Universidad Tecnológica de la Mixteca

"Obtención de características deltas, cores y encierros en una huella dactilar"

Tesis para obtener el título de

Ingeniero en Computación

Presenta

Gilberto Emilio Hernández Sorroza

Asesor

Dr. Mario Aurelio Rodríguez Pineda

HUAJUAPAN DE LEÓN, OAX.

ÍNDICE

A mi madre.
Porque siempre estuviste conmigo
en todo momento,
porque gran parte de tus esfuerzos
fueron dedicados a mí,
ahora el resultado de los míos
te los dedico, son tuyos.
Gracias madre.

A mi asesor.
Por su paciencia, su experiencia
y su apoyo.
Pero sobre todo por su gran enseñanza.
Gracias profe.

Gracias Erika.
Por ser tan importante en mi vida,
y porque a pesar de todo
supimos apoyarnos y lograr nuestras metas.
... gracias por siempre.

A mis niños, Dana y Andy,
por ellos y para ellos.

Introducción

La historia muestra cuán importante es controlar y erradicar la delincuencia, por ello en diversas épocas se han establecido sistemas de identificación. Ya en 1888 Emille Villebrum presentó un interesante estudio sobre la clasificación de las uñas para fines de identidad personal. En el mismo año, Frigorio consideró que la oreja era la parte del cuerpo que más ventajas ofrecía para establecer un sistema de identificación el cual él mismo propuso y llamó otometría; este se basaba en la forma y medidas de las orejas. Al siguiente año, el alemán Levinshon propuso para el mismo fin la obtención de retinogramas o fotografías para el fondo del ojo, el cual se podía clasificar según su forma, contorno y profundidad. El juez italiano Luis Alfonso ideó en 1896 un aparato que denominó el craneógrafo, para tomar las medidas del cráneo desde la raíz hasta la nuca.

De todos los métodos señalados y otros que no hemos mencionado, no hubo alguno mediante el cual se pudiera identificar especialmente a delincuentes y criminales, sin atenerse a sus propios informes o declaraciones.

Fue por medio de la dactiloscopia como vino a solucionarse, de manera inequívoca y definitiva, el conflicto que durante tantos siglos afrontó la humanidad sobre la identidad de las personas; pues concluyentes experimentos científicos han demostrado que las crestas papilares poseen tres cualidades fundamentales, a saber: *perennes, inmutables y diversiformes*. Estas particularidades de las huellas proceden de la estructura de la piel que forma líneas, en general paralelas, las cuales varían suavemente alrededor del dedo. Esta estructura permanece invariante durante la vida, y permite diferenciar personas a través de varias formas características que forman las líneas [31].

Los fundamentos de la identificación por medio de huellas dactilares fueron establecidos en los trabajos de F. Galton [16] y E. Henry [21] a finales del siglo XIX. Una huella dactilar está formada por una mezcla de segmentos de curvas. Las regiones claras de las huellas dactilares se llaman *crestas* mientras las regiones obscuras se llaman *valles*. En sus trabajos Galton introdujo el concepto de *minutiae* (minucias) para representar las discontinuidades en los patrones de flujo de los valles, como características discriminantes y mostró su unicidad y permanencia. Henry estudió la estructura global de las huellas dactilares y estableció el famoso *"Sistema Henry"* de clasificación. Tales trabajos han sido ampliados por Vucetich [11], y muchos otros. Durante el siglo XX, las huellas dactilares fueron formalmente aceptadas como un signo válido de identidad [27] por las agencias de impartición de justicia.

La identificación manual de huellas dactilares es tediosa y consume mucho tiempo, sobre todo cuando es necesario realizar la clasificación en una colección muy grande de personas. Por esta razón, en 1960, la Oficina de Investigaciones del Reino Unido y el Departamento de Policía de París iniciaron estudios en sistemas de identificación automática de huellas dactilares. Para ello, básicamente asumieron la clasificación dada por Henry, las Características de Galton (fig. 1.2) y la existencia de dos tipos especiales de características llamadas *deltas* y *cores* (fig. 1.3).

Para clasificar automáticamente huellas dactilares existen básicamente cuatro enfoques: *sintáctico* ([5], [29-30]), *estructural* ([23], [32]), *redes neuronales* ([15], [17-19], [26], [33]) y *estadístico* ([10], [14]).

En este trabajo, utilizando métodos de la Topología Combinatoria, se identifican los puntos candidatos a ser *deltas* o *cores* por medio del índice de Poincaré [25-26].

Además, usando el concepto de índice de un punto ([2],[35]) y los algoritmos dados en [1], se calculan los 1- números de Betti de la imagen con el fin de obtener el número de encierros de la huella.

Para lograr lo anterior, se han seguido los siguientes pasos:

a) Obtener la imagen digital en tonos de gris, de una huella dactilar
b) Definir un campo vectorial sobre la imagen digital
c) Obtener una matriz de direcciones
d) Suavizar el campo vectorial
e) Seleccionar vecindades adecuadas alrededor de cada punto de la imagen
f) Hallar la dirección predominante en cada vecindad
g) Calcular el índice de Poincaré en cada punto de la matriz
h) Obtener una relación entre el valor del índice obtenido, con el tipo de puntos singulares presentes en la huella.
i) Calcular el número de encierros.

El objetivo fundamental de este trabajo, es obtener la relación entre el valor del índice de Poincaré y el tipo de punto singular presente en una huella dactilar, así como un método para calcular el número de encierros.

El trabajo está organizado de la siguiente manera:

En el Capítulo I se describen los aspectos de dactiloscopia considerados en los capítulos siguientes.

El Capítulo II trata los principios del procesamiento de imágenes, mismos que se han utilizado para realizar el trabajo. Asimismo, se dan los conceptos necesarios de topología digital y topología combinatoria que conllevan a establecer una relación entre puntos singulares de una huella dactilar y los puntos singulares de un campo vectorial y el cálculo de números de encierros.

En el Capítulo III se presentan los algoritmos utilizados, así como los resultados obtenidos.

El anexo 1 contiene el código de los algoritmos motivo del presente trabajo.

Capítulo I
Conceptos de dactiloscopia.

Con el fin de precisar aquellas características de huellas dactilares que serán obtenidas en este trabajo, el presente capítulo trata los conceptos básicos de dactiloscopia.

1.1. Identificación a través de la historia [31].

La identificación de personas ha sido uno de los problemas en el que se han ocupado muchas personas en diversas épocas. Entre ellas encontramos que las leyes de Manú, emanadas de la India, establecían, para facilitar la identificación de los malhechores, imprimir con hierro candente en la frente de los delincuentes una marca con características especiales para cada delito.

Marcas semejantes se usaron en Grecia y Roma, solamente que eran practicadas en diversas partes del cuerpo humano (época del emperador Constantino).

En España durante el siglo XV, se herraba el rostro de los esclavos. En Rusia se cortaba la nariz o las manos a ciertos criminales para identificarlos.

En Cuba también se vio mutilar a los esclavos cimarrones, inspirados en la ley más antigua que se conoce, la del rey babilónico Hamurabi.

El jurisconsulto y filósofo Bentham, en 1820, proponía nuevamente en Alemania el tatuaje como procedimiento identificativo. La idea fue desechada.

En la actualidad, los tatuajes voluntarios son frecuentes y se toman en cuenta dentro de las descripciones de señas particulares de los individuos.

En 1910, Icard, de Marsella, aconsejaba las inyecciones subcutáneas de parafina, que dejarían nudosidades indelebles.

La utilización sistemática de las mediciones óseas ha sido el punto de partida y el origen del método personal de identificación, conocido mundialmente con el nombre de antropometría.

Este se basa en los tres principios siguientes:

1. La estabilidad del esqueleto humano desde los veinticinco años.
2. La múltiple variedad de dimensiones que presenta el esqueleto humano comparando un ser con otro ser.
3. La facilidad y la precisión relativa con que puede verificarse las mediciones sobre el ser humano, y solo con un sencillo compás o la barra de medir.

Emile Villebrum presentó en 1888 un interesante estudio sobre la clasificación de las uñas para fines de identidad personal. Frigorio en 1888 consideró que la oreja era la parte del cuerpo que más ventajas ofrecía para establecer un sistema de identificación y propuso su sistema que denominó otometría basado en las medidas y formas de la oreja. El juez italiano Luis Alfonso ideó en 1896 un aparato que denominó el craneógrafo, el cual servía para tomar las medidas del cráneo desde la raíz de la nariz hasta la nuca y aplicarla como sistema de identificación. En 1889, el alemán Levinshon propuso la obtención de retinogramas o fotografías del fondo del ojo, el cual se podía clasificar según su forma, contorno y profundidad.

De todos los métodos señalados y otros que no hemos mencionado, no hubo alguno mediante el cual se pudiera identificar especialmente a delincuentes y criminales, sin atenerse a sus propios informes o declaraciones.

1.2. La ciencia de las huellas dactilares y su historia [11]

La observación de huellas dactilares se remonta hacia muchos años atrás. En las antiguas civilizaciones del Lejano Oriente durante muchos siglos la impresión dactilar del pulgar del Emperador fue el signo usual con que el gobernante certificaba los documentos de Estado; así, pues, en China, Oriente y Egipto, se aceptaban las impresiones digitales en substitución de las firmas de personas analfabetas, así como también para identificar criminales, práctica que al ser revivida en la India tuvo una influencia decisiva en los componentes de una comisión investigadora denominada TROUP.

Cuenta Edmond Locard que el hombre de Aurignac acostumbraba reproducir entre los medios decorativos de sus dibujos, especialmente los de su propia mano.

En el siglo XVII, por los años de 1628 a 1694, el anatomista italiano Marcelo Malpighi fue el primer europeo que de modo científico se interesó por las huellas dactilares e hizo referencia a las diversas figuras que presentan las palmas de las manos, observó que las líneas en las yemas de los dedos formaban lazos, círculos y espirales y con esto entrevió la posibilidad de llegar por ese camino a la formación de una clasificación.

Teniendo en cuenta esto, Locard considera a Marcelo Malpighi como el "abuelo de la Dactiloscopia".

Juan Evangelista Purkinje, llamado por Locard "padre de la Dactiloscopia", nació en el año de 1787 en Leitmeritz, Bohemia. En 1823, cuando desempeñaba las funciones de catedrático de Anatomía y Fisiología en la Universidad de Breslau, dio a conocer una tesis en latín, de cuyo largo epígrafe sólo nos interesan las últimas palabras: *Systematis cutanei*. Purkinje no sólo reclamaba la atención acerca de la diversidad de las huellas dactilares, sino que fue el primer europeo que creó un sistema para clasificarlas.

El sistema dividía las impresiones dactilares en nueve tipos fundamentales, algunos de los cuales continúan vigentes.

Purkinje era una personalidad ilustre en el mundo científico, pero durante toda su vida no logró despertar el interés de sus contemporáneos por sus obras dactiloscópicas, por lo que sus investigaciones se olvidaron, tanto así que cuando los que continuaron investigando sobre esta ciencia buscaron ejemplares de su tesis, se conjetura que sólo existían tres, uno de los cuales fue hallado por Francis Galton que tradujo la tesis e hizo alusión de los párrafos que le interesaban en su obra *Finger prints*, publicada en 1892.

William James Herschel, jefe administrativo británico del distrito de Hoogly en Bengala, India, se interesó en las huellas dactilares en el año 1858.

Al continuar la costumbre oriental de estampar la huella del pulgar o de otro dedo en los recibos y contratos, enriqueció su colección de huellas dactilares y así descubrió las características de éstas al observar que ninguno de los individuos que habían impreso sus huellas tenía el mismo modelo de líneas en sus dedos. Aplicó un término obtenido de un volumen de Anatomía y llamó al relieve de estos modelos "líneas papilares". También descubrió que aún después de veintiocho años de intervalo, los modelos de estas impresiones permanecían inalteradas y por lo tanto reconoció que estas observaciones podían aplicarse en el campo de la criminalística.

En el año de 1878 escribió al director de prisiones de Bengala recomendando el uso de las huellas digitales como un medio efectivo y preciso para identificar a los reclusos en las instituciones penales, pero no se le prestó atención a esta sugerencia.

En 1880 Henry Faulds, médico escocés, pasó a formar parte del personal facultativo del Hospital Tsukiji de Tokio, Japón, en donde se interesó en principio por los diversos modelos de impresiones digitales como determinantes de tipos raciales, luego su atención se desvió hacia la identificación criminalística después de que su ayuda fue requerida por la policía japonesa para resolver un robo cometido por escalamiento y para ello se necesitaba cotejar las impresiones digitales de un sospechoso con las huellas encontradas en la escena del delito.

Al profundizar en su investigación, en 1880 Henry Faulds hizo un importante hallazgo: descubrió que las glándulas sudoríparas y las secreciones aceitosas de la epidermis pueden dejar una huella tan clara como si la mano hubiese sido cubierta con tinta u hollín.

Considerando que tenía en su poder los conocimientos suficientes para revolucionar la investigación criminalística (identificación de un sospechoso por medio de las huellas dactilares dejadas en la escena del delito), le propuso sus conocimientos acerca de la ciencia de las huellas dactilares al Secretario del Interior británico y al comisionado policíaco de la nueva Scotland Yard, pero, como en el caso de Herschel, su propuesta no fue acogida.

En el año de 1888 el notable antropólogo británico Francis Galton adoptó los materiales que Herschel logró reunir durante sus investigaciones y en los cuales hacía prácticamente una demostración por medio de dos impresiones de su dedo índice derecho que fueron tomadas con 28 años de diferencia. Con ayuda de los mismos, Galton pudo confirmar científicamente lo que hasta entonces eran hipótesis sobre la perennidad, la inmutabilidad y diversidad de los dibujos papilares, dejando establecido tres principios antes de proponer el empleo de dactiloscopia en investigaciones criminales o de cualquier otra clase.

- Precisó que las crestas papilares se forman a partir del sexto mes de la vida intra-uterina y desde ese momento el dibujo dactilar es *perenne* a través de toda la existencia del ser humano.
- Patentizó que los dibujos dactilares son *inmutables* porque nacen con el individuo y no cambian a lo largo de la vida, al extremo de que ni por propia voluntad, ni por circunstancias patológicas o traumatismos se modifican. El dibujo dactilar no desaparece mientras no haya sufrido una lesión o quemadura que afecte profundamente a la dermis.
- Demostró matemáticamente que las huellas dactilares son *diversiformes* y que no pueden encontrarse dos semejantes ni en una serie de sesenta y cuatro mil millones. Por otra parte, la práctica diaria de los servicios de identidad comprueba que dos huellas procedentes de sujetos diferentes jamás corren el peligro de confundirse; podrá encontrarse una similitud de aspecto general, pero hay siempre un grandísimo número de puntos característicos que las diferencian.
- Enriqueció el acervo dactiloscópico al aplicar su clasificación formada por cuarenta y un tipos diferentes y fue el que inventó la línea *"delto-central o galtoniana "* de la que nos servimos para la cuenta de crestas papilares de las presillas interna y externa.

Juan Vucetich Kovacevich nació en Lezina, Dalmacia, Austria-Hungría el 20 de julio de 1858. Tras de emigrar con toda su familia a la República Argentina adquirió la ciudadanía de este país y el día 15 de noviembre de 1888 ingresó a la policía de la provincia de Buenos Aires en el Departamento Central de la Plata, en calidad de meritorio, siendo designado para prestar servicios a las órdenes de Ernesto M. Boero en la Oficina de Contaduría y Mayoría, siendo en aquel entonces director de policía Carlos J. Costa. El primero de mayo del año siguiente pasó a la Oficina de Estadística y el día 26 de septiembre de este mismo año ascendió a director de ésta, dando inicio al estudio de un proyecto de reorganización que tiempo después fue aprobado por la jefatura y puesto en vigencia el primero de enero de 1890.

Por iniciativa suya comenzó a publicarse en 1891 el boletín de Estadística y en junio del mismo año el director de policía Guillermo J. Nunes le encomendó un estudio para establecer el servicio de identificación antropométrico.

Pocos días después del mencionado encargo, el director de policía le entregó un ejemplar de la "*Revue Scientifique*" donde aparecía un artículo de H. De Varigny en el cual se resumían las conclusiones del antropólogo Francis Galton respecto a los caracteres y el valor identificativo de las impresiones dactilares.

Vucetich que había estudiado la Antropometría y comprobado su falta de exactitud y convencimiento, advirtió en las impresiones dactilares la solución del problema identificativo y comprendió que con ellas se presentaba un vasto horizonte al porvenir de la técnica policial y a la protección y seguridad de la personalidad humana.

En agosto de 1891 se aprobó el proyecto en el que se establecía el servicio identificativo en la forma proyectada por Juan Vucetich y el día primero de septiembre del mismo se inauguró la Oficina de Identificación quedando establecidos los dos métodos: el de Antropometría y el de Ignofalangometría.

En la ya referida Oficina de Estadística ese día primero se tomaron las diez impresiones digitales, pero se clasificaban solamente tres impresiones con la clasificación propuesta por Galton que contaba 41 tipos diferentes y se archivaba de acuerdo con esta clasificación. También hay que mencionar que posteriormente Feré introdujo ligeras variantes y con ellas aumentó a 46 el número de tipos.

Así comenzó la aplicación práctica de la identificación dactiloscópica, gracias a los ensayos de Galton, quien los realizó a su vez fundándose en la experiencia empírica de Herschel y por sugerencia de Faulds.

Más tarde, a iniciativa del doctor Francisco Latzina, se diseñó un sistema enteramente nuclear o sea que lo precisa preferentemente en el núcleo y de acuerdo con el dibujo del mismo de designa un número que ya en conjunto las impresiones de los diez dedos forman una nomenclatura con la que se organizan los archivos de donde comienza la investigación criminal con muchas impresiones dactilares hasta llegar a la escena del delito.

El conjunto de estas impresiones con sus respectivos números que forman la clasificación, y que están impresas de pulgar a meñique mano derecha y de pulgar a meñique mano izquierda respectivamente, le sirvió para formar o crear la primera cédula de identidad conocida en todo el mundo con el nombre de: *Individual Dactiloscópica o Ficha Decadactilar.*

De este conjunto se derivan todos los sistemas decadactilares en vigor. Entre ellos el español con ligeras modificaciones realizadas por Federico Oloriz Aguilera, catedrático de Anatomía de la Universidad de Madrid, y más tarde modificado por Victoriano Mora Ruiz.

Oloriz aún adoptando el sistema dactiloscópico de Vucetich como base para su sistema, las modificaciones que hace es cambiar de nombre a los cuatro tipos de Henry.

Además de este cambio de nombres, clasificó los deltas en dos clases: A los que denominamos hundidos o blanco los clasificó en abiertos y cerrados y a los salientes, negros o en trípode, así como también en cortos y largos.

Edward Richard Henry, creador del segundo gran sistema de clasificación decadactilar, era funcionario de policía en Bengala, India, donde como suplente de William James Herschel se distinguió por sus condiciones de investigador. Reemplazó a su jefe en el cargo y en la época en que se estaba usando un sistema rudimentario para la toma y clasificación de las impresiones digitales, se dio a la tarea de establecer un sistema útil de clasificación de las precitadas impresiones digitales.

El sistema creado por Henry está establecido en los trabajos realizados por Galton y Herschel y es conocido también por sistema Galton-Henry o sistema Bengalés.

Henry consideró que para la clasificación se podría agrupar los dibujos dactilares en cuatro tipos básicos, para lo cual tuvo en cuenta la existencia ó ausencia de puntos fijos que él llamó "delta y corazón".

Observemos ahora qué concepto tenía acerca de lo que él llamó "delta" o "término externo".

El delta puede formarse:

a) Por la bifurcación de una cresta.
b) Por la aproximación de las directrices de los tres sistemas crestales.
c) Cuando se observan varias bifurcaciones, la más cercana al corazón es tomada como delta.

En la figura 1.1 se muestra una huella dactilar con presencia de cores y deltas.

Figura 1.1. Puntos característicos cores y deltas.

Benjamín A. Martínez, quien fue fundador del Servicio de Identificación Dactiloscópica de la Policía Judicial Militar, del Servicio de Identificación Dactiloscópica de la Policía de México, del Laboratorio de Investigación del Crimen y del Servicio de Identificación del Ejército Mexicano, introdujo ligeras modificaciones al sistema Vucetich, en donde a los deltas blancos los subclasificó en abiertos y cerrados, y los negros en cortos y largos. Adicionó los centros nucleares de las presillas, aunque en forma limitada, así como también la formación de los deltas. Las normas del conteo de crestas papilares en las presillas y el trazo en los verticilos para la subclasificación de los mismos, así como la determinación de los centros nucleares fueron tomados del sistema Henry-Galton.

La clasificación primaria se verificó en base al sistema Vucetich y la subclasificación por el sistema Henry que hasta la fecha continúa vigente.

Por otro lado, las propiedades de Galton son detalles formados por los cruces y finales de las crestas, y son comúnmente conocidos en la literatura como *minutiaes* (minucias). En su trabajo, Galton definió cuatro características, entre las cuales se encuentran los llamados *encierros.* Esta última característica se muestra en la figura 1.2.

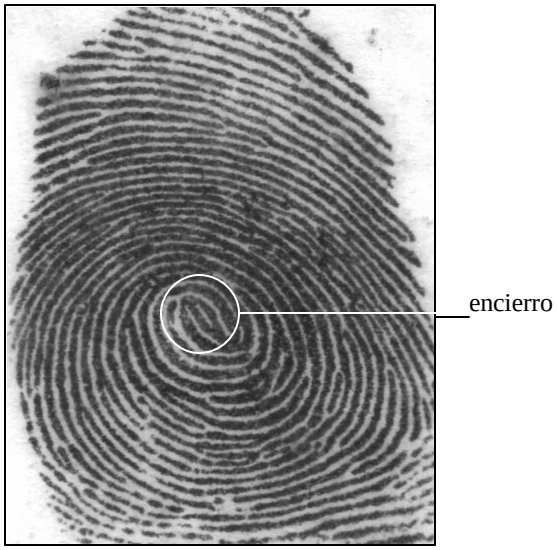


Figura 1.2. Huella dactilar con presencia de encierro.

Como puede observarse, existen varias extensiones de las características originales de Galton, pero muchas de ellas no se utilizan en sistemas automáticos de identificación. En cambio, y de acuerdo con la representación de huellas dactilares utilizada por el FBI [13], los *finales* de crestas y bifurcaciones son considerados como propiedades distintivas de las huellas. En este método, la ubicación y el ángulo de la característica se toman para representar la huella y se utilizan en el proceso de comparación.

La identificación manual de huellas es tediosa y consume mucho tiempo, sobre todo cuando es necesario realizar la clasificación en una colección muy grande de personas. Por esta razón, en 1960, la Oficina de Investigaciones del Reino Unido y el Departamento de Policía de París iniciaron estudios en sistemas de identificación automática de huellas dactilares. Para ello, básicamente asumieron la clasificación dado por Henry, las características de Galton (entre las que se encuentran los llamados encierros) y la existencia de dos tipos especiales de características llamadas *deltas* y *cores*.

En el presente trabajo, luego de digitalizar la huella dactilar por medio de un escáner de cama plana, se define un campo vectorial sobre la imagen correspondiente con el fin de obtener la respectiva imagen binaria; al mismo tiempo que las crestas y valles de la huella son considerados como curvas integrales de tal campo vectorial. Con esto, se puede aplicar la herramienta teórica de sistemas dinámicos para detectar puntos singulares del campo vectorial; mismos que, en nuestro caso, corresponden con los puntos *deltas* y *cores.*

Por otra parte, utilizando topología digital y combinatoria, calculamos también el número de *encierros*.

Capítulo II
Fundamentos teóricos

2.1. Adquisición de la imagen.

En esta sección se describe cómo se adquiere la imagen hasta tenerla en forma de una matriz de intensidades.

El método más común de capturar una imagen de huella dactilar es obtener una impresión de ella y luego escanearla para lograr su digitalización. Pero, este método puede dar como resultado una imagen bastante distorsionada, lo que hace necesaria la participación de un experto en dactiloscopia. Otra manera comúnmente utilizada para capturar imágenes de huellas dactilares es escanear la imagen directamente por medio de una cámara CCD [24], obteniendo mejores imágenes, salvo por la resequedad o alguna enfermedad de la piel, el sudor, mugre o humedad.

El dispositivo que se utilizó en este trabajo para obtener la imagen fue un escáner de cama plana. La imagen se capturó en escala de grises (8 bits), con resolución de 300 dpi, y tamaño máximo de 350x350 pixeles. Posteriormente se guarda la imagen en un archivo de formato PCX, JPG ó TIF.

El lenguaje utilizado para programar los algoritmos y desplegar la imagen en la pantalla del monitor fue Microsoft Visual Basic, versión 6.

Para obtener la información de las imágenes se utilizó la librería VIC32, la cual contiene funciones que obtienen el ancho y alto de la imagen, así como los tonos de gris de cada pixel. Dicha librería lee y crea archivos de imagen de formatos PCX, JPG, TIF, BMP, GIF, TGA y PNG.

El programa primero deberá identificar que el formato de la imagen sea adecuado; luego leerá y mostrará en pantalla la información general de dicha imagen, tales como el ancho y alto en pixeles, versión, paleta y bits de resolución. Enseguida se carga la imagen en memoria y se obtienen sus tonos de gris mediante la función GETPIXELCOLOR, en donde los valores se van asignando al mismo tiempo a una matriz para su procesamiento digital, la cual es representada como una matriz $I = (I_{ij})$ que contiene en cada una de sus entradas el valor correspondiente al nivel de intensidad de gris presente en la imagen original en el mismo punto. Estos valores son tomados del conjunto {0, 1, 2, ..., 254, 255}. Luego, esta misma imagen es mostrada en pantalla a través de la función VIEWIMAGE (Fig. 2.1).

254 254 254 254 254 254 254 254 254 243 241 246 254 254 254 254 246 241 204 207 217 254 254 254
254 254 254 254 225 254 254 246 254 254 254 254 254 254 251 254 238 215 220 248 254 254 254 254 254
254 178 254 254 254 246 254 254 254 254 254 254 241 251 254 254 254 254 254 254 254 254 137 176
132 49 75 124 196 254 254 254 238 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254
254 254 254 254 254 254 254 254 254 254 254 251 248 254 254 254 254 254 254 254 254 254 254 254
254 254 254 254 254 145 186 158 173 69 196 0 0 0 0 25 0 7 155 204 254 254 254 254 254 254 254 88
121 127 228 254 95 254 254 248 171 98 0 59 254 254 254 254 254 254 254 155 59 57 181 225 254 254 254
254 142 204 142 241 254 254 254 254 171 64 88 106 145 228 254 254 254 254 254 207 0 0 108 254 241
246 254 254 254 207 212 101 152 145 230 254 207 254 254 254 254 254 254 254 254 254 254 254 254
254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254
254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254 254
254 254 254 254 254 254 220 106 54 90 254 254 254 254 254 254 254 254 254 254 254 254 254 222 64 31
127 132 222 254 238 217 222 254 176 222 251 254 254 254 254 254 254 254 254 152 51 254 254 254
254 254 254 254 254 254 176 168 215 254 254 254 254 254 254 254 254 202 254 254 248 238 254 254
254 254 254 254 246 254 217 254 254 254 254 251 254 254 254 215 119 139 228 254 254 254 158 222
199 67 168 204 186 196 217 191 158 95 155 103 168 145 181 230 238 251 254 254 254 254 254 254

a) b)

Figura 2.1. (a) Imagen original, (b) Parte de la matriz de intensidades de la imagen (a).

2.2. Campos vectoriales

Para clasificar una huella en algún tipo de patrón que indique su forma o el sentido de sus crestas o valles, vamos a considerar estas y estos como curvas integrales de un campo vectorial adecuado, para luego, en base al índice de Poincaré localizar los puntos singulares de dicho campo; los que en nuestro caso serán *deltas* y *cores*.

2.2.1. Campos vectoriales en general [20].

El primer trabajo que desarrolló Poincaré dentro del campo de la Topología iba en búsqueda de resolver sistemas de ecuaciones diferenciales. Desde entonces, la aplicación en las ecuaciones diferenciales han jugado un papel importante en Topología, especialmente en Topología Algebraica y Topología Combinatoria. Podrá parecer extraño que temas como Topología y Ecuaciones Diferenciales pudieran estar relacionados, sin embargo en Matemáticas tal tipo de relaciones son comunes. Un estudio desarrollado entre ellos es el concepto de campo vectorial. Un campo vectorial sobre un subconjunto D en el plano es una función que asigna a cada punto P de D un vector en el plano. Intuitivamente, podríamos imaginar a V como algo que nos da la velocidad de alguna sustancia que se presenta en D en algún estado de agitación, como el agua al correr por un tubo.

Colocar al vector $V(P)$ con su inicio en P es precisamente lo que dificulta la visualización. Dado que para los vectores comunes las cualidades que nos interesan son su longitud y dirección, para efectos prácticos, se acostumbra hacer coincidir el inicio del vector en el origen. Así, $V(P)$ puede ser descrito por las coordenadas de su cabeza:

$$V(P) = (F(x,y),G(x,y)) \qquad (1)$$

donde F y G son funciones reales de $P = (x,y)$. El campo vectorial se denomina continuo cuando la función (1) es continua. Si definimos $¦(P)$ como el punto en la cabeza del vector $V(P)$ cuando se coloca su inicio en P, obtenemos una transformación en D, el vector suma de P y $V(P)$ (fig. 2.2):

$$¦(P) = P + V(P) = (x + F(x,y), y + G(x,y)) \qquad (2)$$

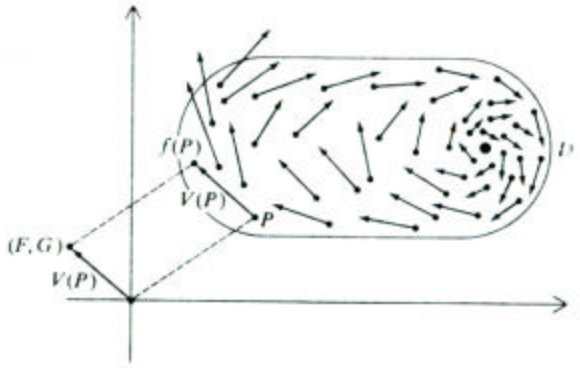


Figura 2.2. Definición de un campo vectorial sobre un subconjunto *D* del plano.

De acuerdo a un teorema importante de campos vectoriales, según el cual la suma de transformaciones continuas es continua, se tiene que ¦ será continua si *V* es un campo vectorial continuo. Recíprocamente, suponiendo que una transformación ¦ está definida en *D*, entonces el campo vectorial *V* puede obtenerse definiendo *V(P)* como el vector del punto *P* al punto ¦ *(P)*. El campo vectorial será entonces continuo si la transformación ¦ es continua.

Los campos vectoriales tienen muchas aplicaciones importantes. Los campos de fuerza de la gravedad y electromagnetismo son campos vectoriales; los vectores de velocidad de un fluido en movimiento, como en la atmósfera (vectores de viento), forman un campo vectorial; y gradientes como el gradiente de presión en un mapa climático o el gradiente de altura en un cartógrafo son campos vectoriales. Estos ejemplos son estudiados usualmente desde el punto de vista de las ecuaciones diferenciales.

Un campo vectorial (1) determina un sistema de ecuaciones diferenciales *x* y *y*. Estas variables se toman para representar la posición de un punto en movimiento en el plano dependiendo en una tercera variable, el tiempo *t*. El sistema de ecuaciones diferenciales toma la forma

$$x' = F(x, y)$$
$$y' = G(x, y) \qquad (3)$$

donde la diferenciación es con respecto a *t*. Tal sistema es llamado "autónomo" porque los dos lados derechos son independientes del tiempo.

Los sistemas autónomos son de particular interés, porque las leyes fundamentales de la naturaleza pueden ser consideradas como independientes del tiempo. Una solución del sistema (3) consiste de dos funciones expresando *x* y *y* en términos de *t*. Esto puede ser considerado como la ecuación paramétrica de una trayectoria en el plano: la trayectoria de una molécula de gas o líquido, la órbita de un planeta o un electrón. El campo vectorial original *V(P)* nos da el vector tangente a la trayectoria de movimiento en el punto *P=(x,y)*.

Ejemplos.

1. Consideremos el campo vectorial $V(x,y)=(2,1)$. Todos los vectores son iguales, entonces la transformación correspondiente, $f(x,y)=(x+2,y+1)$, es una traslación. El correspondiente sistema de ecuaciones diferenciales, $x'=2$, $y'=1$, tiene solución $x = 2t + h$, $y = t + k$, las trayectorias solución (fig. 2.3a) son la familia de líneas rectas con pendiente ½.
2. Consideremos el campo vectorial $V(x,y)=(-y, x)$. Algunos de los vectores se muestran en la figura 2.3b. La transformación correspondiente $f(x,y)=(x - y, x + y)$ es una rotación de $45°$ en cuanto a las manecillas del reloj combinada con un extendimiento desde el origen por un factor de $\sqrt{2}$. El sistema de ecuaciones diferenciales, $x' = -y$, $y' = x$, tiene la solución $x = k \cos(t)$, $y = k \operatorname{sen}(t)$. Las trayectorias solución son los círculos centrados en el origen. Ellas son tangentes en cada punto al campo vectorial V.

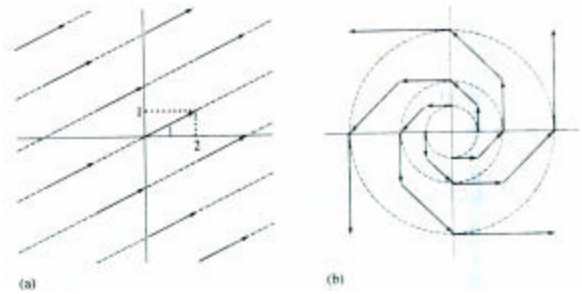


Figura 2.3. Las trayectorias son tangentes al campo vectorial

Al momento de obtener las soluciones de una ecuación diferencial y realizar el "retrato de fases" de las soluciones particulares de esta ecuación diferencial, Poincaré descubrió que la forma de este retrato de fases era determinado por puntos excepcionales P, llamados puntos críticos, donde $V(P) = 0$.

Dentro de los campos vectoriales, nuestro interés se centra en el estudio de curvas cerradas, específicamente lo referente al comportamiento de los vectores que aparecen al seguir una trayectoria sobre la curva, estudiando la dirección que van llevando los vectores hasta completar un recorrido.

Números sinuosos.

Considérese un campo vectorial continuo V sobre una curva cerrada γ. Supóngase que V nunca es cero sobre la curva. Comenzando en un punto fijo Q, imaginemos un punto que se mueve sobre la curva en dirección opuesta a las manecillas del reloj hasta volver a Q. El vector $V(\boldsymbol{P})$, que es en el que estamos interesados, comenzando en $\boldsymbol{Q}$, se moverá durante el trayecto sobre γ regresando a la posición $V(\boldsymbol{Q})$. Durante el recorrido, $V(P)$ realizará giros completos. Contando estas vueltas positivas si son en sentido contrario a las manecillas del reloj y negativas si son en el mismo sentido que ellas, el resultado que es la suma algebraica del número de vueltas es llamado número sinuoso de V en γ y se denota como $W(\gamma)$. Este

número, llamado ***índice de Poincaré*** será importante en nuestros estudios posteriores. Aunque para campos vectoriales continuos, este número puede calcularse como la integral de la razón de cambio de orientación en un contorno cerrado**;** nosotros basaremos nuestro trabajo en un método alternativo desarrollado por el mismo Poincaré y que consiste en fijarse sólo en una dirección particular y se registra únicamente el número de veces que el vector *V(P)* apunta en esa dirección. Si el vector *V(P)* alcanza esa dirección moviéndose en dirección opuesta a las manecillas del reloj contamos **+1**; si alcanza la dirección moviéndose en el mismo sentido que las manecillas del reloj entonces contabilizamos **–1**, y contamos **0** si el vector llega a la dirección establecida pero inmediatamente regresa al camino por el que venía (fig. 2.4).

Para el propósito de ilustrar tal método, consideremos sólo las tres direcciones mostradas en la figura 2.4a.

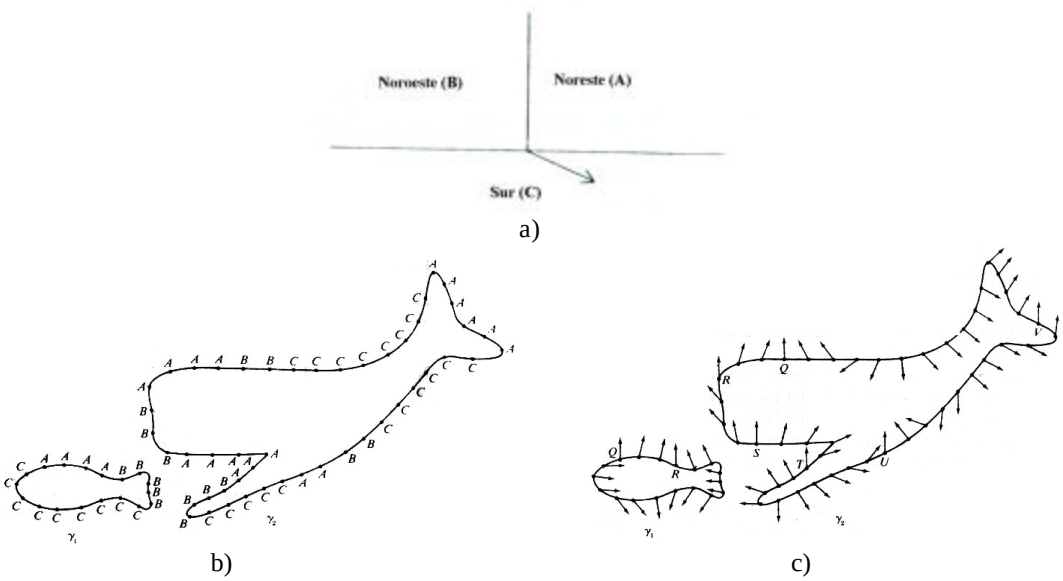


Figura 2.4. (a) Direcciones para calcular el índice de Poincaré de la figura (b,c). (b) algunos puntos de las curvas cerradas se han etiquetado con las direcciones que se les asignará de acuerdo a la figura (a). (c) Vectores asignados.

En la figura 2.4c los lugares han sido marcados donde el vector *V(P)* apunta hacia el norte. En la curva g_1 esto ocurre en *Q* y *R*. En *R* el vector *V(P)* llega a la posición vertical en sentido de las manecillas del reloj, mientras en *Q* se aproxima a la vertical desde la derecha pero revierte la dirección en *Q* y regresa al mismo lado. Contando **–1** en *R*, **0** en *Q*, el ***índice de Poincaré es* –1**. En g_2 son seis veces las que el vector *V(P)*: *R, T,* y *U* cuentan como **+1**, *Q* y *S* cuentan como **–1**, y *V* cuenta como **0;** así, el ***índice de Poincaré es* +1.**

Una definición equivalente para el **índice de Poincaré,** es la siguiente:

"Si $D \subset \Re^2$ es un conjunto abierto, acotado y simplemente conexo, $V : \overline{D} \to \Re^2$, con $V(u,v) \in C^2(D) \cap C(\overline{D})$ y $\Gamma \subset D$ es una curva de Jordan que circunda al punto **O,** entonces suponiendo que $V(x) \neq 0$ para todo $x \in \Gamma$, el número entero

$$I(V,\Gamma,0) = \frac{1}{2\pi}\int_{\Gamma} d\theta$$

determina el número de vueltas que V da alrededor del punto **O** y se denomina el **índice de Poincaré** de V sobre Γ con respecto a **O**" (fig. 2.5) .

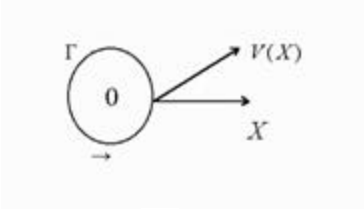


Figura 2.5. Curva Γ alrededor del punto **O**, sobre la cual se calcula el índice de Poincaré.

Esta es la versión que será asumida en este trabajo para la detección de candidatos a ser cores o deltas en una huella dactilar. En la sección 3.6. se presentan los resultados obtenidos.

2.2.2. Campos vectoriales en huellas dactilares [25].

Con el fin de extraer información direccional a partir de las tonalidades de gris de cada punto, es decir, determinar el sentido o circulación de las líneas (crestas y valles) alrededor de cada punto de la imagen, se construye una transformación que asigne a cada punto su dirección; para ello se escogerán primero las direcciones posibles de cada punto y el tamaño y forma de la región alrededor del punto que se tendrá en cuenta.

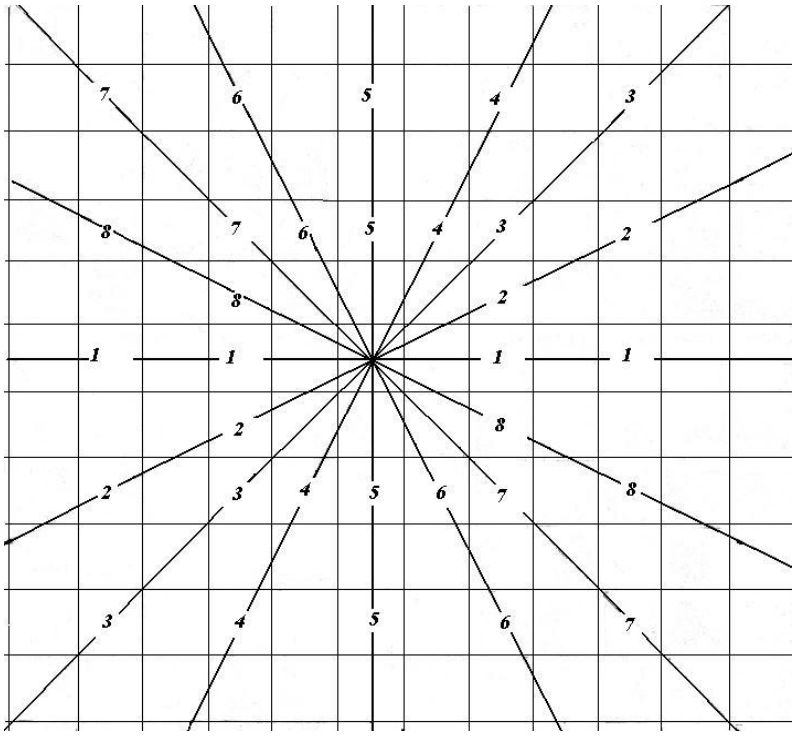
Sobre la matriz I[i,j] se definen las vecindades de radio *N* de un punto [i,j], como los conjuntos *R*[i,j] :

$$R_N\,[i,j] = \{\, a[k, l] : | k - i | \leq N, | l - j | \leq N\}$$

La selección del radio de las vecindades usado para el procesamiento es muy importante ya que se debe garantizar que en la vecindad de cada punto haya suficiente información acerca de la dirección de las líneas vecinas. Es por esto que *N* depende del grosor de las crestas y valles de la huella, lo cual es un parámetro muy ligado con el método de adquisición y su resolución. En nuestro caso, hemos elegido *N = 4* (fig. 2.6a), así como ocho direcciones (fig. 2.6b).

7		6		5		4		3
8		7	6	5	4	3		2
		8				2		
1		1		**P**		1		1
		2				8		
2		3	4	5	6	7		8
3		4		5		6		7

a)



b)

Figura 2.6. (a) Vecindad de 9x9 alrededor de cada punto.
(b) Direcciones para definir el campo vectorial sobre la huella dactilar.

Si tenemos M posibles direcciones numeradas de 0 a M – 1, y definimos como dirección M, la que se presenta en regiones con alto nivel de ruido o en regiones muy uniformes donde no se pueda estimar la dirección, el conjunto de direcciones posibles está dado por:

$$S = \{ 0, 1, 2, 3, \ldots, M\}.$$

Las direcciones se escogen como se muestran en la figura 2.6b y el ángulo correspondiente a cada dirección está dado en la tabla de la figura 2.7.

Dirección	Ángulo
1	0°
2	26.5°
3	45°
4	63.5°
5	90°
6	116.5°
7	135°
8	153.5°
9	Indefinida

Figura 2.7. Direcciones con sus correspondientes ángulos
para definir el campo vectorial en la huella dactilar

A lo largo de cada dirección y dentro de la vecindad de cada punto, se define un conjunto de puntos

$$T^k_{[i,\,j]} = \{\, [m, n] \mid [m, n] \in \mathbb{R}_N\, [i, j] \text{ y } [m, n] \text{ está en la dirección } k\}$$

En cada punto se puede definir un estimador $S^k_{[i,\,j]}$ para cada una de las direcciones posibles:

$$S^k_{[i,\,j]} = \sum_{[m,\,n] \,\in\, T^k_{[i,\,j]}} \delta(I[m, n], I[i,j]),$$

donde

$$\delta(I[i, j], I[k, l]) = |\, I[i, j] - I[k, l]|.$$

La transformación asigna al punto la dirección donde $S^k_{[i,\,j]}$ toma el valor mínimo (si existe). Es posible que $S^k_{[i,\,j]}$ sea el mismo para todo k, lo cual indicaría que el punto [i, j] está localizado en una región muy ruidosa o donde no se presentan valles o crestas. De esta forma la transformación dirección[i, j] queda definida por:

$$\text{dirección } [i, j] = \{k: S^k_{[i,\,j]} = \text{máx}\{\, S^k_{[i,\,j]} \mid m = 0, 1, \ldots, M\}\}$$

Si al calcular el máximo de las sumas $S^k_{[i,\,j]}$ alrededor de un punto, éste se presenta en exactamente dos direcciones adyacentes, se elige cualquiera de ellas; de otra forma, se le asigna la dirección indefinida.

El resultado de aplicar este algoritmo nos dará la imagen binarizada y en cada punto de la imagen su correspondiente dirección.

Para el análisis de las huellas se definieron ventanas de 9x9, donde el pixel central corresponde al pixel que se le va a determinar su dirección y su tono. En cada vecindad se definen ocho sumas direccionales S_i donde i =1, 2, 3, 4, 5, 6, 7, 8 (fig.2.6a).

Sumas direccionales

En correspondencia con el estimador direccional elegido, definimos las 8 siguientes sumas direccionales:

$S_1 = I(i, j-4) + I(i,j-2) + I(i,j+2) + I(i,j+4)$
$S_2 = I(i+2, j-4) + I(i+1,j-2) + I(i-1,j+2) + I(i-2,j+4)$
$S_3 = I(i+4, j-4) + I(i+2,j-2) + I(i-2,j+2) + I(i-4,j+4)$
$S_4 = I(i+4, j-2) + I(i+2,j-1) + I(i-2,j+1) + I(i-4,j+2)$
$S_5 = I(i+4, j) + I(i+2,j) + I(i-2,j) + I(i-4,j)$
$S_6 = I(i-4, j-2) + I(i-2,j-1) + I(i+2,j+1) + I(i+4,j+2)$
$S_7 = I(i-4, j-4) + I(i-2,j-2) + I(i+2,j+2) + I(i+4,j+4)$
$S_8 = I(i-2, j-4) + I(i-1,j-2) + I(i+1,j+2) + I(i+2,j+4)$

Obtenemos la suma direccional SUMADIR, la cual se obtiene como la suma de todas las sumas direccionales

$$SUMADIR = \Sigma S_i \quad i = 1, \ldots ,8$$

SUMADIR definirá un umbral para que a partir de ella se pueda clasificar el punto como blanco o negro. En las sumas direccionales se detecta la S_i que sea máxima y mínima.

$$S_q = S_{max}$$
$$S_p = S_{min}$$

Obtenemos S que es el valor del pixel del centro multiplicado por 4, ya que cada suma direccional es la suma de las intensidades de cuatro pixeles.

$$S = 4 * P$$

El método de definición del umbral local asigna a un pixel el color blanco (1) si la suma S es mayor que el promedio de los pixeles considerados en la ventana.

La comparación de sumas direccionales toma el promedio de los valores máximos y mínimo de las sumas direccionales como parámetro para definir el umbral.

Así, un pixel será blanco si:

$$S + S_{max} + S_{min} > 3/8\ SUMADIR$$

Mientras que, será negro si

$$S + S_{max} + S_{min} < 3/8\ SUMADIR$$

Si son iguales, podemos tomar cualquier valor. En el caso de que los valores de S_{max} o S_{min}, sean iguales, tenemos que detectar si son adyacentes los valores, si son adyacentes se toma cualquier valor de dirección, en caso contrario se asigna la dirección indefinida.

Como resultado de lo anterior, tenemos una imagen binaria (fig. 2.8) y una matriz de direcciones (fig. 2.9); es decir, una matriz con valores enteros tomados del conjunto {0, 1, 2, ..., 8, 9}.

Figura 2.8. Matriz binaria

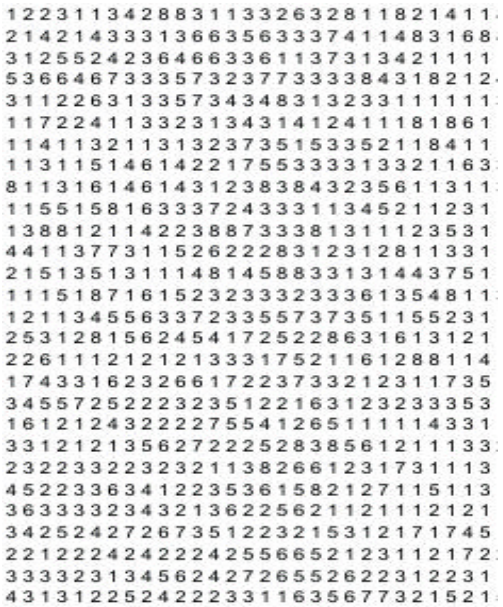


Figura 2.9. Matriz de direcciones.

Una vez que se tiene la huella dactilar en forma de una matriz binaria, el siguiente paso es asociar a cada entrada de dicha matriz, un punto de coordenadas enteras en el plano. Si la entrada es **0** el punto se denomina como *blanco*, mientras que si la entrada es **1** entonces el punto es denominado *negro*. De tal manera se obtiene una representación de la imagen binaria en términos de puntos sobre el plano discreto (fig. 2.10).

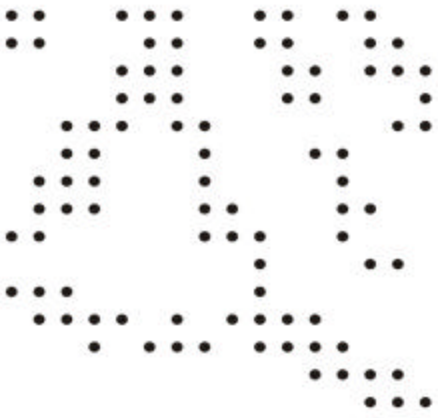


Figura 2.10. Imagen binaria con puntos lattice.

2.3. Conceptos de Topología Digital

Ahora que ya se tiene la imagen binaria en términos de puntos lattice, podemos aplicarle la herramienta de topología digital con el fin de obtener el número de encierros presentes en una huella dactilar.

En la presente sección se revisan los conceptos necesarios que serán utilizados.

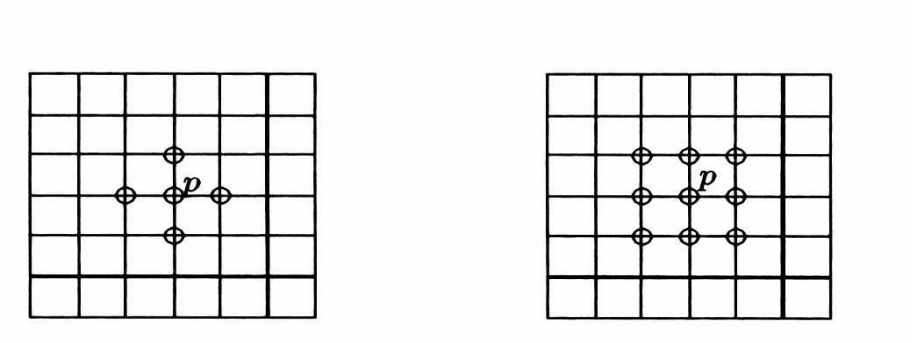
2.3.1. Relación de conexidad en un conjunto de puntos lattice [34].

Sea Z el conjunto de todos los números enteros, $Z^2 = ZxZ = \{(z_1, z_2) : z_1, z_2 \in Z\}$ el conjunto de todas las parejas ordenadas de números enteros.

Definición 2.3.1.1. Los elementos $(x, y) \in Z^2$ corresponden a los puntos con coordenadas enteras en el plano euclideano y se llaman *puntos lattice.*

Definición 2.3.1.2. Dos puntos lattice ***p, q*** $\hat{I}$ $\boldsymbol{Z^2}$ se dicen ***8-adyacentes*** si son distintos, y cada coordenada de uno de ellos, difiere en a lo más uno, de la respectiva coordenada del otro, es decir, para ***p*****=** (x_1, x_2) y ***q*** **=** (y_1, y_2) se tiene $|x_i - y_i| = 0$ **o** $|x_i - y_i| = 1$**,** i = 1, 2.

Dos puntos lattice ***p, q*** $\hat{I}$ Z^2 son *4-adyacentes* si ellos son *8-adyacentes* y difieren en a lo más una de sus coordenadas. Para n = 4 u 8, un *n – vecino* del punto ***p,*** es un punto ***q*** que es *n-adyacente* a ***p*** (fig. 2.11)**.**



(a) (b)

Figura 2.11. (a) p y sus 4-vecinos, (b) p y sus 8-vecinos

Definición 2.3.1.3. Para cualquier conjunto de puntos lattice S, $S \subseteq Z^2$, un m – *camino en S* es una sucesión $\langle p_i \mid 1 \leq i \leq n \rangle$ de puntos en S tales que p_i es m – adyacente a p_{i+1} para $1 \leq i < n - 1$. Un camino $\langle p_i \mid 1 \leq i \leq n \rangle$ es un m – camino de p_1 a p_n. Se dice que los puntos p_1 y p_n son unidos por el camino $\langle p_i \mid 1 \leq i \leq n \rangle$.

Un camino $\langle p_i \mid 1 \leq i \leq n \rangle$ es *cerrado* si $p_1 = p_n$.

Un camino $\langle p_i \mid 1 \leq i \leq n \rangle$ es *simple* si $p_i \neq p_j$ para $i \neq j$ e $(i, j) \neq (1, n)$ (un camino simple puede ser cerrado).

Un camino $\langle p_1 \rangle$ es un caso especial de camino cerrado, que consiste del único punto p_1.

Definición 2.3.1.4. Se dice que dos puntos p y q del conjunto S están en *relación* de m – *conexidad* R_m si, y sólo si p y q pueden ser unidos por un m – camino en S.

La relación R_m así definida es de equivalencia sobre S. Las clases de equivalencia inducidas se llaman *m-componentes*. Se dice que S es *m-conexo* si el conjunto cociente S / R_m tiene una sola *m*-componente.

Definición 2.3.1.5. Una imagen digital P es una tetrada $P = (Z^2, m, n, B)$, donde $B \subseteq Z^2$, con $(m, n) = (8, 4)$ o $(4, 8)$. Los puntos de B se llaman puntos negros, mientras que los puntos de $Z^2 \setminus B$ se llaman puntos blancos de la imagen, y constituyen el *fondo* de la misma. Cuando B es un conjunto finito la imagen P se denomina finita.

En lugar de Z^2, vamos a considerar un subconjunto finito $S \subset Z^2$, el cual contiene a todas las imágenes. En la practica, S puede considerarse como la colección de todos los puntos de la pantalla de un monitor. De tal manera que, nos referimos a la imagen $P = (S, m, n, B)$.

Definición 2.3.1.6. Dos puntos negros en una imagen digital $P = (S, m, n, B)$, se denominan *adyacentes* si ellos son m – adyacentes, y dos puntos blancos o bien un punto blanco y uno negro se dicen adyacentes si ellos son n – adyacentes. Una imagen digital (S, m, n, B), también se llamará una (m, n) imagen digital (fig. 2.12).

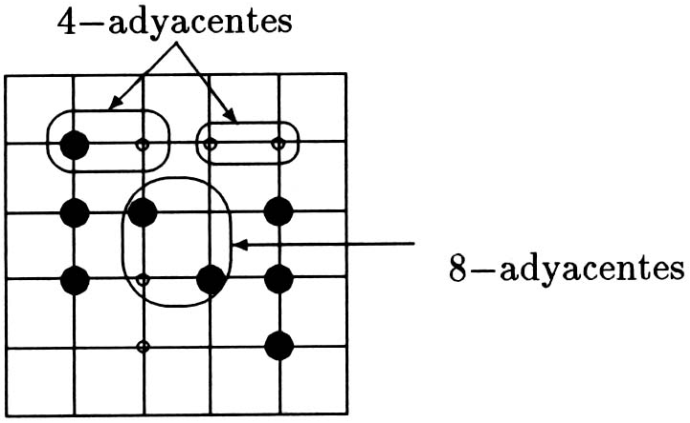
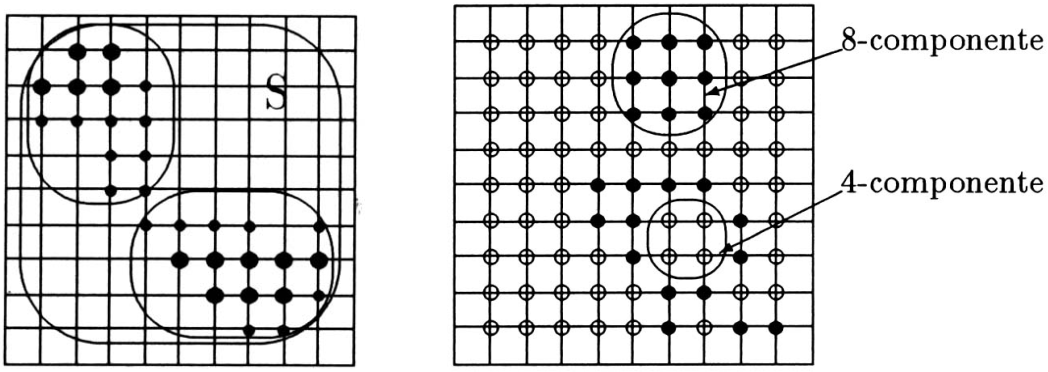


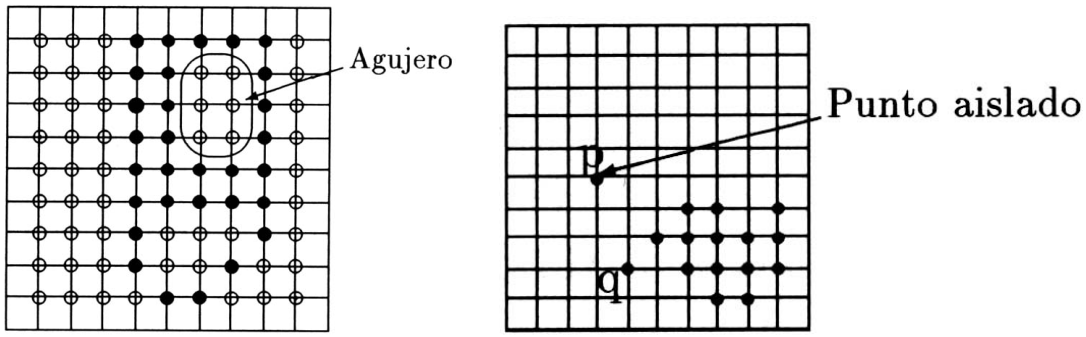
Figura 2.12. Adyacencia entre puntos.

Definición 2.3.1.7. Un conjunto *S* de puntos negros (blancos) en una imagen digital (*m*, *n*) es conexo si S es *m* – conexo (resp. *n* – conexo) (fig. 2.13a). En una imagen digital (*m*, *n*), una componente de un conjunto *S* de puntos negros es una *m* – componente, mientras que una componente de un conjunto de puntos blancos es una *n* – componente de este conjunto. Una componente del conjunto de todos los puntos negros de una imagen digital se llama *componente negra*, y una componente del conjunto de puntos blancos se llama *componente blanca* (fig. 2.13b). Las componentes blancas finitas de *P* se llaman *agujeros* (fig. 2.13c). Un punto negro se llama *aislado* si no es adyacente a algún otro punto negro (fig. 2.13d).

Nota. Esta definición es de particular importancia para nuestro trabajo, ya que los *agujeros* de una imagen digital, serán los *encierros* de una huella dactilar, cuyo número será calculado utilizando directamente el concepto de punto aislado.



a) b)



c) d)

Figura 2.13. (a) El conjunto S de puntos negros es conexo.
(b) Componentes de una imagen. (c) Agujero en una imagen.
(d) Punto aislado *p*.

2.3.2. Realización geométrica de una imagen digital [1-2].

En primer lugar, indicaremos la forma de relacionar los conceptos de la topología digital en términos de puntos lattice, con sus análogos en términos de pixeles.

A cada punto lattice $p \in Z^2$ le asignamos un *complejo cúbico 2 – dimensional* consistente de:

- La única *2-celda* (cuadrado unitario centrado en el punto *p*, cuyos lados son paralelos a los ejes coordenados);
- Cuatro *1-celdas* a_1, a_2, a_3, a_4 llamadas aristas;
- Cuatro *0-celdas* v_1, v_2, v_3, v_4 llamadas vértices.

Este complejo se llamará *pixel* .

Notación: $v = |p|$.

El pixel *v* será llamado la realización geométrica del punto lattice *p* (fig. 2.14).

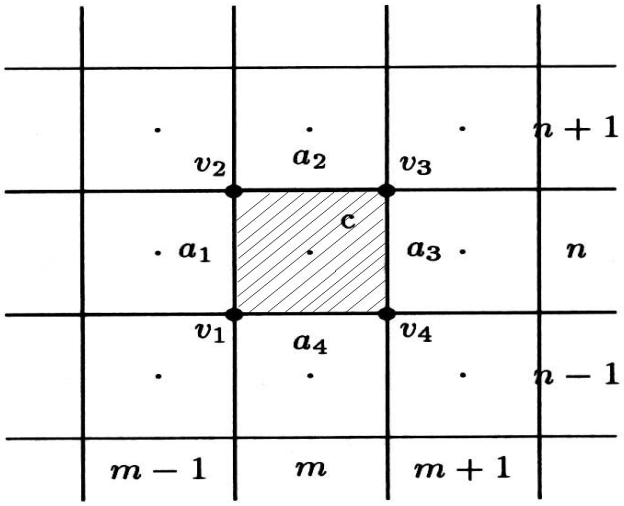


Figura 2.14. Pixel v

$v = \{ v_1, v_2, v_3, v_4, a_1, a_2, a_3, a_4, c\}$

Bajo esta suposición, dos pixeles que corresponden a puntos lattice 4-adyacentes tienen dos vértices comunes y una arista común. Dos pixeles que corresponden a puntos lattice 8-adyacentes tienen al menos un vértice común.

Sea $|Z^2|$ la unión de todos los pixeles como complejos:

$$|Z^2| = \cup \{ |p| : p \in Z^2 \}.$$

Definición 2.3.2.1. Cualquier conjunto $K \subset |Z^2|$ lo llamaremos *complejo cúbico*.

Si $L \subset K$, decimos que L es un *subcomplejo* de K. El complejo $\overline{K} = |Z^2| \setminus K$ se llama *complejo complemento* de K.

En virtud de esta última definición, la imagen binaria mostrada en la figura 2.10, ahora tiene otra representación, siendo ésta un complejo cúbico (fig. 2.15).

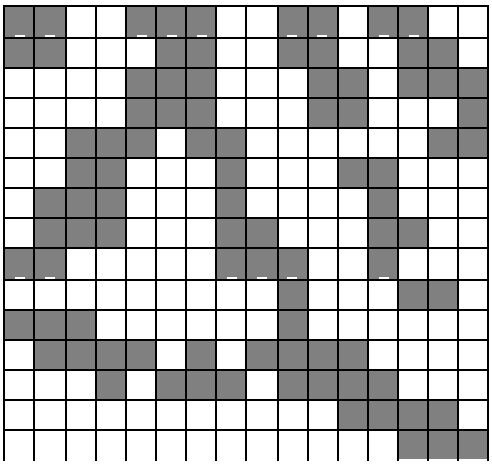


Figura 2.15. Imagen binaria como complejo cúbico.

Definición 2.3.2.2. La celda s será llamada *una cara* de la celda s', si $s \subset s'$. Si además $s \neq s'$, diremos que s es una *cara propia*. La celda s es *cara* (cara propia) del *complejo* K si s es cara (cara propia) de alguna celda de K. La *frontera* δK del complejo K es el complejo que consiste de todas las celdas de K las cuales son caras de $\overline{K}$ y de todas las celdas de $\overline{K}$ las cuales son caras de K, es decir $s \in \delta K \Leftrightarrow$ (s *es cara de* K) $\wedge$ (s *es cara de* $\overline{K}$).

El complejo K se llama *cerrado* si $\delta K \subset K$.

El complejo K se llama *abierto* si el complejo complemento $\overline{K}$ es cerrado.

Por ejemplo, en la figura 2.14, las caras propias de la celda c son $v_1, v_2, v_3, v_4, a_1, a_2, a_3, a_4$; caras propias de la celda a_1 son v_1, v_2,etc..

En la figura 2.11, $\delta v = \{ v_1, v_2, v_3, v_4, a_1, a_2, a_3, a_4 \}$; $\delta\{c\} = \delta v$; $\delta\{a_1\} = \{ v_1, v_2, a_1 \}$; $\delta\{v_1\} = \{v_1\}$.

Observación.

De las definiciones anteriores se sigue que δK es cerrado y que si K y L son cerrados entonces K$\cap$L y K$\cup$L son cerrados. Además, la *clausura* $K^c = K \cup dK$ es también cerrado (por lo tanto el píxel $|p| = |p|^c$ es un complejo cerrado, para $p \in Z^2$).

Así mismo, se deduce que K es abierto si y sólo si $K \cap dK = f$. Más aún, es abierto el complejo $K^0 = K \setminus dK$, llamado *interior* de *K*.

Definición 2.3.2.3. Para cualquier conjunto $C \subset Z^2$, su *realización geométrica como un complejo cerrado es* $|C| = \cup \{ |p| : p \in C \}$, y su *realización geométrica como un complejo abierto es* $|C|^0 = |C| \setminus d|C|$.

Definición 2.3.2.4. Sea P = (S, 8, *4*, B) una imagen digital. *Realización geométrica del conjunto de puntos negros* $C \subset B$ es un complejo cúbico cerrado $|C|$. En particular, $|B|$ se llama *realización geométrica de la imagen P*.

Realización geométrica del conjunto de puntos blancos W es un complejo abierto $|W|^0 = |W| \setminus \delta |W|$.

A cada complejo $K \subset Z^2$ podemos asociarle su natural realización "poliédrica" $|K|$, i.e. su realización como subconjunto del espacio euclídeo R^2.

Definición 2.3.2.5. La realización poliédrica de cualquier complejo K es el conjunto

$$|K| = \cup \{|s| : s \in K\}.$$

La *realización poliédrica* $\|P\|$ *de la imagen* P = (Z^2, 8, 4, B) se llama al espacio cerrado $\|B\|$, y *realización poliédrica* del fondo se llama al espacio abierto $\left\|F\right|^0\Big|$.

No es difícil ver, que para la imagen P = (Z^2, 8, *4*, B), el subconjunto $\|P\| \subset R^2$ puede considerarse como un análogo continuo *C(P)* de la imagen P.

Definición 2.3.2.6. Si L es cualquier complejo cúbico 2-dimensional cerrado y finito, el número $c(L) = a_0 - a_1$ se llama *característica de Euler* de L, donde a_0 es el número de *celdas 0-dimensionales* (0-número de Betti), y a_1 es el número de *celdas 1-dimensionales* (1-número de Betti).

Una interpretación de estos números es la siguiente: en una imagen digital binaria bidimensional (*S, 8, 4, B*), a_0 es el número de componentes negras, mientras que a_1 es el número de agujeros *(encierros)* presentes en la imagen.

2.3.3. Índice de un pixel y conservación de la topología [1-2],[35].

Definición 2.3.3.1. Sea K un subcomplejo cúbico de $|Z^2|$, y $p \in Z^2$. *Indice del pixel* $v = |p|$ con respecto al complejo K, *es el número*

$$Ind_K(v) = \chi(|\delta v \cap K|);$$

En caso de que K sea cerrado, podemos usar la siguiente expresión:

$$Ind_K(v) = \chi(\delta v \cap K) = a_0 - a_1;$$

donde a_i es el número de *i*-celdas ($0 \le i \le 1$) de dv que pertenecen a K.

Si $p \in Z^2$, $v = |p|$, y K un complejo cerrado o abierto. Entonces

$$0 \leq Ind_K\ v \leq 4.$$

En la figura 2.16 se muestran algunas configuraciones para los distintos valores de Ind_K.

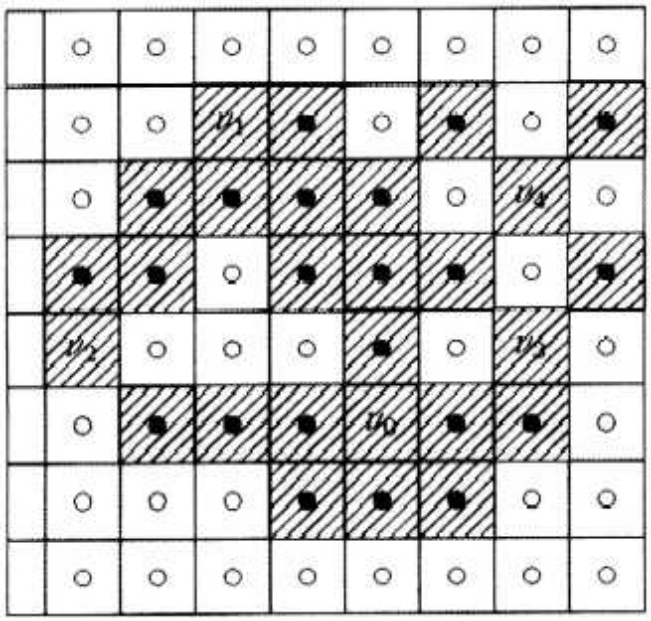


Figura 2.16. El Pixel v_i tiene índice i

Definición 2.3.3.2. Sea $P = (Z^2, 8, 4, B)$. *El Indice de un punto lattice p con respecto al subconjunto $C \subset B$ de puntos negros* es el índice del pixel $|p| = v$ con respecto al complejo cerrado $|C \setminus \{p\}|$:

$$Ind_C\, p = Ind_{|C\setminus\{p\}|}\, |p|.$$

En particular, al número $Ind_B p$ lo llamaremos *índice del punto p con respecto a la imagen P*, y en virtud de la observación que sigue a la definición 2.3.2.2., lo podemos calcular como la diferencia entre el número de vértices de $|p|$ comunes con $|B\setminus\{p\}|$, y el número de aristas de $|p|$ comunes con $|B\setminus\{p\}|$.

El concepto fundamental de la Topología digital y sus aplicaciones a problemas en Reconocimiento de Patrones y procesamiento de imágenes es el de invariancia de una transformación aplicada a la imagen. A continuación, se presenta lo que en este trabajo se ha asumido como criterio de conservación de la topología en imágenes binarias bidimensionales.

Definición 2.3.3.3. Sea $P = (Z^2, 8, 4, B)$ una imagen digital bidimensional. El punto $p \in B$ es *simple* si, y sólo si $Ind_B p = 1$. Equivalentemente, el *poliedro* $|B \setminus \{p\}|$ es un *retracto fuerte de deformación de* $|B|$.

Sabemos también que $c(L) = a_0 - a_1$, donde a_0 es el número de vértices de L, y a_1 es el número de aristas de L. Así,

$a_0 - a_1 = 1$.
$a_0 = 1$ y $a_1 = 0$ (fig. 2.17a).
$a_0 = 2$ y $a_1 = 1$ (fig. 2.17b).
$a_0 = 3$ y $a_1 = 2$ (fig. 2.17c).
$a_0 = 4$ y $a_1 = 3$ (fig. 2.17d).

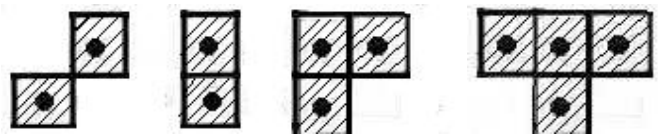


Figura 2.17. Casos posibles para Ind_B (p)=1

Definición 2.3.3.4. Diremos que la imagen $P' = (Z^2, m, n, B')$, se obtiene de una imagen $P=(Z^2, m, n, B)$ por "elementary push", si $B' = B\{p\}$ y el punto p es simple para B.

Análogamente, diremos que la imagen P se obtiene de P' por "elementary pull".
Contracción combinatoria de la imagen P se llama a una sucesión de elementary pushes

$$P > P_1 > P_2 > \ldots > P_n = P',$$

tal que P' no tiene puntos simples (fig. 3.10b,c).

La imagen P' se llama "resultado" de la contracción de la imagen P

Enseguida introducimos el concepto dual a la noción de contracción combinatoria.

En lugar de Z^2, consideraremos un subconjunto finito Q el cual contenga a todas las imágenes. En la practica Q puede considerarse como el conjunto de puntos de una pantalla.

Definición 2.3.3.5. Sea $P=(Q, m, n, B)$, una imagen digital, donde Q es un subconjunto finito de puntos. Una sucesión de elementary pulls

$$P < P_1 < P_2 < \ldots < P_n = P',$$

donde $P' = (Q, m, n, B')$, se llama expansión combinatoria si $Q - B'$ no tiene puntos simples p con respecto a $B' \cup \{p\}$.

La imagen P' se llama "resultado" de la expansión de la imagen P (fig. 3.10d).

Definición 2.3.3.6. Sean $P = (Z^2, 8, 4, B)$ y $P' = (Z^2, 8, 4, B')$. La transformación T de P a P' es una sucesión $T=\{P_0,\ldots, P_n\}$, donde $P= P_0$, y $P'= P_n$.

Si $T_1 =\{P, P_1,\ldots, P' \}$ es una transformación de P a P', y $T_2 = \{ P', P_1',\ldots, P''\}$ es una transformación de P' a P'', podemos definir una composición T_2 o T_1 de transformaciones como una transformación de P a P'':

$$T_2 \; o \; T_1 =\{P, P_1,\ldots, P', P_1',\ldots, P'' \}.$$

La composición así definida es asociativa.

Definición 2.3.3.7. Dos imágenes $P = (Z^2, 8, 4, B)$ y $P' = (Z^2, 8, 4, B')$, se llaman homotópicamente equivalentes en sentido de combinatoria $(P\sim_T P')$ si existe una transformación $P = P_0, P_1,\ldots, P_n = P'$, donde P_{i+1} es obtenido de P_i por elementary push o elementary pull.

La transformación $P = P_0, P_1,\ldots, P_n = P'$, se llama transformación homotópica en sentido de combinatoria.

Nótese que la equivalencia homotópica en sentido de combinatoria, es una relación de equivalencia en el conjunto de todas las imágenes digitales $\{(Z^2, 8, 4, B)\}$.

Decimos que las imágenes $P = (Z^2, 8, 4, B)$ y $P' = (Z^2, 8, 4, B')$, tienen el mismo tipo homotópico en sentido de combinatoria si ellas pertenecen a la misma clase de equivalencia, respecto a la relación dada en la definición 2.3.3.6.

El siguiente resultado fue demostrado en [2].

Teorema 2.3.3.8.

Si las imágenes $P = (Z^2, m, n, B)$ y $P' = (Z^2, m, n, B')$ son equivalentes en sentido de combinatoria, entonces sus correspondientes realizaciones poliédricas tienen el mismo tipo homotópico en sentido ordinario ($\|P\| \cong \|P'\|$).

Este teorema es de fundamental importancia para nuestro trabajo porque garantiza la conservación de las propiedades homotópicas de la imagen, al pasar de una imagen binaria a su contracción. Entre las propiedades invariantes está el número de componentes conexas, el número de agujeros y la característica de Euler. Tales propiedades nos permiten obtener el número de encierros a partir de la contracción y expansión combinatoria.

Capítulo III
Resultados

El sistema creado en la presente tesis consta de las siguientes secciones: la sección que asigna a cada punto (i, j) un vector de dirección y un valor binario, nos dará como resultado una imagen binaria, con su correspondiente matriz de direcciones.

La sección del cálculo del índice de Poincaré, el cual nos servirá para localizar deltas y cores, y por ultimo la sección de cálculo de encierros en una imagen. Estas secciones se explicarán detalladamente más adelante.

El funcionamiento se muestra en la figura 3.1.

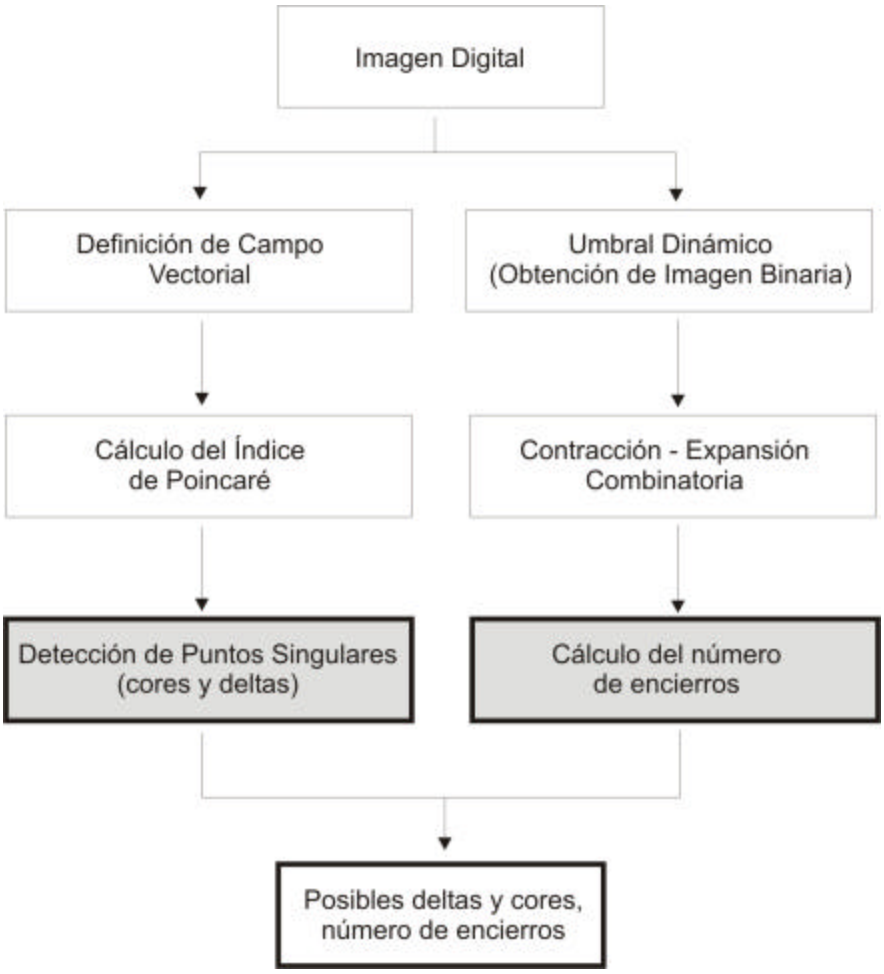


Figura 3.1. Diagrama del Sistema

3.1. Filtro contraste

El sistema tiene la capacidad de almacenar tonalidades de gris en una matriz de cualquier tamaño. Al digitalizar la imagen, es posible que se capture con ruido, por lo que es necesario tratar de eliminarlo en la medida de lo posible. Para este trabajo se realizaron una serie de pruebas con distintos filtros.

Se logró un mejor resultado al aplicar el filtro de contraste. Para ello fue necesario fijar un umbral; después de una serie de pruebas definimos un umbral con el valor de 135 y otro umbral con el valor de 210, logrando eliminar el ruido alrededor de la imagen sin eliminar información que formara parte de las líneas de la huella. En la figura 3.2a mostramos la imagen original, la figura 3.2b muestra la imagen al aplicar el filtro variación del contraste.

Algoritmo de contraste.

Este algoritmo tendrá como datos de entrada, la imagen original, almacenada en una matriz llamada MatXY[][].

Paso 1. Si MatXY[i][j] <= 135
MatXY[i][j] = 0
Si MatXY[i][j] >= 210
MatXY[i][j] = 255
En caso contrario, le dejamos su valor de gris original

Fin del Algoritmo.

a) b)

Figura 3.2. La imagen (a) Muestra la huella original, la imagen (b) muestra la huella aplicando contraste

3.2. Binarización y asignación de direcciones.

Aplicamos inmediatamente después un algoritmo que nos permitirá asignar a cada punto su correspondiente vector de dirección y su correspondiente valor binario (0 ó 1). Para lograr esto aplicamos el siguiente algoritmo.

Algoritmo para la asignación de dirección y construcción de imagen binaria.

Este módulo, tiene como datos de entrada una matriz llamada MatXY[][]. Además hace uso de un arreglo llamado ArrS[], que nos servirá para guardar las sumatorias para determinar las direcciones; MatXY[i][j] es el punto del centro de la vecindad y al cual se le asignará una dirección y un valor binario, MatImagDirec[ ][ ] guardará la dirección de cada punto, y en MatImagBina[ ][ ] se le asignará el valor binario a cada punto.

Paso 1. punto = 4 * MatXY[i][j]

Paso 2. De acuerdo a la figura 2.3, obtenemos sumas direccionales

```
k = 0
ArrS[ k ] = 0; k++;
ArrS [ k ] = MatXY[ i ][ j-4 ]+ MatXY[ i ][ j-2 ]+ MatXY[ i ][ j+2 ]+ MatXY[ i ][ j+4 ]; k++;
ArrS [ k ] = MatXY[ i+2 ][ j-4 ]+ MatXY[ i+1 ][ j-2 ]+ MatXY[ i-1 ][ j+2 ]+ MatXY[ i-2 ][ j+4 ]; k++;
ArrS [ k ] = MatXY[ i+4 ][ j-4 ] + MatXY[ i+2 ][ j-2 ] + MatXY[ i-2 ][ j+2 ] + MatXY[ i-4 ][ j+4]; k++;
ArrS [ k ] = MatXY[ i+4 ][ j-2 ] + MatXY[ i+2 ][ j-1 ] + MatXY[ i-2 ][ j+1 ] + MatXY[ i-4 ][ j+2]; k++;
ArrS [ k ] = MatXY[ i+4 ][ j ] + MatXY[ i+2 ][ j ] + MatXY[ i-2 ][ j ] + MatXY[ i-4 ][ j ]; k++;
ArrS [ k ] = MatXY[ i-4 ][ j-2 ] + MatXY[ i-2 ][ j-1 ] + MatXY[ i+2 ][ j+1 ] + MatXY[ i+4 ][ j+2]; k++;
ArrS [ k ] = MatXY[ i-4 ][ j-4 ] + MatXY[ i-2 ][ j-2 ] + MatXY[ i+2 ][ j+2 ] + MatXY[ i+4 ][ j+4]; k++;
ArrS [ k ] = MatXY[ i-2 ][ j-4 ] + MatXY[ i-1 ][ j-2 ] + MatXY[ i+1 ][ j+2 ] + MatXY[ i+2 ][ j+4]; k++;
```

Paso 3. Calculamos SUMADIR = ArrS[1] + ArrS[2] + ArrS[3] + ArrS[4] + ArrS[5] + ArrS[6] + ArrS[7] + ArrS[8]

Paso 4. De ArrS[], obtenemos la suma mínima *Sumamin* y la suma máxima *Sumamax*

Paso 5. Si *Sumamin* = *Sumamax*
Si *min* es igual a *max* ó se diferencian en 1
MatImagDirec[i][j] = *min*
MatImagBina[i][j] = 1
En caso contrario
MatImagDirec[i][j] = *es una dirección indefinida (9)*
MatImagBina[i][j] = 0
Sino Si (punto = punto + *Sumamin* + *Sumamax* > SUMADIR * 3/8)
MatImagDirec[i][j] = *min*
MatImagBina[i][j] = 1
En caso contrario
MatImagDirec[i][j] = *max*
MatImagBina[i][j] = 0

Fin del algoritmo.

De esta manera, obtenemos su correspondiente matriz de vectores (fig. 3.4b). En la figura 3.3a se muestra parte de la matriz binaria, en la figura 3.3b mostramos parte de la matriz binaria aumentada, esto para mostrar los valores 0´s y 1's y la imagen binaria (fig. 3.3c), estas matrices nos permitirán más adelante realizar los cálculos para los puntos característicos.

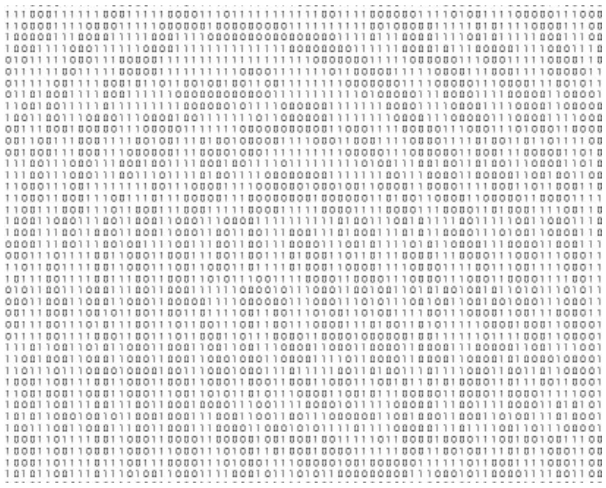


Figura 3.3.(a) Parte de la matriz binaria compuesta por 0's y 1's.

Figura 3.3.(b) Parte de matriz binaria aumentada compuesta por 0's y 1's.

Figura 3.3.(c) Imagen binaria.

Hasta el momento contamos con una matriz de direcciones, por lo que cada punto de la imagen tiene asociado un vector. Los ángulos de dirección que se tomaron fueron los que se muestran en la figura 2.7.

En la figura 3.4b se muestra parte de la imagen representada con vectores, en la figura 3.4a el respectivo campo vectorial definido sobre la huella dactilar.

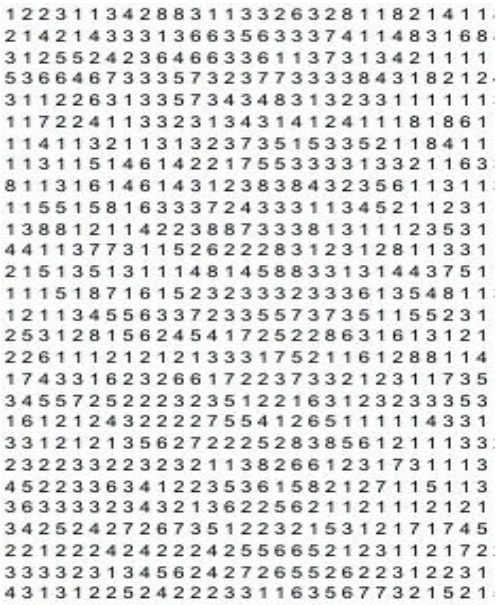


Figura 3.4.(a) Matriz de direcciones de la figura 2.14a.

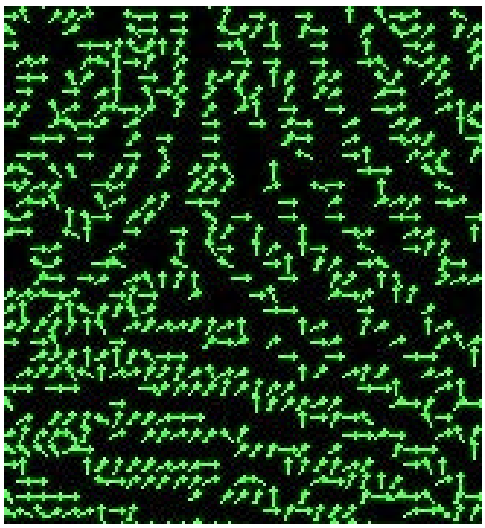


Figura 3.4.(b) Imagen representada con vectores.

Hasta aquí se tiene todavía una matriz (tanto binaria como de direcciones) de tamaño 194x194. Enseguida se divide dicha matriz en submatrices cuadradas de 8x8, en cada una de las cuales se determinará una sola dirección, llamada dirección predominante. De esto trata la siguiente sección.

3.3. Algoritmo para calcular ángulo predominante.

Los datos de entrada para esta función, es la matriz de direcciones MatImagDirec[][]. Dividir el ancho y alto de la imagen entre 8, para que sea nuestra sentencia de paro. Los ángulos predominantes se van a almacenar en una matriz llamada MatAng8[][].

Paso 1. Ancho/8 y Largo/8.

Paso 2. Tomar vecindades de 8x8 de la matriz de direcciones.

Paso 3. Se realiza el cálculo de la distribución de probabilidades hallando la frecuencia de cada dirección en el segmento y se almacena en un arreglo.

Paso 4. Se localizan las probabilidades mínima y máxima.

Paso 5.

Si Probabilidad máxima - Probabilidad mínima = 0.1
MatAng8[i][j] = 9, ya que existe mucho ruido en ese segmento.
En caso contrario
Se define un umbral (umbral = .75(probabilidad máxima – probabilidad mínima) + probabilidad mínima)y se cuentan cuantas sobrepasan dicho umbral, que en este caso serían las Direcciones Predominantes.
Si la Dirección Predominante = 1
MatAng8[i][j] = Dirección Predominante
Sino Si la Dirección Predominante = 2
MatAng8[i][j] = Dirección del ángulo medio (aproximada) entre las dos predominantes.
En otro caso
MatAng8[i][j] = 9

Fin del algoritmo.

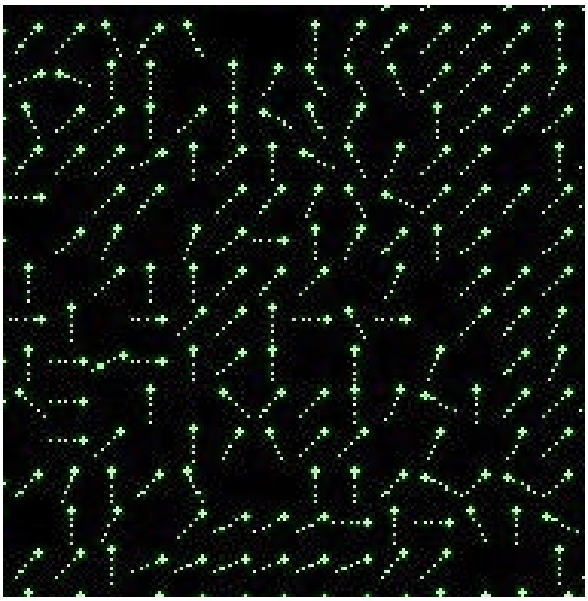


Figura 3.5. Imagen representada con vectores predominantes.

Hasta ahora, tenemos ya un campo vectorial definido sobre un subconjunto finito de Z^2. Los vectores son unitarios y sus ángulos están dados en la matriz MatAng[][].

Después de hacer una serie de pruebas, se le aplicó un proceso de suavizamiento a los valores predominantes.

Por lo regular, después de haber aplicado el algoritmo anterior, el campo vectorial tiene cambios bruscos, por lo que es necesario un algoritmo de "suavizamiento".

3.4. Suavizamiento del campo vectorial.

Las direcciones que hasta este momento tenemos cambian bruscamente, por ello necesitamos suavizarlas en una vecindad local. Para cada pixel, calculamos las direcciones α en grados ($\alpha \in [0, 180°]$), multiplicamos este resultado por dos, y los representamos como un vector unitario en esa dirección $v = (\cos 2\alpha, \operatorname{sen} 2\alpha)$. Una imagen representada de esta manera puede ser suavizada aplicando el siguiente algoritmo.

Algoritmo de Suavizamiento.

Este módulo tendrá como datos de entrada, la matriz de direcciones MatImagDirec[][].

Paso 1. Tomar Vecindades de 3x3 de MatImagDirec[][].

Paso 2. Sumar los valores de la vecindad con sus correspondientes vectores (*coseno 2q, seno 2q*), excluyendo los valores con dirección indefinida.

Paso 3. Se obtiene el promedio de los valores del coseno y del seno.

Paso 4.

Si el promedio del coseno = 0
    El valor del ángulo seguirá siendo el mismo
En caso contrario
    Angulo = $\tan^{-1}$(Promedio del seno / Promedio del coseno) / 2
    Si el ángulo es negativo
        Ángulo = 360 + Ángulo

Fin del algoritmo

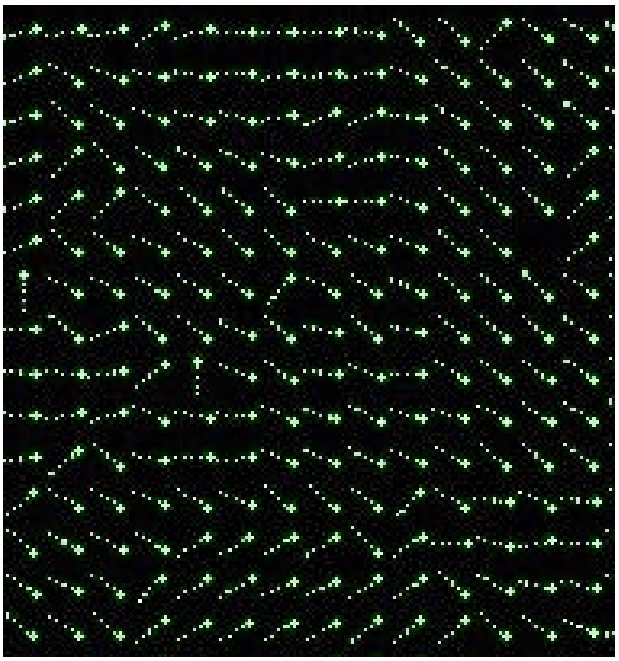


Figura 3.6. Imagen representada con vectores predominantes y suavizada.

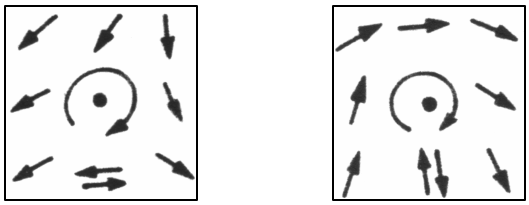
3.5. Índice de Poincaré.

En la subsección 2.2.1. se ha explicado cómo calcular el índice de Poincaré a lo largo de una curva cerrada (fig. 2.4), en donde se han elegido tres posibles direcciones. En la subsección 2.2.2. hemos mencionado que para establecer un campo vectorial sobre una imagen de huella dactilar se han considerado ocho direcciones (fig. 2.6), con lo cual "discretizamos" el campo vectorial requerido para nuestro caso.

Hemos elegido una curva cerrada discreta Γ formada por ocho puntos, recorridos estos en sentido contrario a las manecillas del reloj. Procedemos a calcular el índice de Poincaré, como la suma de los cambios en el ángulo de dirección a lo largo de Γ .

Con ello detectamos el número de posibles candidatos a ser puntos *cores* o *deltas.*

Aunque en la literatura especializada [25-26] se dice que al calcular el índice de Poincaré se obtiene la clasificación mostrada en la figura 3.7, en la práctica nos damos cuenta que no es así. Por lo que se deben establecer unos rangos para ubicar a los posibles candidatos a fungir como cores o deltas.



a) delta b) core

Figura 3.7. (a) Indice = -0.5 , (b) Indice = 0.5

En este trabajo, hemos hallado los rangos mostrados en la tabla de la figura 3.8.

	deltas	**Cores**
Rango de índice de Poincaré	**[-0.6, -0.4]**	**[0.4, 0.6]**

a)

Huellas analizadas	**No. de posibles deltas**	**No. de posibles cores**
0 deltas y 1 core	0	3
2 deltas y 1 core	4	3
2 deltas y 2 cores	3	4

b)

Figura 3.8. (a) Rangos de puntos característicos, (b) Huellas analizadas.

Una vez que se tienen los candidatos a ser cores o deltas, el siguiente paso consiste en distinguir cuales sí son y cuáles no. Esto se puede realizar de distintas maneras; por ejemplo siguiendo el método sugerido en [3], [9], [14], o aplicando el algoritmo c-media [23], [29].
Pero, esta tarea es motivo de otro trabajo.

El algoritmo que se ha propuesto para obtener tales resultados es el siguiente.

Algoritmo para calcular el índice de Poincaré.

Este algoritmo tomará los datos de la matriz llamada MatAng[][], el resultado de las variaciones de los puntos se guardarán en un arreglo llamado MatRes[], los resultados obtenidos los guarda en un archivo de texto.

Tomar los ocho puntos, y se almacenan en un arreglo Ar[]

Ar[0] = MatAng [i-1][j-1]
Ar[1] = MatAng [i][j-1]
Ar[2] = MatAng [i+1][j-1]
Ar[3] = MatAng [i+i][j]
Ar[4] = MatAng [i+1][j+1]
Ar[5] = MatAng [i][j+1]
Ar[6] = MatAng [i-1][j+1]
Ar[7] = MatAng [i-1][j]

Paso 1. Se obtienen las *Variaciones* entre los 8 puntos y se guardan en otro arreglo ArDif[]

Paso 2. Se suman las variaciones y el resultado se divide entre 360.

Paso 3. El resultado se almacena en una matriz MatRes[] para después almacenarse en un archivo.

Fin del algoritmo.

Algoritmo para calcular las variaciones de los ángulos.

Este algoritmo tomará los datos del arreglo Ar[].

Para cada elemento del arreglo Ar[]

Paso 1. K1 = (-0.5) – (Ar[i+1] – Ar[i]) / 2π
K2 = (0.5) – (Ar[i+1] – Ar[i]) / 2π

Paso 2. Se busca un entero K, donde K1 < K < K2

Paso 3. *Variación* = (Ar[i+1] – Ar[i]) + 2Kπ

Fin del algoritmo.

Habiendo aplicado el algoritmo anterior, hemos obtenido una matriz de *ÍNDICES* (fig. 3.9), a partir de la cual detectamos el número de posibles candidatos a ser puntos *cores* o *deltas.*

Fig. 3.9. Matriz de Índice de Poincaré de la fig. 3.2a.

3.6. Cálculo de encierros.

Ya se ha dicho al final de la sección **1.2.,** que una de las características *minutiaes* (minucias) consideradas para clasificar una huella dactilar, es lo que se conoce como *encierro* (fig. 1.2).

Como parte del presente trabajo de tesis, nos dimos cuenta que esta característica cumple con la definición de *agujero* (definición 2.3.1.7. y figura 2.13c). Así, el problema de calcular el número de encierros presentes en una huella dactilar, lo trasladamos a encontrar el número de agujeros en la correspondiente imagen binaria (fig. 3.10a).

En [1] se ha dado un algoritmo para el cálculo del número de agujeros en imágenes binarias bidimensionales basado en el proceso de expansión combinatoria (definición 2.3.3.5), y en [2] se dio un algoritmo para el mismo cálculo utilizando la eliminación de pixeles de índice 2. En ambos casos, se hace de manera indirecta ya que se realiza a través de la característica de Euler (definición 2.3.2.6).

Nosotros hemos modificado el algoritmo dado en [1], para cuyo efecto, nos basamos en la herramienta de topología digital (subsección 2.3.1), en la realización geométrica de la imagen binaria (subsección 2.3.2), y sobre todo en el concepto de índice de un pixel introducido por primera vez por Schepin [35], así como en el concepto de invarianza de una transformación dado en Bykov-Rodríguez [1-2]. En la subsección 2.3.3. se ha mencionado que la eliminación o agregación de un pixel de índice 1 no cambia el tipo homotópico de la realización poliédrica de la imagen binaria (definición 2.3.3.4, definición 2.3.3.7 y teorema 2.3.3.8.); por lo que la eliminación sucesiva de pixeles de índice 1 seguida de agregación sucesiva del mismo tipo de pixeles, mantiene invariantes tanto el número de componentes conexas como el número de agujeros.

La idea general es aislar de alguna manera los encierros. Para ello, hacemos expansión combinatoria (agregación de pixeles negros, que siendo blancos tienen índice 1 respecto al conjunto de pixeles negros); logrando con esto aislar algunos encierros. Repitiendo el proceso, obtenemos al final el número de encierros.

El algoritmo resultante que proponemos para el cálculo del número de encierros, es el siguiente:

Algoritmo para el cálculo de encierros.

El dato de entrada para este módulo, es la imagen binaria MatImagBina[][].

Paso 1. Se aplica Contracción Combinatoria:

Bandera=0
Para cada punto de la imagen, Si el Pixel es negro y su índice es igual a 1
Pixel se vuelve blanco
Bandera=1

Si Bandera=1 se vuelve a repetir el Paso 1.

Paso 2. Se eliminan los pixeles aislados:

Bandera=0
Para cada punto de la imagen, Si el Pixel es negro y aislado de otros puntos negros
Pixel se vuelve blanco
Bandera=1

Si Bandera=1 se vuelve a repetir el Paso 2.

Paso 3. Se aplica Expansión Combinatoria:

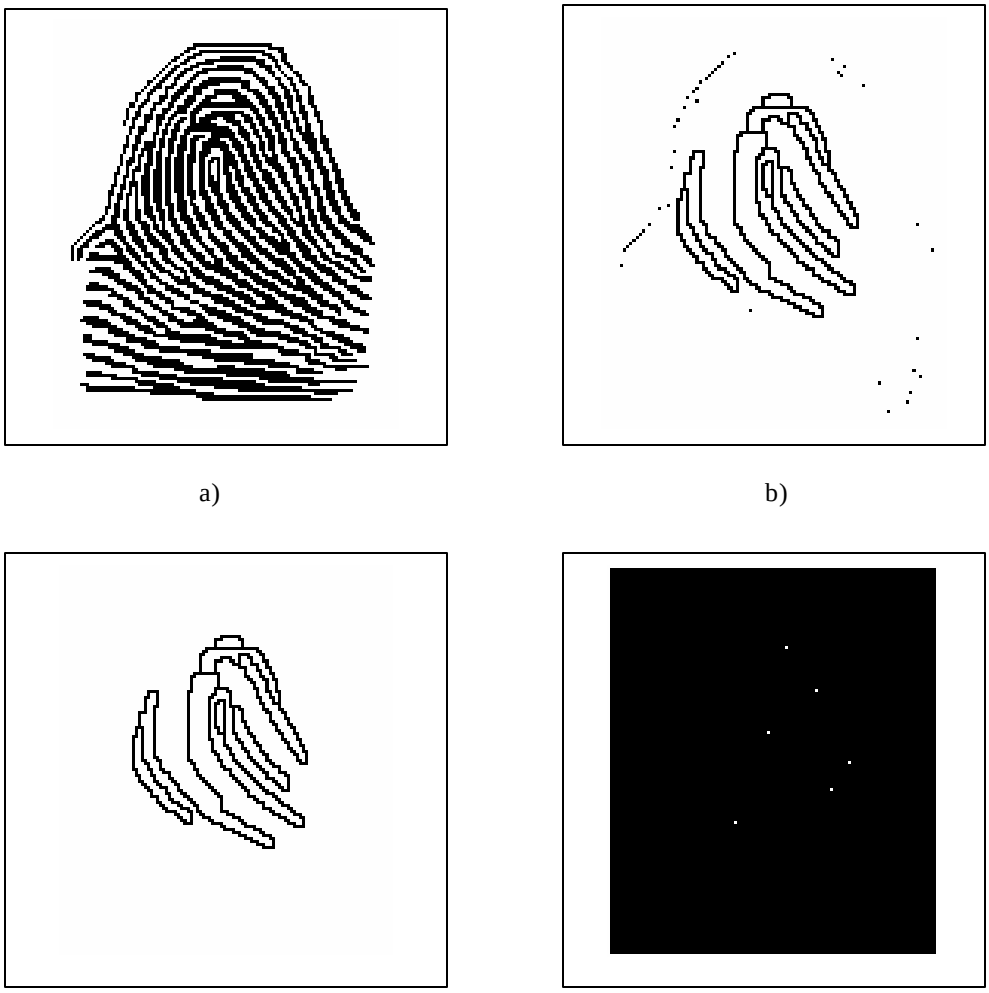
Bandera=0
Para cada punto de la imagen, Si el Pixel es blanco y su índice es igual a 1
Pixel se vuelve negro
Bandera=1

Si Bandera=1 se vuelve a repetir el Paso 3.

Paso 4. Se cuentan los encierros:

Agujeros=0
Para cada punto de la imagen, Si el Pixel es blanco y aislado de otros puntos blancos
Agujeros=Agujeros+1

Fin del Algoritmo.



a) b)

c) d)

Figura 3.10. (a) Matriz binaria con seis encierros, (b) Imagen después de la contracción combinatoria, (c) Eliminación de pixeles aislados negros, (d) Imagen después de la Expansión combinatoria.

CONCLUSIONES

Se utilizaron métodos de Topología Combinatoria y Sistemas Dinámicos para obtener posibles *deltas* y *cores* en una huella dactilar. Se utilizaron también herramientas de Topología Algebraica y Topología Digital para calcular el número de *encierros* presentes en la misma huella. Los algoritmos propuestos fueron programados en Visual Basic 6.0.

Por todo ello, se ha cumplido el objetivo inicialmente planteado.

Los resultados de este trabajo son un aporte para la creación de un sistema completo de clasificación y reconocimiento automático de huellas dactilares.

En el desarrollo de este trabajo, se observó que una de las limitantes para el procesamiento y cálculo adecuado de los puntos característicos en la huella es la imagen original; ya que regularmente contiene mucho ruido. Por este motivo, los expertos en el análisis de las huellas regularmente limpian la imagen a mano. Aunque de igual forma, existen herramientas especiales para obtener estas imágenes en mejor estado, como lo son los escaners ópticos.

Este trabajo forma parte del Proyecto de Investigación II-42G01, financiado por el Consejo Nacional de Ciencia y Tecnología (CONACYT).

ANEXO 1

Código de los algoritmos propuestos.

‘ Declaraciones iniciales de la librería VIC32

```
Option Explicit

Global Const NO_ERROR = 0        ' No error
Global Const RanMat = 7
Global Const Pi = 3.14159265358979

Declare Function pcxinfo Lib "VIC32.DLL" (ByVal Fname As String, pdata As PcxData)
As Long
Declare Function tiffinfo Lib "VIC32.DLL" (ByVal Fname As String, tdat As TiffData) As
Long
Declare Function jpeginfo Lib "VIC32.DLL" (ByVal Fname As String, jdat As JpegData)
As Long

Declare Function allocimage Lib "VIC32.DLL" (image As imgdes, ByVal wid As Long,
ByVal leng As Long, ByVal BPPixel As Long) As Long

Declare Function loadpcx Lib "VIC32.DLL" (ByVal Fname As String, desimg As imgdes)
As Long
Declare Function loadtif Lib "VIC32.DLL" (ByVal Fname As String, desimg As imgdes)
As Long
Declare Function loadjpg Lib "VIC32.DLL" (ByVal Fname As String, desimg As imgdes)
As Long

Declare Sub freeimage Lib "VIC32.DLL" (image As imgdes)

Declare Function savejpg Lib "VIC32.DLL" (ByVal Fname As String, srcimg As imgdes,
ByVal quality As Long) As Long
Declare Function savepcx Lib "VIC32.DLL" (ByVal Fname As String, srcimg As imgdes)
As Long
Declare Function savetif Lib "VIC32.DLL" (ByVal Fname As String, srcimg As imgdes,
ByVal cmp As Long) As Long

Declare Function getpixelcolor Lib "VIC32.DLL" (image As imgdes, ByVal xcoord As
Long, ByVal ycoord As Long) As Long
Declare Function setpixelcolor Lib "VIC32.DLL" (image As imgdes, ByVal xcoord As
Long, ByVal ycoord As Long, ByVal level As Long) As Long
Declare Function viewimage Lib "VIC32.DLL" (ByVal hWnd As Long, ByVal hdc As
Long, hpal As Long, ByVal xpos As Long, ByVal ypos As Long, image As imgdes) As
Long
```

```
' Image descriptor
Type imgdes
   ibuff As Long
   stx As Long
   sty As Long
   endx As Long
   endy As Long
   buffwidth As Long
   palette As Long
   colors As Long
   imgtype As Long
   bmh As Long
   hBitmap As Long
End Type

Type PcxData
   PCXvers As Long
   width As Long
   length As Long
   BPPixel As Long
   Nplanes As Long
   BytesPerLine As Long
   PalInt As Long
   vbitcount As Long
End Type

Type TiffData
   ByteOrder As Long
   width As Long
   length As Long
   BitsPSample As Long
   comp As Long
   SamplesPPixel As Long
   PhotoInt As Long
   PlanarCfg As Long
   vbitcount As Long
End Type

Type JpegData
    ftype As Long
    width As Long
    length As Long
    comps As Long
    precision As Long
    sampfac0 As Long
    sampfac1 As Long
    sampfac2 As Long
```

```
    sampfac3 As Long
    vbitcount As Long
End Type

Type BITMAPINFOHEADER
    biSize As Long
    biWidth As Long
    biHeight As Long
    biPlanes As Integer
    biBitCount As Integer
    biCompression As Long
    biSizeImage As Long
    biXPelsPerMeter As Long
    biYPelsPerMeter As Long
    biClrUsed As Long
    biClrImportant As Long
End Type

' Declaraciones en la ventana principal de las matrices y otras variables que van a ser
' usadas
Option Explicit

Dim MatXY() As Long
Dim MatImagBina() As Integer
Dim MatImagDirec() As Integer
Dim MatAng() As Double
Dim MatAng8() As Integer

Dim AnchoSegmento As Integer
Dim AnchoImagen As Long
Dim AltoImagen As Long
Dim BitImagen As Integer
Dim FormatoImagen As Integer
Dim PuntoX As Long
Dim PuntoY As Long
Dim ColSuav As Integer

Public formaImagen As frmImagen

' Asignación de valores iniciales al cargar la ventana principal
Private Sub MDIForm_Load()

lblCores.Visible = False
lblDeltas.Visible = False
lblEncierros.Visible = False
lblDatImagen.Visible = False
cmdProcesa.Enabled = False
```

```
PuntoX = 0
PuntoY = 40
ColSuav = 0
End Sub

‘ Función pricipal de procesamiento
Private Sub cmdProcesa_Click()

ReDim MatImagBina(AnchoImagen - 1, AltoImagen - 1) As Integer
ReDim MatImagDirec(AnchoImagen - 1, AltoImagen - 1) As Integer
Dim Anc As Long
Dim Lar As Long
Dim Punto As Long
Dim ArrS(1 To 8) As Long
Dim SumaDir As Long
Dim SumaMin As Long
Dim SumaMax As Long
Dim Dmin As Integer
Dim Dmax As Integer
Dim x As Long
Dim y As Long
Dim fso, txtfile
Dim fso10, txtfile10
Dim fsodir, txtfiledir
Dim cadena As String
Dim cadena10 As String
Dim cadenadir As String
Dim rcode As Long
Dim tmpimage As imgdes

Screen.MousePointer = vbHourglass

' se aplica filtro a la imagen a traves del contraste
Contraste

For Lar = 4 To AltoImagen - 5
   For Anc = 4 To AnchoImagen - 5
      ' paso 2 del logaritmo: se obtiene el valor de punto
      Punto = 4 * MatXY(Anc, Lar)
'      ' paso 3 del algoritmo: se obtienen las sumatorias para vecindades de 9x9
      ArrS(1) = MatXY(Anc - 4, Lar) + MatXY(Anc - 2, Lar) + MatXY(Anc + 2, Lar) +
MatXY(Anc + 4, Lar)
      ArrS(2) = MatXY(Anc - 4, Lar + 2) + MatXY(Anc - 2, Lar + 1) + MatXY(Anc + 2,
Lar - 1) + MatXY(Anc + 4, Lar - 2)
      ArrS(3) = MatXY(Anc - 4, Lar + 4) + MatXY(Anc - 2, Lar + 2) + MatXY(Anc + 2,
Lar - 2) + MatXY(Anc - 4, Lar + 4)
```

```
      ArrS(4) = MatXY(Anc - 2, Lar + 4) + MatXY(Anc - 1, Lar + 2) + MatXY(Anc + 1,
Lar - 2) + MatXY(Anc + 2, Lar - 4)
      ArrS(5) = MatXY(Anc, Lar + 4) + MatXY(Anc, Lar + 2) + MatXY(Anc, Lar - 2) +
MatXY(Anc, Lar - 4)
      ArrS(6) = MatXY(Anc - 2, Lar - 4) + MatXY(Anc - 1, Lar - 2) + MatXY(Anc + 1, Lar
+ 2) + MatXY(Anc + 2, Lar + 4)
      ArrS(7) = MatXY(Anc - 4, Lar - 4) + MatXY(Anc - 2, Lar - 2) + MatXY(Anc + 2, Lar
+ 2) + MatXY(Anc + 4, Lar + 4)
      ArrS(8) = MatXY(Anc - 4, Lar - 2) + MatXY(Anc - 2, Lar - 1) + MatXY(Anc + 2, Lar
+ 1) + MatXY(Anc + 4, Lar + 2)
      ' paso 4 del algoritmo: se obtienen las suma total de las sumatorias
      SumaDir = ArrS(1) + ArrS(2) + ArrS(3) + ArrS(4) + ArrS(5) + ArrS(6) + ArrS(7) +
ArrS(8)
      ' paso 5 del algoritmo: se obtiene la suma mínima y la suma máxima
      SumaMin = ArrS(1)
      Dmin = 1
      SumaMax = ArrS(1)
      Dmax = 1
      For x = 2 To 8
         If ArrS(x) < SumaMin Then
            SumaMin = ArrS(x)
            Dmin = x
         End If
         If ArrS(x) > SumaMax Then
            SumaMax = ArrS(x)
            Dmax = x
         End If
      Next x
      ' paso 6 del algoritmo: se determina la direccion del PUNTO
      If SumaMin = SumaMax Then
         If (Abs(Dmin - Dmax) = 1) Or (Dmin = Dmax) Then
            MatImagDirec(Anc, Lar) = Dmin   ' direccion minima en la matriz de direcciones
            MatImagBina(Anc, Lar) = 1       ' color blanco en la matriz binaria
         Else
            MatImagDirec(Anc, Lar) = 9      ' direccion indefinida
            MatImagBina(Anc, Lar) = 0       ' color negro en la matriz binaria
         End If
      ElseIf (Punto + SumaMin + SumaMax) > (SumaDir * (3 / 8)) Then
         MatImagDirec(Anc, Lar) = Dmin       ' direccion minima en la matriz de direcciones
         MatImagBina(Anc, Lar) = 1           ' color blanco en la matriz binaria
      Else
         MatImagDirec(Anc, Lar) = Dmax       ' direccion maxima en la matriz de
direcciones
         MatImagBina(Anc, Lar) = 0           ' color negro en la matriz binaria
      End If
   Next Anc
Next Lar
```

```
' las orillas se vuelven blancas en la matriz binaria e indefinidas en la de direcciones
For y = 0 To 3
   For x = 0 To AnchoImagen - 1
      MatImagBina(x, y) = 1
      MatImagDirec(x, y) = 9
   Next x
Next y
For y = AltoImagen - 4 To AltoImagen - 1
   For x = 0 To AnchoImagen - 1
      MatImagBina(x, y) = 1
      MatImagDirec(x, y) = 9
   Next x
Next y
For y = 0 To AltoImagen - 1
   For x = 0 To 3
      MatImagBina(x, y) = 1
      MatImagDirec(x, y) = 9
   Next x
Next y
For y = 0 To AltoImagen - 1
   For x = AnchoImagen - 4 To AnchoImagen - 1
      MatImagBina(x, y) = 1
      MatImagDirec(x, y) = 9
   Next x
Next y

' se crean archivos para la matriz de tonos de gris, la de direcciones y la binaria
Set fso = CreateObject("Scripting.FileSystemObject")
Set txtfile = fso.CreateTextFile("matriz.txt", True)
Set fso10 = CreateObject("Scripting.FileSystemObject")
Set txtfile10 = fso10.CreateTextFile("matriz10.txt", True)
Set fsodir = CreateObject("Scripting.FileSystemObject")
Set txtfiledir = fsodir.CreateTextFile("matrizdir.txt", True)

For y = 0 To AltoImagen - 1
   cadena = ""
   cadena10 = ""
   cadenadir = ""
   For x = 0 To AnchoImagen - 1
      cadena = cadena & MatXY(x, y) & " "
      cadena10 = cadena10 & MatImagBina(x, y) & " "
      cadenadir = cadenadir & MatImagDirec(x, y) & " "
   Next x
   txtfile.Write (cadena)
   txtfile.WriteBlankLines (1)
   txtfile10.Write (cadena10)
   txtfile10.WriteBlankLines (1)
```

```
    txtfiledir.Write (cadenadir)
    txtfiledir.WriteBlankLines (1)
Next y
txtfile.Close
txtfile10.Close
txtfiledir.Close

' asigna espacio para una imagen
rcode = allocimage(tmpimage, AnchoImagen, AltoImagen, 1)
If (rcode <> NO_ERROR) Then
  MsgBox "error: No hay suficiente memoria para continuar", vbCritical
  Exit Sub
End If

' se crea la imagen a partir de la matriz binaria
For y = 0 To AltoImagen - 1
    For x = 0 To AnchoImagen - 1
        If MatXY(x, y) = 255 Then
            MatImagBina(x, y) = 1
        Else
            MatImagBina(x, y) = 0
        End If
    Next x
Next y
For y = 0 To AltoImagen - 1
    For x = 0 To AnchoImagen - 1
        If MatImagBina(x, y) = 1 Then
            setpixelcolor tmpimage, x, y, 255
        Else
            setpixelcolor tmpimage, x, y, 0
        End If
    Next x
Next y
' se manda a presentar la imagen binaria en pantalla
CargaImagenEnForma tmpimage, AnchoImagen, AltoImagen, "Imagen binaria"
' se libera la imagen
freeimage tmpimage

' se buscan los encierros en la huella
BuscaEncierros MatImagBina, Anc, Lar

' se asignan a la matriz MATANG, los angulos de la matriz de direcciones
ConvertirAangulo MatImagDirec, AnchoImagen, AltoImagen

' se genera el archivo de la matriz de angulos
Set txtfile = fso.CreateTextFile("matrizAng.txt", True)
```

```
For y = 0 To AltoImagen - 1
   cadena = ""
   For x = 0 To AnchoImagen - 1
      cadena = cadena & MatAng(x, y) & " |"
   Next x
   txtfile.Write (cadena)
   txtfile.WriteBlankLines (1)
Next y
txtfile.Close

' se cargan en pantalla las direcciones
CargaImagenDirec8 MatAng, MatImagBina, AnchoImagen, AltoImagen, "Direcciones
originales", _
   PuntoX, PuntoY

Anc = Int(AnchoImagen / 8)
Lar = Int(AltoImagen / 8)
ReDim MatAng8(Anc - 1, Lar - 1) As Integer

' realiza proceso de reduccion a direcciones predominantes
x = Int((AnchoImagen - (Anc * 8)) / 2)
y = Int((AltoImagen - (Lar * 8)) / 2)
MatAng8 = MatA8Reduc(MatImagDirec, Anc, Lar, x, y)

' se asignan a la matriz MATANG, los angulos de la matriz de direcciones
ConvertirAangulo MatAng8, Anc, Lar

' se genera el archivo de la matriz resultante
Set txtfile = fso.CreateTextFile("matrizA8.txt", True)
For y = 0 To Lar - 1
   cadena = ""
   For x = 0 To Anc - 1
      cadena = cadena & MatAng(x, y) & " |"
   Next x
   txtfile.Write (cadena) ' Escribe una línea.
   txtfile.WriteBlankLines (1)
Next y
txtfile.Close

' se muestran las direcciones en pantalla
CargaImagenDirec88 MatAng, Anc, Lar, "Direcciones predominantes"

' realiza suavizamiento de la matriz de direcciones predominantes
MatAng = Suavizamiento(MatAng, MatImagBina, Anc, Lar, 2)
```

```
' muestra las direcciones en ventana
CargaImagenDirec88 MatAng, Anc, Lar, "Direcciones predominantes con  suavizamiento"

' se buscan cores y deltas en la huella y se guardan en un archivo
EncCaract4b MatAng, Anc, Lar

' se liberan los objetos creados
Set fso = Nothing
Set fso10 = Nothing
Set fsodir = Nothing

Screen.MousePointer = vbDefault

End Sub
```

‘ Función para seleccionar el archivo de imagen a procesar

```
Private Sub cmdAbrir_Click()

Dim s As String

   ' Establecer CancelError a True
    CommD.CancelError = True
On Error GoTo ErrHandler
    ' Presentar el cuadro de diálogo Imprimir
    CommD.Flags = cdlOFNExtensionDifferent + cdlOFNNoChangeDir +
cdlOFNFileMustExist + _
          cdlOFNHideReadOnly + cdlOFNNoReadOnlyReturn
    CommD.DialogTitle = "Cargar archivo gráfico"
    CommD.DefaultExt = ".*"
    CommD.Filter = "Archivos PCX (*.pcx)|*.pcx" _
          & "|Archivos TIFF (*.tif)|*.tif|Archivos JPG (*.jpg)|*.jpg"
    CommD.ShowOpen

    lblDatImagen.Visible = True

    ' se envia a la funcion que lee la informacion del archivo y la carga a una matriz
    InfoArchivo CommD.FileTitle, CommD.FileName

    Exit Sub
ErrHandler:
    ' El usuario ha hecho clic en el botón Cancelar


End Sub
```

‘ Función que revisa formatos de imagen y manda a leer la información de la imagen

```
Function InfoArchivo(NomArc As String, Direc As String)

If UCase(Right(NomArc, 3)) = "PCX" Then
   InfoPCX NomArc, Direc
ElseIf UCase(Right(NomArc, 3)) = "TIF" Then
   InfoTIFF NomArc, Direc
ElseIf UCase(Right(NomArc, 3)) = "JPG" Then
   InfoJPG NomArc, Direc
Else
   MsgBox "Tipo de archivo no permitido", vbCritical
   cmdProcesa.Enabled = False
   lblCores.Visible = False
   lblDeltas.Visible = False
   lblEncierros.Visible = False
   Exit Function
End If

cmdProcesa.Enabled = True
lblCores.Visible = False
lblDeltas.Visible = False
lblEncierros.Visible = False
AnchoSegmento = AnchoImagen / 2

End Function
```

' Lee y carga a una matriz el archivo tipo TIF

```
Private Sub InfoTIFF(Nom As String, NombreArc As String)

Dim rcode As Long
Dim bdat As TiffData  ' reserva espacio para una estructura TIFF
Dim tmpimage As imgdes    ' descriptor de imagen
Dim comp As Long

' Obtiene informacion del archivo TIF a cargar
rcode = tiffinfo(NombreArc, bdat)
If (rcode <> NO_ERROR) Then
   MsgBox "error: No es posible obtener la información del archivo", vbCritical
    lblDatImagen.Visible = False
   Exit Sub
End If

' variables globales sobre datos de la imagen
AnchoImagen = bdat.width
AltoImagen = bdat.length
BitImagen = bdat.vbitcount
FormatoImagen = 2   'tif
```

```
BitImagen = bdat.vbitcount
If (BitImagen >= 16) Then  ' 16-, 24-, o 32-bit se carga en un buffer de 24-bit
   BitImagen = 24
End If

' asigna espacio para una imagen
rcode = allocimage(tmpimage, bdat.width, bdat.length, BitImagen)
If (rcode <> NO_ERROR) Then
  MsgBox "error: No hay suficiente memoria", vbCritical
  Exit Sub
End If

' Carga imagen
rcode = loadtif(NombreArc, tmpimage)
If (rcode <> NO_ERROR) Then
   freeimage tmpimage ' libera la imagen sobre este error
   MsgBox "error: No se puede cargar el archivo", vbCritical
   Exit Sub
End If

Dim x As Long
Dim y As Long
ReDim MatXY(AnchoImagen - 1, AltoImagen - 1) As Long

' se obtienen los colores o tonos de gris en la matriz MatXY
For y = 0 To AltoImagen - 1
    For x = 0 To AnchoImagen - 1
       MatXY(x, y) = getpixelcolor(tmpimage, x, y)
    Next x
Next y

' se muestra la informacion de la imagen en pantalla
lblDatImagen.Caption = "Nombre:       " & UCase(Nom) & Chr(13) & Chr(13) _
    & "ByteOrder:           " & bdat.ByteOrder & Chr(13) _
    & "Ancho:               " & bdat.width & Chr(13) _
    & "Alto:              " & bdat.length & Chr(13) _
    & "BitsPSample:         " & bdat.BitsPSample & Chr(13) _
    & "comp:              " & bdat.comp & Chr(13) _
    & "SamplesPPixel:       " & bdat.SamplesPPixel & Chr(13) _
    & "PhotoInt:            " & bdat.PhotoInt & Chr(13) _
    & "PlanarCfg:           " & bdat.PlanarCfg & Chr(13) _
    & "Bits:              " & bdat.vbitcount
```

```
' se manda a cargar la imagen en pantalla en una nueva ventana
CargaImagenEnForma tmpimage, bdat.width, bdat.length, "Imagen original"
' se libera la imagen
freeimage tmpimage

End Sub

' Lee y carga a una matriz el archivo tipo PCX
Private Sub InfoPCX(Nom As String, NombreArc As String)

Dim rcode As Long
Dim bdat As PcxData  ' reserva espacio para una estructura PCX
Dim tmpimage As imgdes     ' descriptor de imagen


' Obtiene informacion del archivo PCX a cargar
rcode = pcxinfo(NombreArc, bdat)
If (rcode <> NO_ERROR) Then
   MsgBox "error: No es posible obtener la información del archivo", vbCritical
    lblDatImagen.Visible = False
   Exit Sub
End If

' variables globales sobre datos de la imagen
AnchoImagen = bdat.width
AltoImagen = bdat.length
BitImagen = bdat.vbitcount
FormatoImagen = 1   'pcx

BitImagen = bdat.vbitcount
If (BitImagen >= 16) Then  ' 16-, 24-, o 32-bit se carga en un buffer de 24-bit
   BitImagen = 24
End If

' asigna espacio para una imagen
rcode = allocimage(tmpimage, bdat.width, bdat.length, BitImagen)
If (rcode <> NO_ERROR) Then
  MsgBox "error: No hay suficiente memoria", vbCritical
  Exit Sub
End If

' Carga imagen
rcode = loadpcx(NombreArc, tmpimage)
If (rcode <> NO_ERROR) Then
   freeimage tmpimage ' libera la imagen sobre este error
   MsgBox "error: No se puede cargar el archivo", vbCritical
   Exit Sub
```

```
End If
Dim x As Long
Dim y As Long
ReDim MatXY(AnchoImagen - 1, AltoImagen - 1) As Long

' se obtienen los colores o tonos de gris en la matriz MatXY
For y = 0 To AltoImagen - 1
   For x = 0 To AnchoImagen - 1
      MatXY(x, y) = getpixelcolor(tmpimage, x, y)
   Next x
Next y

' se muestra la informacion de la imagen en pantalla
lblDatImagen.Caption = "Nombre:       " & UCase(Nom) & Chr(13) & Chr(13) _
   & "Version PCX:      " & bdat.PCXvers & Chr(13) _
   & "Ancho:            " & bdat.width & Chr(13) _
   & "Alto:             " & bdat.length & Chr(13) _
   & "BPPixel:          " & bdat.BPPixel & Chr(13) _
   & "Nplanes:          " & bdat.Nplanes & Chr(13) _
   & "Bytes por linea:  " & bdat.BytesPerLine & Chr(13) _
   & "PalInt:           " & bdat.PalInt & Chr(13) _
   & "Bits:             " & bdat.vbitcount

' se manda a cargar la imagen en pantalla en una nueva ventana
CargaImagenEnForma tmpimage, bdat.width, bdat.length, "Imagen original"
' se libera la imagen
freeimage tmpimage

End Sub

' Lee y carga a una matriz el archivo tipo JPG
Private Sub InfoJPG(Nom As String, NombreArc As String)

Dim rcode As Long
Dim bdat As JpegData  ' reserva espacio para una estructura JPG
Dim tmpimage As imgdes    ' descriptor de imagen

' Obtiene informacion del archivo JPG a cargar
rcode = jpeginfo(NombreArc, bdat)
If (rcode <> NO_ERROR) Then
   MsgBox "error: No es posible obtener la información del archivo", vbCritical
    lblDatImagen.Visible = False
   Exit Sub
End If

' variables globales sobre datos de la imagen
AnchoImagen = bdat.width
```

```
AltoImagen = bdat.length
BitImagen = bdat.vbitcount
FormatoImagen = 0   'jpg

BitImagen = bdat.vbitcount
If (BitImagen >= 16) Then  ' 16-, 24-, o 32-bit se carga en un buffer de 24-bit
   BitImagen = 24
End If

' asigna espacio para una imagen
rcode = allocimage(tmpimage, bdat.width, bdat.length, BitImagen)
If (rcode <> NO_ERROR) Then
  MsgBox "error: No hay suficiente memoria", vbCritical
  Exit Sub
End If

' Carga imagen
rcode = loadjpg(NombreArc, tmpimage)
If (rcode <> NO_ERROR) Then
   freeimage tmpimage ' libera la imagen sobre este error
   MsgBox "error: No se puede cargar el archivo", vbCritical
   Exit Sub
End If

Dim x As Long
Dim y As Long
ReDim MatXY(AnchoImagen - 1, AltoImagen - 1) As Long

' se obtienen los colores o tonos de gris en la matriz MatXY
For y = 0 To AltoImagen - 1
    For x = 0 To AnchoImagen - 1
       MatXY(x, y) = getpixelcolor(tmpimage, x, y)
    Next x
Next y

' se muestra la informacion de la imagen en pantalla
lblDatImagen.Caption = "Nombre:       " & UCase(Nom) & Chr(13) & Chr(13) _
    & "Tipo:            " & bdat.ftype & Chr(13) _
    & "Ancho:           " & bdat.width & Chr(13) _
    & "Alto:            " & bdat.length & Chr(13) _
    & "comps:           " & bdat.comps & Chr(13) _
    & "Precision:       " & bdat.precision & Chr(13) _
    & "Sampfac0:        " & bdat.sampfac0 & Chr(13) _
    & "Sampfac1:        " & bdat.sampfac1 & Chr(13) _
    & "Sampfac2:        " & bdat.sampfac2 & Chr(13) _
    & "Sampfac3:        " & bdat.sampfac3 & Chr(13) _
    & "Bits:            " & bdat.vbitcount
```

```
' se manda a cargar la imagen en pantalla en una nueva ventana
CargaImagenEnForma tmpimage, bdat.width, bdat.length, "Imagen original"
' se libera la imagen
freeimage tmpimage

End Sub

' Carga la imagen en pantalla
Private Sub CargaImagenEnForma(image As imgdes, Lancho As Long, Llargo As Long, _
      VTexto As String)

Dim rcode As Long

Screen.MousePointer = vbHourglass

Set formaImagen = New frmImagen
formaImagen.Hide
formaImagen.Caption = VTexto
formaImagen.ScaleMode = vbPixels
formaImagen.width = Lancho * 15.4
formaImagen.Height = Llargo * 16.4
formaImagen.Pic1.ScaleMode = vbPixels
formaImagen.Pic1.width = Lancho
formaImagen.Pic1.Height = Llargo

rcode = viewimage(formaImagen.Pic1.hWnd, formaImagen.Pic1.hdc, formaImagen.palette,
0, 0, image)

formaImagen.Show
Screen.MousePointer = vbDefault

End Sub

‘ Función que realiza contraste a la imagen
Function Contraste()

Dim x As Long
Dim y As Long

For y = 0 To AltoImagen - 1
   For x = 0 To AnchoImagen - 1
      ' se aplica el umbral para contraste
      If MatXY(x, y) <= 135 Then
         MatXY(x, y) = 0
      ElseIf MatXY(x, y) >= 208 Then
         MatXY(x, y) = 255
      End If
```

```
   Next x
Next y

End Function

‘ Carga la imagen en pantalla de los vectores
Private Sub CargaImagenDirec8(Mat() As Double, MatColor() As Integer, Lancho As
Long, _
       Llargo As Long, VTexto As String, PosX As Long, PosY As Long)
' muestra las direcciones a traves de vectores de un segmento de la matriz

Screen.MousePointer = vbHourglass

Set formaImagen = New frmImagen
formaImagen.Hide
formaImagen.Caption = VTexto
formaImagen.width = Lancho * 15.4
formaImagen.Height = Llargo * 16.4
formaImagen.Pic1.width = Lancho * 50
formaImagen.Pic1.Height = Llargo * 50
formaImagen.Pic1.BackColor = vbBlack

Dim t As Double
Dim a As Long
Dim b As Long
Dim c As Long
Dim d As Long
Dim x As Long
Dim y As Long
Dim i As Long
Dim j As Long
Dim AncFlecha As Integer

AncFlecha = 40

For j = PosY To PosY + AnchoSegmento
   For i = PosX To PosX + AnchoSegmento
      If Mat(i, j) <> -1 Then     ' si es diferente a la direccion indefinida
         a = Val(Format((Cos(Mat(i, j) * (Pi / 180)) * ((-1) * AncFlecha)) + ((i + 1 - PosX) *
(AncFlecha * 2)), "Fixed"))
         b = Val(Format((Sin(Mat(i, j) * (Pi / 180)) * (AncFlecha)) + ((j + 1 - PosY) *
(AncFlecha * 2)), "Fixed"))
         c = Val(Format((Cos(Mat(i, j) * (Pi / 180)) * (AncFlecha)) + ((i + 1 - PosX) *
(AncFlecha * 2)), "Fixed"))
         d = Val(Format((Sin(Mat(i, j) * (Pi / 180)) * ((-1) * AncFlecha)) + ((j + 1 - PosY) *
(AncFlecha * 2)), "Fixed"))
         If MatColor(i, j) = 0 Then
```

```
            For t = 0 To 1 Step 0.2
               x = a + (t * (c - a))
               y = b + (t * (d - b))
                ' se pinta la linea del vector
               formaImagen.Pic1.Line (x, y)-(x + 1, y + 1), QBColor(10), BF
            Next t
            ' se pinta la punta del vector
            formaImagen.Pic1.Circle (x, y), 15, QBColor(10)
         Else
         End If
      End If
   Next i
Next j

formaImagen.Show
Screen.MousePointer = vbDefault

End Sub
```

‘ Función que calcula el número de encierros

```
Function BuscaEncierros(Mat() As Integer, Ancho As Long, Largo As Long)

Dim i As Long
Dim j As Long
Dim MatTemp() As Integer
ReDim MatTemp(Ancho - 1, Largo - 1) As Integer
Dim encierros As Long
Dim MarcaPases As Boolean
Dim Bandera As Boolean

Dim rcode As Long
Dim tmpimage As imgdes

' asigna espacio para una imagen
rcode = allocimage(tmpimage, Ancho, Largo, 1)
If (rcode <> NO_ERROR) Then
  MsgBox "error: No hay suficiente memoria para continuar", vbCritical
  Exit Function
End If

MatTemp = Mat
encierros = 0

paso1:
Bandera = False
' se hace la contraccion combinatoria
MarcaPases = True
```

```
While MarcaPases
   MarcaPases = False
   For j = 1 To Largo - 2
      For i = 1 To Ancho - 2
         If MatTemp(i, j) = 0 Then
            If IndiceE(MatTemp, i, j, 1) = 1 Then
               MatTemp(i, j) = 1
               MarcaPases = True
            End If
         End If
      Next i
   Next j
Wend

' se eliminan los pixeles aislados
MarcaPases = True
While MarcaPases
   MarcaPases = False
   For j = 1 To Largo - 2
      For i = 1 To Ancho - 2
         If MatTemp(i, j) = 0 Then
            If VerticesE(MatTemp, i, j, 1) = 4 And AristasE(MatTemp, i, j, 1) = 4 Then
               MatTemp(i, j) = 1
               MarcaPases = True
               Bandera = True
            End If
         End If
      Next i
   Next j
Wend

' se hace la expansion combinatoria
MarcaPases = True
While MarcaPases
   MarcaPases = False
   For j = 1 To Largo - 2
      For i = 1 To Ancho - 2
         If MatTemp(i, j) = 1 Then
            If IndiceE(MatTemp, i, j) = 1 Then
               MatTemp(i, j) = 0
               MarcaPases = True
            End If
         End If
      Next i
   Next j
Wend
```

```
' se buscan encierros
For j = 1 To Largo - 2
   For i = 1 To Ancho - 2
      If MatTemp(i, j) = 1 Then
         If VerticesE(MatTemp, i, j) = 4 And AristasE(MatTemp, i, j) = 4 Then
            encierros = encierros + 1
         End If
      End If
   Next i
Next j

' se crea la imagen a partir de la matriz binaria
For j = 0 To Largo - 1
   For i = 0 To Ancho - 1
      If MatTemp(i, j) = 1 Then
         setpixelcolor tmpimage, i, j, 255
      Else
         setpixelcolor tmpimage, i, j, 0
      End If
   Next i
Next j
' se manda a presentar la imagen binaria en pantalla
CargaImagenEnForma tmpimage, Ancho, Largo, "Imagen de encierros"
' se libera la imagen
freeimage tmpimage

lblEncierros.Visible = True
lblEncierros.Caption = encierros & " encierro(s)"

End Function
```

‘ Función que calcula las aristas

```
Function AristasE(Mat() As Integer, x As Long, y As Long, Optional Cb As Integer = 0)
As Integer

Dim Cont As Integer

Cont = 0

If Mat(x - 1, y) = Cb Then
   Cont = Cont + 1
End If
If Mat(x, y + 1) = Cb Then
   Cont = Cont + 1
End If
If Mat(x + 1, y) = Cb Then
   Cont = Cont + 1
```

```
End If
If Mat(x, y - 1) = Cb Then
   Cont = Cont + 1
End If

AristasE = Cont

End Function

‘ Función que calcula los vértices
Function VerticesE(Mat() As Integer, x As Long, y As Long, Optional Cb As Integer = 0)
As Integer

Dim Cont As Integer

Cont = 0

If (Mat(x - 1, y) = Cb) Or (Mat(x - 1, y - 1) = Cb) Or (Mat(x, y - 1) = Cb) Then
   Cont = Cont + 1
End If
If (Mat(x - 1, y) = Cb) Or (Mat(x - 1, y + 1) = Cb) Or (Mat(x, y + 1) = Cb) Then
   Cont = Cont + 1
End If
If (Mat(x, y + 1) = Cb) Or (Mat(x + 1, y + 1) = Cb) Or (Mat(x + 1, y) = Cb) Then
   Cont = Cont + 1
End If
If (Mat(x + 1, y) = Cb) Or (Mat(x + 1, y - 1) = Cb) Or (Mat(x, y - 1) = Cb) Then
   Cont = Cont + 1
End If

VerticesE = Cont

End Function

‘ Función que calcula el índice
Function IndiceE(Mat() As Integer, x As Long, y As Long, Optional Cb As Integer = 0) As
Integer

Dim Cont As Integer

Cont = VerticesE(Mat, x, y, Cb) - AristasE(Mat, x, y, Cb)

IndiceE = Cont

End Function
```

‘ Función que convierte las direcciones en ángulos

```
Function ConvertirAangulo(Mat() As Integer, Ancho As Long, Largo As Long)

ReDim MatAng(Ancho - 1, Largo - 1) As Double
Dim i As Long
Dim j As Long

For j = 0 To Largo - 1
   For i = 0 To Ancho - 1
      Select Case Mat(i, j)
         Case 1
            MatAng(i, j) = 0
         Case 2
            MatAng(i, j) = 26.5
         Case 3
            MatAng(i, j) = 45
         Case 4
            MatAng(i, j) = 63.5
         Case 5
            MatAng(i, j) = 90
         Case 6
            MatAng(i, j) = 116.5
         Case 7
            MatAng(i, j) = 135
         Case 8
            MatAng(i, j) = 153.5
         Case 9
            MatAng(i, j) = -1
      End Select
   Next i
Next j

End Function
```

‘ Función que determina las direcciones predominantes

```
Function MatA8Reduc(Mat() As Integer, Ancho As Long, Alto As Long, PIniX As Long, _
   PIniY As Long) As Integer()

Dim ArrD(1 To 8) As Double
ReDim MatRes(Ancho - 1, Alto - 1) As Integer
Dim i As Long
Dim j As Long
Dim x As Long
Dim y As Long
Dim IniX As Long
Dim IniY As Long
```

```
Dim ArMin As Double
Dim ArMax As Double
Dim Umbral As Double
Dim VModa As Integer
Dim DirModa As Integer

IniY = PIniY
For j = 0 To Alto - 1
   IniX = PIniX
   For i = 0 To Ancho - 1
      ' se obtienen los datos a tratar
      For x = 1 To 8
         ArrD(x) = 0
      Next x
      For y = IniY To IniY + 7
         For x = IniX To IniX + 7
            Select Case Mat(x, y)
               Case 1
                  ArrD(1) = ArrD(1) + 0.015625
               Case 2
                  ArrD(2) = ArrD(2) + 0.015625
               Case 3
                  ArrD(3) = ArrD(3) + 0.015625
               Case 4
                  ArrD(4) = ArrD(4) + 0.015625
               Case 5
                  ArrD(5) = ArrD(5) + 0.015625
               Case 6
                  ArrD(6) = ArrD(6) + 0.015625
               Case 7
                  ArrD(7) = ArrD(7) + 0.015625
               Case 8
                  ArrD(8) = ArrD(8) + 0.015625
            End Select
         Next x
      Next y
      ' se localizan las probabilidades minima y maxima
      ArMin = ArrD(1)
      ArMax = ArrD(1)
      For x = 2 To 8
         If ArrD(x) < ArMin Then
            ArMin = ArrD(x)
         End If
         If ArrD(x) > ArMax Then
            ArMax = ArrD(x)
         End If
      Next x
```

```
        If (ArMax - ArMin) <= 0.1 Then  ' significa que hay mucho ruido
            MatRes(i, j) = 9            ' se asigna la direccion indefinida
        Else
            ' se define el umbral
            Umbral = ((ArMax - ArMin) * (3 / 4)) + ArMin
            ' se cuentan cuantas sobrepasan el umbral
            VModa = 0
            DirModa = 0
            For x = 1 To 8
                If ArrD(x) > Umbral Then
                    VModa = VModa + 1
                    DirModa = Int((DirModa + x) / VModa)
                End If
            Next x

            If VModa = 1 Or VModa = 2 Then  ' unimodal o bimodal
                MatRes(i, j) = DirModa      ' se asigna la direccion modal
            Else
                MatRes(i, j) = 9            ' se asigna la direccion indefinida
            End If
        End If

        IniX = IniX + 8
    Next i
    IniY = IniY + 8
Next j

MatA8Reduc = MatRes
End Function
```

‘ Función que realiza suavizamiento a la matriz de direcciones predominantes

```
Function Suavizamiento(Mat() As Double, MatBN() As Integer, AnchoF As Long, _
        AltoF As Long, Tipo As Integer, Optional Color As Integer) As Double()

Dim i As Long
Dim j As Long
Dim MatTemp() As Double
Dim RSen As Double
Dim RCos As Double
Dim x As Integer
Dim y As Integer
Dim angulo As Double
Dim Divisor As Integer
ReDim MatTemp(AnchoF - 1, AltoF - 1) As Double
```

```
For j = 1 To AltoF - 2
   For i = 1 To AnchoF - 2
      RSen = 0
      RCos = 0
      Divisor = 0
      If Tipo = 1 Then 'cuando se trate de la matriz de tamaño normal
         For y = 1 To 3
            For x = 1 To 3
               If Mat(x + i - 2, y + j - 2) <> -1 Then
                  angulo = 2 * Mat(x + i - 2, y + j - 2) * (Pi / 180)
                  RCos = RCos + Cos(angulo)
                  RSen = RSen + Sin(angulo)
                  Divisor = Divisor + 1
               End If
            Next x
         Next y
      ElseIf Tipo = 2 Then 'cuando se trate de la matriz reducida a 8
         For y = 1 To 3
            For x = 1 To 3
               If Mat(x + i - 2, y + j - 2) <> -1 Then
                  angulo = 2 * Mat(x + i - 2, y + j - 2) * (Pi / 180)
                  RCos = RCos + Cos(angulo)
                  RSen = RSen + Sin(angulo)
                  Divisor = Divisor + 1
               End If
            Next x
         Next y
      End If

      If Divisor > 0 Then
         RCos = Val(Format(RCos, "#.########")) / Divisor
         RSen = Val(Format(RSen, "#.########")) / Divisor
         If RCos = 0 Then
            MatTemp(i, j) = Mat(i, j)   ' se asigna el mismo valor del punto
         Else
            MatTemp(i, j) = Val(Format((Atn(RSen / RCos) * (180 / Pi))/2, "Fixed"))  ' se
regresa de radianes a grados
            If MatTemp(i, j) < 0 Then
               MatTemp(i, j) = 360 + MatTemp(i, j)     ' se asigna el nuevo valor del punto
            End If
         End If
      Else
         MatTemp(i, j) = Mat(i, j)   ' se asigna el mismo valor del punto
      End If
   Next i
Next j
```

```
Suavizamiento = MatTemp

End Function

‘ Función que muestra las direcciones predominantes a través de vectores en pantalla
Private Sub CargaImagenDirec88(Mat() As Double, Lancho As Long, Llargo As Long, _
      VTexto As String)

Screen.MousePointer = vbHourglass

Set formaImagen = New frmImagen
formaImagen.Hide
formaImagen.Caption = VTexto
formaImagen.width = Lancho * 15.4
formaImagen.Height = Llargo * 16.4
formaImagen.Pic1.width = Lancho * 300
formaImagen.Pic1.Height = Llargo * 300
formaImagen.Pic1.BackColor = vbBlack

Dim t As Double
Dim a As Long
Dim b As Long
Dim c As Long
Dim d As Long
Dim x As Long
Dim y As Long
Dim i As Long
Dim j As Long
Dim AncFlecha As Integer

AncFlecha = 100

For j = 0 To Llargo - 1
   For i = 0 To Lancho - 1
      If Mat(i, j) <> -1 Then         ' si es diferente a la direccion indefinida
         a = Val(Format((Cos(Mat(i, j) * (Pi / 180)) * ((-1) * AncFlecha)) + ((i + 1) *
(AncFlecha * 2)), "Fixed"))
         b = Val(Format((Sin(Mat(i, j) * (Pi / 180)) * (AncFlecha)) + ((j + 1) * (AncFlecha *
2)), "Fixed"))
         c = Val(Format((Cos(Mat(i, j) * (Pi / 180)) * (AncFlecha)) + ((i + 1) * (AncFlecha *
2)), "Fixed"))
         d = Val(Format((Sin(Mat(i, j) * (Pi / 180)) * ((-1) * AncFlecha)) + ((j + 1) *
(AncFlecha * 2)), "Fixed"))
         For t = 0 To 1 Step 0.2
            x = a + (t * (c - a))
            y = b + (t * (d - b))
            ' se pinta la linea del vector
```

```
            formaImagen.Pic1.Line (x, y)-(x + 1, y + 1), QBColor(10), BF
        Next t
        ' se pinta la punta del vector
        formaImagen.Pic1.Circle (x, y), 15, QBColor(10)
      End If
   Next i
Next j

formaImagen.Show
Screen.MousePointer = vbDefault

End Sub

‘ Función que calcula el Indice de Poincaré
Function EncCaract4b(Mat() As Double, AnchoF As Long, AltoF As Long)

Dim MatRes() As Double
Dim Ar(0 To 7) As Double
Dim ArDif(0 To 7) As Double
Dim i As Long
Dim j As Long
Dim x As Integer
Dim ContF As Integer    ' contador de factor mayor
Dim ContN As Integer    ' contador de angulos negativos
Dim AngRes As Double
Dim AngRad As Double
Dim K1 As Double
Dim K2 As Double
Dim K As Integer
Dim AncFlecha As Integer
Dim Core As Long
Dim Delta As Long

AncFlecha = 100
Core = 0
Delta = 0

ReDim MatRes(AnchoF - 3, AltoF - 3) As Double

For j = 1 To AltoF - 2
   For i = 1 To AnchoF - 2
      ContF = 0
      ContN = 0
      ' se obtiene el arreglo de angulos en el orden de las manecillas del reloj
      Ar(0) = Mat(i - 1, j - 1)
      Ar(1) = Mat(i, j - 1)
      Ar(2) = Mat(i + 1, j - 1)
```

```
Ar(3) = Mat(i + 1, j)
Ar(4) = Mat(i + 1, j + 1)
Ar(5) = Mat(i, j + 1)
Ar(6) = Mat(i - 1, j + 1)
Ar(7) = Mat(i - 1, j)

' se obtienen las variaciones
For x = 0 To 6
   AngRad = Val(Format((Ar(x + 1) - Ar(x)) * (Pi / 180), "Fixed"))
   K1 = (-0.5) - (AngRad / (2 * Pi))
   K2 = (0.5) - (AngRad / (2 * Pi))
   K = -5
   Do
      If K >= K1 And K <= K2 And K < 5 Then
         Exit Do
      Else
         K = K + 1
      End If
   Loop
   ArDif(x) = Val(Format((AngRad + (2 * Pi * K)) * (180 / Pi), "Fixed"))
Next x

AngRad = Val(Format((Ar(0) - Ar(7)) * (Pi / 180), "Fixed"))
K1 = (-0.5) - (AngRad / (2 * Pi))
K2 = (0.5) - (AngRad / (2 * Pi))
K = -5
Do
   If K >= K1 And K <= K2 And K < 5 Then
      Exit Do
   Else
      K = K + 1
   End If
Loop
ArDif(7) = Val(Format((AngRad + (2 * Pi * K)) * (180 / Pi), "Fixed"))

' se suman las variaciones
AngRes = 0
For x = 0 To 7
   AngRes = AngRes + ArDif(x)
Next x

MatRes(i - 1, j - 1) = Val(Format(AngRes / 360, "#.##"))

If MatRes(i - 1, j - 1) > 0.4 And MatRes(i - 1, j - 1) < 0.6 Then
   Core = Core + 1
ElseIf MatRes(i - 1, j - 1) < -0.4 And MatRes(i - 1, j - 1) > -0.6 Then
   Delta = Delta + 1
```

```
        End If

    Next i
Next j

' se crea un archivo con los resultados
Dim fso, txtfile
Dim cadena As String
Set fso = CreateObject("Scripting.FileSystemObject")
Set txtfile = fso.CreateTextFile("matrizCarac.txt", True)

For j = 0 To AltoF - 3
    cadena = ""
    For i = 0 To AnchoF - 3
        cadena = cadena & MatRes(i, j) & " |"
    Next i
    txtfile.Write (cadena)
    txtfile.WriteBlankLines (1)
Next j
txtfile.Close
Set fso = Nothing

lblCores.Caption = Core & " core(s)"
lblCores.Visible = True
lblDeltas.Caption = Delta & " delta(s)"
lblDeltas.Visible = True

End Function
```

BIBLIOGRAFÍA

1. A. Bykov, L. Zerkalov, Algorithm for Homotopy Classification of binary images, Pattern Recognition V. 29,N.4, 1996.
2. A. Bykov,L. Zerkalov, Mario Rodríguez, Index of a point of 3-D digital binary image and algorithm for computing its Euler characteristic, Pattern Recongnition, V. 32.,N.5,pp.845-850,1999.
3. B. Miller, Vital signs of identity, IEEE Spectrum, Vol.31, No. 2,pp 22-30, 1990.
4. Baldi, P. and Chauvin, Y., "Neural Networks for Fingerprint Recognition," *Neural Computation*, 5, pp. 402-418, 1993.
5. Blue, J.L., Candela, G.T., Gropher, P.J., Chellapna, R., and Wilson, C.L., *"Evaluation of Pattern Classifiers for Fingerprint and OCR Applications"*, Pattern Recognition,24(4), pp. 485-501, 1994.
6. Canny, J., "A computational Approach to Edge Detection," IEEE Trans. Pattern Analysis and Machine Intelligence, 8, pp.679-698, 1986.
7. Casadei, S. And Mitter, S., "Hierarchical Image Segmentation – Part 1: Detection of Regular Curves in a Vector Graph," *Int. J. Of Computer Vision*, 27(1), pp.71-100, 1998.
8. C.L. Wilson, Neural Network Fingerprint Classification, J. Artificial Neural Networks, V.1,pp. 1-25, 1993.
9. C.V.K. Rao, Type classification of fingerprint: A syntactic approach, IEEE Trans, V.2,pp 223-231, 1980.
10. Coetzee, L. and Botha, E.C., "Fingerprint Recognition in Low Quality Images," *Pattern Recognition*, 26, pp. 1441-1460, 1993.
11. Correa R.A., Identificación forense, Ed. Trillas, 1990,
12. Erol, A. *Automated Fingerprint Recognition*, Ph.D. Thesis Proposal Report, Dept. of Electrical and Electronics Eng., Middle East Technical University, Ankara, Turkey, 1998
13. Federal Bureau of Investigation, *The Science of Fingerprints: Classification and Uses*, Washington, DC. 1984,
14. Fitz, A.P. and Green, R.J., "Fingerprint Classification Using a Hexagonal Fast Fourier Transform," *Pattern Recognition*, 29(10), pp. 1587-1597, 1991.
15. Freeman, J.A. and Skapura, D.M., *Neural Networks: Algorithms, Applications, and Programming Techniques*, Addison-Wesley, 1991.
16. Galton, F., Finger Prints, Macmillan, London, 1892.
17. Halici, U. and Ongun, G., "Fingerprint Classification Trough Self-Organizing Feature Maps Modified to Treat Uncertainties," *Proc. Of the IEEE*, 84(10), pp. 1497-1512, 1996.
18. Halici, U., Erol, A., and Ongun, G., "Industrial Applications of the Hierarchical Neural Networks: Character Recognition and Fingerprint Classification," *Industrial Applications of Neural Networks*, L.C. Jain (Ed.) OCR Press, USA, 1998.
19. Halici, U. and Gelenbe, E., Lecture Notes on Neurocomputers, Middle East Technical University Online Courses, http://www.ii.metu.edu.tr/metuonline ,1998.
20. Henle Michael, *A Combinatorial Introduction topology*, Dover Publications, Inc., New York.
21. Henry, E.R., Classification and Uses of Finger Prints, Routledge, London,1900.

22. H.C. Lee, R.E Gaesslen, Advances in Fingerprint Technology, Elsevier, New York, 1990.
23. Hrechak, A.K. and McHugh, J.A., "Automated Fingerprint Recognition Using Structural Matching", *Pattern Recognition*, 23, pp. 893-904, 1990.
24. Jain, A.K., Hong, L., Pankanti, S., and Bolle, R., "An Identity-Authentication System Using Fingerprint", *Proc. Of the IEEE,85(9),*pp.1365-1288,1997.
25. Kalle Karu, *Fingerprint Classification,* Deparment of Computer Science, Michigan State University East lansing, MI 48824, 1996.
26. Karu, K. and Jain, A.K., "Fingerprint Classification," Pattern Recognition, 29(3), pp. 389-403, 1996.
27. Lee, H.C. and Gaenssley, R.E., *Advances in Fingerprint Technology*, New York, Elsevier, 1991.
28. M. Kawagoe, A. Tojo, Fingerprint Pattern Classification, Pattern Recognition Letters, V. 16, No.3,pp 295-303, 1984.
29. Moaver, B. and Fu, K.S., *"An Application of Stochastic Languages to Fingerprint Pattern Recognition," Pattern Recognition*, 8, pp. 173-179, 1976.
30. Moayer, B. and Fu, K.S. *"A Tree System Approach for Fingerprint Recognition,"IEEE Transactions on Pattern Analysis and Machine Intelligence,* 9(3). Pp. 376-387, 1986.
31. Paz T. E., Dactiloscopia y otras técnicas de identificación PATE-341012- 14-70.
32. Rao, T.C., "Feature Extraction for Fingerprint Classification," *Pattern Recognition*, 8, pp. 181-192, 1976.
33. Ratha, N.K., Karu, K., Shaoyun, C., and Jain, A.K., "A Real-Time Matching System for Large Fingerprint Databases," *IEEE Trans. Pattern. Analysis and Machine Intelligence*, 18(8), pp. 799-813, 1996.
34. Rosenfeld A., Kong T. Y., *Digital Topology: Introduction and Survey,* Computer Vision, Graphics and Image Processing, 48, 357-393, 1989.
35. Shchepin E. V., Nepomnyashchii G. M., On Topology Analysis of Images, Mezvuzovskii Abornik nauchinh trudo, Moscú, MIR, 1990.

HUAJUAPAN DE LEÓN, OAX.
SEPTIEMBRE 2002