatura del espejo bajo prueba y k es la constante de conicidad.

Por otro lado la función $Do(x,y)$ puede ser muy complicada, pero para nuestros propósitos resulta suficiente definirla como

$$Do(x,y) = A(x^2+y^2) + By(x^2+y^2) + c(x^2+3y^2) + D(x^2+y^2), \qquad (2.3)$$

donde podemos identificar a las aberraciones primarias como funciones de deformación.

El cálculo tradicional para generar un ronchigrama utiliza una superficie de revolución *(Malacara, 1978)*, sin embargo, según la *ec. 2.3* nuestra superficie no cumple con esa condición. Por esta razón y con el fin de no restringir aún mas la función de deformación $Do(x,y)$, nosotros utilizamos una técnica desarrollada con anterioridad que nos permite encontrar el ronchigrama a partir del trazo exacto de rayos *(Alcalá, 1988)*. Aunque esta técnica se desarrollo para la reducción de datos en la prueba de Hartmann, su aplicación a la prueba de Ronchi resulta inmediata.

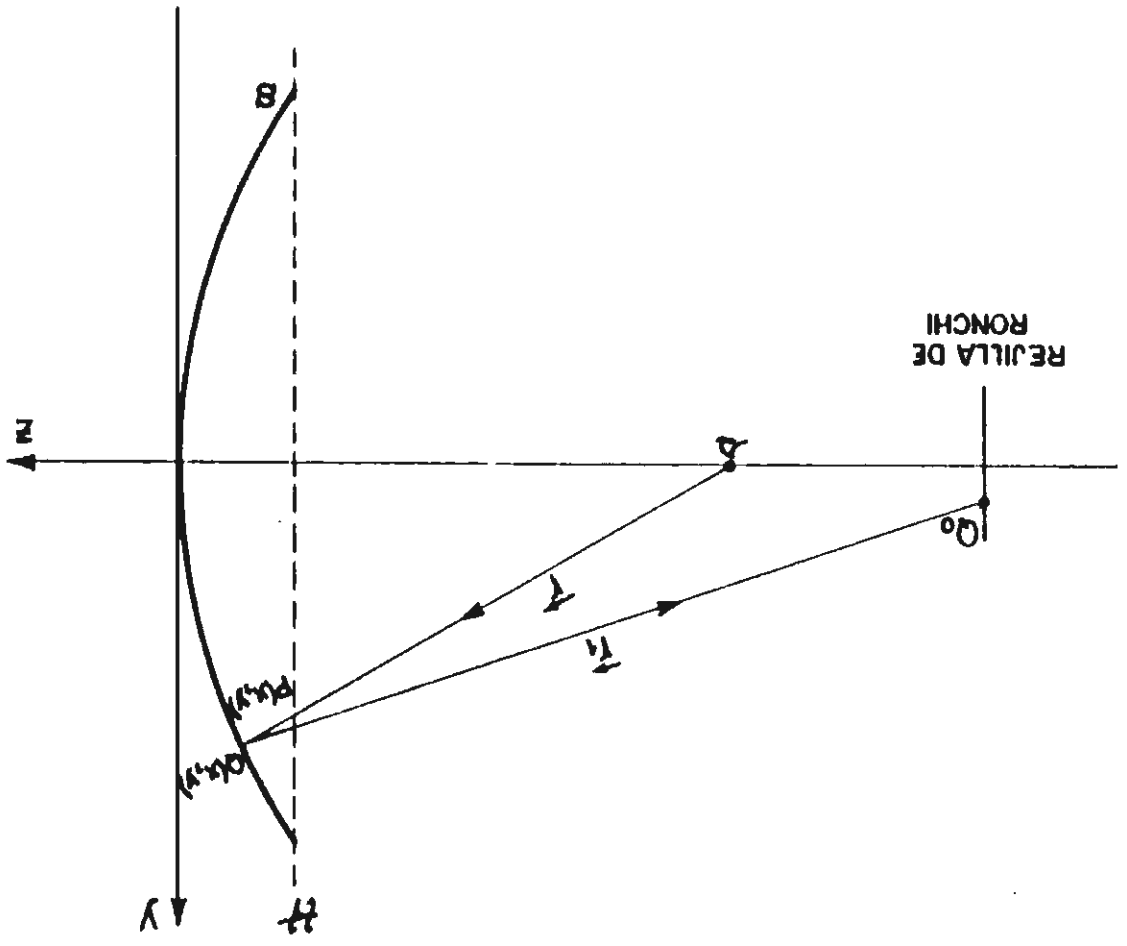


Fig. 2-1 Esquema del trazo exacto de rayos en la prueba de Ronchi.

El metodo de trazo de rayos se puede resumir en los siguientes cuatro pasos, referidos a la *fig. 2.1.*

i).- Transferencia de un rayo r de la fuente puntual σ a un punto $P(x,y)$ de un plano H tangente al borde de la superficie S.

ii).- Transferencia del rayo del punto $P(x,y)$ al punto $Q(x,y)$ en la superficie S bajo prueba;

iii).- Reflexión del rayo en la superficie S; y

iv).- Transferencia del rayo de la superficie S al punto Q_o de la rejilla.

Utilizamos las aberraciones transversales $(TAx(Q)$ y $TAy(Q))$ de los rayos reflejados $(\vec{r}_1)$ para decidir si estos incidieron en posiciones que debieran ser de máxima intensidad. Por ejemplo si colocamos la rejilla horizontal y deseamos conocer las lineas de máxima intensidad del ronchigrama, hacemos trazo exacto de rayos y localizamos los puntos $Q(x,y)$ que cumplen la siguiente relación

$$| TAx(x,y) - md | < \varepsilon \qquad (2.4)$$

donde m es un número entero, d es la frecuencia de la rejilla y ε es una constante mucho menor que uno.

Este proceso lo aplicamos dos veces: primero colocando la "rejilla" a *cero grados* y posteriormente *a noventa grados (secc. 1.4).*

Una vez que hemos simulado los ronchigramas con el proceso anterior procedemos a muestrearlos, esto es, a crear dos archivos con los datos de los ronchigramas horizontal (x,y,m_x) y vertical (x,y,m_y). En el *capitulo 3* explicamos con más detalle el algoritmo que utilizamos para realizar este muestreo.

2.2 MÉTODO DE REDUCCIÓN DE DATOS.

A partir de los dos archivos de datos que obtuvimos en la sección anterior calcularemos las aberraciones transversales $TA_x(x,y)$ y $TA_y(x,y)$ usando las *ecs. 1.7*. Con esta información utilizaremos un método de reducción de datos para resolver las *ecs. 1.9*, y obtener asi la función de aberración del frente de onda. Más aún, obtendremos la función de deformación $Do(x,y)$ de la superficie, mediante la siguiente aproximación.

$$D(x,y) \cong w(x,y)/2 \qquad (2.5)$$

El método de reducción de datos consiste en resolver el sistema de *ecuaciones 1.9* en el sentido de minimos cuadrados *(Cornejo,1976)*, el cual explicaremos brevemente a continuación

Supongamos que la superficie óptica bajo prueba es lo suficientemente suave para que su frente de onda $w(x,y)$ se pueda representar adecuadamente por un polinomio bidimensional de cierto grado k. Entonces

$$w(x,y) = \sum_{i=0}^{k} \sum_{j=0}^{i} B_{ij}\, x^j\, y^{i-j} \quad , \qquad (2.6)$$

Si derivamos $w(x,y)$ respecto a x y a y,

$$\frac{\partial w(x,y)}{\partial x} = \sum_{i=0}^{k-1} \sum_{j=0}^{i} (j+1) B_{i+1,j+1} \; x^j \; y^{i-j} \quad , \qquad (2.7a)$$

$$\frac{\partial w(x,y)}{\partial y} = \sum_{i=0}^{k-1} \sum_{j=0}^{i} (i-j+1) B_{i+1,j} \; x^j \; y^{i-j} \quad , \qquad (2.7b)$$

que podemos reescribirlas como

$$\frac{\partial w(x,y)}{\partial x} = \sum_{i=0}^{k-1} \sum_{j=0}^{i} C_{ij} \; x^j \; y^{i-j} \quad , \qquad (2.8a)$$

$$\frac{\partial w(x,y)}{\partial y} = \sum_{i=0}^{k-1} \sum_{j=0}^{i} D_{ij} \; x^j \; y^{i-j} \quad , \qquad (2.8b)$$

donde

$$B_{ij} = \frac{C_{i-1,j-1}}{j} \quad , \qquad \begin{matrix} i = 1,2,\ldots,k \\ j = 1,2,..,i \end{matrix} \qquad (2.9a)$$

y

$$B_{ij} = \frac{D_{i-1,j}}{i-j} \quad , \qquad \begin{matrix} i = 1,2,\ldots,k \\ j = 0,1,\ldots,i-1 \end{matrix} \qquad (2.9b)$$

Para conseguir mayor precisión calcularemos los coeficientes B_{ij} como un promedio de las *ecs. 2.9a y 2.9b*.

Excepto por el coeficiente B_{00}, de las *ecs. 2.8, 2.9, y 2.6* podemos obtener la expresión para el frente de onda $w(x,y)$ en toda la pupila. Conociendo $w(x,y)$, la función de deformación $D(x,y)$ la obtenemos usando la relación *2.5*.

Notemos que hasta aqui no hemos calculado los terminos TAx_s y TAy_s de la expresión *1.9*; es decir, nos falta determinar las aberraciones transversales de la superficie ideal. Sin embargo, ahora que conocemos los ronchigramas del espejo deformado, resulta relativamente fácil hacerlo. Basta con muestrear las franjas, y aplicar a cada punto muestreado los pasos *iii) y iv)* del proceso de trazo exacto de rayos descrito en la sección anterior. Las intersecciones de los rayos reflejados por la superficie ideal $Z_o(x,y)$ con el plano de la rejilla nos darán los valores de las aberraciones transversales TAx_s y TAy_s. Las TAx_s se obtendran si la rejilla se coloca verticalmente, y las TAy_s si se coloca horizontalmente.

2.3 ESTIMACIÓN DEL ERROR INTRODUCIDO POR EL MÉTODO DE REDUCCIÓN DE DATOS.

Finalmente, obtenemos el error que introduce el método de reducción de datos comparando la función de deformación introducida $D_o(x,y)$ *(ec. 2.3)* con la calculada $D(x,y)$, tomando como criterios las desviaciones *RMS* y *PICO-PICO*. Estos los definimos como sigue:

$$RMS = \left[\sum_{j=1}^{N} \sum_{i=1}^{N} \left(D(x_i,y_j) - D_o(x_i,y_j) \right) / (N \times N) \right]^{1/2}, \qquad (2.10)$$

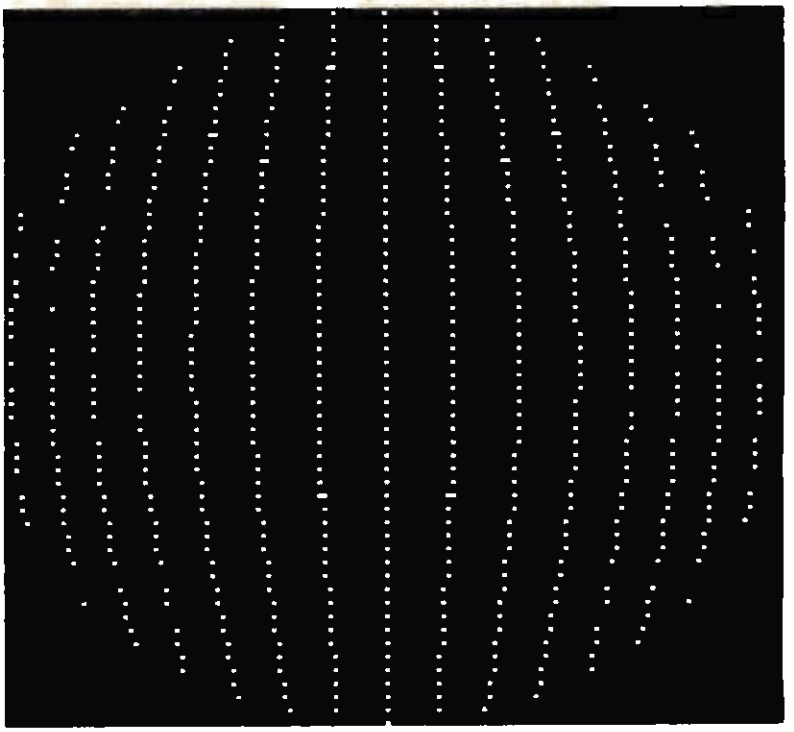
$$PICO = Max\ (\ D(x_i, y_j) - D_o(x_i, y_j)\ )\ , \qquad (2.11a)$$

$$VALLE = Min\ (\ D(x_i, y_j) - D_o(x_i, y_j)\ ), \qquad (2.11b)$$

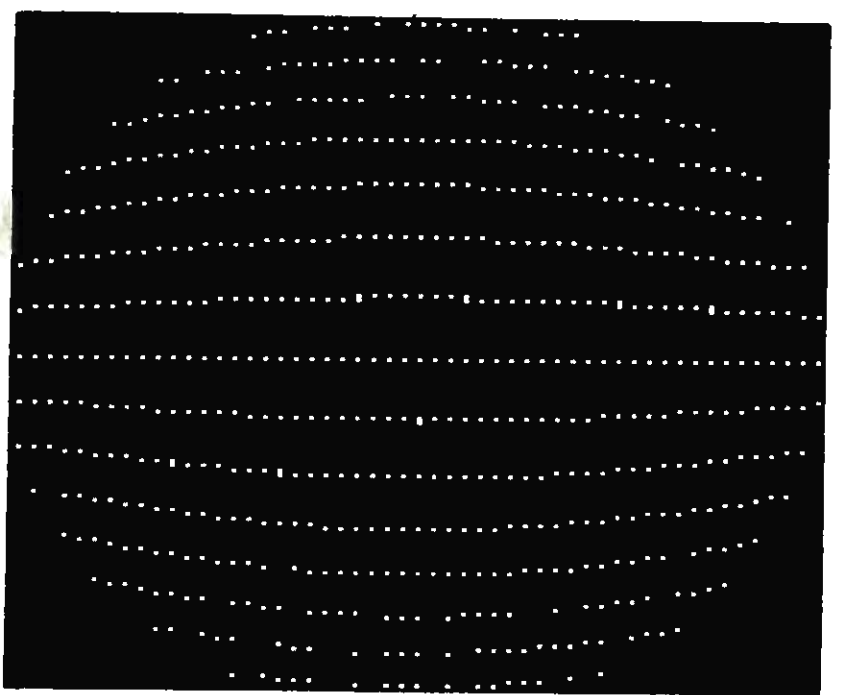
$$PICO\text{-}PICO = |PICO| + |VALLE|\ , \qquad (2.11c)$$

donde N es el numero de puntos (x_i) que utilizamos para calcular el error. Estos puntos no son los muestreados, sino que generamos una malla de puntos equiespaciados en el espejo, y en estos puntos evaluamos las funciones de deformación $D(x,y)$ y $D_o(x,y)$.

A continuación presentamos algunos ronchigramas generados con el método descrito en la sección anterior, y una tabla de errores para diferentes "espejos" caracterizados por los coeficientes de la función de deformación $D(x,y)$ *(ec.2.3)*, y diferentes razones focales. En esta tabla k es la constante de conicidad de cada superficie, R es el radio de curvatura, DD es el diámetro del espejo y A,B,C,D son los coeficientes de deformación que introducimos *(ec. 2.3)*, expresados en longitudes de onda.



VERTICAL



HORIZONTAL

Fig. 2.2 Simulacion de ronchigramas de una superficie parabolica que posee deformaciones del tipo de la ec. 2.3 ($A=B=C=1\lambda$, $D=5\lambda$).

k	R	DD	f/#	A	B	C	D	PICO	VALLE	RMS
-1	75	15	f/2.5	0	1	1	5	0.31	-0.16	0.11
-1	240	15	f/8	0	1	1	5	0.05	-0.17	0.043
0	240	15	f/8	0	1	1	5	0.02	-0.45	0.104

Fig. 2.3 Tabla de errores introducida por el método de reducción de datos.

CAPITULO 3

DESCRIPCION DEL PROCESO AUTOMATICO Y RESULTADOS EXPERIMENTALES

Como vimos anteriormente, la automatización de la prueba de Ronchi requiere de varias etapas que al integrarlas nos darán como resultado la figura del espejo bajo prueba. En este capítulo se describen cada una de las etapas del proceso automático: la captura y digitalización del ronchigrama, su tratamiento posterior por medio de las tecnicas de visión por computadora y finalmente, haciendo uso de la técnica de reducción de datos *(secc. 2.2)* se presentan los resultados del proceso automático.

En lo que concierne a la etapa de digitalización del ronchigrama se hace énfasis en el equipo idispensable y en algunas consideraciones preliminares a la captura. Por otra parte en la etapa del tratamiento del ronchigrama se describen las técnicas que nos permitieron encontrar las lineas de máxima intensidad en cada franja brillante, así como la técnica implementada para muestrearlas. En la última sección se presentan los resultados de la reducción de los datos muestreados en forma de gráfica tridimensional y de errores RMS y PICO-PICO.

3.1 CAPTURA Y DIGITALIZACIÓN DE RONCHIGRAMAS

El método para capturar y digitalizar los ronchigramas se puede dividir en dos etapas. La primera consiste en la elección preliminar de los elementos que constituyen la prueba y la segunda en la alineación de sus elementos. Como veremos la calidad de las componentes es un factor esencial que implica la necesidad de utilizar algunos procedimientos de calibración *(Lenz, 1987)*, sin embargo, en este trabajo hemos supuesto una calidad suficiente en

las componentes, de modo que nos permitimos omitir estos procedimientos. Por otra parte, los resultados demuestran una buena repetibilidad, lo cual confirma la confiabilidad del proceso.

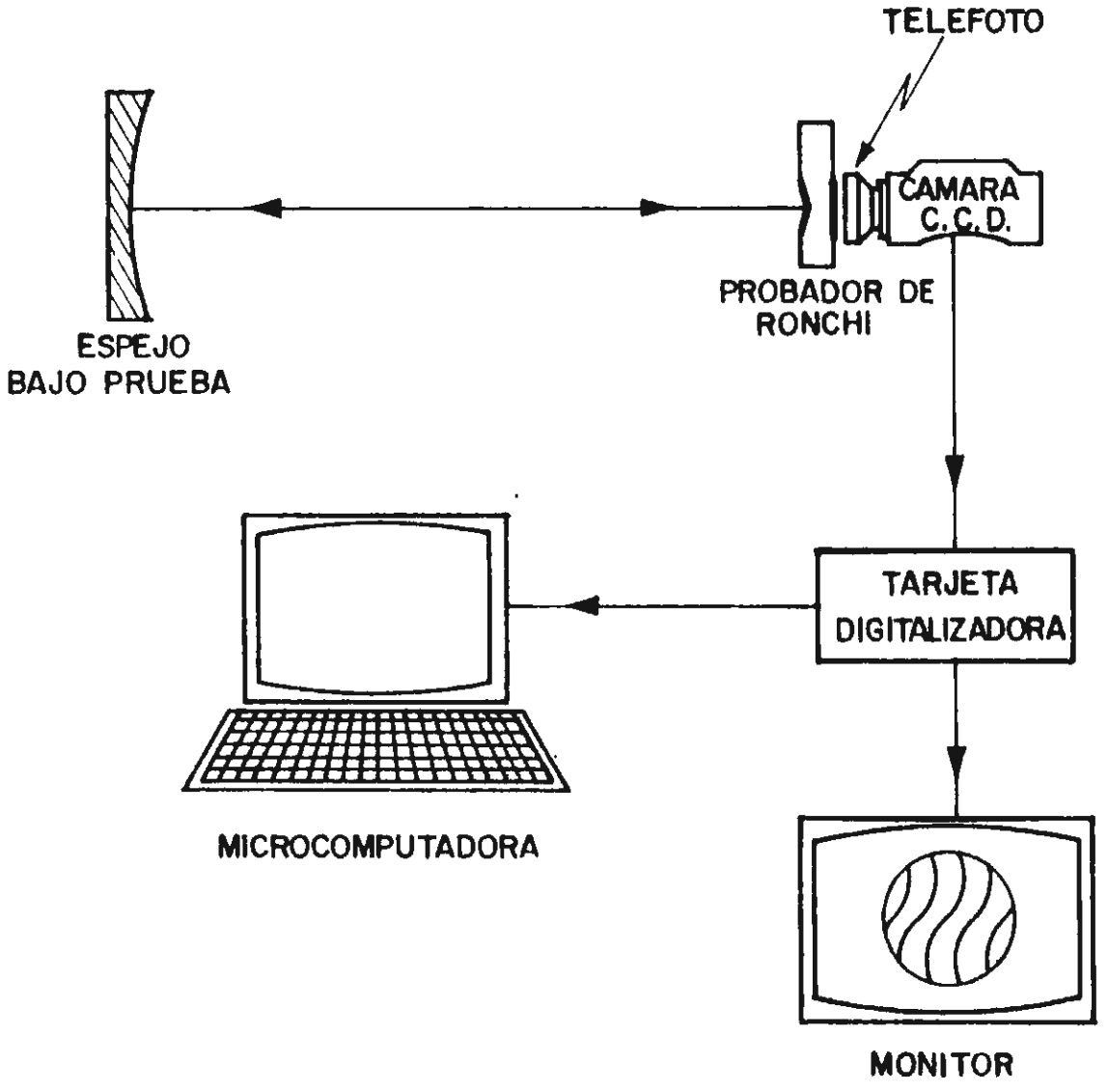


Fig 3.1 Arreglo para capturar un ronchigrama

Como se acaba de mencionar la elección adecuada de los elementos que constituyen la prueba de Ronchi es indispensable para obtener buenos resultados. La *fig. 3.1* muestra el esquema del arreglo que utilizamos para capturar ronchigramas. En él tenemos un espejo bajo prueba que iluminamos con el probador de Ronchi. En la posición donde normalmente se coloca el ojo para observar el patrón de franjas, colocamos una objetivo telefoto acoplado con la *cámara CCD* para amplificar el ronchigrama. Finalmente utilizamos una tarjeta digitalizadora para enlazar la cámara sensora con el computador y observar los patrones de franjas en el monitor de TV. Nosotros utilizamos una cámara *panasonic WV-CD51 de 256x256 pixeles* y un computador *Denki corona ctel-286.*

Es evidente que uno de los elementos más importantes del arreglo anterior es el probador de Ronchi, que se muestra esquematicamente en la *fig. 3.2* en sus dos posibles configuraciones: fuera de eje y en eje. Sus elementos principales son: el sistema de iluminación, la rejilla (o rejillas) de baja frecuencia y la montura móvil en *3* ejes perpendiculares.

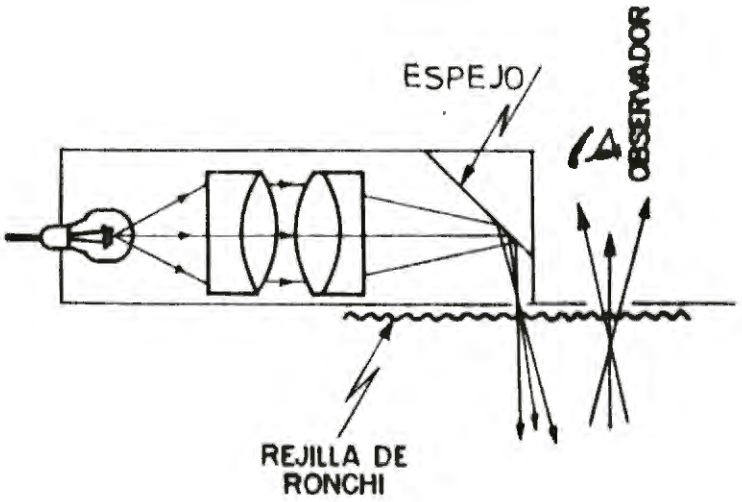
La elección adecuada de la frecuencia de la rejilla es importante para evitar fenómenos de difracción. Al respecto, Cornejo y Malacara *(1970)*, encontraron que el periodo mínimo permisible para evitar estos fenómenos de difracción es

$$p = 10\lambda \left[\frac{R}{D}\right] \left\{ 1 + \left[1 + \frac{\Delta R}{25\lambda (R/D)^2} \right]^{1\backslash 2} \right\}, \quad (3.1)$$

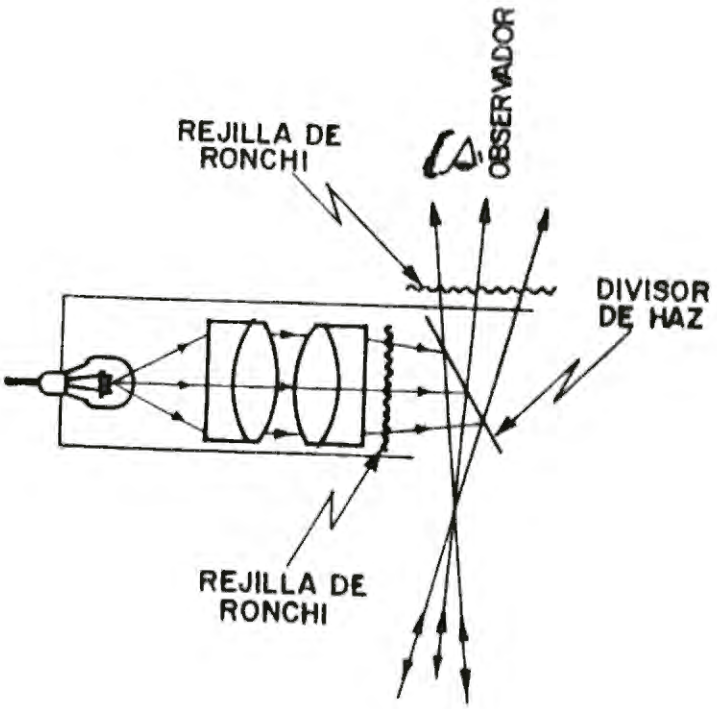
donde R el radio de curvatura paraxial, D es el diámetro del espejo, ΔR la diferencia entre los radios de curvatura paraxial y marginal, y λ la longitud de onda de la luz.

Una vez seleccionada la rejilla y colocada en el el probador se procede a la alineación. Esta se hace manualmente, colocando el probador y el espejo bajo prueba de modo que la luz que emite el probador ilumine al espejo y al ser reflejada por éste atraviese de nuevo la rejilla. Para conseguir esto se observa a través de la rejilla en la posición *C* de la *fig. 3.1* . Posteriormente se coloca el sistema telefoto acoplado a la cámara *CCD* en esta posición. La importancia del sistema telefoto radica en que permite amplificar ronchigramas generados a partir de espejos con razones focales relativamente grandes, por ejemplo *f/8*.

Después de la alineación y colocación de la cámara es posible observar la imágen del ronchigrama en el monitor de televisión utilizando la tarjeta digitalizadora colocada en la computadora. Una vez alineado el sistema se digitaliza el ronchigrama. En este proceso los niveles de intensidad son transformados a números enteros que son almacenados para su procesamiento posterior. Aqui es importante recordar que la prueba de Ronchi es insensible a aberraciones transversales en las direcciones paralelas a las lineas de la rejilla; por esta razón se repite el proceso para dos posiciones perpendiculares de la rejilla.



(a)



(b)

fig. 3.2 Diagramas de dos configuraciones posibles en el probador de Ronchi. (a) para pruebas fuera de eje (b) para pruebas en eje.

3.2 TRATAMIENTO DE LOS RONCHIGRAMAS POR MEDIO DE LAS TECNICAS DE VISION POR COMPUTADORA.

Una vez que hemos capturado los ronchigramas horizontales y verticales, podemos pasar a la etapa de maximización, es decir, a encontrar las lineas de máxima intensidad de cada franja. Este proceso lo dividimos en tres etapas: *suavizamiento, esqueletización, y adelgazamiento.*

3.2.1 Suavizamiento del ronchigrama por promedios de vecindades.

El suavizamiento es una etapa importante para eliminar el ruido que se introduce cuando se captura un ronchigrama. El ruido se debe a multiples factores relacionados directa ó indirectamente con los instrumentos que utilizamos en la captura. En general desconocemos la naturaleza del ruido, ó que dispositivos tienen mayor contribución. Encontramos experimentalmente que el método de suavización o eliminación de este tipo de ruido resulta indispensable en la localización de las lineas de máximos en las franjas.

Dado el ronchigrama como una función discreta $f(n,m)$ con NxN elementos, el procedimiento consiste en generar una función suavizada $g(n,m)$ cuyos valores de intensidad en cada punto (n,m) se obtienen promediando los valores de intensidad en su vecindad, esto es

$$g(n,m) = \frac{1}{M} \sum_{(k,l) \in S} f(k,l) \qquad (3.2)$$

para $n,m = 0,1,\ldots,N-1$, s es el conjunto de coordenadas de los puntos en la vecindad del punto (n,m), incluyendo a (n,m), y M es el numero de total de puntos en la vecindad. Para una vecindad cuadrada con 9 pixeles, como la de la *figura 3.3*, la ecuación anterior se reduce a

$f(k-1, l-1)$	$f(k, l-1)$	$f(k+1, l-1)$
$f(k-1, l)$	$f(k, l)$	$f(k+1, l)$
$f(k-1, l+1)$	$f(k, l+1)$	$f(k+1, l+1)$

fig. 3.3 Vecindad utilizada en el proceso de suavizamiento

P_{-22}	P_{-12}	P_{02}	P_{12}	P_{22}
P_{-21}	P_{-11}	P_{01}	P_{11}	P_{21}
P_{-20}	P_{-10}	P_{00}	P_{10}	P_{20}
P_{-2-1}	P_{-1-1}	P_{0-1}	P_{1-1}	P_{2-1}
P_{-2-2}	P_{-1-1}	P_{0-2}	P_{1-2}	P_{2-2}

Fig. 3.4 Vecindad de 5x5 pixeles usada en el algoritmo de esqueletizacion.

$$g(n,m) = 1/9 \{ f(n-1,m-1) + f(n,m-1) + f(n+1,m-1) + f(n-1,m) + f(n,m) + f(n+1,m) + f(n-1,m+1) + f(n,m+1) + f(n+1,m+1) \} \quad (3.3)$$

con la cual obtenemos el ronchigrama suavizado. Este proceso se itera varias veces hasta obtener el suavizado adecuado, que por otras parte nos permite obtener continuidad en las lineas de máxima intensidad.

3.2.2 Esqueletizado

El procedimiento para extraer las lineas de máxima intensidad sugerido por *Yatagai et.al (1982)*, se aplica a los ronchigramas suavizados, y consiste básicamente en utilizar una matríz cuadrada de *5x5 pix (Fig. 3.4)*, en la cual determinamos si el pixel central, con un nivel de gris P_{00}, es o no un máximo de intensidad. Para ello se establecen cuatro pares de condiciones sobre los niveles de gris de la matríz, una para cada dirección mostrada en la *fig. 3.5*. El pixel central será reconocido como un máximo si dos o mas parejas de condiciones se satisfacen. Estas parejas de condiciones las podemos escribir como

Dirección X

$$P_{00} + P_{0-1} + P_{01} > P_{-21} + P_{-20} + P_{-2-1} \quad (3.4a)$$

$$P_{00} + P_{0-1} + P_{01} > P_{21} + P_{20} + P_{2-1} \quad (3.4b)$$

Dirección Y

$$P_{00} + P_{-10} + P_{10} > P_{-1-2} + P_{0-2} + P_{1-2} \quad (3.5a)$$

$$P_{00} + P_{-10} + P_{10} > P_{-12} + P_{02} + P_{12} \quad (3.5b)$$

Dirección XY

$$P_{00} + P_{-1-1} + P_{11} > P_{-22} + P_{-21} + P_{-12} \quad (3.6a)$$

$$P_{00} + P_{-1-1} + P_{11} > P_{2-2} + P_{2-1} + P_{1-2} \quad (3.6b)$$

Dirección -XY

$$P_{00} + P_{-11} + P_{1-1} > P_{22} + P_{21} + P_{12} \qquad (3.7a)$$

$$P_{00} + P_{-11} + P_{1-1} > P_{-2-2} + P_{-2-1} + P_{-1-2} \qquad (3.7b)$$

Este procedimiento se aplica a cada punto del ronchigrama suavizado.

Despues que hemos obtenido los maximos de intensidad del ronchigrama , procedemos a binarizarlo, esto es, si el valor de P_{00} es un máximo, se compara con un valor de umbral, y si lo excede se le asigna el valor uno en la imagen de salida; de otra forma se le asigna valor cero.

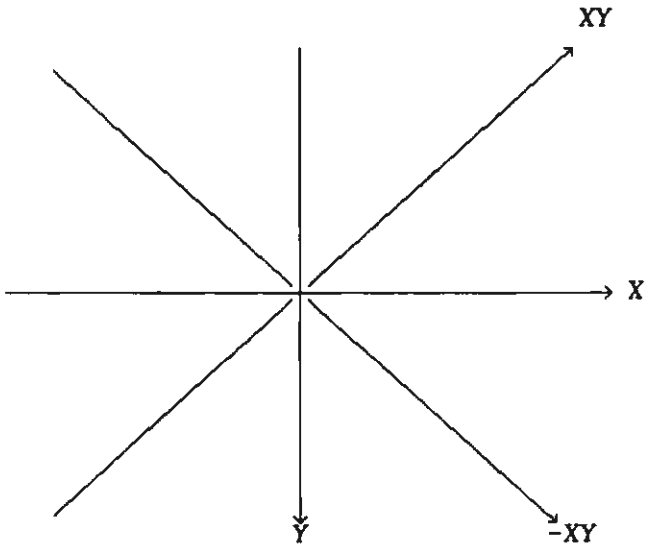


Fig. 3.5 Direcciones de detección en el algoritmo de esqueletizacion.

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

Fig. 3.6 Vecindad utilizada en el algoritmo de adelgazamiento.

3.2.3 Adelgazado

Después de binarizar el Ronchigrama, utilizamos una tecnica para adelgazar la imagen binaria hasta reducirla a su anchura mínima posible: un pixel *(Gonzales, 1987)*.

Este proceso de adelgazamiento consiste en utilizar una ventana de *3x3 (fig. 3.6)* y desplazarla sobre la imágen binaria dos veces (dos pasos).

En el primer paso se analiza si el pixel central de la matríz cumple las condiciones siguientes

$$\begin{aligned} &i).- \; 2 \leq N(p_1) \leq 6 \; , \\ &ii).- \; S(p_1) = 1 \; , \\ &iii).- \; p_2 \, p_4 \, p_8 = 0 \; , \\ &iv).- \; p_4 \, p_6 \, p_8 = 0 \; , \end{aligned} \qquad (3.8)$$

donde $S(p_1)$ es el numero de transiciones *0-1* que existe en la matríz siguiendo la numeración de la matríz en orden creciente, y $N(p)$ es el numero de valores diferentes de cero en la vecindad de p_1 es decir,

$$N(p_1) = p_2 + p_3 + \ldots + p_8 \quad . \qquad (3.9)$$

Si el pixel p_1 cumple las cuatro condiciones anteriores, la posición del pixel p_1 será almacenada y asignada en ella el nivel de gris cero cuando se haya barrido toda la imágen.

En el segundo paso se reemplazan las condiciones *(iii)* y *(iv)* por las siguientes

$$p_2 \, p_4 \, p_6 = 0 \; , \qquad (3.10a)$$

$$p_2 \, p_6 \, p_8 = 0 \; , \qquad (3.10b)$$

y se procede de manera similar al primer paso. Estos dos pasos son repetidos iterativamente hasta que las franjas sean reducidas a un pixel de ancho.

3.2.3 **Muestreo de datos.**

Una vez adelgazados los esqueletos a un pixel de ancho, procedemos a muestrearlos, modificando el algoritmo de adelgazamiento descrito en la sección anterior. Esta modificación consiste básicamente en encontrar un punto extremo g_1 para cada franja y a partir de allí iniciar su muestreo. Estos puntos iniciales son reconocidos como puntos extremos cuando se aplica el primer paso básico al esqueleto del ronchigrama y cada uno de los puntos g_1 satisface las cuatro condiciones *i), ii), iii)* y *iv)* de las *ecs. 3.8.*

3.3 RESULTADOS EXPERIMENTALES.

Ahora que hemos muestreado los ronchigramas de sus lineas adelgazadas aplicamos el procedimiento de reducción de datos que describimos en el capitulo anterior *(secc. 2.2)* y obtenemos la forma funcional de frente de onda $W'(x,y)$ *(Ec. 2.3).* Sin embargo, cuando hacemos éste cálculo suponemos que medimos correctamente la separación entre la rejilla y el vértice del espejo, y que la franja cero fué en realidad el centro de simetría del patrón, lo que no ocurre en la realidad. Por ello es conveniente sustraer a $W'(x,y)$ una función de error $W_o(x,y)$ *(Ghozeil, 1978)* con la forma

$$W_o(x,y) = A + Bx + Cy + D(x^2 + y^2) \qquad (3.11)$$

en donde $A, B, C, y\ D$ son los coeficientes: de término constante, inclinación en "x", inclinación en "y" y desenfocamiento respectivamente. Una vez calculados los coeficientes A, B, C, y D obtenemos la función de aberración buscada $W(x,y)$

$$W(x,y) = W'(x,y) - W_0(x,y) \qquad (3.12)$$

Ahora podemos reescribir la ec. *2.6* como

$$W(x,y) \overset{k}{\underset{i=0}{\sum}} \overset{i}{\underset{j=0}{\sum}} B_{ij}\, x^{i}\, y^{i-j} \quad - \quad A+Bx+Cy+D(x^2+y^2) \qquad (3.13)$$

y la función de deformación $D(x,y)$ de la superficie bajo prueba se obtiene con la aproximación

$$D(x,y) \cong W(x,y) / 2. \qquad (3.14)$$

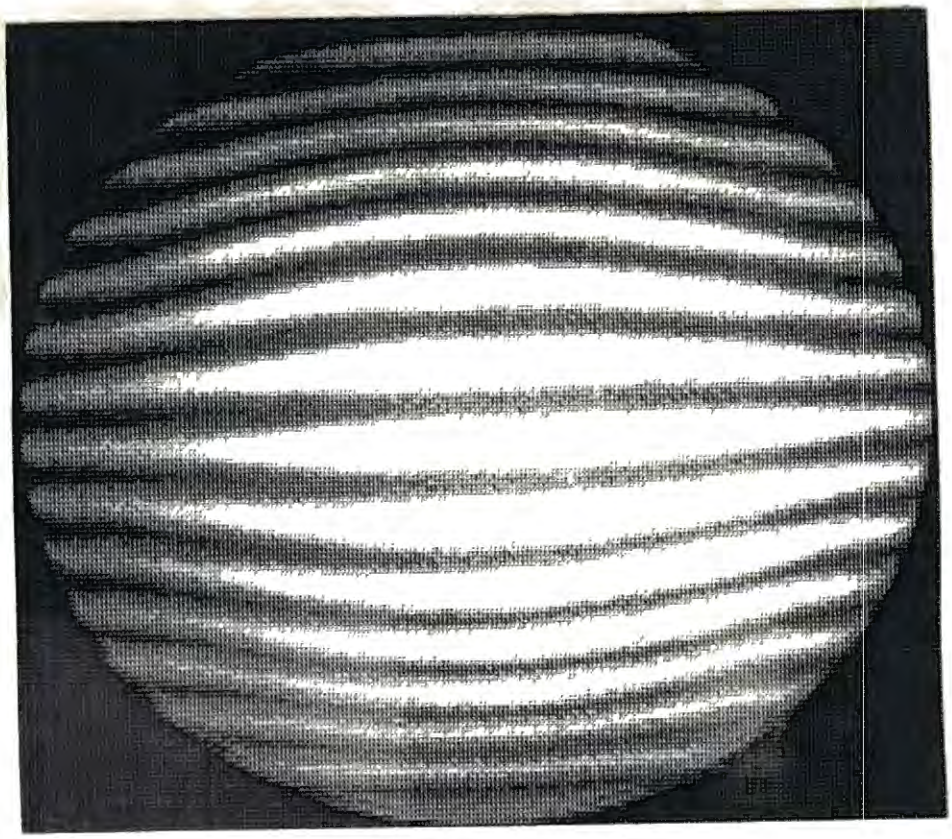
Para comprobar este método se utilizó una superficie cóncava con las siguientes características: constante de conicidad $k=-1$, radio de curvatura de *75cm*, diámetro del espejo de *15cm* y frecuencia de la rejilla de *40 lin/cm (Pérez, 1988)*.

Elegimos como superficie de referencia el paraboloide ideal que deseamos obtener es decir, la función $z_0(x,y)$ de referencia que mencionamos en la *secc. 2.1* y que en nuestro caso ($k=-1$) es *(ec.2.2)*

$$f(x,y) = c\, s^2/2 \qquad (3.15)$$

A continuación mostraremos en forma gráfica el procedimiento de adelgazamiento, eskeletización, adelgazamiento, muestreo, y la gráfica de los errores de la superficie bajo prueba, respecto a la superficie $z_0(x,y)$, definida por la *ec. 3.15*.

La *fig. 3.7* muestra los dos ronchigramas digitalizados de los cuales partimos. En la *fig. 3.8* tenemos los esqueletos de estos ronchigramas despues de aplicarles dos veces el procedimiento de filtraje descrito en la *sección 3.2.1.* En ésta figura podemos notar que en algunas partes las lineas poseen mas de un pixel de ancho, por lo cual se hace necesario el proceso de adelgazamiento que describimos en la sección *3.2.3* y cuyos resultados mostramos en la *fig. 3.9.*



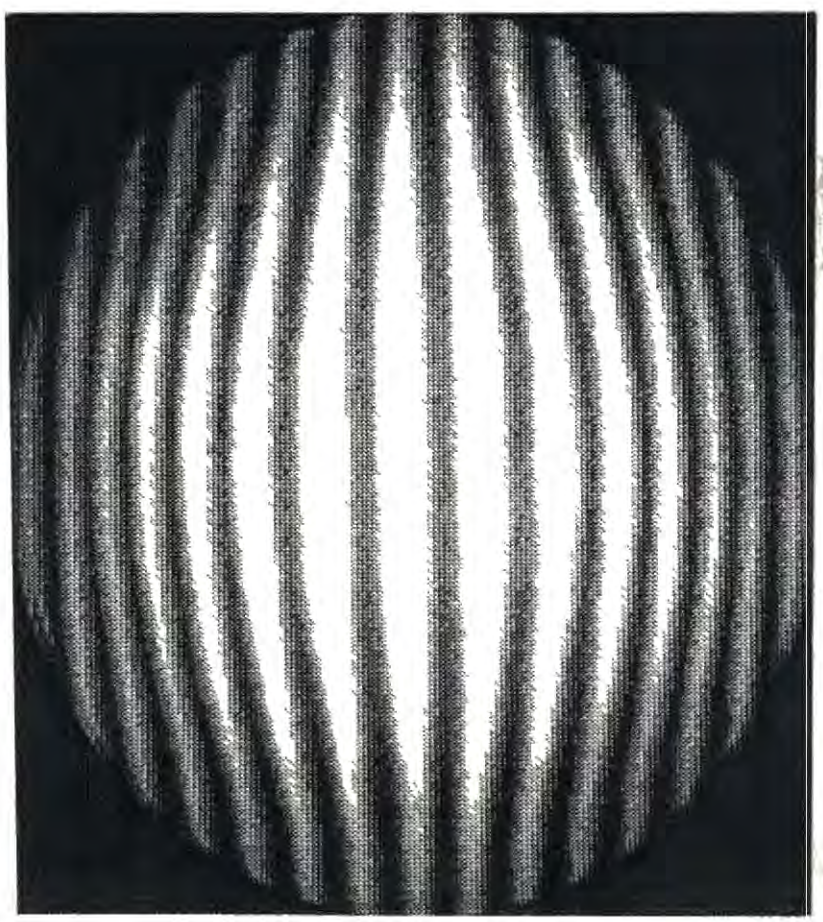


Fig. 3.7 Ronchigramas reales capturados con el programa SEMPER

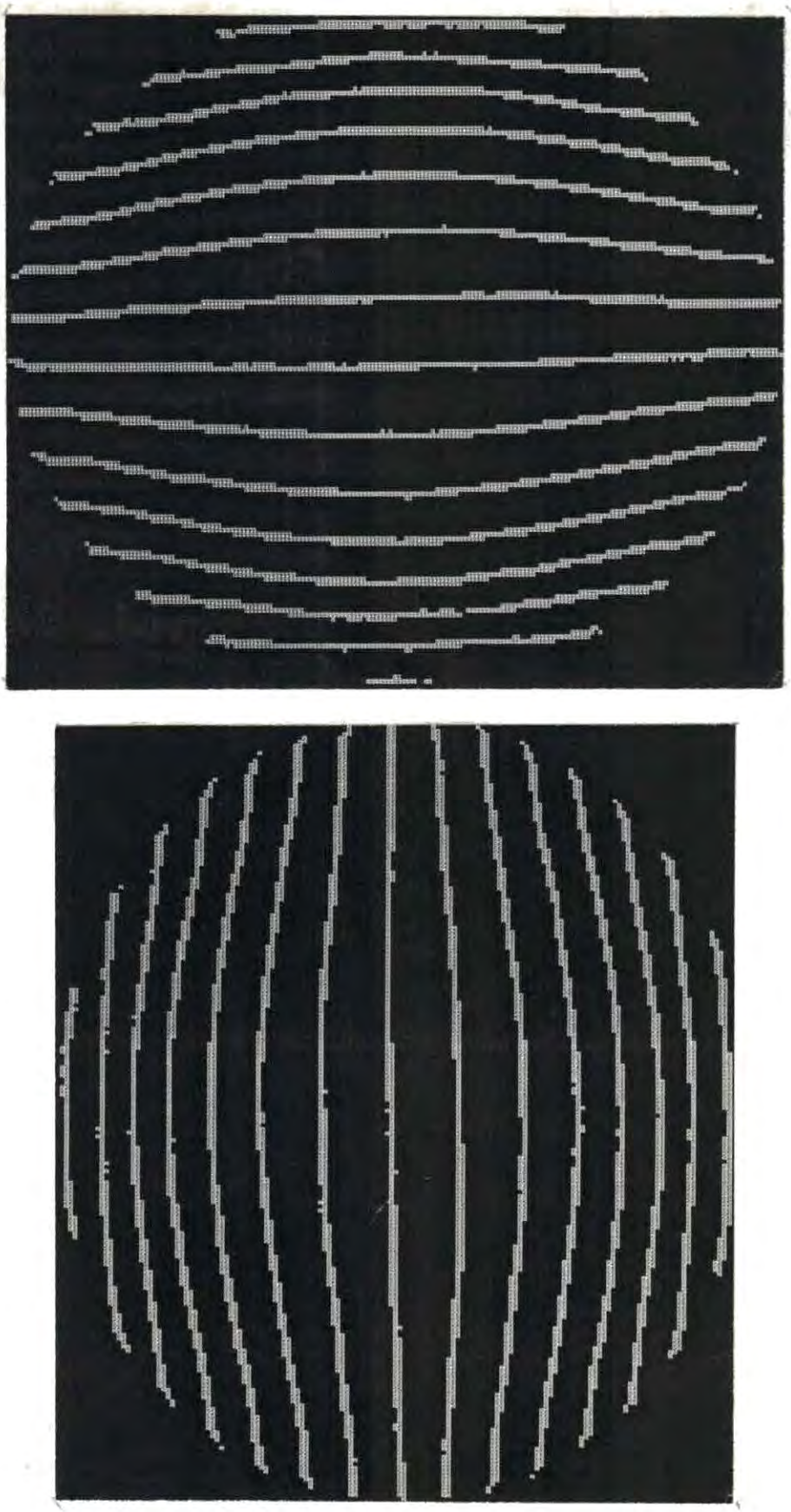
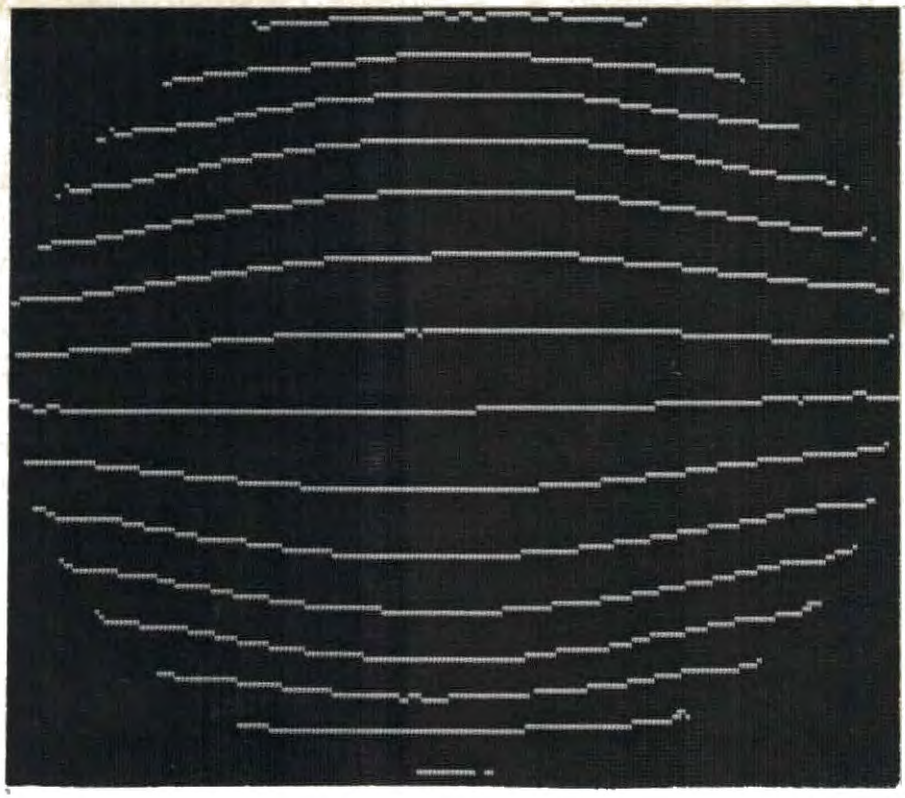


Fig. 3.8 ronchigrams esqueletizados como resultado de aplicar el procedimiento de la secc. 3.2.2 (programas **FILBAJ Y ESKEL**.



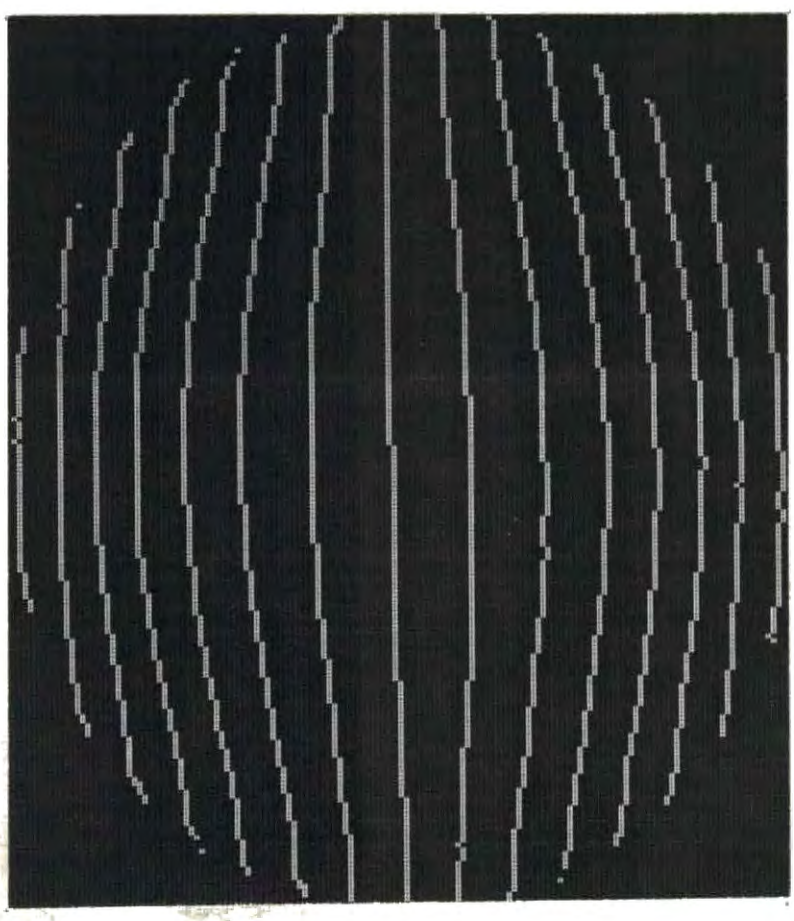


Fig. 3.9 Ronchigrams adelgazados siguiendo el procedimiento de la secc. 3.2.3. (programa THIN)

Finalmente, despues de muestrear los ronchigramas anteriores y de utilizar estos datos para hacer el ajuste de minimos cuadrados que describimos en la *secc. 2.2*, presentamos en la *fig. 3.10* los resultados obtenidos en una gráfica tridimensional. Para apreciar mejor estos resultados, en las *figuras 3.18 y 3.19* presentamos los perfiles de la superficie en los planos z-x y z-y, respectivamente.

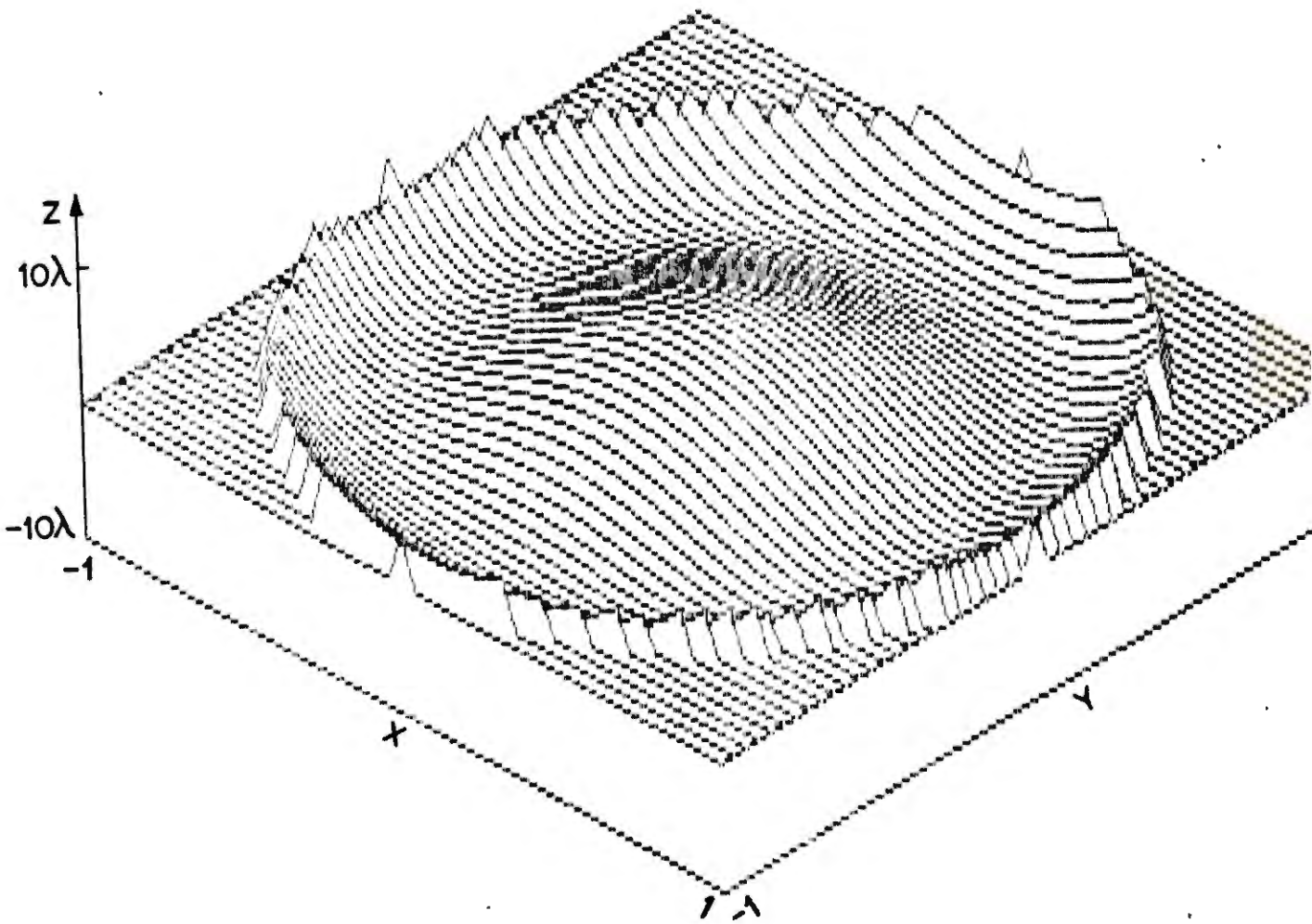


Fig. 3.10 Figura tridimensional de la superficie bajo prueba despues de aplicar el método de reducción de datos (secc.2.2) (programa GRA_PRU).

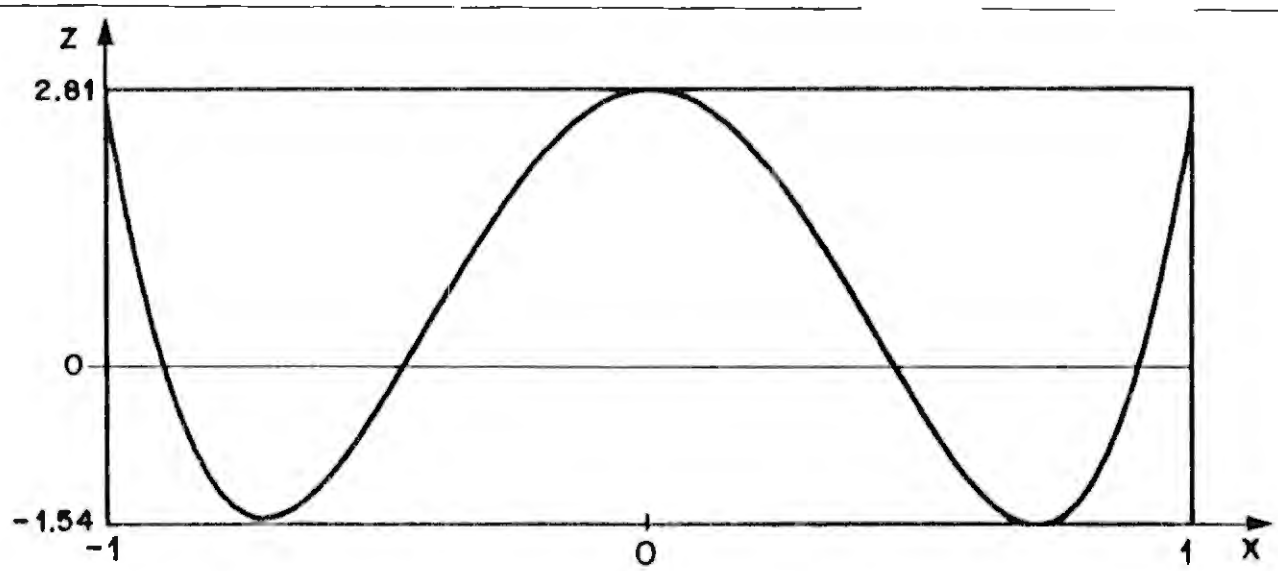


Fig. 3.11 Perfil de la superficie en el plano Z-X

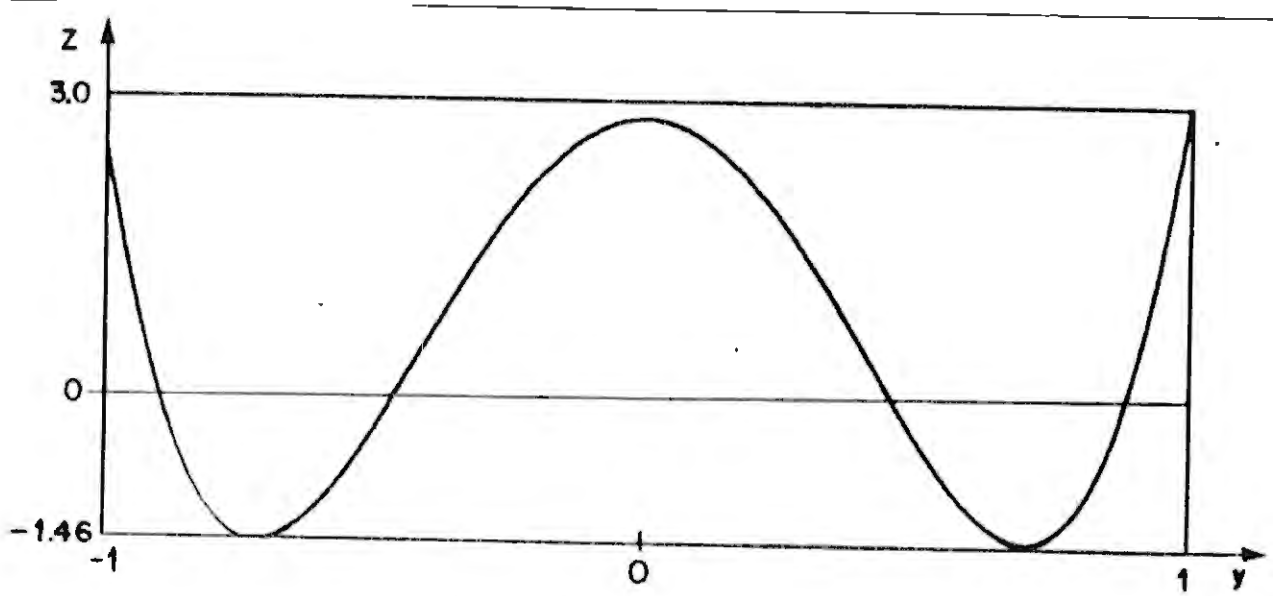


Fig. 3.12 Perfil de la superficie en el plano Z-y

Los errores de la superficie real respecto a la ideal fueron los siguientes

$$\begin{aligned} &\textit{Error RMS} = && 1.2638\ \lambda \\ &\textit{Error PICO} = && 3.1479\ \lambda \\ &\textit{Error VALLE} = && -1.579\ \lambda \\ &\textit{Error PICO-PICO} = && 4.7269\ \lambda \end{aligned} \qquad (3.17)$$

Estos errores fueron obtendidos a partir de los ronchigramas de la *fig. 3.7*. Para asegurarnos que efectivamente estas fueron las deformaciones, desmontamos de su soporte a la superficie bajo prueba, variamos el numero de franjas y calculamos nuevamente las deformaciones. Repetimos este proceso *6* veces y encontramos que los resutados no diferian en mas de *1/10* λ en el error *PICO_PICO* y *1/100* λ en el *RMS* (λ=*6328A*).

Para ajustar en el sentido de minimos cuadrados la función del frente de onda deformado $w(x,y)$, usamos una base monomial de 5 orden en dos dimensiones, y un soporte de aproximadamente *300* puntos por ronchigrama. Estos puntos los obtuvimos muestreando ronchigramas de aproximadamente *15* franjas.

El tiempo de cómputo que utilizamos para procesar los ronchigramas y obtener a partir de ellos la topografía de la superficie que los originó nos tomó aproximadamente *8 minutos*.

CAPITULO V

CONCLUSIONES

A lo largo de este trabajo se han presentado los fundamentos de la prueba de Ronchi y su teoría geométrica. Se han resumido la técnicas mas comunes de la visión por computadora necesarias para el procesamiento de los ronchigramas y tambien se han aplicado los resultados de esta teoría al proceso de automatización. Finalmente, se ha evaluado el proceso con diferentes espejos y se ha demostrado una buena repetibilidad de los resultados experimentales. Entre las ventajas del proceso automatico podemos citar las siguientes:

1.- Nos permite medir la superficie de espejos con gran exactitud.
2.- Con su uso se evita la incertidumbre asociada a la interpretación visual de los ronchigramas.
3.- El tiempo que se requiere para obtener los resultados es relativamente bajo si se utiliza una computadora *AT-16Mhz.*
4.- El procedimiento es aplicable a superfices asfericas.

estas ventajas resultan ser de gran importancia en el taller de fabricación óptica.

Desde nuestro punto de vista presente este trabajo se puede extender hasta llegar a considerarlo como una etapa preliminar a la prueba de Hartmann, que se utiliza para evaluar espejos grandes (mayores a *60 cm de diametro).* Sin embargo resultaría necesario hacer las pruebas pertinentes. Esta extensión queda como material para futuras investigaciones.

APENDICE 1

El programa que utilizamos para interpretar cuantitativamente los ronchigramas en una superficie óptica bajo prueba y obtener sus deformaciones es muy extenso. Para facilitar su programación agrupamos los programas en tres partes: *Esqueletización, ajuste de minimos cuadrados, y graficación.* A continuación explicaremos cada una de ellas y listaremos los programas de que consta.

ETAPA DE ESQUELETIZACIÓN

Consiste en capturar los ronchigramas, grabarlos en la memoria de la computadora y posteriormente encontrar sus lineas de máxima intensidad. Para ello utilizamos los programas *SEMPER, FILBAJ, ESKEL Y THIN.*

SEMPER

Este es un paquete numérico de procesamiento digital de imágenes comercializado por la compañia *Data Translation,* destinado a visualizar imágenes capturadas con una cámara sensora de tipo *CCD, VIDICON, ETC.* Lo usamos para grabar los ronchigramas horizontal y vertical, necesarios para conocer las aberraciones transversales de una superficie óptica bajo prueba. Estos ronchigramas los escribimos en los archivos *RONH.DAT* y *RONV.DAT*, respectivamente.

Ahora que tenemos almacenados los ronchigramas procedemos a esqueletizarlos, esto es, a encontrar sus lineas de máxima intensidad. En este proceso aplicamos secuencialmente a cada ronchigrama los tres programas restantes: *filbaj, eskel, y thin.*

FILBAJ

Este programa realiza la suavización *(secc. 3.2.1)* de los niveles de gris que poseen los ronchigramas almacenados en los archivos *RONH.DAT* Y *RONV.DAT* capturadas por el paquete de programas *SEMPER*. El resultado se escribe en el archivo *R1F.DAT*.

ESQUEL

Este programa utiliza la imagen filtrada de los Ronchigramas *R1F.DAT* y aplica el algoritmo de esqueletización descrito en la *secc. 3.2.2*. El resultado de este procedimiento es archivado nuevamente en *R1F.DAT*, para ahorrar memoria.

THIN

La imágen esqueletizada es finalmente adelgazada a un pixel de ancho usando el algoritmo de la *secc. 3.2.3*. El archivo resultante será escrito en *THINH.DAT* si el archivo inicial fué *RONH.DAT*, será respaldado en *THINV.DAT* si lo fué *RONV.DAT*.

ETAPA DE AJUSTE POR MINIMOS CUADRADOS

En esta etapa muestreamos los esqueletos de los ronchigramas obtenidos anteriormente, y a partir de los datos muestreados hacemos el ajuste de mínimos cuadrados para obtener la función de deformación buscada *(ec. 2.6)*. Empleamos los programas *malla, corrige, taxyp1_t, matrt2, inver_t, matfoc, e invfoc.*

MALLA

Este programa exhibe los esqueletos *THINH.DAT* y *THINV.DAT* y realiza el muestreo en el monitor, con interacción del usuario, usando el algoritmo de muestreo que explicamos en la sección *3.2.4*. EL muestreo se guarda en los archivos *ENTRAH.DAT* y *ENTRAV.DAT* y son archivados como (x, y, m_x) y (x, y, m_y), donde (x, y) es la coordeanda cartesiana en pixeles del punto muestreado, y m_x y m_y son los numeros de las franjas asigandos por el usuario de acuerdo a la convención de signos que definimos en la *secc. 1.2.*

CORRIGE

Aqui utilizamos los archivos *ENTRAH.DAT* y *ENTRAV.DAT* anteriormente generados para calcular el centroide de cada esqueleto y situarlo numéricamente en el origen de coordenadas del monitor. Los puntos corregidos los asignamos a los archivos *ENTRAH1.DAT* y *ENTRAV1.DAT*, respectivamente.

Los programas *TAXYP1_T, MATRT2 e INVMC,* se aplican secuencialmente a los datos del ronchigrama vertical y posteriormente al horizontal.

TAXYP1_T

Ahora que hemos corregido los puntos muestreados, procedemos a evaluar las aberraciones transversales reales *(ecs. 1.7)* y las aberraciones transversales ideales *(secc. 2.1)* en cada punto *(x,y)* muestreado. Posteriormente procedemos a restarlas y las guardamos en los archivos *TAX.DAT* y *TAY.DAT*. Empleamos *TAX* si deseamos usar los datos del ronchigrama vertical, y *TAY* si son los del horizontal.

MATRT2

Aqui usamos los archivos *ENTRAH1.DAT* ó *ENTRAV1.DAT*, y *TAX.DAT* ó *TAY.DAT* obtenidos de los programas *corrige* y *taxyp1_t* para formar las matrices del sistema de ecuaciones *2.8*. Estas matrices se guarda en los archivos *BSAL.DAT* y *FSAL.DAT*.

INVVER_T

Este programa invierte la matriz de minimos cuadrados *BSAL.DAT* y la multiplica por el vector *FSAL.DAT* para obtener los coeficientes de minimos cuadrados de la *ec. 2.2*. Posteriormente emplea las relaciones *ec. 2.4* para obtener la función de aberración *W(x,y) (ec. 2.6)*.

Estos coeficientes son guardados en los archivos *SOLTAX.DAT* y *SOLTAY.DAT*. El primero si en *TAXYP1_T* se utilizó *TAX.DAT*, y el otro si se usa *TAY.DAT*.

MATFOC

Emplea la expresión continua del frente de onda $w(x,y)$ obtenida en el programa anterior para discretizarla . Los puntos obtenidos serán el soporte de un nuevo ajuste de mínimos cuadrados, pero ahora para las *ecs. 3.12*. La matríz de *5x5* formada se guarda en los archivos *SOLTAX.DAT* y *SOLTAY.DAT*.

INVFOC

Invierte la matríz formada en *MATFOC* y la multiplica con el vector de mínimos cuadrados tambien formado en el programa anterior. De esta manera encontramos los coeficientes de la *ec. 3.12*.

ETAPA DE GRAFICACIÓN

Finalmente pasamos a la etapa de graficación que consta de los programas *GRAF* y un paquete de gráficos desarrollado en el *Centro de Investigaciones en Optica*.

GRAF

Aqui empleamos los dos archivos de coeficientes generados en *INVMAT e INVFOC* para eliminar los errores introducidos en la captura de los ronchigramas. Evaluamos tambien los errores *PICO, VALLE* y *RMS (ecs. 2.10)* definidos en la *secc. 3.12*, y generamos el archivo de datos *DATA* que utiliza la subrutina de graficación *NOE.C*.

LISTADO DE LOS PROGRAMAS UTILIZADOS

PROGRAMA F I L B A J

```
/*
 Programa que realiza el adelgazamiento de los maximos de
franjas encontrados por ESKELT1.C
*/
#include  <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <math.h>
#define  ABS(NUM) (NUM<0 ? -NUM : NUM)
#define  NMAX 256
#define  NMONITOR  240
char far *scrn = (char far *) 0xD0000000;

  /* Subrutina para asignar memoria dinamica */
unsigned char **tablab( m, n )
int m, n ;
{
  int i ;
  unsigned char **p ;
  p = (  unsigned char **) malloc( m*sizeof( unsigned char *) ) ;
  for( i = 0; i < m; i++ )
    {
    p[i] = ( unsigned char * ) malloc ( n*sizeof(unsigned char) ) ;
    }
    return( p ) ; }

/* Programa principal */
main(argc, argv)
int argc;
char *argv[] ;
{
  unsigned char **im, ban[5][256];
  unsigned char nc[256] ;
  int i,j,k,l,m,ii,lj,IT,DI,cond ;
  int p1,p2,p3,p4,p5,p6,p7,p8,p9 ;
```

```
int np1,prod1,prod2 ;
char st[17] ;


FILE *fp1 ;  /* Direccion del Archivo de entrada  */
fp1 = fopen (argv[argc-1], "rb" ) ;
printf( " archivo  ==> %s",argv[argc-1]) ;

 /* Asignacion de memoria para la matriz imagen */
im=tablab( NMAX, NMAX ) ;

/* Lectura de la imagen a suavizar */
for ( i = 0; i<NMAX; i++ )
 {
   fread (nc, 1, NMAX, fp1) ;
   for ( j=0; j<NMAX; j++ )
   {
 im[i][j] = nc[j] ;
 scrn[i*NMAX + j] = im[i][j] ;
   }
 }

 /* Parametro de suavizamiento. La imagen se suaviza dos veces */
IT = 2 ;

 /*  Manipulacion de la matriz imagen im[i][j]  */
ii=0;
  for ( k=1; k<=IT; k++ )
  {
 /* Inicializacion de la matriz a[i][j] */
  for(i=0; i<=3; i++)
  for(j=0; j<NMAX; j++)
  ban[i][j] = 0 ;
  ii=0;
```

```
   for ( i=0; i<NMONITOR; i++)
    {
 if(ii==3) ii=0 ;
   for ( j=0; j<NMAX-2; j++)
       {
   p9 = (int)(im[i][j]) ;
   p2 = (int)(im[i][j+1]) ;
   p3 = (int)(im[i][j+2]) ;
   p8 = (int)(im[i+1][j]) ;
/*   p1 = (int)(im[i+1][j+1]) ;        Punto a asignar  */
   p4 = (int)(im[i+1][j+2]) ;
   p7 = (int)(im[i+2][j]) ;
   p6 = (int)(im[i+2][j+1]) ;
   p5 = (int)(im[i+2][j+2]) ;
   np1 = p2+p3+p4+p5+p6+p7+p8+p9 ;
   np1 = np1/8 ;

  /* Instrucciones de abanderamiento  */
  if( ii==0 ) ban[1][j+1] = (unsigned char)(np1) ;     /* 1 */
  if( ii==1 ) ban[2][j+1] = (unsigned char)(np1) ;     /* 2 */

  /*   Instruccion de suavizamiento */
 if( ii==2 )
 {
  im[i-2][j+1] = ban[0][j+1] ;
  scrn[ NMAX*(i-2) + (j+1) ] =im[i-2][j+1] ;
  ban[0][j+1] = 0 ;

  im[i-1][j+1] = ban[1][j+1] ;
  scrn[ NMAX*(i-1) + (j+1) ] = im[i-1][j+1] ;
  ban[1][j+1] = 0 ;
   if( j==NMAX-3 ){ for(lj=0; lj<NMAX-2; lj++){
   im[i][lj]=ban[2][lj] ;
  scrn[NMAX*i+lj]=im[i][lj] ;
   ban[2][lj]=0 ; }}
}
```

```
if( ii==2) ban[0][j+1] = (unsigned char)(np1) ;      /*  3  */
      }  /*  j  */
 ii=ii+1 ;
   }      /*  i  */
     printf( " termina filtrado %d",k) ;
  }       /*  k  */
  printf( " Termina filtrado final " ) ;
 }        /* main */
```

```
/*              PROGRAMA E S K E L  */

/* Programa que encuentra los maximos de patrones de franjas. Para ello
   utiliza una matriz de 5x5 y los niveles de gris dentro de ella - al-
   goritmo sugerido por Yatagi. */

#include  <stdio.h>
#include <alloc.h>
#include <math.h>
#define  ABS(NUM) (NUM<0 ? -NUM : NUM)
#define  NMAX 256
char far *scrn = (char far *) 0xD0000000;

  /* Asignacion de memoria dinamica */
unsigned char **tablab( m, n )
int m, n ;
{
  int i ;
  unsigned char **p ;
  p = (unsigned char **) malloc( m*sizeof( unsigned char *) ) ;
  for( i = 0; i < m; i++ )
    p[i] = ( unsigned char * ) malloc ( n*sizeof(unsigned char) ) ;
    return( p ) ;
}

  /* Programa principal */
main(argc, argv)
int argc;
char *argv[] ;
{
  unsigned char **im, nc[256] ;
  int i,j,k,l,x,y,xy,mxy,asig,DI,NGRIS;
  int pm22,pm12,p02,p12,p22,pm21,pm11,p01,p11,p21,pm20,pm10,
      p00,p10,p20,pm2m1,pm1m1,p0m1,p1m1,p2m1,pm2m2,pm1m2,p0m2,p1m2,p2m2 ;
  char st[17] ;
```

```
  FILE *fp1 ;  /* Direccion del Archivo de entrada  */
  fp1 = fopen (argv[argc-1], "rb" ) ;
  printf( " archivo  ==> %s",argv[argc-1]) ;

   /* Asignacion de memoria para la matriz  */
  im=tablab( NMAX, NMAX ) ;

  for ( i = 0; i<NMAX; i++ )
   {
     fread (nc, 1, NMAX, fp1) ;
     for ( j=0; j<NMAX; j++ )
     {
   im[i][j] = nc[j] ;
/*   scrn[i*NMAX + j] = im[i][j] ; */
     }
   }

     /*  Limpieza fuera del ronchigrama */
/*
 printf( " Nivel de gris permitido fuera del Ronchigrama") ;
 fgets(st,15,stdin) ;NGRIS = atoi(st) ;
  for (i=0; i<NMAX; i++)
  for (j=0; j<NMAX; j++)
  if( im[i][j] <= NGRIS )
   {
     im[i][j] = 0 ;
     scrn[ NMAX*i + j ] = 0 ;
   }
*/
   /* Borrado de la pantalla. Esto asegura que las lineas que
    aparezcan en la pantalla sean el resultadoe de este
     algoritmo */
  for (i=0; i<NMAX; i++)
  for (j=0; j<NMAX; j++)
  scrn[ NMAX*i + j ] = 0 ;
```

```
/*  Manipulacion de la matriz imagen im[i][j]  */
   for ( i=0; i<NMAX-4; i++)
    {
      for ( j= 0; j<NMAX-4; j++)
    {
   pm22 = im[i][j] ;
   pm12 = im[i][j+1] ;
    p02 = im[i][j+2] ;
    p12 = im[i][j+3] ;
    p22 = im[i][j+4] ;

   pm21 = im[i+1][j] ;
   pm11 = im[i+1][j+1] ;
    p01 = im[i+1][j+2] ;
    p11 = im[i+1][j+3] ;
    p21 = im[i+1][j+4] ;

   pm20 = im[i+2][j] ;
   pm10 = im[i+2][j+1] ;
    p00 = im[i+2][j+2] ;
    p10 = im[i+2][j+3] ;
    p20 = im[i+2][j+4] ;

  pm2m1 = im[i+3][j] ;
  pm1m1 = im[i+3][j+1] ;
   p0m1 = im[i+3][j+2] ;
   p1m1 = im[i+3][j+3] ;
   p2m1 = im[i+3][j+4] ;

  pm2m2 = im[i+4][j] ;
  pm1m2 = im[i+4][j+1] ;
   p0m2 = im[i+4][j+2] ;
   p1m2 = im[i+4][j+3] ;
   p2m2 = im[i+4][j+4] ;
   x=0 ;
   y=0 ;
```

```
  xy=0 ;
  mxy=0 ;
  /* Condiciones para ser maximo en la direccion X */
 if( (p00+p0m1+p01 > pm21+pm20+pm2m1) &&
     (p00+p0m1+p01 > p21+p20+p2m1) )    x=1 ;

  /* Condiciones para ser maximo en la direccion Y */
if( (p00+pm10+p10 > pm1m2+p0m2+p1m2) &&
 (p00+pm10+p10 > pm12+p02+p12) )     y=1 ;

  /* Condiciones para ser maximo en la direccion XY */
if( (p00+pm1m1+p11 > pm22+pm21+pm12) &&
 (p00+pm1m1+p11 > p2m2+p2m1+p1m2) )  xy=1 ;

  /* Condiciones para ser maximo en la direccion -XY */
if( (p00+pm11+p1m1 > p22+p21+p12) &&
    (p00+pm11+p1m1 > pm2m2+pm2m1+pm1m2) ) mxy=1 ;

  /* Evaluacion de la condicion de punto de
     maxima intensidad. Si es verdadera se
     identificara con el nivel de gris= 50 */
if( x+y+xy+mxy >=1 )  scrn[NMAX*(i+2) + (j+2)] = 50 ;
      } /*  j  */
   }     /*  i  */

 } /* main */
```

```
                /*      PROGRAMA  T H I N       */
/* Programa que realiza el adelgazamiento de los maximos de franjas
   encontrados por el programa ESKEL.C */
#include  <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <math.h>
#define  ABS(NUM) (NUM<0 ? -NUM : NUM)
#define  NMAX 256
char far *scrn = (char far *) 0xD0000000;

  /* Subrutina para asignar memoria dinamica */
unsigned char **tablab( m, n )
int m, n ;
{
  int i ;
  unsigned char **p ;
  p = (  unsigned char **) malloc( m*sizeof( unsigned char *) ) ;
  for( i = 0; i < m; i++ )
    {
    p[i] = ( unsigned char * ) malloc ( n*sizeof(unsigned char) ) ;
    }
    return( p ) ;
}

main(argc, argv)
int argc;
char *argv[] ;
{
  unsigned char **im, ban[3][256];
  unsigned char nc[256] ;
  int i,j,k,l,m,ii,kk,lj,IT,DI,cond ;
  int p1,p2,p3,p4,p5,p6,p7,p8,p9 ;
  int sp1,np1,prod1,prod2 ;
  char st[17] ;
```

```
  FILE *fp1 ;  /* Direccion del Archivo de entrada  */
  fp1 = fopen (argv[argc-1], "rb" ) ;
  printf( " archivo  ==> %s",argv[argc-1]) ;

    /* Asignacion de memoria para la matriz  */
  im=tablab( NMAX, NMAX ) ;

  for ( i = 0; i<NMAX; i++ )
   {
     fread (nc, 1, NMAX, fp1) ;
     for ( j=0; j<NMAX; j++ )
     {
   im[i][j] = nc[j] ;
/*   scrn[i*NMAX + j] = im[i][j] ; */
  /* Binarizacion de la imagen */
   if( im[i][j] > 0 ) im[i][j] = 1 ;
     }
   }

     /*  Borrado de la pantalla */
/*
 printf( " Numero de iteraciones ") ;
 fgets(st,15,stdin) ;IT = atoi(st) ;
*/
  /* Asignacion del parametro de adelgazamiento IT */
 IT = 1 ;

  /* Las iteraciones deben ser pares */
 IT = 2*IT ;

  /* Inicializacion de la matriz a[i][j] */
 for(i=1; i<=3; i++)
 for(j=0; j<NMAX; j++)
 ban[i][j] = 0 ;
```

```
 /*  Manipulacion de la matriz imagen im[i][j]  */
kk=0; ii=0;
  for ( k=1; k<=IT; k++ )
  {
kk=kk+1 ;
  for ( i=0; i<NMAX-2; i++)
   {
if(ii==3) ii=0 ;
     for ( j= 0; j<NMAX-2; j++)
      {
 p9 = im[i][j] ;  p2 = im[i][j+1] ;
 p3 = im[i][j+2] ; p8 = im[i+1][j] ;
/*  p1 = im[i+1][j+1] ;       punto a borrar  */
 p4 = im[i+1][j+2] ; p7 = im[i+2][j] ;
 p6 = im[i+2][j+1] ; p5 = im[i+2][j+2] ;

 sp1 = 0 ;
 cond= 0 ;
 np1 = p2+p3+p4+p5+p6+p7+p8+p9 ;

 if( (p2==0) && (p3==1) ) sp1 = 1 + sp1 ;
 if( (p3==0) && (p4==1) ) sp1 = 1 + sp1 ;
 if( (p4==0) && (p5==1) ) sp1 = 1 + sp1 ;
 if( (p5==0) && (p6==1) ) sp1 = 1 + sp1 ;
 if( (p6==0) && (p7==1) ) sp1 = 1 + sp1 ;
 if( (p7==0) && (p8==1) ) sp1 = 1 + sp1 ;
 if( (p8==0) && (p9==1) ) sp1 = 1 + sp1 ;

 /* Condicion para el paso 1  */
 if( kk==1 )
 {
 prod1 = p2*p4*p6 ;
 prod2 = p4*p6*p8 ;
 }
```

```
 /* condicion para el paso 2  */
 if( kk==2 )
 {
  prod1 = p2*p4*p8 ;
  prod2 = p2*p6*p8 ;
 }
 /* Condiciones de borrado */
 if( (np1>=2) && (np1<=6) ) cond = 1 ;
 if( sp1==1 && prod1==0 && prod2==0 ) cond = cond + 1 ;

 /* Instrucciones de abanderamiento  */
 if( cond==2 && ii==0 ) ban[1][j+1] = im[i+1][j+1] ;     /* 1 */
 if( cond==2 && ii==1 ) ban[2][j+1] = im[i+1][j+1] ;     /* 2 */


 /* Instruccion de adelgazamiento */
if( ii==2 )
{
  if( ban[0][j+1]==im[i-2][j+1] ) { im[i-2][j+1]=0; ban[0][j+1]=0;
        scrn[ NMAX*(i-2) + (j+1) ] =0;}

  if( ban[1][j+1]==im[i-1][j+1] ) { im[i-1][j+1]=0; ban[1][j+1]=0;
        scrn[ NMAX*(i-1) + (j+1) ] =0;}

  if( j==NMAX-3 ){ for(lj=0; lj<NMAX; lj++){
  if( ban[2][lj]==im[i][lj]) { im[i][lj]=0; ban[2][lj]=0;
    scrn[NMAX*i+lj]=0;}}}
}
if( cond==2 && ii==2) ban[0][j+1] = im[i+1][j+1] ;
      }  /*  j  */
ii=ii+1 ;
   }     /*  i  */
if(kk==2) kk=0 ;
printf( " Termina adelgazamiento %d",k ) ;
  }      /*  k  */
 }       /* programa  */
```

```
                    /* PROGRAMA  M U E S  */

/*
 Programa que hace un muestreo semi_automatico de las lineas
  de maxima intensidad de los ronchigramas
 */

#include  <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <dos.h>
#include <math.h>
#define  NMAX 256
#define   IO  128

char far *scrn = (char far *) 0xD0000000;

unsigned char **tablab( m, n )
int m, n ;
{
  int i ;
  unsigned char **p ;
  p = (unsigned char **) malloc( m*sizeof( unsigned char *) ) ;
  for( i = 0; i < m; i++ )
    p[i] = ( unsigned char * ) malloc ( n*sizeof(unsigned char) ) ;
    return( p ) ;
}

  int GRIS(i,j)
 int i,j ;
 {
  int ii,jj,sp1=0,cond=0,np1=0,prod1,prod2 ;
  int GR[10] ;
```

```
/* Formacion de la ventana */
 GR[1] = scrn[i*NMAX + j] ;
 GR[2] = scrn[(i-1)*NMAX + j] ;
 GR[3] = scrn[(i-1)*NMAX + j+1] ;
 GR[4] = scrn[i*NMAX + j+1] ;
 GR[5] = scrn[(i+1)*NMAX + j+1] ;
 GR[6] = scrn[(i+1)*NMAX + j] ;
 GR[7] = scrn[(i+1)*NMAX + j-1] ;
 GR[8] = scrn[i*NMAX + j-1] ;
 GR[9] = scrn[(i-1)*NMAX + j-1] ;

/*Normalizacion de los niveles de gris */
 for(ii=1; ii<=9; ii++)
  if( GR[ii] !=0 ) GR[ii]=1;
    /* Evaluacion de las condiciones */
    /*1*/
 for(ii=2; ii<=9; ii++) np1 +=GR[ii] ;
    /*2*/
 for(ii=2; ii<=8; ii++)  if(GR[ii]==0 && GR[ii+1]>=1) sp1+=1 ;
 if(GR[9]==0 && GR[2]>=1 ) sp1+=1;
    /*3*/
  prod1=GR[2]*GR[4]*GR[6];
  prod2=GR[4]*GR[6]*GR[8];

    /* Asignacion de las condiciones */
  if( np1>=2 && np1<=6)
   {
    cond=1 ;
    if(sp1==1 && prod1==0 && prod2==0) cond+=1 ;
   }
   if(cond==0 && np1==1) return(1) ;
   else return(0) ;
  } /* subrutina */
```

```
main(argc, argv)
int argc;
char *argv[] ;

{
  unsigned char **im, nc[NMAX] ;
  int i,j,cond1,cond2,ii,jj,NPUN=0,rij ;
  int FREC=7,cond,iy,jx,yi,xj,paso=0,NFRA ;
  int XCE,YCE,RAD ;
  char st[15] ;

  FILE *fpe ;  /* Direccion del Archivo de entrada  */
  FILE *fps ;

 fpe = fopen (argv[argc-1], "rb" ) ;
 fps = fopen ("cursor.dat", "w" ) ;
 printf( " archivo  ==> %s",argv[argc-1]) ;

  /* Asignacion de memoria para la matriz  */
  im=tablab( NMAX, NMAX ) ;
  printf( " voy a leer el archivo") ;
  for ( i=0; i<NMAX; i++ )
   {
  fread (nc, 1, NMAX, fpe) ;
     for ( j=0; j<NMAX; j++ )
     {
   im[i][j] = nc[j] ;
   if( im[i][j]!=0 )  scrn[i*NMAX + j ] =40  ;
   else scrn[i*NMAX +j]=0 ;
   im[i][j] = 0 ;
     }
   }
  for ( i=1; i<NMAX-1; i++ )
   {
  for ( j=1; j<NMAX-1; j++ )
  {
```

```
  if( scrn[i*NMAX +j] < 40 ) goto cont ;
  cond1 = GRIS(i,j) ;    /* Evaluacion de las condiciones de franja */
if( cond1==1)
 {

  for(ii=-5; ii<-2; ii++)
   scrn[(i+ii)*NMAX +j]=90 ;

   printf( " Numero de franja?  ");
   fgets(st,15,stdin);
   NFRA = atoi(st) ;
   if( NFRA >= 50) goto cont1 ;
   paso=0 ;

  for(ii=-6; ii<-2; ii++)
   scrn[(i+ii)*NMAX +j]=0 ;

yi =i;
xj =j;
im[yi][xj]=1;

scrn[yi*NMAX +xj]=30 ;
cont1:
for(ii=-1; ii<=1; ii++)
{
 for(jj=-1; jj<=1; jj++)
 {
  jx = xj + jj;
  iy = yi + ii;

  if( (ii!=0 || jj!=0) && scrn[iy*NMAX+jx]  >30)
   {
  scrn[iy*NMAX +jx]=30;
  im[iy][jx]=1;
  paso+=1;
```

```
    if(paso==FREC)
     {
      fprintf(fps,"%d %d %d ",jx-IO,IO-iy,NFRA);
      NPUN++;
      im[iy][jx]=0;
      scrn[iy*NMAX +jx]=20;
      paso=0;
      }
     xj=jx;
     yi=iy;
      goto cont1 ;
     }
   }
    }     /* jj */
   }      /* ii */
   cont: ;
   }
  }
   jx =10000;   iy =10000; NFRA=1 ;
   fprintf(fps,"%d %d %d ",jx,iy,NFRA);
   printf( "Numero de puntos = %d",NPUN);

} /* main  */
```

```
/* PROGRAMA  C O R R I G E */

/*
   Programa que localiza el centroide de los datos muestreados
   de los ronchigramas. Utiliza las coordenadas del centroide
   como coordenadas origen.
  Este programa posee la estructura para introducir un para-
   metro de distorsion que corregira la distorsion introducida por
   la camara CCD.
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <dos.h>

#define  NILLOS  500
#define     NMAX    256
#define     IO      128
#define ABS(x) ( x<0 ?  -x : x )

char far *scrn = (char far *) 0xD0000000;

   sgn(x)
  double x ;
   {
  double y ;
  if(x==0) y=0 ;
  else y=x/ABS(x) ;
  return(y) ;
   }
/*
 entrav.dat :  Archivo que contiene los puntos muestreados
       del ronchigrama vertical.
 entrah.dat :  Archivo que contiene los puntos muestreados
     del ronchigrama horizontal.
```

```
 entrav1.dat:  Archivo de salida con las coordenadas corre-
        gidas de entrav.dat.
 entrah1.dat:  Archivo de salida con las coordenadas corre_
        gidas de entrah.dat.
   SX:  Factor de escala horizontal necesario para
        pasar del modo de camara CCD al modo de monitor.
   SY:  Factor de escala vertical.
    k:  Parametro para corregir la distorsion que introduce
        una camara CCD en las imagenes capturadas.
 xcenv,ycenv:  Coordenadas del centroide correspondiente al ronchi-
         grama vertical.
 xcenh,ycenh:  Coordenadas del centroide correspondiente al ronchi-
         grama horizontal.
     Xrp,Yrp:  Coordenas corregidas.
*/

 main( )
   {
  double xcenv=0,ycenv=0,xcenh=0,ycenh=0,mxx,kxx ;
  double B,k,Sx,Sy,Cx,Cy,Xrp,Yrp,aux1 ;
  double Rv,X,Y,R,Xv,Yv;
  int kx,ky,mx,kkx,kky,i,j,l,m,punv=0,punh=0 ;
  char st[15] ;

   FILE *fv ;
   FILE *fh ;
   FILE *fv1 ;
   FILE *fh1 ;
   fv = fopen("entrav.dat","r") ;
   fh = fopen("entrah.dat","r") ;
   fv1 = fopen("entrav1.dat","w") ;
   fh1 = fopen("entrah1.dat","w") ;

  Sx = 1 / 1.28 ;
  Sy =1 ;
```

```
 /* Parametros de distorsion */
k  =0.0 ;
Cx = 0.0 ;
Cy = 0.0 ;
 /* Calculo de los centroides */
for(i=1; i<NILLOS; i++)
 {
   fscanf(fv," %d %d %d ",&kx,&ky,&mx) ;
   if(kx>=1000 ) break ;
  punv++;
  xcenv +=(float)(kx);
  ycenv +=(float)(ky);
  }
  xcenv /=punv ;
  ycenv /=punv ;
  printf( " xcenv=%f ycenv=%f ",xcenv,ycenv ) ;
  fseek(fv,0,0);
for(i=1; i<NILLOS; i++)
 {
   fscanf(fh," %d %d %d ",&kkx,&kky,&mx) ;
   if(kkx>=1000 ) break ;

  /* En nuestras mediciones podemos rotar la
  rejilla o rotar el espejo. Si rotamos el
  espejo es necesario hacer el siguiente cambio */
   aux1 = kky ;
 kky = kkx ;
 kkx = -aux1 ;
  punh++;
  xcenh +=(float)(kkx);
  ycenh +=(float)(kky);
  }
  xcenh /=punh ;
  ycenh /=punh ;
  printf( " xcenh=%f ycenh=%f",xcenh,ycenh );
  fseek(fh,0,0);
```

```
      /* Calculo de la coordenada no distorsionada (Xrp,Yrp)
    a partir de la distorsionada (Xr,Yr) */
for(i=1; i<NILLOS; i++)
 {
   fscanf(fv," %d %d %d ",&kx,&ky,&mx) ;
   if(kx>=1000 ){fprintf(fv1," %f",(float)(kx)); break ;}

  Yv = ky - ycenv;
  Xv = kx - xcenv;

  mxx= mx ;
  Rv = Xv*Xv + Yv*Yv ;
  X=Xv/(1.+k*Rv) ;
  Y=Yv/(1.+k*Rv) ;

  Xrp = X/Sx  + Cx ;
  Yrp = Y/Sy  + Cy ;
    fprintf(fv1," %f %f %f",Xrp,Yrp,mxx);

   }

      /* Calculo de la coordenada no distorsionada (Xrp,Yrp)
    a partir de la distorsionada (Xr,Yr) */
for(i=1; i<NILLOS; i++)
 {
   fscanf(fh," %d %d %d ",&kx,&ky,&mx) ;
   if(kx>=1000 ){fprintf(fh1," %f",(float)(kx)); break ;}
 aux1 = ky ;
 ky = kx ;
 kx = -aux1 ;
  Yv = ky - ycenh ;
  Xv = kx - xcenh ;
  mxx=  mx ;
  Rv = Xv*Xv + Yv*Yv ;
  X=Xv/(1.+k*Rv) ;
  Y=Yv/(1.+k*Rv) ;
```

```
  Xrp = X + Cx ;
  Yrp = Y/Sx + Cy ;

scrn[(IO-(int)(Yrp))*NMAX+(int)(Xrp)+IO]= 50 ;
    fprintf(fh1," %f %f %f",Xrp,Yrp,mxx);

   }

 }  /* END */
```

```
/*   PROGRAMA   T A X Y P 1 _ T   */

/* Programa que utiliza los archivos de datos corregidos  en
   el programa CORDATT para calcular la diferencia existente
   entre las aberraciones transversales reales y las aberra-
   ciones transversales ideales.
*/

#include <stdio.h>
#include  <math.h>
#define I0  128
#define NMAX  256
#define ABS(x) ( x<0 ?  -x : x )
#define SQR(x) ( x*x)
#define      NILLOS  500
#define      ITMAX   100
 /* Coeficientes de asfericidad */
#define   A1  0.0
#define   A2  0.0
#define   A3  0.0
#define   A4  0.0
#define   NFRA  10
#define   LAM   0.00006328
 char far *scrn = ( char far *) 0xD0000000 ;

   double func(x,y,k,c,a)
   double x,y,k,c,a ;
   {
  double zi,sa,s ;
    s = sqrt( x*x + y*y ) ;
    sa=s/a ;
    zi = c*s*s / ( 1. + sqrt(1.-(k+1.)*(s*s*c*c))) ;
    zi+= (A1*pow(sa,4)+A2*pow(sa,6)+A3*pow(sa,8)+A4*pow(sa,10))*LAM ;
     return (zi) ;
   }
```

```
 double parx(x,y,k,c,a)
 double x,y,k,c,a ;
 {
double arg1,sa,s ;
s = sqrt( x*x + y*y ) ;
sa = s/a ;
arg1 = x*c/sqrt(1.-(k+1.)*(c*c*s*s)) ;
arg1 += (A1*4*pow(sa,3)+A2*6*pow(sa,5)+
  A3*8*pow(sa,7)+ A4*10*pow(sa,9))*LAM/a;
return (arg1) ;
 }
 double pary(x,y,k,c,a)
 double x,y,k,c,a ;
 {
double arg1,sa,s ;
s = sqrt( x*x + y*y ) ;
 sa = s/a ;
arg1 = y*c/sqrt(1.-(k+1.)*(c*c*s*s)) ;
arg1 += (A1*4*pow(sa,3)+A2*6*pow(sa,5)+A3*8*pow(sa,7)+
   A4*10*pow(sa,9))*LAM/a;
return (arg1) ;
 }
/*
    entrav1,entrah1:   Archivos que contienen los datos corregidos
  por el programa CORDATT. Estos datos estan
  trasladados a sus respectivos centroides.
     tax,tay:   Archivos de salida que tienen las diferencias
  entre las aberraciones transversales reales
  y las ideales.
 FREC:   Frecuencia de la rejilla utilizada (en pulgadas)
    k:   Constante de conicidad.
    r:   Radio de curvatura del espejo bajo prueba.
    c:   Curvatura del espejo bajo prueba.
    a:   Radio fisico del espejo.
    CL,CM,CN:   Cosenos directores de los rayos incidentes al
  espejo.
```

```
   CLP,CMP,CNP:   Cosenos directores de los rayos reflejados por
    el espejo.
 */
   main ( )
{
   double xi,yi,si,se,xe,ye,sy,xp,yp,eps,zant,DRG ;
   double tar,tai,taxr,tayr,taxi,tayi,tx,ty ;
   double arg2,arg3,ALFA,BETA,GAMA ;
   double D,CL,CM,CN,CLP,CMP,CNP ;
   double par_x,par_y,par_z ;
   double a,d,r,c,L,k,FREC ;
   double s[10],t[10] ;
   double zi,za ;
   int i,ii,j,NPUN,m ;
   int my,ix,jy,i1 ;
   float kx,ky,mx,smaxv=-1000,smaxh=-1000;
   int np[2] ;
   char st[15] ;

   FILE *fv ;
   FILE *fh ;
   FILE *fx ;
   FILE *fy ;

   fv = fopen("entrav1.dat","r") ;
   fh = fopen("entrah1.dat","r") ;
   fx = fopen("tax.dat","wb") ;
   fy = fopen("tay.dat","wb") ;

printf ( " Separacion  rejilla - superficie " ) ;
 fgets( st,15,stdin);  L = atof(st) ;
 printf( " L ====> %f ",L ) ;

 printf ( " Radio de curvatura paraxial" ) ;
 fgets( st,15,stdin);  r = atof(st) ;
 printf( " r ====> %f ",r ) ;
```

```
printf ( " Diametro del espejo (cm)" ) ;
fgets( st,15,stdin);  a = atof(st) ;
printf( " a ====> %f ",a );
a= a/2.;

printf ( " Constante de conicidad " ) ;
fgets( st,15,stdin);  k = atof(st) ;
printf( " k ====> %f ",k ) ;

  FREC = 102.0 ;     /* Lineas por cm */
  NPUN = 0.0 ;
  d = 1.0/FREC ;
  c = 1/r ;
  eps = 0.0001 ;
  za = func(a,0.0,k,c,a) ; /* Sagita Maxima */

for (ii=1; ii<=2; ii++)
 {
  for ( i=1; i<=NILLOS; i++ )
       {
  if(ii==1) {
    fscanf(fv," %f %f %f",&kx,&ky,&mx) ;
    if(kx>=1000.0 ) break ;
    ky = kx*kx + ky*ky;
    if(ky>smaxv) smaxv=ky ;
     }
  if(ii==2) {
    fscanf(fh," %f %f %f ",&kx,&ky,&mx);
    if(kx>=1000.0 ) break ;
    ky = kx*kx + ky*ky;
    if(ky>smaxh) smaxh=ky ;
     }
 }
   }
  DRG=sqrt((smaxv+smaxh)/2.);
  printf( " DRG ====> %f ",DRG ) ;
```

```
 /* Inicio de archivo  */
  fseek(fv,0,0);
  fseek(fh,0,0);

 /* Ciclos para calcular las aberraciones transversales */
for (ii=1; ii<=2; ii++)
 {
  for ( i=1; i<=NILLOS; i++ )
      {
 /*  Lectura de datos  */
 if( ii==1 ) fscanf(fv," %f %f %f",&kx,&ky,&mx) ;
 if( ii==2 ) fscanf(fh," %f %f %f ",&kx,&ky,&mx);
  if(kx>=1000.0 ) break ;
  if( kx<1000.0 )
  {
 /*  Normalizacion de coordenadas  */
   xp = (double)(kx)*a/DRG ;
   yp = (double)(ky)*a/DRG ;

/*  Calculo de los cosenos directores (L,M,N) del rayo incidente */
  D = sqrt( xp*xp + yp*yp + (L-za)*(L-za) ) ;
 CL = xp/D ;
 CM = yp/D ;
 CN = (L-za)/D ;

/* Algoritmo para proyectar los puntos muestreados, a la superficie */
 zi = func(xp,yp,k,c,a) ;
 se = 10000.0 ;
 for( i1=1; i1<=ITMAX; i1++)
  {
    xe = xp + ( za + zi )*CL/CN ;
    ye = yp + ( za + zi )*CM/CN ;
    zant = zi ;
    zi = func(xe,ye,k,c,a) ;
    if( ABS(zant-zi)<=eps )
     {
```

```
      se = sqrt( xe*xe + ye*ye ) ;
   }
    if( se < 10000.0 ) break ;
   }
    /* Condicion de no convergencia */
if(se==10000.0){  printf("Punto no convergio"); goto nex ; }

 par_x = parx(xe,ye,k,c,a) ;
 par_y = pary(xe,ye,k,c,a) ;
 par_z = 1.0 ;
 arg2 = sqrt( par_x*par_x + par_y*par_y  + par_z*par_z) ;
 ALFA =   par_x/arg2 ;
 BETA =   par_y/arg2 ;
 GAMA =   par_z/arg2 ;
 arg3 = ALFA*CL + BETA*CM + GAMA*CN ;

 /* Cosenos directores de los rayos reflejados */
 CLP = CL - 2.*ALFA*arg3 ;
 CMP = CM - 2.*BETA*arg3 ;
 CNP = CN - 2.*GAMA*arg3 ;

/*  Identificacion de las aberraciones transversales  */
 if( ii==1 )
  {
/* Aberraciones transversales de  un ronchigrama vertical */
    taxi = xe - CLP*(L-zi)/CNP ;
    taxr = mx*d ;
    tx =  (1000.0)*(taxi-taxr)/L ;
 /* Normalizacion de coordenadas respecto al radio del espejo */
    xe = xe/a ; ye = ye/a ;
    s[0] = xe ; s[1] = ye ;
    s[2] = tx ;
    fwrite(s,8,3,fx) ;
   }
```

```
  if( ii==2 )
   {
 /* Aberraciones transversales de  un ronchigrama horizontal*/
     tayr = mx*d ;
     tayi = ye - CMP*(L-zi)/CNP ;
     ty =  (1000.0)*(tayi-tayr)/L ;

       /* Normalizacion de coordenadas respecto al radio del espejo */
     xe = xe/a; ye = ye/a ;
     s[0] = xe ; s[1] = ye ;
     s[2] = ty ;
     fwrite( s,8,3,fy) ;
    }
  /* Graficacion de los ronchigramas  */
  if( ii==2 )
   {
    jy = (int)(ye*DRG) ;
    ix = (int)(xe*DRG) ;
    scrn[NMAX*(IO-jy) + (ix+IO)] =60 ;
    }

 /* Conteo del numero de puntos muestreados */
        NPUN = NPUN + 1 ;
   }  /* if  */
nex:  ;
} /*  i  */
  }  /*  ii */
   /* Etiquetas de final de archivo */
s[0] = 1000.0; s[1] = 1.; s[2] = 1.;
fwrite(s,8,3,fx) ;
fwrite(s,8,3,fy) ;
fclose(fx) ;
fclose(fy) ;
printf( " Numero de puntos =====> %d",NPUN);
}  /* principal */
```

/* PROGRAMA MATRT2 */

```
  /* Programa que forma la matriz de minimos cuadrados */
#include <stdio.h>
#include <math.h>

  /* Subrutina para elevar un numero a la m_esima potencia
     considerando que ambos pueden ser cero */
double pow1(x,m)
int m;
double x;
{
   double v;
   if(m==0) return(1.0);
   else
 {
   v = pow(x,m);
   return (v);
 }
}

 /* Subrutina para evaluar un monomio de la forma (X^i)*(Y^j)
    conociendo unicamente un indice m como funcion de i y j */
double b(m,x,y)
int m;
double x,y;
{
   int i,j ;
   double pol ;
 /* Paso de un indice unidimensional a bidimensional */
   i = (int)( 0.9 + ( -3.+sqrt(9.+8.*(m-1)) )/2. ) ;
   j = m-1 - i*(i+1)/2 ;
   pol = pow1(x,j)*pow1(y,i-j) ;
   return(pol);
}
```

```
/* Programa principal */
 main(argc,argv)
 int argc;
 char *argv[];
{
/* Declaracion de variables */
 double x,y,ta,c[60][60],f[60],z[500][4];
 double pow1,blxy ;
 double ss[10] ;
 int i,k,l,m,n,GRADO,NTER,NPUN;
 int np[2] ;
 char st[15] ;

/* Apertura de archivos */
FILE *fof;
FILE *fob;
FILE *fi;

/* Grado de la funcion de aberracion a ajustar */
   GRADO = 5 ;
printf( " programa ====> MATRT.C ") ;
printf( " GRADO del polinomio bidimensional==> %d",GRADO);

/* Archivos utilizados */
fob = fopen("bsal.dat","wb");
fof = fopen("fsal.dat","wb");
fi = fopen(argv[argc-1],"rb");

/*
   argv[argc-1] es el nombre de un archivo de entrada. Contiene
   las diferencias de TA's con su correspondiente coordenada (x,y).
   Este archivo es TAX.DAT si se esta ajustando el ronchigrama
   vertical; y sera TAY.DAT si es el vertical.
*/
```

```
 /* Numero de terminos de la derivada de la funcio de aberracion */
NTER = GRADO*(GRADO+1)/2 ;

 /* Inicializacion de las matrices utilizadas */
 for(m=1; m<=NTER; m++)
{
 f[m] = 0.0;
 for (n=1; n<=NTER; n++)
{
 c[m][n] = 0.0 ;
}
}
 /* Lectura de los puntos muestreados. Se utiliza el archivo tax.dat
 si estos datos corresponden al ronchigrama vertical, y se utiliza el
 archivo TAY.DAT si es el horizontal */
   NPUN = 0 ;
 for(i=1; i<=1000; i++)
 {
  fread(ss,8,3,fi);
  z[i][1] = ss[0] ;
  z[i][2] = ss[1] ;
  z[i][3] = ss[2] ;
  if( z[i][1] >=1000.0 ) break ;
  NPUN++ ;
 }
 printf( " NPUN ==>%d",NPUN);

 /* Formacion de la matriz de minimos cuadrados */
   for(i=1; i<=NPUN; i++)
 {
  x = z[i][1] ;
  y = z[i][2] ;
  ta= z[i][3] ;
 for(l=1; l<=NTER; l++)
{
```

```
 blxy = b(l,x,y) ;
 f[l] = f[l] + ta*blxy ;
 for(k=1; k<=NTER; k++)
{
 c[l][k] = c[l][k] + blxy*b(k,x,y);
}   /* k */
}   /* l */
}   /* i */
 fclose(fi);

 /*  Escritura de los datos en los archivo bsal.dat y fsal.dat */
 np[0] = NTER ;
 fwrite(np,2,1,fob) ;
 for(m=1; m<=NTER; m++)
 {
  np[0] = m ;
  ss[0] = f[m] ;
  fwrite(ss,8,1,fof);
   for (n=1; n<=NTER; n++)
   {
    np[1] = n ;
    ss[0] = c[m][n] ;
    fwrite(np,2,2,fob) ;
    fwrite(ss,8,1,fob) ;
   }   /* n */
 }   /* m */
 fclose(fof);
 fclose(fob);
}/* main */
```

```
                /*  PROGRAMA  I N V E R _ T   */
/*
  inversa de la matriz de minimos cuadrados formada en MATRT2.C
  referencia: Numerical Recipes
*/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define KG  5
#define N   (KG)*(KG+1)/2
#define ND N+1
#define  NP  N
double c[ND][ND], a[ND][ND], y[ND][ND], f[ND],b[ND] ;
int indx[ND];

 /* subroutine LUDCMP(a,N,NP,indx,d)  */
void ludcmp(d)
int  d;
{
 int ie, je, i, j, k, imax;
    double  vv[100], tiny=1.0e-20, aamax;
 double sum, dum;
 d = 1;
    /* loop over rows to get the implicit scaling information */
 for( i = 1; i<=N; i++ )
 {
   aamax = 0.0;
   for(j=1; j<=N; j++)
   if( fabs(a[i][j]) > aamax ) aamax = fabs( a[i][j] );
                    /* nonzero largest element */
   if( aamax == 0.0 ) { printf("matriz singular"); abort(); }
   vv[i] = 1./aamax ;  /* save scaling */
    }
```

```
for( j=1; j<=N; j++ ) /* the loop over columns of Crout's method */
 {
 if( j > 1 )
    {
      for( i=1; i <= (j-1); i++ )
      {
         sum = a[i][j];
   if( i > 1) { for( k=1; k<=(i-1); k++) sum -= a[i][k]*a[k][j];
        a[i][j] = sum;
        }
     }

    }
 aamax = 0.0;  /* initialize for the search for largest pivot element */
 for( i=j; i<=N; i++)
 {
  sum = a[i][j];
  if( j > 1 )
    {
    for(k=1; k<=(j-1); k++) sum -= a[i][k]*a[k][j];
    a[i][j]=sum;
   }
  dum = vv[i]*fabs(sum); /* figure of merit for the pivot */
/* printf("");*/
  if( dum >= aamax ){
        imax = i;
        aamax = dum;
      }
 }
 if( j != imax ){
     for( k=1; k<=N; k++ )
      { dum = a[imax][k];
       a[imax][k]=a[j][k];
       a[j][k]=dum;
```

```
    }
    d = -d;
    vv[imax]=vv[j];
   }
 indx[j]=imax;
 if( j != N ) {
         if( a[j][j] == 0.0 ) a[j][j] = tiny;
         dum = 1./a[j][j];
         for( i=(j+1); i<=N; i++ ) a[i][j]*=dum;
        }
 }
 if( a[N][N] == 0.0 ) a[N][N] = tiny;
}

/* sunbroutine LUBKSB(a,N,NP,indx,b) */
void lubksb()
{
    double  sum;
 int ii=0, i, ll, j ;
 for( i=1; i<=N; i++ )
 {
  ll = indx[i];
  sum = b[ll];
  b[ll] = b[i];
  if( ii != 0 ) for( j=ii; j<=(i-1); j++ ) sum -= a[i][j]*b[j];
  else if( sum != 0.0 ) ii = i;
  b[i] = sum;
 }
 for( i=N; i>=1; i-- )
 {
  sum = b[i];
  if( i < N ) for( j=(i+1); j<=N; j++ ) sum -= a[i][j]*b[j];
  b[i] = sum/a[i][i];
 }
}
```

```
   /* Programa principal */
 main(argc,argv)
 int argc;
 char *argv[];
{
   /* Declaracion de variables */
 double so,x1, inv ;
 int d,i,j,l,m,NPUN,IDEN ;


 int i1,j1,l1;
 double ss[10] ;
 int np[2] ;
   /* Apertura de archivos */
 FILE *fp;
 FILE *fp1;
 FILE *fp2;
 fp = fopen("bsal.dat","rb") ;
 fp1 = fopen("fsal.dat","rb") ;
 fp2 = fopen(argv[argc-1],"wb") ;

 /*
   argv[argc-1] es el nombre de un archivo de salida. Sera SOLTAX.DAT
   si se esta ajustando un ronchigrama vertical, y sera SOLTAY.DAT si
   se ajusta un horizontal.
 */

   /* lectura de datos */
 fread(np,2,1,fp) ;
 NPUN = np[0] ;
 printf(" orden de la matriz %d",NPUN);

for( i1=1; i1<=N; i1++)
  {
```

```
    fread(ss,8,1,fp1) ;
    f[i1] = ss[0] ;
    for(j1=1; j1<=N; j1++)
  {
    fread(np,2,2,fp) ;
    l = np[0] ;
    m = np[1] ;
    fread(ss,8,1,fp);
    x1 = ss[0] ;
    a[l][m] = x1;
    c[l][m] = x1;
  }
   }

for( i=1; i<=N; i++ )
   {
    for( j=1; j<=N; j++ ) y[i][j] = 0.0;
    y[i][i] = 1.0;
   }
ludcmp( d ) ;
for( j = 1; j<=N; j++ )
   {
    for( i = 1; i<=N; i++ ) b[i] = y[i][j];
    lubksb();
    for( i=1; i<=N; i++ ) y[i][j]=b[i];
   }

   /*  calculo del vector solucion  */
  np[0] = N ;
  fwrite(np,2,1,fp2) ;
  for(i=1; i<=N; i++)
   {
   so = 0.0;
```

```
 for(j=1; j<=N; j++)
  {
so = so + y[i][j]*f[j] ;
  }
  /* paso de un indice unidimensional a bidimensional */
i1 = (int)( 0.9 + (-3. + sqrt(9.+8.*(i-1)))/2) ;
m = i-1-((i1+1)*i1)/2 ;
np[0] = i1 ;
np[1] = m ;
ss[0] = so ;
fwrite(np,2,2,fp2) ;
fwrite(ss,8,1,fp2) ;
 }
 } /* main  */
```

```
/*   PROGRAMA   M A T F O C _ R   */

/*
   Formacion de la matriz de minimos cuadrados de orden 6 que contiene
   los errores de centramiento en las direcciones "x" y "y", el error de
   defoco, un termino constante y las contribuciones a la aberraciones
   de COMA y ASTIGMATISMO
*/

#include <stdio.h>
#include <math.h>
#include <dos.h>

#define    ABS(x)     ( x<0 ?   -x:x )
#define  GRADO  5
#define  AJUSTE 6
#define   PASO   25

 /* Evaluacion de las expresiones de Seidel */
double bb(m,x,y)
 int m;
 double x,y;
 {
   int i,j ;
   double pol ;
   if( m==1 )   pol = 1  ;            /* Constante */
   if( m==2 )   pol = x  ;            /* Tiltx */
   if( m==3 )   pol = y  ;            /* Tilty */
   if( m==4 )   pol = y*(x*x+y*y) ;   /* Coma  */
   if( m==5 ) pol = x*x+3.*y*y ;     /* Astigamtismo */
   if( m==6 )   pol = x*x+y*y ;       /* Defoco */
   return(pol);
  }
 double pow1(m,x)
 int m;
```

```
 double x;
 {
 double v;
 if(m==0) return(1.0);
 else
 {
   v = pow(x,m);
   return (v);
 }
}
  /* programa principal */
  main( )
 {
    int i,j,k,l,m,n,NTER,NPUN;
  double s,wxy,tax,tay,OPTIMO;
  double b[11][11],bc[11][11],bd[11][11],c[11][11],d[11][11];
  double a,LAM,Exy,er,si,rms,pico ;
  double sx[3],sy[3],delta,deform ;
  double xmax,ymax,xmin,ymin,wmax,wmin ;
  double A1,A2,A3,A4,s2,ta,polex,argumento ;
  double r,DEFOCO,picomax,picomin,picopico ;
  int npx[2],npy[2],x1,x2,x3,PRAD,np[5] ;
  char st[15] ;
  double x,y,aux,aux1,aux2 ;
  double cc[10][10],f[10],ss[10],bixy ;

    /* Archivos utilizados */
 FILE *fc;
 FILE *fd;
 FILE *fof;
 FILE *fob;
 fc = fopen("soltax.dat","rb");
 fd = fopen("soltay.dat","rb");
 fob = fopen("bsal.dat","wb");
 fof = fopen("fsal.dat","wb");
```

```
LAM = 0.00006328;
printf ( " Diametro del espejo (cm)" ) ;
  fgets( st,15,stdin);  a = atof(st) ;
  printf( " a ====> %f ",a );
  a = a/2.;
/*  soltax.dat ==>  archivo generado por INV4.C que contiene los
      coeficientes de la aproximacion en la dire -
      ccion x  */

/*  soltay.dat ==> archivo generado por INV4.C en la direccion y */
/*  tay.dat ===>   archivo generado por ATRANY4.c. Contiene las coor-
     denadas (x,y,tay) */
/*  solfin.dat ==> Archivo de salida para graficacion */

 /* lectura de datos */
fread(npx,2,1,fc) ;
fread(npy,2,1,fd) ;
printf(" GRADOx ===> %d",GRADO) ;
printf(" GRADOy ===> %d",GRADO) ;
NTER = GRADO*(GRADO + 1)/2;

for (i=1; i<=NTER; i++)
 {
  /* lectura de datos de soltax.dat */
 fread(npx,2,2,fc); fread(sx,8,1,fc);
   l = npx[0]; m = npx[1];
   tax = sx[0]; c[l][m] = tax ;

  /* lectura de datos de soltay.dat */
 fread(npy,2,2,fd); fread(sy,8,1,fd);
   j = npy[0] ; k = npy[1] ;
   tay =sy[0] ; d[j][k] = tay ;
 }
```

```
 /* calculo de los coeficientes de la funcion de aberracion
    usando unicamente datos del ronchigrama vertical */
for ( i=1; i<=GRADO; i++)
 {
  for ( j=1; j<=i; j++)
   {
  aux = c[i-1][j-1] ;
  bc[i][j] = a*aux/(double)(j) ;
   }
 }
 /* calculo de los coeficientes de la funcion de aberracion
 usando unicamente datosb del ronchigrama horizontal */
for ( i=1; i<=GRADO; i++)
 {
   for ( j=0; j<i; j++)
  {
    aux = d[i-1][j] ;
    bd[i][j] = a*aux/(double)(i-j);
     }
  }
/* Promedio de los coeficientes anteriormente encontrados,
   teniendo en cuenta sus restricciones */
for ( i=1; i<=GRADO; i++)
 {
   for( j=0; j<=i; j++)
 {
   if( j == 0 ) { aux = bd[i][j] ; b[i][j] = aux; }
   else {
    if(j == i) { aux = bc[i][j]; b[i][j] = aux; }
    else { aux1=bd[i][j]; aux2=bc[i][j];
     b[i][j] = ( aux1 + aux2 )/2. ;
      }
   }
  }}
```

```
 /* Inicializacion de las matrices utilizadas */
 for(m=1; m<=AJUSTE; m++)
{
 f[m] = 0.0;
 for (n=1; n<=AJUSTE; n++)
{
 cc[m][n] = 0.0 ;
}
}
   NPUN = 0 ;
   delta= 1.0/PASO ;
 for(j=-PASO; j<=PASO; j++)
  {
  y= j*delta ;
 for(i=-PASO; i<=PASO; i++)
  {
  x = i*delta ;
  si = sqrt( x*x + y*y ) ;
   if( si <=1.0 )
    {
  NPUN = 1 + NPUN ;

 /* Discretizacion de la funcion de aberracion W(x,y) anteriormente
   encontrada */
   s = 0.0 ;
   for(l=1; l<=GRADO; l++)
   {
       for(m=0; m<=l; m++)
    {
   s = s + b[l][m] * pow1(m,x)*pow1(l-m,y);
        }
    }

 /* Uso de un parametro de amortiguamiento */
    deform =  s/(1000.0*LAM) ;
```

```
/* Formacion de la matrix de minimos cuadrados */
 for(l=1; l<=AJUSTE; l++)
  {
 blxy = bb(l,x,y) ;
 aux1 = f[l] ;
 f[l] = aux1 + deform*blxy ;
   for(k=1; k<=AJUSTE; k++)
    {
  aux2 = cc[l][k] ;
  cc[l][k] = aux2 + blxy*bb(k,x,y);
    }   /* k */
   }}   /* l */
} /*  i  */
} /*  j  */
/*  Escritura de las matrices de minimos cuadrados en sus
  respectivos archivos */
np[0] = AJUSTE ;
fwrite(np,2,1,fob) ;
for(m=1; m<=AJUSTE; m++)
{
  np[0] = m ;
  ss[0] = f[m] ;
  fwrite(ss,8,1,fof);
  for (n=1; n<=AJUSTE; n++)
   {
  np[1] = n ;
  aux = cc[m][n] ;
  ss[0] = aux ;
  fwrite(np,2,2,fob) ;
  fwrite(ss,8,1,fob) ;
   }   /* n */
  }  /* m */
fclose(fof);
fclose(fob);
} /* final */
```

```
                /* PROGRAMA  I N V F O C - R  */
/*
   inversa de la matriz de errores formada en MATFOC_R.C
   referencia: Numerical Recipes
*/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define KG 5
#define N  6
#define ND N+1
#define NP N
double a[ND][ND], c[ND][ND], y[ND][ND], f[ND],b[ND] ;
int indx[ND];
/* subroutine LUDCMP(a,N,NP,indx,d)  */
void ludcmp(d)
int  d;
{
 int ie, je, i, j, k, imax;
    double  vv[100], tiny=1.0e-20, aamax;
 double sum, dum;
 d = 1;
    /* loop over rows to get the implicit scaling information */
 for( i = 1; i<=N; i++ )
 {
   aamax = 0.0;
   for(j=1; j<=N; j++)
   if( fabs(a[i][j]) > aamax ) aamax = fabs( a[i][j] );
                    /* nonzero largest element */
   if( aamax == 0.0 ) { printf("matriz singular"); abort(); }
   vv[i] = 1./aamax ;  /* save scaling */
    }
 for( j=1; j<=N; j++ ) /* the loop over columns of Crout's method */
 {
 if( j > 1 )
    {
```

```
      for( i=1; i <= (j-1); i++ )
      {
         sum = a[i][j];
   if( i > 1) { for( k=1; k<=(i-1); k++) sum -= a[i][k]*a[k][j];
        a[i][j] = sum;
        }
     }
    }
 aamax = 0.0;  /* initialize for the search for largest pivot element */
 for( i=j; i<=N; i++)
 {
  sum = a[i][j];
  if( j > 1 )
    {
    for(k=1; k<=(j-1); k++) sum -= a[i][k]*a[k][j];
    a[i][j]=sum;
   }
  dum = vv[i]*fabs(sum); /* figure of merit for the pivot */
/* printf("");*/
  if( dum >= aamax ){
        imax = i;
        aamax = dum;
      } )
 if( j != imax ){
     for( k=1; k<=N; k++ )
      { dum = a[imax][k];
       a[imax][k]=a[j][k];
       a[j][k]=dum;
     }
    d = -d;
    vv[imax]=vv[j];
   }
 indx[j]=imax;
 if( j != N ) {
         if( a[j][j] == 0.0 ) a[j][j] = tiny;
         dum = 1./a[j][j];
```

```
          for( i=(j+1); i<=N; i++ ) a[i][j]*=dum;
         }
 }
 if( a[N][N] == 0.0 ) a[N][N] = tiny;
}

/* sunbroutine LUBKSB(a,N,NP,indx,b) */
void lubksb()
{
    double  sum;
 int ii=0, i, ll, j ;
 for( i=1; i<=N; i++ )
 {
  ll = indx[i];
  sum = b[ll];
  b[ll] = b[i];
  if( ii != 0 ) for( j=ii; j<=(i-1); j++ ) sum -= a[i][j]*b[j];
  else if( sum != 0.0 ) ii = i;
  b[i] = sum;
 }
 for( i=N; i>=1; i-- )
 {
  sum = b[i];
  if( i < N ) for( j=(i+1); j<=N; j++ ) sum -= a[i][j]*b[j];
  b[i] = sum/a[i][i];
 }
}

  main( )
{
 double so,x1, inv ;
 int d,i,j,l,m,NPUN,IDEN ;
 int i1,j1,l1;
 double ss[10] ;
 int np[2] ;
```

```
/* Archivos utilizados */
 FILE *fp;
 FILE *fp1;
 FILE *fp2;
 FILE *fs;

 fp = fopen("bsal.dat","rb") ;
 fp1 = fopen("fsal.dat","rb") ;
 fp2 = fopen("sale_focu.dat","w+") ;
 fs  = fopen("grafica2.dat","r+") ;

   /* lectura de datos */
 fread(np,2,1,fp) ;
 NPUN = np[0] ;
 printf(" orden de la matriz %d",NPUN);
 for( i1=1; i1<=N; i1++)
  {
 fread(ss,8,1,fp1) ;
 f[i1] = ss[0] ;
 for(j1=1; j1<=N; j1++)
    {
 fread(np,2,2,fp) ;
 l = np[0] ;
 m = np[1] ;
 fread(ss,8,1,fp);
 x1 = ss[0] ;
 a[l][m] = x1;
 c[l][m] = x1;
    }}

  for( i=1; i<=N; i++ )
   {
  for( j=1; j<=N; j++ ) y[i][j] = 0.0;
  y[i][i] = 1.0;
   }
```

```
ludcmp( d ) ;
for( j = 1; j<=N; j++ )
   {
    for( i = 1; i<=N; i++ ) b[i] = y[i][j];
    lubksb();
    for( i=1; i<=N; i++ ) y[i][j]=b[i];
   }

   /*  calculo del vector solucion  */
   fprintf(fp2, " %d",N);
  for(i=1; i<=N; i++)
   {
      so = 0.0;
      for(j=1; j<=N; j++)
 {
  so = so + y[i][j]*f[j] ;
 }
 if(i==1) printf(" CONS= %f",so);
 if(i==2) printf(" TILTX= %f",so);
 if(i==3) printf(" TILTY= %f",so);
 if(i==4) printf(" COMA= %f",so);
 if(i==5) printf(" ASTIG= %f",so);
 if(i==6) printf(" DEFOCO= %f",so);

   /* paso de un indice unidimensional a bidimensional */
      if (i==1)
      {
       np[0] = i-1 ;
       np[1] = i-1 ;
       ss[0] = so ;
    fprintf(fp2, " %d %d %le", i-1,i-1,so);
    fprintf(fs," %f",so);
      }
      else
      {
```

```
for(i1=1; i1<=N-1; i1++)
  {
    for(m=0; m<=i1; m++)
     {
       IDEN = i1*(i1+1)/2 + (m+1);
       if( IDEN == i)
        {
    fprintf(fp2," %d %d %le", i1, m, so);
    fprintf(fs," %f",so);
        }
     }
  }
 }
     }
   } /*  final  */
```

```
/*    PROGRAMA  G R A _ P R U   */

/* Programa para calcular elos errores RMS, PICO, VALLE
   y para formar el archivo de datos a graficar */

#include <stdio.h>
#include <math.h>
#define    ABS(x)    ( x<0 ?   -x:x )
#define  GRADO  5
#define   PASO   25

   sgn(x)
   double x ;
{
   double y ;
   if(x==0) y=0 ;
   else y=x/ABS(x) ;
   return(y) ;
}
/* calculo de la potencia entera de un real considerando que ambos pueden
   ser ceros  */
double pow1(m,x)
int m;
double x;
{
 double v;
 if(m==0) return(1.0);
 else
 {
   v = pow(x,m);
   return (v);
 }
}
```

```
/* programa principal */
  main( )
 {
  double b[11][11],bc[11][11],bd[11][11],c[11][11],d[11][11];
  double a,LAM,si,rms,pico;
  double sx[3],sy[3],delta,deform;
  double xmax,ymax,xmin,ymin,wmax,wmin ;
  double ta,poler,wxy,r,s,tax,tay ;
  float CTE,TILTX,TILTY,COMA,ASTIG,DEFOCO ;
  float x,y,rad_inf,rad_sup ;
  int i,j,k,l,m,n,NTER,NPUN,npx[2],npy[2];
  char st[15] ;
 FILE *fc;  FILE *fd;
 FILE *fp;  FILE *fe;
 LAM = 0.00006328;
   a = 7.5 ;

/* Archivos de datos utilizados */
 fc = fopen("soltax.dat","rb");
 fd = fopen("soltay.dat","rb");
 fe = fopen("grafica2.dat","r");
 fp = fopen("data","w");

   /* lectura de datos */
 fread(npx,2,1,fc) ;
 fread(npy,2,1,fd) ;
 printf(" GRADOx ===> %d",GRADO) ;
 printf(" GRADOy ===> %d",GRADO) ;

 NTER = GRADO*(GRADO + 1)/2;
 for (i=1; i<=NTER; i++)
  {
    /* lectura de datos en <x> */
   fread(npx,2,2,fc); fread(sx,8,1,fc);
   l = npx[0]; m = npx[1] ;
   tax = sx[0]; c[l][m] = tax ;
```

```
   /* lectura de datos en <y> */
  fread(npy,2,2,fd); fread(sy,8,1,fd);
  j = npy[0] ; k = npy[1] ;
  tay =sy[0] ; d[j][k] = tay ;
 }

 /* calculo de Bij  */
for ( i=1; i<=GRADO; i++)
 {
  for ( j=1; j<=i; j++)
   {
     bc[i][j] = (a*c[i-1][j-1])/(j) ;
   }
 }

for ( i=1; i<=GRADO; i++)
 {
   for ( j=0; j<=i-1; j++)
     {
       bd[i][j] = (a*d[i-1][j])/(i-j);
     } }

for ( i=1; i<=GRADO; i++)
{
  for( j=0; j<=i; j++)
  {
    if( j == 0 ) b[i][j] = bd[i][j];
    else if(j == i) b[i][j] = bc[i][j];
 else  b[i][j] = ( bd[i][j] + bc[i][j] )/2. ;
  }
}
  /* Lectura de los coeficientes de error */
fscanf(fe,"%f",&CTE);
printf( " CTE ==>%f",CTE);
fscanf(fe,"%f",&TILTX);
printf( " TILTX ==>%f",TILTX);
```

```
fscanf(fe,"%f",&TILTY);
printf( " TILTY ==>%f",TILTY );
fscanf(fe,"%f",&COMA);
printf( " COMA ==>%f",COMA);
fscanf(fe,"%f",&ASTIG);
printf( " ASTIG ==>%f",ASTIG);
fscanf(fe,"%f",&DEFOCO);
printf( " DEFOCO ==>%f",DEFOCO);

  /* Pregunta limites de la grafica */
   printf( "Rango de analisis?") ;
   printf( "Radio inferior? --[0,1]") ;
   fgets(st,15,stdin) ;
 rad_inf= atof(st) ;
   printf( "Radio superior? -->[Rad-inf,1]") ;
   fgets(st,15,stdin) ;
 rad_sup= atof(st) ;

    /* Asignacion de valores iniciales para encontrar los errores
    RMS y PICO_PICO  */
   NPUN = 0 ;
   xmax =-1000.0 ;
   ymax =-1000.0 ;
   wmax =-1000.0 ;
   xmin = 1000.0 ;
   ymin = 1000.0 ;
   wmin = 1000.0 ;
 rms = 0.0 ;
   delta= 1.0/PASO ;
for(j=-PASO; j<=PASO; j++)
 {
 y= j*delta ;
for(i=-PASO; i<=PASO; i++)
 {
 x = i*delta ;
 si = sqrt( x*x + y*y ) ;
```

```
   if( si>rad_sup || si<rad_inf)
 {
   deform = 0.0 ;
   fprintf( fp," %f ",deform);
 }
 if( si <=rad_sup && si >=rad_inf)
  {
NPUN = 1 + NPUN ;
 s = 0.0 ;

  /* Evaluacion del polinomio de error */
poler= CTE +TILTX*x +TILTY*y +DEFOCO*(x*x + y*y) ;

 /* Evaluacion del polinomio aberrado por minimos cuadrados  */
 for(l=1; l<=GRADO; l++)
 {
     for(m=0; m<=l; m++)
  {
 s = s + b[l][m] * pow1(m,x)*pow1(l-m,y);
      }
  }

 /* Sustraccion del termino de error al polinomio de
    aberracion */
  deform =  s/(2*1000.0*LAM) - poler/2;
  wxy=deform ;

  if( si > 0.0)       /*  Se exceptua el si=0  */
 {

 /* calculo de los valores maximos */
   rms = rms + wxy*wxy ;   /* Calculo del RMS  */
   if( x > xmax ) xmax = x ;
   if( y > ymax ) ymax = y ;
   if( wxy > wmax ) wmax = wxy ;
```

```
      /* calculo de los valores minimos */
    if( x < xmin  ) xmin = x ;
    if( y < ymin )  ymin = y ;
    if( wxy < wmin ) wmin = wxy ;

 }  /* Escritura del archivo de datos a graficar */
    fprintf( fp," %f ",deform);
   }
} /*  i  */
} /*  j  */
   printf( "  ymin= %f ymax= %f ",ymin,ymax) ;
   printf( " xmin= %f xmax= %f",xmin,xmax) ;
   printf( " wmin= %f wmax= %f",wmin,wmax ) ;
   printf( " rms = %f", sqrt(rms/(double)NPUN) ) ;
} /* final */
```

BIBLIOGRAFIA

1.- *Cornejo-Rodríguez A., "Ronchi Test", en Optical Shop testing, Jhon Wiley and Sons, New york, (1978), cap. 9.*

2.- *Rayces, J. L., "Exact Relation between wave aberration and ray aberration, " Opt. Acta, 11 85(1964).*

3.- *Malacara, D. "Ronchi test and transversal Spherical aberration, " Bol. Obs. Tonantzintla Tacubaya, 4, 73(1965).*

4.- *Malacara D. ed. "Optical Shop Testing", Wiley, New York (1978), Apéndice 1.*

5.- *Alcalá, Noé y Landgrave, Enrique " Reducción de datos en la prueba de Hartmann utilizando splines bicuadrados", Tésis de licenciatura, C.I.O, León, Gto.*

6.- *Cornejo, A y Malacara, D. "Wavefront determination using Ronchi and Hartmann test", Bol. Obs. Tonantzintla Tacubaya, 2,127,(1976).*

7.- *Lenz, Reimar "Tutorial Notes on CCD-Camera Calibration", Held at the Workshop on Digital Image Processing, Centro de Investigaciones en Optica [1987].*

8.-*Cornejo, A. y Malacara, D. "Ronchi test of aspherical surfaces, analysis and accuracy", Applied Optics 9,1897,(1970).*

9.- Yatagai,T. , Nakadate,S. Idesawa,M. and Saito,H. "Automatic Fringe Analysis Using Digital Image Processing Techniques", Opt. Eng. 21,432,(1982).

10.- González Rafael, C y Wintz, Paul [1987], Digital Image Processing, Addison-Wesley Publishing company, pp 398-402.

11.- Ghozeil, I. "Hartmann & other screen tests", en Optical shop testing, Jhon Wiley and Sons, New york, (1978), cap. 9.

12.- Pérez Santos Carlos, "Pruebas de Superficies Cóncavas Asféricas con el método de Burch-Offner",Rep. Tec. No. 10, Centro de Investigaciones en Óptica, (1988).

13.- Morales A., Malacara D., Menchaca C, "Pruebas Geométricas de Ronchi y del Alambre Para Superficies Ópticas Asféricas", Rep. Tec. No. 6, Centro de Inv. en Óptica.