CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS
DEL
INSTITUTO POLITECNICO NACIONAL

DEPARTAMENTO DE INGENIERIA ELECTRICA
SECCION DE COMPUTACION

LOCALIZACION DE PUNTOS DE SOLDADURA EN CIRCUITOS IMPRESOS USANDO LA TRANSFORMADA DE HOUGH

Tesis que presenta el Licenciado en Computación Leopoldo Altamirano Robles [1] para obtener el grado de MAESTRO EN CIENCIAS en la especialidad de Ingenieria Eléctrica. Trabajo dirigido por el Prof. B. Radig de la Universidad Técnica de Munich y codirigido por la parte del CINVESTAV por el Dr. Guillermo Morales Luna

México, D.F. Enero de 1991

[1]Miembro del Instituto de Física de la Universidad Autónoma de Puebla

Contenido

Resumen

El objetivo del presente trabajo es el de evaluar tres métodos usados en Visión por Computadora para encontrar los bordes de un objeto situado dentro de una imagen digitalizada. La evaluación de los métodos se hace en base a la eficiencia que presenta la transformada de Hough para localizar puntos de soldadura (solder joints) en circuitos impresos, usando los bordes de los objetos producidos por los tres métodos. Para obtener una evaluación aceptable los métodos son utilizados para encontrar los puntos de soldadura en un conjunto de circuitos impresos, obteniéndose de este proceso estadísticas que ayudan a evaluarlos.

A mis Padres:

Leopoldo Altamirano Pérez

y

Graciela Robles Pérez

AGRADECIMIENTOS

Primero que nada quiero dar las gracias al Prof. B. Radig por la posibilidad que me brindó de trabajar bajo su dirección asi como por su acertada asesoria durante la elaboración de este trabajo. A la Dr. S. Pfleger, por el trabajo conjunto sobre diversos temas no solo relacionados directamente con la tésis, sino relacionados con otras áreas de la computación, lo cual hizo posible la exitosa conclusión de este trabajo.

Posteriormente quiero agradecer al ahora Instituto de Física por el apoyo institucional que hizo posible mi estancia aqui en Munich, Alemania Federal, donde se realizó este trabajo. Especialmente al Ing. Luis Rivera Terrazas (en memoria) y a los Doctores José Luis Carrillo, Germán Luna, Hugo Navarro y Pedro Hugo Hernández.

Al CINVESTAV-IPN, por la aceptación de este trabajo. En especial al Dr. Guillermo Morales por su apoyo y sus anotadas observaciones en relación a este trabajo. Al Dr. Sergio Chapa V. y al Dr. José Luis Gordillo por la revisión del mismo.

A mi familia entera, por su apoyo tanto moral como económico, sin los cuales no se podria haber llevado acabo esta tésis.

A Estela Gómez por la revisión y ayuda moral durante la realización de este trabajo, los cuales aminoraron el esfuerzo necesario para la elaboración de este documento.

Finalmente a mis amigos: Héctor Osorio, Tomás León, Aurelio López, Alfonso Garcés, Lilia Meza y Yolanda Sánchez por su apoyo indirecto para hacer posible este trabajo.

Capítulo 1

LA TRANSFORMADA DE HOUGH

En esta sección se explica el funcionamiento y las aplicaciones de la Transforma de Hough, así como la versión de ésta que aquí se usa.

1.1 Introducción

La interconexión de Circuitos Integrados (CI) entre ellos mismos y con otros dispositivos es una etapa muy importante en el diseño y construcción de sistemas digitales. La importancia que tiene la interconexión de CI's se puede notar más en el número de patas que un CI tiene, el cual se ha venido incrementando en los últimos años, de circuitos con 6, 8 y 14 patas hasta los más modernos con 140 ó 150 patas. Para realizar esta conexión entre CI's usando circuitos impresos existen muchas técnicas. Dos de las técnicas más importante son: la técnica de dispositivos montados sobre superficies (surface mounted devices) y la técnica que usa orificios aluminizados (plated holes). Esta última técnica usa orificios que se hacen sobre la tableta (en la que se ha fijado el circuito impreso) en los lugares donde se desea una conexión, insertando posteriormente en los orificios la pata del CI, uniendolos a continuación con soldadura. Esta técnica fue una de las más usadas anteriormente, ya que actualmente la técnica de dispositivos montados sobre superficies la está desplazando, debido principalmente a sus ventajas. No obstante, la mayoría de fábricas de dispositivos digitales siguen utilizando la técnica de orificios aluminizados, las cuales se enfrentan a los diferentes problemas que esta técnica presenta.

Uno de los problemas graves de la técnica de orificios aluminizados es el error producido al soldar el orificio con la pata del CI. Esta unión puede

presentar muchos tipos de errores, que van desde la insuficiencia de soldadura, hasta el sobrecalentamiento de las puntas del cautí, las cuales originan el derramamiento de la soldadura sobre los hoyos cercanos. Los errores son más graves cuando son ocasionados por sistemas automáticos de soldadura, los cuales no tienen un sistema de retroalimentación que controle la calidad de las uniones producidas por la soldadura de los hoyos y las patas del CI lo cual hace difícil su corrección inmediata, ocasionando pérdidas considerables a las compañías elaboradoras de sistemas digitales, aún cuando el promedio de errores en las uniones sea tan bajo como el 0.05%. Es debido a ésto que se han empezado a crear sistemas que puedan evaluar automáticamente las condiciones de una unión producida con soldadura. Hasta la fecha existen muy pocos sistemas que lleven a cabo esta tarea y la mayoria de los existentes están en la fase de sistemas de investigación, los cuales todavia no han madurado lo suficiente como para formar parte de un sistema comercial. Entre ellos podemos mencionar los sistemas elaborados por [Nak 87] y [Bar 88].

Una de las razones por las cuales no se ha podido desarrollar sistemas automáticos para la evaluación de este tipo de uniones es debido al hecho de que el problema de analizarlas es bastante complejo, ya que básicamente es un problema que se debe tratar en 3 dimensiones, puesto que la soldadura junto con la pata del CI no genera formas planas (i.e. en dos dimensiones), sino que da origen a objetos en forma de montañas con las más diversas características y volúmenes, ocasionando que los patrones o formas-ejemplo usados para la aceptación de una unión, como una unión en buen estado, sean muy númerosos y con características muy diferentes, además de los problemas que ocasiona la alta reflexividad de la soldadura.

Al tratar de crear un sistema automático para la inspección de uniones se encuentra uno primeramente con el problema de la localización, i.e. la computadora no tiene conocimiento en dónde se encuentran éstas en la imagen digitalizada que se está analizando. Como el objetivo a largo plazo es la creación de sistemas automáticos, no se puede exigir que el usuario del sistema localice manualmente las uniones, sino que el sistema debe de ser capaz de hacerlo por sí solo.

La búsqueda de una solución a los problemas antes mencionados es la razón del surgimiento de este trabajo, en el cual se evaluan los métodos de "Split" y "Merge", detector óptimo de bordes y un método basado en el operador Roberts que son usados para la búsqueda de los bordes de las uniones, los cuales describiremos en las siguientes secciones.

1.2 Transformada de Hough (TH)

La Transformada de Hough es una técnica muy popular en Visión por Computadora usada principalmente para la localización de objetos en imagenes digitalizadas [Ill 88]. Esta transformada fue introducida en 1962 por P. Hough [Hou 62] y fue usada para detectar curvas en fotografías de una cámara con burbujas. En 1969 Rosenfeld [Ros 69] introdujo esta técnica al área de procesamiento de imagenes. Desde entonces las aplicaciones de la TH han aumentado muy rápido, las cuales incluyen: la localización de los bordes de carreteras para la conducción de robots, detección de características en sismogramas, reconocimiento de caracteres hebreos, análisis de huellas digi tales, etc. Una de las razones por las cuales la TH ha sido de mucha utilidad se debe a que los objetos tanto naturales como los hechos por el hombre se pueden descomponer en líneas rectas y otras curvas simples, las cuales pueden ser detectadas muy fácilmente por la TH.

Funcionamiento. En general la TH transforma la imagen original en otra imagen, en la cual es mas fácil la localización de ciertas características que determinan la posición de un objeto. En este sentido esta transformada es muy parecida a la Transformada de Fourier (TF) solamente que en la imagen producida por la TF, es más difícil localizar las características que definen a los objetos que se quieran localizar.

Con la TH transformamos características que se encuentran muy dispersas o incompletas en la imagen original en características que se presentan agrupadas en la imagen transformada, de esta forma la TH convierte la detección global de un objeto, que es un problema difícil, en la detección puntual de ciertas características, que es un problema relativamente sencillo. Pasemos ahora a dar un ejemplo de esta transformación.

Definamos 2 tipos de espacio:

- Un espacio en el cuál esta situada la imagen original, el cual denotaremos por la letra I y los pares cartesianos (x, y) y que denominaremos como espacio original o espacio de la imagen. Cada par cartesiano (x,y) representa un pixel, esto significa que cada par (x,y) tiene asociado un valor i, que es el nivel de gris en la posición (x,y). Las dimensiones del espacio son las mismas que las de la imagen original, en este caso 512×512.

$$I = \{(x, y)\} \qquad 0 \leq x \leq 511, 0 \leq y \leq 511 \tag{1.1}$$

- El otro espacio que definiremos es el espacio al cual será transformada la imagen original y que denominaremos espacio de parámetros o espacio

paramétrico P. El espacio está constituido por todos los pares (α, β) definidos dentro del rango válido de los parámetros. Hay que hacer notar que la dimensión y el rango válido del espacio paramétrico dependen del tipo de características con las que se esté trabajando. En nuestro caso consideraremos a P como un espacio de 2 dimensiones:

$$P = \{(\alpha, \beta)\} \tag{1.2}$$

La TH define entonces una transformación entre el espacio de la imagen y el espacio paramétrico

$$TH : I \rightarrow P \tag{1.3}$$

el cual es un mapeo de un elemento de I a muchos de P:

$$TH : (x, y) \mapsto \{(\alpha, \beta)\} \tag{1.4}$$

La transformación de un punto perteneciente al espacio paramétrico la tenemos que definir también como una transformación de uno a muchos, ya que un par (α, β) nos debe definir la posición del objeto buscado y este par nos debe de reproducir al objeto original, compuesto de una sucesión de puntos pertenecientes a I.

Para el caso en el cual el objeto de interés es una línea recta, los espacios quedarían definidos:

$$\begin{aligned} I &= \{(x, y)\}, \quad x \in [0, 511], y \in [0, 511] \\ P &= \{(\alpha, \beta)\}, \quad \tan(\alpha) = m \in [-\pi/2, \pi/2], \quad \beta = k \in [0, 511] \end{aligned} \tag{1.5}$$

debido a la definición de la ecuación de una linea recta:

$$y = mx + k \tag{1.6}$$

representada gráficamente en la Fig. 1

Los espacios I y P son representados gráficamente para la ecuacion (1.6) con parámetros $m = 1$ y $k = 2$ en las Fig. 2 y 3.

Los parámetros $m = 1$ y $k = 2$ no se conocen de antemano, sino que son precisamente los parámetros buscados. En la práctica solo se tienen las coordenadas de todos los puntos que aparecen en la imagen original. Supongamos que tenemos los puntos con coordenadas $(0, 2), (1, 3), (2, 4)$ y $(2, 2)$. Si a estos puntos los sustituimos en la ecuación (1.7), nos definen 4 lineas rectas en el espacio paramétrico, ya que estos puntos vienen a sustituir las variables x y y en esa ecuación:

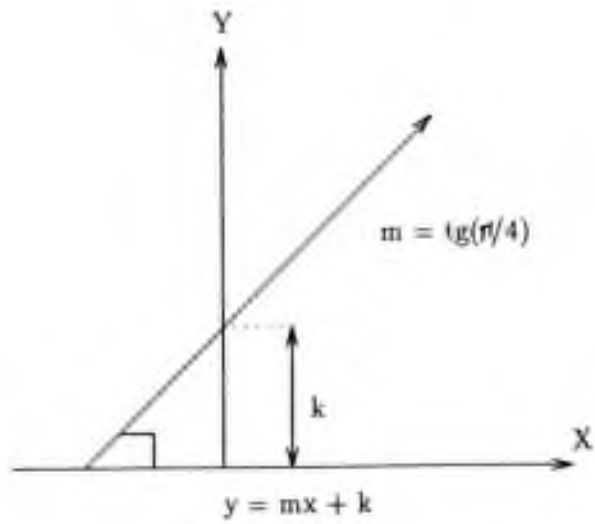

Fig. 1. Representación gráfica de los parámetros de la ecuación de una linea recta

$$TH(x,y) : k = y - mx \tag{1.7}$$

$$k = 2, \qquad k = 3 - m, \qquad k = 4 - 2m, \qquad k = 2 - 2m \tag{1.8}$$

Se puede decir entonces que estos puntos fueron transformados en lineas rectas. La representación gráfica de las ecuaciones se puede ver en la Fig. 3. Nótese el punto de intersección P1 producido por las 3 primeras líneas que provienen de los tres primeros puntos (los que definen una línea recta en el espacio de la imagen) a diferencia de la cuarta línea que no coincide con las otras tres en este punto. Las coordenadas del punto de coincidencia nos definen un par de parámetros que al ser proyectados sobre el espacio de la imagen nos definen una línea recta, localizándonos de ésta forma el objeto deseado. En este caso $m = 1$ y $k = 2$.

En la práctica los dos espacios están representados por matrices, las cuales contienen la información necesaria para realizar la TH. La matrix de la imagen original es la matriz que contiene los valores de los diferentes niveles de gris que nos definen al objeto. La matrix del espacio paramétrico contiene en cada una de sus localidades un contador que se incrementa en uno, cada vez que una linea recta pasa sobre él (ver Fig. 4).

En la figura también se puede ver como existe un contador con un valor mayor que el de los demás, de esta forma la búsqueda de los parámetros deseados se reduce a la búsqueda del máximo en esa matrix.

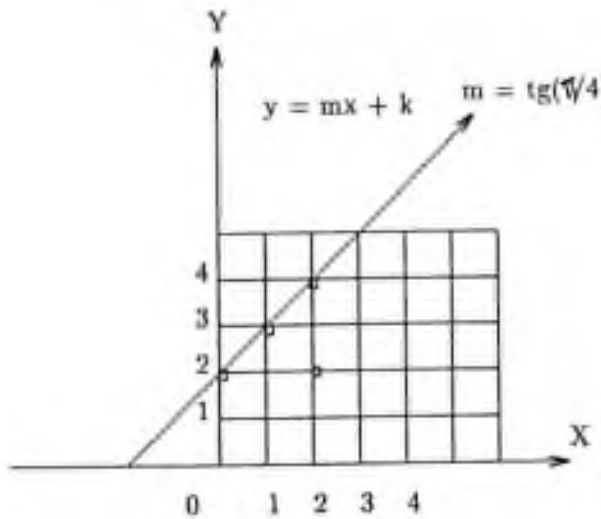

Fig. 2. Espacio de la imagen (I)

La TH puede verse como un procedimiento de votación, en el cual cada punto vota por un conjunto de parámetros. Dos de las ventajas del metodo es que puede ser paralelizado fácilmente, y de que como cada punto vota independientemente de los otros, es posible reconocer objetos parcialmente ocultos. La propiedad de reconocer objetos semiocultos se basa en la transformación de cada pixel del contorno, la cual se realiza independientemente de la existencia de los demas pixeles. La decisión del reconocimiento de un objeto se basa entonces en los elementos del contorno que existan. Naturalmente hay que considerar la cuestión de si los elementos del contorno presentes son suficientes para determinar unívocamente a un objeto.

La TH puede ser adaptada para localizar no solamente lineas rectas, sino tambien otro tipo de curvas. En la siguiente sección se explica una adaptación de la TH para localizar círculos.

1.3 Aplicación de la TH para la localización de círculos

Para el caso de la localización de círculos por medio de la TH el espacio paramétrico se convierte en un espacio de tres dimensiones, puesto que un círculo está determinado por tres parámetros. Los parámetros los podemos determinar al considerar la ecuación del círculo.

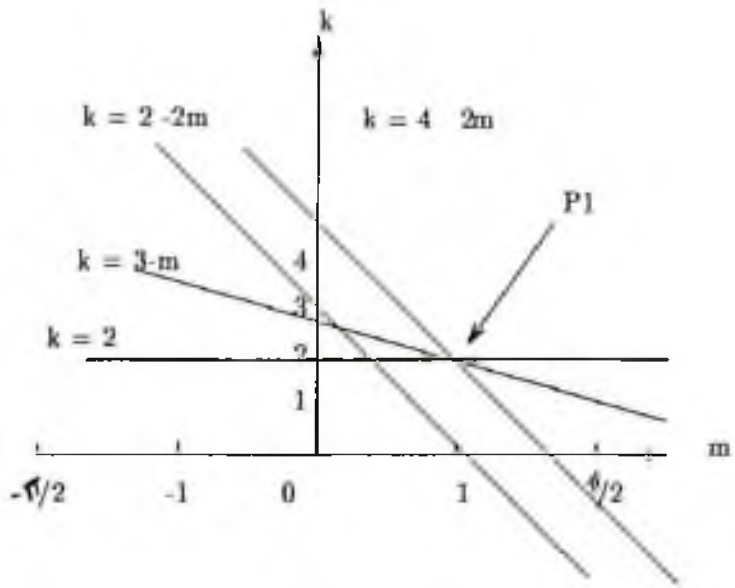

Fig. 3. Espacio paramétrico (P)

$$(x-a)^2-(y-b)^2=r^2 \tag{1.9}$$

De aquí se puede ver que los parámetros que definen a un círculo son a, b, y r es decir, las coordenadas de su centro y la longitud de su radio. Se tiene que hacer notar que en este caso la búsqueda del máximo elemento en el espacio paramétrico no es sencilla, ya que se tiene que hacer una búsqueda en un espacio de tres dimensiones. Apesar del inconveniente mencionado se han desarrollado métodos que logran encontrar el máximo valor en espacios de 3 dimensiones [Ill 88].

En el presente trabajo se usa una simplificación para el cálculo de la TH para obtener un espacio paramétrico de dos dimensiones y poder realizar una búsqueda más sencilla. La simplificación que se hizo fué la siguiente: de acuerdo a [Ill 87] primero se trata de calcular los parámetros a y b que definen el centro del círculo, dejando para después la determinación del parámetro r, ocasionando que tanto el espacio de la imagen como el espacio parámetrico tengan las mismas dimensiones. A cada pixel en la imagen original se le asocia un vector perpendicular al vector gradiente calculado en este pixel. Como consecuencia los pixels que están en el borde[1] de algún círculo tendrán vectores perpendiculares que se cruzan en un punto (punto de acumulación);

[1] En el texto se usan las palabras: borde, orilla y contorno equivalentemente.

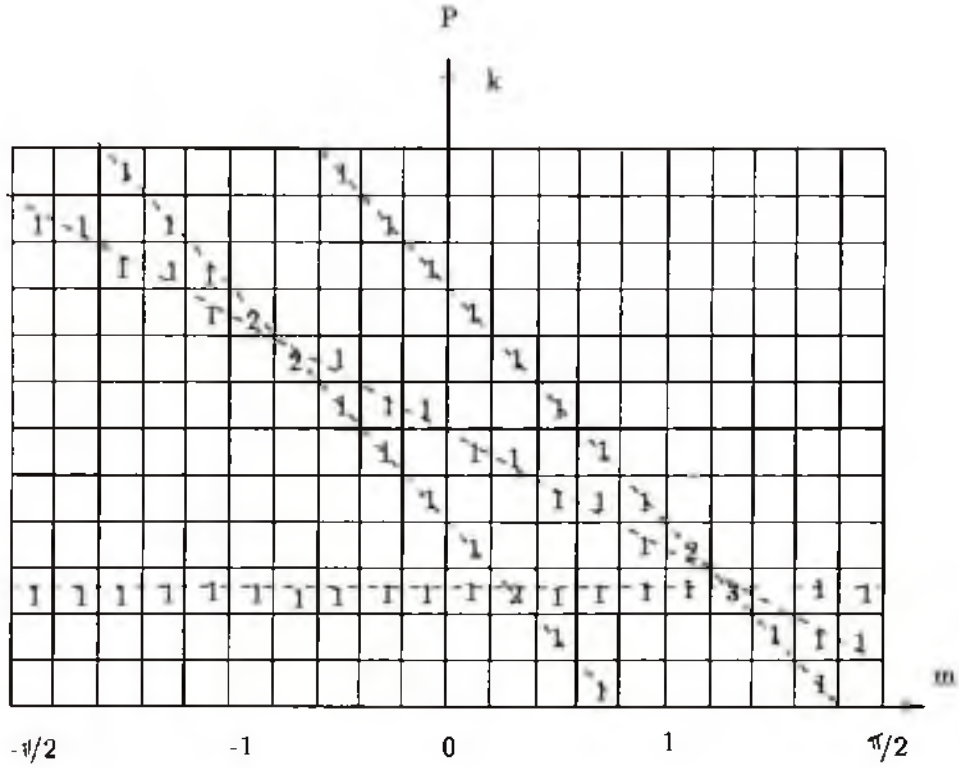

Fig. 4 Matrix que representa el espacio paramétrico.

este punto será precisamente el centro del círculo que ellos definen. Ver Fig. 5, vectores v1, v2, v3. Los puntos que pertenezcan a otros objetos que no tengan forma circular generan también vectores perpendiculares, pero no producen un punto de acumulacion (v4, v5, Fig. 5), en el cual coinciden muchos vectores perpendiculares.

En el caso de la simplificación mencionada, los espacios quedan definidos de la siguiente forma:

$$\begin{aligned} I &= \{(x,y)\} \qquad x \in [0,511], \qquad y \in [0,511] \\ P &= \{(\alpha,\beta)\} \qquad \alpha = a \in [0,511], \qquad \beta = b \in [0,511] \end{aligned} \tag{1.10}$$

y la función de transformación:

$$TH(x,y) : \{\perp (\overline{x,y})\} \tag{1.11}$$

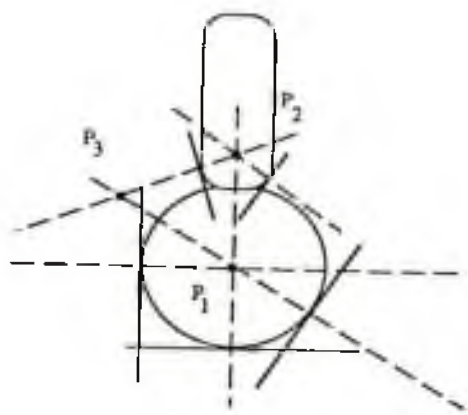

Fig. 5 Vectores perpendiculares y puntos de acumulación.

La ecuación anterior significa que el punto x,y es transformado el vector perpendicular con respecto a la dirección del vector gradiente calculado en (x, y), ver Fig. 6. Gráficamente los espacios están representados por 2 matrices, una matriz para la imagen original y otra matriz acumulador para el espacio P, que tiene la misma función que en el caso de la localización de rectas. Cada elemento de la matriz se puede considerar como un contador que se incrementa en 1 cada vez que un vector perpendicular pasa encima de él. (Fig. 7)

El cálculo de los vectores perpendiculares y la localización de un círculo con la ayuda de estos vectores se lleva a cabo como sigue:

- Cada punto de la imagen original se proyecta en el espacio paramétrico utilizando la función de transformación ya definida, ecuación número (1.11). De esta transformación se obtienen los parámetros que definen a los vectores perpendiculares, los cuales sirven para llenar los elemen tos de la matriz acumulador de la misma forma que en el caso de la localización de rectas.
- A continuación se realiza la búsqueda del elemento máximo en la matriz-acumulador, la cual nos da los parámetros buscados. En este

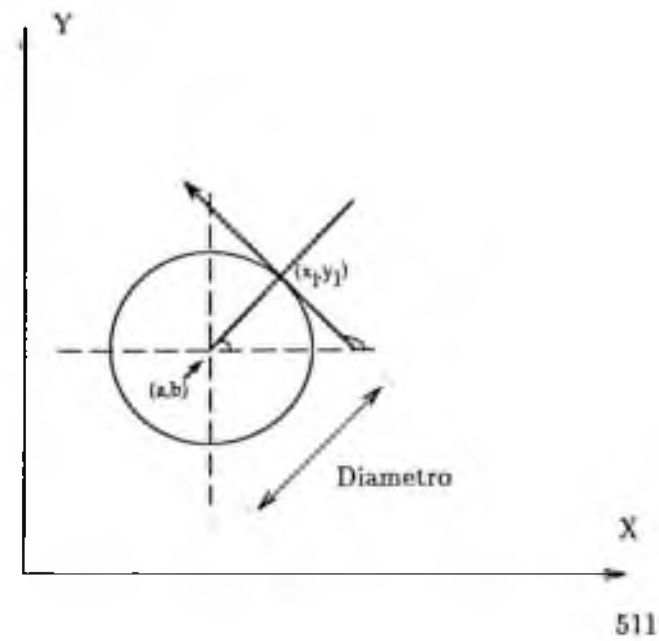

Fig. 6. Representación gráfica del vector tangente al punto (x_1, y_1).

caso la transformación de los parámetros encontrados en el espacio de la imagen no nos reproduce el círculo buscado, sino que solo encontramos las coordenadas de su centro.

- Posteriormente el cálculo del radio del círculo se lleva a cabo mediante la ayuda de un histograma. El histograma se construye a partir de la distancia entre cada pixel perteneciente al borde de cada objeto y el centro del círculo encontrado en el paso anterior. A continuación se encuentra el valor máximo del histograma, el cual será igual al valor del radio buscado. La justificación del procedimiento se basa en el hecho de que la distancia entre los pixels pertenecientes al contorno de el círculo y el centro del círculo es constante, lo que originará el valor máximo en el histograma correspondiente [Ill 88].

Por último para calcular la dirección del vector perpendicular en un pixel específico se parte de que la dirección del vector gradiente esta dada por $\alpha = \tan(G_x/G_y)$

A su vez G_x y G_y se pueden calcular tomando en cuenta los ocho vecinos del pixel donde se piensa calcular el vector, de acuerdo a la siguiente fórmula [Shi 87] también denominada operador Sobel:

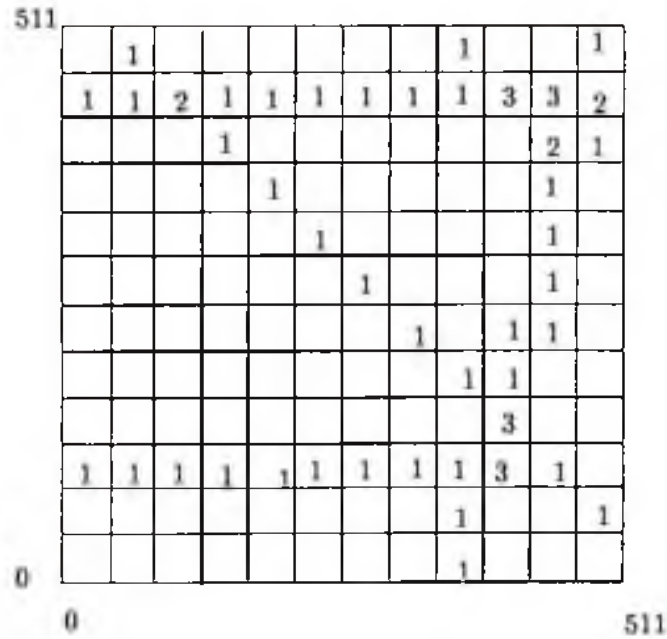

Fig. 7. Espacio paramétrico para la localización de círculos.

$$G_x = \frac{\partial f(x,y)}{\partial y} = (C + 2F + I) - (A + 2D + G) \tag{1.12}$$

$$G_y = \frac{\partial f(x,y)}{\partial y} = (A + 2B + C) - (G + 2H + I) \tag{1.13}$$

donde los vecinos del pixel (x, y) están definidos como:

A	B	C
D	x,y	F
G	H	I

De esta forma también:

$$\alpha = \tan(Gx/Gy) \qquad \theta = \alpha + \pi/2 \tag{1.14}$$

donde α es la dirección más pronunciada de $f(x,y)$ en (x,y) y θ es la dirección del borde.

1.4 Transformada de Hough Adaptiva

El primer algoritmo de la TH que se uso en el presente trabajo para la localización de círculos fue el algoritmo propuesto en [Ill 88], el cual se denomina Transformada de Hough Adaptiva. Esta versión de la transformada usa una matriz-acumulador de solo 9x9 elementos, lo que hace al procedimiento bastante atractivo, debido a la poca memoria usada por la matriz acumulador (en comparación a una matriz de 512×512 elementos que usan otros algoritmos). Naturalmente la reducción de las dimensiones de la matriz acumulador lleva consigo un aumento en la complejidad del software que es utilizado para manipularla como se verá a continuación. El cálculo de la TH adaptiva difiere de la TH normal, tan solo en la forma en la que se evalúa a los vectores perpendiculares y la forma en la que se procesan los máximos de la matriz-acumulador. Expliquemos la primera diferencia.

Una vez que se tienen los parámetros de un vector normal este se evalúa en todo el espacio paramétrico cuantizado, es decir, como solo se tienen 9 columnas en la matriz- acumulador, el rango del parámetro se tiene que discretizar, obteniéndose obviamente, sólo 9 puntos donde se puede calcular el vector perpendicular, Fig. 8. Debido a la discretización del espacio parámetrico el punto de intersección de los vectores perpendiculares no se produce en un solo elemento de la matriz, sino en un conjunto de elementos, los cuales forman un agrupamiento (cluster), ver en la Fig. 8 las regiones sombreadas.

Los bordes de los agrupamientos son buscados por medio de un algoritmo de segmentación el cual encuentra todos los elementos conectados en una matriz (es decir, elementos con el mismo valor) y después son analizados estadísticamente, buscando entre otras cosas el número de elementos dentro del agrupamiento, para ayudar a la eliminación de clusters que no corresponden a puntos de cruce. Una vez que se ha escogido el agrupamiento más apropiado, el valor de sus bordes sirve para volver a discretizar el espacio parámetrico y realizar un acercamiento más real al punto de intersección de los vectores perpendiculares, es decir, el centro del círculo buscado. Partiendo de la Fig. 8 el nuevo espacio paramétrico estaría comprendido dentro del área que abarcan las flechas, $[192-448, 128-256]$. Este procedimiento es parecido a la función "zoom" que algunos equipos de fotografía poseen. Para realizarlo se necesitan arreglos que guarden los valores actuales del espacio paramétrico, así como las rutinas necesarias para actualizarlos y consultarlos. Al algoritmo se le puede considerar como la adaptación de la TH al rango de parámetros que se tenga, de ahí que se le llame "adaptiva".

La segunda diferencia de la TH normal con la adaptiva se origina cuando

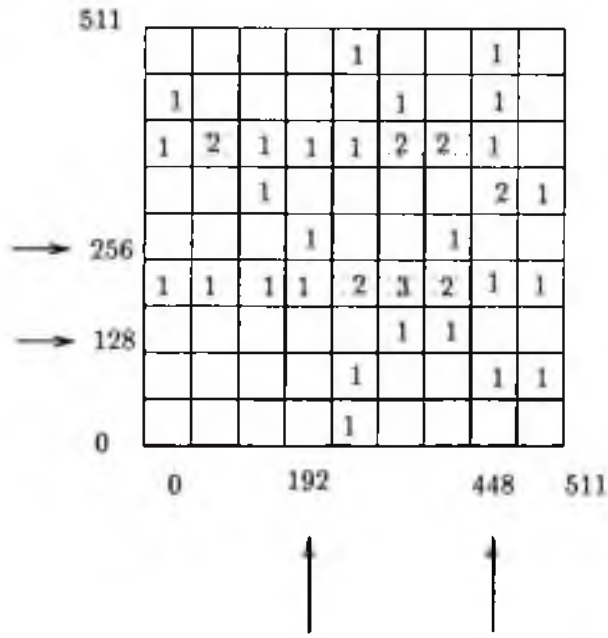

Fig. 8. Matriz-acumulador para la TH adaptiva.

se tienen varios puntos de soldadura en la imagen original, lo que ocasiona que se tengan varios agrupamientos en la matriz-acumulador. Para poder procesar todos los agrupamientos, se tiene que tener una lista de espera en la cual se van colocando los "clusters" que faltan por analizar. De esta lista se toma el primer elemento y se reduce al rango del espacio paramérico en concordancia con las dimensiones del "cluster". Se vuelven a calcular los vectores perpendiculares en este nuevo rango y se buscan los agrupamientos en la matrix-acumulador. Las dimensiones de los agrupamientos se vuelven a guardar en la lista de espera y se elige un nuevo "cluster" para seguir analizando. La sequencia anterior se repite hasta que se alcanza una aproximación deseada, ya que con cada iteración se reduce el rango en el que un agrupamiento puede estar. Los últimos valores del rango del espacio paramétrico son las coordenadas del centro del circulo buscado.

1.5 Límites de la TH

Existen varios problemas con los que se encuentra uno al programar la TH, los cuales mencionamos a continuación.

El cálculo de la dirección correcta de los vectores perpendiculares a los bordes de los objetos y en especial de los vectores perpendiculares al contorno de los círculos es muy importante, puesto que si se producen desviaciones en la dirección, los vectores perpendiculares no se cruzan en un punto. La parte en la que se debe tener más cuidado es en la sección que calcula el operador Sobel, puesto que de aqui se encuentran α y θ (ver la primera sección de este capítulo). El operador Sobel es uno de los mejores entre los operadores calculados en una vecindad de 3x3 elementos ya que produce contornos de muy buena calidad pero no obstante introduce mucho error en el cálculo de α y θ como se puede ver en la siguiente tabla.

Variaciones angulares de 2 operadores.

Angulo exacto	Angulo calculado con el Operador Soberl	Angulo calculado con el Operador Circular
0	0.0	0.0
5	4.97	5.47
10	9.95	10.81
15	15.0	15.83
20	19.99	20.07
25	24.42	24.62
30	28.86	29.89
35	33.64	35.43
40	38.87	40.30
45	45.0	45.0
Error (rms en grados)	0.75	0.47

Tratando de eliminar el error producido usamos un operador circular que reduce el error en un 30% [Dav 84]. Los coeficientes para el cálculo del operador en la dirección x se dan a continuación:

Y

0.0	-0.294	0.0	0.294	0.0
-0.582	-1.0	0.0	1.0	0.58
-1.085	-1.0	0.0	1.0	1.085
-0.582	-1.0	0.0	1.0	0.582
0.0	-0.294	0.0	0.294	0.0

X

Estos coeficientes se multiplican con el valor de los 24 vecinos del pixel donde se esta calculando el operador y después se suman, obteniéndose como

resultado la aplicación del operador en la dirección x. Para encontrar los coeficientes del operador en la dirección y se realiza una rotación de 90^o de de la matriz anterior en sentido contrario a las manecillas del reloj.

Otro de los problemas que se tienen con la TH es el número de puntos de cruce falsos, es decir, puntos en los que se origina un agrupamiento a pesar de que éste no sea el centro de un círculo, ver en la figura 5 los puntos P_2 y P_3.

Una de las primeras razones del surgimiento de los puntos falsos es el cálculo de los vectores perpendiculares en todo el rango del espacio paramétrico. Una forma alternativa de evaluar los vectores perpendiculares es reducir el espacio de evaluación a la longitud del diámetro de los círculos esperados y solo hacia el centro del círculo, ver Fig. 5. El diámetro de los círculos se le da al programa como dato de entrada, no necesariamente exacto sino aproximado. La dirección que apunta hacia el centro del círculo se obtiene al calcular el operador Sobel sobre la imagen original, ya que aplicando el operador en diferentes lados de un disco se obtienen diferentes direcciones, es decir, diferentes θ y α. En nuestro caso cada punto de soldadura se puede considerar como un disco si se toma este en la imagen original. (Nótese la diferencia entre un disco y un círculo. En el primero cada valor de los pixels interiores es aproximadamente el mismo que el valor de los pixels del borde. En un círculo tanto los pixels internos como los externos son iguales y difieren del valor de los pixels de los bordes).

Por otra parte, al usar el acumulador de dimension 9×9 se tuvieron varios problemas. Por ejemplo, en la mayoría de las imagenes analizadas la matriz-acumulador no presentaba ningún máximo local que tuviera un valor mucho mas grande que los otros o los valores máximos estaban mal situados con respecto al centro de los círculos, Otros problemas fueron la no convergencia de la TH adaptiva, y la no detección de la totalidad de los círculos. Es por esto que se decidió en una segunda versión del programa aumentar el tamaño de la matriz-acumulador a una matriz de 512×512, con la cual se obtuvieron los resultados que en el siguiente capítulo se presentan. Por último, el cálculo de los vectores perpendiculares no se hizo en toda la imagen sino solamente en los pixels que pertenecian al borde de objetos (conjuntamente con bordes causados por el ruido). Los bordes fueron almacenados ademas en otra matriz, la cual sirve como una variable lógica, ya que si existe un borde en el pixel que se va a procesar, se procede a calcular su vector perpendicular, de otra manera no se lleva a cabo el cálculo. Con esta última mejora el programa aumenta su velocidad considerablemente, pero por otro lado hace al programa dependiente de la calidad de los bordes que se obtienen. Debido a la dependencia mencionada se analizan en el siguiente capítulo tres diferentes métodos para encontrar bordes, los cuales fueron aplicados a la imagen

original antes de obtener los vectores perpendiculares.

1.6 Comparación de la TH con el método de Minimos Cuadrados

Existen otros métodos utilizados para la localización de objetos en imagenes digitales, entre los que podemos mencionar el de Mínimos Cuadrados y el de Template Matching. En los siguientes párrafos compararemos el primer método con la TH para mostrar las ventajas de esta última.

El método de Mínimos Cuadrados consiste, expresado brevemente, en la reducción al máximo del error cuadrático que se origina entre los puntos observados $x_1, ...x_n$ y los puntos de referencia $y_1, ...y_n$, los cuales representan al modelo del objeto que se esta buscando en la imagen digitalizada. Este procedimiento tiene como objetivo la minimización de la siguiente función:

$$\epsilon^2 = \sum_{i=1}^{n} (y_n - x_n)^2 \tag{1.15}$$

Uno de los principales problemas que se encuentra uno al aplicar la ecuación anterior es la alta sensibilidad que tiene con respecto al ruido, lo cual se manifiesta en los resultados poco satisfactorios que se obtienen al tratar de ajustar un conjunto de puntos que contiene algunos bastante dispersos.(Ver. Fig. 9) [Har 89].

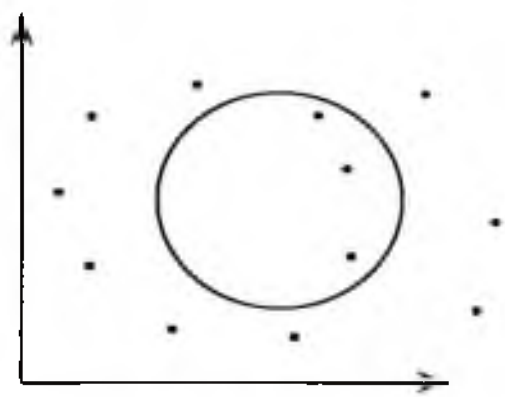

Fig. 9. Localización por medio de Mínimos Cuadrados.

Otro problema con el que se encuentra uno es el hecho de que el conjunto de puntos que se quiere ajustar no se origina siguiendo un orden que

corresponda al de los puntos que forman al modelo a comparar, sino en una forma aleatoria. Tratando de resolver este problema se han propuesto otras formulas para el ajuste por Mínimos Cuadrados. Entre otras esta la siguiente [Har 89]:

$$\epsilon^2 = \sum_{n=1}^{n} \omega_n \|y_n - (Rx_n + t)\|^2 \tag{1.16}$$

dónde R representa una matriz de rotación y t una traslación realizada sobre el conjunto de puntos $\{x_n\}$. Los ω_n son factores que determinan que tan importante es un punto para la localización del objeto buscado y son determinados de una forma iterativa. El objetivo en este caso es determinar R y t los cuales minimizan la sumatoria de los errores ϵ^2. (El proceso completo puede verse en [Har 89]).

Debido a los problemas que ocasiona la aplicación del método de Mínimos Cuadrados, aún cuando se usara la fórmula con los factores ω_n, es que se decidio utilizar la TH para la localización de puntos de soldadura en circuitos impresos, ya que para el caso de la TH no se necesita que los puntos esten ordenados de alguna forma, ademas de que la TH es hasta cierto punto estable aún con la presencia de ruido.

Capítulo 2

ALGORITMOS PARA LA DETECCION DE BORDES

En este capítulo examinamos el método basado en el operador Roberts, el algoritmo de "split" y "merge" y el detector óptimo de bordes, utilizados para la obtención de bordes de objetos en una imagen digitalizada.

2.1 Método basado en el operador Robert

El primer método que se utilizó para detectar los bordes de los objetos contenidos en la imagen original está compuesto de tres etapas. La primera etapa consiste en segmentar la imagen original, es decir, la separación de los objetos presentes del fondo. En nuestro caso se trata de separar los puntos de soldadura y las pistas metálicas de la tableta en la cual se construyó el circuito impreso, como puede observarse en la imagen 1 de las conclusiones. Para realizar dicha segmentación se usó el método del Umbral Dinámico, que como su nombre lo indica, utiliza un umbral diferente para cada pixel. Normalmente una imagen contiene ruido, el que ocasiona que los objetos no sean representados nitidamente. Asi por ejemplo la representación en tres dimensiones de la Fig. 5 se muestra en la Fig. 10 y la representación gráfica del renglón de la misma figura, marcado con una flecha, en la Fig. 11. En las figuras se nota como debido a la iluminación insuficiente y a la óptica usada para tomar la imagen, algunas regiones están más iluminadas que otras (ver también Imagenes 7, 9 y 10).

En la figura 11 se nota la necesidad de segmentar con un umbral variable, ya que si llevamos a cabo la segmentación con un umbral constante no se obtienen buenos resultados. Por ejemplo, si llevamos a cabo la segmentación

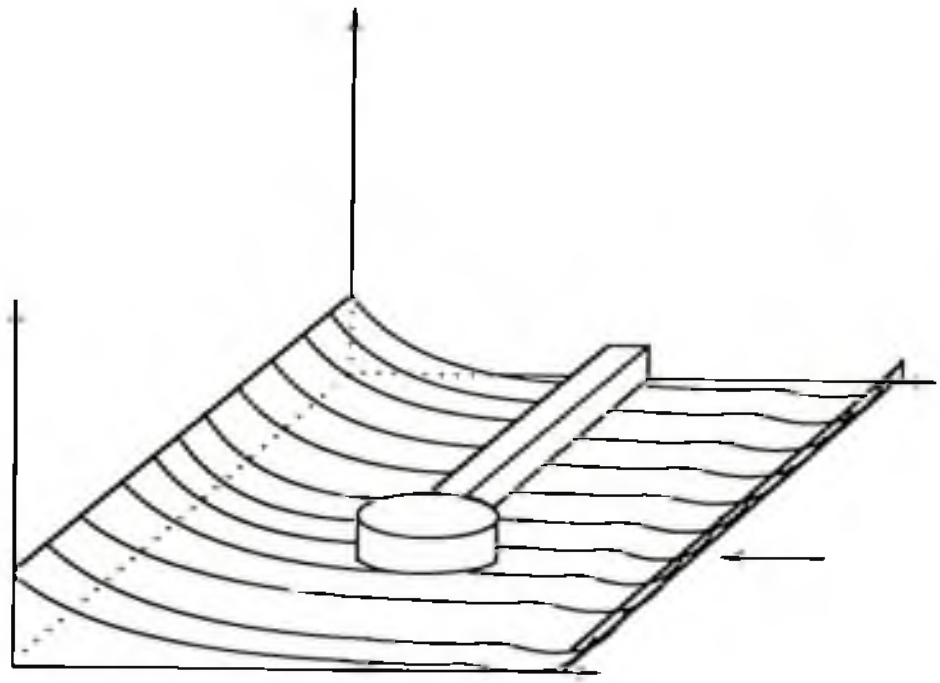

Fig. 10. Representación en 3 dimensiones de un Punto de Soldado.

con un umbral que tiene un valor de 150, eliminamos el ruido de la parte derecha de la Fig. 10, pero no el de la parte izquierda.

Para obtener mejores resultados se hizo la segmentación con un umbral dinámico, el cual se obtiene al suavizar la imagen original. Esta suavización se lleva a cabo al sustituir cada pixel de la imagen original con el promedio del valor de sus pixels vecinos. A la imagen resultante de este proceso se le suma una constante y sus valores son usados como un umbral, el cual varía en cada pixel. Ver Fig. 12.

El tamaño de la vecindad que hay que tomar para llevar a cabo la suavización, dependende del tamaño de los objetos que se estén analizando. Así por ejemplo para pixels de tamaño 2×2 se utiliza una vecindad de 3×3. El procedimiento de segmentación quedaría expresado entonces de la siguiente forma:

Sea $f(x, y)$ la imagen original, $g(x, y)$ la imagen segmentada, $s(x, y)$ la imagen suavizada con $x \in [0, 511]$ y $y \in [0, 511]$ y v el número de pixels vecinos que se debe de considerar para la suavización de la imagen (por ejemplo, si la vecindad es igual a 3×3 entonces $v = 1$).

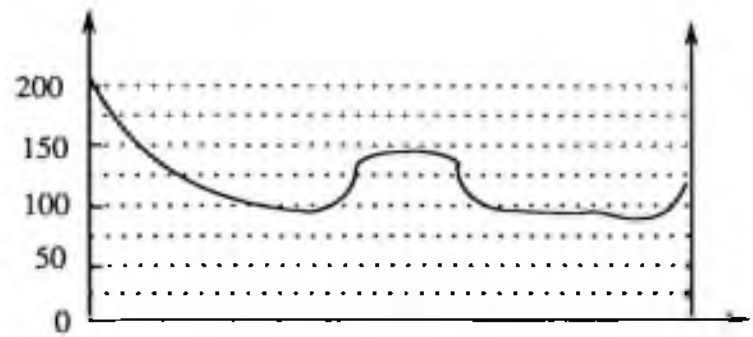

Fig. 11. Sección transversal de un Punto de Soldado.

$$s(x,y) = \sum_{i=v-x, i\neq x}^{v+x} \sum_{j=v-y, j\neq y}^{v+y} f(i,j)/8v \tag{2.1}$$

(2.2)

$$g(x,y) = \begin{cases} f(x,y) & \text{si} \quad f(x,y) > s(x,y) + k \\ 0 & \text{si} \quad f(x,y) \leq s(x,y) + k \end{cases}$$

Después que se hizo la segmentación (ver Fig. 14), el siguiente paso es la detección de los bordes de los objetos presentes, mediante el uso del operador Roberts. Se escogió este operador por la razón de que produce bordes delgados. La desventaja del operador es que es muy sensible al ruido y a consecuencia ante la presencia de ruido produce bordes irreales. En nuestro caso como este operador se aplica a una imagen ya segmentada, el ruido que distorsiona a la imagen fue ya suprimido, lo que asegura buenos resultados con el operador.

El operador Roberts se define como:

$$b(x,y) = |f(x,y) - f(x+1, y+1)| + |f(x+1,y) - f(x,y+1)| \tag{2.3}$$

Los resultados obtenidos con el operador se muestran en la Fig. 15.

Después de la detección de los bordes lo único que nos queda por hacer es la eliminación de pixels aislados, los que a pesar de todo el proceso llevado a

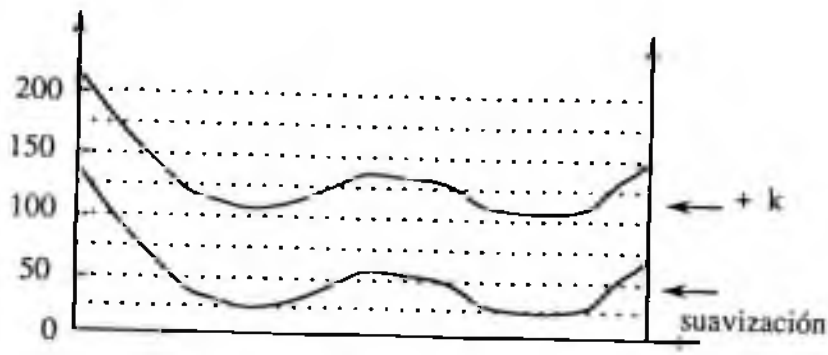

Fig. 12. Resultado de la suavización de la Fig. 12.

cabo, permanecieron en la imagen, y la reducción de la anchura de los bordes ya obtenidos. Esta reducción es necesaria porque al aplicar la TH, cada punto perteneciente a un borde implica una gran cantidad de cálculos, por lo tanto entre más delgados sean los bordes menor es el tiempo de procesamiento.

El adelgazamiento de los bordes se llevo a cabo de acuerdo al siguiente algoritmo.

Se dice que un pixel (x, y) es un pixel conector si cumple la siguiente característica [Tri 70]:

$$(x,y) \text{ es conector} \longleftrightarrow \sum_{i=1}^{8} a(i;x,y) \neq a(i+1;x,y) > 2 \qquad i = 1,...,8 \tag{2.4}$$

En la fórmula anterior $a(i;x,y)$ denota a los pixels vecinos del pixel (x,y) en una vecindad de 3×3. Se presupone que la imagen a la cual se le están adelgazando (thinning) los bordes de los objetos contenidos en ella es binaria, i.e., contiene pixels que solamente adquieren el valor 1 ó 0. En algunos casos también pueden tomar los valores max ó 0, donde max es el valor máximo que un pixel puede tener en el sistema que se esté trabajando. Si este es el caso el número 2 de la ecuación (2.4) se tiene que multiplicar por max.

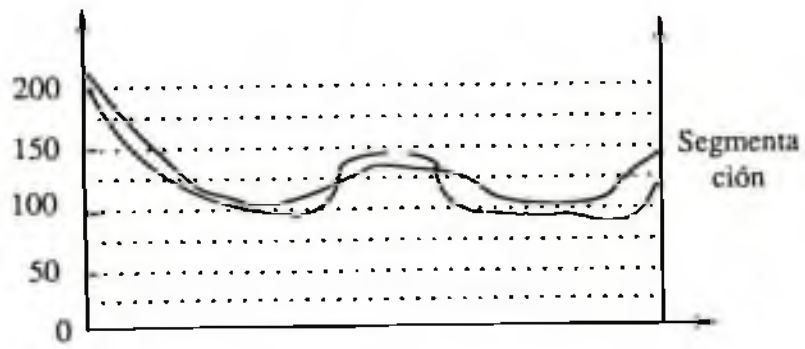

Fig. 13 . Segmentación mediante un Umbral dinámico.

Partiendo de la ecuación (2.4) se tiene entonces que para adelgazar los bordes de un objeto se procede a eliminar todos los puntos que no sean conectores. La justificación de por qué un punto conector no se debe de borrar está basada en el hecho de que el punto conecta al menos a dos grupos separados de pixels, los cuales no deben dejar de estar comunicados para que el objeto al cual estos pixels pertenecen no se deforme. Ver Fig. 16.

La aplicación de este algoritmo a la imagen 1 nos da como resultado la imagen 6.

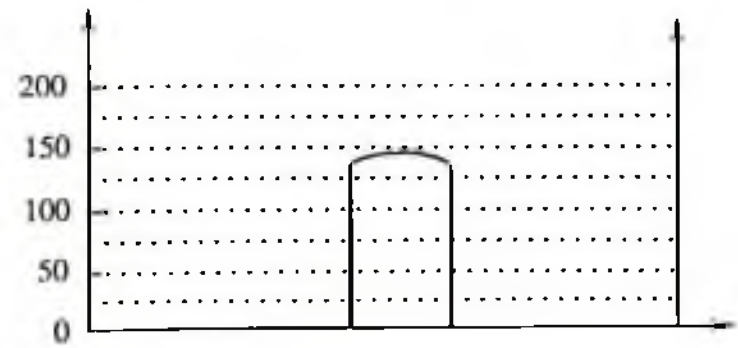

Fig. 14. Resultado de la segmentación de la Fig. 11.

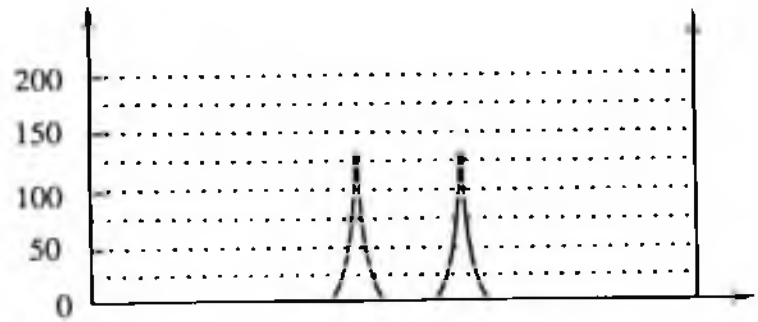

Fig. 15. Bordes de la Fig. 14 obtenidos con el operador Roberts.

2.2 El algoritmo de "Split" and "Merge"

El segundo método que se utilizó para detectar los bordes de los objetos que forman la imagen original es el algoritmo de División y Unión (Split and Merge).

Este método pertenece a la clase de algoritmos que dividen a la imagen en regiones, las cuales deben cumplir con un criterio de homogeneidad que depende del tipo de imagen que se esté considerando. La formulación matemática de este tipo de segmentación es la siguiente [Nie 81]:

Sea f la imagen original, la cual está dividida en regiones p_n, tales que:

$$f \rightarrow \{p_n, n = 1, N\}$$
$$\bigcup_{n=1}^{N} p_n = f \qquad p_i \bigcap p_j = 0 \text{ para } i \neq j \tag{2.5}$$

Cada región además debe de cumplir con un criterio de homogeneidad el cual denotamos como H. Este criterio define una función H(f) la cual es 1 si f satisface H y 0 de otra forma. Por ejemplo H podría ser definida de acuerdo a la siguiente ecuación:

$$H(p_n) = \begin{cases} 1 & \text{si } |f(x1, y1) - f(x2, y2)| < k \\ & \text{para todo } f(x1, y1), f(x2, y2) \in p_n \\ 0 & \text{de otra forma} \end{cases} \tag{2.6}$$

donde k es una constante. Además se debe satisfacer que:

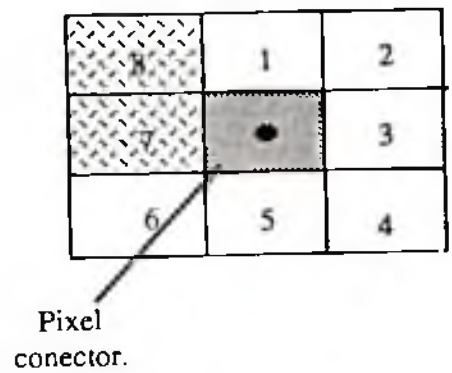

Fig. 16. Definición de un pixel conector.

$$H(p_n) = 1 \text{ para todo } n = 1, ...N \tag{2.7}$$

$$H(p_i \bigcup p_j) = 0 \tag{2.8}$$

para $i \neq j$ siendo p_i, p_j adyacentes

La definición anterior de segmentación nos asegura que cada punto $f(x,y) \in p_i$ debe pertenecer solamente a una región, que todas las regiones p_i son homogéneas en el sentido de H y que la unión de 2 regiones que poseen un borde común causa una violación a H. Más adelante explicamos el criterio de homogeneidad aqui usado. Pasemos ahora a analizar los métodos de Union (Merge) y división (Split) asi como la combinación de ambos.

2.2.1 "Merging"

El algoritmo divide a la imagen original en subimagenes p_n de tal manera que su unión nos vuelve a reproducir la imagen original. Esta división se puede expresar como sigue:

$$f = \bigcup_{n=1}^{N} p_n \tag{2.9}$$

$$p_n(i,j) = f(i,j), \qquad i,j = 1, ..., M \tag{2.10}$$

Cada subimagen p_n tiene $M \times M$ elementos. Esta partición normalmente cumple con la ecuación (2.5) , pero no con las ecuaciones (2.7,2.8). Para satisfacer las se elige un criterio de homogeneidad y se empiezan a unir (merge) regiones adyacentes que cumplan con el criterio. Como un ejemplo de un criterio de homogeneidad se puede escoger que el valor promedio de los pixels que están dentro de cada region no exceda una constante determinada.

$$H(p_i \bigcup p_j) \begin{cases} 1 & \text{si } |\overline{p_i(x,y)} - \overline{p_j(x,y)}| < k \\ & x, y = 1, ...M. \ k \ constante \\ 0 & \text{de otra manera} \end{cases} \tag{2.11}$$

De esta forma se asegura que dos regiones son unidas si es que son "parecidas" con respecto al criterio de homogeneidad. Con esto se evita la union de regiones que podrían pertenecer a diferentes objetos.

La desventaja del método es que para que produzca buenos resultados la partición con la que se empieza debe de ser bastante fina, la cual origina regiones muy pequeñas, cuya unión necesita mucho tiempo de procesamiento. Otra desventaja es que los resultados que se obtienen dependen de la region con la que se empieze el algoritmo.

2.2.2 "Splitting"

En este algoritmo no se empieza con una partición de la imagen original, sino con la imagen original completa, la cual se considera en su totalidad como una región. Esta normalmente satisface la ecuacion (2.5), pero no las ecuaciones (2.7) y (2.8). Para satisfacerlas, lo que se hace es dividir la imagen original en subimagenes, en las cuales se espera que el criterio de homogeneidad se cumpla. Si éste no es el caso se vuelven a subdividir las regiones que no lo satisficieron, procediendo iterativamente hasta que todas las regiones cumplan con el criterio de homogeneidad.

La desventaja del método de "splitting" es que el contorno de las regiones que resultan no tienen una forma suave, sino como si fuera trazada con una linea poligonal.

2.2.3 "Split" and "Merge"

Este método trata de aprovechar las ventajas de los dos métodos anteriores y al mismo tiempo reducir las desventajas.

El criterio de homogeneidad aquí usado para este algoritmo se define a continuación:

$$H(p_n) = \begin{cases} 1 & \text{si } [max(p_n(i,j)) - min(p_n(i,j))] \leq 2\epsilon \\ & \text{para todo } i,j \in 1,...M \\ 0 & \text{en otro caso} \end{cases} \quad (2.12)$$

donde ϵ es una constante. Las operaciones de Unión (Merge) y Divisón (Split) son realizadas con la ayuda de una estructura de datos denominada árbol de segmentación (Segmentation Tree), que en esencia no es más que un Quadtree, al cual se le adhiere en cada nodo información adicional que sirve para realizar las operaciones de Unión y División (por ejemplo el máximo y el mínimo valor dentro de la región, número de elementos, coordenadas de inicio de la región, etc.) [Hor 76]. La raíz del árbol es la imagen original completa, las hojas son pixels o subregiones, cada nodo corresponde a una región de forma cuadrada. El sucesor de cada nodo representa una subregión que resulta de la partición de la región que éste nodo representa en 4 subregiones, como se muestra en las figuras 17 y 18.

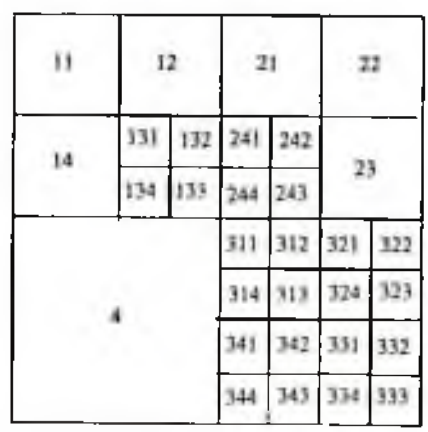

Fig. 17. Estructura de datos para el método de Split and Merge

El algoritmo de Split and Merge se lleva a cabo de acuerdo a los siguientes pasos:

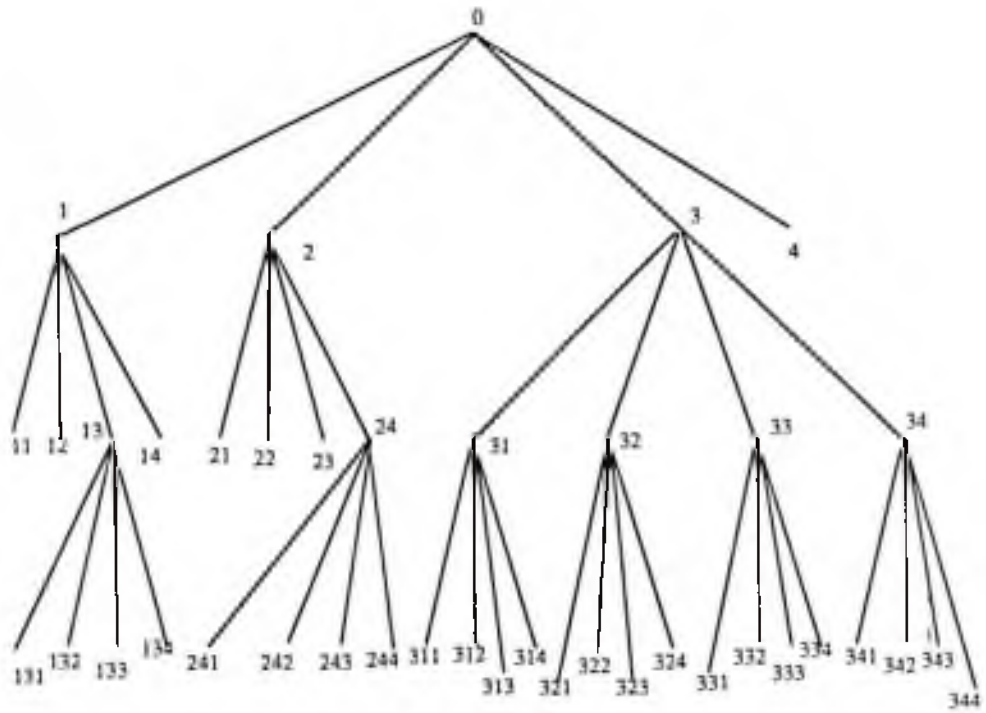

Fig. 18. Arbol de Segmentación correspondiente a la Fig. 17.

1. La imagen original se divide en un número determinado de regiones, formando una segmentación inicial, a la cual se le construye conjuntamente su árbol de segmentación. El tamaño de las regiones iniciales puede ser arbitrario, o puede ser determinado por el tamaño de los objetos buscados.

2. Apoyados en el árbol de segmentación se llevan a cabo todas las uniones de nodos que sean permitidas por el criterio de homogeneidad.

3. Las regiones que no satisfagan el criterio de homogeneidad son subdivididas hasta que las regiones resultantes lo cumplan, asignando a cada región un número que las identifica.

4. Después de las operaciones de Unión y División se lleva a cabo un agrupamiento (Grouping), operación que une dos regiones adyacentes que satisfacen el criterio de homogeneidad. Esta operación abandona el árbol de segmentación y construye la Matriz de adyacencia, que tiene las dimensiones de la imagen original. A cada elemento de la matriz se le asigna el número de la región a la cual pertenece, ver Fig. 19.

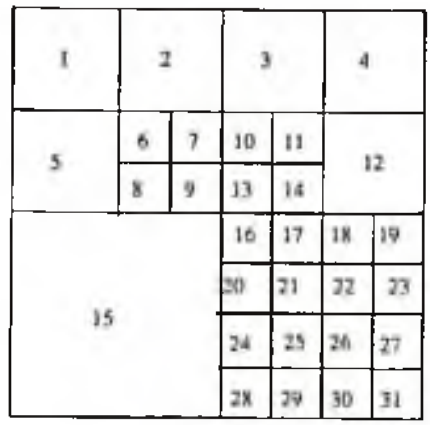

Fig. 19. Matriz de adyacencia de la Fig. 16.

La matrix de adyacencia nos sirve para que a cada región que se esta procesando se le pueda encontrar sus regiones adyacentes. La forma de encontrarlas es recorrer todo el borde de la región que se esta procesando y colectar en una lista los números de las regiones que se van encontrando. Posteriormente se analiza cada region que está en la lista para probar si esta region puede ser unida con la region-centro. Si es el caso, esta última dará su número a la región adyacente para que posteriormente, si vuelve a ser unida con otra region, puedan ser renombradas no sólo esta misma región, sino todas las regiones que contengan su mismo número. La región es marcada como procesada, para no volverla a considerar, y asi se procede con todas las regiones que no han sido procesadas. Este paso es necesario para unir, por ejemplo, las regiones 8 y 9 de la Fig. 19, ya que el primer paso del algoritmo, que también realiza unión de regiones, no perimitiría esta unión, puesto que permite solo la unión de 4 regiones que pertenezcan a un mismo cuadrante.

5. Finalmente las regiones muy pequeñas se unen con la región adyacente más cercana a ellas. El criterio que se usa para determinar si una región es pequeña es por el número de pixels que ésta ocupa, el cual no debe de sobrepasar una constante que se da como parámetro al algoritmo.

La ventaja del algoritmo de Split and Merge con respecto a los algoritmos de Union y Division considerados por separado, es que el primero utiliza

menos tiempo de procesado para realizar la segmentacion. Analicemos las causas del aumento de la velocidad de procesamiento en este algoritmo.

Consideremos el proceso de segmentación como la construcción de un Arbol de segmentación, ver Fig. 18 y 20.

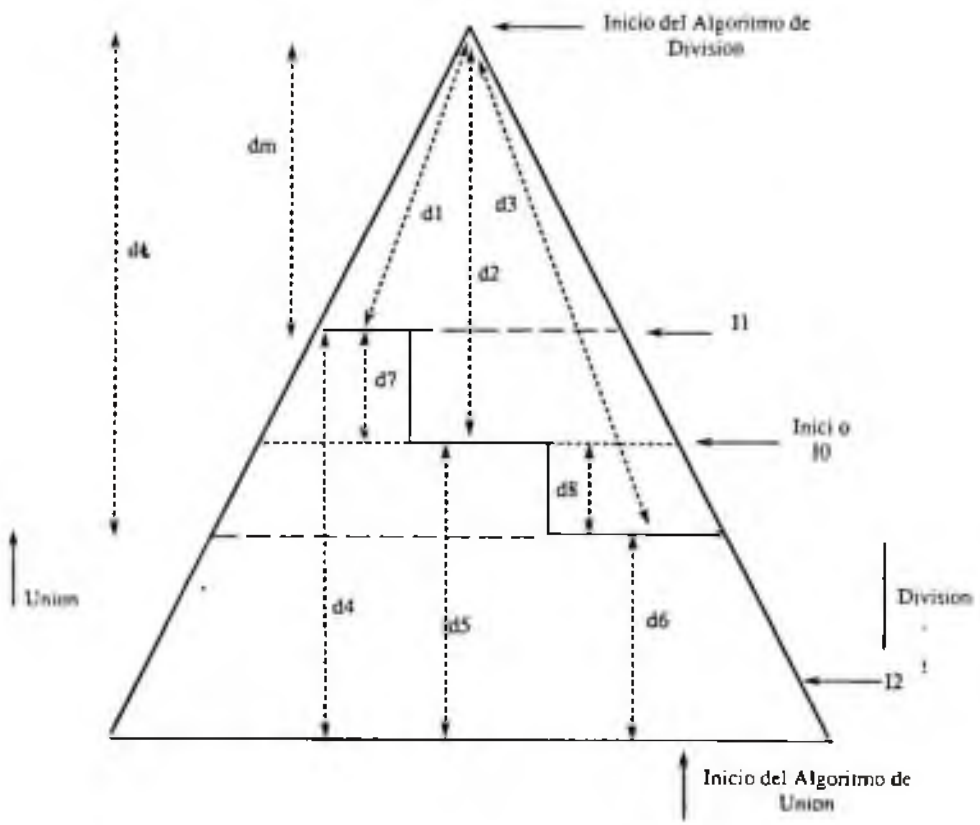

Fig. 20. Operaciones de Unión y División en el Arbol de segmentación.

Cuando el algoritmo de "Split" and "Merge" realiza la unión de algunas regiones, se eliminan dos hojas del árbol y un nodo intermedio se convierte en hoja. Cuando el algoritmo realiza una división se procede al contrario, se crean dos nuevas hojas. Este proceso se lleva a cabo a partir de cierta distancia de la raíz del árbol, la cual se busca que sea la mitad de su longitud total.

Si suponemos que las regiones de una imagen satisfacen el criterio de homogeneidad en los niveles I0, I1, I2 del árbol de segmentación (Fig. 20),

entonces este algoritmo recorre solamente las distancias d7 y d8 para poder segmentar la imagen.

Por el contrario el algoritmo de división empieza en la raíz del árbol, ya que este algoritmo se aplica a la imagen original completa. Posteriormente el algoritmo subdivide a la imagen en subregiones, construyendo por consiguiente el árbol de segmentación. Al termino del algoritmo se habrán recorrido las distancias d1, d2 y d3, ya que hay un conjunto de regiones que necesitan pocas subdivisiones y para éstas el algoritmo recorre la distancia d1. Existe otro conjunto de regiones, el cual necesita más subdivisiones, para el cual el algoritmo recorre la distancia d2, y otro más que casi llegan a ser regiones con uno o dos pixels para los cuales el algoritmo recorre la distancia d3.

De esta discusión se concluye que el algoritmo de División y Unión (Split and Merge) es más eficiente que el de División y el de Unión aplicados separadamente(Fig. 20). Hay que hacer notar que las ventajas de este algoritmo dependen de la altura del árbol de segmentación en la que este empiece, ya que si el punto de inicio es el I1 o el I2 (Fig. 18), la eficiencia disminuye, puesto que se convierte solamente en un algoritmo de Unión o de División.

Para obtener los bordes de los objetos presentes, después de haber aplicado el algoritmo de División y Unión a la Imagen No. 1, se aplica el operador Roberts a la imagen segmentada. Los resultados se muestran en la Imagen 8.

2.3 Detector óptimo de bordes

El tercer y último método que se utilizó para detectar los bordes de los objetos que forman una imagen original es el método de Deriche [Der 87]. El método está basado principalmente en el llamado "Detector óptimo de Bordes" desarrollado por Canny [Can 86]. A continuacion damos su formulación matemática.

Formulación del método de Canny en 1 dimension. Esta sección está basada principalmente en las referencias [Can 86] y [Der 87], donde se pueden encontrar más detalles acerca del detector.

Para facilitar el análisis consideramos primero bordes en una dimension. Supongamos que tenemos un borde inmerso en ruido (Fig. 21a), el cual tiene una distribución gausiana. El problema aqui es encontrar la posición donde se encierra el borde. Para localizarlo se convuelve el borde con un filtro cuya respuesta puntual puede ser representada por la Fig. 21b o 21d, obteniéndose como respuesta los bordes mostrados en las Figs. 21c y 21e, respectivamente.

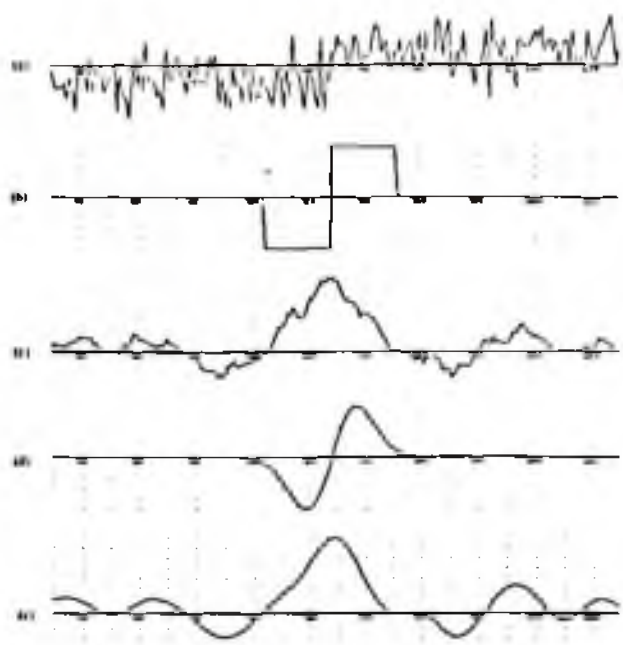

Fig. 21. (Tomada de [Can 86])
a) Borde del tipo "step" con ruido
b) Un ejemplo de un operador para detectar bordes.
c) Resultado de aplicar *b* a *a*.
d) Primera derivada del operador Gaussiano.
e) *d* aplicado a *a*.

Si observamos las Fig. 21c y 21e concluimos que el centro del borde se encuentra en la posición donde esta el valor máximo de la función.

El problema del diseño de un filtro óptimo se reduce entonces a encontrar un operador que optimice la función de salida respecto a ciertos criterios de eficiencia que se establecen dependiendo del tipo de borde bajo consideración, ya que por ejemplo el Filtro dado en la Fig. 21d produce mejores resultados que el de la Fig. 21b, porque la función respuesta de este último no es una señal continua o suave sino que tiene picos, esto es, aún tiene ruido.

En nuestro caso los criterios de eficiencia aqui utilizados son:

1. Buena Deteccion, i.e. la probabilidad de detectar bordes reales debe de ser grande y la probabilidad de detectar bordes falsos debe de ser pequeña. Este criterio corresponde a maximizar la razón señal-ruido (signal-to-noise -ratio) del filtro que se diseñe.

2. Buena localización. Los pixels marcados como bordes por el operador deben de estar situados lo más cerca posible al centro del borde verdadero.

3. Partiendo de un borde aislado el operador debe de producir una respuesta única. Nótese que esta condición está implícitamente expresada en el 1er. criterio puesto que cuando existen 2 respuestas a un mismo borde, una de ellas se considera como falsa. Sin embargo la fórmula matemática del primer criterio no incluye este tercer criterio, el cual se expresa con una fórmula adicional.

Estos criterios se formulan matemáticamente como sigue:

Sea $f(x)$ la función que representa al borde inicial. Este borde se presupone que es del tipo "step" y esta representado explícitamente por:

$$f(x) = A \cdot u(x) \tag{2.13}$$

donde $u(x)$ es la derivada de la función impulso y A es la amplitud del step. Supondremos también que el borde está centrado en $x = 0$ y que la detección de un borde se realiza al convolver la función $f(x)$ con una función especial antisimetrica $g(x)$:

$$s(x) = \int_{-W}^{+W} f(-x)g(x)dx \tag{2.14}$$

suponiendo que el filtro tiene una respuesta a la función impulso delimitada por $[-W, W]$. La rms de la respuesta del filtro aplicada al ruido $n(x)$ seria:

$$r(x) = n_o[\int_{-W}^{+W} g^2(x)dx]^{(1/2)} \tag{2.15}$$

donde n_o^2 es la media cuadrada de la amplitud del ruido por unidad de área. Para el criterio de localización necesitamos una medida que se incremente de acuerdo al mejoramiento de la localización. Con este propósito usaremos el inverso de la rms de la distancia entre el borde marcado y el centro del borde verdadero. Como se decidió situar un borde a la altura del valor máximo de la función que resulta al aplicar el operador $g(x)$, la primera derivada de la respuesta será cero en esos puntos. Nótese también que como los bordes están centrados en x=0, en la ausencia de ruido debe de haber un máximo local en la respuesta a la altura de $x = 0$. Sea $r(x)$ la respuesta al ruido solamente y sea $s(x)$ la respuesta al borde y supongamos que existe un máximo local en $x = x_0$ en la respuesta total del filtro. De aqui tenemos que

$$r'(x_0) + s'(x_0) = 0 \tag{2.16}$$

La expansión de Taylor de $s'(x_0)$ a partir del origen da como resultado:

$$s'(x_0) = s'(0) + s''(0)x_0 + O(x_0^2) \tag{2.17}$$

De acuerdo a nuestras suposiciones $s'(0) = 0$, i.e. la respuesta del filtro a la ausencia de ruido tienen un máximo local en el origen, por consecuencia el primer término en la expansión puede ser ignorado. El desplazamiento x_0 del máximo actual se presupone que es pequeño, de tal forma que se pueden ignorar los términos cuadráticos y de mayor orden. De 2.16 y 2.17 tenemos:

$$s''(0)x_0 \approx -r'(x_0) \tag{2.18}$$

Ademas como $r'(x_0)$ es una cantidad aleatoria con distribución gausiana, cuya varianza es el valor medio cuadratico de $r'(x_0)$ tenemos que:

$$E[r'(x_0)^2] = n_0^2 \int_{-W}^{+W} g'^2(x)dx \tag{2.19}$$

donde $E[y]$ es el valor esperado de y. Combinando el resultado anterior con 2.18 y substituyendolo por $s''(0)$ tenemos:

$$E[x_0^2] \approx \frac{n_0^2 \int_{-W}^{+W} g'^2(x)dx}{[\int_{-W}^{+W} f'(-x)g'(x)dx]^2} = \delta x_0^2 \tag{2.20}$$

donde δx_0 es una aproximación a la desviación estándard de x_0.

El primer criterio de eficiencia antes mencionado se puede expresar entonces como la maximización de la razón de la señal con respecto al ruido (SNR) la cual se define como:

$$SNR(x) = \frac{A}{n_0} \cdot \frac{|\int_{-W}^{+W} f(-x)g(x)dx|}{\left\{\int_{-W}^{+W} g^2(x)dx\right\}^{1/2}} = \frac{A}{n_0}\Sigma(g) \tag{2.21}$$

La localización la definimos como el inverso de δx_0:

$$Localizacion = \frac{A}{n_0} \cdot \frac{\left| \int_{-W}^{+W} f'(x)g'(x)dx \right|}{\left\{ \int_{-W}^{+W} g'^2(x)dx \right\}^{1/2}} = \frac{A}{n_0}\lambda(g) \tag{2.22}$$

Para el tercer criterio el detector no debe producir respuestas múltiples cuando se trata de un borde aislado. Existe entonces la necesidad de limitar el número de valores máximos locales que puedan existir en la función respuesta de tal forma que la probabilidad de declarar como resultado la existencia de dos o más bordes sea pequeña. Para lograrlo la distancia entre dos picos denotada por x_{max}, se hace igual al doble de la distancia entre los "zero-crossings" de la respuesta de un filtro Gaussiano a una función g, la cual es a su vez una función k de la amplitud del operador:

$$x_{max} = 2\pi \cdot \left[\frac{\int_{-\infty}^{+\infty} g'^2(x)dx}{\int_{-\infty}^{+\infty} g''^2(x)dx} \right]^{1/2} = kW \tag{2.23}$$

Después de haber desarrollado los criterios para la detección, localización y limitación del No. de máximos locales, estos tres criterios se combinan para que juntos definan al operador óptimo. La forma de combinarlos es maximizar el producto $\Sigma(x) \cdot \lambda(x)$ bajo la condición del tercer criterio.

Expresando el criterio resultante como una funcional compuesta se encontró que esta funcional conduce a una solución g(x) que cumple la siguiente ecuación diferencial:

$$2 \cdot g(x) - 2 \cdot \lambda_1 \cdot g''(x) + 2 \cdot \lambda_2 \cdot g'''(x) + \lambda_3 = 0 \tag{2.24}$$

La solución general de la ecuación diferencial en el rango $[0, W]$ es de la forma:

$$\begin{aligned} g(x) = {} & a_1 \cdot e^{\alpha \cdot x} sen\omega \cdot x + a_2 \cdot e^{\alpha \cdot x} \cos\omega \cdot x + \\ & a_3 \cdot e^{-\alpha \cdot x} sen\omega \cdot x + a_4 \cdot e^{-\alpha \cdot x} \cos\omega \cdot x + c \end{aligned} \tag{2.25}$$

sujeta a las siguientes condiciones de frontera:

$$g(0) = 0, \qquad g(-W) = 0, \qquad g'(0) = S, \qquad g'(W) = 0 \tag{2.26}$$

donde S es una constante desconocida igual a la pendiente de la función $g(x)$ en el origen y c una constante positiva. Debido a que $g(x)$ es impar, la solución anterior se puede extender al rango $[-W, +W]$ usando la igualdad:

$$g(-x) = -g(x) \tag{2.27}$$

Ahora bien, por medio de las cuatro condiciones de frontera calculamos las cantidades a_1, a_2, a_3, a_4 de la ecuación número 2.25.

Primero escribamos el sistema de ecuaciones lineales:

$$\begin{aligned}
a_2 + a_4 + c &= 0 && (2.28)\\
ae^{\alpha} sen\omega + a_2 e^{\alpha}\cos\omega + a_3 e^{-\alpha} sen\omega + a_4 e^{-\alpha}\cos\omega + c &= 0 && (2.29)\\
a_1\omega + a_2\alpha + a_3\omega \cdot a_4\alpha &= S && (2.30)\\
a_1 e^{\alpha}(\alpha sen\omega + \omega\cos\omega) + a_2 e^{\alpha}(\alpha\cos\omega - wsen\omega) & \\
+a_3 e^{-\alpha}(-\alpha sen\omega + \omega\cos\omega) + a_4 e^{-\alpha}(-\alpha\cos\omega - \omega sen\omega) &= 0 && (2.31)
\end{aligned}$$

y las soluciones a estas ecuaciones son:

$$\begin{aligned}
a_1 = {} & c(\alpha(\beta-\alpha)sen2\omega - \alpha\omega\cos 2\omega + \\
& (-2\omega^2 senh\alpha + 2\alpha^2 e^{-\alpha})sen\omega + 2\alpha\omega senh\alpha\cos\omega + \\
& \omega e^{-2\alpha}(\alpha+\beta) - \beta\omega)/4(\omega^2 senh^2\alpha - \alpha^2 sen^2\omega) && (2.32)
\end{aligned}$$

$$\begin{aligned}
a_2 = {} & c(\alpha(\beta-\alpha)\cos 2\omega + \alpha\omega sen2\omega - 2\alpha\omega\cosh\alpha \\
& sen\omega - 2\omega^2 senh\alpha\cos\omega + 2\omega^2 e^{-\alpha} senh\alpha \\
& +\alpha(\alpha-\beta))/4(\omega^2 senh^2\alpha - \alpha^2 sen^2\omega) && (2.33)
\end{aligned}$$

$$\begin{aligned}
a_3 = {} & c(-\alpha(\beta+\alpha)sen2\omega + \alpha\omega\cos 2\omega + \\
& (2\omega^2 senh\alpha + 2\alpha^2 e^{\alpha})sen\omega + 2\alpha\omega senh\alpha\cos\omega \\
& +\omega e^{2\alpha}(\beta-\alpha) - \beta\omega)/4(\omega^2 senh^2\alpha - \alpha^2 sen^2\omega) && (2.34)
\end{aligned}$$

$$\begin{aligned}
a_4 = {} & c(-\alpha(\beta+\alpha)\cos 2\omega - \alpha\omega sen2\omega + \\
& 2\alpha\omega\cosh\alpha \cdot sen\omega + 2\omega^2 senh\alpha\cos\omega \\
& -2\omega^2 e^{\alpha} senh\alpha + \alpha(\alpha-\beta))/4(\omega^2 senh^2\alpha - \alpha^2 sen^2\omega) && (2.35)
\end{aligned}$$

donde $\beta = s/c$.

La función g está ahora parametrizada en términos de las constantes α, ω, β y c. Falta encontrar los valores de estos parámetros, los cuales maximizan el producto $\Sigma(g) \cdot \lambda(g)$.

Utilizando optimización numérica se encontraron diferentes valores para los parámetros antes mencionados, los cuales se muestran en la siguiente tabla:

Diferentes valores para los parámetros: $x_{max}, \Sigma\lambda, \alpha, \omega, \beta$

n	x_{max}	$\Sigma\lambda$	α	ω	β
1	0.15	4.21	24.5955	0.12250	63.97
2	0.3	2.87	12.4712	0.38284	31.26
3	0.5	2.13	7.858	2.62856	18.28
4	0.8	1.57	5.065	2.56770	11.06
5	1.0	1.33	3.4558	0.07161	4.806
6	1.2	1.12	2.0522	1.56939	2.915
7	1.4	0.75	0.00297	3.50350	7.477

De estos valores se escogió el conjunto No. 6 ya que presenta un buen balance entre los tres criterios mencionados anteriormente, lo cual se refleja en el valor de los parámetros x_{max} y $\Sigma\lambda$, los cuales son casi iguales. La gráfica del operador definido por este conjunto de valores se puede ver en la Fig. 22. En esta misma figura se muestra la gráfica de la función obtenida al derivar una función Gaussiana. Nótese la semejanza entre ellas. Debido a que ambas funciones gráficamente son parecidas se escogió la derivada de la función Gaussiana como un equivalente al operador antes encontrado. Una segunda razón de esta selección es que ya existen métodos muy eficientes para calcular la convolución de una función con la derivada de una Gaussiana, mientras que para calcular la convolución del operador óptimo antes encontrado no existen.

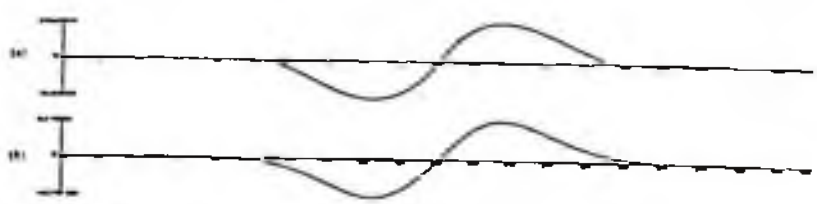

Fig. 22. Filtro No.6 y derivada de una Función Gaussiana.

La expresión de la función Gaussiana es:

$$gs(x) = exp[-\frac{x^2}{2\sigma^2}] \tag{2.36}$$

y su derivada es:

$$G(x) = -\frac{x}{\sigma^2} exp(-\frac{x^2}{2\sigma^2}) \tag{2.37}$$

La eficiencia del operador con respecto al operador óptimo, se obtiene del producto $\Sigma(g)\lambda(g)$ que es igual a 0.92. Esto significa que el nuevo filtro es en un 20% menos eficiente que el operador óptimo, no obstante Canny lo utiliza para obtener los bordes de algunas imagenes, obteniendo buenos resultados [Can 86]. El único problema con este nuevo operador es su cálculo, ya que implica la utilización de la operación de convolución, la cual necesita mucho tiempo de procesamiento.

Tratando de resolver este problema, Deriché [Der 87] propuso otro filtro, que se calcula más fácilmente y tiene una eficiencia mayor que el basado en la derivada de la Gaussiana.

Deriché hace su análisis partiendo de la suposición de que el operador óptimo buscado es un filtro IIR, el cual cuando procesa una función impulso produce una respuesta infinita, a diferencia de Canny, que supone un filtro con respuesta finita (FIR filter). Deriché llega a la misma ecuación diferencial que Canny y por lo tanto a la misma solución general representada por la ecuación No. (2.25). Sin embargo las condiciones de frontera cambian, de acuerdo a la siguiente expresión, donde d es equivalente a la función g usada anteriormente:

$$d(0) = 0, \qquad d(+\infty) = 0, \qquad d'(0) = S, \qquad d'(+\infty) = 0 \tag{2.38}$$

aplicando estas condiciones a la ecuación número (2.25) se obtiene la siguiente solución:

$$d(x) = (s/w) \cdot e^{-\alpha \cdot x} sen\omega \cdot x \tag{2.39}$$

siendo $\alpha y \omega$ reales positivas.

Por otro lado, evaluando las integrales 2.21 y 2.22 que definen los criterios de eficiencia se obtiene:

$$\lambda = \sqrt[3]{2.a} \tag{2.40}$$

$$\Sigma \cdot \lambda = \frac{2 \cdot a}{\sqrt[3]{a^2 + \omega^2}} \tag{2.41}$$

$$k = \frac{\sqrt[3]{a^2 + \omega^2}}{5 \cdot a^2 + \omega^2} \tag{2.42}$$

y considerando $\alpha = m \cdot \omega$ tenemos:

$$\text{si:} \qquad m >> 1, \qquad \lambda = (2 \quad \alpha)^{1/2}, \qquad \Sigma = \sqrt[2]{2/\alpha}$$

$$\Sigma \cdot \lambda \approx 2, \qquad k \approx 0.44 \tag{2.43}$$

El producto $\Sigma \cdot \lambda$ tiene su valor máximo cuando ω tiene el valor 0 y puesto que $sen(\omega \cdot x) \approx x$ para ω pequeña, podemos reescribir la ecuación (2.39) como:

$$d(x) = S \cdot xe^{-\alpha} \cdot |x| \tag{2.44}$$

Este operador tiene mayor eficiencia que el operador de Canny basado en la derivada de una Gaussiana, es muy sencillo y presenta solo un párametro, α, que se ajusta de acuerdo a la razón señal-ruido que se desee. Si se decrementa α, se disminuirá la exactitud con la que se localiza un borde, pero se aumenta la razón señal ruido.

En los siguientes párrafos se presenta una implementación eficiente del filtro, la cual no utiliza la operación de convolución para su cálculo, sino la técnica de filtramiento recursivo.

Sea d(n) un muestreo de la función d(x) y sea D(Z) su transformada Z, entonces tenemos que:

$$D(Z) = \sum_{n} d(n) \cdot Z^{-n} \qquad \text{para } n = -\infty, ..., +\infty \tag{2.45}$$

Dividamos la función $d(n)$ en dos mitades: $d_-(n)$ y $d_+(n)$ tal que:

$$d_-(n) = \begin{cases} 0 & n \geq 0 \\ S \cdot n \cdot e^{\alpha \cdot n} & n < 0 \end{cases} \tag{2.46}$$

$$d_+(n) = \begin{cases} S \cdot n \cdot e^{\alpha \cdot n} & n \geq 0 \\ 0 & n < 0 \end{cases} \tag{2.47}$$

dando como resultado:

$$d(n) = d_-(n) + d_+(n) \qquad \text{para } n = -\infty, ..., +\infty \tag{2.48}$$

Usando la transformada Z y simplificando obtenemos:

$$G(Z) = G_{-}(Z) + G_{+}(Z^{-1}) \tag{2.49}$$

$$G_{+}(Z^{-1}) = \frac{a \cdot Z^{-1}}{1 + b_1 Z^{-1} + b_2 Z^{-2}} \tag{2.50}$$

$$G_{-}(Z) = \frac{-a \cdot Z}{1 + b_1 Z + b_2 Z^2} \tag{2.51}$$

donde:

$$a = S \cdot e^{-\alpha} \qquad b_1 = 2 \cdot e^{-\alpha} \qquad b_2 = e^{-2\alpha} \tag{2.52}$$

Las dos transformadas Z corresponden a dos funciones de transferencia de dos sistemas racionales de filtros estables de segundo orden, los cuales corren de izquierda a derecha (G_{+}) y de derecha a izquierda (G_{-}). En particular si aplicamos al sistema con función respuesta $g(n)$ la sucesión de valores $x(m)$ que contiene M elementos, obtenemos la sucesión $y(m)$, la cual puede ser obtenida recursivamente de acuerdo a la siguiente fórmula:

$$y^{+}(m) = x(m-1) - b_1 \cdot y^{+}(m-1) - b_2 y^{+}(m-2) \qquad m = 1, ..., M \tag{2.53}$$

$$y^{-}(m) = x(m+1) - b_1 \cdot y^{-}(m+1) - b_2 \cdot y^{-}(m+2) \qquad m = M, ..., 1 \tag{2.54}$$

$$y(m) = a \cdot (y^{+}(m) - y^{-}(m)) \qquad m = 1, ..., M \tag{2.55}$$

con a,b_1 y b_2 dados por la ecuacion (2.52). La constante S está definida como:

$$S = -(1 - e^{-\alpha})^2 / (e^{-\alpha}) \tag{2.56}$$

La importancia en usar estas ecuaciones para el cálculo del filtro $d(x)$ es que no se utiliza la operación de convolución, sino solamente algunas multiplicaciones y adiciones.

De la misma forma se pueden emplear fórmulas equivalentes para obtener $h(x)$ que es la integral de la función $d(x)$ y que se usará para la aplicación del filtro en dos dimensiones:

$$h(x) = \int d(x)dx = k \cdot (\alpha \cdot |x| + 1) \cdot e^{-\alpha|x|} \tag{2.57}$$

Calculando la transformada Z de $h(n)$ tenemos:

$$H(Z) = H_+(Z^{-1}) + H_-(Z) \tag{2.58}$$

donde:

$$H_+(Z^{-1}) = \frac{a_0 + a_1 Z^{-1}}{1 + b_1 \cdot Z^{-1} + b_2 Z^{-2}} \tag{2.59}$$

$$H_-(Z) = \frac{a_2 \cdot Z + a_3 Z^2}{1 + b_1 Z + b_2 Z^2} \tag{2.60}$$

con:

$$\begin{array}{llll} a_0 = k & a_1 = k(\alpha - 1)e^{-\alpha} & a_2 = a_1 \cdot k b_1 & \\ a_3 = -k \cdot b_2 & b_1 = -2e^{-\alpha} & b_2 = e^{-2} \cdot \alpha & \end{array} \tag{2.61}$$

Si se quiere aplicar el filtro con función respuesta $\{h(n)\}$ a la sucesión de valor $\{x(m)\}$, se obtiene la sucesión $\{y(m)\}$, todas con M elementos, de acuerdo a las siguientes fórmulas recursivas:

$$\begin{aligned} y^+(m) &= a_0 x(m) + a_1 x(m-1) - b_1 y^+(m-1) - b_2 y^+(m-2) \\ &\quad \text{para } m = 1, ..., M \qquad (2.62) \\ y^-(m) &= a_2 x(m+1) + a_3 x(m+2) - b_1 y^-(m+1) - b_2 y^-(m+2) \\ &\quad \text{para } m = M, ..., 1 \qquad (2.63) \\ y(m) &= y^+(m) + y^-(m) \qquad (2.64) \\ &\quad \text{con } m = 1, ...M \end{aligned}$$

donde:

$$k = (1 - e^{-\alpha})^2/(1 + 2\alpha e^{-\alpha} - e^{-2\alpha}) \tag{2.65}$$

En este filtro así como en el anterior, se tiene que α es el único parámetro que determina el tamaño del filtro, el cual se varía de acuerdo a la eficiencia en la detección de los contornos de los objetos que se desee.

Formulación para dos dimensiones.

Para la detección de contornos en dos dimensiones se utilizan los operadores $h(n)$ y $d(n)$ conjuntamente. La forma de hacerlo es considerar que la pendiente de una superficie en cualquier dirección, puede ser determinada exactamente a partir de dos direcciones perpendiculares cualesquiera. Para calcular entonces la dirección y el tamaño de un contorno, tenemos entonces que calcular la pendiente en la dirección "x" y en la perpendicular "y" para combinarlas posteriormente. Para calcular la dirección del contorno en la dirección x, donde la dirección x la definiremos como la dirección en la que corren las columnas de la matriz de la imagen original, se hace primero un filtrado recursivo con el operador $d(n)$ en la dirección x. Posteriormente un filtrado recursivo en la dirección y con el operador $\{h(n)\}$, obteniéndose una imagen que denominaremos $\{C_x(m,n)\}$. Para obtener la dirección del contorno en la dirección y se procede en una forma similar, empezando con el operador $d(n)$ y la dirección y, continuando con el operador $h(n)$ en la dirección x, obteniendose también otra imagen que denominaremos ahora como $\{C_y(m,n)\}$.

Después, para obtener la dirección del contorno $\{C(m,n)\}$ y la amplitud de éste $\{A(m,n)\}$ aplicamos las siguientes fórmulas:

$$A(m,n) = \sqrt[2]{C_x(m,n)^2 + C_y(m,n)^2} \tag{2.66}$$

$$C(m,n) = \text{arc tg}\left[\frac{C_y(m,n)}{C_x(m,n)}\right] \tag{2.67}$$

A la imagen $\{A(m,n)\}$ se le suprimen posteriormente todos los valores que no sean máximos locales, utilizando para tal propósito la direción del contorno. Esta supresión se hace calculando para cada pixel de la imagen un vector perpendicular a la dirección del contorno que pasa por ese pixel con una longitud predeterminada. Este pixel es entonces eliminado si es que no corresponde al máximo dentro de este vector.

Posteriormente la imagen obtenida es binarizada con hysteresis. Esta técnica es muy usada en electrónica y consiste en la selección de dos umbrales u_1 y u_2, tal que $u_1 > u_2$. A cada pixel de la imagen mayor que u_1 se le asigna el valor 1, así como a todos los pixels conectador con la condición que sean mayores que u_2, de otra forma se les asignará un cero. La idea de esta

técnica es la de introducir un cierto "contexto" en los pixels que forman un contorno. Si un pixel pertenece a un contorno todos sus vecinos conectados con él también pertenecendebido a que son igualmente máximos locales, solo que debido a imperfecciones en la toma y procesamiento de la imagen poseen un valor más pequeño.

Para las imagenes aquí tratadas no se utilizo la técnica de binarización con hysteresis, sino solo una binarización normal porque los resultados de ambas técnicas fueron muy similares. Los resultados de este procedimiento se muestran en la imagen número 10.

2.4 Algorítmo seguidor de contornos

Después de haber localizado los centros de los puntos de soldadura, el paso siguiente es la evaluación de su calidad. Existen varias técnicas para realizar la evaluación, algunas se basan en el volumen que ocupa la soldadura en ese punto y otras en su forma. Algunas otras mas buscan características especiales en los puntos de soldadura que hacen clasificarlos como puntos no aceptables. Una de estas características es la dimensión y forma de su contorno, que fué la que aqui se empleo. En esta sección se explica un algoritmo para encontrar dicha forma.

El algoritmo que se usó es un algoritmo seguidor de contornos que considera la búsqueda de un contorno como la búsqueda de un camino óptimo entre dos nodos pertenecientes a una gráfica G construida a partir de los pixels de la imagen original. Cada arco de la gráfica G que une dos nodos tiene asignado un costo.

Para el problema de la búsqueda del camino óptimo en gráficas se han propuesto muchas soluciones, ya que este problema es muy común en muchas areas, en especial en el área de Inteligencia Artificial y en particular en la optimización combinatoria. En nuestro caso se usa el algoritmo A^* propuesto por Hart, Nilsson y Raphael [Nil 80], el cual utiliza la información heurística disponible acerca del problema que se trata de solucionar, para acelerar el proceso de búsqueda.

El algoritmo empieza con el nodo S y genera la gráfica G en cada iteración hasta que se llega al nodo-meta. En cada iteración se selecciona un nodo ya generado al cual se le buscan sus sucesores. Los sucesores se conectan con su nodo generador por medio de arcos. A estos arcos se les asigna un costo, el cual se calcula por la siguiente función:

$$\hat{f}(x_n, y_n) = \hat{g}(x_n, y_n) + \hat{h}(x_n, y_n) \tag{2.68}$$

A ésta se le llama función de evaluación y estima el costo de un camino óptimo desde el nodo inicial s hasta un nodo meta m, pasando por el nodo n.

La función $\hat{g}(x_n, y_n)$ es una función que estima el costo de un camino óptimo desde s hasta n y la función $\hat{h}(x_n, y_n)$ aproxima el costo de una ruta mínima desde n hasta m. Para nuestro problema en particular $\hat{g}(x_n, y_n)$ la tomaremos como la suma de tres cantidades como después se explica y $\hat{h}(x_n, y_n)$ se obtiene de la información heurística que se tiene del problema.

Se puede probar que si $\hat{h}(n)$ es menor que $\mathbf{h}(n)$, que es la función que calcula exactamente el costo de un camino óptimo de n hasta m, entonces el algoritmo A^* encontrará una ruta óptima desde s hasta m. En particular si no se dispone de información para poder definir $\hat{h}$ (lo que significaría que $\hat{h}(n) = 0$), entonces el algoritmo A^* lleva a cabo una búsqueda exhaustiva denominada "a lo ancho" (breadth-first).

La versión del algoritmo A^* usada es la siguiente:

Algoritmo A^*

1. Marcar al pixel inicial s como no procesado y hacer $\hat{g}(x_s, y_s) = 0$.
2. Si existe algun pixel marcado como no procesado continuar, de otra forma terminar el algoritmo con "error".
3. Escoger de los pixels no procesados, el que tenga el menor costo y marcarlo como procesado. A este pixel lo denotamos como pixel n.
4. Si n es un pixel final terminar el algoritmo. El contorno del objeto en cuestión se obtiene al recorrer los apuntadores que unen al pixel final con el pixel inicial, partiendo del pixel final.
5. Si el pixel n no es un pixel final, encontrar los sucesores x_{ni}, y_{ni} de él y asignarles un costo $c(s, ni) = \hat{f}(x_{ni}, y_{ni}) + c(s, n)$. Unir con arcos a los sucesores de n con n, asignando a cada arco su costo. Si no hay sucesores continuar con el paso 2.
6. Si algún punto de los sucesores de n ya ha sido procesado, descartarlo y continuar con el paso 2.

Pasemos ahora a explicar cómo se construye la gráfica que hay que recorrer considerando el problema de la localización del contorno de los puntos de soldadura.

Primeramente hay que enfatizar que antes de realizar esta etapa del procesamiento ya se cuenta con la localización de los centros de los puntos de

soldadura. Esta información se usa para definir el conjunto de pixels con los que se empieza la gráfica. Este conjunto se define como el grupo de pixels localizados en la parte superior del centro del punto de soldadura que se esté analizando, a la altura del radio aproximado y en una vecindad de 10 pixels (Ver Fig. 23).

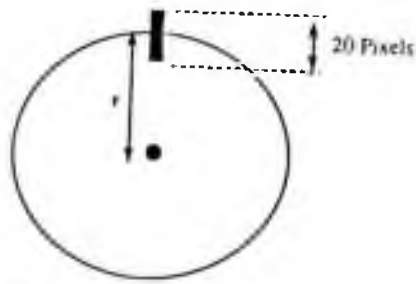

Fig. 23. Conjunto de puntos iniciales para la búsqueda del camino óptimo.

Posteriormente a cada pixel se le asigna su valor $\bar{g}(x,y)$ el cual esta definido por la suma de tres valores diferentes de acuerdo a tres criterios que a continuación se explican:

El primer criterio asigna un valor $\bar{g}_1(x,y)$ si es que el pixel en cuestión es un pixel que está sobre el borde del punto de soldadura. Estos bordes fueron encontrados por el procedimiento basado en el operador Roberts de la sección anterior.

El segundo criterio asigna un valor $\bar{g}_2(x,y)$ proporcional al monto del gradiente en este pixel.

El tercer criterio asigna un valor $\bar{g}_3(x,y)$ que es proporcional a la distancia entre el pixel (x,y) y un círculo de referencia cuyo centro coincide con el centro del punto de soldadura ya encontrado anteriormente.

Todos estos valores están escalados, de tal forma que el valor máximo que pueden tener es de 500 y el mínimo es de 0. Además cada uno de los valores $\bar{g}_1(x,y)$, $\bar{g}_2(x,y)$, $\bar{g}_3(x,y)$ está multiplicado por una constante k_1, k_2, k_3 respectivamente, las cuales ayudan a controlar el proceso de búsqueda, asignando prioridad a un criterio o a otro. Estas constantes están en el rango $[0,1]$.

De esta forma $\bar{g}(x,y)$ queda definida como sigue:

$$\bar{g}(x,y) = k_1\bar{g}_1(x,y) + k_2\bar{g}_2(x,y) + k_3\bar{g}_3(x,y) \tag{2.69}$$

Las razones por las cuales se escogieron los tres criterios anteriormente descritos son las siguientes:

1. Primeramente se necesita encontrar el contorno del punto de soldadura siguiendo los pixels que se encuentran sobre el borde del punto.

2. Posteriormente se necesita hacer una clasificación más fina de estos pixels, ya que considera más importante a los pixeles que tienen un gradiente más grande que los que lo tienen pequeño. Esta información no se tiene a disposición en los pixels del primer paso.

3. Finalmente hay que centrar la búsqueda ya que asi se obliga al algoritmo a encontrar un contorno redondo. Con este criterio se evita que el algoritmo siga contornos que no corresponden a objetos redondos, por ejemplo las pistas metálicas de unión.

4. Por último tenemos que $\hat{h}(x_n, y_n)$ sólo nos define la forma en la que se escogen los sucesores del pixel (x_n, y_n) y no contribuye explícitamente al valor de $\hat{f}(x_n, y_n)$, pero ayuda a dirigir la búsqueda evitando la expansión de todos los sucesores del pixel (x_n, y_n). Esta función solo toma los sucesores que se producen al seguir la dirección que tiene el gradiente en el pixel (x_n, y_n), ver figura 24.

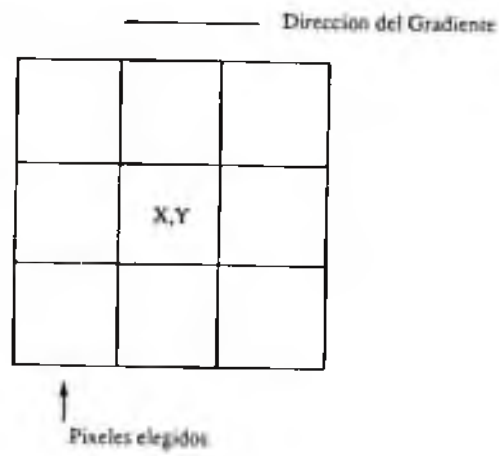

Fig. 24. Sucesores del pixel (x, y) considerando la dirección del gradiente en este pixel.

La búsqueda termina cuando el algoritmo encuentra algún pixel que está en el conjunto de pixels finales, esto es el conjunto de pixels con los que se inició la búsqueda.

Como el costo $c(s, n)$ se va acumulando a medida que la búsqueda va avanzando, llega el momento que los pixels del inicio llegan a tener un costo menor que los pixels que van a la cabeza del camino correcto, haciendo que el

algoritmo empiece nuevamente a analizar estos puntos del inicio, aun cuando éstos no están sobre un camino que pudiera ser óptimo posteriormente. Para evitar esto solo se toman en consideración los últimos 15 pixels analizados para escoger el pixel con el mínimo costo en el paso número tres del algoritmo A^*.

Por último el camino más apropiado se obtiene al recorrer los apuntadores que se fueron estableciendo en la búsqueda de camino óptimo, partiendo del último pixel analizado.

Los resultados de esta técnica se pueden ver en la imagen 12.

Capítulo 3

DISCUSION Y CONCLUSIONES

3.1 Discusión

En las secciones anteriores se ha analizado la Transformada de Hough y tres diferentes métodos para encontrar el contorno de objetos en imagenes digitalizadas. Estos métodos fueron probados con 20 diferentes imagenes de 512×512 pixels, tomadas de diferentes circuitos impresos. Una de estas imagenes se muestra en la imagen número 1. Los bordes de los objetos de esta imagen obtenidos por los diferentes métodos se muestran en las imagenes 6, 8 y 10.

En la imagen 6 podemos observar que los contornos de los objetos son bastante delgados en comparación con los de la imagen 8. Esto se debe al proceso de adelgazamiento llevado a cabo, el cual reduce la anchura de los bordes, además de que elimina el ruido presente en la imagen. Podemos ver tambien en esta figura que existen puntos de soldadura que no tienen una forma redonda, aunque en la imagen original sí la presenten. Esto se debe al hecho de que algunos puntos de soldadura se encuentran muy cerca unos de otros y consecuentemente se sobreponen en el transcurso de su procesamiento.

En la imagen 8 se muestran los bordes obtenidos por el procedimiento de Split and Merge. En esta imagen podemos ver que los bordes de los objetos presentan la típica forma cuadrada producida por este algoritmo. En este caso se puede lograr una mejor aproximación con los bordes reales cambiando los parámetros de entrada del algoritmo, pero esto ocasiona también el aumento del tiempo de procesamiento de 3 a 10 minutos y el aumento por consiguiente de la memoria necesaria para correr este algoritmo. Una solución a este dilema es escoger un conjunto de parámetros que equilibre la precisión con la que

se obtienen los bordes y el tiempo de procesamiento necesario. A pesar de que el algoritmo produce bordes con muy poca curvatura éste se adapta bien para la localización de puntos de soldadura, ya que además de los bordes reales produce otros bordes secundarios alrededor de estos puntos, los que refuerzan la suposición de su existencia, aunque en algunos casos también evitan su detección. Por esta última razón este método se adapta más para la localización de puntos de soldadura con un radio mayor de 20 pixels.

Finalmente en la imagen 10 se muestran los bordes obtenidos por el detector de bordes óptimo. Este método también produce bordes secundarios que refuerzan la suposición de un punto de soldadura.

Considerando los resultados mostrados en las Figs. 25, 26 y 27 se puede hacer una comparación cuantitativa de los métodos empleados . Primeramente en Fig. 25 se graficó el porcentaje de puntos de soldadura localizados correctamente por cada algoritmo. De esta gráfica se puede ver que los tres métodos producen el mismo porcentaje de puntos de soldadura correctamente localizados. En la Fig. 26 se graficó el porcentaje del número de puntos de soldadura localizados incorrectamente. Este porcentaje se calculó sumando el número de puntos de soldadura que existen pero que no fueron localizados más el número de puntos de soldadura que fueron localizados pero que no existen. Debe de hacerse notar que para cada algoritmo existe un conjunto de parámetros con el cual se encuentran todos los puntos de soldadura existentes en la imagen que se está analizando, pero el número de puntos de soldadura erróneamente localizados también se incrementa.

Finalmente en la Fig. 27 se graficó el tiempo de procesamiento promedio necesario para procesar una imagen. De esta gráfica se puede ver que el método No. 1 es el más rápido de todos, debido principalmente a la poca complejidad que presentan sus componentes.

3.2 Conclusiones

El método de División y Unión (Split and Merge) produce el mayor porcentaje de puntos de soldadura correctamente localizados. Este método usa un tiempo de procesado relativamente corto. El único problema con el algoritmo es que el tiempo de procesado y la cantidad de memoria utilizada dependen de la imagen que se esté procesando, ocasionando que algunas veces la memoria exceda el tamaño de la memoria disponible (en nuestro caso fueron 3 Mbytes). Debido a esto se puede concluir lo siguiente.

Si se tiene como objetivo la localización de los puntos de soldadura en el menor tiempo posible, entonces se recomienda el método basado en el ope-

rador Roberts. De otra forma se recomienda el método basado en el detector óptimo de bordes, ya que el método de Split and Merge utiliza bastante memoria, además de que el tiempo de procesado es dependiente de la imagen que se tenga.

El problema de la falsa detección de puntos de soldadura se puede subsanar haciendo uso de más información acerca de su posición. Esta información está presente en la imagen original y podría ser, por ejemplo, la corroboración de la existencia de un punto de soldadura por medio de las características de la región a la cual pertenece el centro del punto de soldadura. Estas características podrían ser por ejemplo el valor promedio de los pixels pertenecientes a la región o las dimensiones de ésta.

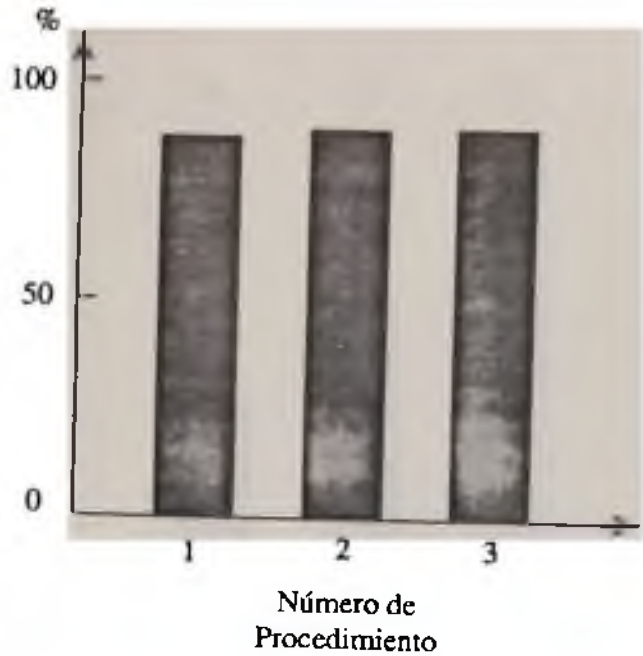

Fig. 25. Porcentaje de Puntos de soldadura correctamente localizados utilizando los bordes producidos por cada uno de los 3 métodos: Método basado en el operador Roberts, algoritmo de Split and Merge y Detector Optimo de Bordes.

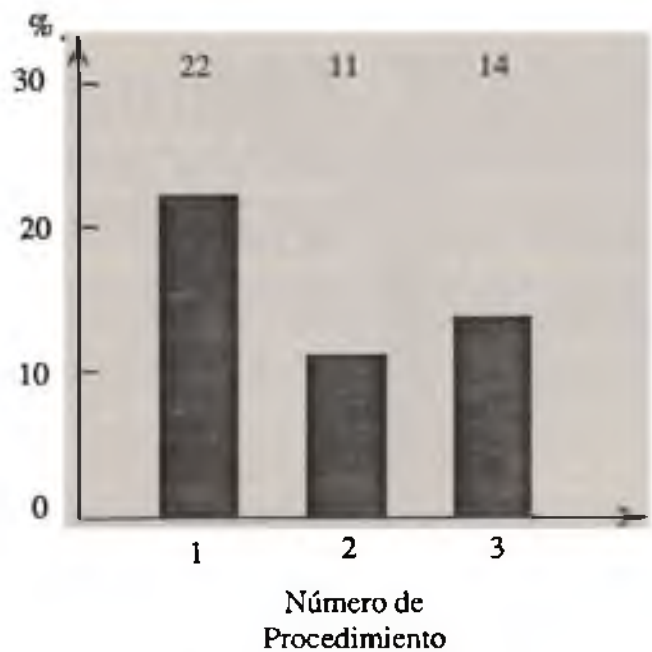

Fig. 26. Porcentaje de puntos de soldadura incorrectamente localizados.

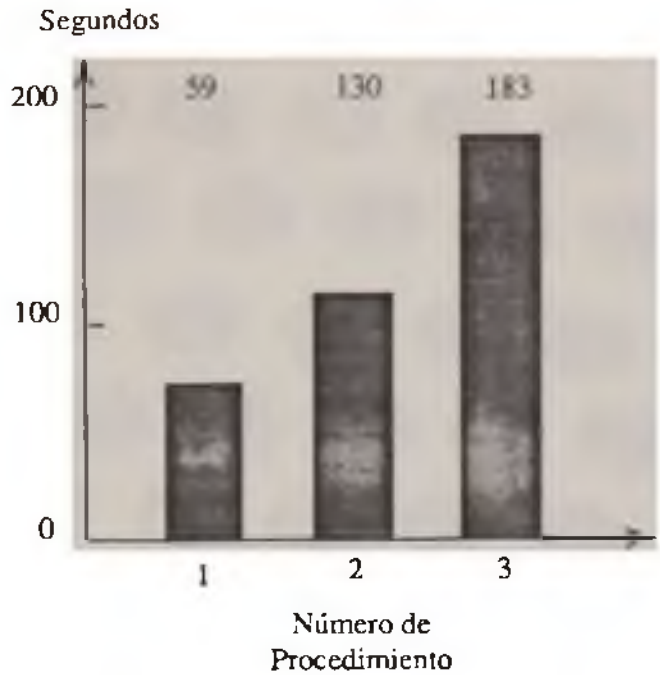

Fig. 27. Promedio del tiempo que cada uno de los 3 métodos necesita para procesar una imagen.

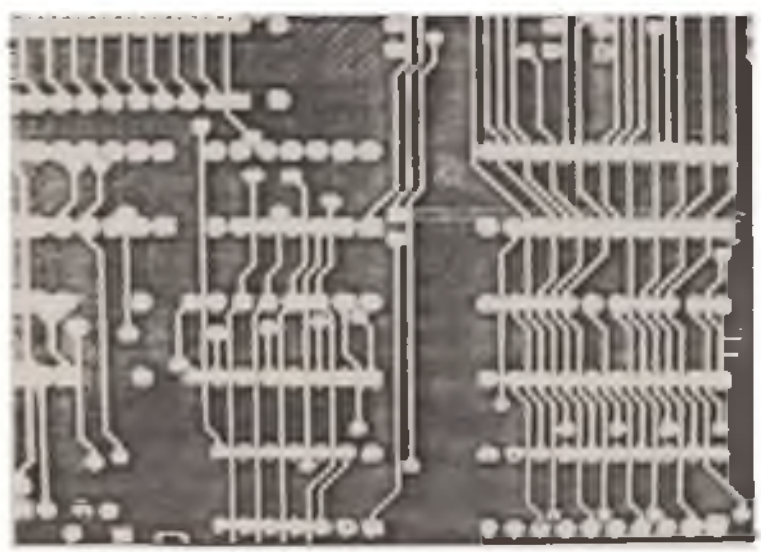

Imagen 1. Imagen original de una sección de un circuito impreso. Nótense los puntos de soldadura representados por pequeños círculos blancos.

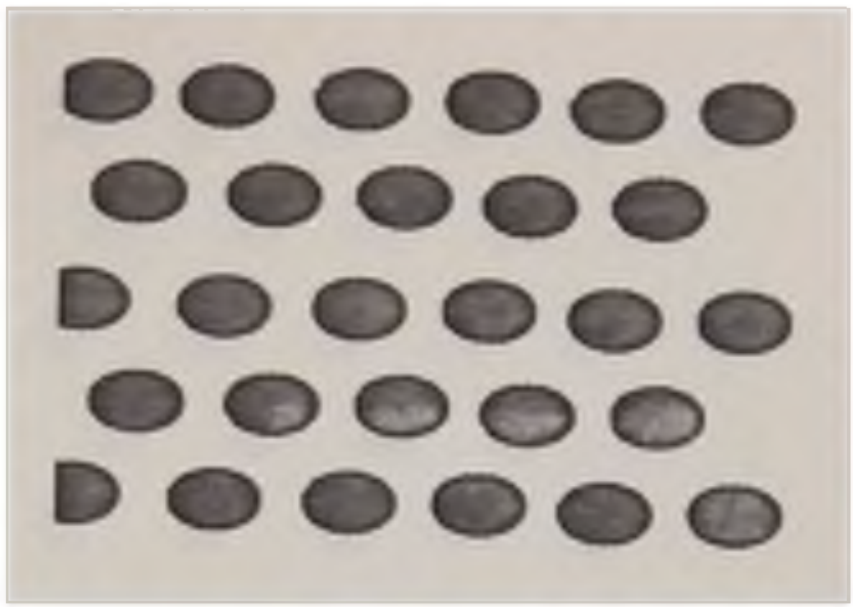

Imagen 2. Conjunto de círculos para probar la Transformada de Hough

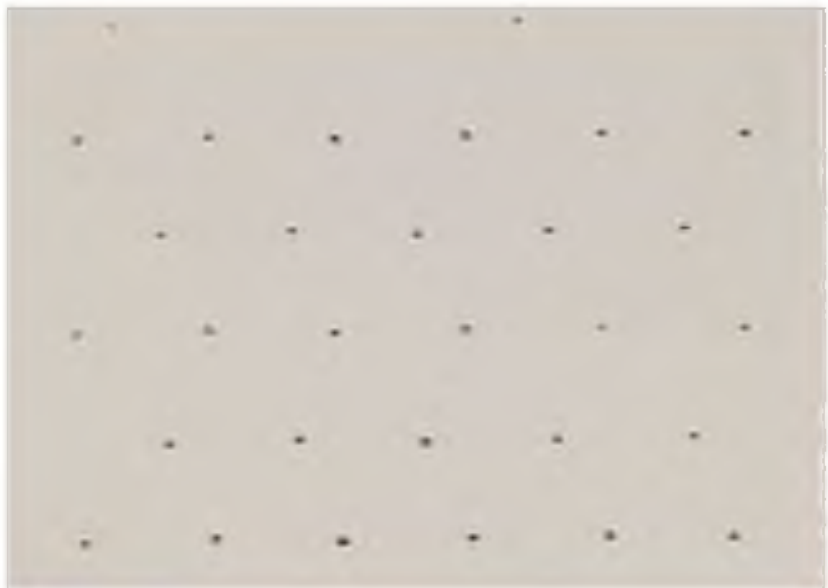

Imagen 3. Resultado obtenido después de la aplicación de la Transformada de Hough a la Imagen 2.

Imagen 4. Binarización de la Imagen 3. La magnitud de los vectores que apuntan hacia el centro de los círculos es menor o igual al diámetro de éstos.

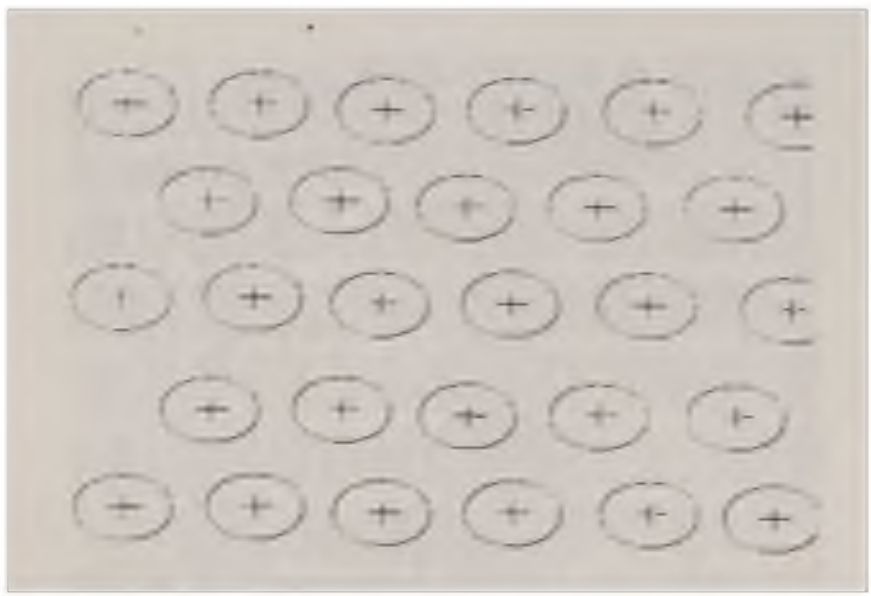

Imagen 5. Resultados obtenidos después de haber aplicado la Transformada de Hough a la Imagen No. 2. Nótese cómo el algoritmo encuentra todos los círculos que están en la imagen.

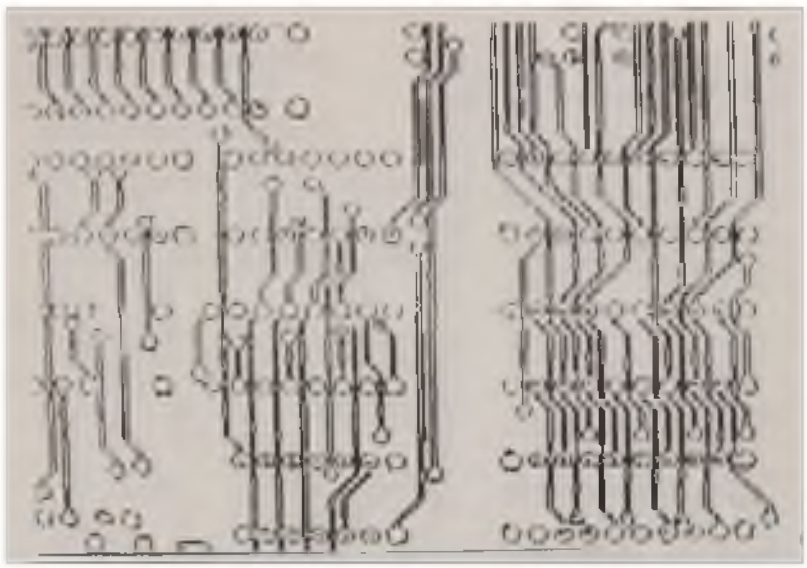

Imagen 6. Bordes obtenidos después de aplicar el método No. 1 basado en el operador Roberts.

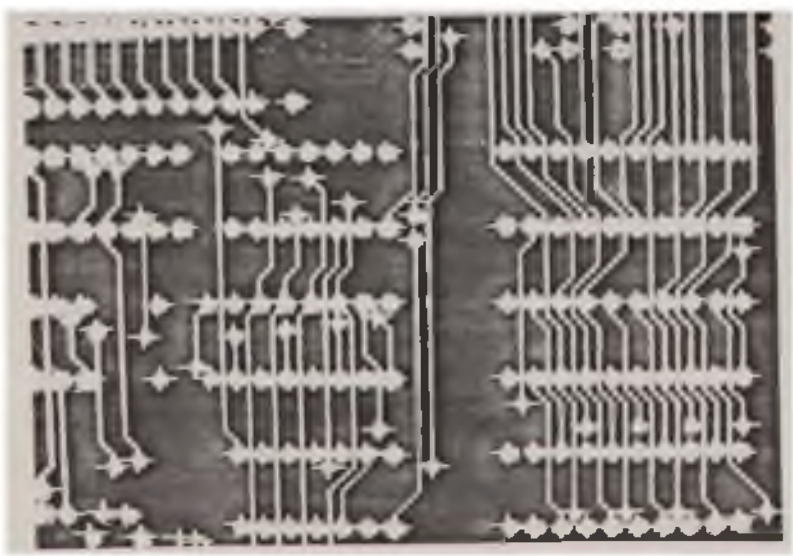

Imagen 7. Resultados de la localización de puntos de soldadura habiendo usado los bordes de la Imagen 6. Por cada punto de soldadura localizado se trazó una cruz.

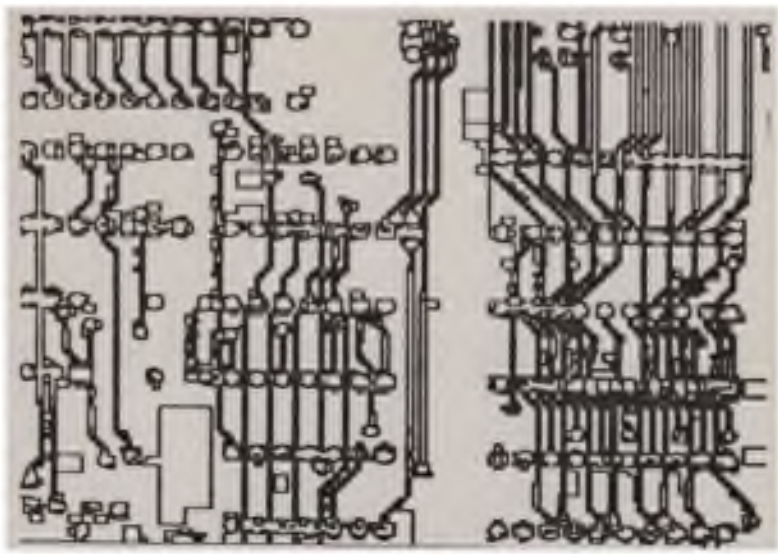

Imagen 8. Bordes obtenidos por medio del algoritmo de División y Unión y el operador Roberts.

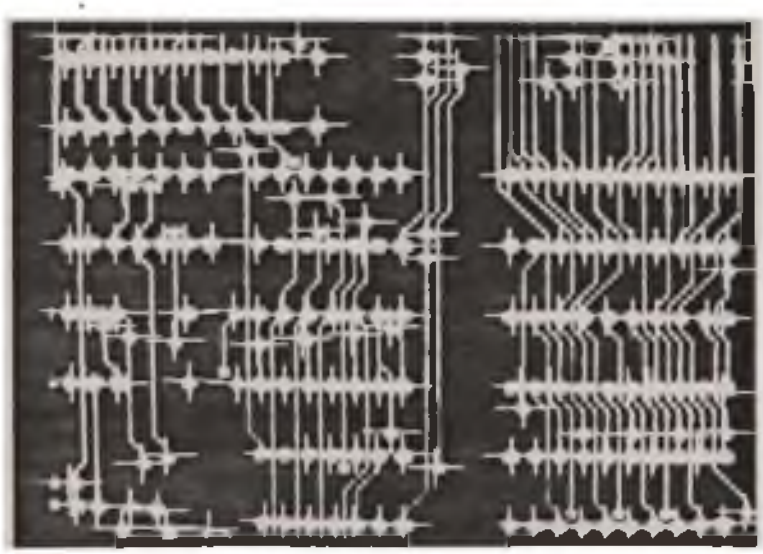

Imagen 9. Resultado de la localización de puntos de soldadura usando los bordes de la Imagen 8.

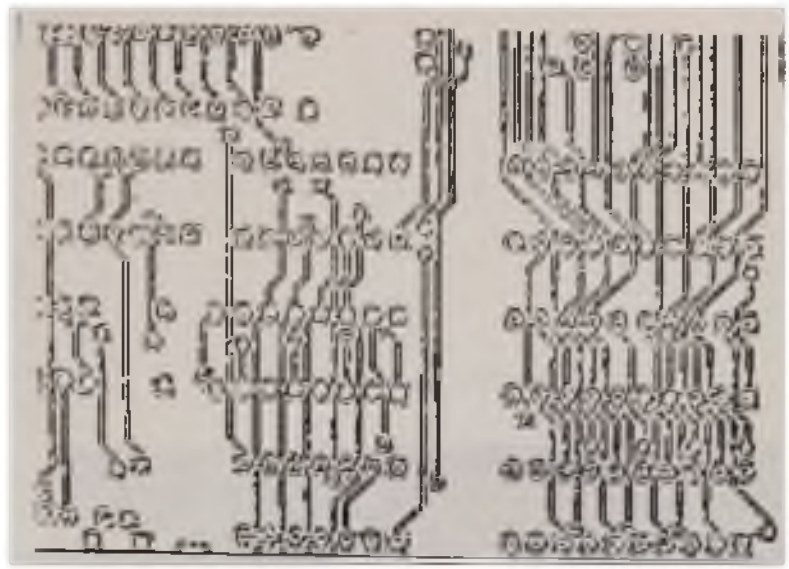

Imagen 10. Bordes obtenidos por medio del Algoritmo Optimo para Detección de Bordes.

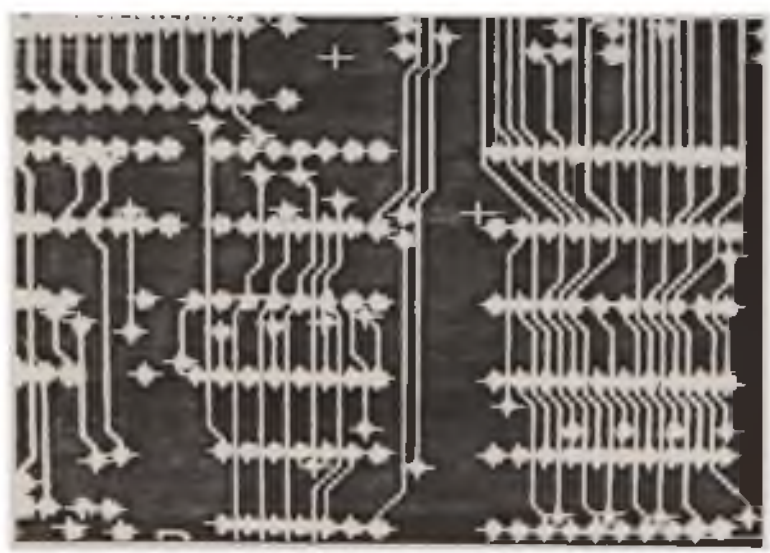

Imagen 11. Localización de puntos de soldadura usando los bordes de la Imagen 10.

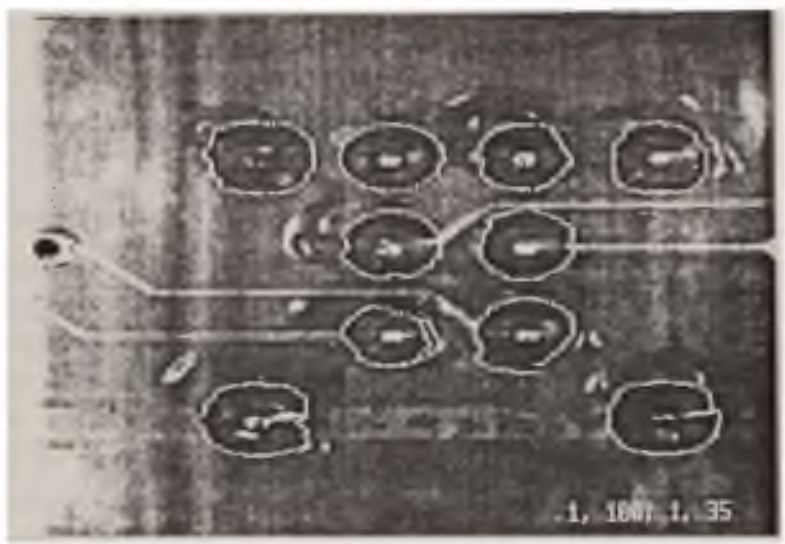

Imagen 12. Contornos finales producidos por el algoritmo seguidor de bordes.

APENDICE

4.1 El procesador denominado Transputer

Introducción.

La primera Transputer que se desarrollo fue en 1985 y se le denominó IMS T414. Esta Transputer es un CPU de 32 Bits y 2 Kbyte de RAM. Su procesador posee un reloj de 20 Mhz con el cual alcanza una eficiencia de 10 MIPS (Millones de Instrucciones por segundo). Además posee 4 líneas bidireccionales, las que sirven como interfaces de entrada y salida y un controlador de memoria que es programable.

Posteriormente a fines de 1987 empezó la producción del IMS T800, que es una versión mejorada del T414. Debido al empleo de una mejor tecnologia para fabricar memorias se logró aumentar la cantidad de la memoria a 4 Kbytes en un solo chip. Ademas se pudo implementar, también en el mismo chip, un procesador de punto flotante, el cual origina que el T800 pueda alcanzar una velocidad de 1.5 MFLOPS o 4 millones de Wheststones con un reloj de 20 Mhz.

Características. En la Fig. 28 se presenta un diagrama de las diferentes partes que componen a este procesador.

La Transputer, como ya se dijo anteriormente, posee además del CPU y del Bus de datos, 4 lineas (Links) bidireccionales, que sirven para comunicar a este procesador con otros 4. La comunicación y la transmisión de datos entre procesadores se lleva a cabo usando la técnica de DMA (Direct Memory Access), razón por la cual la comunicación se lleva a cabo independientemente de la parte de control de la Transputer receptora, permitiendo con esto el acceso a su memoria local, sin perturbar la ejecución de los procesos que la Transputer realiza en el momento de la comunicación y sin quitarles tiempo

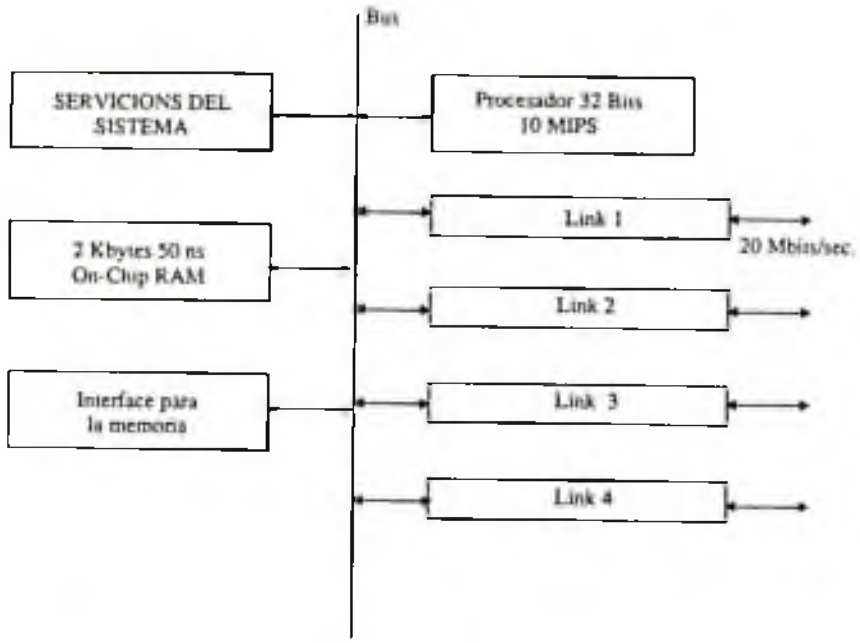

Fig. 28. Organización de la Transputer

de procesamiento. Esta característica de la Transputer origina que la transmisión en un sistema compuesto de varias Transputers no se vea limitada por el bus de comunicaciones del sistema.

Memoria y Registros

La memoria que una Transputer puede direccionar tiene una extensión de 4 Gbytes, ya que el procesador tiene un bus de direcciones de 32 lineas (4 Gbytes = 2**32). La memoria interna del CPU (2K Bytes para el IMS T414, 4K Bytes para el IMS T800) está situada en las primeras direcciones de la memoria. Debido a la existencia de esta memoria local no se incluyó un gran número de Registros internos en el diseño de la Transputer. Este procesador sólo posee 10 registros de los cuales 6 registros se usan para la ejecución secuencial de los programas y 4 más para su ejecución en paralelo.

En la Fig. 29 están representados los 6 primeros registros que la transputer posee y su función. A continuación aclaramos el funcionamiento de cada uno de ellos.

- *Apuntador al área de trabajo* (Work space pointer). Este registro realiza la función de un apuntador, el cual determina el área actual de trabajo de un proceso. El área está localizada en memoria principal y sirve para almacenar datos.

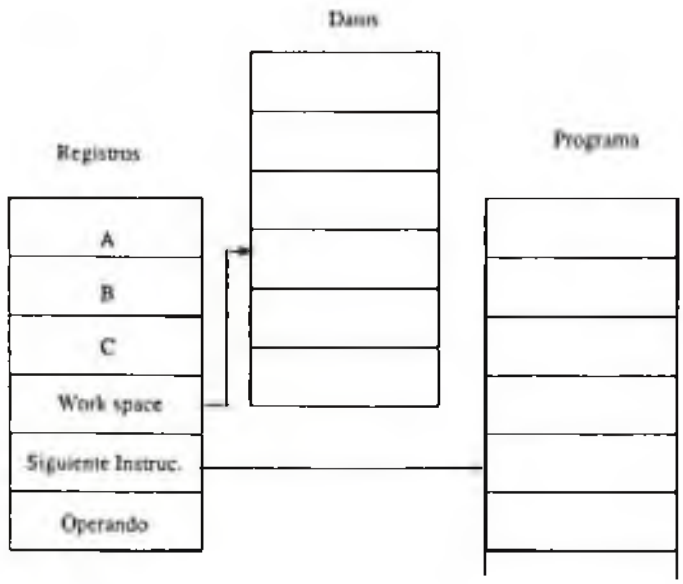

Fig. 29. 6 registros de la Transputer.

- *Apuntador a las instrucciones* (Instruction Pointer). Contiene la dirección de la siguiente instrucción a ejecutarse.
- *Registro de Operadores.* Este registro se usa para la formación de instrucciones más grandes de lo normal. En él se almacenan los operadores de éstas como se explica posteriormente.
- *Registros A, B y C.* Estos registros forman una pila, donde se encuentran los operandos que usan casi todas las instrucciones aritméticas y donde se depositan los resultados de las mismas. Estos registros equivalen a lo que en las computadoras CISC se denomina acumulador. Por ejemplo la instrucción Add suma el contenido de A y B, dejando el resultado de esta suma en A. Debido a estos registros muchas instrucciones no necesitan ninguna dirección para localizar sus operandos, sino que se refieren implícitamente al Stack. La decisión de escoger tres registros se debe a que basándose en estadísticas existentes se ha encontrado que con tres registros se puede alcanzar un balance muy eficiente entre el tamaño del código y la complejidad de su implementación (en hardware), además de que para el cambio de Tarea (task) dentro del procesador, es recomendable un número. de registros reducido.

Formato de las Instrucciones. La meta que se tenía (con respecto a

la programación) al desarrollar la Transputer fué la de poder programar el procesador con un lenguaje de alto nivel. Es por esto que para el diseño del formato de las instrucciones de máquina se cuidó que facilitaran e hicieran eficiente la compilación de programas escritos en lenguajes de alto nivel y no que facilitaran la escritura de programas a nivel ensamblador. A consecuencia las instrucciones de máquina poseen un formato regular (Fig. 30) y fueron escogidas de tal forma que ayudaran a la elaboración de las instruciones más comunes que ocurren en lenguajes de alto nivel. El conjunto de instrucciones es independiente del número de Bits que el procesador maneja. Las instrucciones poseen la propiedad de poder ser recolocadas en cualquier parte de la memoria.

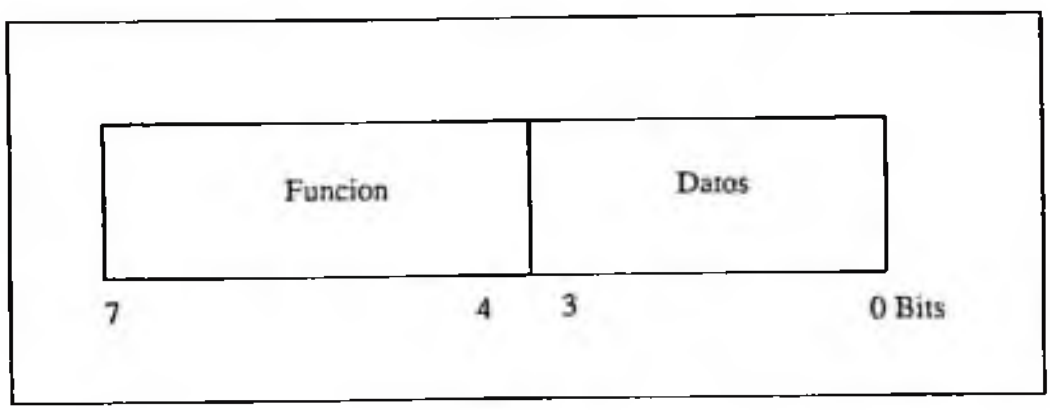

Fig. 30. Formato de las instruciones de Máquina de la Transputer.

Todas las instrucciones de máquina tienen una longitud de 1 Byte. Ademas de las funciones que se pueden representar en los 4 Bits dedicados al código de la instrucción, se pueden incluir cualquier número de funciones adicionales asignando a uno de estos códigos la función de definir una instrucción más larga que la normal. Este código juega el mismo papel que juega por ejemplo el código ESC en la codificación de instrucciones para impresoras. Con un ESC se pasa del modo normal de operación a un modo gráfico por ejemplo. En la Transputer a esta función se le llama prefijo.

Multiprocesamiento. Debido a que durante el desarrollo de la Transpu-

ter se tomó al lenguaje OCCAM [Dow 88] como su lenguaje de máquina, el lenguaje otorga un completo apoyo al Multitasking y a la comunicación sincronizada de procesos. El CPU de una Transputer contiene un despacha dor (Scheduler) implementado en microcódigo que es capaz, sin la ayuda de "Software" externo al procesador, de administrar un sinnúmero de procesos que son ejecutados concurrentemente, los cuales se reparten el tiempo del procesador. Es por esto que no es necesaria la utilización de Software especial (Kernel) para la administración de procesos [Try 89].

El despachador no consume tiempo de procesamiento en procesos inactivos. Los procesos activos, que están en espera de tiempo de procesador son anexados a una lista encadenada, figura 31. Esta lista está delimitada por dos apuntadores que señalan hacia el área de trabajo de cada proceso. Existen dos listas de este tipo debido a que la Transputer posee dos tipos de prioridades (un nivel bajo y otro alto).

Un proceso de baja prioridad se ejecuta hasta que se vuelve inactivo debido a la espera por una acción de entrada o salida, o a la espera de que un Timer termine de contar cierto tiempo, o debido a que el tiempo de procesador que le correspondía se ha terminado. Por el contrario, un proceso de alta prioridad permanece activo hasta que el mismo se dé por terminado o hasta que alguna causa de fuerza mayor lo haga esperar (E/S, Timer), pero no por falta de tiempo de procesador. El tiempo necesario para un cambio de proceso es muy pequeño, ya que existen muy pocos registros para ser almacenados.

Comunicación. De acuerdo al modelo de Comunicación del lenguaje OCCAM, la comunicación entre dos procesos se lleva a cabo mediante dos canales, los cuales llevan a cabo una comunicación privada punto a punto. La comunicación se realiza sin la intervención de un buffer y sin la intervención de colas de espera. Un canal establecido entre dos procesos que se están ejecutando en la misma Transputer está representado por una localidad de memoria, en cambio cuando los procesos se ejecutan en dos procesadores diferentes, se utilizan las lineas de comunicación (links) que cada Transputer posee. Esta comunicación se lleva a cabo por medio de instrucciones microprogramadas, las cuales determinan en el momento de su ejecución, si los canales que se van a usar para la comunicación son canales físicos o lógicos. Esta característica tiene la ventaja de que la compilación de un programa que usa canales físicos se lleva a cabo de la misma forma que un programa que usa canales lógicos. De esta forma se pueden desarrollar programas en un solo procesador que posteriormente utilizarán varios procesadores, sin tener que reprogramarlos o hacerles algun cambio.

Desarrollo futuro. Uno de los principales problemas que presenta la

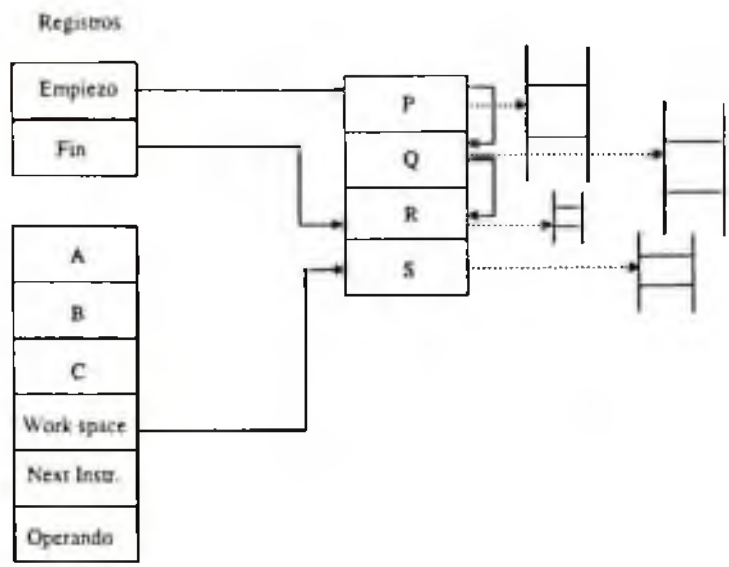

Fig. 31. Multitasking en la Transputer.

Transputer para el desarrollo de sistemas con varios procesadores es el número de lineas de comunicación, (que actualmente son 4) ya que existen algunas topologias en las que es conveniente tener más de 4 lineas de comunicación por procesador. Esta parece ser una de las limitaciones más importantes a vencer en versiones futuras de la Transputer.

Por último en la siguiente tabla se muestra la eficiencia de la Transputer con respecto a otros procesadores. Nótese la eficiencia de la Transputer comparada con la de otros procesadores que trabajan a la misma velocidad de reloj.

Comparación de la eficiencia de la Transputer

Procesador	Frecuencia (MHz)	Whetstones/s
Intel 80286/80287	8	300
IMS T414-20 (Transputer)	20	663
NS32332/32081	15	728
MC68020/68881	16/12	755
AT & T 32000/32100	18	1000
Clipper	33	2220
IMS T800-20	20	4000
IMS T800-30	30	6000

Bibliografía

[Ball 82] Ballard D.H. and Brown C. M. "Computer vison". Prentice Hall Inc., 1982.

[Bar 88] Bartlet S. L.,et al. "Automatic solder joint inspection". IEEE Transactions on PAMI, 10, 1988.

[Can 83] Canny J.F. "Finding edges and lines in images". MIT Artificial intelligence laboratory, TR 720, 1983.

[Can 86] Canny J.F. "A computational approach to edge detection". IEEE PAMI 8, 1986.

[Dav 84] Davies E.R. "Circularity - A new principle underlying the design of accurate edge orientation operators". Image and vision computing, 2, 1984.

[Der 87] Deriche R. "Optional edge detection using recursive filtering". Proceedings of the First international conference on computer vision. June 8-11, London, England, 1987.

[Dow 88] Dowsing R.D. "Introduction to currency using OCCAM". Van Nostrand Reinhold Co. Ltd., 1988.

[Fre 89] Freeman Herbert, ed., "Machine Vision for Inspection and Measurements", Academic Press Inc., Boston, San Diego, New York, 1989.

[Har 89] Haralick Robert M., et. al., "Pose estimation from corresponding point data", in [Fre 89].

[Hor 76] Horowitz S.L. and Pavlidis T. "Picture segmentation by a tree travel algorithm". Journal of the asssociation for computing machinery, 23, 368,1976

[Hor 86] Horn B. K. P. "Robot vision." The MIT Press. Mc Graw-Hill book Co., 1986.

[Hou 62] Hough P.V.C. "A method and means for recognizing complex patterns". U.S. Patent 3, 069,654, 1962.

[Ill 87] Illingworth J. and Kittler J."The adaptive Hough Transform". IEEE Transactions on Pattern Analysis and Machine Intelligence, 9, 1987.

[Ill 88] Illingworth J. and Kittler J. "A survey of the Hough transform". Computer vision, graphics and image processing 44, 87, 1988.

[Mar 76] Martelli A. "An application of heuristic search methods to edge and contour detection". Communication of the ACM 19, 1976.

[Nak 87] Nakugawa Y., Ninomiya T., "Three dimensional Vision systems using the structural light mehtod for inspecting solder joints and assembly robots", in Three-dimensional machine vision, Ed. Takeo Kanade 543, 1987.

[Nil 80] Nilson N.J. "Principles of artificial intelligence". Tioga Publishing Co., 1980.

[Nie 81] Niemann H., "Pattern Analysis", Springer-Verlag, Berlin Heidelberg New York, 1981.

[Pou 87] Pountain O., Rudolph R. "OCCAM-Das Handbuch". Verlag Heinz Heise Gmgh, Hannover, 1987.

[Ros 69] Rosenfeld A. "Pictures processing by computer". Academic Press, N.Y., 1969

[Ros 76] Rosenfeld A. and Kak A.C. "Digital picture processing". Academic Press, 1976.

[Shi 87] Shirai Y., "Three-dimensional Computer Vision", Springer- Verlag, Berlin, 1987.

[Schu 88] Schutte A. "OCCAM 2". Handbuch B.G. Teubner, Stutgart, 1988.

[Tri 70] Triendl E.E., "Skeletonization of Noisy Handdraw symbols using parallel operations", Pattern Recognition, Vol. 2, pp. 215-226, 1970.

[Try 89] Tryrrel A. M., J. D. Nicoud, "Scheduling and Parallel Operations on the Transputer", Microprocessing and Microprogramming, Vol. 26, 1989.

[Tsu 78] Tsuji S., Matsumoto F. "Detection of ellipses by a modified Houg Transformation". IEEE Trans. on computer, Vol. C-27, No. 8, 777, 1978.

[Vee 81] Veen van T.M., Green F.C., "Discretization errors in the Hough Transform", Pattern Recognition Vol. 14, No. 1-6, 1981.

[You 86] Young T.Y. and Fu K.S. "Handbook of pattern recognition and image processing". Academic Press Inc., 1986.

El jurado designado por la Sección de Computación del Departamento de Ingeniería Eléctrica del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó esta tesis el día 23 de enero de 1991

Dr. Guillermo Benito Morales Luna

Dr. José Luis Gordillo Moscoso

M. en C. Sergio Víctor Chapa Vergara

AUTOR ALTAMIRANO ROBLES, L.

TITULO LOCALIZACION DE PUNTOS DE SOLDADURA EN CIRCUITOS ....

CLASIF XM 91.5 RGTRO. B1-12045

NOMBRE DEL LECTOR	FECHA PREST.	FECHA DEVOL.
[illegible]	[illegible]	23/X/91
J. Garcia Ruiz	5/2/92	27/Feb/92
Dr. Guillermo Ulamba	7-12-95	21-12-95