PROCESAMIENTO Y COMPACTACION DE IMAGENES DE MARCAS DE GANADO

Ing.Marisa R. DE GIUSTI (*) - Ing.Guillermo A. JAQUENOD (**)

Resumen: Este articulo describe los programas desarrollados para la adquisición y procesamiento de imágenes de marcas de ganado, con el fin de su integración a una base de datos. Los datos digitalizados de cada marca son filtrados en forma interactiva para la eliminación de grafismos indeseados, y la imagen resultante es procesada para almacenarla con el mínimo gasto de memoria. El proceso de compresión se compone de una etapa de esqueletización, una segmentación en ramas, y una etapa final de codificación de cada rama en forma de serie con códigos de largo variable. Además de las máscaras de procesamiento empleadas, que muestran una aplicación convencional de tratamiento de imágenes, este articulo presenta como novedoso el método desarrollado para el filtrado simultáneo de 16 pixels por vez, exclusivamente por programa, mediante el uso de un procesador convencional.

Palabras Clave: IMAGE COMPRESSION - PARALLEL IMAGE PROCESSING

I. INTRODUCCION

El tatuaje de la piel de vacunos mediante un hierro a alta temperatura con un dibujo característico ("yerra") ha sido el método tradicional por el cual cada establecimiento ganadero señala su propiedad sobre cada uno de sus animales.

A fin de coordinar el uso de estas marcas, en la Provincia de Buenos Aires existe el Registro de Marcas de Ganado, dependiente de la Dirección de Ganadería del Ministerio de Asuntos Agrarios y Pesca, que posee actualmente cerca de 300.000 fichas de registro de marcas de ganado, de las cuales se encuentran en uso activo cerca de 150.000.

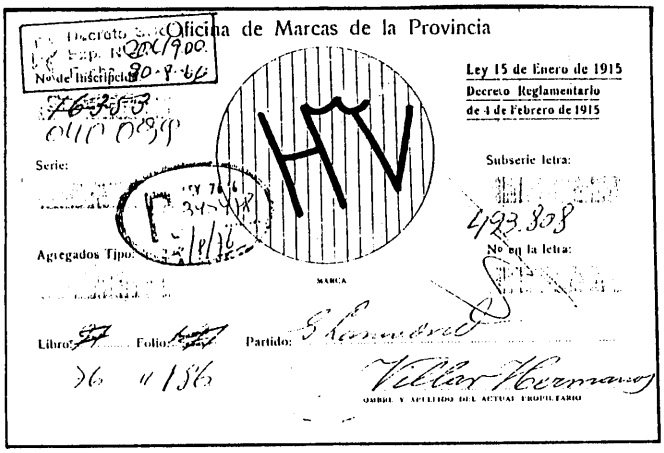

FIGURA 1

Cada registro se realiza sobre una tarjeta (Figura 1) que contiene los datos del establecimiento y un área cuadriculada donde se dibuja la marca correspondiente, con la cual serán tatuados los animales de dicho establecimiento.

La consulta manual de las fichas es totalmente impráctica por el gran número de registros, por lo que se planificó el desarrollo de una base de datos sobre computadora personal (PC), previéndose un producto a desarrollar en tres etapas: una inicial y dos ampliaciones para el futuro:

a) Producto inicial: Un sistema para la consulta integral de la información de marcas, ingresando por número de marca, del propietario, estado de pago, ubicación geográfica, etc.
b) Primera ampliación: posibilidad de buscar en forma automática una marca ingresando el dibujo de la marca de establecimiento.
c) Segunda ampliación: dada una marca presumiblemente adulterada mediante el agregado de trazos, detectar las posibles marcas preexistentes a partir d las cuales ésta pueda haber sido generada.

Mientras que el almacenaje de los datos de los establecimientos (nombre, responsable del mismo, número de ficha, etc.) es una tarea convencional de bases de datos, el archivo de la imagen de cada marca planteó un desafío novedoso, pues de usarse una resolución estimada como mínimo en 256x256 puntos (65536 puntos SI/NO = 8192 bytes) y efectuarse su almacenamiento directo, serían necesarios 2.400 Megabytes de capacidad de archivo, de difícil disponibilidad para un ámbito basado en PC, tanto por volúmen como por costo.

Para solucionar este inconveniente y dada la necesidad de tener en cuenta las ampliaciones mencionadas se optó por el uso de técnicas de procesamiento de imágenes; ello permite almacenar cada marca en una forma sumamente estructurada y con mínimo gasto de memoria.

Este proceso se compone de:

a) una etapa de adquisición mediante un digitalizador manual o de página.

(*) *Comisión de Investigaciones Científicas Provincia de Buenos Aires*
(**) *Facultad de Ingeniería - Universidad Nacional de La Plata.*
Calle 24 N.709 - (1900) LA PLATA - ARGENTINA
Tel: 54-21-24.31.52; email: guille@pir.edu.ar

b) una etapa interactiva de filtrado, para encuadrar y eliminar de la imagen de cada marca la retícula de fondo existente en la tarjeta de registro.

c) una etapa de esqueletización que 'adelgaza' a cada uno de los tramos de la marca hasta transformarla en una línea de espesor unitario. Es importante notar que el espesor de cada tramo de la marca no contiene información útil, sino sólo su forma (las marcas realizadas en los animales tienen espesor inherentemente variable, ya sea por la temperatura de la marca, el tipo de cuero del animal, su pelo, la edad, etc).

d) una etapa automática de segmentación del esqueleto en ramas, realizada junto a la codificación de cada rama en forma de una serie de microdesplazamientos (códigos de FREEMAN [1]), empleando códigos de largo variable tipo HUFFMAN [1] que aprovechan la redundancia inherente de la imagen para obtener la máxima relación de compresión.

Este artículo detalla la forma en que se han realizado las distintas etapas descriptas: La parte II describe el proceso de digitalización; la parte III relata el proceso de limpieza del fondo y encuadre del área a procesar poniendo énfasis en el algoritmo de procesamiento paralelo desarrollado; la parte IV describe la esqueletización de la marca en base a un algoritmo similar; la parte V la segmentación y codificación de cada rama del esqueleto y el factor de compresión obtenido; la parte VI los descriptores invariantes calculados en previsión a la primera ampliación y finalmente la parte VII las conclusiones y previsiones para la puesta en marcha de dichas ampliaciones.

II. ETAPA DE ADQUISICION

Esta etapa es totalmente convencional, y se realiza mediante un "scanner" comercial (en las pruebas se usó un Scanner manual **DFI/HD2000** [2]). Esta digitalización binaria de la marca (**imagen** = 1; **fondo** = 0) se realiza con una resolución de 100 puntos por pulgada sobre un área de 4"x4", resultando una retícula de trabajo inicial de 400x400 elementos de imagen ("*pixels*") .

A fin de simplificar las posteriores acciones de procesamiento, esta etapa permite al operador el encuadre de un area de interés de la imagen de 256x256 pixels y el consiguiente descarte de la zona externa.

III. ETAPA INTERACTIVA DE FILTRADO

Ya encuadrada, la imagen aún posee información indeseable, en forma de letras, números, manchas, y la retícula de fondo preexistente en las tarjetas de registro. La Figura 2.a es un ejemplo de imagen inicial, que posee incluso una línea indeseada producida por una pequeña basura en el scanner.

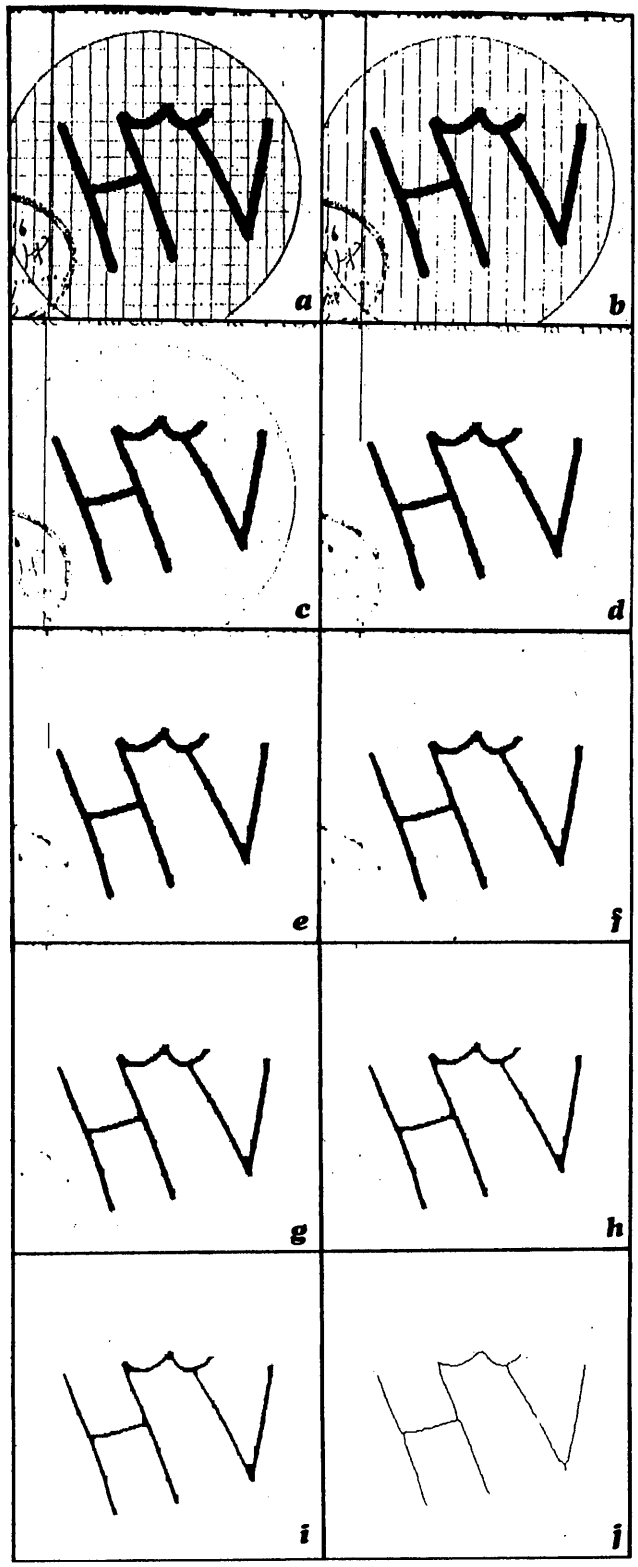

FIGURA 2

Como el color de la retícula y el de la marca son similares no existe la posibilidad de ningún filtrado óptico en la digitalización,

y la imposibilidad de reconfección de las tarjetas forzó a elegir un proceso de filtrado interactivo por programa, mediante la aplicación controlada de un operador del tipo '**contracción**' a la imagen para eliminar la reticula de fondo. En las figuras 2.b hasta 2.h se observa el resultado de la aplicación de sucesivos operadores de contracción a la imagen original, y cómo este proceso permite eliminar la casi totalidad de los elementos indeseables de la imagen.

Este operador opera sobre 'ventanas' de 3x3 pixels y su especificación es:

- *si alguno de los 9 pixels de la ventana de la imagen origen vale '0' (es decir corresponde a NO_IMAGEN) el pixel correspondiente al centro de la ventana de la imagen destino valdrá '0'.*

Dado que el procesador de un sistema tipo PC no es especialmente apto para el manejo de arreglos bidimensionales de información ni para instrucciones al bit, se aprovechó la particular configuración de la imagen en memoria, y la agilidad de estos procesadores para la realización de acciones elementales repetitivas [3].

La imagen de una marca está organizada por líneas, y cada línea se compone de sucesivos paquetes de 16 pixels almacenados como palabras de 16 bits en direcciones también consecutivas de memoria (Figura 3), donde el bit más significativo corresponde al pixel de la izquierda del paquete, y el menos significativo al de la derecha.

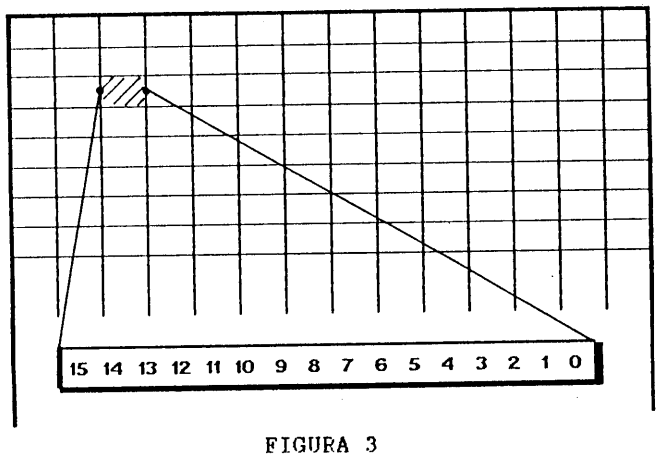

FIGURA 3

Si se indica ARRIBA con **N** (Norte), ABAJO con **S** (Sud), IZQUIERDA con **O** (Oeste) y DERECHA con **E** (Este), y se numeran los bits de una palabra desde 0 (LSB: bit de peso menos significativo) hasta 15 (MSB: bit de peso más significativo); dado que la imagen tiene 16 palabras por línea (256 pixels), la ventana de 3x3 pixels asociada al bit '**n**' de la palabra '**m**' (n = 1..14) tiene por vecinos:

Vecino NO: bit 'n+1' de la palabra (m-16)
Vecino N: bit 'n' de la palabra (m-16)
Vecino NE: bit 'n-1' de la palabra (m-16)
Vecino O: bit 'n+1' de la palabra m
Vecino E: bit 'n-1' de la palabra m
Vecino SO: bit 'n+1' de la palabra (m+16)
Vecino S: bit 'n' de la palabra (m+16)
Vecino SE: bit 'n-1' de la palabra (m+16)

Para acelerar la tarea de cálculo, a la imagen de la marca se le agrega una línea inicial y otra final de 16 palabras en '0', y en base a esta información (que se denomina ***TABLA0***) se generan dos nuevas tablas:

Tabla1: la imagen original desplazada un bit hacia la derecha, con todas las palabras asociadas al margen derecho de la imagen con el MSB en '0'.
Tabla2: la imagen original desplazada un bit hacia la izquierda, con todas las palabras asociadas al margen izquierdo de la imagen con el LSB en '0'.

Con estas tres tablas, la ventana de trabajo de 3x3 pixels asociada al bit '*n*' de la palabra '*m*' de la ***TABLA0*** está dada por:

Vecino NO: bit 'n' de TABLA1[m-16]
Vecino N: bit 'n' de TABLA0[m-16]
Vecino NE: bit 'n' de TABLA2[m-16]
Vecino O: bit 'n' de TABLA1[m]
Vecino E: bit 'n' de TABLA2[m]
Vecino SO: bit 'n' de TABLA1[m+16]
Vecino S: bit 'n' de TABLA0[m+16]
Vecino SE: bit 'n' de TABLA2[m+16]

Gracias a estas tablas los 9 componentes de una ventana corresponden al mismo bit en distintas palabras en las tres tablas y con distinto '*offset*' (figura 4); esto permite aprovechar la capacidad de direccionamiento y lógica del procesador, que ejecuta en una única instrucción el **AND(&)**, **OR(|)**, **NOT(~)** o **XOR(^)** bit a bit de 16 bits por vez.

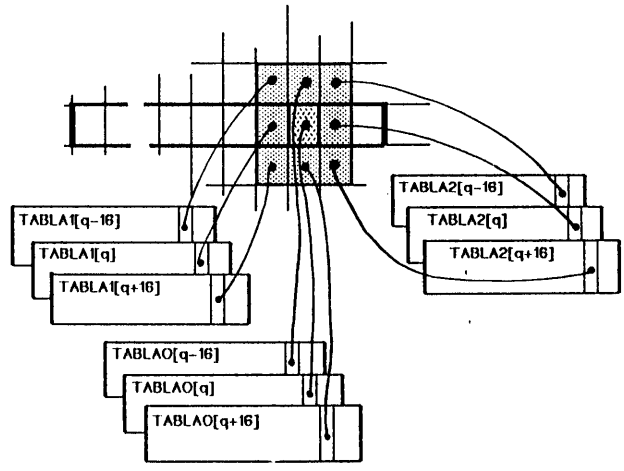

FIGURA 4

Si se llama ***TABLA_D*** al resultado de realizar una acción de contracción sobre la ***TABLA_0***, el procedimiento de obtención de ***TABLA_D*** es:

```
for (int q = 16; q < 4112; q++) {
 TABLA_D[i] =
 TABLA0[q] & TABLA1[q] & TABLA2[q] &
 TABLA1[q-16] & TABLA0[q-16] & TABLA2[q-16] &
 TABLA1[q+16] & TABLA0[q+16] & TABLA2[q+16];}
```

Esta descripción muestra que con sólo 4096 iteraciones se procesa toda la imagen de 256x256 pixels. Además, se puede ver que el cálculo de '**q+16**' e '**q-16**' en los índices de las tablas es innecesario: si se considera como valor asignado al nombre de una tabla la dirección de inicio de dicha tabla, el valor de ***TABLAO[q+16]*** puede ser visto en realidad como el contenido de una nueva tabla virtual ***TABLAOP[q]*** y el de ***TABLAO[q-16]*** como el contenido de otra tabla virtual ***TABLAOM[q]***, todas ellas superpuestas en memoria, sólo que ***TABLAOP[]*** comienza 16 palabras después de ***TABLAO[]*** y ***TABLAOM[]*** 16 palabras antes. Con este criterio, la operación resultante es:

```
for (int q = 16; q < 4112; q++) {
 TABLA_D[i] =
  TABLA_0 [q] & TABLA_1 [q] & TABLA_2 [q] &
  TABLA_1M[q] & TABLA_0M[q] & TABLA_2M[q] &
  TABLA_1P[q] & TABLA_0P[q] & TABLA_2P[q] ;}
```

Como en las tarjetas de registro existen manchas y otros elementos de gran espesor, imposibles de eliminar por contracciones sin afectar también al dibujo de la marca, cuando se ha eliminado la retícula del fondo, el usuario dispone aún de la posibilidad de aplicar un '**borrador**' manual para quitar todo trazo de imagen indeseada.

IV. ESQUELETIZACION DE LA IMAGEN

Ya con la imagen '**limpia**' (figura 2.i), el programa pasa a operar en modo automático y procede a la esqueletización de la imagen usando el método de '**incendio de pastos**' [4], usando en forma iterativa cuatro funciones de adelgazamiento direccional, desde el Norte, Sur, Este y Oeste, hasta que la imagen no adelgaza más (figura 2.j).

Si se llama '***imagen_0***' a la imagen original, e '***imagen_q***' (**q=1..4**) a imágenes intermedias, la esqueletización se describe por:

```
do {imagen_1 = imagen_0;
    imagen_2 = adelgazo_N (imagen_1);
    imagen_3 = adelgazo_E (imagen_2);
    imagen_4 = adelgazo_S (imagen_3);
    imagen_0 = adelgazo_O (imagen_4);
   } while (imagen_1 != imagen_0);
```

Para los operadores de adelgazamiento se emplea la misma técnica que para el de contracción (es decir, se procesan 16 pixels por vez), sólo que las funciones son algo más complejas, y ya no sólo el **AND** u **OR** de los pixels de la ventana.

Para ello se generan tres nuevas tablas llamadas TABLA_3, TABLA_4 y TABLA_5, donde:

- *TABLA_3 es una copia de TABLA_0 con todos sus elementos negados.*
- *TABLA_4 es una copia de TABLA_1 con todos sus elementos negados.*
- *TABLA_5 es una copia de TABLA_2 con todos sus elementos negados.*

Con estas tablas, la función ***adelgazo_N()*** (incendio de pastos desde el Norte) se define como:

```
for (q=16;q<4112;q++) TABLA_D[q] &= ~skN(q);
```

Donde la función esqueletizadora de palabra desde el Norte es:

```
skN (q) = g*j*((e*i) + (e*d) + (k*d));
```

Siendo:

```
a = TABLA_1[q-16];    b = TABLA_0[q-16];
c = TABLA_2[q-16];    d = TABLA_1[q];
e = TABLA_2[q];       f = TABLA_1[q+16];
g = TABLA_0[q+16];    h = TABLA_2[q+16];
i = TABLA_4[q-16];    j = TABLA_3[q-16];
k = TABLA_5[q-16];    l = TABLA_4[q];
m = TABLA_5[q];       n = TABLA_4[q+16];
o = TABLA_3[q+16];    p = TABLA_5[q+16];
```

Debe aclararse que acá vale también el análisis que en la parte III respecto a lo innecesario del cálculo de '**q+16**' y '**q-16**'.

De igual modo, para las otras direcciones de esqueletización, los operadores respectivos, cuya obtención se detalla en el Apéndice I, resultan:

```
skE (q) = d*m*((g*k) + (g*b) + (p*b));
skS (q) = b*o*((d*o) + (d*e) + (n*e));
skO (q) = e*l*((b*n) + (b*g) + (l*g));
```

V. SEGMENTACION DEL ESQUELETO Y CODIFICACION DE CADA RAMA

El proceso de **segmentación** del esqueleto trata de comprimir la información para que ocupe un espacio mínimo, pero en forma que facilite su uso en "pattern recognition".

Para ello se realiza una iteración sobre la imagen del esqueleto quitando en cada lazo una rama, que es codificada en forma de un punto inicial y una cadena de desplazamientos unitarios en las ocho posibles direcciones de la imagen, del tipo "serie de FREEMAN"[1].

Este proceso, para cada rama, se compone de dos etapas:

a) Búsqueda del punto inicial: ello implica buscar en la imagen algún punto singular de las siguientes posibles jerarquías:

Jerarquía A: extremo de línea o encrucijada: para detectarlos se usa el mismo método que para el adelgazamiento y esqueletización, sólo que con otros operadores.

Jerarquía B: el primer punto que aparece en el proceso de búsqueda: esta acción es mucho más simple, y consiste en ubicar el primer punto de imagen haciendo una inspección tipo "raster" de la memoria.

Si existe un punto con jerarquía **A**, se empieza por él, sino por uno de jerarquía **B**, hasta que se ha codificado toda la imagen. Cuando existe un punto inicial, se almacena su coordenada (2 bytes) y a partir de él se inicia la búsqueda encadenada de vecinos.

b) Codificación de la rama: Dado un punto inicial, se codifica la dirección absoluta del desplazamiento ('rumbo') para llegar al primer punto vecino usando 3 bits, para indicar una de las posibles 8 direcciones:

CODIFICACION ABSOLUTA DE FREEMAN (3 bits):

000 : E	001 : NE	010 : N	011 : NO
100 : O	101 : SO	110 : S	111 : SE

A partir de allí, para los puntos restantes se explota la redundancia de la imagen, donde es más probable que el rumbo siga igual o cambie poco a que sufra cambios bruscos, y se usa para cada desplazamiento códigos de largo variable:

CODIFICACION RELATIVA FH (FREEMAN/HUFFMAN):

```
00    : rumbo actual = rumbo previo
01    : rumbo actual = rumbo previo + 45
10    : rumbo actual = rumbo previo - 45
1100  : rumbo actual = rumbo previo + 90
1101  : rumbo actual = rumbo previo - 90
11100 : rumbo actual = rumbo previo + 135
11101 : rumbo actual = rumbo previo - 135
11110 : indicador de FIN DE RAMA
11111 : fin de marca
```

Tal como surge de la tabla, la codificación de los rumbos más probables requiere sólo dos bits, en oposición a los tres necesarios en el caso de codificación absoluta, dando por ello una mejor relación de compresión. Este proceso finaliza al llegar a un nuevo punto de jerarquía A, situación en la cual se marca la cadena con un indicador de FIN DE RAMA.

Para ejemplificar la relación final de compresión obtenida, una muestra piloto de 57 marcas pudo ser almacenada en 9658 bytes, con un gasto de memoria promedio de 170 bytes por marca, de los cuales 154 son para la imagen en sí, 4 para un número propio de la marca y 12 bytes para almacenar los descriptores que se detallan en la parte VI. Comparando este gasto de memoria con el almacenamiento directo de la imagen, la relación media de compresión resulta de 52:1, lo que significa poder almacenar las 300.000 marcas existentes con sólo 51 Megabytes, fácilmente disponibles y accesibles en costo para sistemas tipo PC.

VI. DESCRIPTORES INVARIANTES

En el desarrollo inicial se incorporó el cálculo de ciertos descriptores para su uso futuro en la localización de marcas en base a su dibujo. Con ellos se pretende que mediante su comparación con los invariantes de una marca incógnita (Por ejemplo: de un animal de dueño desconocido) se genere un pequeño repertorio de marcas similares entre las 150.000 en uso, para su búsqueda manual por un operador.

Para esta acción debe considerarse que, aunque la orientación de la marca puede ser considerada medianamente estable (por ejemplo, con variaciones de 15 grados en más o menos), su tamaño aumenta al crecer el animal, lo que obliga a un proceso de ecualización dimensional (**scaling**) para su encuadre en un marco estándar previo a cualquier posible acción de comparación.

Para esta tarea se hizo uso de un árbol de decisión (´**pasa**´/´**no pasa**´) y se definió una estructura de datos llamada ´***marca***´ dada por:

```
typedef struct {
  int perime;      /* perímetro de la marca */
  int radio,rmax,rmin;  /* radios      medio,
                           máximo y mínimo */
  unsigned char deltar;
  int aspecto; char orienta; } marca;
```

El cálculo de estos valores, requiere del cálculo de los momentos ´***mij***´ ***(i,j=0,1,2)*** del esqueleto para todos los puntos de coordenadas (x,y), y se realiza según sigue:

a) Perímetro, cantidad de puntos (m00) y posición del Centro de Gravedad (C.G.):

´*perime*´: suma de las longitudes de las ramas de la marca: se calcula sumando las longitudes de los sucesivos desplazamientos incrementales.

´*m00*´: este valor es la cantidad de puntos que componen el esqueleto de la marca.

´*m10*´ = SUM (x);
´*m01*´ = SUM (y)
´*CGx*´ = m10/m00;
´*CGy*´ = m01/m00

b) Parámetros *radio, rmax, rmin, deltar, aspecto, y orienta*: se emplearon criterios de agrupamiento y la matriz de dispersión del esqueleto [5] para técnicas de "*line fitting*" mediante autovectores. En este paso, a los valores de ´*x*´ e ´*y*´ empleados se les descontó la posición del Centro de Gravedad.

```
m11     = SUM [x * y];
m20     = SUM [x↑2]; m02     = SUM [y↑2];
r       = sqrt(x↑2 + y↑2)
rmax    = MAX [r];   rmin    = MIN [r];
radio   = sqrt((m20 + m02)/m00);
deltar  = sqrt (SUM( (r - radio)↑2 )/m00);
```

Habiendo calculado los momentos cuadráticos, la "scatter matrix" resulta:

$$\begin{bmatrix} m20 & m11 \\ m11 & m02 \end{bmatrix}$$

Los autovalores son los valores de ´*l*´ (lambda) que satisfacen:

$$det\left(\begin{bmatrix} m20 & m11 \\ m11 & m02 \end{bmatrix} - l * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0$$

$$det\begin{bmatrix} m20-l & m11 \\ m11 & m02-l \end{bmatrix} = (m20-l)*(m02-l)-m11^2 = 0$$

Cuya resolución da dos raíces *alfa* y *beta*:

```
alfa
  = (m20+m02 + sqrt((m20-m02)↑2 + 4*m11↑2))/2

beta
  = (m20+m02 - sqrt((m20-m02)↑2 + 4*m11↑2))/2
```

Y así sale la relación de aspecto, dada por:

```
aspecto = alfa/beta
```

Para el cálculo de la orientación de la marca se calcula la del autovector principal ´*V*´ asociado al mayor autovalor (alfa):

$$\begin{bmatrix} m20 & m11 \\ m11 & m02 \end{bmatrix} * \begin{bmatrix} Vx \\ Vy \end{bmatrix} = \begin{bmatrix} alfa.Vx \\ alfa.Vy \end{bmatrix} = \begin{bmatrix} m20.Vx + m11.Vy \\ m11.Vx + m02.Vy \end{bmatrix}$$

De donde: $(m20 - alfa).Vx + m11.Vy = 0;$
$(m11 - alfa).Vx + m02.Vy = 0;$

Si se supone $Vx = 1$ se tiene para Vy:

$Vy = (m20-alfa)/m11 = (m11-alfa)/m02;$

Y la orientación: *orienta = atan2 (Vy,1);*

VII. CONCLUSIONES

El uso de bases de datos gráficas se encuentra en pleno auge ante la aparición de medios masivos de almacenamiento; estos recursos mantienen, sin embargo, la exigencia de manejo de enormes volúmenes de información y por lo tanto exigen elevado ancho de banda en los procesadores; por su parte, el empleo de técnicas de compresión de imágenes, aunque usado en aplicaciones satelitales [1], es aún poco difundido a nivel masivo. En todos los casos, los métodos de acceso a estas bases de datos son sumamente primitivos y aún es un tema poco explorado (excepto las obvias aplicaciones militares) el desarrollo de productos para la consulta de este tipo de bases de datos por similitud de imágenes (memorias gráficas asociativas).

En este trabajo se realizan algunos pasos elementales en esa dirección, y su idea clave es la explotación inteligente de la capacidad de procesadores convencionales para trabajar con imágenes; el método descripto permite procesar tantos pixels por vez como tenga el 'bus' de datos de la CPU, lo que resulta sumamente interesante para los actuales microprocesadores y DSPs de 32 y 64 bits.

Las líneas futuras de búsqueda se orientarán hacia el empleo de descripciones sintácticas [6] de cada rama del esqueleto, y el desarrollo de gramáticas para análisis y comparación de estas descripciones. Ello permitirá mayor eficacia en la consulta mediante ingreso gráfico, hoy limitada por los descriptores empleados, y en una etapa posterior, quizás pueda ser usada para detectar marcas adulteradas.

VIII. REFERENCIAS

[1] Gonzalez, Rafael & Wintz, Paul."DIGITAL IMAGE PROCESSING". Addison Wesley Pub.Co., 6th.Printing, USA, 1979.
[2] "HD-2000 Handy Scanner Users Manual". DFI, USA, 1987.
[3] Assembler 8086/286. Intel, USA, 1986.
[4] Cernuschi, Bruno. "VISION PARA COMPUTADORAS". III EBAI, Brasil, 1988.
[5] Duda, Richard O. & Hart, Peter E."PATTERN CLASSIFICATION AND SCENE ANALYSIS". John Wiley & Sons, USA, 1973.
[6] Chang, Tien-Chien. "EXPERT PROCESS PLANNING FOR MANUFACTURING". Addison Wesley Pub.Co., USA, 1990.

APENDICE I: Obtención de las funciones de adelgazamiento o esqueletización direccional

Dado un pixel con valor '1', ubicado en el centro de una ventana de 3x3 elementos, existen 256 posibles combinaciones de los 8 elementos que los rodean, que definen si ese pixel debe o no ser puesto a '0' en el proceso de adelgazamiento. Si se asigna peso binario a los vecinos del pixel central, de la siguiente forma:

a = 128	b = 64	c = 32
d = 16	X	e = 8
f = 4	g = 2	h = 1

Y se considera la función de esqueletización desde arriba hacia abajo *sk_N()* para todos los casos, considerando la existencia de simetría central, y con las siguientes reglas:

- *Un punto aislado X es borrable.*
- *Un punto X es borrable sólo si su eliminación no interrumpe ninguna línea de espesor unitario.*
- *Un punto X es borrable sólo si no tiene ningún punto inmediatamente arriba suyo.*
- *Un punto X que sea extremo de línea no debe borrarse.*

Resultan los siguientes valores, para los 256 posibles pesos de la ventana:

0...15	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
16...31	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
32...47	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
48...63	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
64...79	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
80...95	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
96..111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
112..127	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
128..143	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
144..159	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
160..175	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
176..191	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1
192..207	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
208..223	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
224..239	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
240..255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

de donde

```
skN () =
g * /b *((e * /a) + (e * d) + (/c * d))
```

Si se llama:

```
i = ~a; j = ~b; k = ~c; l = ~d;
m = ~e; n = ~f; o = ~g; p = ~h;
```

Resulta:

```
skN () = g*j*((e*i) + (e*d) + (k*d));
```

De igual forma, las funciones para adelgazar desde otras direcciones se obtienen por permutación de variables, quedando:

```
skE () = d*m*((g*k) + (g*b) + (p*b));
skS () = b*o*((d*p) + (d*e) + (n*e));
skO () = e*l*((b*n) + (b*g) + (i*g));
```