SEGMENTACIÓN SUPERVISADA DE IMÁGENES DE HUELLAS DACTILARES

Jorge L. Aching Samatelo, David A. Rojas Vigo
jorge_aching@hotmail.com, davidrojasv@yahoo.es

Facultad de Ingeniería Electrónica
Universidad Nacional Mayor de San Marcos

RESUMEN: Se presenta un algoritmo mejorado de segmentación que separa la información útil de una huella dactilar con la ruidosa, importante paso que se tiene que realizar para el reconocimiento de huellas dactilares. El algoritmo está basado en la extracción de tres características de los pixeles: el valor promedio, la varianza y la coherencia. Se desarrolla una metodología para la determinación del conjunto de entrenamiento y la exclusión de las muestras redundantes a través del uso de redes neuronales auto-organizativas (SOM). Se comparan dos métodos de entrenamiento para obtener un clasificador lineal óptimo: perceptron y fuzzy perceptron. Como etapa de post-procesamiento, se utiliza morfología matemática para uniformizar las regiones clasificadas e incrementar el porcentaje de píxeles correctamente clasificados.

ABSTRACT: An improved algorithm of segmentation is presented that separates the useful information of a fingerprint with the noisy, important step that has to be carried out for an automatic fingerprint recognition system. The algorithm is based on the extraction of three pixels features: the mean, the variance and the coherence. A methodology is developed for the determination of the training set and the exclusion of the redundant samples through the self-organizing neural networks (SOM). Two methods of training are compared to obtain a good lineal classifier: perceptron and fuzzy perceptron. As post-processing stage, mathematical morphology is used to compact clusters and to increase in the rate of correctly classified pixels.

Palabras Claves: Segmentación, Huellas Dactilares, Valor Medio, Varianza, Coherencia, Redes Neuronales, SOM, Perceptron y Morfología Matemática.

I. INTRODUCCIÓN

Una imagen de huella dactilar, también referida en este artículo como impresión dactilar, generalmente está constituida de dos regiones de interés (ver figura 1), la región bien definida o primer plano y la región corrompida o segundo plano. En la primera región las estructuras de crestas y valles están claramente diferenciadas entre sí; esta región es usada en la etapa de reconocimiento, formándose con el primer contacto entre la yema de los dedos y la superficie del sensor. En la segunda región las estructuras de crestas y valles no están claramente diferenciadas debido a la presencia de ruido y distorsión en la imagen como bordes, manchas, arrugas etc; esta región es ignorada en la etapa de reconocimiento formándose en aquellas áreas donde no hubo un contacto adecuado de la yema del dedo con la superficie del sensor.

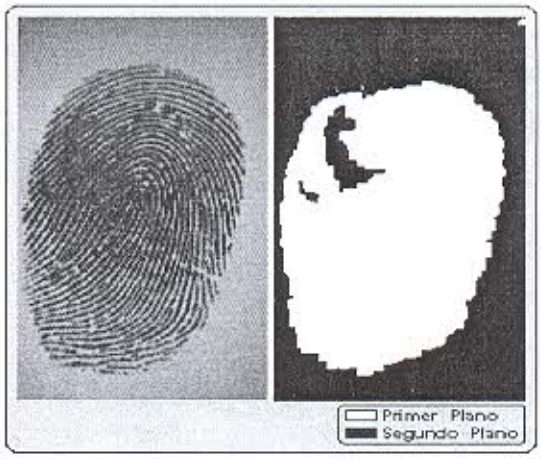

Figura 1 - Regiones en las huellas dactilares

La tarea del algoritmo de segmentación, es decidir cual parte de la imagen corresponde al primer o segundo plano. Una correcta segmentación es especialmente importante para la confiable extracción de las características. La mayoría de algoritmos de extracción de características extraen una porción de falsas características cuando es aplicado al segundo plano. El principal objetivo del algoritmo de segmentación es desechar el segundo plano y por tanto reducir el número de falsas características aumentando la robustez del sistema de reconocimiento.

Varios acercamientos a la segmentación de huellas dactilares han sido encontrados como referencias. En la publicación de Maio (ver referencias bibliográficas), la huella dactilar es particionada en bloques de 16x16, determinándose el promedio de la magnitud del gradiente de cada bloque; y puesto que una impresión dactilar está compuesta de crestas y valles semi-paralelas (direccionadas), el gradiente tendrá una alta magnitud en el primer plano y un valor bajo para el segundo plano. En Bazen se plantea una técnica de segmentación píxel por píxel de la imagen, basada en la coherencia; cuando el nivel de coherencia para cada píxel es menor que un umbral se considera segundo plano, en caso contrario es un primer plano incluyéndose al final una etapa de post procesamiento morfológico usado para eliminar las regiones mal clasificadas.

II. OBTENCIÓN DE LA IMAGEN SEGMENTADA

Como se expuso, dos son las vertientes al momento de obtener las características para la segmentación de la imagen, basados en bloques y píxeles. El presente trabajo se basa en la segunda opción por la exactitud demostrada teniendo como inconveniente el alto costo computacional, que será evitado con del uso de técnicas de clasificación simples pero robustas. Los algoritmos para la segmentación de la imagen, basados en píxeles presentan las siguientes fases:

1. Extracción de características: Selección de características en las cuales se basará la segmentación; además, obtención de conjuntos de entrenamiento con la asistencia de un experto forense.
2. Condensación del conjunto de entrenamiento: Eliminación de muestras de entrenamiento redundantes, utilizando para ello un algoritmo de condensación (agrupamiento) de datos.
3. Clasificación: Definición del clasificador que catalogará cada pixel como perteneciente al primer o segundo plano, tomando como base el conjunto de entrenamiento condensado en el paso anterior.
4. Post Procesamiento: Eliminación de las regiones mal clasificadas utilizando herramientas de morfología matemática de imágenes binarias.

A continuación se verá en detalle cada uno de los pasos anteriormente indicados.

2.1 Extracción de características

El primer paso en el desarrollo del algoritmo de segmentación es la selección de las características o propiedades de los pixeles individuales que puedan contener información útil para la correcta discriminación entre ambas regiones. Tomando en cuenta los trabajos de Bazen se selecciona tres características básicas: La coherencia, el valor promedio local y la varianza local. La coherencia y la varianza serán considerablemente altas en el primer plano, mientras que el valor promedio es en general bajo. Para reducir los efectos del ruido de la imagen, en cada pixel (u,v) las características son promediadas sobre una ventana Gaussiana $\mathbf{W}_{GP}$ de tamaño $N_{W_{GP}} x N_{W_{GP}}$, es decir:

$$\mathbf{W}_{GP}(u,v) = \frac{1}{2\pi\sigma^2}\exp\left\{\frac{1}{2\sigma^2}(u^2+v^2)\right\} \quad (1)$$

Siendo calculado el valor promedio local, y la varianza a través de las expresiones:

$$\text{Mean}[\mathbf{I}(u,v)] = \mathbf{I}(u,v) * \mathbf{W}_{GP}(u,v) \quad (2)$$

$$\text{Var}[\mathbf{I}(u,v)] = \text{Ener}[\mathbf{I}(u,v)] - \text{Mean}[\mathbf{I}(u,v)]^2 \quad (3)$$

Donde:

$$\text{Ener}[\mathbf{I}(u,v)] = \{\mathbf{I}(u,v)\}^2 * \mathbf{W}_{GP}(u,v) \quad (4)$$

La coherencia es calculada a través de las expresiones:

$$\text{Coh}[\mathbf{I}(u,v)] = \frac{\sqrt{(\mathbf{G}_{xx}(u,v) - \mathbf{G}_{yy}(u,v))^2 + 4\mathbf{G}_{xy}^2(u,v)}}{\mathbf{G}_{xx}(u,v) + \mathbf{G}_{yy}(u,v)}$$

Donde:

$$\mathbf{G}_{xx}(u,v) = \{\mathbf{G}_x(u,v)\}^2 * \mathbf{W}_{GP}(u,v)$$

$$\mathbf{G}_{yy}(u,v) = \{\mathbf{G}_y(u,v)\}^2 * \mathbf{W}_{GP}(u,v)$$

$$\mathbf{G}_{xy}(u,v) = \{\mathbf{G}_{xy}(u,v)\} * \mathbf{W}_{GP}(u,v)$$

Siendo $\mathbf{G}_x$ y $\mathbf{G}_y$ los componentes x e y del vector gradiente para cada pixel (u,v) de la imagen de la huella dactilar [Gonzáles et al, 1996]. Estas tres

características, definen la *observación* asociada a cada pixel, es decir:

$\mathbf{x} = [\text{Mean}[\mathbf{I}(u,v)] \quad \text{Var}[\mathbf{I}(u,v)] \quad \text{Coh}[\mathbf{I}(u,v)]]^T$;

$\mathbf{x} \in \mathbb{R}^3$

puesto que cada observación pertenecerá a una de las 2 posibles clases (primer plano o segundo plano) denotadas por $\Omega = \{w_{-1}, w_1\}$, se define un *conjunto de aprendizaje* o *conjunto de entrenamiento* S, al conjunto de pares $S = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \ldots, (\mathbf{x}_N, c_N)\}$, donde N es el número de observaciones y c_j es la clase cierta referida a la observación $\mathbf{x}_j$.

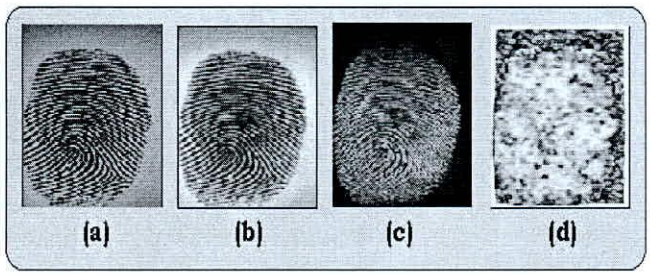

Figura 2 - Características de los pixeles para dos impresiones dactilares; (a) La impresión dactilar original (b) Valor promedio local (c) Desviación estándar local (d) Coherencia.

Es obvio que el conjunto de entrenamiento S estará compuesto por dos subconjuntos, donde, cada uno de ellos agrupa a todas las observaciones pertenecientes a una de las dos clases, es decir, $S = S_{-1} \cup S_1$, bajo esta perspectiva se podrá establecer la siguiente notación:

$\mathbf{X}_k$: denota todas las observaciones realizadas, vinculadas al conjunto de entrenamiento S_K. Es una matriz de tamaño $3xn_k$, tal que, k= -1,1.

$\mathbf{X}^i_{j:k}$:denota el valor de la observación del conjunto de entrenamiento S_k para la característica i-ésima, tal que, i=1,2,3 y $j = 1, 2, \ldots, n_k$

$\mathbf{X}_{j:k}$:denota a la j-ésima observación del conjunto de entrenamiento S_k, es un vector columna 3x1 de la matriz. $\mathbf{X}_k$

$\mathbf{X}^i_k$:denota los valores en las observaciones del conjunto entrenamiento S_k para la i-ésima característica, es un vector fila $1xn_k$ de la matriz $\mathbf{X}_k$.

Hasta este momento la determinación de las observaciones $\mathbf{X}_{j:k}$ está definida; pero no está especificada como determinar la clase a la que pertenece (w_{-1} ó w_1). Éste es en sí un proceso manual realizado por un experto forense; puesto que, utiliza su experiencia y conocimiento para realizar la segmentación. Lamentablemente no siempre el juicio humano es el preciso, lo cual lleva a establecer una metodología para determinar la mejor segmentación manual como se detalla:

1. Se selecciona aquellas imágenes de la base de datos que servirán para generar los conjuntos de entrenamiento.
2. A través de un proceso manual (utilizando para ello cualquier programa editor de imágenes en nuestro caso Corel 10.0) se obtiene para cada imagen seleccionada, una imagen binaria vinculada, denominada *imagen de regiones reconocibles* (ver figura 3), que especifica el primer plano (blanco) y el segundo plano (negro).
3. Se determina las observaciones para cada pixel de las imágenes seleccionadas y se genera los conjuntos de entrenamiento S para cada una de ellas.
4. Se obtiene las matrices $\mathbf{X}_{-1}$ y $\mathbf{X}_1$ vinculadas a cada una de las clases, y se determinó el histograma para cada $\mathbf{X}^i_k$; 'se observo' el nivel de solapamiento que presenta los histogramas vinculados a las pares de características $\mathbf{X}^i_{-1}$ y $\mathbf{X}^i_1$ (ver figura 4)
5. Si el solapamiento es muy marcado entonces volvemos al paso 2, si el problema persiste la imagen es desechada. caso contrario las muestras de entrenamiento obtenidas de la imagen son aceptadas, el proceso continua hasta que todas las imágenes seleccionadas sean analizadas.

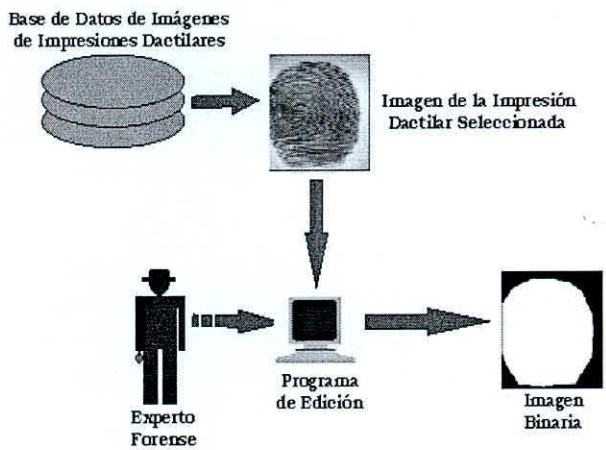

Figura 3 - Proceso manual realizado por un experto forense para la determinación de la imagen binaria.

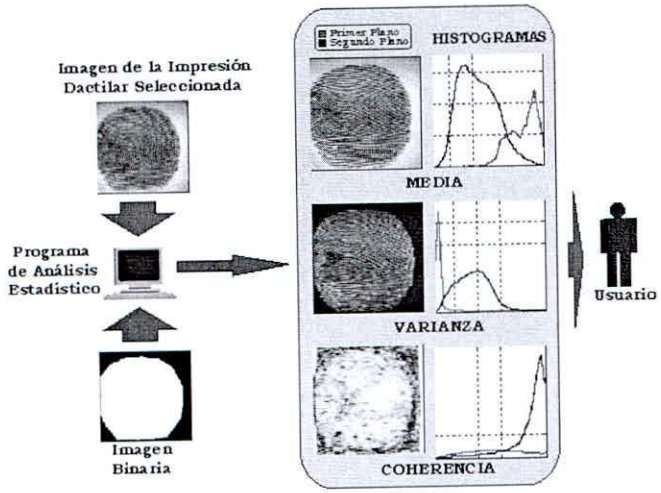

Figura 4 - Proceso manual realizado para la validación de la imagen binaria.

2.2. Condensación del Conjunto de Entrenamiento

Luego de obtener el conjunto de entrenamiento S, habiendo tomado las precauciones expuestas en el apartado anterior, es posible que este tenga muestras de entrenamiento innecesarias (redundantes) para el proceso de clasificación. Una manera de superar este inconveniente es a través de la inclusión de un procedimiento de condensado, puesto que este tiene como objetivo redefinir el conjunto de entrenamiento de forma que el conjunto resultante esté formado por las muestras que definen la frontera de decisión entre clases. En este artículo se usa el método de condensado adaptativo, basado en los mapas auto organizativos de características (SOM). Esta técnica propuesta inicialmente por Kohonen ha sido utilizada para realizar análisis de agrupamiento de datos (clustering) , y tal como es indicado en Cortijo es útil como herramienta para la condensación del conjunto de entrenamiento.

2.2.1. Mapas Autoorganizativos (SOM)

El mapa autoorganizativo de características (del inglés, self-organizing map) es una técnica de análisis de agrupamiento de espacios vectoriales [Platero, 1998] que tiene la particularidad de imponer sobre los agrupamientos convencionales ciertas relaciones de vecindad entre las muestras o representantes de los grupos conseguidos.

La arquitectura general de un SOM (Ver figura 5) puede describirse de la siguiente manera: Trátese de una red de dos capas con N_{int} neuronas en la capa de entrada y N_{out} neuronas en la capa de salida dispuestas en una malla usualmente conocida como *capa de competición*. La forma de la malla puede ser definida rectangular, hexagonal o incluso irregular. La i-ésima neurona de la capa de competición está conectada a cada neurona de la capa de entrada a través de N_{int} conexiones hacia adelante, implicando la asociación de un vector de pesos denotado por $\mathbf{U}_i = \left[\mu_{i1} \quad \mu_{i2} \quad L \quad \mu_{ij} \quad L \quad \mu_{iN_{int}}\right]^T$; además, se define alrededor de ella un conjunto de vecindad $\mathbf{N}_i$ que puede adoptar diversas formas, cuyo efecto es equivalente a conexiones laterales de inhibición implícitas (peso negativo), pues aunque no estén conectadas, las neuronas tendrán cierta influencia sobre sus vecinas. El valor asignado a los pesos $\mathbf{U}_i$ durante el proceso de aprendizaje de la red dependerá precisamente de esta interacción lateral. La entrada de información a la red es un vector $\mathbf{x}(t) = \left[x_1 \quad x_2 \quad L \quad x_{N_{int}}\right]^T$ que conecta en el caso más sencillo, en paralelo a la capa de entrada (puede haber otras configuraciones), notar que el peso μ_{ij} es debido a la conexión entre la neurona j-ésima de la capa de entrada y la neurona i-ésima de la capa de competición. La red funciona de manera que dado un vector de entrada $\mathbf{x}$, activa una única neurona en la capa de competición (la representa el punto del espacio más próximo al vector de entrada). En consecuencia, cada neurona de la capa de competición tiene asociado un punto en el espacio de representación cuyas coordenadas vienen dadas por su vector de pesos. A estos puntos se les denomina *prototipos*.

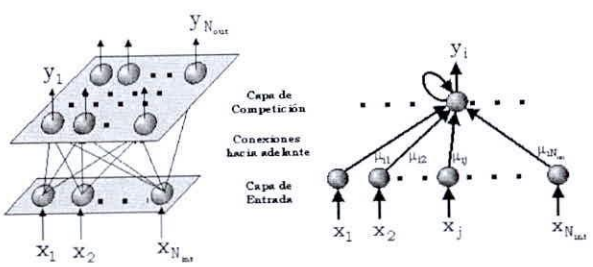

Figura 5 - Arquitectura del SOM

El algoritmo que actualiza los pesos $\mathbf{U}_i$ está compuesto por dos fases fundamentales: la presentación de las muestras de entrenamiento y la corrección de los pesos asociados a la vecindad de la neurona activa, tal que se consiga una malla ordenada topológicamente, en el que cada neurona está especialmente sensibilizada en un

dominio de la señal de entrada. Los pasos del algoritmo son:

1. Inicialización de los pesos: En un principio $t = t_o$, la red se encuentra en un estado inicial desordenado, tal que, los pesos $\mathbf{U}_i(t)$ toman valores aleatorios.
2. Presentación de la entrada: el vector de información $\mathbf{x}(t) \in \mathbb{R}^{N_{int}}$ es pasado a la capa de entrada.
3. Búsqueda de la neurona ganadora: El vector de información $\mathbf{x}(t)$ es comparado con todos los vectores de pesos $\mathbf{U}_i(t)$, hasta encontrar la neurona c cuyo vector de pesos $\mathbf{U}_c(t)$ esté más cercano a $\mathbf{x}(t)$. Se usa frecuentemente la métrica euclídea, pero puede utilizarse otro tipo de métrica.

$$\|\mathbf{x}(t) - \mathbf{U}_c(t)\| = \min_i \{\|\mathbf{x}(t) - \mathbf{U}_i(t)\|\}$$

4. Definición de la vecindad: Definimos una vecindad alrededor de la neurona ganadora c denotada por $N_c(t)$, ésta puede tener diferentes configuraciones (Ver figura 6), y cambia con el tiempo, tal que inicialmente incluye a todas las neuronas de la capa de competición y a medida que el algoritmo transcurre decrece en forma lineal. En la fase de convergencia la vecindad contendrá a la neurona ganadora y las vecinas más próximas.

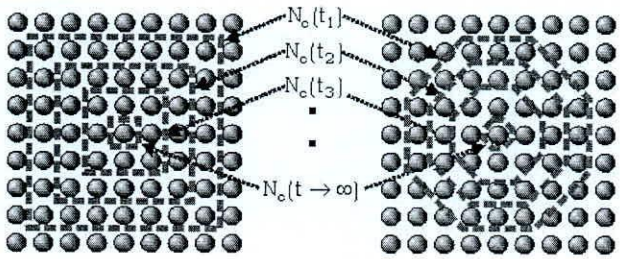

Figura 6 - Conjuntos de Vecindad

5. Modificación de los pesos: Los pesos $\mathbf{U}_i(t)$ son modificados según la ley de aprendizaje.

$$\mathbf{U}_i(t+1) = \begin{cases} \mathbf{U}_i(t) + \alpha(t).h_{ci}(t).(\mathbf{x}(t) - \mathbf{U}_i(t)) & i \in N_c(t) \\ \mathbf{U}_i(t) & i \notin N_c(t) \end{cases}$$

La función $h_{ci}(t)$ es conocida como función de retroalimentación de la neurona i a la neurona ganadora c. Para la convergencia del proceso de aprendizaje, es necesario que $h_{ci}(t) \to 0$ cuando $t \to \infty$. Usualmente viene definida por la función:

$$h_{ci}(t) = \exp\left(-\frac{\|\mathbf{U}_c(t) - \mathbf{U}_i(t)\|^2}{2\sigma^2(t)}\right)$$

en donde $\sigma(t)$ y $\alpha(t)$ es identificado como el factor radial y de aprendizaje respectivamente, $\alpha(t)$ varia entre 0 y 1 y decrece con el número de iteraciones (t) del proceso de entrenamiento.

6. Modificación de las tasas de aprendizaje: Se actualiza el factor radial y de aprendizaje, a través de las relaciones:

$$\alpha(t+1) = \alpha(t).f_\alpha$$
$$\sigma(t+1) = \sigma(t).f_\sigma$$

donde f_α es definida como la constante de aprendizaje, asumiendo un valor cercano a 1 y f_σ definida como la constante radial, también presenta un valor cercano a 1.

7. Volver al paso 2: Para alcanzar un estado de "orden" en la malla de competición, el algoritmo regresa al nivel 2 hasta que las neuronas estén adecuadamente afinadas.

Es normal utilizar una estrategia separada en la ordenación y en la convergencia. Así durante la fase de aprendizaje habrá una evolución tanto de la función de vecindad como del factor de aprendizaje, mientras en la convergencia se mantiene la vecindad decreciendo $\alpha(t)$.

2.2.2. Condensación de las observaciones

Luego de obtener el conjunto de observaciones $\mathbf{X}_{-1}$ y $\mathbf{X}_1$ éstas serán condensadas a través de dos SOM, de manera que cada uno condense las observaciones de cada clase generando los conjuntos de observaciones condensadas $\mathbf{Y}_{-1}$ y $\mathbf{Y}_1$ (figura 7), las cuales conformarán el conjunto de aprendizaje para la clasificación, es decir:

$$S_{\mathbf{Y}} = S_{\mathbf{Y}_{-1}} \cup S_{\mathbf{Y}_1}$$
$$= \{(\mathbf{y}_j, c_j) \mid (\mathbf{y}_j \in \mathbf{Y}_{-1} \vee \mathbf{y}_j \in \mathbf{Y}_1) \wedge (c_j \in \Omega)\}$$

Tal que: $j = 1, 2, .., n_{\mathbf{Y}}$, siendo $n_{\mathbf{Y}}$ la suma del número de observaciones de $S_{\mathbf{Y}_{-1}}$ y $S_{\mathbf{Y}_1}$. La topología de los SOM queda determinada por el número de características en la que esta basada la segmentación (capa de entrada) y por el número de muestras representativas por imagen deseada (capa competitiva). En este trabajo el número de características es tres y el número de muestras representativas por imagen 16x16, a través de esta elección se logra un mayor incremento en el porcentaje de una clasificación exitosa.

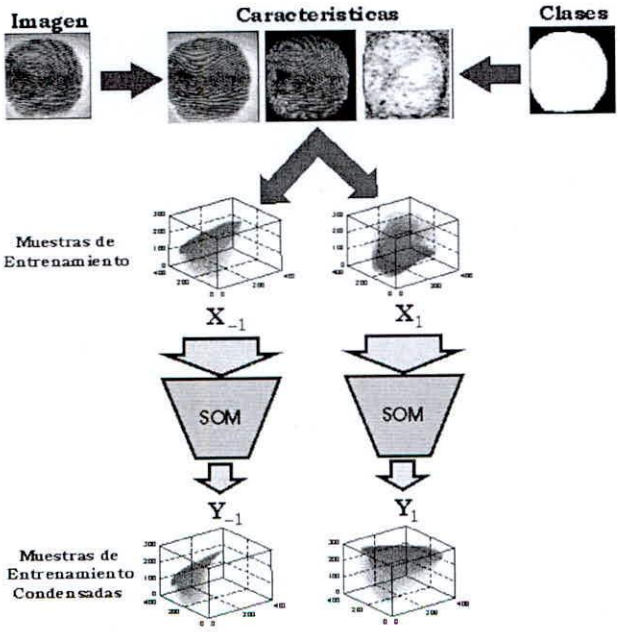

Figura 7 - Diagrama de los SOM para cada clase

2.3. Clasificación

El próximo paso consiste en entrenar un clasificador que realice la tarea de segmentación, es decir, que clasifique cada pixel de la imagen como perteneciente al primer o segundo plano. Como se desea clasificar todos los pixeles, es importante utilizar un algoritmo con la menor complejidad computacional posible. Tales como los clasificadores lineales supervisados que permiten clasificar espacios linealmente separables, puesto que obviamente el problema de segmentación es no lineal y con traslape entre clases. Se deberá utilizar alguna medida de incertidumbre que de alguna manera incluya la pertenencia de clase.

2.3.1. Clasificadores lineales supervisados

Un clasificador lineal denominado también neurona lineal, divide el espacio de salida en dos regiones a través de un hiperplano de separación lineal, cuyo límite está basado en el umbral obtenido de la combinación lineal del vector de entrada con un vector de coeficientes. Para toda j-ésima observación $\mathbf{y}_j$ se definirá el *vector aumentado de observaciones* $\bar{\mathbf{y}}_j$ como: $\bar{\mathbf{y}}_j = [\mathbf{y}_j \; 1]^T$; este vector será multiplicado por otro vector conocido como vector de *pesos* $\mathbf{w}_p$ que determina la ecuación del hiperplano de separación lineal. El cuarto elemento de $\bar{\mathbf{y}}_j$ usualmente denominado *bias* es utilizado para aumentar un grado de libertad al aprendizaje del clasificador, la clase asignada al vector aumentado de observaciones $\bar{\mathbf{y}}_j$ estará dada por la siguiente función de decisión:

$$\%_j = \begin{cases} w_1 & \text{signo}(\bar{\mathbf{y}}_j * \mathbf{w}_p^T) > 0 \\ w_{-1} & \text{caso contrario} \end{cases}$$

Entonces, esta función discrimina correctamente si $c_j = \%_j$. En general, el clasificador es entrenado variando el vector de pesos de acuerdo con el error entre la clase calculada $\%_j$ y la clase verdadera c_j; definida como:

$$e_j = c_j - \%_j$$

Encontrar un hiperplano óptimo que minimice el error no es un problema sencillo [Skapura et al, 1993], puesto que durante el entrenamiento el clasificador estará influenciado por las observaciones de clases traslapadas y no lineales, por tanto serán analizados y evaluados dos métodos de entrenamiento para el clasificador: La regla de aprendizaje del perceptron y del fuzzy perceptron, tal como se describe a continuación.

2.3.2. El Aprendizaje del Perceptron

El algoritmo original de aprendizaje por corrección de errores fue desarrollado por Rosenblatt. Este requiere que sus resultados sean evaluados y se realicen las oportunas modificaciones en sus parámetros libres si fuera necesario, en este caso el error es únicamente diferente de cero para vectores de entrada que son incorrectamente clasificados. Los pasos del algoritmo son.

1. Inicialización del vector de pesos $\mathbf{w}_p$
2. Presentación de un nuevo par $(\mathbf{y}_j, c_j)$ del conjunto de entrenamiento condensado S_Y.
3. Cálculo de la clase asignada $\%_j$.
4. Si el error es diferente de cero, se actualiza el vector de pesos, de acuerdo con:

$$\mathbf{w}_p(t+1) = \mathbf{w}_p(t) + \alpha * e_j(t) * \bar{\mathbf{y}}_j^T$$

   donde, t es el paso de la iteración, α es un factor de ganancia en el rango de 0 a 1.0, que define la velocidad de aprendizaje.
5. Si no se ha alcanzado el número máximo de iteraciones o el error es menor que un valor umbral, volver al paso 2.

El perceptron puede ser utilizado como máquina universal de aprendizaje [Hilera y Martínez, 1995]. Sin embargo, provee convergencia solamente para dos clases linealmente separables. Es decir, que para problemas que no son linealmente separables éste algoritmo no terminará en un número finito de iteracciones. Una forma común para superar el problema de la terminación del algoritmo es forzar a que se detenga estableciendo un número máximo de interacciones permitidas (paso 5).

2.3.3. El Aprendizaje Fuzzy Perceptron

El Algoritmo de Aprendizaje Fuzzy Perceptron propuesto por Keller es una herramienta de gran utilidad, pues provee una forma elegante de tratar con el problema de la terminación del algoritmo y al mismo tiempo disminuye la influencia negativa producida por el traslape entre clases para espacios no linealmente separables, debido a que utiliza la teoría de conjuntos fuzzy para obtener un indicador cuantitativo de incertidumbre sobre el grado de pertenencia a la clase verdadera, de tal forma que los vectores que presentan un alto grado de incertidumbre tengan menos influencia sobre los resultados del entrenamiento. Keller y Hunt establecieron la siguiente forma de asignar los valores de membresía fuzzy para una bipartición .

$$u_1(\mathbf{y}_j) = \begin{cases} 0.5 + \dfrac{\exp(f(d_{-1}(\mathbf{y}_j) - d_1(\mathbf{y}_j))/d) - \exp(-f)}{2(\exp(f) - \exp(-f))} & \mathbf{y}_j \in \mathbf{Y}_1 \\ 1 - u_{-1}(\mathbf{y}_j) & \mathbf{y}_j \in \mathbf{Y}_{-1} \end{cases}$$

$$u_{-1}(\mathbf{y}_j) = \begin{cases} 0.5 + \dfrac{\exp(f(d_1(\mathbf{y}_j) - d_{-1}(\mathbf{y}_j))/d) - \exp(-f)}{2(\exp(f) - \exp(-f))} & \mathbf{y}_j \in \mathbf{Y}_{-1} \\ 1 - u_1(\mathbf{y}_j) & \mathbf{y}_j \in \mathbf{Y}_1 \end{cases}$$

Puesto que el problema de clasificación es de dos clases, se asume que $u_1(\mathbf{y}_j) + u_{-1}(\mathbf{y}_j) = 1$.

Donde :

$u_k(\mathbf{y}_j) \in \leq 0,1 \geq$: Es la función de membresía que describe el grado en que la j-ésima observación del conjunto de entrenamiento $S_{\mathbf{Y}}$ pertenece a la k-ésima clase.

$d_k(\mathbf{y}) = \|\mathbf{y} - \mu_k\|$: Es la distancia entre el vector $\mathbf{y}$ y el vector promedio μ_k de la k-ésima clase, siendo,

$$\mu_k = \frac{1}{n_1}\sum_{j=1}^{n_c} \mathbf{Y}_{j:k} = [\mu_k^{Mean} \quad \mu_k^{Var} \quad \mu_k^{Coh}]^T$$

$d = \|\mu_1 - \mu_{-1}\|$: Es la distancia entre los dos vectores promedio de cada clase, es decir:

f: es una constante que controla el índice en que los grados de membresía decrecen hacia 0.5.

El algoritmo de aprendizaje Fuzzy Perceptron consta de los siguientes pasos:

1. Inicialización del vector de pesos $\mathbf{w}_p$
2. Presentación de un nuevo par $(\mathbf{y}_j, c_j)$ del conjunto de entrenamiento condensado $S_{\mathbf{Y}}$.
3. Cálculo de la clase asignada $\%_j$.
4. Si el error es diferente de cero, se actualiza el vector de pesos, de acuerdo con:

$$\mathbf{w}_p(t+1) = \mathbf{w}_p(t) + \alpha * \left|u_1(\mathbf{y}_j) - u_{-1}(\mathbf{y}_j)\right|^m * e_k(t) * \overline{\mathbf{y}}_j^T$$

5. Si no se ha alcanzado el número máximo de iteraciones, volver al paso 2.

α es la tasa de aprendizaje y m es una constante positiva que controla el grado de "incertidumbre", entonces la observación $\mathbf{y}_j$ más "incierta" tendrá menos influencia en la adaptación de los pesos. Este método de entrenamiento puede interpretarse como una generalización del algoritmo de aprendizaje del perceptron, puesto que si suponemos que no hay incertidumbre de las muestras de entrenamiento, entonces la expresión: $\left|u_1(\mathbf{y}_j) - u_{-1}(\mathbf{y}_j)\right|^m$ se reduce a 1, obteniéndose la regla de aprendizaje del perceptron estándar. Los pasos del entrenamiento del Fuzzy Perceptron son similares a los pasos 1 al 3 del Perceptron estándar. Sin embargo, el cuarto paso incluye el cálculo de los valores de membresía de las observaciones. Además, el quinto paso relacionado con la terminación del algoritmo incluye que el valor de membresía esté dentro del siguiente rango: [0.5-ξ, 0.5+ξ] donde ξ es una constante determinada experimentalmente. Por lo tanto, la condición adicional de terminación es que si los errores son producidos por las muestras cuyos valores de membresía están dentro de éste rango, entonces el algoritmo debería terminar. El parámetro ξ debe ser adecuadamente seleccionado, de tal forma que las muestras de entrenamiento cuyos valores de membresía que están fuera del rango sean linealmente separables.

2.4. Post Procesamiento

Tal como muestra la figura 9, la clasificación de los pixeles tendrá cierto nivel de error, como consecuencia de que cualquier perceptron es un clasificador lineal;

reflejándose en pequeñas áreas espurias de una clase presentes dentro de una área más grande de la otra clase (pequeñas manchas oscuras o bordes no definidos sobre el área definida como primer plano).

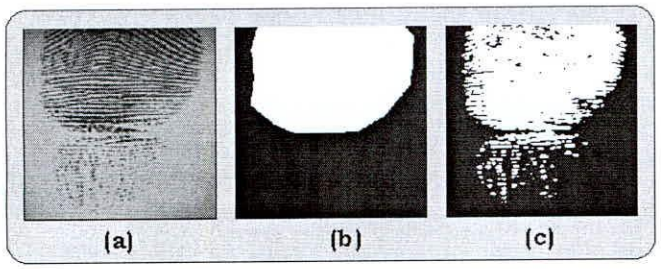

Figura 9 - (a) Imagen de la huella dactilar. (b) Imagen de regiones reconocibles determinada por el experto forense (c) Imagen de regiones reconocibles determinada por el perceptron

Una manera de eliminar estas regiones mal clasificadas es a través de técnicas basadas en procesamiento de imágenes binarias; definiendo como la imagen binaria a tratar, a la imagen de regiones reconocibles determinada por el clasificador, donde el primer plano vendrá representado por el objeto binario (color blanco) y el segundo plano vendrá representado por el fondo (color negro). Bajo esta perspectiva la corrección de la clasificación de pixeles se resume al mejoramiento de bordes del objeto y la eliminación de huecos. Como es recomendado en Bazen, se aplica morfología matemática para tales labores. La morfología matemática, que comenzó a finales de los años sesenta, forma de alguna manera un cuerpo separado dentro del análisis de imágenes. Sus principales protagonistas son Matheron y Serra. La morfología matemática, es un sistema algebraico de operadores composicionales, que, cuando actúan sobre formas complejas son capaces de descomponerlas en sus partes más significativas y separarlas de las partes que le son extrañas. Un sistema de operadores de este tipo y su composición permite que las formas subyacentes sean identificadas y reconstruidas de forma óptima a partir de sus formas distorsionadas y ruidosas. Además, permite que cada forma sea entendida en función de una descomposición, siendo cada entidad de esa descomposición una forma simple apropiada. Las operaciones morfológicas pueden simplificar los datos de la imagen, preservar las características esenciales y eliminar aspectos irrelevantes. Dos de tales operadores son el operador de apertura y el operador de cierre; la apertura generalmente suaviza el contorno de una imagen, rompe istmos estrechos y elimina protuberancias delgadas. El cierre también tiende a suavizar secciones de contornos pero, en oposición a la apertura generalmente fusiona separaciones estrechas y entrantes delgadas y profundas, elimina pequeños huecos y rellena agujeros del contorno; por lo expuesto se puede concluir que:

1. Los conjuntos de pixeles aislados que son incorrectamente asignados al primer plano son removidos por medio de una operación de apertura.
2. Los conjuntos de pixeles aislados que son incorrectamente asignados al segundo plano son removidos por una operación de cierre.

Los resultados obtenidos después de realizar las operaciones de apertura y cierre son evidentes en la figura 10, puede apreciarse, que la mayoría de distorsiones son eliminadas. Debido a la propia naturaleza de la operación de apertura, ésta podrá generar en la imagen final de regiones reconocibles "huecos" (conjunto de pixeles aislados pertenecientes al primer plano inmersos en el segundo plano); debido a ello se incluye una etapa de filtrado binario, eliminando todos aquellos componentes conectados cuya área sea menor que un umbral (300) determinado empíricamente.

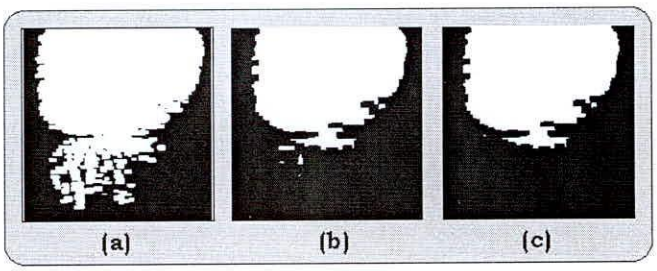

Figura 10 - (a)Aplicación del operador de apertura (b) Aplicación del operador de cierre (c) Eliminación de huecos.

III. RESULTADOS EXPERIMENTALES

Todos los Algoritmos fueron implementados utilizando el paquete de software Matlab 5.3, y corridos en un procesador Pentium I de 133 Mhz.

3.1. Obtención de las muestras de entrenamiento

Para este experimento utiliza la base de datos 2 del FVC2002. De ella seleccionamos las imágenes 10_3.tif, 10_4.tif, 10_5.tif, 11_7.tif, 12_7.tif, 13_6.tif, 14_2.tif, 14_6.tif, 18_4.tif, 18_6.tif, 19_6.tif, 22_1.tif, 22_6.tif,

32_3.tif, 36_2.tif, 40_1.tif, 42_3.tif, 52_8.tif, 53_2.tif, 55_7.tif,59_2.tif, 60_2.tif, 63_6.tif, 64_5.tif, 66_6.tif, 67_7.tif, 72_6.tif, 73_6.tif, 76_2.tif, 84_5.tif, 95_6.tif, 99_1.tif, 100_3.tif, 100_6.tif. Para cada una de estas imágenes siguiendo el procedimiento explicado en el apartado 2.1 se determina la imagen binaria de zonas reconocibles y no reconocibles y las 3 características asociadas a cada pixel. Puesto que son 34 imágenes de tamaño 560x296, se generaran 5 635 840 muestras de entrenamiento, que conforma el corpus inicial. Para eliminar las muestras redundantes, se condensa el corpus inicial tal como es explicado en el apartado 2.2 a través del uso de dos SOM de 3 neuronas en la capa de entrada y 16x16 neuronas en la capa de competición, obteniéndose un total de 8 704 observaciones por clase. En la figura 11 se muestran histogramas vinculados a cada característica, tanto para las muestras sin condensar (a) como condensadas (b), se puede observar que después de la condensación el traslapamiento entre clases es menor, y los histogramas tienen una apariencia suavizada puesto que únicamente quedan las muestras de entrenamiento más representativas (analizar el concepto de correlado, es decir muestras altamente correladas)

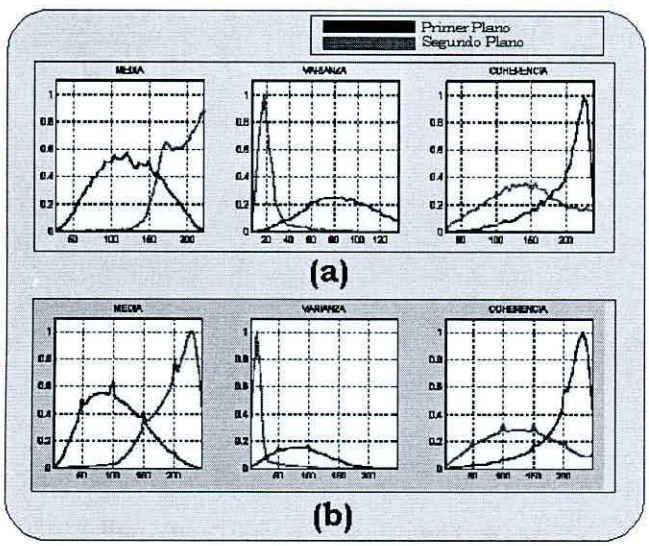

Figura 11 - Histogramas para cada característica (a) Antes de la condensación (b) Después de la condensación.

3.2. Rendimiento del Clasificador

Se implementaron los dos algoritmos para el entrenamiento de los clasificadores lineales descritos en el apartado 2.3. Teniendo como conjunto de entrenamiento a S_Y, se obtuvo una considerable mejora en el tiempo de entrenamiento, pues comparado con las 10^6 épocas que fueron necesarias para la convergencia del algoritmo del perceptron publicada en Bazen, éste número fue reducido al orden de 10^4 para ambos métodos.

Para reducir la probabilididad que el entrenamiento Fuzzy Perceptron quede estancado en un mínimo local se utilizó un algoritmo por "lotes", es decir se presentan todas las entradas y con sus respectivas salidas se calculan los errores parciales para todo el corpus de entrenamiento, luego éstos son promediados para obtener un error global y se actualiza el vector de pesos solamente a través de éste error, repitiéndose estos pasos hasta que cumpla con el criterio de terminación. Además, se determinaron experimentalmente las constantes f=0.05 y ξ=0.1 a través del método de prueba y error, puesto que ambas permiten al usuario definir el grado de influencia positiva o negativa que poseen las muestras cercanas a la frontera de decisión. Los resultados muestran que en un menor número de épocas el Fuzzy Perceptron provee un hiperplano de separación más óptimo que el obtenido por el Perceptron, disminuyendo su error promedio en aproximadamente 1.5%. La figura 12, muestra una comparación del error de clasificación entre el Perceptron y el Fuzzy Perceptron para una impresión dactilar.

A través de una simple inspección visual se puede deducir que éstos algoritmos proveen resultados bastante satisfactorios. El efecto de la morfología es mostrado en la figura 10.

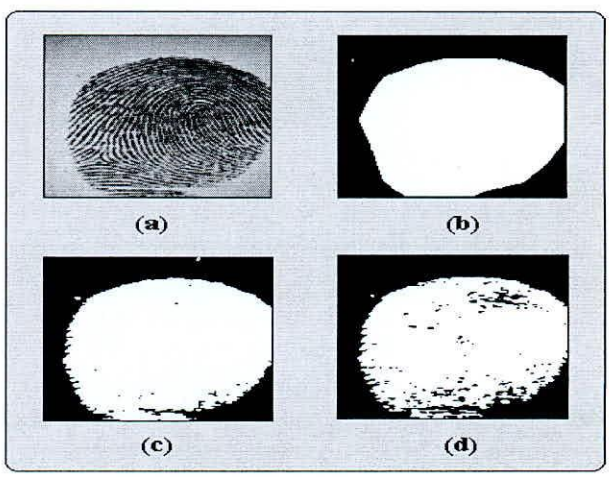

Figura 12 - (a) Impresión dactilar (b) Imagen de regiones (c) Fuzzy Perceptron (d) Perceptron

Aparte de la inspección humana, se tiene varias maneras de evaluar cuantitativamente los resultados del

algoritmo de segmentación. Por ejemplo, el número de errores ocurridos en la clasificación puede ser usado como una medida de rendimiento. Esta es exactamente la medida utilizada durante el entrenamiento, determinándose que era del 6% al 4% para una clasificación óptima. Otras posibilidades son evaluar el algoritmo de segmentación por el conteo del número de características de huellas dactilares perdidas y falsas tales como las minucias y puntos singulares.

La figura 13, muestra la matriz de confusión [Aching y Rojas, 2003] del clasificador perceptron y del fuzzy perceptron para la imagen 60_2.tif.

Perceptron

Clase Verdadera	Clase hipotetica: 1er Plano	Clase hipotetica: 2do Plano
1er Plano	74025	2146
2do Plano	2613	86976

Tasa Clasificacion: 97.129%

Fuzzy Perceptron

Clase Verdadera	Clase hipotetica: 1er Plano	Clase hipotetica: 2do Plano
1er Plano	72854	3317
2do Plano	762	88827

Tasa Clasificacion: 97.5392%

Figura 13 – Matriz de confusión

La figura 14, muestra gráficamente el proceso de segmentación realizado por el algoritmo, los bordes resaltados en color celeste y rojo son los límites obtenidos, el área fuera de éstos representa el segundo plano, y el área interior el primer plano.

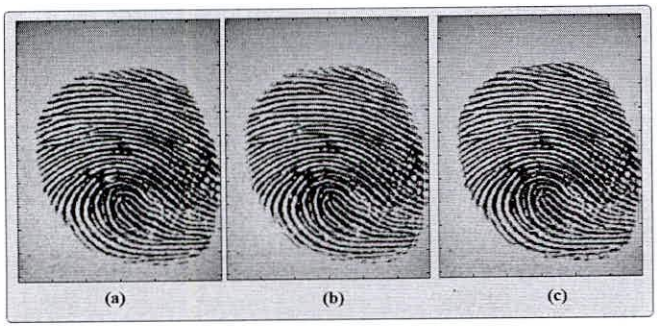

Figura 14 - Resultados de la segmentación (a) imagen original (b) Clasificador Perceptron (c) Clasificador Fuzzy Perceptron.

IV. CONCLUSIONES

La condensación del conjunto de entrenamiento es un paso importante para la segmentación, pues aumenta la probabilidad de convergencia del algoritmo de entrenamiento del clasificador en un número reducido de iteraciones.

El método Fuzzy Perceptron provee una forma robusta y eficiente de encontrar un hiperplano de separación lineal, sobretodo cuando el problema posee muestras de entrenamiento no lineales y con traslape entre clases, puesto que permite al usuario definir el grado de influencia que tendrán las muestras con mayor incertidumbre o cercanas a la frontera de decisión.

REFERENCIAS

Aching, J.L. y D. Rojas (2003), Algoritmos para reconocimiento de imágenes de huellas dactilares", Revista Electrónica UNMSM, págs.11-20.

Bazen, A. y S., Gerez (2000). "Directional field computation for fingerprints based on the principal component analysis of local gradients", 11th Annual Workshop on Circuts, Systems and Signal Processing, Veldhoven, Netherlands.

Bazen, A. y S., Gerez (2001). Segmentation of fingerprint images, In Proc. ProRISC2001, 12th Annual Wokshop on Circuits, Systems and Signal Processing, Veldhoven, Netherlands,.

Bazen, A., S., Klein y R., Veldhuis (2001). Fingerprint image Segmentation Based on Hidden Markov Models.

Cortijo, F. (1997). Estudio Comparativo De Métodos De Clasificación de Imágenes Multibanda, Tesis Doctoral, Universidad de Granada, Departamento de Ciencias de la Computación e Inteligencia Artificial, España, Granada.

Gonzáles, R., R.,Woods (1996). Tratamiento digital de imágenes. Addison-Wesley/Diaz de Santos, Primera edición en Español.

Kohonen, T. (1982). Self-organized formation of topologically correct features maps, Biological Cybernetics, Vol. 43, pp. 59-69.

Keller D. (1985). Hunt Incorporating fuzzy membership functions into the perceptron algorithm, IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. PAMI-7, pp. 693-699.

Maio, D. y D., Maltoni. (1997). Direct Gray-Scale Minutiae Detection in Fingerprints, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 19, no. 1.

Platero, C. (1998). Inspección Automatizada de Superficies Homogéneas Mediante Visión Artificial, Tesis Doctoral, Universidad

Politécnica de Madrid, Departamento de Electrónica, Automática e Informática industrial, España, Granada.

Skapura, D. y J., Freeman (1993). Redes Neuronales, Algoritmos, Aplicaciones y Técnicas de Programación, Addison-Wesley/Diaz de Santos, primera edición en Español.