ISSN 1870-4069

Sistema de reconocimiento de placas vehiculares haciendo uso de modelos asociativos

Luis Edgar Alanís Carranza, Moisés Vicente Márquez Olivera,
Viridiana Gudelia Hernández Herrera Olivera, Octavio Sánchez García

Instituto Politécnico Nacional, Centro de Investigación e Innovación Tecnológica,
Ciudad de México, México

Vehicle Plates Recognition System using Associative Models

Abstract. This study used of the associative memories alpha-Beta (AMαβ) applied to the problem of the recognition of vehicle plates. We propose using a Haar Cascade classifiers with the AdaBoost in the detection of the plate finally proposes to use the AMαβ as a classifier to determine the alpha-numeric characters inside the plate. The AMαβ are trained with characteristic vectors extracted from images of an own database. In the detection stage the system can locate 98.3% of 707 car images and for the recognition stage the accuracy was 93%.

Keywords: inteligencia artificial, placas vehiculares, memorias asociativas, Haar Cascade, reconocimiento de patrones, caracteres alfanuméricos.

1. Introducción

Hoy en día la visión artificial ha tenido una amplia gama de aplicaciones en el reconocimiento de objetos, una de ellas son los sistemas de reconocimiento de placas vehiculares (LPR), los cuales buscan reconocer las placas de forma no intrusiva; es decir, que emplean dispositivos externos al auto para monitorear y ubicar el auto. El dispositivo por excelencia de estos sistemas son las cámaras, las cuales adquieren imágenes estáticas o dinámicas; en ocasiones las cámaras son mejoradas con tecnología infrarroja para oprimir factores que afectan a la detección, como son las condiciones ambientales o la hora del día. Idealmente, un sistema LPR debería estar optimizado con algoritmos de reconocimiento que le permitan trabajar en tiempo real y bajo condiciones reales del campo de aplicación, logrando así brindar una respuesta asertiva e instantánea del análisis realizado en una escena vial. Favorablemente, los avances tecnológicos en hardware, en conjunto con los nuevos modelos de Inteligencia Artificial, han dado como resultado el desarrollo de nuevos algoritmos, nuevos e híbridos, que buscan la solución del problema tratando diferentes problemas como iluminación, ángulos, tiempo real, seguimiento, entre muchos otros [1, 2].

Tabrizi y Cavus [1] utilizaron el método de detección de bordes utilizando un filtro Prewit y posteriormente utilizaron un algoritmo híbrido de inteligencia artificial para el

reconocimiento de los caracteres SVM y KNN, finalmente obtuvo un índice de asertividad del 94 al 97 %. Patel [2] utilizó el filtro de la mediana para eliminar ruido que pudiera afectar el reconocer la placa vehicular, posteriormente utilizó las redes neuronales para reconocer los caracteres. Ktata et al. [3] desarrollaron un sistema de reconocimiento de placas utilizando el algoritmo skew correction para detectar zonas que posiblemente serían una placa vehicular y finalmente usaron redes neuronales para reconocerla, obtuvieron un índice de asertividad del 90.78%. Sedighi y Vafadust [4] emplearon filtro Gaussiano para la eliminación de ruido y redes neuronales para tratar de reconocer un total de 500 imágenes con placas vehiculares, de las cuales solo el 90.05% fueron reconocidas.

Este trabajo describe el desarrollo de un Reconocimiento automático de matrículas (ANPR) utilizando memorias asociativas Alpha-Beta. El sistema propuesto consta de seis etapas. La primera etapa es la adquisición de la imagen, donde se obtendrá la imagen del automóvil. En la segunda etapa, se desarrollan algoritmos de procesamiento de imágenes digitales para mejorar la calidad de la imagen, posteriormente, se aplica el algoritmo de Haar Feature-based Cascade (HFC) con AdaBoost para detectar dentro de la imagen dónde se encuentra la placa. La tercera etapa es la segmentación que permitirá extraer la imagen de la matrícula, así como los caracteres intrínsecos. La cuarta etapa consiste en extraer información relevante sobre los caracteres segmentados y aquellos que pueden clasificarse, en la siguiente etapa se busca reconocer de forma particular los caracteres alfanuméricos para que en la última etapa se haga una interpretación general de la identificación de la placa vehicular.

2. Metodología

2.1. Proceso General

Los sistemas LPR de forma general presentan fases claramente definidas, los cuales fueron resumidas por Gonzales y Woods [7], en esta sección se mencionan los pasos a seguir, siendo la adquisición el proceso en donde se realiza la obtención de las imágenes estáticas o dinámicas (video), posteriormente durante el preprocesamiento se busca mejorar la calidad; la segmentación es la encargada de extraer de la imagen mejorada solo las área que son de interés para analizar, consecuentemente, se realiza la extracción de características esenciales de los caracteres a reconocer, los cuales son procesados por el algoritmo de Inteligencia Artificial (IA) supervisado previamente entrenado, el modelo de IA da como resultado cuál es el carácter que se encuentra contenido en cada una de las imágenes segmentadas, finalmente y una vez terminando con el análisis de reconocimiento a la salida se tiene la interpretación de placa objetivo.

2.1.1. Adquisición

Como se muestra en la Fig. 1, el primer paso es la adquisición, por lo que para el presente trabajo se propuso crear una base de datos propia, la cual está integrada con 2000 imágenes con modo de color RGB tomadas con una cámara GoPro HERO4 Session con un tamaño de 1280x720 pixeles, las cuales contienen automóviles que portan placas vehiculares de la zona metropolitana.

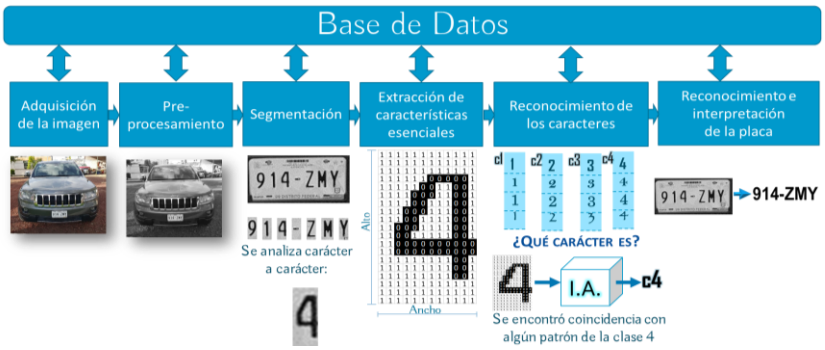

Fig. 1. Proceso general para el reconocimiento de placas vehiculares.

Fig. 1. Imagen del automóvil incluido en la base de datos creada.

2.1.2. Preprocesamiento

En esta etapa se utilizarán todos aquellos algoritmos para mejorar la calidad de la imagen y obtener un mejor resultado durante la etapa de segmentación. El primer preprocesamiento propuesto convertir de modo de color RGB a escala de grises para ello se utilizó el modelo YIQ o YUV [6], el cual consiste en multiplicar cada canal de color de la imagen por un cierto porcentaje y guardar todos esos valores en uno solo canal como se muestra en la siguiente expresión:

$$Y = R * 0.3 + G * 0.59 + B * 0.11. \quad (1)$$

Finalmente se obtendrá como salida una imagen a escala de grises con una profundidad de 8 bits (Fig. 2). Esto reducirá el costo computacional, debido a que los procesamientos que se realicen solo trabajaran en un solo canal.

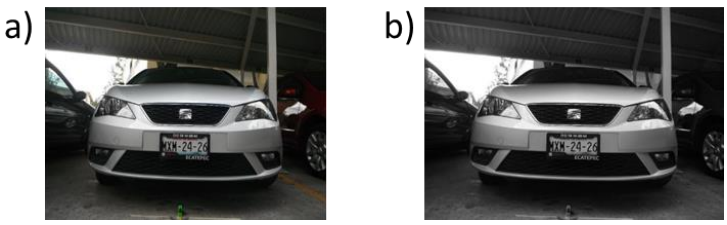

Fig. 2. Conversión escala de grises. a) Imagen original, b) Imagen en escala de grises.

Ecualización

Ahora bien, existen factores que puedan afectar en el reconocimiento de la placa y uno de ellos es el contraste ya que la imagen puede estar muy obscura o con brillo y esto se debe a que la imagen no tiene una buena distribución de los pixeles más obscuros o blancos en toda la imagen y no sería posible llegar a ver la placa vehicular al momento de buscarla por el algoritmo de segmentación. Por lo cual se utilizará el algoritmo de ecualización usado en [7], el cual consiste en mover los pixeles más oscuros o con más brillo en toda la imagen (Fig. 3).

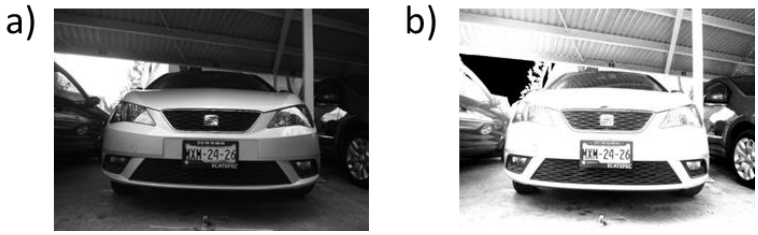

Fig. 3. Ecualización. a) Imagen en escala de grises, b) Imagen en escala de grises ecualizada.

Filtro gaussiano

La imagen puede llegar a tener ruido debido a los posibles defectos que llegue a tener la cámara, para poder llegar a eliminar parte de ese ruido y suavizar la imagen es necesario utilizar una máscara o también conocido en la literatura como Kernel. Este se desplazará por toda la imagen y realizará una operación matemática para tener un nuevo valor el cual será agregado a una nueva imagen.

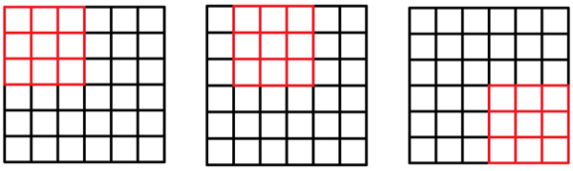

Fig. 4. Imagen de 6 x 6 usando un Kernel de 3 x 3.

En la Fig. 4 se puede observar que hay una imagen con resolución de 6 x 6 y tiene colocada sobre ella un Kernel de 3 x 3. No obstante, antes de realizar este procedimiento es necesario obtener los valores del Kernel y se pueden obtener por medio de la siguiente ecuación.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}}. \quad (2)$$

En este caso se utilizó una varianza (σ) de 0.25 ya que es la más utilizada por la literatura y las coordenadas x, y son las del Kernel. Finalmente se tiene un Kernel con los siguientes valores Fig. 5.

0.0183	0.1353	0.0183
0.1353	1	0.1353
0.0183	0.1353	0.0183

Fig. 5. Kernel con los valores

Una vez obtenido este Kernel se realiza la siguiente ecuación para la obtención de los nuevos pixeles de cada recorrido:

$$g(x,y) = h(x,y) \times f(x,y) = \sum_{i=0}^{i=2} \sum_{j=0}^{j=2} f(i,j)h(x,y). \quad (3)$$

Teniendo finalmente como salida una imagen con filtro Gaussiano Fig. 6.

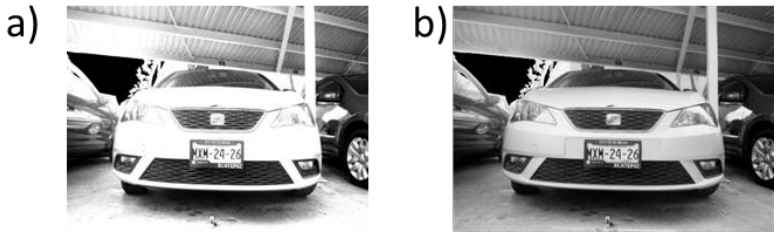

Fig. 6. Filtro gaussiano. a) Imagen ecualizada, b) Imagen con filtro Gaussiano.

2.2. Segmentación

En la etapa de segmentación se utilizará un algoritmo para detectar la zona que posiblemente contenga una placa vehicular. Para ello se utilizó el algoritmo Haar Feature-based Cascade (HFC) creado por Viola y Jones [8], el cual consiste en la extracción de características por medio de una imagen integral. Una vez que se tengan todas las características llamadas Haar, estas serán utilizadas para los clasificadores pequeños de AdaBoost para generar un clasificador fuerte. Este discriminará imágenes que no contengan una placa vehicular, a lo cual le llamaremos zonas candidatas. Adicionalmente, para mejorar los resultados de la clasificación se utilizó un árbol de clasificadores el cual es la combinación de varios AdaBoost (véase la Fig. 7).

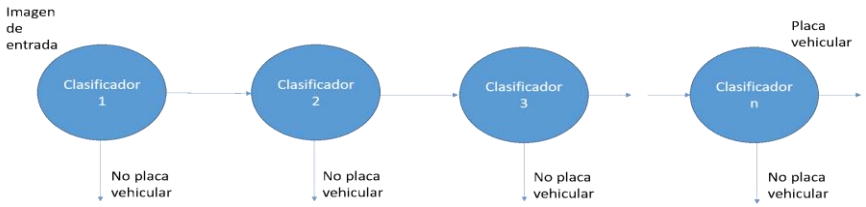

Fig. 7. Árbol de clasificadores (AdaBoost).

El algoritmo HLF fue programado en C# bajo la IDE de Visual Studio, el cual fue entrenado con 2,000 imágenes positivas (placas vehiculares) y 4,000 imágenes negativas las cuales no son placas; por ejemplo, letreros, retrovisores, árboles, etc. Una

vez entrenado el algoritmo, el resultado de detección obtenido se presenta en la Fig. 8, en donde el que el HLF detecta la placa dentro de la imagen.

Fig. 8. Detección de placa vehicular con HLF.

2.3. Extracción de características

Para comenzar con el proceso de extracción característica se propone realizar una binarización utilizando un umbral dinámico y después se realiza un escalamiento para normalizarla utilizando el método de interpolación lineal (Fig. 9)

Fig. 9. Imagen con Binarización y Rescalamiento.

Posteriormente, se buscan los caracteres para segmentarlos, para ello se propone realizar una búsqueda por contornos que cumplieran con las características de un carácter considerando la proporción existente entre el alto y el ancho, bajo este parámetro de normalización, es posible detectar caracteres más altos o anchos dependiendo del tamaño de la placa vehicular segmentada, el resultado se muestra en al Fig. 10.

Una vez detectados y segmentados los caracteres de forma individual se extrae el vector característico, el cual se forma a partir de concatenar los valores binarios (0 y 1) que tienen cada pixel de la imagen del carácter (véase la Fig. 11).

Fig. 10. Placa vehicular con caracteres segmentados.

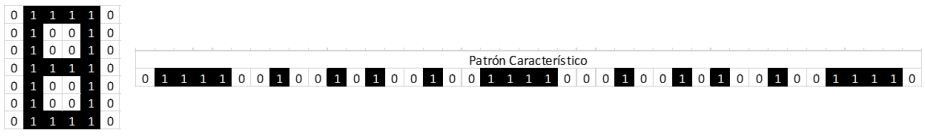

Fig. 11. Extracción de características y formación de patrón característico.

2.4. Reconocimiento

Para esta etapa se implementó el algoritmo de memorias asociativas Alfa-Beta [10] [11] [12] para reconocer cada uno de los caracteres. La idea básica de una memoria asociativa αβ es que es una matriz que almacena información de patrones que previamente aprendió usando el operador α y posteriormente los recupera utilizando el operador β (véase la Fig. 12).

Fig. 12. Matriz M generada por patrones de entrada X.

La memoria asociativa αβ está constituida de tres fases.

Fase de aprendizaje: En esta fase se genera la memoria asociativa o la matriz M por medio de los patrones de entrada; para ello se utilizará el patrón concatenado representados como x_i^n, con lo que se alimenta la memoria. Al comenzar con la generación de esta matriz se obtienen las matrices transpuestas de cada uno de los patrones (véase la Fig. 4).

$$[\mathrm{M}']^{\mathrm{n}} = [\mathrm{x}^{\mathrm{n}}\ \alpha\ (\mathrm{y}^{\mathrm{m}})^{\mathrm{t}}] = \begin{bmatrix} \mathrm{x}_1^{\mathrm{n}} \\ \mathrm{x}_2^{\mathrm{n}} \\ \vdots \\ . \\ \mathrm{x}_{\mathrm{l}}^{\mathrm{n}} \end{bmatrix} \alpha [\mathrm{y}_1^{\mathrm{m}}\ \ \mathrm{y}_2^{\mathrm{m}}\ \ \ldots\ \ \mathrm{y}_{\mathrm{l}}^{\mathrm{m}}]. \tag{4}$$

Tabla 1. Operador binario α.

X	**Y**	$\boldsymbol{\alpha(x,y)}$
0	0	1
0	1	0
1	0	2
1	1	1

Una vez que se tiene la transpuesta de cada uno de los patrones se procede a utilizar el operador α de la Tabla 1 para obtener una nueva matriz por cada patrón.

Finalmente se busca el número más grande utilizando el operador max (V) de cada una de las matrices para obtener solamente una sola, esta será la memoria asociativa.

$$\mathrm{v}_{\mathrm{ij}} = \bigvee_{\mathrm{n}=1}^{\mathrm{p}} [\boldsymbol{\alpha}(\mathrm{x}_{\mathrm{i}}^{\mathrm{n}}, (\mathrm{y}_{\mathrm{i}}^{\mathrm{m}})^{\mathbf{t}})]^{\mathrm{n}}, \tag{4}$$

$$\boldsymbol{M} = \begin{bmatrix} v_{11} & v_{12} & \ldots & v_{1l} \\ v_{21} & v_{22} & \ldots & v_{2l} \\ . & . & \ldots & . \\ . & . & \ldots & . \\ . & . & \ldots & . \\ v_{l1} & v_{2l} & \ldots & v_{ll} \end{bmatrix}. \tag{5}$$

Fase de recuperación: Recupera patrones previamente aprendidos usando la matriz de la ecuación (6), una de las ventajas de este algoritmo es su factor de olvido del 0%. Esto quiere decir que no olvidan el patrón previamente aprendido. Se realiza una comparación del patrón de entrada con la matriz de aprendizaje utilizando el operador β de la Tabla 2. Una vez que se tienen la matriz resultante del operador β ecuación (7) se procede a obtener el operador min (⋀) de cada fila de la matriz para así obtener como salida un patrón recuperado ecuación (8).

Tabla 2. Operador β.

X	Y	$\boldsymbol{\beta(x,y)}$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

$$\boldsymbol{y_i^m} = \begin{bmatrix} \boldsymbol{\beta}(v_{11}, x_1^n) & \boldsymbol{\beta}(v_{12}, x_1^n) & \dots & \boldsymbol{\beta}(v_{1l}, x_1^n) \\ \boldsymbol{\beta}(v_{21}, x_2^n) & \boldsymbol{\beta}(v_{22}, x_2^n) & \dots & \boldsymbol{\beta}(v_{2l}, x_2^n) \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ \boldsymbol{\beta}(v_{l1}, x_l^n) & \boldsymbol{\beta}(v_{2l}, x_l^n) & \dots & \boldsymbol{\beta}(v_{ll}, x_l^n) \end{bmatrix}, \tag{6}$$

$$\boldsymbol{y_i^m} = \begin{bmatrix} \bigwedge_{i=1}^{l} \boldsymbol{\beta}(v_{1i}, x_i^n) \\ \bigwedge_{i=1}^{l} \boldsymbol{\beta}(v_{2i}, x_i^n) \\ \cdot \\ \cdot \\ \cdot \\ \bigwedge_{i=1}^{l} \boldsymbol{\beta}(v_{li}, x_i^n) \end{bmatrix}. \tag{7}$$

Fase de prueba: La memoria asociativa tratará de asociar un patrón no aprendido con algunos de los que aprendió previamente, utilizando el mismo método de la fase de recuperación.

3. Resultados

Se obtuvieron resultados mostrados en las tablas 3 y 4 como parte de la detección y reconocimiento respectivamente de placas vehiculares.

Tabla 3. Resultados de la parte de detección.

Número de AdaBoost	Imágenes positivas	Imágenes negativas	Imágenes probadas	Placas detectadas	Índice de asertividad
16	2000	4000	707	585	82.7 %
18	2000	4000	707	695	98.3 %
20	2000	4000	707	590	83.4 %
22	2000	4000	707	572	80.9 %

En la Tabla 4 se puede apreciar que al utilizar 18 AdaBoost se obtuvieron los mejores resultados al momento de la detección de la placa. En la base de datos se probaron 707 imágenes de carros y el 98 .3% de las placas fueron detectadas.

Tabla 4. Resultados de la parte de reconocimiento.

Número de patrones probados	Tamaño del patrón	Resolución	Tiempo de procesamiento	Índice de asertividad
3200	1, 024	32 x 32	0.254s	82 %
3200	20, 000	100 x 200	0.615s	93 %

Se utilizaron 100 patrones por cada carácter existente en una placa vehicular en México. En la Tabla 4 se puede observar que mientras mejor resolución tenía la imagen del patrón con el que se alimentaba la memoria asociativa se obtenían mejores resultados, no obstante, el tiempo de procesamiento es mayor.

3.1.1. Preprocesamiento

Todos los algoritmos fueron programados en lenguaje C# bajo el entorno de Visual Studio en una PC con Windows 8, 8GB de RAM, y un procesador Intel Core i7-4700HQ a 2.40GHz. Se realizó una interfaz gráfica para mostrar los resultados del sistema la cual se muestra en la Fig. 13.

Fig. 13. Matriz *M* generada por patrones de entrada X.

4. Conclusiones

Durante la fase de detección de placas empleando el HFC se obtuvo un índice de asertividad máximo de 98.3%, para ello fue necesario probar con diferente número de Adaboost, esto para determinar el entrenamiento y sobre entrenamiento del algoritmo.

El HFC depende del número y tipo de imágenes positivas y negativas con las que se entrenan influyen en el índice de precisión del algoritmo durante la fase de prueba.

Las memorias asociativas Alfa-Beta son capaces de reconocer caracteres alfanuméricos en imágenes vehiculares al entrenar el algoritmo con el método de validación K fold cross validation con k=10, en el que se obtuvo un índice de exactitud máxima de 93% utilizando una resolución de 100 x 200 pixeles.

La resolución de la imagen influye directamente en el tiempo de procesamiento, no obstante, si se selecciona una resolución muy baja para las imágenes buscando mejorar el tiempo de respuesta, el índice de precisión obtenido por las memorias Alfa-Beta también se ve reflejado de forma negativa, ya que la pérdida de información en el patrón da como resultado que las memorias se confundan entre caracteres similares.

Referencias

1. Atiwadkar, A., Mahajan, S., Lande, T., Patil, K.: Vehicle License Plate Detection: A Survey. International Research Journal of Engineering and Technology (IRJET), vol. 2, nº 8, pp. 354–360 (2015)
2. Du, S., Ibrahim, M., Shehata, M., Badawy, W.: Automatic license plate recognition (ALPR): A state-of-the-art review. IEEE Transactions on circuits and systems for video technology, vol. 23, nº 2, pp. 311–325 (2013)
3. Tabrizi, S. S., Cavus, N.: A hybrid KNN-SVM model for Iranian license plate recognition. Procedia Computer Science, vol. 102, pp. 588–594 (2016)
4. Patel, S. G.: Vehicle license plate recognition using morphology and neural network. International Journal on Cybernetics & Informatics, vol. 2, pp. 1–7 (2013)
5. Ktata, S., Khadhraoui, T., Benzarti, F., Amiri, H.: Tunisian License Plate Number Recognition. Procedia Computer Science, vol. 73, pp. 312–319 (2015)
6. Sedighi, A., Vafadust, M.: A new and robust method for character segmentation and recognition in license plate images. Expert Systems with Applications, vol. 38, pp. 13497–13504 (2011)
7. Gonzalez, R., Wood, R. E.: Tratamiento digital de imágenes. vol. 3, Addison-Wesley New York (1996)
8. Zhang, X., Xu, F., Su, Y.: Research on the License Plate Recognition based on MATLAB. Procedia Engineering, vol. 15, pp. 1330–1334 (2011)
9. Shidore, M. M., Narote, S. P.: Number plate recognition for indian vehicles. IJCSNS International Journal of Computer Science and Network Security, vol. 11, pp. 143–146, (2011)
10. Viola, P., Jones, M. J.: Robust real-time face detection. International journal of computer vision, vol. 57, pp. 137–154 (2004)
11. Yánez-Márquez, C.: Associative memories based on order relations and binary operators (in Spanish). Phd Thesis (2000)
12. Yáñez, C., Díaz de León, J.: Memorias Autoasociativas Morfológicas max: condiciones suficientes para convergencia, aprendizaje y recuperación de patrones. IT-I77, Serie Azul, *CIC-IPN*, Mexico (2003b) ISBN, pp. 970–36 (2003)
13. Yáñez, C., Díaz-de-León, J. L.: Introducción a las memorias asociativas. Research in Computing Science, México (2003)