TRABAJO ESPECIAL DE GRADO

DESARROLLO DE UN MODELO COMPUTACIONAL PARA UN SISTEMA DE RECONOCIMIENTO DE MATRÍCULAS A TRAVÉS DE UNA IMAGEN PROVENIENTE DE UNA CÁMARA DE TRÁFICO

Tutor Industrial: Gustavo Attias
Prof. Guía: William La Cruz

Presentado ante la ilustre
Universidad Central de Venezuela
por el Br. Arcadio Fernández
para optar al titulo de
ingeniero electricista

Caracas, 2011

CONSTANCIA DE APROBACIÓN

Caracas, 26 de octubre de 2011

Los abajo firmantes, miembros del Jurado designado por el Consejo de Escuela de Ingeniería Eléctrica, para evaluar el Trabajo Especial de Grado presentado por el Bachiller Arcadio Fernández, titulado:

"DESARROLLO DE UN MODELO COMPUTACIONAL PARA UN SISTEMA DE RECONOCIMIENTO DE MATRÍCULAS A TRAVÉS DE UNA IMAGEN PROVENIENTE DE UNA CÁMARA DE TRÁFICO"

Consideran que el mismo cumple con los requisitos exigidos por el plan de estudios conducente al Título de Ingeniero Electricista en la mención de Electrónica y Control, y sin que ello signifique que se hacen solidarios con las ideas expuestas por el autor, lo declaran APROBADO.

Profa. Tamara Pérez
Jurado

Prof. Alejandro González
Jurado

Prof. William La Cruz
Profesor guía

DEDICATORIA

A mi familia; mis padres María Cristina Núñez de Fernández y Arcadio Fernández, que estuvieron pendientes de mí desde mi nacimiento, que me ayudaron a llevar mi carrera a la culminación, estuvieron siempre pendiente de mi y pusieron su esfuerzo y empeño para que no me faltara nada. A mi hermano Arcadio Miguel Fernández, mi única compañía desde siempre, al que he tratado de serle ejemplo con mis logros.

A mi amada novia Eurelis Marrero, quien me apoyo en estos de meses de novios para sacar este trabajo adelante, una gran persona llena de amor, inteligente, trabajadora, de gran alegría, buena amiga, buena compañera, amorosa y respetuosa. Dentro de un tiempo seremos colegas dios mediante. Te estoy esperando mi amor.. Te Amo!!

AGRADECIMIENTOS

En primer lugar le doy gracias a dios todo poderoso, por permitirme tener vida, darme salud, y permitir alcanzar mis metas, sobre todo la de graduarme de Ingeniero Electricista.

A mi familia; mis padres María Cristina Núñez de Fernández y Arcadio Fernández, por darme la vida, creer en mi, animarme en los momentos difíciles, y darme todo lo necesario para haber llegado hasta aquí. A mi hermano Arcadio Miguel Fernández, por escucharme y estar ahí para compartir.

A mi novia Eurelis Marrero, por estar conmigo, acompañarme en mis trasnochos, darme ánimos en los momentos de mayor apuro, por transmitirme confianza en forma de cariño, cosa que formo parte importante de este trabajo.

A mi tutor industrial, Gustavo Attias, por brindarme su ayuda incondicional en la realización de mi tesis, por brindarme confianza y con ello la libertad que necesitaba para cumplir con mis objetivos, y lo más importante por ser mas que un tutor; por ser un amigo.

A mi profesor guía, mi tutor académico, William La Cruz, por el esfuerzo realizado para que mi trabajo estuviese apto académicamente, su apoyo y la gran dedicación que tiene con sus alumnos,por enseñarme que todo en esta vida se puede hacer, siempre y cuando tengamos las ganas, habilidades y el conocimiento.

A mis compañeros del CENDIT, Dionmar Valera, Tomas Delgado, Javiér Castellanos, Gerson Castellanos, Eduardo Mata, Héctor Bolívar, Hon Moy, Damián Rossi y el resto de investigadores, por su apoyo, su camaradería y su amistad.

Fernández N. , Arcadio J.

DESARROLLO DE UN MODELO COMPUTACIONAL PARA UN SISTEMA DE RECONOCIMIENTO DE MATRÍCULAS A TRAVÉS DE UNA IMAGEN PROVENIENTE DE UNA CÁMARA DE TRÁFICO

Tutor Académico Profesor William La Cruz. Tutor Industrial Ing. Gustavo Attias. Tesis. Caracas. U.C.V. Facultad de Ingeniería. Escuela de Ingeniería Eléctrica. Ingeniero Electricista. Opción: Electrónica y Control. Institución: Cendit,2011,105h + anexos.

Palabras Claves: Procesamiento de imágenes, filtrado de imágenes, sistema OCR, reconocimiento de caracteres, binarización, redes neuronales, perceptrón multicapa.

Resumen. En el presente trabajo se propone el diseño de un modelo computacional a partir de un una imagen proveniente de una cámara de trafico. Para ello se hizo una búsqueda de las distintas fases que componen un sistema de reconocimiento de caracteres (OCR). Luego se hizo una búsqueda de distintos métodos y algoritmos aplicables en cada una de estas fases y se seleccionaron los más convenientes basados en pruebas hechas a cada método. El OCR consta de cuatro etapas. Para la primera etapa, extracción de área útil, se utiliza un filtro de gradiente morfológico para resaltar los bordes y así obtener un área útil para facilitar la ubicación de la matrícula. En la etapa extracción de matrícula, se utilizó un método morfológico combinado que consiste en aplicar filtros no lineales de dilatación y erosión con distintos elementos estructurantes para discriminar la ubicación de la matrícula. Para la etapa caracterización, se utilizó el método de características binarias, para reducir el tamaño de información que se debe procesar en la etapa de reconocimiento. En la etapa reconocimiento, se utilizaron redes neuronales por requerimientos de los objetivos, siendo el modelo perceptrón multicapa el utilizado por sus bondades en el reconocimiento de patrones. Finalmente, este modelo fue probado utilizando la herramienta computacional Scilab obteniéndose aciertos mayores al 77%, con lo cual se puede concluir que los objetivos del trabajo, se alcanzaron satisfactoriamente.

ÍNDICE GENERAL

ÍNDICES DE TABLA

ÍNDICE DE FIGURAS

ÍNDICES DE DIAGRAMAS

INTRODUCCIÓN

El exceso de velocidad es una de las principales causas de accidentes de tránsito. Algunas de las redes viales están rodeadas de zonas habitadas por comunidades, ubicándose dentro: viviendas, instituciones educativas, locales, entre otros lugares, que son beneficiados por los medios de transporte automotrices pero sufren algunas consecuencias como son los accidentes originados de muchos conductores que violentan los limites de velocidad. Por tanto toda persona que exceda los límites de velocidad incurre en delito y debe ser sancionado. Para ello se requiere de un mecanismo de vigilancia que permita la identificación de los usuarios que incurran en delitos por exceso de velocidad, controlando así los accidentes.

Para cubrir estas necesidades de control y vigilancia surgieron una serie de sistemas, entre los cuales se encuentran los sistemas de reconocimiento de matrícula que son también comúnmente llamados reconocimiento automático del número de placa por sus siglas en inglés (ANPR) o bien reconocimiento de placa vehicular por sus siglas en inglés (LPR). Estos sistemas están compuestos básicamente por una cámara; la cual es utilizada para capturar las imágenes de las placas, una serie de sensores que permitan detectar la velocidad de los vehículos, y un computador donde se encuentra la aplicación; que mediante algoritmos se hará el proceso de reconocimiento óptico de los caracteres de las placas, extrayéndolos de las imágenes capturadas por la cámara, este programa es también conocido como sistema de reconocimiento óptico de caracteres (OCR).

La presente investigación consta de cinco capítulos, donde en el primer capitulo se hace la descripción del proyecto; conteniendo justificación, planteamiento del problema y objetivos. El segundo contiene los fundamentos teóricos del trabajo. En el tercero se hace la selección de los algoritmos de cada etapa del sistema de reconocimiento de matriculas. El cuarto capitulo trata de la implementación de cada

uno de los algoritmos seleccionados para cada etapa del OCR . Y el quinto capitulo en el cual se expone los resultados obtenidos en el trabajo de grado. Finalmente se tienen las conclusiones y recomendaciones provenientes de la realización de este trabajo.

CAPÍTULO I : DESCRIPCIÓN DEL PROYECTO

1.1 Antecedentes y Justificación

Los sistemas de reconocimiento óptico de caracteres engloban un conjunto de métodos y algoritmos, los cuales permiten reconocer de forma automática los mismos. La necesidad de tener sistemas de reconocimiento automático o semiautomático de caracteres ha sido reconocida por décadas, lo que ha adquirido el desarrollo e integración de diversas áreas como la visión, representación y entendimiento del conocimiento, inteligencia artificial y teoría de control.

El modelo computacional a desarrollar para el Sistema de Reconocimiento de matrículas, permitirá tomar los datos de la placa de un vehículo, mediante algoritmos en distintas etapas que permitirá identificarlos, generando así investigaciones en el área de electrónica.

En cuanto a las investigaciones sobre reconocimiento de caracteres en imágenes, la Universidad Simón Bolívar cuenta con una publicación del año 2009 por Cadore Joyner quien realizó el Desarrollo de un analizador de imágenes basado en el reconocimiento óptico de caracteres en la Universidad Simón Bolívar [1].

El desarrollo del presente trabajo adquiere valor y justificación unas vez que se consideran los resultados que se pretenden alcanzar al culminar los objetivos que más adelante se plantearán.

1.2 Planteamiento del Problema

Venezuela es un país con un alto índice de accidentes viales por exceso de velocidad. Nuestra sociedad se ve afectada a diario por el número de decesos a raíz de accidentes en las redes viales, accidentes que afectan en mayor medida a nuestra

población joven.

La Fundación Centro Nacional de Desarrollo e Investigación en Telecomunicaciones (CENDIT), a través de un seminario realizado sobre los problemas y realidades nacionales, notó la preocupación acerca del exceso de velocidad de los vehículos que circulan las avenidas y carreteras rodeadas por comunidades (Escuelas, Alcabalas, Urbanizaciones). Entre las comunidades que presentan dicho problema es la Base Aérea Generalísimo Francisco de Miranda, La Carlota, lugar donde se encuentran varias instituciones militares y organismos del Estado (CENDIT, ABAE, CENIT,CNTQ), que son transitadas por una gran cantidad de automóviles que violentan la velocidad máxima permitida del área, que en este caso está fijada a 40 Km/h .

La Dirección de Electrónica de Comunicaciones del CENDIT, comprometida a dar una solución no sólo al problema que se plantea, sino a dar una solución amplia que permita resolver la problemática de exceso de velocidad en las vías donde ocurren grandes accidentes automovilísticos, ocasionados por conductores que exceden la velocidad permitida en el canal, encontró una posible solución. Dicha solución no es más que un sistema detector de velocidad, el cual en principio fue pensado para que cense la velocidad de los vehículos en circulación y capture el número de matrícula de aquellos quienes violenten el límite de velocidad fijado. El sistema de censado de velocidad está concebido bajo tecnología de sensores infrarrojos, mientras que para la captura de la matrícula se plantea un sistema OCR. Es por ello que se plantea en este trabajo especial de grado el desarrollo de un modelo computacional que permita el reconocimiento de matrículas para el sistema de detección de velocidad vehicular.

1.3 Objetivo General

Desarrollar un modelo computacional para un sistema de reconocimiento de matrículas mediante una imagen proveniente de una cámara de tráfico.

1.4 Objetivos Específicos

- Analizar las técnicas y algoritmos de las distintas etapas del preprocesamiento de imágenes y extracción de caracteres de una matrícula vehicular.
- Seleccionar las técnicas y algoritmos de las distintas etapas del preprocesamiento de imágenes de una matrícula vehicular.
- Seleccionar el algoritmo a utilizar en la extracción de caracteres de una matrícula vehicular.
- Diseñar una red neuronal empleando una librería de un paquete computacional que permita el reconocimiento de los tres(3) últimos caracteres de una matrícula vehicular.
- Implementar en una herramienta computacional de software libre, la propuesta del modelo que permita el reconocimiento de los tres(3) últimos caracteres de una matrícula vehicular.
- Verificar la propuesta realizada del modelo computacional de manera que el permita el reconocimiento de los tres(3) últimos caracteres de una matrícula vehicular.

1.5 Metodología

La metodología utilizada para cumplir con los objetivos planteados se basó en una investigación dinámica, donde se estudió la teoría necesaria para aplicarla a un problema de características y circunstancias específicas. La metodología de trabajo se dividió en cinco fases de interés:

Fase I

Revisión Bibliográfica: Consulta de bibliografía orientada a sistemas de reconocimiento de caracteres y tratamiento de imágenes. Se indagó en las diferentes

fases que componen el sistema de reconocimiento óptico de caracteres OCR y en los diferentes algoritmos que se ven involucrados en sus respectivas partes.

Fase II

Análisis y Selección de Algoritmos: dividida en dos partes:

1. Se realizó el análisis de los diferentes algoritmos correspondientes a las respectivas fases del sistema OCR, haciendo comparaciones entre los algoritmos existentes para cada una de dichas fases.

2. Basado en las comparaciones realizadas en la parte anterior se seleccionó el algoritmo más conveniente para cada fase del sistema OCR.

Fase III

*Programación de Algoritmos:*Luego de la selección, se procedió a programar en el entorno Scilab, el algoritmo de cada fase del OCR.

Fase IV

Pruebas: Se comprobó el funcionamiento de los algoritmos diseñados.

Fase V

Documentación: se realizó el informe final con los resultados obtenidos en las etapas anteriores.

CAPÍTULO II: FUNDAMENTOS TEÓRICOS DE UN SISTEMA DE RECONOCIMIENTO DE MATRÍCULAS

2.1 Generalidades

2.1.1 Imagen Digital

Una imagen es una representación visual de un objeto real a través de técnicas de fotografía, pintura, video entre otras disciplinas, por lo que entonces una imagen digital es la representación bidimensional de una imagen capturada por algún medio electrónico, la cual es representada mediante bits. Dependiendo de si la resolución de la imagen es estática o dinámica, puede hablarse de imagen rasterizada o imagen vectorizada [2], [3].

Una **imagen vectorial** es una imagen digital formada por objetos geométricos independientes (segmentos, polígonos, arcos, etc.), cada uno de ellos definido por distintos atributos matemáticos de forma, de posición, de color. Como muestra la Figura 1, la imagen vectorial puede ser ampliada en tamaño sin que sufra perdida de calidad al ser impresa [4].

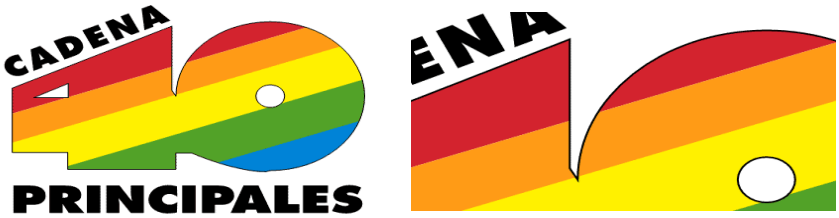


Figura 1: Imagen vectorial, obtenida de [5]

Una **imagen rasterizada,** también llamada mapa de bits, imagen matricial o bitmap, es una estructura que representa una rejilla rectangular en forma de matriz compuesta de pixeles o puntos de color, denominada raster. Las imágenes rasterizadas se las suele caracterizar por su altura y anchura (en pixels) y por su profundidad de color(en bits por pixel), que determina el número de colores distintos que se pueden almacenar en cada pixel, y por lo tanto, en gran medida, la calidad del color de la imagen. Como muestra la Figura 2, imagen rasterizada a diferencia de la vectorial al ser agrandada sufre de pérdidas de calidad [6].

Figura 2: Imagen rasterizada

A menos que se indique lo contrario se define una imagen digital como una imagen rasterizada.

2.1.2 Espacios de Color y Color en las Imágenes Digitales

Un espacio de color es aquel que define un modelo de composición del color. Por lo general un espacio de color lo define una base de N vectores cuya combinación lineal puede generar todo el espacio de color. Los espacios de color más generales intentan englobar la mayor cantidad posible de los colores visibles por el ojo humano, aunque existen espacios de color que intentan aislar tan solo un subconjunto de ellos.

Existen espacios de color de:

- Una dimensión :escala de grises, escala Jet, etc.
- Dos dimensiones: sub-espacio RG, sub-espacio XY, etc.
- Tres dimensiones: espacio RGB, HSV, YCbCr, YUV, YIQ, etc.
- Cuatro dimensiones: espacio CMYK..

De los cuales, los espacios de color de tres dimensiones son los más utilizados. Entonces, un color se especifica usando tres coordenadas, que representan su posición dentro de un espacio de color específico. Estas coordenadas no nos dicen cuál es el color, sino que muestran dónde se encuentra un color dentro de un espacio de color en particular.

El **espacio RGB** es conocido como un espacio de color aditivo (colores primarios) porque cuando la luz de dos diferentes frecuencias viajan junta, desde el punto de vista de un observador, estos colores son sumados para crear nuevos tipos de colores. Con la combinación apropiada de rojo, verde y azul se pueden reproducir muchos de los colores que pueden percibir los humanos. La Figura 3 muestra como se distribuyen los vectores que forman espacio RGB.

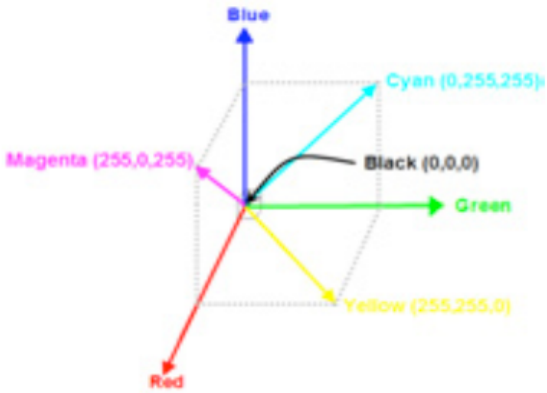


Figura 3: Espacio RGB [7..,Capítulo 8, p. 2]

El **espacio YIQ** fue una recodificación realizada para la televisión americana (NTSC), que tenía que ser compatible con la televisión en blanco y negro, que

solamente requiere del componente de iluminación. Los nombres de los componentes de este modelo son Y por luminancia (*luminance*), I fase (*in-phase*) y Q cuadratura (*quadrature*). La componente Y por si sola contiene la imagen en escala de grises de la imagen original proveniente del espacio RGB [8].

La codificación del espacio RGB puede ser transformada a la YIQ y viceversa. Las conversiones vienen dadas por las siguientes ecuaciones [9]:

Conversión de RGB a YIQ

$$Y = 0.299\mathrm{R} + 0.587\mathrm{G} + 0.114\mathrm{B} \quad (1)$$

$$I = 0.596\mathrm{R} - 0.274\mathrm{G} - 0.322\mathrm{B} \quad (2)$$

$$Q = 0.211\mathrm{R} - 0.523\mathrm{G} + 0.312\mathrm{B} \quad (3)$$

Las imágenes digitales rasterizadas dependiendo del espacio de color en las que esté definida su composición se clasifican básicamente en:

1. A color: que podría estar representada en función de cualquiera de los espacios tridimensionales de color, siendo el más comúnmente utilizado para representarlas el espacio de color RGB.
2. A escala de grises: no es más que la representación de imágenes utilizando la componente Y del espacio de color YIQ, por tanto lo que estaría en un espacio unidimensional de solo iluminación donde el máximo valor indica totalidad de iluminación (tono blanco) y el mínimo valor indica ausencia de iluminación (tono negro).

   También un pixel puede ser llevado a escala de grises proyectando sus componentes RGB sobre el vector [1,1,1] [9] como muestra la Figura 4, lo cual se lleva a la práctica mediante la siguiente ecuación:

$$PixelGris = \frac{R+G+B}{3} \quad (4)$$

3. Blanco y negro: estas imágenes son representadas con una simplificación del espacio unidimensional de escala de grises, donde se representan las imágenes a base de dos únicos tonos de color, que son el máximo y el mínimo de la escala de iluminación, es decir, totalmente blanco y totalmente negro, simplificación que se logra con técnicas que se mencionan más adelante.

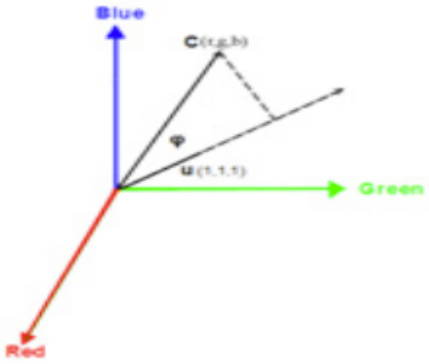


Figura 4: Espacio de colores RGB y Vector Escala de Grises [7, capítulo 8, p. 12]

Todos los espacios unidimensionales son llamados monocromas, un ejemplo para ilustrar esto sería el espacio RGB, que está compuesto por las monocromas R(rojo), G(verde) y B(azul) , donde cada monocroma tiene una escala de 256 niveles de color, que van desde el color blanco que vale 0 hasta el máximo de la monocroma que tomaría el valor de 255.

2.2 Procesamiento de Imágenes Digitales

Se denomina procesamiento de imágenes digitales al conjunto de técnicas que se aplican a dichas imágenes para mejorar la calidad o facilitar la búsqueda de información. Las técnicas que se describen en este trabajo están orientadas mayormente a imágenes en escala de grises o bien imágenes binarias, porque a éstas

se le puede extraer la información requerida con mayor simplicidad y eficacia.

2.2.1 Proceso de Filtrado

Es el conjunto de técnicas que se pueden encontrar dentro de lo que es el procesamiento de imágenes cuyo objetivo fundamental es obtener, resaltar o suprimir, de forma selectiva, información contenida en una imagen a diferentes escalas espaciales, para destacar algunos elementos de la imagen, o también para ocultar valores anómalos, mejorando ciertas características de la misma que posibilite efectuar posteriores operaciones del procesado sobre ella. El filtrado puede hacerse en el dominio del espacio o en el dominio de la frecuencia, para los efectos de este trabajo se describen los filtros en el dominio del espacio.

Filtrado espacial es la operación que se aplica a imágenes digitales rasterizadas para mejorar o suprimir detalles espaciales con el fin de mejorar la interpretación visual. El filtrado espacial es una operación que modifica el valor de cada pixel de la imagen de acuerdo con los valores de los píxeles que lo rodean.

Una característica común a todos los imágenes digitales rasterizadas es la llamada "frecuencia espacial", que define la magnitud de cambios de los datos por unidad de distancia en una determinada zona de la imagen. Áreas de la imagen con pequeños cambios o con transiciones graduales en los valores de los datos se denominan áreas de bajas frecuencias (por ejemplo la superficie de una masa de agua en reposo). Áreas de grandes cambios o rápidas transiciones se conocen como áreas de altas frecuencias (por ejemplo suelo urbano con densas redes de carreteras). Así, los filtros espaciales se pueden dividir en dos categorías; filtros lineales y filtros no lineales. La descripción detallada se puede apreciar en el Anexo A.

2.2.2 Transformaciones Geométricas

Las trasformaciones geométricas son aquellas que modifican las relaciones

espaciales entre los pixeles que conforman la imagen.

A continuación se mencionan algunas:

- Traslación: Transformación geométrica que mapea la posición de cada pixel de la imagen de entrada a una nueva posición en la imagen de salida [10].

  Esta transformación se define por la matricial siguiente:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & Tx \\ 0 & 1 & Ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  de lo cual se tiene:

$$x' = x + Tx \tag{5}$$

$$y' = y + Ty \tag{6}$$

- Rotación: Transformación geométrica la cual mapea la posición de un pixel de una imagen de entrada en una posición a una imagen de salida por rotación de la misma a través de un ángulo especificado por el usuario y un origen[10].

  Esta transformación se define por la ecuación matricial siguiente:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

de la cual se obtiene:

$$x' = \cos(\theta)x - \sin(\theta)y \tag{7}$$

$$y' = \sin(\theta)x + \cos(\theta)y \tag{8}$$

- Escalado: Transformación geométrica utilizada para reducir o ampliar el tamaño una imagen o parte de ella [10].

  La transformación se define por la ecuación matricial siguiente:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

de la cual se obtiene:

$$x' = x * Sx \tag{9}$$

$$y' = y * Sy \tag{10}$$

La ampliación de imágenes se conoce como zooming, y se hace recurriendo a métodos de interpolación, esto debido a que las imágenes digitales son señales discretas y la transformación definida anteriormente las considera continuas, ocasionando que queden espacios vacíos entre los pixeles transformados, como se observa en la Figura 5.

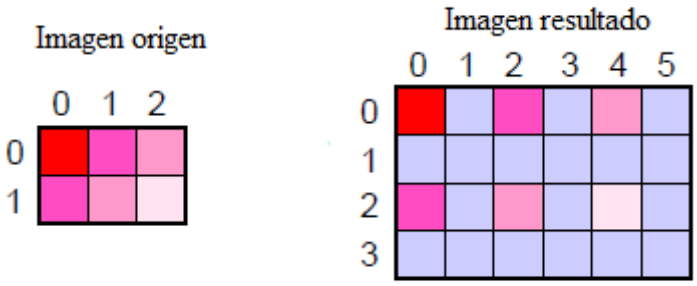


Figura 5: Efecto de aplicar la transformación de escalado ampliación a una imagen

Matemáticamente la interpolación de una imagen se expresa de la siguiente manera:

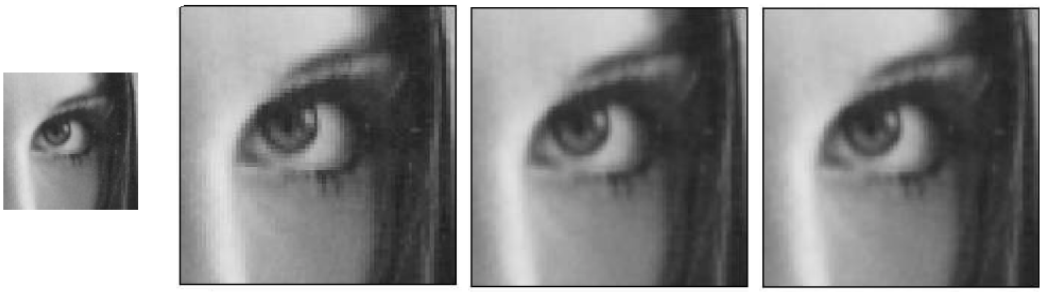
$$p(x',y') = \sum_{x=-n}^{n} \sum_{y=-m}^{m} p(x,y) * h(distancia((x,y),(x',y'))) \qquad (11)$$

Donde p(x',y') es el valor final del pixel, p(x,y) es el valor del pixel original y h(distancia((x,y),(x',y'))) es el núcleo de la interpolación.

Entre los métodos de interpolación más importantes se encuentran:

a) Interpolación por vecinos más cercanos
b) Interpolación bilineal
c) Interpolación bicúbica

En la Figura 6 se muestra el resultado de aplicarle los tres métodos antes descritos una imagen en escala de grises.



(a) *(b)* *(c)* *(d)*

Figura 6: Muestra de los resultados al aplicar los tres métodos de interpolación. (a) Imagen original, (b) Vecinos más cercanos, (c) Bilineal, (d) Bicúbica.

La reducción de la imagen es un proceso similar a la ampliación, se conoce comúnmente como submuestreo, y se lleva a cabo reemplazando un grupo de valores de píxeles por un pixel escogido de forma arbitraria de entre los que forman parte de ese grupo o por interpolación entre valores de píxeles vecinos.

2.2.3 Binarización de Imágenes y Procesado de Imágenes Binarias

La **binarización** de una imagen digital consiste en convertir dicha imagen en una imagen en blanco y negro, de tal manera que se preserven las propiedades esenciales de la imagen. Cabe destacar que la imagen original debe ser una imagen en escala de grises, ya que como fue mencionado en la Sección 2.1.1, el espacio de color blanco y negro es una simplificación del espacio en escala de grises.

La simplificación del espacio en escala de grises se hace definiendo o encontrando un umbral, el cual viene a indicar el valor límite de la escala de grises a partir del cual los valores superiores se consideran blanco (255) e inferiores negro (0). Para simplificar y además entrar en el concepto de lo binario, el valor del blanco en vez de ser 255 sera “1” y el negro “0”, y con esos valores se trabajan en general las imágenes binarias. Para hallar este umbral existen diversos algoritmos los cuales se

basan en histogramas o combinaciones del mismo.

Un **histograma** es la representación gráfica de una variable, donde en el eje horizontal de dicho gráfico se encuentran los valores que toma la variable, y en el eje vertical se encuentra la frecuencia con que aparece cada valor de la variable. En una imagen de escala de grises, en el eje horizontal del histograma aparecerían los niveles de luminosidad de dicha escala, variando de 0 a 255, y en el eje vertical estarían las frecuencias con que aparecen dichos niveles en la imagen, es decir, la cantidad de pixeles que tienen los diversos niveles de luminosidad.

De los histogramas pueden darse distintas situaciones las cuales se nombran a continuación:

- Distribución unimodal: Los objetos están poco contrastados respecto al fondo. El histograma integra toda la información y sólo aparece un pico dominante (usualmente el fondo).

- Distribución bimodal: Los objetos aparecen claramente contrastados respecto al fondo y todos ellos presentan la misma distribución de niveles de gris. Esta presenta dos picos o dos cimas

- Distribución multimodal: Varios objetos bien contrastados con distribuciones de gris diferentes. El histograma tiene forma con varios picos con valles de separación [11].

En Figura 7 se observan algunos ejemplos representativos de los tipos de histogramas. Para la nomenclatura de los ejes, se entiende “g” como el valor en escala de grises y “N°” como la frecuencia en que aparece cada valor en la gráfica.

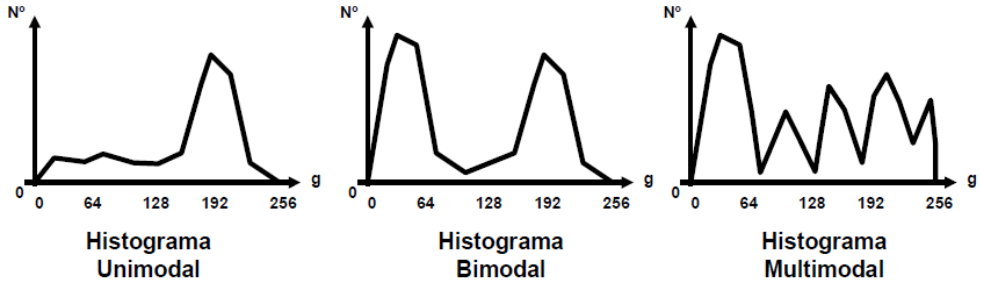


Figura 7: Formas de un histograma [11, p.20]

A continuación algunos métodos para hallar el umbral para binarización:

- Método P-Tile : Este método se basa en el conocimiento previo de la imagen, el tamaño de sus objetos y que estos asumen un porcentaje de la misma, porcentaje al cual se le llama p%. El algoritmo se basa precisamente en hallar el punto del histograma que suma la cantidad de pixeles equivalentes a p%, procedimiento que se ve en la Figura 8:

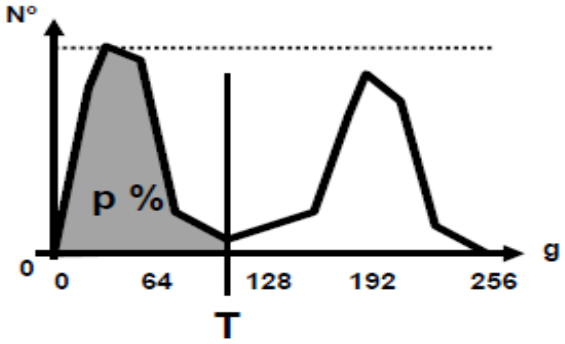


Figura 8: Umbral hallado por P-Tile [11, p.21]

Este método es muy limitado por tanto muy poco usado, funciona mayormente en imágenes con fondo blanco predominante como por ejemplo un archivo de texto impreso digitalizado, por tanto como muestra la Figura 8 el histograma tiene que ser bimodal [11].

Tomando un vector, el cual se denota por h(i), que contiene los valores de frecuencia de los niveles i del histograma, entonces el algoritmo es como sigue:

1. Definir p%
2. Sumar al acumulador acum el valor de h(i)
3. si acum no es igual a p% aumentar contador i y volver a 2.

- Método de Búsqueda Gaussiana: Esta técnica, al igual que la anterior, parte de la hipótesis de que las imágenes tienen unos histogramas bimodales. Pero además, pueden aparecer multitud de pequeños picos denominados máximos locales. Para aplicar el algoritmo es necesario en primer lugar realizar un suavizado del histograma de forma que, finalmente, los únicos máximos presentes en él sean los correspondientes a los lóbulos. Alguna de las técnicas de suavizado podrían ser suavizado por promediado y promediado gaussiano. [11]. El algoritmo tiene tres pasos:

1. Suavizado del histograma hasta conseguir eliminar los máximos locales y dejar tan sólo los dos máximos globales correspondientes a ambos lóbulos.
2. Partiendo del máximo situado más a la izquierda, se recorre el histograma calculando la pendiente en una ventana de tamaño dado, hasta que esta pendiente supere un cierto umbral p% de pendiente positiva.
3. Cuando la pendiente supere el umbral p% se determina que el umbral T para la umbralización es la posición media de la ventana, en dicho instante.

- Método Isodata: Se trata de un algoritmo iterativo desarrollado por Ridler & Calvard. Este método consiste en conseguir el umbral de forma iterativa; dividiendo la imagen en dos regiones a las que se le halla un umbral individual, y luego se promedian estos umbrales parciales[12]. Este método es iterativo y tiene cinco pasos:

1. Seleccionar una umbralización inicial Ti. Normalmente es la media entre los

pixeles de la imagen entrada.

2. Dividir la imagen en dos, R1 y R2 utilizando Ti.
3. Calcular la media (m1) de R1 y la media (m2) de R2.
4. Tomar la siguiente umbralización: $T_{i+1} = \frac{(m1 + m2)}{2}$.
5. Repetir los pasos 2-4 hasta que las medias m1 y m2 en sucesivas iteraciones no cambien.

- Método de Simetría del Fondo: Esta técnica asume que el fondo forma un pico simétrico y dominante en el histograma tal como muestra la Figura 9. Esta técnica requiere de suavizado previo del histograma y requiere conocimiento previo de la imagen. El método consiste en 4 pasos:

1. Inicialmente se suaviza el histograma.
2. Se obtiene el máximo global del histograma Tmax (Fondo).
3. Se busca a la derecha (en el lado opuesto al objeto) el punto que corresponde al p% del histograma (por ejemplo el 95 %).
4. Puesto que se supone que el lóbulo del fondo es simétrico, se toma como umbral el máximo menos un desplazamiento igual al del punto del p%.
   T = Tmax. - (p% - Tmax).

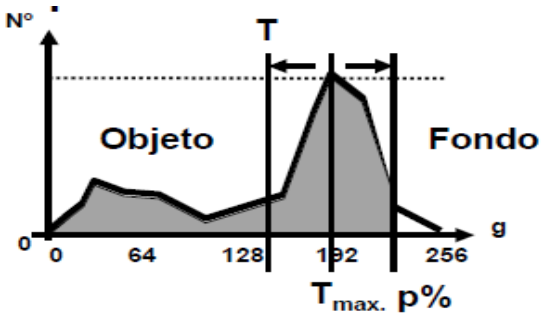


Figura 9: Umbral hallado por simetría de fondo[11, p.29]

- Método del Triángulo: Este método también asume que el fondo representa un pico simétrico. Halla el umbral trazando una recta entre el punto más alto del histograma que seria el pico del fondo y el punto más bajo. Luego el punto de umbral sera aquel punto para el cual la distancia a la recta hallada es máxima. Esta técnica se muestra en la Figura 10:

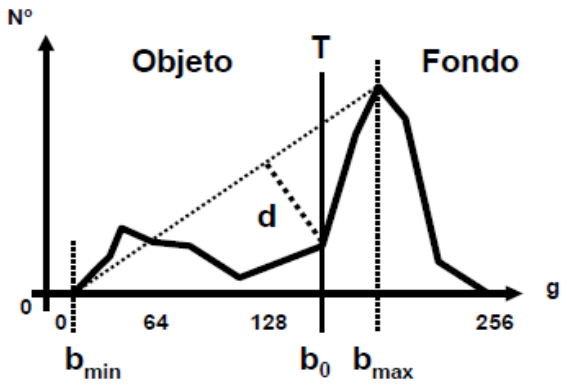


Figura 10: Umbral con el método del triángulo[11, p.32]

Este tiene tres pasos:

1. Se traza una línea entre el valor máximo del histograma (al nivel de gris bmax), y el mínimo gris en la imagen bmin = (p=0%).
2. Se calcula la distancia desde dicha línea al histograma h[b] para todos los valores de b = bmin… bmax.
3. Se escoge el valor de gris b0 para el que la distancia entre h[b0] y la línea es máxima, siendo el umbral T = b0.

Una vez obtenida la imagen binaria existen una serie de técnicas para procesarlas y así poder obtener la información requerida. La mayoría de estos procesamientos fueron tratados en la sección 2.2.2 entre los cuales están los filtros morfológicos erosión, dilatación, apertura, y cierre. Pero también existe otra diversidad de procesamientos para imágenes binarias, diversas técnicas entre las

cuales se pueden nombrar la esqueletización,el rellenado de agujeros,y el etiquetado de zonas, además de que se siendo binarias se le pueden hacer a estas imágenes las operaciones matemáticas binarias como son and, or, or exclusivo, etc.

La **esqueletización** de una imagen binaria busca representar una imagen binarizada por un grafo fino, que usualmente tiene un solo pixel de grosor tal como muestra en la Figura 11, cuyos puntos cumplen que la distancia local respecto a los bordes de la imagen binarizada es máxima. Esta operación también puede verse como una erosión que concluye cuando se alcanza la linea final de pixeles del objeto [13], [13].

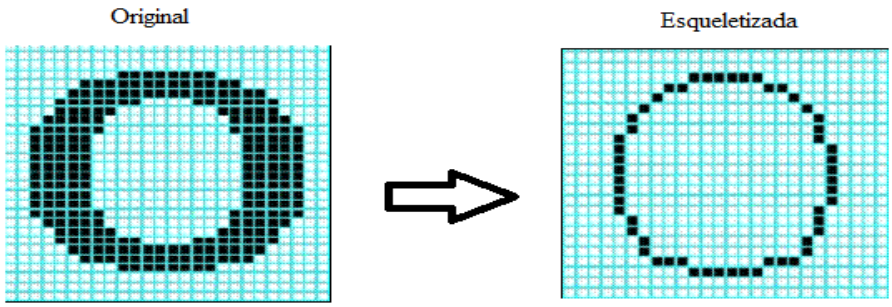


Figura 11: Efecto de la esqueletización en una imagen binaria

Los algoritmos de esqueletización se basan en la ejecución de un conjunto de iteraciones, donde en cada una se realiza el borrado de los pixels pertenecientes a los bordes de la imagen, hasta que solamente queda el esqueleto. El borrado o no cada pixel requiere de una análisis local de los pixels vecinos, para determinar si pertenece al borde de la imagen y si su borrado permite conservar conectividad con el resto del esqueleto. Para los fines de este trabajo se describirá uno de los métodos más utilizado para la esqueletización que es el método de Zhang Zuen.[13]-[14]

El método de Zhang Zuen es un método paralelo, es decir, cada pixel se puede calcular utilizando el valor de la iteración anterior. El método consta de dos subiteraciones. En cada una de ellas se eliminarán aquellos píxeles que cumplan todas las reglas definidas para la subiteración. El pixel que se analiza se hace basándose en

sus ocho vecinos, siendo el pixel a analizar el pixel central como se observa en la matriz siguiente:

$$\begin{bmatrix} I(i-1,j-1) & I(i-1,j) & I(i-1,j+1) \\ I(i,j-1) & I(i,j) & I(i,j+1) \\ I(i+1,j-1) & I(i+1,j) & I(i+1,j+1) \end{bmatrix}$$

La primera subiteración consiste en un conjunto de reglas que se comprueban para cada pixel de la imagen, y son las siguientes:

1. La conectividad debe ser uno, es decir, la cantidad de cambios de blanco a negro en los vecinos deben ser iguales a uno.
2. Debe tener al menos dos vecinos negros y no más de seis.
3. Al menos uno de I(i,j+1), I(i-1,j) e I(i,j-1) es blanco.
4. Al menos uno de I(i-1,j), I(i+1,j) e I(i,j-1) es blanco[13].

La segunda subiteración es igual que la primera salvo por las dos últimas reglas, que son las siguientes:

1. La conectividad debe ser uno, es decir, la cantidad de cambios de blanco a negro en los vecinos deben ser iguales a uno.
2. Debe tener al menos dos vecinos negros y no más de seis.
3. Al menos uno de I(i-1,j), I(i,j+1) e I(i+1,j) es blanco.
4. Al menos uno de I(i,j+1), I(i+1,j) e I(i,j-1) es blanco[13].

2.3 Sistema de Reconocimiento Óptico de Caracteres (OCR)

Un sistema de reconocimiento óptico de caracteres (OCR) es un software que traduce texto impreso digitalmente en una imagen, a un formato que se pueda utilizar en programas de edición de texto.

Este sistema comprende un conjunto de técnicas basadas en estadísticas,

análisis de formas, transformaciones y comparaciones, que tienen el fin identificar de forma automatizada de símbolos o caracteres pertenecientes a un determinado alfabeto a partir de una imagen capturada. En la actualidad no solo se reconocen exactamente los caracteres de un determinado alfabeto, si no también es posible distinguir entre cualquier conjunto de formas o símbolos que se pueden definir [15].

En un sistema OCR se distinguen al menos cuatro etapas básicas en el proceso, las cuales dependiendo de su complejidad podrían subdividirse en otros conjuntos de etapas. Estas cuatro etapas son: Pre-procesamiento, segmentación, extracción de características y reconocimiento [15].

(a)Etapa de Preprocesamiento. En esta fase de preprocesamiento lo que se busca es la adecuación de la imagen. El objetivo que se persigue es eliminar de la imagen de cualquier tipo de ruido, imperfección o información que no pertenezca al caracter o al conjunto de caracteres. Para lograr este objetivo es necesario aplicar un conjunto de técnicas para tratamiento de imágenes, de las que fueron tratadas en la Sección 2.2.2 en este capítulo, como lo son filtrado, ampliación, binarización, entre otros, por tanto esta etapa puede subdividirse en otra serie de subetapas, dependiendo de la aplicación.

(b)Etapa de Segmentación. Una vez preprocesada la imagen se deberá fragmentar o segmentar en las diferentes componentes que forman el texto a extraer. Entonces el objetivo de esta etapa consiste en dividir o fragmentar la imagen en una serie de componentes de la misma, que son conformados cada uno por un caracter.

Para alcanzar este objetivo es necesario aplicar alguna técnica de segmentación, entre las cuales están: perfilado manual de contornos, detección de bordes, selección de umbrales de gris y crecimiento de regiones, que son técnicas no utilizables para la aplicación de esta etapa ya que contemplan imágenes en escala de grises y en esta etapa se tienen imágenes binarias[14.].

Para las imágenes binarias características de un OCR lo mejor es un

algoritmo basado en proyecciones horizontales, ya que en éste una vez preprocesada la imagen se tiene cierto conocimiento acerca de la distribución y ubicación de los caracteres en la misma [15].

A la hora de segmentar un conjunto de caracteres lo que se hace es detectar los distintos renglones que forman el texto. Para conseguirlo se realiza el siguiente procedimiento:

1. Se hace una proyección horizontal (histograma) consistente en contar los elementos de tinta que existen en cada una de las filas o columnas, traspasando estos valores a otra matriz, unidimensional, resultado de la proyección, en la que existirán diferentes zonas de densidad de tinta separadas por otras vacías o de muy poca densidad. Cada zona donde la proyección dé un valor no nulo será interpretado como un hipotético renglón.

2. Se analiza la matriz unidimensional para detectar los posibles renglones de los que está compuesto el texto. Si se detecta una línea con densidad de proyección no nula y además la anterior estaba en blanco o muy cercano a este, en esa línea comienza un renglón. A continuación se realiza la misma operación pero a la inversa, se busca la línea posterior que sea blanca o muy cercano a esto y que la anterior no lo fuera, ahí estará el final del renglón. Este método se aplica sucesivamente hasta el final de la matriz de proyección, consiguiendo así delimitar los renglones que forman el texto [15].

(c)Etapa de Caracterización . Una vez realizada la segmentación, se tienen imágenes normalizadas en la que se encuentra la información susceptible de ser "reconocida". La información así representada, es una matriz bidimensional de valores binarios, que no codifica de forma óptima las características más discriminativas del objeto al que representa [15].

Desde el punto de vista del reconocimiento de formas, la matriz

bidimensional se ve como un vector de tantas dimensiones como componentes tiene la matriz. La dimensión de estos vectores es normalmente elevado, lo que supone un gran coste computacional a la hora de procesar el mismo. Y no solo eso si no que está comprobado que al intentar clasificar o reconocer vectores de este tamaño aparece un efecto, llamado maldición de la dimensionalidad, que provoca que los resultados, independientemente del método de clasificación utilizado, no sean satisfactorios [15].

Por ello se han desarrollado multitud de técnicas, denominadas "técnicas de selección y extracción de características", mediante las cuales es posible obtener una representación del objeto a reconocer más eficiente. Eficiencia, en este caso, significa que con una representación más compacta se consigue un poder discriminativo igual o superior al que se tenía con la representación original. Esto no es solo importante por el ahorro de espacio en el almacenamiento de las muestras, sino que durante el proceso de reconocimiento reduce los costos computacionales, debido a la reducción en el volumen de información procesado [15].

En el campo de investigación del reconocimiento de formas se tiene experiencia en el uso de algunos métodos de extracción de características basados en transformaciones del espacio de representación de las muestras. Ejemplos de estos métodos son: PCA (Principal Component Analysis o Análisis por Componentes Principales), LDA (Linear Discriminant Analysis o Análisis por Componente Lineal), ICA (Independent Component Analysis o Análisis por Componentes Independientes), NDA (Non-linear Discriminant Analysis o Análisis por Discriminante No Lineal). Pero estos métodos son utilizados en imágenes de escala de grises, porque en imágenes binarias no tendrían mayor efecto.

Para imágenes binarias de caracteres se tienen dos métodos que podrían ajustarse a la aplicación de esta etapa:

- Método Basado en Diezmado: El diezmado consiste en dividir la imagen de entrada en una serie de rectángulos de tamaño $m_0 x n_0$. Cada uno de estos

rectángulos se va a convertir en un único pixel de la imagen resultado. El pixel resultado es, habitualmente, calculado como la media del rectángulo de entrada. También podrían usarse otras operaciones para el cálculo del pixel resultado, lo importante es que el método nos da como resultado una imagen reducida. Lo negativo de este método es que al reducir de tamaño la imagen de la forma planteada le resta resolución y por tanto detalles, por lo que entonces introduce ruido en el proceso.

- Método de Características Binarias: Este método consiste en extraer características conocidas propias de la imagen binaria y utilizarlas para formar un vector binario de salida que ahora sera la representación de dicha imagen [15].

  Con este método el vector binario de salida se forma con las siguientes características tales como Transiciones verticales y horizontales, agujeros en el caracter, puntos finales y puntos de árbol, entre otros que podrían extraerse de las formas de un caracter.

(d)Etapa de Reconocimiento. En esta última etapa es donde el reconocimiento (o clasificación) de los objetos, en nuestro caso imágenes de caracteres, se realiza. El problema que en esta etapa se plantea consiste en desarrollar algún método que sea capaz distinguir la clase a la que pertenece un objeto entre un conjunto limitado de clases posibles. Entre los métodos que existen se tienen el knn (vecinos más cercanos), árbol de decisiones, clasificador de Bayes y redes neuronales. Para los fines de este trabajo solo las redes neuronales serán consideradas.

2.4 Redes Neuronales: El Perceptrón Multicapa

Las **Redes Neuronales Artificiales** (denominadas habitualmente como RNA o en inglés como ANN Artificial Neural Networks) son sistemas de computación que permiten la resolución de problemas que no pueden ser descritos fácilmente mediante un enfoque algorítmico tradicional, como, por ejemplo, en el reconocimiento de formas. Con las redes se expresa la solución de un problema, no como una secuencia de pasos, sino como la evolución de un sistema inspirado en el funcionamiento del cerebro humano y dotado, por tanto, de cierta "inteligencia".

Una red neuronal se compone por elementos llamados neuronas artificiales. Cada neurona recibe una serie de entradas Xi, que equivalen a las dendritas de una neurona biológica, además, reciben unas entradas Ui, llamadas umbrales, a través de interconexiones que contienen un peso Wi, que representarían la sinapsis en neuronas biológicas y emite una salida A, que es la suma de las entradas ponderadas por los pesos; y, por último esta salida de ponderación llega a la función de activación que transforma este valor en otro en el dominio que trabajen las salidas de las neuronas (Yi). En la Figura 12 se observa mejor lo antes descrito [16].

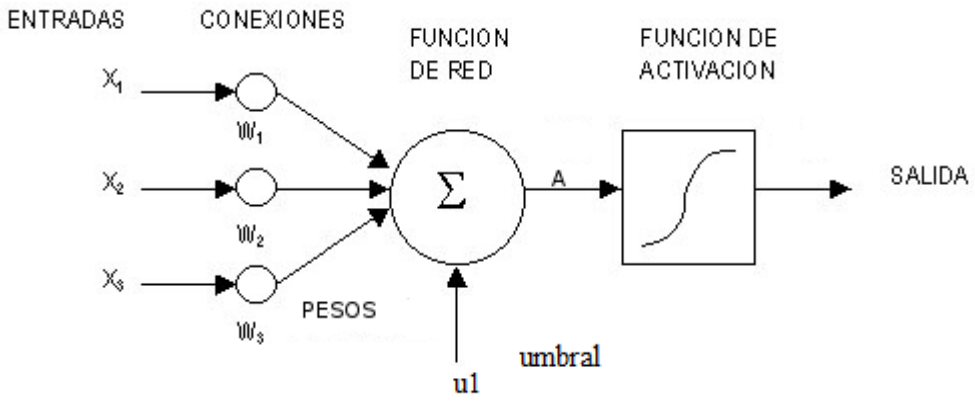


Figura 12: Neurona artificial

El resultado de la suma ponderada que ocurre en el cuerpo de la neurona y

que representa la salida del mismo, se denomina A y se representa de la siguiente manera:

$$A = X1 * W1 + X2 * W2 + ... + Xn * Wn + u \quad (12)$$

Siguiente a la señal E viene la última parte que conforma la neurona que es la función de activación. Esta función de activación procesa dicha señal E, para finalmente producir la señal de salida. De esta función de activación existen variedades, entre las que están: la función escalón, la función signo, la función rampa entre otras que pueden observarse en la Figura 13 [16].

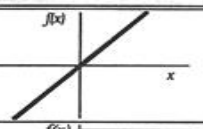
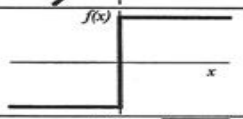
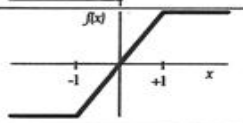
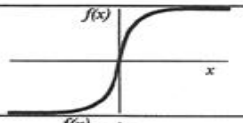
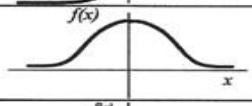
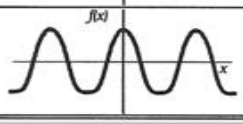
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

Figura 13: Funciones de activación usadas en redes neuronales[17]

2.4.1 Estructura Básica de las redes neuronales

En el modelo de una red neuronal, como se muestra en Figura 14, a la izquierda llegan una serie de entradas a un grupo de neuronas, cada una de estas neuronas realiza el cálculo que se mencionó anteriormente y propaga el resultado a

las siguientes neuronas vía conexiones de salida, hasta llegar al último grupo neuronas a la derecha.

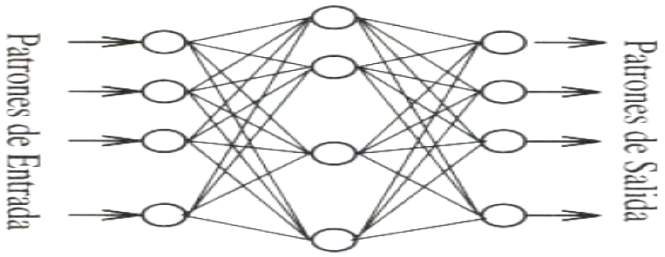


Figura 14: Modelo de red neuronal[16, p. 7]

A cada grupo de neuronas se le denomina capa y a la forma en que se conectan un grupo de neuronas con otro grupo de ellas se le denomina patrón de conectividad o arquitectura. La distribución de neuronas dentro de la red se realiza formando niveles o capas de un número determinado de neuronas. A partir de su situación dentro de la red se pueden distinguir tres tipos de capas:

***De entrada**: estas capas reciben la información desde el exterior, es decir actúa como un órgano sensor.

***De Salida:** estas envían la información hacia el exterior, es decir actúa como un órgano efector.

***Ocultas**: son capas que solo sirven para procesar información y comunicar otras Capas. Una capa oculta no tiene una conexión directa con el entorno, es decir, no se conecta directamente ni a órganos sensores ni a efectores. Este tipo de capa oculta proporciona grados de libertad a la red neuronal gracias a los cuales es capaz de representar más fehacientemente determinadas características del entorno que trata de modelar.

Teniendo en cuenta su arquitectura existen dos tipos de redes neuronales las monocapa y las multicapa:

Redes monocapa: Son redes con una sola capa. Para unirse las neuronas crean conexiones laterales para conectar con otras neuronas de su capa.

Redes multicapa: Las redes multicapa están formadas por varias capas de neuronas (dos o más). Estas redes se pueden a su vez clasificar atendiendo a la manera en que se conectan sus capas de la siguiente manera:

- *Feedforward o hacia adelante:* Usualmente, las capas están ordenadas por el orden en que reciben la señal desde la entrada hasta la salida y están unidas en ese orden. Este tipo de redes contienen solo conexiones entre capas hacia delante. Esto implica que una capa no puede tener conexiones a una que reciba la señal antes que ella en la dinámica de la computación [16].
- *Feedback o hacia atrás (realimentadas)*: Por el contrario existen ciertas redes en las que sus capas, aparte del orden normal, también esta unidas desde la salida hasta la entrada en el orden inverso en que viajan las señales de información. Las conexiones de este tipo se llaman conexiones hacia atrás, feedback o retroalimentadas [16].

Este tipo de redes se diferencia a las anteriores en que pueden existir conexiones de capas hacia atrás y por tanto la información puede regresar a capas anteriores en la dinámica de la red. Este tipo de redes suelen ser bicapas.

2.4.2 Aprendizaje de una Red Neuronal

El aprendizaje en una red neuronal artificial consiste en la determinación precisa de los pesos para todas sus conexiones, de tal forma que la capacite para la resolución eficiente de un problema. El proceso general de aprendizaje consiste en ir introduciendo paulatinamente todos los ejemplos de un conjunto de aprendizaje, y modificar los pesos siguiendo un determinado esquema de aprendizaje. Una vez introducidos todos los patrones de ejemplo se verifica si se ha cumplido cierto criterio de convergencia, de no ser así, se vuelve a repetir el proceso y todos los ejemplos

vuelven a ser introducidos. La finalización del ciclo se puede determinar a partir de tres condiciones: una es mediante la definición de la cantidad de ciclos, la otra es cuando el error descienda por debajo de un valor establecido y la última es cuando el cambio de los pesos sea irrelevante [16]. En las redes neuronales se pueden distinguir tres tipos de aprendizaje:

a)Aprendizaje supervisado: Es el modo más intuitivo , que consiste en que la red dispone de los patrones de entrada llamados también ejemplos y los patrones de salida que deseamos para esa entrada que también llaman etiqueta, es decir se conoce el patrón, y la salida que debería dar ese patrón. La manera para modificar los pesos se muestra en la Figura 15, esto es cada vez que un ejemplo es introducido a la red esta, lo procesa y se obtiene una salida, esta salida es comparada con la salida que tenia que haber producido, y la diferencia entre ambas influirá en como se modifican los pesos. Para este tipo de aprendizaje se dice que hay un profesor externo, que esta encargando de ver si la red se comporta de forma adecuada mediante la comparación entre la salida producida y la deseada.

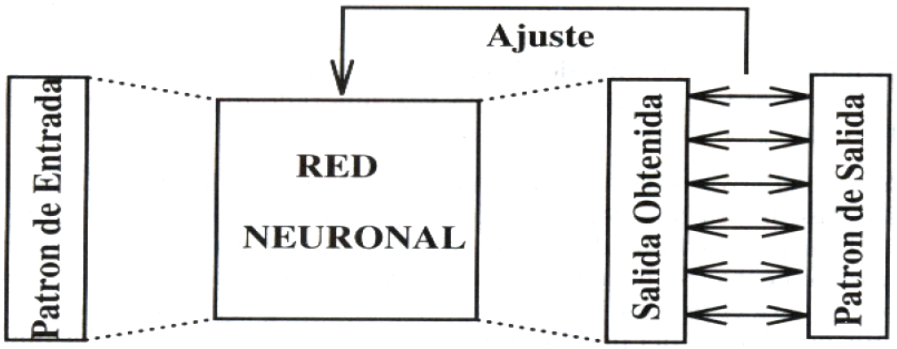


Figura 15: Esquema de aprendizaje supervisado de las redes neuronales artificiales [16, p. 12]

b)Aprendizaje No Supervisado: En este tipo de aprendizaje la modificación de los pesos solo se hace a partir de los patrones de entrada, por tanto en este caso no existe profesor externo que determine el aprendizaje, y dejar a la red clasificar dichos

patrones en función de las características comunes de los patrones, es decir la red modifica los pesos a partir de información interna. A este modelo se le conoce como sistema autoorganizado, debido a que la red se ajusta dependiendo únicamente de los valores de entrada. El esquema con que se modifican los pesos se observa en la Figura 16:

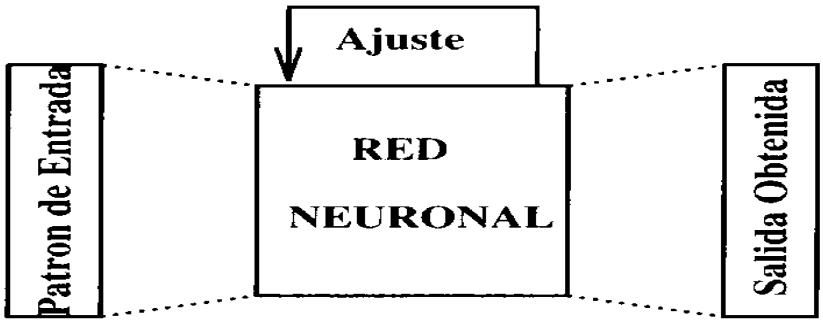


Figura 16: Figura 42: Esquema de aprendizaje no supervisado de las redes neuronales artificiales[16, p. 13]

c) Aprendizaje Reforzado: Este tipo de aprendizaje es una variante del aprendizaje no supervisado, solo que para este no se dispone del error cometido por la red con cada ejemplo de aprendizaje, sino que solamente se determina si la salida para dicho patrón es o no adecuada [16].

Existen diversos tipos de redes neuronales cuyas características son representadas según las descritas anteriormente. Para aprendizaje supervisado existen: las redes monocapa Adeline y Perceptrón simple, y las redes multicapa Madeline y Perceptrón multicapa, siendo todas ellas de conexión feedforward. Para el aprendizaje no supervisado existen diversas redes como la art1, la art2 , la competitiva simple entre otras.

Para la tarea de reconocimiento de características se requieren redes que tengan la capacidad de clasificar conjuntos que no son linealmente separables, es decir, que se requieren de redes multicapa, adicionalmente se requiere de una red cuyo aprendizaje sea supervisado ya que el problema parte de que se conocen los patrones para dicho aprendizaje y además se sabe cual será la salida para ese patrón.

Entonces las redes más adecuadas son las de aprendizaje supervisado multicapa de las que se tienen el Perceptrón Multicapa y la red Madeline.

La red madaline por su característica es utilizada en el diseño y realización de filtros, para llevar a cabo la eliminación del ruido en señales portadoras de información, módem, etc, en cambio la red Perceptrón multicapa es utilizada para resolver problemas de asociación de patrones, segmentación de imágenes, compresión de datos, y es muy común encontrarlas en aplicaciones de reconocimiento.

A continuación se describe el perceptrón multicapa ya que en la actualidad es una de las arquitecturas más usadas en el área de reconocimiento de voz, en el reconocimiento óptico de caracteres, reconocimiento de caracteres manuscritos ,y además, por ser considerado un aproximador universal en reconocimiento, así como por su uso y fácil aplicabilidad.

2.4.3 El Perceptrón Multicapa

El Perceptrón Multicapa es una red neuronal que surge por la limitacion que posee el perceptrón simple de la no separabilidad lineal, es decir, perceptrón simple presenta incapacidad para clasificar conjuntos que no son linealmente independientes. Esto quedo asentado en la obra Perceptrons que en 1969 demostró que un perceptrón es incapaz de aprender una función tan fácil como la XOR.

El perceptrón multicapa es una ampliación del perceptrón simple al cual se añade una serie de capas que, básicamente, hacen una transformación sobre las variables de entrada, que permiten eludir el problema anteriormente nombrado.

La arquitectura del perceptrón multicapa consta de una capa de entrada y una capa de salida y una o más capas ocultas. Dichas capas se unen de forma total hacia delante (feedforward), esto es, la capa entrada se une con la primera capa oculta y esta con la siguiente y la última capa oculta se une con la capa de salida como se observa en la Figura 17. Los valores que el perceptrón multicapa acepta son reales

[16].

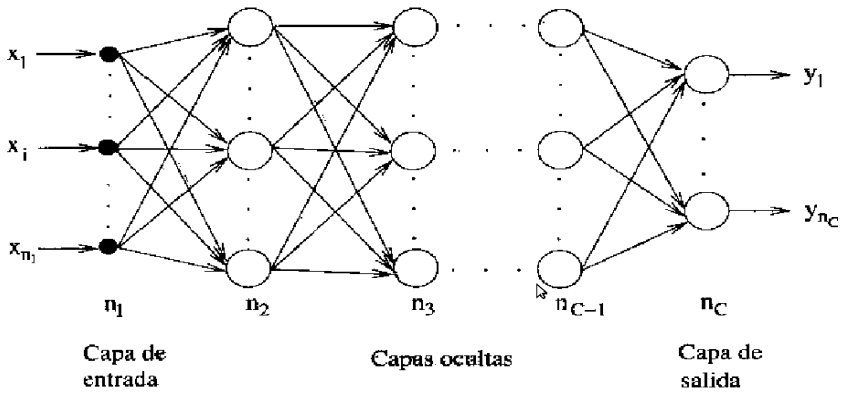


Figura 17: Modelo perceptrón multicapa [16, p.47]

El perceptrón multicapa define una relación entre las variables de entrada y las variables de salida de la red. Esta relación se obtiene propagando hacia delante los valores de la variable de entrada. Para ello cada neurona procesa la información recibida por sus entradas y produce una respuesta o activación que se propaga, a través de las conexiones correspondientes, hasta las neuronas de la siguiente capa [16].

Sea un perceptrón multicapa con *C* neuronas de entrada, *C-2* capas ocultas y n_c neuronas en la capa *c*, para *c=1,2,...,C.* Sea w^c_{ij} los pesos asociados a las conexiones de una capa c a la capa c+1, donde el subíndice i representa la neurona de la capa *c* y *j* representa la neurona de la capa c+1 para *c=1,2,...,C-1*, y sea u_i los umbrales de la capa c, para *c=2,...,C*. Se denomina a^c_i a la activación de la neurona *i* de la capa *c*, las expresiones para calcular las activaciones de las neuronas son las siguientes [16]:

La activación de las neuronas de la capa de entrada simplemente lo que hacen es transmitir la señal proveniente del exterior a las capas internas, siendo el valor de sus pesos igual a uno y su expresión es la siguiente:

$$a_i^1 = x_i^1 \; para \; i = 1,2,\ldots, n_1 \tag{13}$$

donde $X=(x1,x2,..,x_{n1})$ representa el valor del vector de entrada a la red.

La activación de las neuronas de la capa oculta c $(a^c{}_i)$ procesan la información recibida aplicando la función de activación f a la suma de los productos de las activaciones que recibe por sus correspondientes pesos, la cual se calcula de la siguiente forma:

$$a_i^c = f\left(\sum_{j=1}^{n_{c-1}} w_{ij}^{c-1} a_j^{c-1} + u_i^c\right) \quad para \quad i = 1,\ldots, n_c \quad y \quad c = 2,\ldots, C-1 \tag{14}$$

La activación de las neuronas de la capa de salida $(a^C{}_i)$, al igual que en el caso anterior viene dada por la función de activación f aplicada a la suma de los productos de las entradas que recibe por sus correspondientes pesos:

$$y_i = a_i^C = f\left(\sum_{j=1}^{n_{C-1}} w_{ij}^{C-1} a_j^{C-1} + u_i^C\right) \quad para \quad i = 1,\ldots, n_C \tag{15}$$

donde $y=(y1,y2,..,y_{n1})$ representa el valor del vector de salida de la red.

Para el perceptrón multicapa la función de activación f más utilizada es la sigmoidea, que se puede observar en la Figura 13 donde dependiendo del problema se puede utilizar la sigmoidal o bien la tangente hiperbólica [16].

Aprendizaje del Perceptrón Multicapa: Como se había mencionado anteriormente el aprendizaje en el perceptrón multicapa es del tipo supervisado. Entonces puesto que el objetivo en este tipo de aprendizaje es que la salida de la red sea lo más próxima posible a la salida deseada, entonces el aprendizaje se formula como la minimización mediante el conjunto de parámetros (pesos y umbrales), de una función E denominada función de error la cual se denomina como[16]:

$$E = \frac{1}{N} \sum_{n=1}^{N} e(n) \tag{16}$$

Donde N es el número de patrones o muestras y $E(n)$ es el error cometido por la red para el patrón *n y viene dado por:*

$$e(n)=\frac{1}{2}\sum_{i=1}^{n_c}(s_i(n)-y_i(n))^2 \qquad (17$$

siendo $Y(n)=(y_1(n),...,y_{nc}(n))$ el vector de salidas de la red y $S(n)=(s_1(n),...,s_{nc}(n))$ el vector de salidas deseadas para el patrón *n*.

Por tanto el aprendizaje en el perceptrón multicapa es equivalente a encontrar un mínimo en la función de error. Dicha minimización representa un problema no lineal debido a la presencia de funciones de activación no lineales, que hacen a dicho problema no lineal con respecto a los parámetros ajustables de la red, como consecuencia se tiene que utilizar técnicas de minimización no lineales. Dichas técnicas, generalmente, están en basadas en la búsqueda de una adaptación de parámetros siguiendo una cierta dirección de búsqueda. En particular para el perceptrón multicapa la búsqueda más comúnmente usada es la dirección negativa del gradiente, conocida como método de descenso del gradiente, pues conforme al cálculo de varias variables, ésta es la dirección en la que la función decrece [16].

Estrictamente hablando la minimización del error según la ecuación (17) debe realizarse para el error total, sin embargo el procedimiento más utilizado está basado en métodos del gradiente estocástico, los cuales consisten en una minimización sucesiva de los errores para cada patrón *e(n)* en lugar de minimizar el error total *E*. Por tanto el método del descenso del gradiente estocástico para la modificación de cada parámetro *w* de la red, para cada entrada *n* se realiza mediante la siguiente ley de aprendizaje:

$$w(n)=w(n-1)-\alpha\frac{\partial e(n)}{\partial w} \qquad (18)$$

donde α es la razón o tasa de aprendizaje, parámetro que influye en la magnitud del desplazamiento en la superficie del error, como se analizará más adelante [16].

Debido a que las neuronas de la red están agrupadas en capas de distintos niveles, es posible aplicar el método del gradiente de forma eficiente mediante el algoritmo de *retropropagación o regla delta generalizada.* El término

retropropagación se utiliza debido a la forma en que se implementa dicho algoritmo de retropropagación, pues el error cometido en la salida es propagado hacia atrás, transformándolo en un error para cada una de las capas ocultas de la red.

En el desarrollo de la regla delta generalizada es necesario distinguir dos casos para la propagación de errores uno para los pesos de las capa oculta *C-1* hacia la salida y para los umbrales de la capa de salida y otro para el resto de los pesos y umbrales [16].

Para el caso de la capa oculta *C-1* y umbrales de capa de salida, a partir del desarrollo de la derivada de la ecuación (18) se define el término δ que asociado a la neurona *i* de la capa de salida (capa *C*), y al patrón *n es* denotado como $\delta^c{}_i$, de la siguiente manera:

$$\delta(n)_i^C = -(s_i(n) - y_i(n)) f'(\sum_{j=1}^{n_{C-1}} w_{ij}^{C-1} a_j^{C-1} + u_i^C) \tag{19}$$

A partir de la cual se obtiene la formula para modificar los pesos antes mencionados:

$$\begin{gathered} w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \alpha \delta_i^C(n) a_i^{C-1}(n) \\ para\ \ j = 1,\ldots, n_{C-1}\ \ i = 1,\ldots, n_C \end{gathered} \tag{20}$$

Siguiendo la ley de la ecuación (20) se deducen los umbrales para la capa de salida:

$$\begin{gathered} u_i^C(n) = u_i^C(n-1) + \alpha \Delta_i^C(n) \\ para\ \ i = 1,\ldots, n_C \end{gathered} \tag{21}$$

De una forma similar a la que se obtuvo la expresión para la modificación de los pesos de la capa oculta *C-1* a la capa de salida se obtiene la modificación de los pesos para las demás capas, quedando la expresión:

$$\begin{gathered} w_{kj}^C(n) = w_{kj}^C(n-1) + \alpha \delta_i^{C+1}(n) a_k^C(n) \\ para\ \ j = 1,\ldots, n_{C-1};\ k = 1,\ldots, n_C\ y\ c = 1,\ldots, C-2 \end{gathered} \tag{22}$$

donde:

$$\delta(n)_j^{c+1} = f'\left(\sum_{k=1}^{n_c} w_{kj}^c a_k^c + u_i^c\right)\sum_{i=1}^{n_{c+1}} \delta_i^{C+2}(n) w_{ji}^c \tag{23}$$

También se pueden obtener los pesos generalizando la ley de aprendizaje obteniéndose:

$$\begin{gathered} u_j^{C+1}(n) = u_j^{C+1}(n-1) + \alpha \delta_j^{C+1}(n) \\ para \quad j = 1,\ldots,n_{C+1} \quad y \quad c = 1,\ldots,C-2 \end{gathered} \tag{24}$$

Los pasos que componen el aprendizaje del perceptrón multicapa son:

1. Se inicializan los pesos y umbrales de la red. Generalmente es aleatoria.
2. Se toma un patrón *n* del conjunto de entrenamiento y se propaga hacia la salida de la red, donde el vector de entrada *X(n)* utiliza las ecuaciones (13), (14) y (15), para obtener así la respuesta de la red, *Y(n)*.
3. Se evalúa el error cuadrático cometido por la red para el patrón *n* utilizando la ecuación (17).
4. Se aplica la regla delta generalizada para modificar los pesos y umbrales de la red de la siguiente manera:
    1. Se calculan los valores δ para todas neuronas de la capa de salida utilizando la ecuación (17);
    2. Se calculan los valores δ para el resto de las neuronas de la capa de la red utilizando la ecuación (23);
    3. Se modifican los pesos y umbrales de la red siguiendo las ecuaciones (20) y (21) para los pesos y umbrales de la capa de salida y (22) y (23) para el resto de los parámetros de la red.
5. Se repiten los pasos 2, 3 y 4 para todos los patrones de entrenamiento, completando así una iteración o ciclo de aprendizaje.
6. Se evalúa el error total *E* (Ecuación (16)) cometido por la red.
7. Se repiten los pasos 2, 3, 4, 5 y 6 hasta alcanzar un mínimo del error de entrenamiento, para lo cual se realizan m ciclos de aprendizaje [16].

La superficie que define el error *E* en función de los parámetros de la red es compleja y llena de valles y cimas Debido a la utilización del método del gradiente para encontrar un mínimo de esta función de error, se corre el riesgo de encontrar un mínimo local de esta función, y entonces no lograr dar con el entrenamiento adecuado. Una forma de evitar esto es aumentado el número capas ocultas, ya que cuando esto sucede se considera que la red no tiene capacidad de distinguir patrones distintos. Otro parámetro que influye en esto es la razón de aprendizaje α que eligiendo un valor muy bajo haría que decreciera tan lentamente el gradiente, que caería en un mínimo local. Por ello se debe tener cuidado en la selección de su valor, tomando en cuenta también que seleccionándose un valor muy alto podría jamás conseguir el mínimo global.

CAPÍTULO III: DISEÑO DE UN MODELO COMPUTACIONAL PARA UN SISTEMA DE RECONOCIMIENTO DE MATRÍCULAS

Durante esta tapa de diseño, se procedió a buscar las técnicas y algoritmos convenientes para el procesamiento y reconocimiento de imágenes de matrículas, de tal manera que se puedan cumplir los objetivos específicos.

Para el proceso de búsqueda se dividió el sistema de reconocimiento en cuatro etapas que son las básicas en un modelo como este. Las etapas entonces a diseñar son:

a)Etapa de Preprocesamiento: Se busca la adecuación de la imagen. El objetivo es eliminar de la imagen cualquier tipo de ruido, imperfección o información que no pertenezca al caracter o al conjunto de caracteres.

b)Etapa de Segmentación: El objetivo de esta etapa es dividir o fragmentar la imagen en una serie de componentes de la misma, conformados cada uno por un caracter.

c)Etapa de Caracterización: Mediante esta etapa se busca obtener una representación de los caracteres a reconocer más eficiente. Es decir se busca reducir la información propia de cada caracter, para así tener un menor procesamiento y darle mayor poder discriminatorio al proceso.

d)Etapa de Reconocimiento: Es donde se hace la clasificación de las imágenes de los caracteres y se pasan a texto editable.

Basado en el contenido del Capítulo 2, referente a procesamiento de imágenes y reconocimiento, y algunas pruebas realizadas, se seleccionará el método o algoritmo más adecuado para cada una de las etapas antes mencionadas.

3.1 Selección de Métodos y Algoritmos Para la Etapa de Preprocesamiento

Esta etapa fue dividida en cuatro subetapas en las que se aplicarán algoritmos y métodos para procesar la imagen del vehículo, y obtener una conformada por solo la matrícula. Las subetapas son: conversión a escala de grises, extracción de área útil, extracción de matrícula y binarización.

3.1.1 Subetapa Conversión a Escala de Grises

En esta etapa se busca pasar la imagen a color en espacio RGB a escala de grises. Según lo analizado en la Sección 2.1.2 del capítulo anterior, existen dos formas de hacer la conversión a escala de grises; una es hallando la componente Y del espacio YIQ para cada pixel y la otra promediando las Componentes R, G, y B para cada pixel. En las siguientes imágenes se muestra la conversión de una imagen a color a una imagen en escala de grises, utilizando los dos métodos conocidos:

Figura 18: Imagen original para convertir a escala de grises

Figura 19: Imagen convertida a escala de grises con mediante promedio de componentes RGB

Figura 20: Imagen convertida a escala de grises usando componente Y del espacio YIQ

No es necesario hacer un estudio profundo de esta etapa; ambas formas conllevan un costo computacional prácticamente idéntico, ya que en ambas se toma cada pixel de la imagen y se le aplica la respectiva formula de conversión vistas en la sección 2.1.2 del capítulo anterior, cuyas complejidades son similares. En cuanto a las pruebas realizadas se puede observar que la imagen de la Figura 19 es similar a la imagen de la Figura 20, excepto que la imagen de la Figura 19 presenta tonos más obscuros y por tanto es más opaca que la imagen en la Figura 20. Si se observa la imagen original en la Figura 18 entonces se puede notar que es una imagen con un alto brillo por tanto se dice que la imagen de la Figura 20 representa mejor la luminosidad de la imagen.

Cabe acotar que en el tratamiento de imágenes se utiliza por lo general la conversión correspondiente a la componente Y del espacio YIQ, ya que esta asemeja al comportamiento del ojo humano dándole más peso a los colores rojos y menos a los azules. Por tanto la conversión que se va a utilizar es la de la componente Y del espacio YIQ.

3.1.2 Subetapa Extracción de Área Útil

En esta etapa se busca eliminar partes de la imagen que no pertenezca a la matrícula y que pueden resultar ruidosas para las etapas posteriores. Esta etapa puede verse como un corte grueso a la imagen donde no se requiere mayor precisión ya que solo lo que se busca es tener menos área inútil de imagen.

Para la extracción del área útil se hace uso de algoritmos de extracción de bordes, que básicamente consiste en aplicar filtrado direccional o morfológico (ver Sección 2.2.1) a la imagen. En principio se tiene una imagen contentiva del auto con su respectiva matrícula y el resto de la imagen que tiene en este punto solo se tiene el conocimiento de que habrá bordes pronunciados en las partes que conforman los límites del vehículo, es decir, entre el chasis y el ambiente. Además habrán bordes pronunciados en la zona del parachoques y en los alrededores de la placa, sobre todo en aquellas que se les fue instalado algún marco.

En las siguientes imágenes (cuyo color fue invertido para facilitar la visualización) se observa el efecto de los filtros partiendo de la imagen original es la imagen de la Figura 20:

Figura 21: Imagen de automóvil con filtro Roberts dirección horizontal

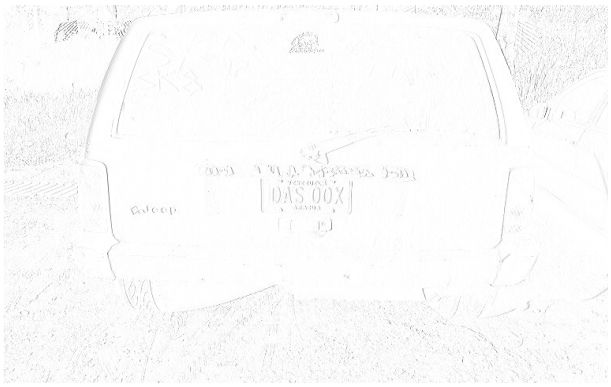


Figura 22: Imagen de automóvil con filtro Roberts dirección vertical

Figura 23: Imagen de automóvil con filtro Prewitt dirección horizontal

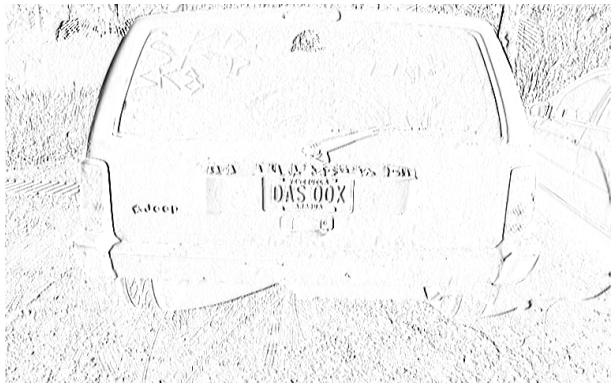


Figura 24: Imagen de automóvil con filtro Prewitt dirección vertical

Figura 25: Imagen de automóvil con filtro Sobel dirección horizontal

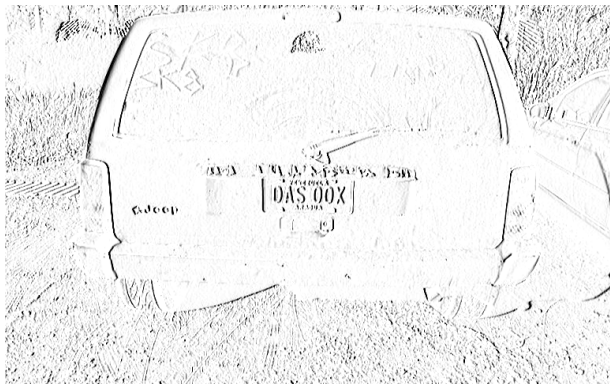


Figura 26: Imagen de automóvil con filtro Sobel dirección vertical

Figura 27: Imagen de automóvil con filtro Frei-Chen dirección horizontal

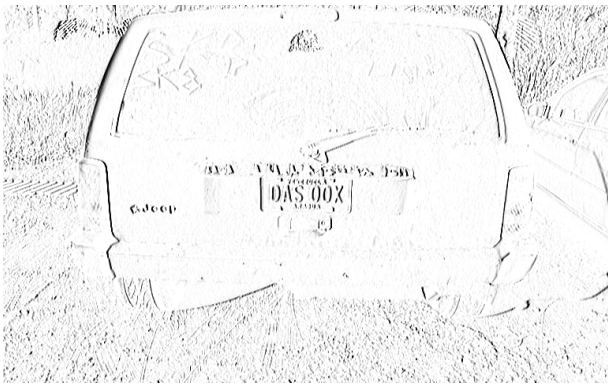


Figura 28: Imagen automóvil con filtro Frei-Chen dirección vertical

Figura 29: Imagen automóvil con gradiente morfológico por dilatación

Los filtros direccionales se basan en resaltar los puntos de mayor cambio de intensidad en una dirección determinada, por tanto se debe aplicar un kernel según la dirección de interés. Para el caso de este diseño se desean los bordes verticales y horizontales por tanto se aplican las máscaras en esas direcciones para cada filtro. De la Figura 21 a la Figura 28 se tienen imágenes a las que se le hicieron pruebas con filtros direccionales en las direcciones verticales y horizontales. Por otra parte el filtro morfológico resalta los bordes en una sola operación como muestra la prueba realizada cuyo resultado se observa la Figura 29, porque la operación que conlleva el gradiente morfológico por dilatación, utilizado en este caso, resalta los valores más claros por la operación de dilatación, luego mediante la sustracción atenúa estas zonas

claras y resalta las zonas pequeñas obscuras, que representan en mayor medida a los bordes.

Según lo visto en la Sección 2.2.1 del capítulo anterior el filtro de Sobel y el de Frei-Chen proporcionan un suavizado que ayuda en la eliminación de ruido. En cuanto a la detección de bordes basado en lo visto en la sección 2.2.1 el filtro Prewitt es el que mejor detecta bordes horizontales de los filtros direccionales, el Sobel los bordes verticales y el Frei-Chen intenta llegar a un compromiso entre ambos.

Observando las figuras resultado que se obtuvieron al hacer pruebas con los diferentes filtros se observa que:

Para la figura con filtro Roberts en la dirección horizontal mostrado en la Figura 21, la detección de bordes no es efectivo, ya que puede observarse que se resaltan tenuemente. Para el mismo filtro pero en la dirección vertical, según lo observado en la Figura 22 también se tiene que la detección de bordes son tenues.

Para la imagen con filtro Prewitt en la dirección horizontal mostrado en la Figura 23, los bordes presentan gran realce, en los que se pueden distinguir los bordes importantes en el vehículo. Para este mismo filtro en la dirección vertical, según lo observado en la Figura 24, también presenta gran realce de dichos bordes, ya que se pueden observar los bordes de interés en dicha dirección.

Para el caso de la imagen con filtro Sobel en la dirección horizontal mostrado en la Figura 25, presenta realce de bordes un poco más marcado que el Prewitt, se puede distinguir que los bordes de interés en el vehículo aparecen ligeramente más gruesos, pero a su vez se realzan más partes de la imagen que no son de interés. Para este mismo filtro en la dirección vertical, según lo observado en la Figura 26, el realce de los bordes en dicha dirección, parece no haber mayor diferencia con el Prewitt, por tanto se observan los bordes de interés resaltados.

En cuanto al filtro Frei-Chen en la dirección horizontal mostrado en la Figura 27, se observa que presenta tanto realce de bordes en esa dirección como el filtro Sobel, con la diferencia que las partes de la imagen que son de menor interés aparecen con menor realce. Para este mismo filtro en la dirección vertical, según lo

observado en la Figura 28, se tiene un realce similar al realce de los filtros Prewitt o Sobel.

Por último para la imagen con filtro morfológico por dilatación en la dirección horizontal mostrado en la Figura 29, se observa que en tanto para la dirección vertical como para la horizontal se marcan los bordes mejor que para los demás filtros, además de que las zonas de menor interés son atenuadas.

A través de la Tabla 1 se hace una comparación entre 5 filtros para realce de bordes. La comparación se hará mediante una puntuación que se le asignará a cada característica que se tome en cuenta, la escala de puntuación será con los números naturales desde el 1 hasta el 5 (1, 2, 3, 4 y 5), donde 1 es lo menos aceptable o conveniente y 5 es el nivel máximo de aceptabilidad, basado en los análisis de los resultados obtenidos de cada método.

Características	**Filtro Frei-Chen**	**Filtro Sobel**	**Filtro Prewitt**	**Filtro Roberts**	**Gradiente morfológico**
Coste computacional	3	3	3	3	5
Localización de bordes verticales	4	3	4	1	5
Localización de bordes horizontales	4	4	3	1	5
Puntuación Promedio	3,67	3,330	3,33	1,67	5

Tabla 1: Comparación de filtros para extracción de bordes

A través de la Tabla 1, se puede observar que el candidato más adecuado para la selección es el gradiente morfológico.

3.1.3 Subetapa Extracción de Matrícula.

En esta subetapa se busca eliminar todo resto de imagen que no pertenezca en si a la matrícula. Esto se puede hacer valiéndose de dos métodos: uno es la extracción de bordes y el otro es un método combinado morfológico que consta de tres pasos: aplicar gradiente morfológico, luego hacer un binarizado y por último aplicar operaciones de dilatación y erosión para resaltar la posición de la matrícula. En las siguientes imágenes se puede observar pruebas realzadas que muestran las bondades de ambos métodos:

Figura 30: Imagen original proveniente de etapa anterior

Figura 31: Imagen original con gradiente morfológico de Beucher

Figura 32: Imagen con gradiente de Beucher binarizada

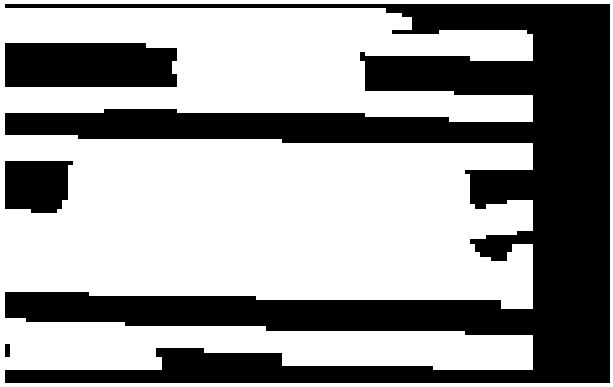


Figura 33: Imagen binarizada con dilatación para selección de algoritmos

Figura 34: Imagen con erosión para eliminar objetos de gran tamaño pero de menos tamaño que la matrícula para selección de algoritmos

Figura 35: Imagen con erosión para eliminar objetos mucho más pequeños que el tamaño de la matrícula

Se tiene una imagen origen en la Figura 30, a la que se le aplicó el gradiente morfológico de Beucher para resaltar sus bordes, cuyo resultado es la Figura 31. A simple vista se observa que para este caso, solo con el realce de bordes no se podría obtener la ubicación de la matrícula, por haber varios bordes con magnitudes similares a los bordes inmediatos a la matrícula.

Entonces haciendo uso de la imagen con bordes realzados ya obtenida, se prueba el método combinado morfológico. Se empieza por aplicarle un binarizado a la imagen de la Figura 31, y se obtiene la Figura 32; luego esta imagen se le aplica un procesos de dilatación con un elemento estructurante horizontal un poco más alargado que el ancho de los caracteres y se obtiene la Figura 33; seguido se le aplica un erosión con un elemento estructurante vertical un poco más largo que el alto de los caracteres y se obtiene lo que se observa en la Figura 34; por último se le aplica una erosión con un elemento estructurante vertical un poco más pequeño que el alto de los caracteres y se obtiene finalmente la imagen de la Figura 35. En la imagen de la Figura 35 se puede notar que ahora si se tiene la ubicación más precisa de la matrícula.

En la Tabla 2 se hace una comparación entre estos métodos se hace una

comparación dando puntajes según las características observadas, basado en los resultados analizados de las pruebas realizadas.

Características	Extracción de borde	Método morfológico
Coste computacional	5	1
Susceptible a confundir matrícula con elementos ajenos a la misma	4	5
Precisión en la localización	2	5
Puntuación Promedio	2,75	3,67

Tabla 2: Comparación de métodos para localización de matrícula

A través de la Tabla 2 , se puede observar que el método más conveniente es el método morfológico.

3.1.4 Subetapa Binarización.

En esta subetapa el objetivo es convertir a blanco y negro la imagen obtenida de la etapa anterior. Dicha imagen está compuesta en su mayoría por la matrícula, y su color está en escala de grises. Para binarizar se debe hallar un umbral a partir del cual se puede convertir o simplificar la escala de grises en únicamente dos tonos de color blanco y negro. Los métodos para umbralización a utilizar están asociados o utilizan el histograma de la imagen, por lo que, se utilizaron algunas imágenes para observar los tipos de histogramas que se pueden tener. A continuación se muestran algunas imágenes con sus respectivos histogramas:

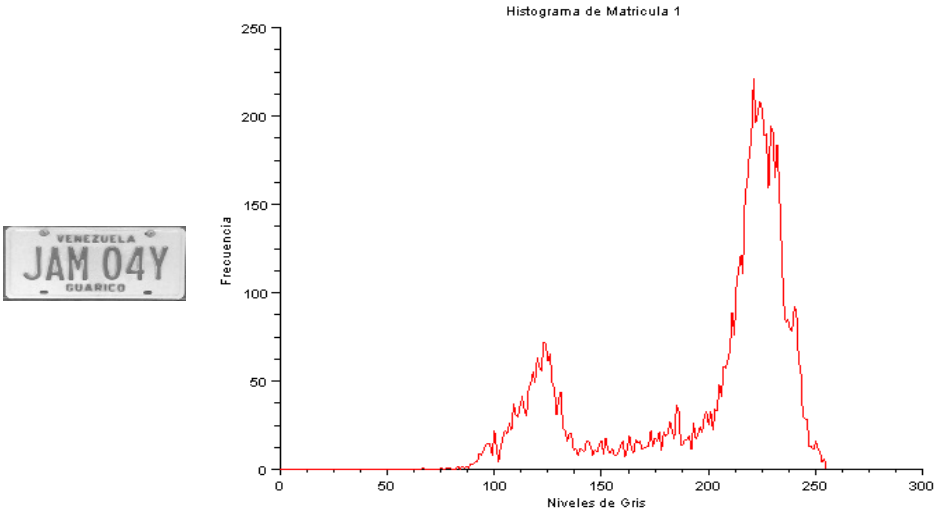


Figura 36: Histograma de matrícula de prueba 1

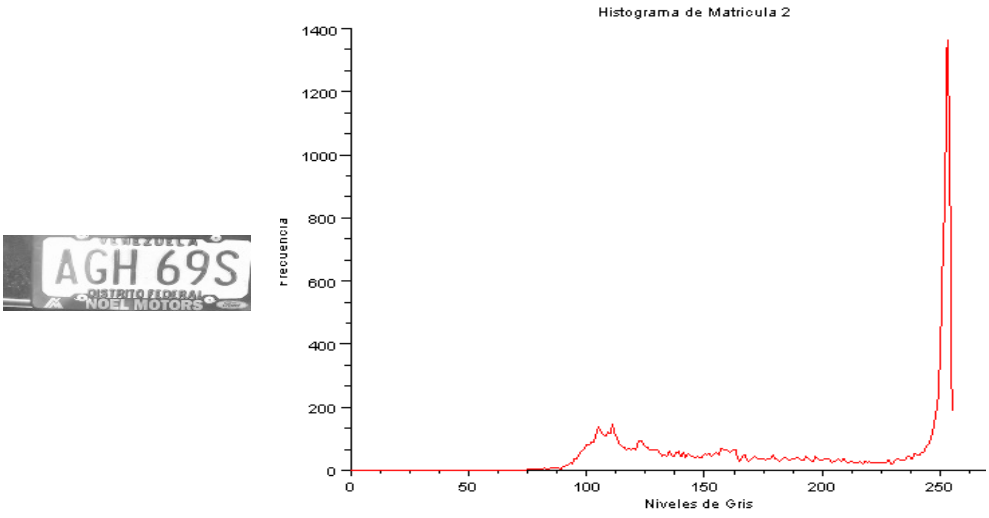


Figura 37: Histograma de matrícula de prueba 2

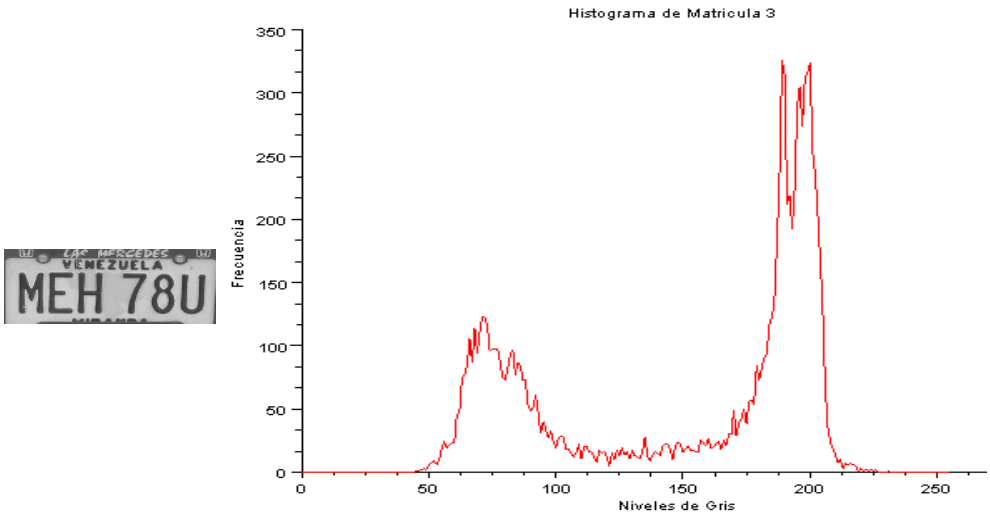


Figura 38: *Histograma de matrícula de prueba 3*

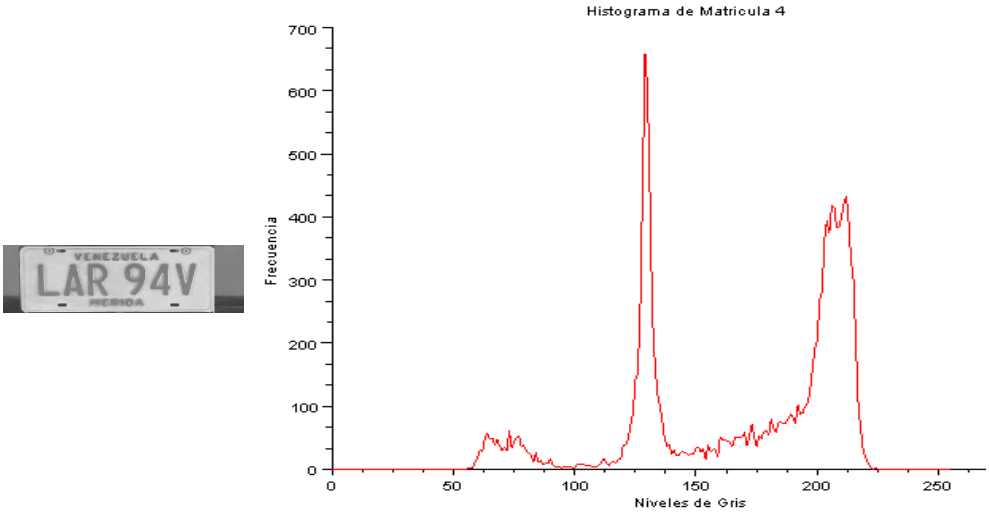


Figura 39: Histograma de matrícula de prueba 4

De las figuras anteriores se observa que: la matrícula de la Figura 36 presenta un histograma bimodal donde el fondo representa un pico simétrico dominante, la matrícula de la Figura 37 presenta un histograma bimodal igualmente donde el fondo se ve perfectamente predominante al punto de aproximarse a un histograma unimodal, la matrícula de la Figura 38 representa un histograma bimodal

con fondo representado con un pico simétrico, por último la matrícula de la Figura 39 presenta un histograma multimodal donde se observa un pico dominante y dos picos menores.

En la Tabla 3 se hace una comparación entre métodos de binarización dando puntajes según las características observadas.

Características	**P-Tile**	**Búsqueda Gaussiana**	**Isodata**	**Simetría de Fondo**	**Método del triángulo**
Coste computacional	5	3	2	3	4
Conocimiento Previo de la Imagen	1	1	5	1	2
Capacidad de procesar histogramas de múltiples modos	2	2	5	1	1
No requiere de técnicas de suavizado de histograma	5	1	5	1	4
Puntuación Promedio	3,25	1,75	4,25	1,5	2,75

Tabla 3: Comparación de métodos para hallar umbral de binarización

A través de la Tabla 3, se puede observar que el método más conveniente es el método Isodata.

3.2 Selección de Métodos y Algoritmos Para la Etapa de Segmentación

En esta etapa el objetivo es obtener la imagen de cada caracter que conforma la matrícula de manera separada. Según lo analizado Sección 2.3 del Capítulo anterior la opción a seguir para lo que se quiere en esta tapa, es utilizar proyecciones verticales para ubicar los caracteres segmentar los caracteres, es decir, hacer uso del

histograma de la suma de los pixeles en la dirección vertical. Es viable esta forma porque la imagen de cada caracter es negra con fondo blanco, donde cada caracter esta separado una cierta distancia, por lo que sería sencillo discriminarlos mediante este método.

3.3 Selección de Métodos o Algoritmos Para la Etapa de Caracterización

El objetivo de la etapa es representar los caracteres de una forma más compacta, tratando de resaltar sus características y así darle mayor efectividad a la etapa de reconocimiento. Lo que se busca en si es reducir principalmente el tamaño de la información de la imagen de cada caracter. Si cada caracter fuese enviado en bruto al reconocimiento esto traería como consecuencia un alto costo computacional, ya que si tenemos una imagen de dimensiones mxn, cada uno de los pixeles que la conforman tendrían que ser trasladados a un vector cuyas dimensiones serian una fila y mxn columnas. Este vector antes mencionado es de gran tamaño lo cual implica una alta probabilidad de falla a la hora de reconocerlos.

Para ello se recurre a algoritmos de caracterización para reducir y volver más eficiente el reconocimientos de los caracteres. Entre algunos métodos de caracterización que son aplicables a esta etapa según lo analizado en la Sección 2.3. del capítulo anterior, están el método basado en diezmado y el método de características binarias.

El ***método basado en diezmado*** busca reducir el tamaño de la información de la imagen dividiéndola en rectángulos de $m0xn0$, rectángulos que pasan a ser un único pixel, cuyo valor en este caso debe ser binario también, por lo que entonces se debe definir una operación acorde a esto. En la Sección 2.3 se había mencionado que la operación podría ser el promedio de los pixeles del rectángulo seleccionado, pero esta operación en la mayoría de los casos conlleva a valores mayores que cero y menores que uno, por tanto no binarios, y esa forma de representar la información no es adecuada para lo que se requiere. Por lo tanto, si se considerase la definición de un

umbral a partir del cual si la operación es mayor se considera uno y si es menor se considera cero, entonces se obtiene finalmente un único pixel de valor binario.

Este método da un vector de dimensiones reducidas con respecto al vector que se tendría al procesar la imagen original, pero con una alta posibilidad de confundir un caracter con otro, ya que no resalta directamente sus características, si no que más bien los valores de las componentes de este vector dependen del tamaño, posición y hasta el ruido que esté presente en la imagen del caracter.

El ***método de características binarias*** se basa en extraer características propias conocidas de la imagen, con el fin de reducir la cantidad de información a procesar y a la vez aumentar la capacidad de discriminación a la hora del reconocimiento. Algunas características básicas que se pueden notar en un caracter y que son objeto de discriminación de este método son: transiciones verticales y horizontales, número de agujeros, número de puntos finales y número de puntos de árbol, tal como se explicará mas adelante. Estas características podrían no ser suficientes por lo que se podrían definir algunas más para hacer efectivo el método, lo importante a destacar es que este método va a requerir el uso de algunos otros métodos para lograr obtener las características, como por ejemplo algunas características van a requerir de métodos como los de esqueletización y etiquetado, y el hecho de discriminar cada una de estas características conlleva un coste computacional más alto comparado con el que se tendría con el método anterior.

En la Tabla 4 se hace una comparación entre los métodos de caracterización dando puntajes según las características observadas.

Características	Método basado en diezmado	Método características binarias
Coste computacional	5	3
Inmunidad al ruido	1	5
Capacidad de discriminación	2	5
Puntuación Promedio	2	4,33

Tabla 4: Comparación de métodos de caracterización

A través de la Tabla 4, se puede observar que el método más conveniente es el método de características binarias.

Luego de haberse determinado que el método de características binarias es el más apropiado, se deben determinar las características que van a ser extraídas. A continuación las características a ser determinadas para cada patrón y la forma en que se distribuirán en el vector resultado:

- Transiciones Verticales y horizontales: Se basa en calcular los ejes de la imagen normalizada. Para ello basta con tomar el tamaño del eje de las "x" y dividirlo entre 2, y realizar lo mismo con el de las "y". Ahora solo hay que recorrer los ejes y contar el número de transiciones que existen. El número de contactos se calcula dividiendo entre 2 el número de transiciones.

  Entre las características binarias se representan:

  - Característica 1:

    Verdadero: si el número de contactos con el eje "x" es menor que 2.

    Falso: en cualquier otro caso.

  - Característica 2:

Verdadero: si el número de contactos con el eje “x” es igual que 2.

Falso: en cualquier otro caso.

- Característica 3:

  Verdadero: si el número de contactos con el eje “y” es menor que 2.

  Falso: en cualquier otro caso.

- Característica 4:

  Verdadero: si el número de contactos con el eje “y” es igual que 2.

  Falso: en cualquier otro caso.

- Agujeros: La segunda característica que se va a tratar es si existen agujeros en la imagen del caracter. Esta característica es muy importante ya que divide muy claramente la imagen dentro de una clase u otra. Además es una característica que no depende de la escala, ni de la rotación; ni siquiera requiere del adelgazamiento de la imagen. Una vez obtenidos los componentes, el número de agujeros será el número de componentes menos 1 (la componente exterior).

  - Característica 5:

    Verdadero: si el número de agujeros es igual a 0.

    Falso: en cualquier otro caso.

  - Característica 6:

    Verdadero: si el número de agujeros es igual a 1.

    Falso: en cualquier otro caso.

  - Característica 7:

Verdadero: si el número de agujeros es igual a 2.

Falso: en cualquier otro caso.

- Puntos finales y puntos de árbol: Los puntos finales son aquellos donde finaliza la tinta del caracter y los puntos de árbol son aquellos donde un pixel se ramifica. Para su cálculo se examinan las 8 adyacencias de cada pixel y según este dato se determina si son punto de árbol, finales, o nada. Si el número de píxeles adyacentes con valor negro es igual a 1, entonces será un punto final. Por otro lado si este número es mayor o igual a 3 será un punto de árbol. A continuación se usa un acumulador para obtener los puntos predominantes de los puntos de árbol y finales, y obtenemos las características:

  - Característica 8+ n:

    Verdadero: si el número de puntos finales es igual a “n”.

    Falso: en cualquier otro caso.

    Para valores de n = 0, 1, 2, 3 .

  - Característica 12 + n:

    Verdadero: si el número de puntos de árbol es igual a “n”.

    Falso: en cualquier otro caso.

    Para valores de n = 0, 1, 2, 3.

- Trazos horizontales, verticales o diagonales: Cantidad de líneas horizontales, verticales o diagonales según sea el tipo de trazo que posee el caracter. Para esta característica es necesario que la imagen esté esqueletizada, y como estos métodos a veces dan como resultados trazos no tan perfectos se definen las siguientes características:

Caso: Trazos Horizontales

- Característica 16:

  Verdadero: si el número de trazos horizontales es mayor que cero y menor o igual a dos.

  Falso: en cualquier otro caso.

- Característica 17:

  Verdadero: si el número de trazos horizontales es mayor que dos y menor o igual a cuatro.

  Falso: en cualquier otro caso.

- Característica 18:

  Verdadero: si el número de trazos horizontales es mayor que cuatro y menor o igual a seis.

  Falso: en cualquier otro caso.

- Característica 19:

  Verdadero: si el número de trazos horizontales es mayor que seis y menor o igual a ocho.

  Falso: en cualquier otro caso.

- Característica 20:

  Verdadero: si el número de trazos horizontales es mayor que ocho.

  Falso: en cualquier otro caso.

Caso: Trazos Verticales

- Característica 21:

  Verdadero: si el número de trazos verticales es mayor que cero y menor o

igual a dos.

Falso: en cualquier otro caso.

- Característica 22:

  Verdadero: si el número de trazos verticales es mayor que dos y menor o igual a cuatro.

  Falso: en cualquier otro caso.

- Característica 23:

  Verdadero: si el número de trazos verticales es mayor que cuatro y menor o igual a seis.

  Falso: en cualquier otro caso.

- Característica 24:

  Verdadero: si el número de trazos verticales es mayor que seis y menor o igual a ocho.

  Falso: en cualquier otro caso.

- Característica 25:

  Verdadero: si el número de trazos verticales es mayor que ocho.

  Falso: en cualquier otro caso.

Caso: Trazos Diagonales

- Característica 26:

  Verdadero: si el número de trazos diagonales hacia la izquierda es mayor que cero y menor o igual a dos.

  Falso: en cualquier otro caso.

- Característica 27:

Verdadero: si el número de trazos diagonales hacia la izquierda es mayor que dos y menor o igual a cuatro.

Falso: en cualquier otro caso.

- Característica 28:

  Verdadero: si el número de trazos diagonales hacia la izquierda es mayor que cuatro y menor o igual a seis.

  Falso: en cualquier otro caso.

- Característica 29:

  Verdadero: si el número de trazos diagonales hacia la izquierda es mayor que seis y menor o igual a ocho.

  Falso: en cualquier otro caso.

- Característica 30:

  Verdadero: si el número de trazos diagonales hacia la izquierda es mayor que ocho.

  Falso: en cualquier otro caso.

- Característica 31:

  Verdadero: si el número de trazos diagonales hacia la derecha es mayor que cero y menor o igual a dos.

  Falso: en cualquier otro caso.

- Característica 32:

  Verdadero: si el número de trazos diagonales hacia la derecha es mayor que dos y menor o igual a cuatro.

  Falso: en cualquier otro caso.

- Característica 33:

  Verdadero: si el número de trazos diagonales hacia la derecha es mayor que cuatro y menor o igual a seis.

  Falso: en cualquier otro caso.

- Característica 34:

  Verdadero: si el número de trazos diagonales hacia la derecha es mayor que seis y menor o igual a ocho.

  Falso: en cualquier otro caso.

- Característica 35:

  Verdadero: si el número de trazos diagonales hacia la derecha es mayor que ocho.

  Falso: en cualquier otro caso.

Como es evidente el uso de algún algoritmo para esqueletizar, se hace uso del ***método de Zhang Zuen***, por ser el método más utilizado en aplicaciones que requieran estudiar solo el esqueleto de la imagen, tales como procesamiento de imágenes de huellas dactilares, procesamiento de imágenes de lenguaje para sordos, entre otras.

3.4 Selección de Métodos o Algoritmos Para la Etapa de Reconocimiento

El objetivo de esta etapa es traducir cada caracter contenido en las imágenes de la etapa anterior en texto editable. Para lograrlo se debe hacer uso de clasificadores (ver Sección 2.3), que para los fines de cumplir uno de los objetivos específicos de este trabajo se seleccionaron previamente las redes neuronales como objeto de estudio. De las redes neuronales existentes se seleccionó el perceptrón multicapa, ya que se requiere una red neuronal con aprendizaje supervisado con capacidad de

clasificar conjuntos que no son linealmente separables, es decir, que se requiere de redes multicapa, además de ser ampliamente utilizado para resolver problemas de asociación de patrones, segmentación de imágenes, compresión de datos, y es muy común encontrarlas en aplicaciones de reconocimiento.

La función de activación a utilizar en este tipo de red neuronal es la de tipo sigmoidea como se explicará en la sección 2.4, y de este tipo de señal la que se ajusta más a la aplicación es la función sigmoidal, ya que se tienen entradas y salidas binarias y el rango de dicha función va de cero a uno. Para el número de capas, el número de neuronas y la constante de aprendizaje no se tiene una forma algorítmica de obtener estos parámetros, si no que se hallan de manera empírica.

Adicionalmente se tiene previsto entrenar dos redes neuronales, una para letras y una para números. Esto por las siguientes razones:

1. Una primera razón es que se podría evitar confusiones entre números y letras como serían los casos de la letra "I" y el número "1", la letra "B" y el número (8), y el de la letra "O" con el número "0", dado que luego de su caracterización, sus vectores característica serían similares, porque comparten características tales como: número de agujeros, número de transiciones, números de trazos, etc.

2. También el separar el reconocimiento de números y letras facilita el entrenamiento, ya que son menos patrones en cada red, además evita el problema de convergencia dado que para caracteres distintos, se tienen vectores característicos similares.

CAPÍTULO IV: IMPLEMETACIÓN DE UN MODELO COMPUTACIONAL PARA UN SISTEMA DE RECONOCIMIENTO DE MATRÍCULAS

Durante esta etapa de implementación, se procedió a programar cada una de las tapas definidas en el capítulo anterior con los métodos o algoritmos seleccionados para cada una de las mismas, de tal manera que se puedan cumplir los objetivos específicos de este trabajo.

Para realizar dicha implementación se hizo el uso de la herramienta computacional Scilab. Esta herramienta fue utilizada por requerimientos del Centro Nacional de Desarrollo e Investigación en Telecomunicaciones (CENDIT).

4.1 Herramienta Computacional Scilab

Scilab es un software libre para cálculo numérico que ofrece un entorno informático de gran alcance para aplicaciones científicas y de ingeniería. Este software recoge tanto las necesidades industriales como los avances científicos, y cubre un amplio espectro de áreas tales como la aeroespacial, finanzas, automotriz, química, biología, medicina, entre otras [18].

Scilab posee un lenguaje de alto nivel que permite el acceso a estructura de datos avanzadas y funciones gráficas 2-D y 3-D.

Scilab ofrece un entorno como el que se observa en la Figura 40 que consta de una consola en la que se puede escribir comando o hacer cálculos directamente a la vez que se obtiene el resultado en esta misma [18].

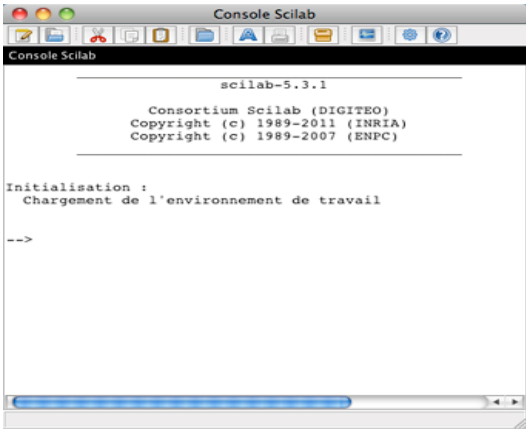


Figura 40: Entorno de consola Scilab

En Scilab también viene un editor integrado llamado *Scinotes* cuyo entorno se observa en la Figura 41, que permite una edición sencilla de script que pueden ser ejecutados en el ambiente Scilab.

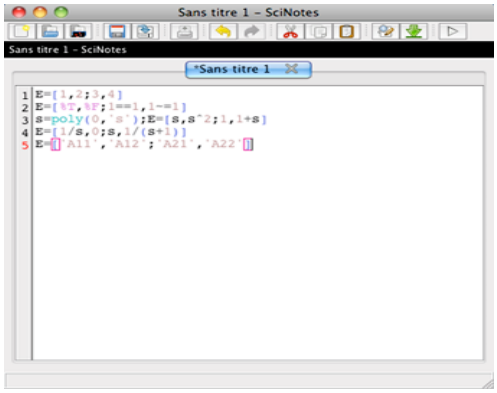


Figura 41: Entorno de edición Scinote

Por último, Scilab posee una gran cantidad de paquetes o toolboxes alojados en un centro de toolbox y más recientemente en lo que es llamado ATOMS que es un centro de gestión automática de módulos de donde se pueden instalar también paquetes, los cuales ofrecen una gran cantidad de herramientas para el desarrollo en

diversas áreas entre las cuales están procesamiento de imágenes, modelaje y control, entre otras.

Para la realización de este trabajo se utilizó Scilab versión 5.2.2, y los toolboxes SIVP versión 0.5.2 para procesamiento de imágenes y el ANN para la implementación de la red neuronal.

4.2 Implementación del Modelo Computacional en Scilab

Para la implantación se trabajó con las siguientes herramientas y respectivas especificaciones:

- Computador procesador Intel Pentium Dual E2160 de 1,80 Ghz y memoria RAM de 1 Ghz.
- Sistema operativo Linux Ubuntu versión 10.04.
- Fotografías de 494x768 pixeles.

Para empezar la implementación se toma en consideración que en un futuro este software va a ser parte de un sistema de reconocimiento de matrículas, por tanto será llamado como una función por un programa principal que gobierne el sistema antes mencionado o bien será controlado por una interfaz de usuario para fines de demostración. Por tanto la estructura que se le dará al programa estará dividida como se muestran en el Diagrama 1:

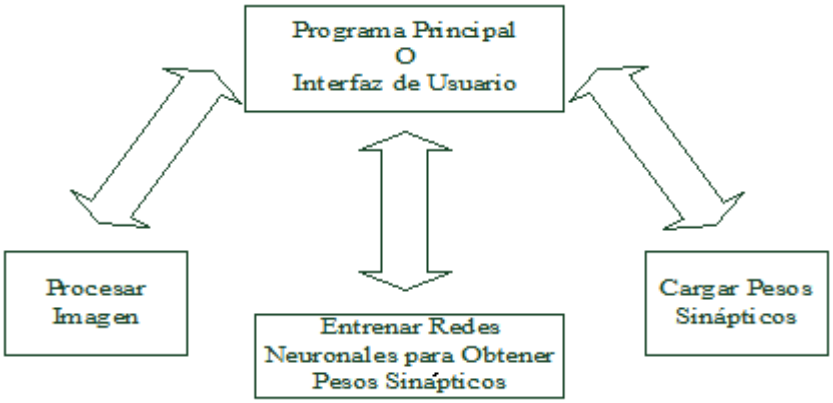


Diagrama 1: Diagrama de bloques de la estructura principal del software

Basado en el Diagrama 1 se programan estos tres bloques con el siguiente contenido:

a)Procesar Imagen

Este bloque se considera el más importante; en este se aplican prácticamente todos los algoritmos de procesamiento de imágenes seleccionados. Este bloque consiste en una vez recibida la imagen, se hace pasar por todas las etapas definidas para el OCR que son preprocesamiento, binarización, caracterización y reconocimiento, para así obtener finalmente los caracteres de la matrícula.

b)Entrenar Redes Neuronales para obtener Pesos Sinápticos

Este bloque se encarga de hacer el entrenamiento de las redes neuronales para así obtener los pesos sinápticos y almacenarlos en un archivo de donde se puedan extraer cada vez que se inicie el programa.

c) Cargar Pesos sinápticos

Este bloque se utiliza una sola vez, o al iniciar el programa, su función es cargar en el programa los pesos sinápticos para que sean utilizados por el bloque de procesamiento.

4.2.1 Implementación de Procesamiento de la Imagen

La estructura del programa para procesar la imagen se va a regir por la mostrada en el Diagrama 2. Con el fin de observar lo que realiza cada uno de los algoritmos, se toma la imagen que se muestra en la Figura 42 como la imagen de entrada a esta parte del programa.

Figura 42: Imagen de entrada como demostración en el bloque de reconocimiento

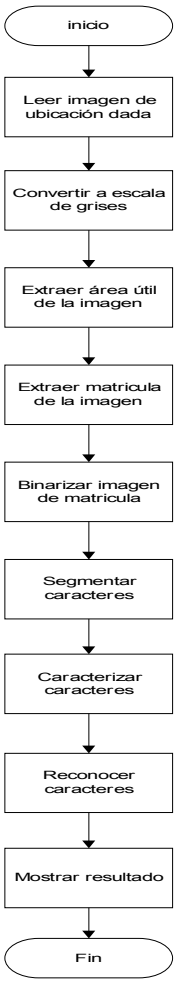


Diagrama 2: Procesamiento de la imagen

4.2.1.1 Leer Imagen de Ubicación Dada

El primer paso entonces es leer la imagen, esto se hace mediante una función del toolbox SIVP de Scilab denominada imread. Esta función procede de la siguiente manera:

$$imrgb = imread('dir') \tag{25}$$

donde *dir* es la dirección que esta ubicada la matrícula, e *imrgb* la variable donde se almacena una hipermatriz de dimensiones mxnx3 que contiene la imagen leída descompuesta en sus componentes RGB.

4.2.1.2 Conversión a Escala de Grises de Imagen RGB

Una vez leída la imagen de la ubicación dada es convertirla a escala de grises. Previamente en la sección 3.2.1 se había seleccionado para esta parte usar la conversión que proporciona la componente Y del espacio YIQ. Sin embargo el toolbox SIVP contiene una función para convertir a escala de grises la cual se denomina *rgb2gray()*. Esta función utiliza la conversión que se había seleccionado para llevar la imagen a escala de grises. La función es como sigue:

$$imgris = rgb2grey(imrgb) \tag{26}$$

Donde imrgb es la hipermatriz que contiene la imagen en sus componentes RGB e imgris es una matriz de dimensiones mxn en escala de grises, que en este caso tomando la imagen de la Figura 42 se tendrá un resultado similar al de la imagen que se muestra en la Figura 43:

Figura 43: Imagen resultado de la función rgb2gray en Implementación

4.2.1.3 Extracción del Área Útil de la Imagen

Con la imagen convertida a escala de grises se procede entonces a eliminar de la imagen las zonas que no son de interés. En la sección 3.1.2 se seleccionó el ***Gradiente de Beucher*** para resaltar los bordes de la imagen.

Para empezar esta etapa y con el fin de obtener rapidez en los algoritmos, se trabajará con una imagen reducida seis veces, de tal forma que se le aplican los métodos a la imagen reducida y luego de haber obtenido los resultados, es decir, las coordenadas en las que se seccionará la imagen, estas se multiplican por seis y entonces se hace el corte de la imagen original sin reducir en las coordenadas obtenidas. Este proceso es posible porque la transformación de reducción es una transformación lineal.

La reducción se hace mediante la función *imresize* la cual es como sigue:

$$imsal = imresize(iment, [nfil\ ncol], interp) \tag{27}$$

donde *iment* es una matriz de mxn que contiene la imagen a cambiar de tamaño, *nfil* es el número de pixeles de alto que quiere que se tenga la imagen de salida, *ncol* es el ancho que se quiere que tenga la imagen de salida, y por último un parámetro

opcional *interp* en donde se coloca el tipo de interpolación que se desea utilizar, este parámetro se puede obviar y la función trae como predeterminado utilizar la interpolación por vecinos más cercanos, que consiste en tomar el pixel más cercano, como el valor del pixel al que se le quiere determinar su valor. Por último la salida imsal será la matriz de salida que va a contener la imagen de tamaño *nfilxncol*.

```
73
74  for i=4:fil-3
75        for j=4:col-3
76
    ee=[im(i-3,j-1),im(i-3,j),im(i-3,j+1),im(i-2,j-2),im(i-2,j-1),im(i-2,j),im(
    i-2,j+1),im(i-2,j+2),im(i-1,j-3),im(i-1,j-2),im(i-1,j-1),im(i-1,j),im(i-1,j
    +1),im(i-1,j+2),im(i-1,j+3),im(i,j-3),im(i,j-2),im(i,j-1),im(i,j+1),im(i,j+
    2),im(i,j+3),im(i+1,j-3),im(i+1,j-2),im(i+1,j-1),im(i+1,j),im(i+1,j+1),im(i
    +1,j+2),im(i+1,j+3),im(i+2,j-2),im(i+2,j-1),im(i+2,j),im(i+2,j+1),im(i+2,j+
    2),im(i+3,j-1),im(i+3,j),im(i+3,j+1)];
77          imresult(i,j)=max(ee);
78        end
79  end
```

Figura 44: Código en Scilab para dilatación con elemento estructurante circular

Con la imagen reducida se procede a extraer los bordes mediante el gradiente de Beucher. Para ello primeramente se le aplica el filtro dilatación a la imagen con un elemento estructurante circular de radio 3 pixeles con una rutina que se programó en Scilab como se muestra en la figura Figura 44, donde *ee* es un vector que guarda el valor de los pixeles que están contenidos en la vecindad que define el elemento estructurante alrededor del pixel que están en la coordenada *(i,j)* de la imagen *im;* denotado en la rutina como *im(i,j)*. La variable *imresult* es una matriz que va almacenando el resultado de la función *max(),* la cual encuentra el valor máximo del vector *ee*.

Esta rutina debe dar un resultado similar al que se observa en la Figura 45:

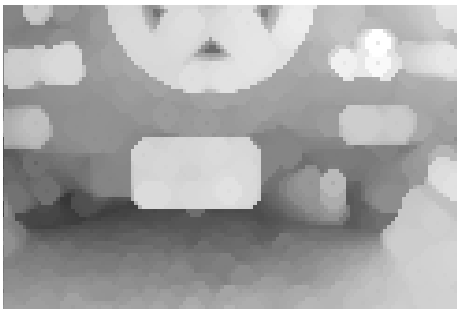


Figura 45: Imagen resultado luego de haber pasado por la rutina de dilatación en implementación

Para resaltar los bordes se hace la diferencia entre la imagen original y la imagen de la Figura 45. Para ello se hace uso de una función del toolbox SIVP denominada *imsubtract()* la cual es de la siguiente manera:

$$imdif = imsubtract(imresult, imo) \tag{28}$$

donde *imresult* es la imagen con filtro dilatación aplicado, e *imo* la imagen original reducida, luego *imdif* es la variable que guarda la matriz de la imagen con los bordes resaltados. El resultado de esta función será similar al que se observa en la Figura 46:

Figura 46: Imagen resultado con bordes resaltados por gradiente de Beucher en implementación

Ya con los bordes resaltados de la imagen con el fin de facilitar el trabajo se pasa la imagen a blanco y negro, es decir, se binariza pero el umbral se define aproximadamente a la mitad de la escala de iluminación, esto es, los pixeles cuyo valor en la imagen que estén por encima de 128 entonces serán substituidos por 255

y se le pondrá el valor 0. Este procedimiento se hace con la función del toolbox SIVP que se denota como *im2bw()* la cual es de la siguiente manera:

$$imbw = im2bw(im, umb); \tag{29}$$

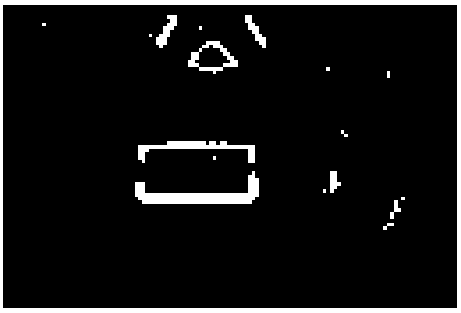


Figura 47: Imagen con bordes resaltados binarizada en el bloque extracción de área útil en implementación

donde *im* es la variable que contiene la imagen a binarizar, *umb* es una variable que va de 0 a 1 la cual representa el umbral, por tanto solo se toma el valor en escala de grises que se desea como umbral y entonces se divide entre 255 para así llevar a la escala de este parámetro. Por último *imbw* es la variable con contiene la imagen binarizada. Para este caso umb=128/255. El resultado de esta función será observado en la Figura 47:

Para extraer las coordenadas lo que se hace es dividir la imagen en cuatro sectores en la dirección vertical, luego sumar los pixeles de la matriz de la imagen, columna por columna, para luego buscar el valor máximo de esa suma en cada sector, y promediar los máximos de los sectores de más a la izquierda. Luego dicho promedio será la coordenada donde se hará el corte de la imagen. Promediar los máximos de los dos sectores más próximos a la derecha y también ese promedio va a ser una coordenada de corte pero hacia la derecha. Se hace exactamente lo mismo para la dirección horizontal.

Finalmente una vez obtenidas las coordenadas en la imagen reducida, se multiplican por seis y se tienen las coordenadas donde se hará el corte en la imagen original. Para ello se hace uso de una función del toolbox SIVP que se denomina

imcrop() la cual es de la siguiente manera:

$$imcut = imcrop(im, rect) \tag{30}$$

donde *im* es la imagen a cortar, *rect* es un vector de cuatro componentes*:* la primera representa el punto inicial del corte en la coordenada horizontal, la segunda representa el punto inicial de la coordenada vertical para el corte, la tercera el largo que va a tener el corte a partir de la coordenada horizontal inicial, y la cuarta el alto que va a tener a partir de la coordenada principal vertical.

El resultado de este bloque será como el que se observa en la Figura 48:

Figura 48: Imagen resultado de la extracción de área útil en implementación

4.2.1.4 Extracción de la Matrícula de la Imagen

Una vez eliminada algunas partes de la imagen que no aportaban información de la placa, se procede a extraer la matrícula. Para este procedimiento se empieza haciendo lo que se hizo en el bloque anterior, con la diferencia que la imagen de entrada original es la observada en la Figura 48 y además se le van a aplicar estos métodos a dicha imagen reducida tres veces su tamaño.

Entonces se empieza por aplicar gradiente de Beucher de la misma manera que para el bloque anterior con lo cual se tendrá una imagen como la que se observa en la Figura 49:

Figura 49: Imagen resultado de resaltar los bordes en el bloque extracción de matrícula en la implementación.

Luego esta imagen se binariza de la misma forma que la etapa anterior, obteniéndose la imagen de la Figura 50:

Figura 50: Imagen resultado de binarizar bordes resaltados en el bloque extracción de matrícula en implementación.

Con la imagen binarizada entonces se procede a continuar con el método morfológico, que permite resaltar la ubicación de la placa. Para ello se aplica a la imagen anterior operaciones morfológicas de dilatación y erosión para eliminar todo el ruido de la imagen binarizada, obteniendo la ubicación de la matrícula discriminada.

La primera operación morfológica a realizar es una dilatación con un

elemento estructurante lineal horizontal de ancho de 15 pixeles para poder rellenar la ubicación de la placa, ya que se requería un elemento estructurante tan ancho como el ancho de la separación entre cada caracter por la cual se programó la siguiente rutina en Scilab que se observa en la Figura 51:

```
112  for i=1:fil
113        for j=1:col-15
114
     ee=[impl(i,j),impl(i,j+1),impl(i,j+2),impl(i,j+3),impl(i,j+4),impl(i,j+5),i
     mpl(i,j+6),impl(i,j+7),impl(i,j+8),impl(i,j+9),impl(i,j+10),impl(i,j+11),im
     pl(i,j+12),impl(i,j+13),impl(i,j+14),impl(i,j+15)];
115              result1(i,j)=max(ee);
116        end
117  end
```

Figura 51: Código en Scilab para dilatación con elemento estructurante lineal horizontal

Dicha rutina es similar a la utilizada en el bloque anterior solo cambia el elemento estructurante. El resultado es la imagen que se observa en la Figura 52.

Figura 52: Imagen resultado de aplicar dilatación en el bloque extracción de matrícula en implementación.

La segunda operación morfológica a aplicar es una erosión con un elemento estructurante lineal vertical de alto siete pixeles, con el fin de eliminar formas que queden dentro de la imagen que puedan representar ruido. Se utiliza 7, ya que se determinó que las imperfecciones que quedaban resultado del filtro anterior podían

ser eliminadas con un elemento estructurante de esta longitud, además de que es una longitud menor a la matrícula, por tanto, preserva su ubicación, para lo cual se programó la siguiente rutina en la Figura 53:

```
123 for i=1:fil-7
124       for j=1:col
125
    ee=[impl(i,j),impl(i+1,j),impl(i+2,j),impl(i+3,j),impl(i+4,j),impl(i+5,j),i
    mpl(i+6,j),impl(i+7,j);
126               result2(i,j)=min(ee);
127       end
128 end
```

Figura 53: Código en Scilab para dilatación con elemento estructurante lineal vertical

Esta rutina es similar a la anterior de dilatación con la diferencia que hace uso de la función *min()* la cual encuentra el mínimo valor en el elemento estructurante. El resultado obtenido es el que muestra la imagen de la Figura 54:

Figura 54: Imagen resultado de aplicar erosión en el bloque extracción de matrícula en implementación.

Por último se vuelve a aplicar una erosión con un elemento estructurante de alto 5 pixeles y con eso se termina de eliminar los objetos de tamaño mediano que no sean de la matrícula, y el resultado es el siguiente:

Figura 55: Imagen resultado de aplicar segunda erosión en el bloque extracción de matrícula en implementación.

Por último, se suman los pixeles en la dirección horizontal para obtener su histograma, luego se definió una región de un ancho de noventa, porque es el ancho de la matrícula de mayor dimensión que se tenía, la cual se irá moviendo de izquierda a derecha en el histograma realizado, sumando todos los valores de las noventa componentes del vector que guarda el histograma sobre la cual esté situada la región dinámica definida. Para la dirección vertical se hizo exactamente lo mismo con la diferencia que se definió una región de treinta y dos componentes de alto, igualmente porque es el alto de la matrícula de mayor dimensión que se tenía. Al final se toman como coordenadas de corte, aquellas en las que tanto en la dirección horizontal como en la vertical, pertenezcan a la posición donde las respectivas regiones definidas dan máximo valor de suma, las cuales se multiplican por tres y se le aplican a la imagen original. Luego de haber aplicado estos algoritmos para obtener un acabado más limpio en la matrícula se tiene el resultado de la Figura 56:

Figura 56: Imagen resultado luego de pasar por bloque de extracción de matrícula en Implementación.

4.2.1.5 Binarización de la Imagen de la Matrícula

En esta sección del capítulo se quiere binarizar la imagen de la matrícula. Para ello se implementa el algoritmo del método Isodata para hallar el umbral de binarización para luego ser usado en la función *im2bw()*. Se implementó el método Isodata con la siguiente rutina en Scilab en la Figura 57:

```
52  while abs(umbfin-umbini)>0
53    umbini=umbfin;
54    umbizq=0;
55    umbder=0;
56    cuentaizq=0;
57    cuentader=0;
58    for i=1:fil1
59      for j=1:col1
60        aux=0;
61        while aux<>im5(i,j)
62          aux=aux+1;
63        end
64        if aux<umbini then
65          umbizq=aux+umbizq;
66          cuentaizq=cuentaizq+1;
67        else
68          umbder=aux+umbder;
69          cuentader=cuentader+1;
70        end
71      end
72    end
73    umbfin=int(((umbder/cuentader)+(umbizq/cuentaizq))/2);
74  end
```

Figura 57: Código en Scilab para algoritmo Isodata

donde *umbini* es el umbral inicial que fue calculado como el promedio de los valores de los pixeles que conforman la imagen de la matrícula y *umbfin* es el umbral al que se llegará cuando se cumpla la condición de parada. Este umbral final será divido entre 255 para luego ser usado en la función *im2bw()*. De esta función resulta lo que se observa en la Figura 58 :

Figura 58: Imagen resultado de binarización en implementación

4.2.1.6 Segmentación de Caracteres

Para la segmentación se los caracteres primeramente se programaron algunas rutinas de limpieza para facilitar el proceso. De esas rutina se tiene la imagen de la Figura 59:

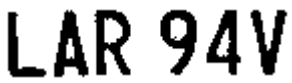


Figura 59: matrícula de la imagen luego de algoritmos de limpieza

Partiendo de la imagen anterior, primeramente se invierte el color de la imagen para facilitar el trabajo, luego se realiza la suma de los pixeles de cada columna de la matriz que la conforma, y se obtuvo el histograma que se observa en la Figura 60:

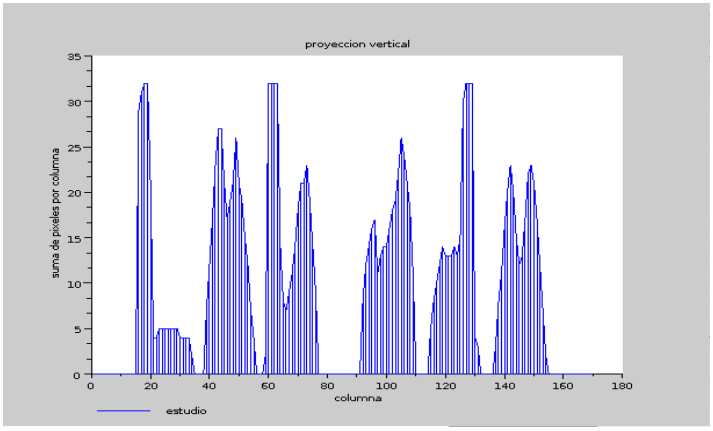


Figura 60: Histograma de suma de los de los valores de los pixeles de la imagen de la matrícula con colores invertidos limpia

Una vez obtenido el histograma se programó una rutina que evalúa columna por columna su valor, tomando la primera como la referencia, luego evalúa la siguiente columna y si es un espacio en blanco pues toma esa columna como referencia, si no evalúa la siguiente. Si luego de haber encontrado varias columnas cuyas sumas sean distintas de vació, se halla una columna vacía, se evalúa la distancia que hay entre la referencia y dicha columna, y si cumple con un ancho mayor o igual que trece columnas, que es el ancho máximo que puede tener un caracter entre las placas con las que se trabajaron, entonces se dice que hay un caracter, procediéndose a segmentarlo de la imagen y a almacenarlo. El resultado de la segmentación se muestra en la imagen de la Figura 61:

Figura 61: Imagen resultado de Implementación de segmentación de caracteres provenientes de imagen de la matrícula

4.2.1.7 Caracterización de los Caracteres Segmentados

Con los caracteres segmentados se procede entonces a implementar la extracción de las características definidas en la Sección 3.3 del Capítulo anterior.

Para empezar se hallan las primeras dos características que son ***transiciones verticales y transiciones horizontales***. Lo que se hizo consiste en una rutina, que ubica un puntero en la mitad superior de la matriz de la imagen y va recorriendo hacia abajo buscando cuando hay pases de blanco a negro acumulándolo en una variable, ubicando así los trazos verticales. Igualmente se hizo otra rutina pero con el centro de la izquierda, y el recorrido se hace hacia la derecha para obtener los trazos

horizontales. Dependiendo de la cantidad de trazos se coloca un valor en cada componente definida para estas características en el vector característica, según lo definido en la Sección 3.3 del Capítulo anterior. A continuación en la Figura 62 el algoritmo de las transiciones verticales programado en Scilab:

```
22  for i=1:fil-1
23    if improc(i,col+2)==1 & improc(i+1,col+2)==0 then
24            cty=cty+1;
25     end
26  end
```

Figura 62: Código en Scilab para determinar transiciones verticales

donde *improc* es la variable que contiene a la imagen y *cty* es la variable que acumula las transiciones. Un código similar al de los trazos horizontales se realizó para los trazos horizontales.

Para continuar se programó una rutina para hallar el ***número de agujeros*** que posee cada caracter. La rutina consiste en colocarse en un pixel y analizar el pixel vecino de arriba y el pixel vecino de la izquierda. La lógica es la siguiente: si ambos pixeles vecinos son negros y el pixel analizado es blanco entonces se crea una etiqueta y se se remplaza el valor de dicho pixel por el valor de la etiqueta, sí ambos vecinos son distintos entonces se observa si el mínimo entre ambos vecinos es cero; sí es así se reemplaza el pixel analizado por el máximo entre ambos vecinos, si no, se reemplaza por el mínimo. Siguiendo esto se programó otra rutina que hace exactamente lo mismo que la anterior, a diferencia que analiza los pixeles vecinos de arriba y a la derecha, además se realizó este proceso una y otra vez en un ciclo que se detendrá en la iteración donde no haya cambio en la imagen lo cual quiere decir que ya están todos los agujeros etiquetados. La parte final de la rutina cuenta el número de etiquetas que existen y se le resta uno que es la componente exterior, y este número corresponde al número de agujeros. A continuación la rutina inicial programada en Scilab en la Figura 63:

```
for i=2:fil-1
  for j=2:col-1
    if improc(i,j)==1 & improc(i-1,j)==improc(i,j-1) then
      if improc(i-1,j)<>0 then
        improc(i,j)=improc(i-1,j);
      end
      if improc(i-1,j)==0 then
        etiqueta=etiqueta+1;
        improc(i,j)=etiqueta
      end
    end
    if improc(i,j)==1 & improc(i-1,j)<>improc(i,j-1) then
      if min(improc(i-1,j),improc(i,j-1))<>0 then
        improc(i,j)=min(improc(i-1,j),improc(i,j-1));
      else
        improc(i,j)=max(improc(i-1,j),improc(i,j-1));
      end
    end
  end
end
```

Figura 63: Código en Scilab para primera rutina para encontrar número de agujeros

Con el número de agujeros determinado, se asignan entonces los valores correspondientes del vector característica, que fueron definidas en el capítulo anterior. Este procedimiento se hace para cada caracter y su vector característico es almacenado.

Seguidamente, se realizó una rutina para el cálculo de ***los puntos finales y puntos de árbol***. Como se había mencionado en la Sección 3.3 del Capítulo anterior, para hallar los puntos finales y los puntos de árbol, es necesario esqueletizar la imagen del caracter, por lo que entonces se implementó el algoritmo de Zhang Zuen el cual consiste en dos subiteraciones como se había explicado en la Sección 2.2.3 del C*apítulo II.* Esta rutina consiste en tomar un pixel y analizar sus ocho pixeles vecinos, luego verificar si se cumplen los criterios de borrado según sea la subiteración. Cada subiteración se va a repetir en un ciclo hasta que no hayan

cambios en la imagen, lo que quiere decir que la imagen está en su esqueleto. A continuación la primera subiteración programada en Scilab en la Figura 64:

```
while sum(imskell-maux)<>0
  maux=imskell;
  imborrado=zeros(tam(1),tam(2));
  for i=2:tam(1)-1
    for j=2:tam(2)-1
      contb=0;
      conttran=0;
      if imskell(i,j)==0 then

vectaux=[imskell(i-1,j),imskell(i-1,j+1),imskell(i,j+1),imskell(i+1,j+1),imskell
(i+1,j),imskell(i+1,j-1),imskell(i,j-1),imskell(i-1,j-1)];
        for v=1:8
          if vectaux(1,v)==0 then
            contb=contb+1;
          end
        end
        for v=1:7
          if (vectaux(1,v)==1 & vectaux(1,v+1)==0) | (vectaux(1,v)==0 &
vectaux(1,v+1)==1) then
            conttran=conttran+1;
          end
        end
        if contb>=3 & contb<=6 &
or([imskell(i-1,j),imskell(i,j+1),imskell(i,j-1)])==%t &
or([imskell(i-1,j),imskell(i+1,j),imskell(i,j-1)])==%t & conttran<=2 then
          imborrado(i,j)=1;
        end
      end
    end
  end
  imskell=imskell+imborrado;
end
```

Figura 64: Código en Scilab para primera subiteración del método Zhang Zuen

donde *imskell* es la variable que contiene la imagen que esta siendo esqueletizada.

Una vez esqueletizado el caracter se procedió a programar una rutina, la cual toma cada pixel de la imagen esqueletizada y hace comparaciones. Si este pixel es negro entonces analiza sus ocho pixeles vecinos, si solo uno de sus ocho vecinos es negro, la rutina cuenta este pixel como un punto final, si entre sus vecinos hay más de un pixel negro entonces la rutina cuenta el pixel como un punto de árbol. Según la cantidad de puntos de árbol y puntos finales entonces se le asigna un valor a las componentes del vector de características.

Para finalizar se realizaron cuatro rutinas, correspondientes a las cuatro

características a determinar, estas son: ***trazos verticales, trazos horizontales, trazos diagonales hacia la derecha y trazos horizontales hacia la izquierda***. Todas estas rutinas se programaron bajo el mismo principio y es hacer un recorrido por la imagen según la dirección del trazo a determinar, y sí se encuentra una cantidad mayor a tres pixeles seguidos en la rutina se cuenta como un trazo en la dirección que se esté analizando la característica. Con estas características determinadas se completa el vector de características que va a representar al caracter a la hora del reconocimiento. Estas características se determinan para cada uno de los caracteres que conforman la placa, que luego serán reconocidos en la etapa siguiente.

4.2.1.8 Reconocimiento de Caracteres

El último paso para la Implementación del OCR es el reconocimiento. En este paso se hace uso de la redes neuronales previamente entrenadas. La rutina que se realizó para esta tarea consiste en partiendo de la posición que ocupaba el caracter en la matrícula, se pasa su vector característica por la red neuronal que corresponda, es decir, sí el caracter esta en la primera, segunda, tercera o sexta posición, entonces se hace uso de la red neuronal entrenada para letras. Si está en la cuarta o quinta posición entonces se hace uso de la red neuronal entrenada para números.

Como en este punto ya se tienen los vectores característica de cada caracter almacenados y la posición que ocupa cada caracter, entonces el primer paso en la rutina es hacer uso de la función *ann_FF_run()* del toolbox ANN la cual es de la siguiente manera:|

$$y = ann\_FF\_run(x, N, w) \tag{31}$$

Esta función tienen tres variables de entrada: *x, N, W.* La primera variable *x* es el patrón de entrada a reconocer, la cual es una matriz donde cada columna representa un patrón. La segunda variable *N* es un vector fila de n componentes, donde la primera componente es el número de neuronas de la capa de entrada (que

para el caso de este trabajo son treinta y dos para ambas redes entrenadas), la última componente contiene el número de neuronas de la capa de salida que para el caso de la red de números es diez y para el caso de la red de letras es veinticinco. Las componentes que están entre la primera y la última varían según la necesidad de la red para el reconocimiento, ya que en ellas se almacenan el número de neuronas de las capas ocultas. Por último *w* es una hipermatriz que contiene todos los pesos sinápticos de cada red neuronal, se tienen dos hipermatrices de este tipo uno para la red de los números y otra para la red de las letras. Esta función da como salida *y,* que es un vector con el mismo número de componentes que la capa de salida el cual es el resultado.

Con la salida de la red determinada, la rutina busca para cuál caracter ésta salida da el menor error, entonces se almacena en un vector dicho caracter. Esta rutina se hace para cada caracter y una vez reconocidos todos los caracteres entonces se presenta en pantalla y se guarda en un archivo ".txt".

4.2.2 Implementación de Entrenamiento de Redes Neuronales para obtener Pesos Sinápticos

Para la Implementación del entrenamiento de las redes neuronales, se realizó una rutina que entrena a las redes neuronales para que logren reconocer los caracteres que les sean patrón. Para ello se hizo uso de funciones del toolbox ANN y se definieron algunos parámetros.

Para el entrenamiento de ambas redes se siguieron los del Diagrama 3:

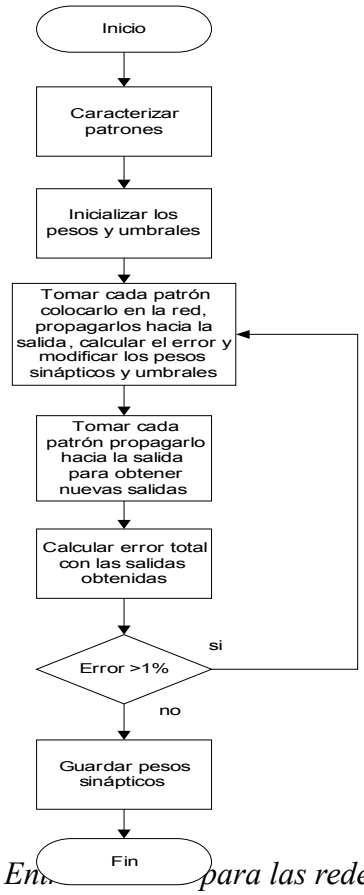


Diagrama 3: En[illegible] para las redes neuronales

Lo primero que hace esta rutina es caracterizar los patrones de para cada red, ya sea de la red neuronal para el reconocimiento de letras o la de números. Para cada tipo de caracter de ambas redes se utilizaron 6 patrones. Lo siguiente es aplicar la rutina de la Sección 4.2.1.7 pero con la diferencia que se repetirá para cada patrón. Los patrones caracterizados son guardados en una matriz.

La inicialización de los pesos se hizo mediante una función del toolbox ANN denominada *ann_FF_init()* la cual es de la siguiente forma:

$$w = ann\_FF\_init(N, r) \tag{32}$$

donde N es el vector que contiene el número de neuronas de cada capa según la red y r es un vector fila de dos componentes que contienen los valores límites entre los que

pueden estar los pesos inicializados. Para este trabajo se inicializaron los pesos entre -0.5 y 0.5.

Para el siguiente paso que es tomar los patrones, colocarlos a la entrada de la red, propagarlos hacia la salida y modificar los pesos sinápticos, se utilizó otra función del toolbox ANN denominada *ann_FF_Std_batch()* la cual aplica retropropagación a partir de las entradas de la función:

$$ws = ann\_FF\_Std\_batch(x, t, N, wa, lp, T) \tag{33}$$

donde las entradas a la función son: *x, t, N, w, lp y T.* La variable *x* contiene la matriz de los vectores caracterizados, *t* es la matriz de salidas deseadas para cada uno de los patrones, *N* es el vector con el número de neuronas para cada capa de la red, *wa* es la matriz que contiene los pesos y umbrales anteriores, lp es la variable que contiene la contante de aprendizaje, y *T* el número de épocas, ya que esta función puede propagar las entradas y modificar los pesos tantas veces como el número que tenga almacenado *T.* Para este trabajo se le asignó a *T* el valor uno ya que se quería ver el comportamiento del error que cometía la red iteración por iteración. La variable de salida ws contiene los nuevos pesos

El siguiente paso, que consiste en obtener ahora las salidas con los nuevos pesos obtenidos, se hace uso de la función *ann_FF_run()* del toolbox ANN la cual es de la siguiente forma:

$$y = ann\_FF\_run(x, N, ws) \tag{34}$$

donde las variables de entrada de esta función fueron descritas anteriormente.

Para el entrenamiento de las redes se seleccionó 4 capas para cada red, es decir, decir una capa de entrada, dos capas ocultas y una capa de salida. Entonces el vector de la variable *N* queda para la red de números como : Nn=[35,20,15,10] y para la red letras como: Nl=[35,42,32,25]. Para terminar la constante para la red de letras se fijó en 0.15. Este valor se seleccionó basado en el efecto que se podía observar durante el entrenamiento, esto es, la constante de aprendizaje tiene un rango de valores entre 0 y 1 (sin incluir el cero), donde se observó que si el valor es cercano a

1 la red entrena más rápido y entre más cercano a cero la red entrena de forma más lenta. Sin embargo cuando una red va siendo entrenada iteración a iteración, si se observan los cambios el error con respecto al cambio de los pesos, esta describe una superficie llena de valles que representan mínimos locales. Si el valor de esta constante es cercano a uno ciertamente entrena más rápido, pero entonces pierda la capacidad de poder salir de estos valles y por tanto, existe la posibilidad que converja a un mínimo local, por otro lado si se selecciona un valor muy cercano a cero la red entrena de forma tan lenta que la convergencia al error deseado, se daría en una gran cantidad de iteraciones, que traducido en tiempo podría decirse que nunca logra la convergencia.

Sabiendo lo anterior, el 0.15 para la red que reconoce los números fue producto de ir probando valores comenzado por los más altos, ya que lo siempre se intenta determinar es el valor más alto posible, porque este garantiza que la red entre lo más rápido posible.

Al igual que para la la red para reconocer los números, para la red de reconocer las letras, se determinó la constante de aprendizaje como 0.045. Si se observa la diferencia entre ambos valores está en que ambas redes tenían una cantidad distinta de clases, teniendo la red para reconocer letras una mayor cantidad de clases a reconocer, por lo que probablemente la trayectoria que describe el error con respecto a los pesos tenía una cantidad mayor de valles, por lo que requería de un valor más pequeño para no converger en un mínimo local.

A continuación la rutina para entrenar la red de números en la Figura 65:

```
14  Nn=[35,20,15,10];
15  lpn=[0.15,0];
16  wn=ann_FF_init(Nn,r);
17  while E>.01
18    wn=ann_FF_Std_batch(xn,tn,Nn,wn,lpn,T);
19    yn=ann_FF_run(xn,Nn,wn);
20    E=ann_sum_of_sqr(yn,tn);
21  end
22  g=1;
23  wn,g]=guarda_pesos(wn);
```

de manera similar se programo la rutina para entrenamiento de la red para reconocer letras.

Del entrenamiento se pudo obtener las gráficas que se observan en las imágenes de la Figura 66 y de la Figura 67, donde se observa como disminuye el error en cada iteración par ambas redes.

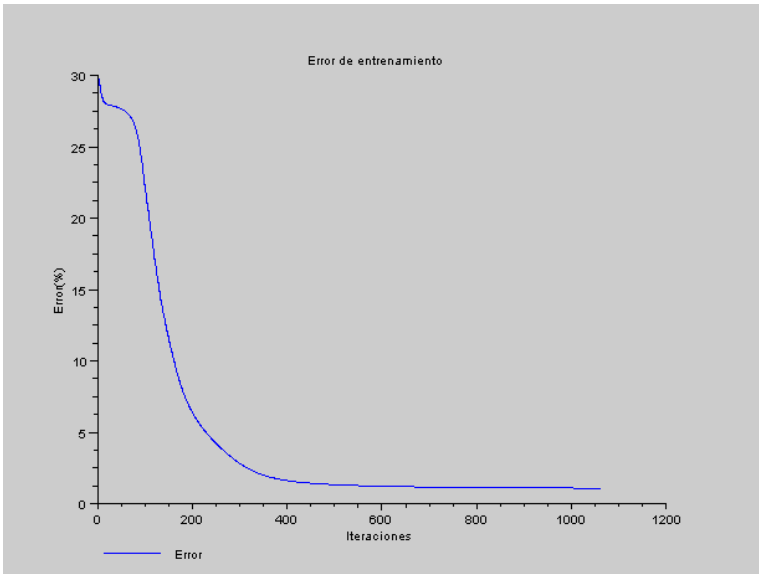
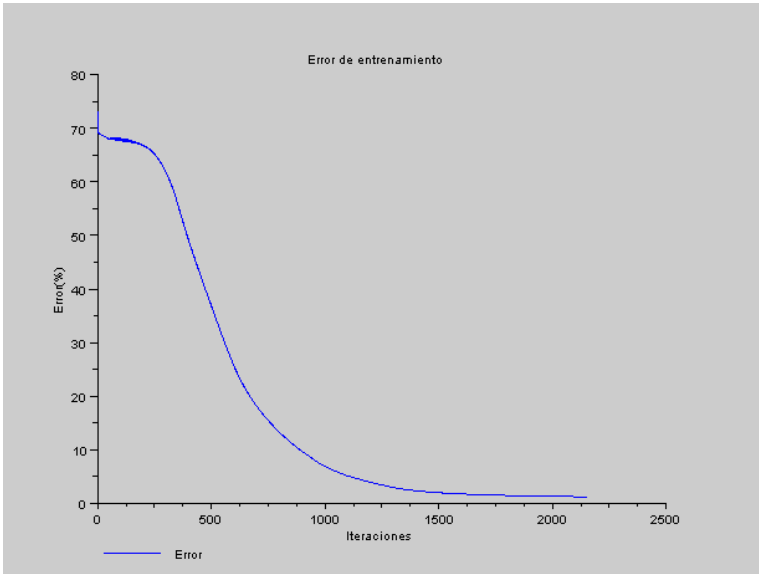


Figura 66: Gráfica del error cometido por la red neuronal para números en cada iteración de su entrenamiento

Se observa que para la red numérica se obtuvo el entrenamiento en un número de 1063 iteraciones, al cumplir un error menor al 1%.



Para la red neuronal alfabética en la Figura 67 el entrenamiento se obtuvo en 2163 iteraciones, al cumplir un error menor al 1%.

4.3 Interfaz de usuario

Adicionalmente se realizó en Scilab para validación visual y con fines demostrativos, una interfaz sencilla de usuario donde se puede observar cada una de las funciones del programa. La interfaz se observa en la Figura 68:

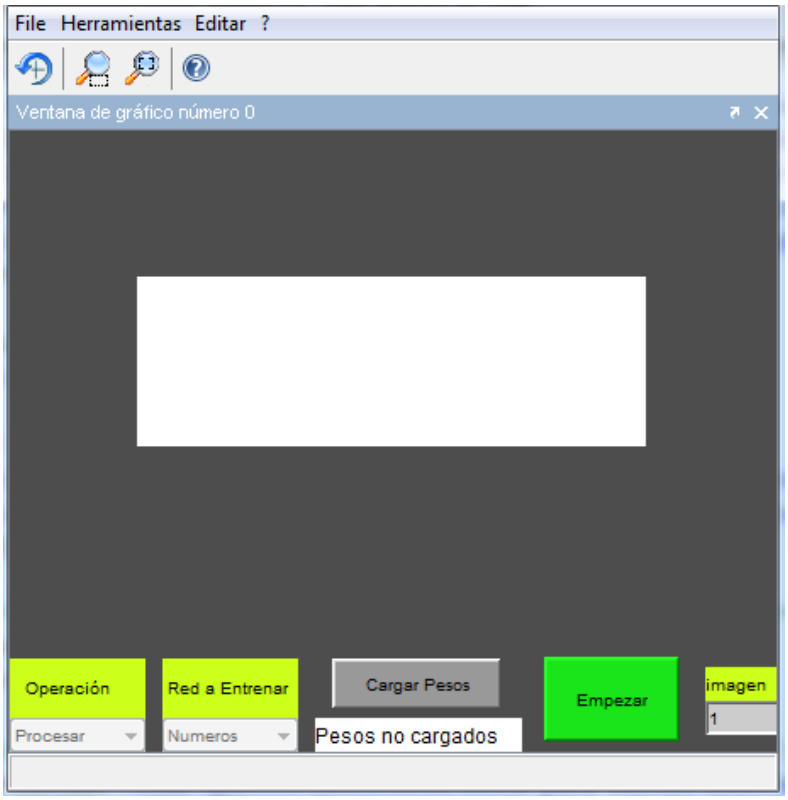


Figura 68: Interfaz de usuario para manejar el programa

En esta interfaz se puede observar la matrícula que arroje como resultado el procesar una imagen, así como el cambio de imagen a procesar, la selección entre las operaciones se requieran realizar ya sea para el procesamiento o entrenamiento de las imágenes.

CAPÍTULO V: RESULTADOS Y ANÁLISIS

En este capítulo se muestran los resultados finales de este trabajo para las pruebas que se hicieron al modelo de reconocimiento de matrículas desarrollado, al cual se le introdujeron una seria de imágenes, y se observó el comportamiento de cada etapa programada.

Para las pruebas finales se tuvo un número de 63 imágenes como muestra, con las cuales el programa arrojó los siguientes resultados por etapas que se observan en la Tabla 5:

Etapa	Entradas recibidas	Salidas exitosas	Efectividad(%)
Conversión a escala de grises	63	63	100
Extracción de área útil	63	62	98,41
Extracción de la matrícula	62	49	79,03
Binarización	49	49	100
Segmentación de caracteres	49	49	100
Total de etapas anteriores	63	49	77,78

Tabla 5: Resultados por etapas del procesamiento de las imágenes de prueba hasta la extracción de la matrícula

Para los resultados de la Tabla 5 se puede notar que para las etapas conversión escala, segmentación y binarización se tuvo total efectividad, lo cual deja ver su correcto funcionamiento, que en el caso de la conversión a escala de grises era de predecirse ya que esta etapa constaba solo de una función del toolbox SIVP de

Scilab. Para los casos de binarización y segmentación, los resultados dejan ver la acertada selección y correcta programación, que no dejan lugar a errores.

Para la extracción de área útil se consigue un fallo producto de que la imagen de la matrícula quedó por debajo de un cuarto de la imagen, por lo que haciendo estudios a la rutina programada se encontró que si las placas quedan muy por debajo de un cuarto de la imagen queda fuera del área útil, y por tanto no pueden ser encontradas.

Para la extracción de matrícula se consigue el resultado más bajo de las etapas analizadas. Analizando las imágenes de prueba se encontró que las dimensiones de las matrículas eran variadas, y rangos de tamaño considerablemente amplios. Analizando la rutina programada se observó que aun y cuando se trabajan en rangos amplios, pues aun se tienen imágenes con matrículas por fuera de los rangos definidos, lo cual explica el resultado obtenido para esta etapa.

Como resultado final el 77,78% (49 imágenes) de las imágenes que se introdujeron al programa, pudo ser procesado completamente hasta la etapa de caracterización, por las múltiples razones expuestas anteriormente.

Cabe destacar que los resultados anteriores se obtuvieron en base a las etapas alcanzadas con éxito de cada imagen, razón por la cual hay diferencias en la cantidad de entradas en cada etapa.

En la etapa de reconocimiento una vez caracterizados los caracteres y procesados con las redes neuronales se obtuvieron los resultados que se observan en la Tabla 6:

	Caracteres procesados	Caracteres acertados	Efectividad(%)
Últimos tres caracteres	147	115	78,23
Primeros tres caracteres	147	116	78,91
Total de caracteres	294	231	78,57

Tabla 6: Resultados del procesamiento de los caracteres con las redes neuronales.

Para los efectos de este trabajo solo se quiere el resultado del reconocimiento de los últimos tres caracteres de la matrícula. Sin embargo el trabajo se hizo en base a reconocer los seis caracteres del tipo de matrícula con que se trabajo.

Según los resultados en la Tabla 6 se tiene un error alrededor de 22% en general, y haciendo una revisión a los fallos, se encontraron confusiones tales como:

- Confusión de la letra “O” con la “D” y viceversa.
- Confusión de la letra “M” con la “H”.
- Confusión de la letra “A” con la “R”.
- Confusión de la letra “X” con la “Y”.
- Confusión de la letra “G” con la “C”.
- Confusión de la letra “G” con la “S” y viceversa.
- Confusión de la letra “J” con la “L” y viceversa.
- Confusión de la letra “T” con la “L”.
- Confusión de la letra “M” con la “W”.
- Confusión del número “1” con el “7” y viceversa.
- Confusión del número “3” con el “5”.

Observando las características de los caracteres en los que sucedieron las confusiones de la red, se encuentra que ante la caracterización propuesta estos poseen características comunes, que hacen que sus vectores característica en ciertas ocasiones puedan a llegar a ser muy similares, y por tanto esto hace que la red equivoque el reconocimiento. Otra de las causas que también podrían haber ocasionado estas confusiones, el hecho de estar trabajando con matrículas de distintas resoluciones, por lo que al final de la rutina de caracterización para estandarizar se toma cada caracter segmentado y se le cambia su tamaño original a un tamaño de 32x16 pixels, lo cual podría hacer que en el caso del caracter estar lejos de estas dimensiones se afecten algunas de sus características. Una última razón podría ser que las redes requieren de entrenamiento con una cantidad mayor de patrones, para así poder aumentar su capacidad de reconocer otros patrones, es decir, aumentar su

capacidad de generalización.

En cuanto a los tiempos de procesamiento de cada imagen, desde que es introducida al programa hasta que son reconocidos los caracteres, se obtuvieron los resultados que se observan en la Tabla 7:

Tiempo mínimo (seg)	Tiempo promedio(seg)	Tiempo máximo(seg)
3,04	4,72	8,31

Tabla 7: Resultados de los tiempos de procesamiento

Los tiempos de entrenamiento también representó un inconveniente, por estar alrededor de horas, para la red de letras, y para red de números, los tiempos fueron más cortos pero aun fueron decenas de minutos.

De la Tabla 7 se extrae que los tiempos de procesamiento resultan ser relativamente cortos para una inmediata observación de resultados, pero resultan largos para la aplicación en la que se requiere este modelo computacional. La razón erradica en que principalmente el modelo computacional fue programado en Scilab.

Scilab posee un código de alto nivel que corre dentro del mismo entorno de desarrollo, es decir es un lenguaje interpretado, lo que significa que el código fuente es interpretado en forma de texto comando por comando cada vez que se ejecutan, por lo que entonces resulta lento, cosa contraria a un lenguaje compilado el cual convierte el código fuente a código de máquina el cual puede ser ejecutado directamente en el computador sin necesidad de ser interpretado dentro de un entorno de desarrollo. También puede notarse que existe un diferencia considerable entre el tiempo máximo y el tiempo mínimo, esto es por las matrículas de diversas dimensiones ya que las matrículas de menos tamaño requieren de mayor cantidad de iteraciones para su búsqueda.

Adicionalmente a los objetivos de este trabajo se había considerado también la posibilidad de procesar otro tipo de matrícula, es decir, el modelo más moderno de matrículas para automóviles venezolanos. Para esto solo se contó con una única imagen , la cual se introdujo al programa para su procesamiento, hasta solo la etapa

de segmentación, ya que no se contaba con una red entrenada ,para el reconocimiento de los caracteres segmentados para este tipo de matrícula. La imagen de prueba es la mostrada en la Figura 69

Al procesarla esta imagen el resultado es el mostrado en la Figura 70:

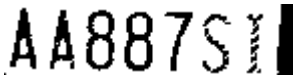


Figura 70: Imagen resultado de procesar matrícula moderna

y al pasar por segmentación se tiene el resultado siguiente que se muestra en la Figura 71:

por tanto se observa que también es posible llegar a segmentar matrículas modernas con este programa. En cuanto al reconocimiento, se tiene que debido a la caracterización utilizada es posible reconocer los caracteres de las matriculas modernas, solo habría que aplicarle algunos filtros morfológicos a los caracteres, de

tal forma de hacerlos totalmente conexos, ya que por el diseño de las matrículas modernas, aparecen lineas diagonales que hacen a los caracteres discontinuos o no conexos.

CONCLUSIONES

Se logró analizar y seleccionar, técnicas y algoritmos para el procesamiento de imágenes y extracción de caracteres de una matrícula vehicular, con lo cual se logró un diseño de las etapas que componen un OCR lo que implica también el diseño de una red neuronal para reconocer los caracteres provenientes de las matrículas. Con el diseño anterior, haciendo uso de una herramienta computacional de software libre y con ayuda de librerías (toolbox) de esta herramienta computacional, se implementó un modelo que permite en un alto porcentaje de veracidad reconocer no sólo los tres últimos caracteres si no los tres primeros, es decir, permite el reconocimiento de todos los dígitos del modelo antiguo de matrícula vehicular venezolana. Además, el modelo permite procesar matrículas no solo antiguas si no tiene la posibilidad de procesar las matrículas venezolanas más modernas pero solo hasta la segmentación de caracteres, debido a que no se cuenta con una red entrenada para el reconocimiento. Por lo que se concluye entonces que los resultados obtenidos en este trabajo especial de grado fueron satisfactorios.

El tamaño de las matrículas fue la razón principal para los fallos en las etapa de extracción de la matrícula, por lo que las dimensiones de la placa son una variable a tomar en cuenta para su eliminación pero de forma externa al programa ya que se hace dificultoso trabajar con una rango amplio de tamaños de matrículas.

Las redes neuronales para el reconocimiento de los caracteres de la matrícula tuvieron un buen entrenamiento, por tanto dentro de sus limitantes se obtuvo un buen desempeño de las mismas, la mayor limitante o razón para sus fallos fue la caracterización, debido a que la misma no fue lo suficientemente completa para discriminar las características de algunos caracteres, causando así que la redes fallaran en múltiples ocasiones. Sin embargo la caracterización cumplió con su objetivo, de reducir la información procesada. Es notoria la necesidad de definir algunas otras características que ayuden a reducir estos fallos.

La herramienta computacional Scilab es una herramienta poderosa para la investigación y desarrollo, pero su entorno de trabajo lleva a un lenguaje de programación interpretado, por lo que el código fuente corre dentro de su entorno. Dadas estas razones puede ser utilizado pero no es recomendable para este tipo de aplicaciones donde se requiere rapidez en el procesamiento. La mejor forma de utilizarlo es aprovechando la bondad que tiene de poder trabajar con funciones compiladas en otros lenguajes como c++, JAVA, etc, y trabajar la aplicación bajo esta forma de programación que ofrece Scilab.

Viendo los distintos escenarios que se pueden presentar al reconocer una matrícula proveniente de un automóvil que esta en movimiento, es de pensarse que para utilizar un programa como este, no podemos conformarnos con obtener el resultado a partir del procesamiento de una sola imagen del automóvil al cual se le quiere reconocer su matrícula, el resultado final debe provenir del procesamiento de una cantidad mayor a una imagen para así segurar una validación automática.

El tiempo fue ajustado para el desarrollo de este trabajo, ya que requería de una gran cantidad de tiempo de investigación debido a la escases de información sobre los tópicos de cada tema que envuelve al mismo e información acerca de antecedentes de trabajos similares en el país, por lo que el desarrollo de este tuvo una gran dificultad, aunado a esto, algunas de las implementaciones tomaban mucho tiempo de pruebas, como las que tienen que ver con la redes neuronales, ya que estas requerían tiempo prolongado de entrenamiento antes de poder ser probadas, así también los métodos como los de localización de matrícula en los cuales se requería hacer pruebas para poder fijar las dimensiones de las matrículas y elementos estructurantes con los que se trabajarían.

RECOMENDACIONES

Se recomienda estudiar y realizar las siguientes mejoras para la programación del modelo computacional:

1. El método utilizado para la etapa de extracción de la matrícula es suficientemente poderoso como para considerar el estudio de la extracción directa de la matrícula sin necesidad de la etapa de extracción de área útil. Por tanto se recomienda estudiar y probar la extracción de matrícula sin extracción de área útil.
2. Se recomienda agregarle otras características para ser extraídas a los caracteres en la etapa de caracterización, de tal forma que se logre mejorar su desempeño. Alguna característica sería por ejemplo, considerar la imagen del caracter en dos mitades a lo vertical, es decir, una mitad derecha y una mitad izquierda. Sumar los pixeles de cada mitad y comparar en cual mitad hay mayor concentración de pixeles negros y con esto generar condiciones para agregar algunas componentes al vector de características. Igualmente considerar la imagen del caracter en dos mitades pero a lo horizontal, es decir, una mitad superior y una inferior, y así mismo comparar donde hay mayor concentración de pixeles negros.
3. Se recomienda luego de las mejoras anteriores realizar entrenamiento a cada red neuronal pero con una cantidad mayor de patrones.
4. Se recomienda agregarle algún algoritmo de validación para aceptar automáticamente el acierto o no del caracter. Un algoritmo de validación podría sacar el error cuadrático comparándolo con la salida que dieron los patrones de entrenamiento. Además procesar al menos una cantidad de 4 imágenes del automóvil del que se requiera extraer el serial de la matrícula, y con el resultado de cada una de estas 4 matrículas se hace una estadística por

posición, donde se da por aceptado absolutamente aquel tipo de caracter que se haya repetido mayor cantidad de veces para su propia posición en la matrícula.

5. Se recomienda agregarle al programa alguna seguridad para considerar casos en los que no se logre encontrar exitosamente la matrícula o en ausencia de la misma.

Se recomienda migrar el código fuente a algún lenguaje en software, que no sea interpretado si no compilado como lo es JAVA o bien trabajar la aplicación con funciones hechas en c++. JAVA u otro lenguaje de programación compilado mediante Scilab, ya que trabajar directamente con la programación en Scilab hace la aplicación lenta.

A la hora de incluir este modelo en un sistema de detección vehicular, se recomienda tratar de recibir las imágenes con las dimensiones de las matrículas lo más estables posible, para que tenga éxito el procesamiento de las mismas.

REFERENCIAS BIBLIOGRÁFICAS

[1] Cadore, Joyner. Desarrollo de un Analizador de Imágenes Basado en el Reconocimiento Óptico de Caracteres trabajo de grado Universidad Simón Bolívar, 2009.

[2] Imagen. *Wikipedia*. <http://es.wikipedia.org/wiki/Imagen> [Consulta: marzo 2011].

[3] Imagen Digital. *Wikipedia,*. <http://es.wikipedia.org/wiki/Imagen_digital> [Consulta: marzo 2011] .

[4] Gráfico vectorial. *Wikipedia*. <http://es.wikipedia.org/wiki/Gr%C3%A1fico_vectorial> [Consulta: marzo 2011] .

[5] Tipos de imágenes: vectoriales, *digitalfotored* <http://www.digitalfotored.com/imagendigital/tiposimagenes.htm>[Consulta: marzo 2011].

[6] Definición de Gráfico rasterizado . *Diccionario de informática.* <http://www.alegsa.com.ar/Dic/grafico%20rasterizado.php> [Consulta: marzo 2011]

[7]Jenaro Carlos Paz Gutiérrez, Generación de Imágenes para Web con GDI+, Ciudad Juárez, Chih. 2007.
<http://www2.uacj.mx/Noticias/Publicaciones/generacion_imagenes.htm>

[8]Teoría del color. *Wikipedia, La enciclopedia libre*. <http://es.wikipedia.org/wiki/Teor%C3%ADa_del_color#Espacios_de_colores> [Consulta: marzo 2011] .

[9] John C. Russ. The Image Processing Handbook third edition, Boca Raton, Florida, USA: CRC Press, inc. , 1998.

[10]Procesamiento de Imágenes, *ensayo no publicado,* España: Universidad de Madrid, <http://www.slidefinder.net/p/procesamiento_im%C3%A1genes_t%C3%A9cnicas_realce_im%C3%A1genes/7760388> [Consulta: noviembre 2011].

[11]Tema 5: Segmentación de imágenes, *ensayo no publicado,* España: Universidad de Valencia*,* <http://miron.disca.upv.es/vxc/Documentos/Microsoft%20PowerPoint%20-%20vxc-5_2a.pdf>[Consulta: diciembre 2010].

[12]Tema 4: Visión de Nivel Medio 2: Segmentación y Umbralización, *sliderfinder,* <http://www.slidefinder.net/C/Cap4vision/Cap4vision/20854964>[Consulta: febrero 2011].

[13] Esqueletos, *ensayo no publicado,* España: Universidad de Deusto*,* fecha de <http://www.ayc.unavarra.es/orduna/LIA/Tutoriales/esqueletos.pdf>[Consulta: marzo 2011].

[14] Coprocesador para la Esqueletización de Huellas Dactilares, ensayo no publicado, España: Universidad Politécnica de Cataluña, <http://petrus.upc.es/emsy/jcra05.pdf>[Consulta: marzo 2011].

[15] Actualidad TIC Revista del Instituto Tecnológico de Informática, N° 2 (2003,p.8) http://www.iti.es/media/about/docs/tic/02/2003-11.pdf

[16]Pedro Isasi Viñuela, Redes Neuronales Artificiales, Ribera del Loira, Madrid, España: Pearson Education s.a. , 1998.

[17]Capítulo 2. Conceptos Básicos sobre RNA <http://softwarelibre.unsa.edu.ar/docs/descarga/2003/curso/htmls/redes_neuronales/c35.html>[Consulta: marzo 2011].

[18]Scilab, <http://www.scilab.org/products/scilab/environment>[Consulta: abril 2011].

[19]Tratamiento Digital de Imágenes; definiciones. *Curso Básico de Teledetección.* <http://www.innovanet.com.ar/gis/TELEDETE/TELEDETE/tradiimg.htm#m34>[Consulta: noviembre 2010] .

[20] Técnicas de Filtrado, *ensayo no publicado,*España: Universidad de Murcia. <http://www.um.es/geograf/sigmur/teledet/tema06.pdf>[Consulta: noviembre 2010].

[21]Filtrado Lineal, *ensayo no publicado de la Universidad del País Vasco*, <http://www.sc.ehu.es/ccwgrrom/transparencias/pdf-vision-1-transparencias/capítulo-5.pdf>[Consulta: noviembre 2010] .

[22] Modelos, Control y Sistemas De Visión, L*os Filtros Espaciales y Frecuenciales,* <http://omarsanchez.net/filtroespa.aspx>[Consulta: marzo 2011].

[23]Técnica de Preprocesado, *ensayo no publicado,* España:Universidad de Madrid*,* <http://www.elai.upm.es:8009/>[Consulta: noviembre 2010]

[24]Francisco Gabriel Ortiz Zamora, *Procesamiento Morfológico de Imágenes. Aplicación a la Reconstrucción Geodésica,* (Tesis Doctoral), Alicante: Universidad de Alicante.2002. p. 58-63.
http://www.lluisvives.com/servlet/SirveObras/57915842105571617400080/008591_1.pdf

[25]Rafael Alberto Gonzáles Gonzáles, *Algoritmo basado en Wavelets Aplicado a la Detección de Incendios Forestales.(*Tesis*),* Puebla. México: Universidad de las Américas Puebla, 2010 ,p. 15-17.
http://catarina.udlap.mx/u_dl_a/tales/documentos/mel/gonzalez_g_ra/index.html

[26] Algoritmos de Interpolación de Imágenes*, ensayo no publicado, España: Universidad Técnica Federico Santa María, <http://www.alumnos.inf.utfsm.cl/~vpena/ramos/ili286/presentacionCC2.pdf>[Consulta: marzo 2011]*.

[27] Transformaciones Geométricas en Imágenes, *Informática.*
http://vehac.blogspot.com/2009/06/transformaciones-geometricas-en.html

[28] Ondrej Martinsky, *Algorithmic and Mathematical Principles of Automatic Number Plate Recognition* Systems, (Tesis), Brno: Brno University of Technology 2007. p. 13-17.
http://www.fit.vutbr.cz/research/view_product.php?file=%2Fproduct%2F30%2Fanpr.pdf&id=30

ANEXOS

ANEXO A: FILTRADO ESPACIAL;TIPOS DE FILTRADOS ESPACIALES

A continuación se definen los tipos de filtros de filtros espaciales.

a) Filtros Lineales en el Dominio del Espacio

Son los que utilizan un kernel o máscara de convolución para llevar a cabo la operación de filtrado. La operación de filtrado lineal se define por una matriz de convolución y la matriz de convolución define la forma como el pixel especificado se ve afectada por los píxeles vecinos en el proceso. En la convolución por lo general se utilizan máscaras de 3x3 donde la misma afecta al pixel central y a los ocho vecinos que rodean a dicho pixel. En el proceso de convolución el pixel representado por la celda "y" de la imagen destino que se muestra en la Ilustración 1, se ve afectada por los píxeles x0 ... x8 de la imagen origen en la Ilustración 1 de acuerdo con la fórmula siguiente[7], [19]:

$$y = x_0 \cdot m_0 + x_1 \cdot m_1 + x_2 \cdot m_2 + x_3 \cdot m_3 + x_4 \cdot m_4 + x_5 \cdot m_5 + x_6 \cdot m_6 + x_7 \cdot m_7 + x_8 \cdot m_8 \qquad (1)$$

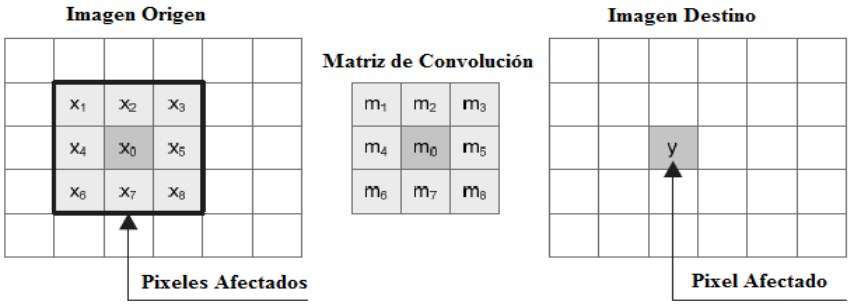


Ilustración 1: Convolución de una Imagen con máscara 3x3 [7, p.14]

Los tipos de filtros lineales más resaltantes que se pueden encontrar son los siguientes:

- **Filtros Pasa Bajo** : Su objetivo es suavizar la imagen, son útiles cuando se supone que la imagen tiene gran cantidad de ruido y se quiere eliminarlo. También pueden utilizarse para resaltar la información correspondiente a una determinada escala (tamaño de la matriz de filtrado) [20]. La suma de los coeficientes de estos filtros suman 1 para evitar ganancias indeseadas [21]. Entre estos filtros están el filtro promedio y el filtro promedio ponderado. En la Ilustración 2 se observa el efecto que tiene este filtro sobre una imagen en escala de grises.

(a) *(b)*

Ilustración 2: Efecto de filtro de promedio sobre imagen. (a)original con ruido sal pimienta, (b)promedio.

- **Filtros Pasa Alto**: Intensifica los detalles, bordes y cambios de alta frecuencia, mientras que atenúa las zonas de tonalidad uniforme. Esto permite una mejor identificación posterior de los objetos que se encuentren en la imagen, por tanto su objetivo es resaltar las zonas de mayor variabilidad eliminando lo que sería la componente media, precisamente la que detectan los filtros de paso bajo. Por otra parte la respuesta de cada pixel está contaminada por la de los pixeles vecinos ya

que, considerando la superficie terrestre como lambertiana, la radiación reflejada por un pixel se reparte hacia los pixeles vecinos. Los filtros de paso alto consiguen también eliminar en parte esta contaminación [20]. La suma de los coeficientes de estos filtros suman cero. De estos filtros se tienen el filtro de sustracción de la media y el filtro laplaciano.

En la Ilustración 3 se observa el efecto de aplicarle el filtro laplaciano a una imagen en escala de grises, donde la imagen resultante tiene sus bordes resaltados.

(a) *(b)*

Ilustración 3: Efecto de filtro de laplaciano sobre imagen. (a)original , (b)laplaciano.

- **Filtros Direccionales**: Son un caso especial de los filtros pasa alto. Empleados para destacar y resaltar con mayor precisión los bordes que se localizan en una dirección determinada de la imagen. Estos filtros en general calculan las derivadas parciales aproximadas de la imagen, detectando cambios de intensidad existentes entre píxeles contiguos[21]. Las derivadas parciales en las que se basan estos tipos de filtros provienen del cálculo del gradiente de la imagen. El vector gradiente siempre apunta en la dirección de la máxima variación de la imagen en el punto (x,y). En

la detección de bordes es muy importante la magnitud de este vector.

De estos filtros se tienen:

- ✔ Filtro de Roberts: Se obtiene a partir de diferenciales discretos, dando como resultado dos máscaras en las direcciones horizontal y vertical que se muestran a continuación:

$$\begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix}$$

  Una extensión natural seria tratar de capturar las variaciones en los sentidos diagonales y se logra con las máscaras que se muestran a continuación:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

- ✔ Filtro de Prewitt : Sus máscaras surgen, al igual que las de Roberts, de la discretización del operador gradiente pero extendida a una máscara 3x3, siendo las máscaras de este filtro vertical, horizontal y diagonales respectivamente las siguientes :

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$$

- ✔ Filtro de Sobel: El operador Prewitt resulta ser muy sensible con el ruido. Al reforzar el valor del pixel central se reduce la influencia del ruido en la detección de discontinuidades, entonces se le coloca el valor 2 a los píxeles centrales que representan las variaciones en la vertical, horizontal y la diagonal. A las máscaras resultantes de aplicar tal variación se les denomina *máscaras de Sobel*. A continuación se observan las máscaras de este filtro[22]:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

- ✔ Filtro de Frei-Chen: Mientras que Prewitt detecta mejor los bordes verticales, Sobel mejora su localización en los bordes horizontales. El operador isotrópico o de Frei-Chen intenta llegar a un compromiso entre ambos [23], teniendo las máscaras tal como siguen:

$$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -\sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & \sqrt{2} \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & -\sqrt{2} \\ 1 & 0 & -1 \\ \sqrt{2} & 1 & 0 \end{bmatrix}$$

Los operadores de *Sobel* y de *Frei-Chen* tienen la ventaja de que proporcionan un suavizado además del efecto de derivación. Ya que la derivación acentúa el ruido, el efecto de suavizado es particularmente interesante, puesto que elimina parte del ruido.

En la Ilustración 4 se muestra los efectos de aplicar distintos filtros direccionales a una imagen en escala de grises, observándose como todos estos resaltan los bordes de la imagen y atenúan los demás

detalles.

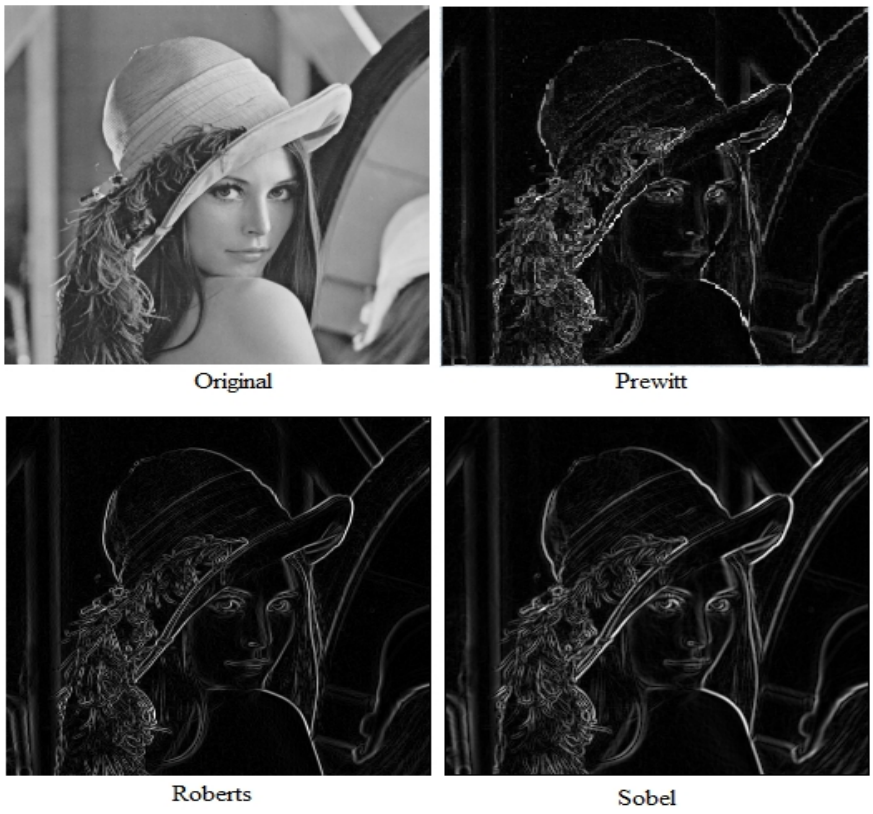


Ilustración 4: Efectos de filtros direccionales

b) Filtros No Lineales en el Dominio Espacial

Son aquellos que utilizan operaciones basadas en procesar la forma y estructura de la imagen digital. Estos al igual que los filtros lineales poseen un kernel, pero no es un kernel de convolución si no uno que representa un operador con características propias de los filtros que se describirán más adelante.

Entre los filtros más comunes de es tipo están los siguientes:

- Filtro de Mediana: Tiene la ventaja de que el valor final del pixel es un valor real presente en la imagen y no un promedio, de este modo se reduce el efecto borroso que tienen las imágenes que han sido procesadas por un filtro de media. Además de esto el filtro de la mediana es menos sensible a valores extremos y es útil para eliminar ruido impulsivo. Su kernel opera de manera similar a los de los filtros lineales, con la diferencia que se basa en el cálculo de la mediana de los pixeles de la máscara, cuyo tamaño es el de una matriz 3x3 [20].

  En la Ilustración 5 se observa como el filtro de mediana suaviza el fuerte ruido de la imagen.

(a) *(b)*

Ilustración 5: Efecto de filtro mediana sobre imagen. (a)original con ruido sal y pimienta, (b)mediana.

- Filtros Morfológicos: Son métodos no lineales de procesar imágenes digitales basándose en la forma. Su principal objetivo es la cuantificación de estructuras geométricas. Aquí los filtros también vienen definidos por su kernel, pero no es un kernel de convolución sino un elemento estructurante[19]. Las aplicaciones básicas de estos filtros son la

atenuación de ruido y la extracción selectiva de objetos.

Los elementos estructurantes como se había mencionado antes, son en morfología matemática lo que las máscaras (o kernel) de convolución son en los filtros lineales. Las formas más usadas de éstos elementos estructurantes son las que se muestran en la Ilustración 6:

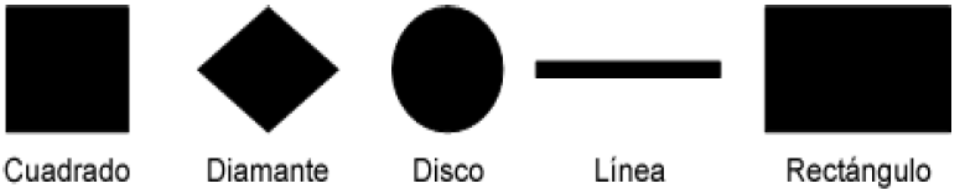


Ilustración 6: Formas típicas de elementos estructurantes[16 p. 15]

De estos filtros u operadores morfológicos podemos encontrar:

- ✔ Erosión: La erosión de una imagen *f* mediante un elemento estructurante *Y* se define como el mínimo de las traslaciones de los elementos de *f* (definidas por el elemento estructurante) pertenecientes a *Y*, o lo que es lo mismo se define como el valor mínimo entre los elementos de la imagen que están dentro de la vecindad definida por el elemento estructurante [24], [25].

  En imágenes binarias el efecto de este filtro es la desaparición de elementos de menor tamaño que el elemento estructurante, y en imágenes en escala de grises el resultado es una imagen más obscura, puesto que el objetivo de la erosión es minimizar los valores de la imagen [24].

La Ilustración 7 muestra como la erosión en imágenes en escala de grises oscurece la imagen original.

Ilustración 7: Efecto de erosión sobre imagen. (a)original , (b)Erosión.

- ✔ Dilatación: La dilatación es el dual de la erosión. La erosión de una imagen f mediante un elemento estructurante Y se define como el máximo de las traslaciones de los elementos de f (definidas por el elemento estructurante) pertenecientes a Y, o lo que es lo mismo se define como el valor máximo entre los elementos de la imagen que están dentro de la vecindad definida por el elemento estructurante [24], [25].

  En imágenes binarias el efecto de este filtro es el realce de elementos de mayor tamaño que el elemento estructurante, en imágenes en escala de grises el resultado es una imagen más clara, puesto que el objetivo de la erosión es maximizar los valores de la imagen [24].

  Para los filtros anteriores es importante el lugar donde se defina el origen, ya que este define la orientación de las traslaciones.

La Ilustración 8 muestra como la dilatación en imágenes en escala de grises aclara la imagen original, resaltando los sitios blancos.

Ilustración 8: Efecto de dilatación sobre imagen. (a)original , (b)dilatación.

- ✔ Apertura (Opening): Generalmente las operaciones de dilatación y erosión no admiten inversa, por tanto no hay manera de determinar el origen de la imagen a la cual le ha sido aplicada cualquiera de dichas operaciones. Debido a su dualidad es posible intentar recuperar la imagen original, si una vez habiéndole efectuado alguna operación morfológica como la de erosión por tomar un ejemplo, se le aplica su dual que es una dilatación [24], [25].

  Se define apertura u opening de una imagen mediante un elemento estructurante como la erosión, seguida por la dilatación de la misma. El efecto de esta operación en imágenes binarias da como resultado que no se puede recuperar totalmente la imagen original, ya que por el efecto de la erosión los elementos menores al elemento estructurante son eliminados, y en el caso de las imágenes en escala de grises elimina las formas claras más pequeñas que el elemento estructurante, pero si se puede tener una imagen muy similar tal como se observa en

la Ilustración 9[15].

Ilustración 9: Efecto de apertura sobre imagen. (a)original , (b)apertura.

- ✔ Cierre (Closing): Se define cierre o closing de una imagen mediante un elemento estructurante como la dilatación, seguida por la erosión de la misma. El efecto de esta operación en imágenes binarias da como resultado que no se puede recuperar totalmente la imagen original, ya que por el efecto de la dilatación los elementos mayores al elemento estructurante son agrandados causando que se formen estructuras que luego la erosión no puede deshacer, y en el caso de las imágenes en escala de grises elimina las formas obscuras más pequeñas que el elemento estructurante quedan sensiblemente atenuadas, tal como se observa en la Ilustración 10.

  La aplicación de los filtros de apertura y cierre son invariantes ante la ubicación del origen en el elemento estructurante [24].

Ilustración 10: Efecto de cierre sobre imagen. (a)original , (b)Cierre.

En la Ilustración 11 se presenta el efecto de los filtros morfológicos anteriores ante una imagen binaria:

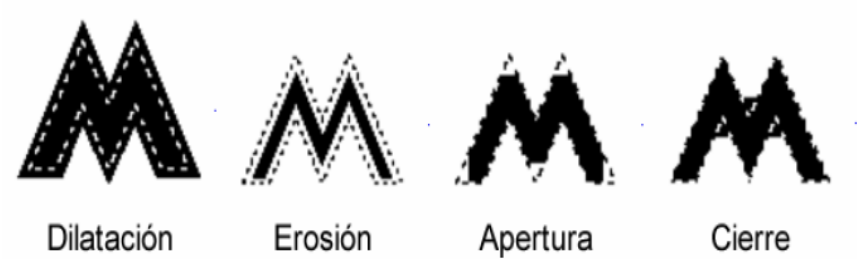


Ilustración 11: Efectos de filtros morfológicos, dilatación, erosión, apertura y cierre ante una imagen binaria [25, p. 16]

Las lineas discontinuas representan a la imagen original.

- ✔ Gradiente Morfológico o Gradiente de Beucher: se define como la diferencia entre la dilatación y la erosión de una imagen o bien llamado gradiente simétrico, la dilatación y la imagen original llamado gradiente por dilatación o una diferencia entre la erosión y la imagen original llamado gradiente por erosión.

  En escala de grises el uso de este tipo de filtro intensifica la presencia

de bordes en la imagen tal como muestra la Ilustración 12 [24].

En la Ilustración 12 se observa el efecto de aplicar este gradiente morfológico en una imagen.

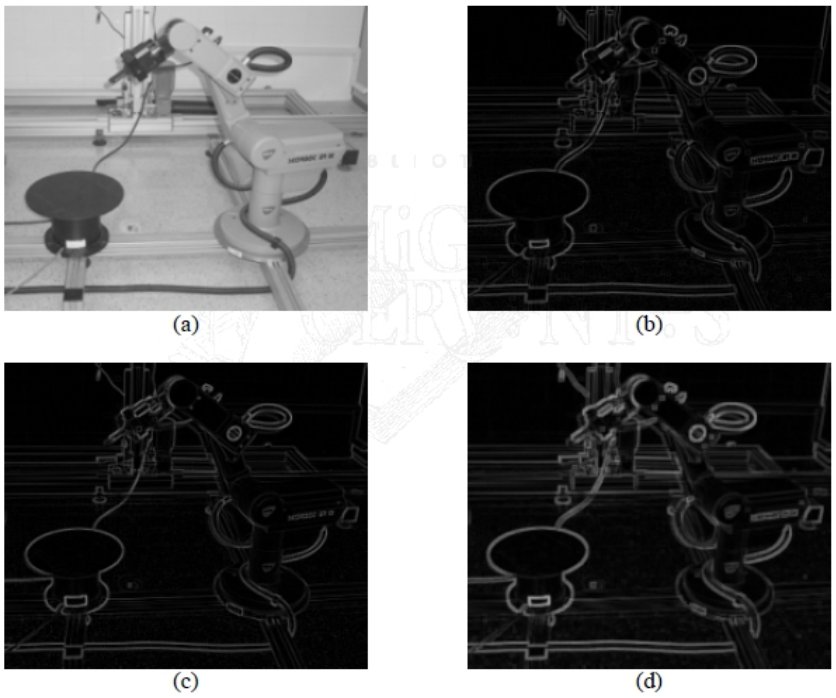


Ilustración 12: Muestra de los resultados al aplicar gradiente morfológico. (a)imagen original,(b)imagen gradiente por erosión,(c) gradiente por dilatación,(d)imagen con gradiente simétrico.[24, p. 56]

- ✔ Top-Hat: Este filtro es el resultado de dos posibles operaciones, una de estas es la diferencia entre la apertura de una imagen y ella misma, a lo cual se le llama top-hat por apertura, y la otra operación es la diferencia del cierre de una imagen y ella misma, a lo cual se le llama top-hat por cierre. La utilidad de estos filtros están de la siguiente manera: en el caso del top-hat por cierre destaca los elementos oscuros

que fueron eliminados en la operación de cierre, y el top-hat por apertura destaca los elementos claros que fueron eliminados en la operación de apertura. Estos efectos se pueden en la Ilustración 13 [24].

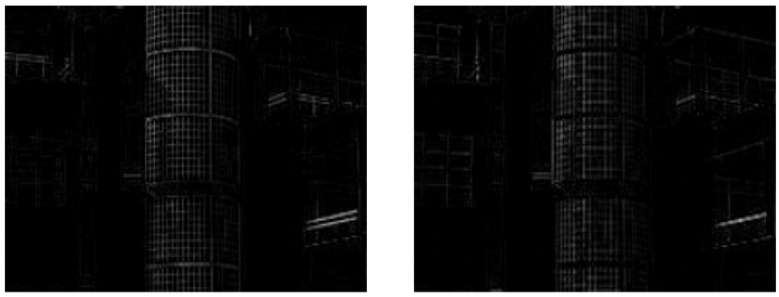
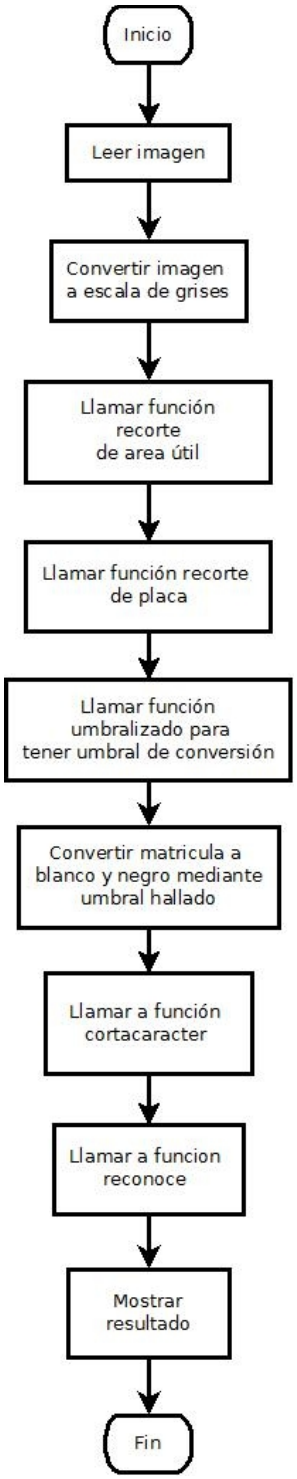


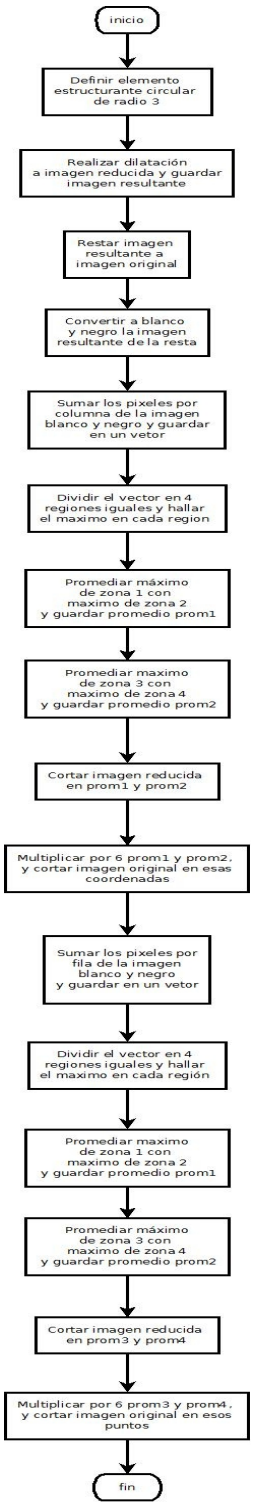
Ilustración 13: Efecto de top-hat sobre imagen. (a)top-hat por apertura, (b)top-hat por cierre.

ANEXO B: DIAGRAMAS DE BLOQUE DEL PROGRAMA.

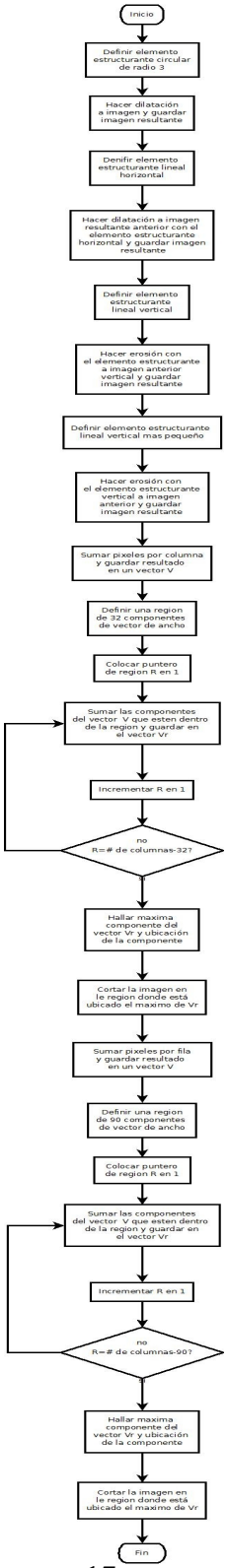
1)Función Principal



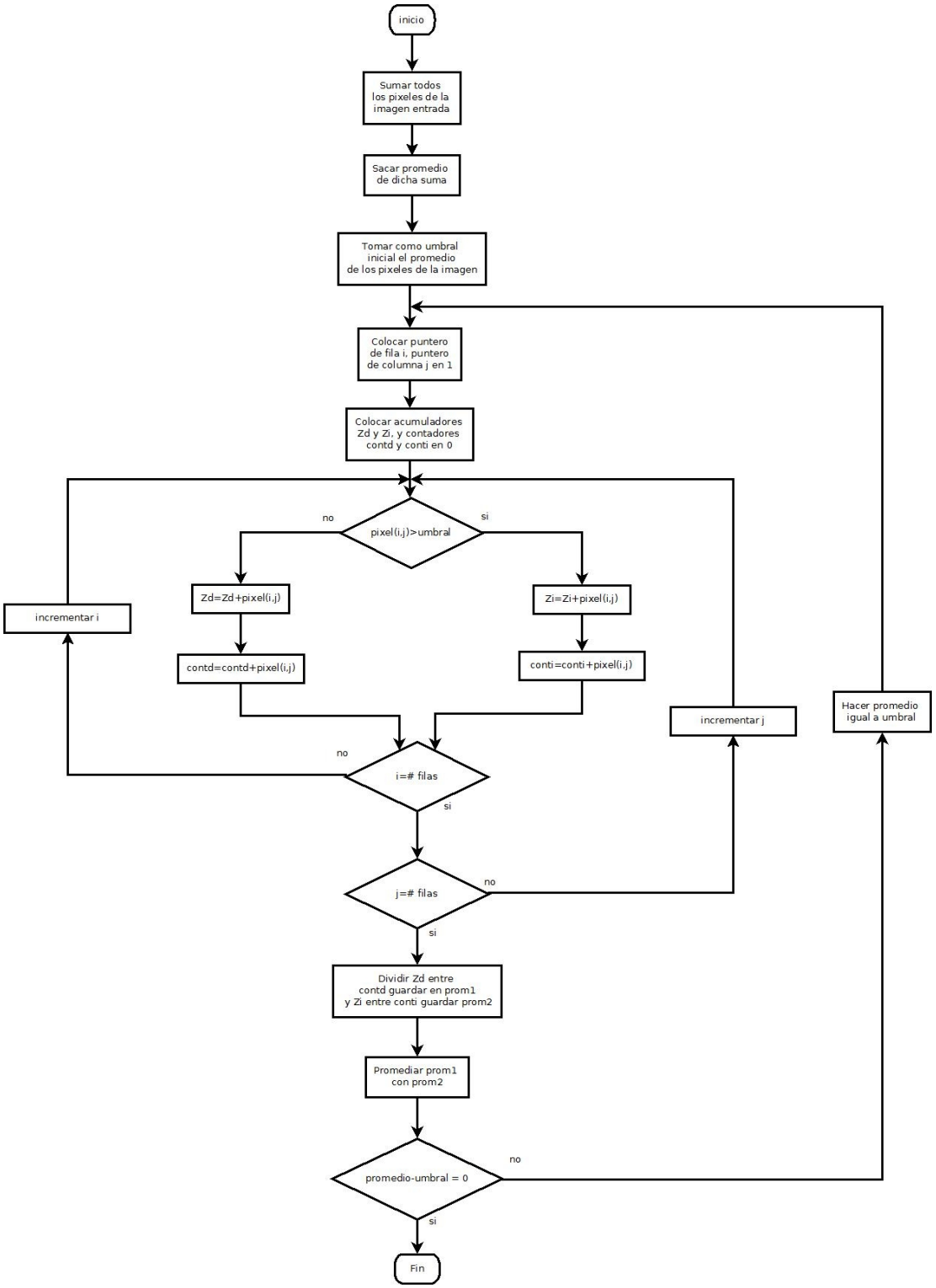
2) Función Recorte de Área Útil



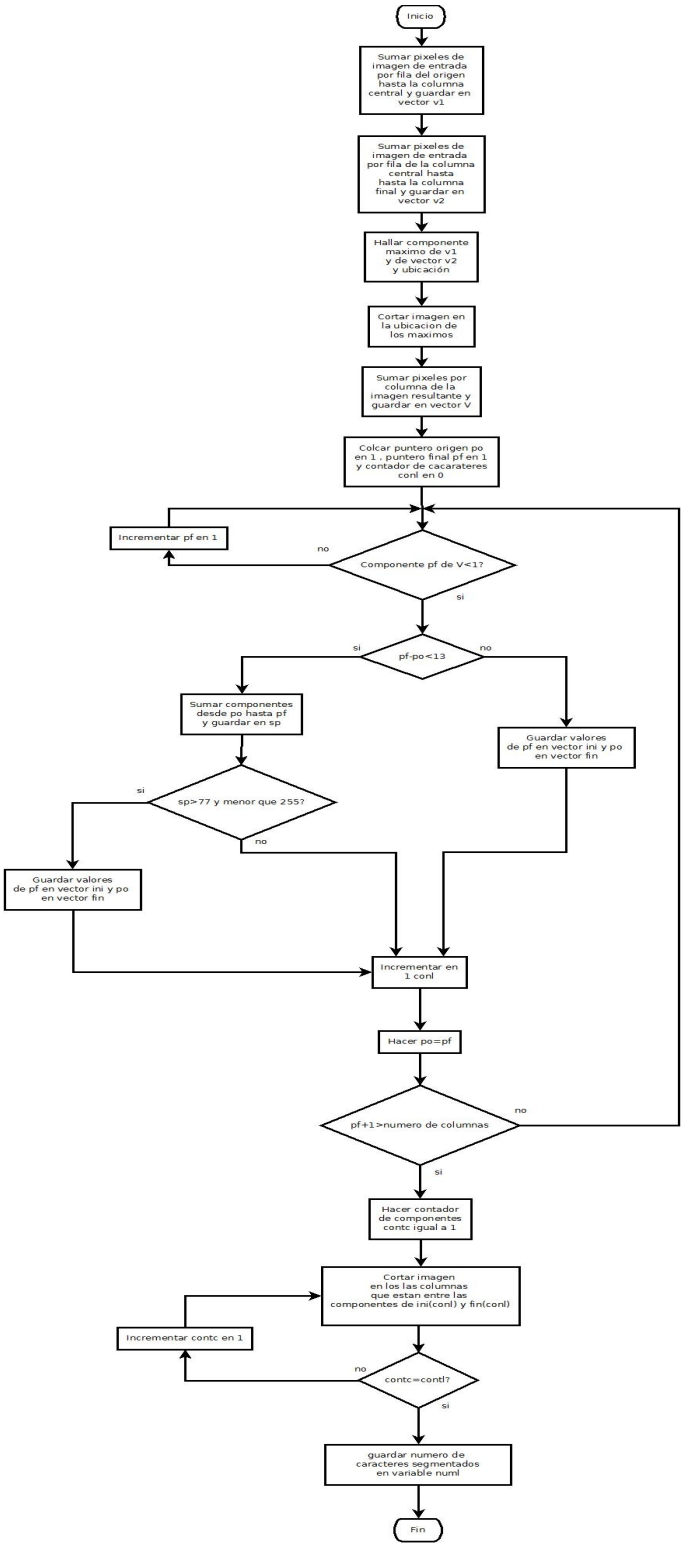
3)Función Recorte de Placa



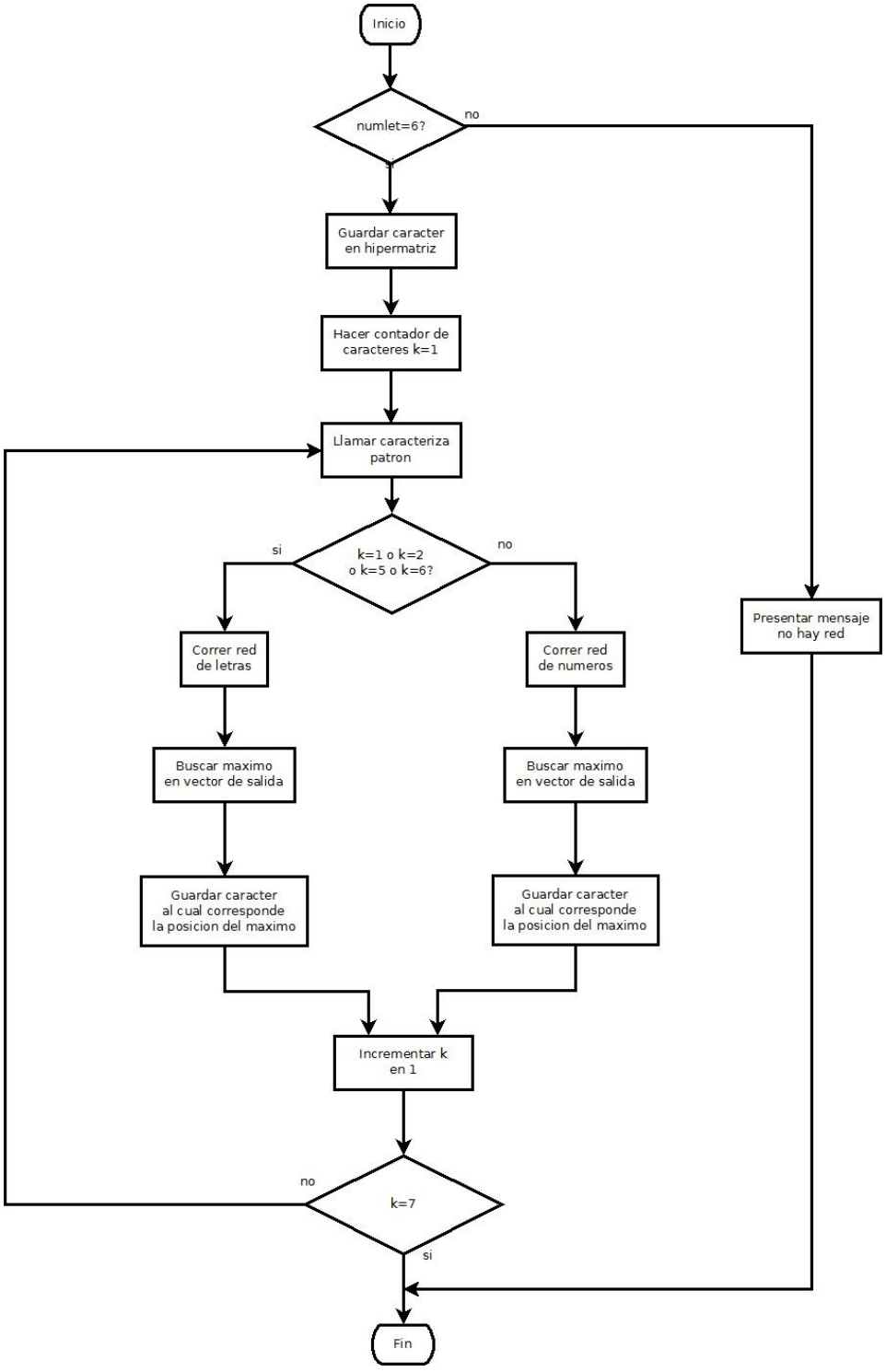
4)Función Umbralizado



5)Función Corta Carácter



6) Función Reconoce



7)Función Entrena Red