Centro de Investigaciones en

Óptica, A. C.

RECONOCIMIENTO AUTOMÁTICO DE MATRÍCULAS DE AUTOMÓVIL

Tesis presentada por

Luis Enrique Toledo Muñoz

Como requisito para obtener

el grado de Maestro en Ciencias (Óptica)

Asesor
Dr. Francisco Javier Cuevas de la Rosa

León, Gto Agosto del 2005

RESUMEN

Presentamos un algoritmo para la lectura de placas de automóvil. Presentamos un nuevo método de detección de la posición de la placa en una imagen, comparando sus resultados con el método de correlación. Seleccionamos el área específica de los caracteres en la placa usando los perfiles de la imagen obtenidas a partir de la imagen de bordes. Explicamos la implementación de una red neuronal de tres capas entrenada con el algoritmo de retropropagación. para el reconocimiento de letras y números. Proponemos un método de verificación para decidir si la imagen que hemos seleccionado es realmente una placa, y en caso de que no lo sea, de buscar en otra zona dentro de la misma imagen.

Palabras clave: Reconocimiento de patrones, procesamiento digital de imágenes, reconocimiento de caracteres, red neuronal, correlación cruzada, lectura de placas de automóvil.

Agradecimientos

Quiero agradecer a mis padres y hermanos, cuyo apoyo me ha permitido llegar hasta aquí.

A mis compañeros, pues juntos hemos recorrido esta etapa.

A mi asesor, Dr. Francisco Cuevas, por el apoyo y consejos que me ha dado y el tiempo que me ha dedicado.

Al personal de DFA por todo el apoyo que nos han brindado (Guille, Laura, Mary, Dr. Servín, Dr. Horacio Barbosa, etc).

A Guillermo Garnica Campos por su invaluable apoyo en la escritura del código fuente, y a Juan Antonio Rayas Álvarez por su apoyo técnico en el manejo de la cámara CCD y sistemas ópticos.

En general al personal del CIO por su comprensión y cooperación.

Al CONACYT por la beca que me fue otorgada, y que me permitió realizar los estudios de la Maestría en Ciencias (Óptica).

Tabla de contenidos

Lista de figuras

Capítulo 1

Capítulo 2

Capítulo 3

Capítulo 5

Capítulo 6

Lista de Tablas

Capítulo 3

Capítulo 5

CAPÍTULO 1

Consideraciones Generales

1.1 Antecedentes.

El interés por los métodos de tratamiento digital de imágenes deriva de dos áreas principales de aplicación: la mejora de la información visual para la interpretación humana y el procesamiento de los datos de la escena para la percepción autónoma por una máquina. Una de las aplicaciones iniciales de la primera categoría de técnicas de tratamiento de imágenes consistió en mejorar las fotografías digitalizadas de periódico enviadas por cable submarino entre Londres y Nueva York. La introducción del sistema Bartlane de transmisión de imágenes a principios de la década de 1920 redujo el tiempo necesario para enviar una fotografía a través del Atlántico de una semana a sólo tres horas[1]. Un equipo especializado de impresión codificaba las imágenes para transmitirla por cable y luego las reconstruía en el extremo de recepción. Algunos de los problemas iniciales para mejorar la calidad visual de éstas imágenes estaban relacionados con la selección de procedimientos de impresión y la distribución de niveles de brillo. Los sistemas Bartlane eran capaces de modificar imágenes en cinco niveles de brillo distinto, y se incrementaron a quince niveles en 1929 [2].

Las mejoras en los métodos de procesamiento para las imágenes digitales transmitidas continuaron durante los siguientes treinta y cinco años, pero fue con el desarrollo de las computadoras digitales de gran potencia, y del programa espacial, lo que mostró el potencial del tratamiento digital de imágenes. En 1964 las imágenes de la luna mandadas por el Ranger 7 fueron procesadas por una computadora para corregir la distorsión inherente a la cámara de televisión a bordo [3].

Desde entonces, las técnicas de procesamiento digital de imágenes (PDI) se han refinado, y actualmente se utilizan para resolver problemas de diferentes áreas: en medicina, se preprocesan las imágenes para que el médico las interprete (rayos X, tomografías, etc); en arqueología se han utilizado métodos de PDI para reconstruir imágenes borrosas que eran los únicos registros de piezas perdidas o dañadas después de haber sido fotografiadas; en física, se utilizan para realzar imágenes de los experimentos en áreas como plasmas de alta energía y la microscopia electrónica; en astronomía, biología, investigaciones judiciales, militares e industriales. Los problemas típicos que son abordados son: reconocimiento automático de caracteres, la visión industrial mecanizada para la inspección de productos, los reconocimientos militares, métodos biométricos de identificación, las muestras de sangre, las imágenes de rayos X y el procesamiento automático de las imágenes aéreas y de satélites para la predicción del tiempo y la evaluación de cultivos [4].

Las técnicas de visión robótica se aplican en una amplia gama de aplicaciones tales como: metrología, óptica, inspección industrial, diagnóstico médico, reconocimiento óptico de caracteres, percepción remota, y otras.

Se estima que aproximadamente las tres cuartas partes de la información que maneja un ser humano es visual, lo que pone de relieve qué tan vital es la visión para los seres vivos. Así, parece natural pensar que el objetivo de dotar a las máquinas del "sentido de la vista" supondrá un salto cualitativo en sus capacidades de actuación.

En los primeros años del siglo XXI se disparó el interés por desarrollar nuevos mecanismos de seguridad, lo que permitió el surgimiento de nuevas tecnologías de control y acceso poco conocidas por el público a fines del siglo XX. Tecnologías como los nuevos sistemas de identificación biométricos (voz, rostro, firma, etc) aparecieron como nuevas opciones además de las huellas dactilares, y se han vuelto tema de discusión en los foros públicos.

El reconocimiento automático de placas emergió como área de investigación en los 80's, y comenzaron a venderse hacia 1993 [5]. Los sistemas para control vehicular mediante el reconocimiento de sus matrículas destacan entre las nuevas tecnologías por su amplia gama de aplicaciones, aunque su eficiencia no resultó buena en un principio y solo recientemente ha mejorado con el uso de sistemas infrarrojos y algoritmos refinados.

En Londres existe desde el 2003 un programa gubernamental para reducir el congestionamiento que ocurría en horas pico en el centro de la ciudad; para lograr esto instalaron cámaras en la zona, y mediante un software de reconocimiento de placas pueden saber qué vehículo ingresó, y comparar el número así obtenido con una base de datos donde se ve si el dueño prepagó el impuesto o no, y en caso de no haber pagado se le envía a su

casa un requerimiento de pago. Aunque el sistema se planeó originalmente para ser temporal, las autoridades británicas pretenden dejarlo en forma permanente y extenderlo a todo el Reino Unido ya que, aseguran, permitirá combatir la delincuencia en una forma más eficiente [6]. Por ahora existe un encendido debate enfocado principalmente en la privacidad, pero también se discuten aspectos relacionados con su rendimiento, ya que el software usado no es completamente efectivo y se habla de tasas de error de hasta 40 %, además de que han surgido problemas derivados de la clonación de autos [7]. En general, se pueden obtener porcentajes de reconocimiento de hasta 90% en condiciones ideales de iluminación y colocación óptima de la cámara [8].

En Italia la compañía Advanced Technologies implementó un sistema de reconocimiento de placas para control de acceso a lugares restringidos [9].

1.2 Aplicaciones.

Aún así, existen muchas otras aplicaciones para esta tecnología, no tan controversiales como la explicada en la sección anterior. En total podemos contar con (siguiente página):

1 **Estacionamientos.** Puede darse acceso inmediato a personas que hayan prepagado su entrada a ciertas zonas, o calcular el tiempo para los demás autos. Se puede incluso generar un boleto por auto, que incluya el número de la placa y/o la imagen del conductor, para evitar robos.

2 **Control de acceso.** La puerta se abre automáticamente para dar acceso a personas autorizadas. Los eventos pueden guardarse en una base de datos para mantener el historial de accesos.

3 **Pago de cuotas.** El número de matrícula puede usarse para calcular el pago de cuotas en autopista. Generalmente se usa junto con un transpondedor para individualizar el auto. En telecomunicaciones, un transpondedor o transponder es un dispositivo que emite una señal identificable en respuesta a una interrogación. El término surge de la fusión de las palabras Transmitter (Transmisor) y Responder (Respondedor). Básicamente existe dos tipos de transpondedor: los pasivos y los activos. Los transpondedores pasivos son aquellos elementos que son identificados por scaners, robots u ordenadores, tales como las tarjetas magnéticas, las tarjetas de créditos, o esas etiquetas en forma de espiral que llevan los productos de los grandes almacenes. Para ello es necesario que interactue con un sensor que decodifica la información que contiene y la transmite al centro de datos. Generalmente estos transpondedores tienen un alcance muy limitado, del orden de un metro. Los transpondedores activos son empleados en sistemas de localización, navegación o posicionamiento. En estos sistemas, el transpondedor responde en una frecuencia distinta en la que fue preguntado, y ambas la de entrada y salidas de datos, están predefinidas de antemano. En estos casos los alcances son gigantescos tanto que se emplean sin problema alguno en toda la transmisión actual de equipos espaciales.

4 **Detección de autos robados.**

5 **Control de velocidad de tráfico vehicular.**

1.3 Objetivo**s.**

Nuestro objetivo final es el la obtención de una cadena de caracteres ASCII(American Standard Code for Information Interchange) a partir de la imagen en escala de grises, que nos reconozca el número impreso en la matrícula del auto en cuestión.

Por ahora sólo nos ocuparemos de las cuestiones relativas al software del sistema; partiremos de una imagen frontal en escala de grises, sin tomar en cuenta cómo fue obtenida. Aunque esto no siempre es posible, pues hay diferencias en las imágenes dependiendo de cómo fueron tomadas; estos casos especiales se tratan dentro de la sección del algoritmo que es afectada.

Existe una gran variedad de formatos para placas. Nosotros queremos un algoritmo que pueda, en principio, tratar en forma general con cualquier placa. Pero por ahora sólo nos ocuparemos de placas de Guanajuato o que tengan una estructura parecida (figs. 1.1 y 1.2).

Figura 1.1 Placa del estado de Guanajuato.

Figura 1.2 Placa del estado de Jalisco.

Las placas del estado de Guanajuato pueden tener varios formatos. La "Ley de tránsito y transporte del Estado de Guanajuato" en el capítulo V, artículo 47, lista los tipos de placas que expide la Secretaría de Planeación y Finanzas:

I.- Para vehículos de uso particular.

II.- Para vehículos de servicio público.

III.- Para vehículos destinados a patrullas.

IV.- Para demostración o traslado de vehículos.

V.- Para unidades tipo remolque.

VI.- Para motocicletas.

VII.- Para vehículos para uso de personas con discapacidad.

De los tipos anteriormente listados sólo trabajaremos sobre vehículos de uso particular (fig. 1.1).

Así mismo, no consideraremos ejemplos de placas que violen los siguientes puntos del "Reglamento de Tránsito de la Ley de Tránsito y Transporte del Estado de Guanajuato" indicadas en la página 43:

- Con colgajos o adherencias.
- Dobladas o colocadas en forma incorrecta.

Una característica extra que se busca es minimizar los costos de implementación del sistema, lo cuál daría una ventaja comercial. Para lograr este objetivo de utilizará hardware comercial (computadora, cámara digital y tarjeta digitalizadora), así como software propio no dependiente de librerías comerciales.

1.4 Algoritmo.

Podemos dividir todo el proceso necesario para obtener la cadena de caracteres ASCII en las siguientes fases (fig. 1.3).

Captura de imagen.

Obtención de la imagen, ya sea a color o en escala de grises. En caso de ser a color, debemos convertirla a escala de grises o hacer un pretratamiento en color que faciliten los procesos posteriores.

Detección de la placa.

Eliminación de áreas no necesarias para nuestro objetivo. Deseamos disminuir el área de búsqueda.

Detección de caracteres.

Delimitación del área de caracteres dentro de la placa. Podemos hacer uso de estructuras generales de las placas para lograr este objetivo. Con este paso pretendemos facilitar el trabajo de segmentado.

Segmentación de caracteres

Extracción de las imágenes de cada caracter individualmente.

Reconocimiento óptico de caracteres (OCR por sus siglas en inglés)

A partir de las imágenes obtenidas en el paso anterior, verificar si son o no son caracteres y a cuál caracter corresponden. En este trabajo escogimos una red neuronal.

Cadena de caracteres

Salida del sistema, caracteres codificados en ASCII correspondiendo al número mostrado en la placa. Nos enfocaremos en conseguir un algoritmo rápido, para que pueda ser usado en aplicaciones donde la velocidad de reconocimiento sea importante.

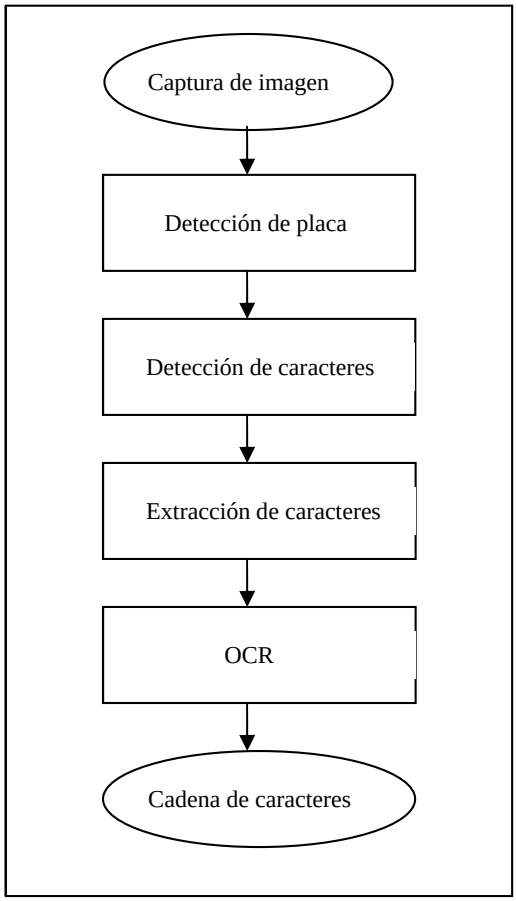


Figura 1.3 Diagrama de bloques de un sistema reconocedor de placas de auto.

1.5 Obtención del grupo de imágenes y equipo utilizado.

Los algoritmos se corrieron en una computadora Pentium III 1.5 GHz, 256 Mb RAM. Se usó Visual C++ para la implementación de todos los algoritmos. Como tarjeta digitalizadora usamos Meteor II de Matrox Graphix. Como cámara de captura de imágenes usamos una cámara CCD COHU y cámaras digitales varias.

Usamos un grupo de 73 imágenes para la pruebas de todos los algoritmos. Las imágenes fueron tomadas durante el día, a diferentes distancias (entre dos y cuatro metros) y condiciones de iluminación (interiores y exteriores).

1.6 Estructura de los capítulos.

El capítulo 2 hace un repaso de los conceptos fundamentales de reconocimiento de patrones y bases teóricas de los algoritmos de procesamiento digital de imágenes.

En el capítulo 3 veremos los algoritmos de detección de la placa, lo que consideramos representa la aportación de nuestro trabajo. Están basados en la búsqueda de zonas de alta frecuencia, que suponemos asociadas a la presencia de caracteres. Estas zonas de altas frecuencias se encuentran a partir del suavizado de la imagen de bordes verticales. La imagen de bordes se encuentra usando Sobel. El suavizado lo realizamos con la convolución de la imagen con una ventana de *100x20* (columnas x filas), cien píxeles de ancho por veinte pixeles de alto. Si ordenamos las zonas resultantes según el valor máximo que presentan, tenemos un parámetro que nos dice donde es probable encontrar una placa.

El capítulo 4 muestra cómo, una vez recortada, la placa, podemos encontrar los caracteres. Si suponemos que la placa casi no presenta giros, podemos separar el área principal de caracteres a partir de la imagen de bordes verticales. Sumando los píxeles de cada renglón, obtenemos una proyección vertical de la imagen de la placa. Umbralizando esta proyección, obtenemos varias zonas de las cuáles la más grande y centrada debe corresponder a los caracteres. Delimitamos esta zona de caracteres en horizontal usando el gradiente de la imagen de la placa, obteniendo la proyección de las columnas de la imagen. Suavizando esta proyección, obtenemos el área de los caracteres. Finalmente separamos los caracteres umbralizando la imagen.

El capítulo 5 hace un repaso del uso de las redes neuronales para el reconocimiento de patrones, y su aplicación para el reconocimiento de caracteres. Explica cómo obtenemos el valor *E* que nos permite clasificar el área como una placa, sumando el cuadrado de la máxima salida de la red para cada caracter. Suponemos que las placas son semejantes a la mostrada en las figuras 1.1 y 1.2, así que buscamos las posiciones de los guiones, lo que nos sirve para decidir si los caracteres son números o letras. Lo anterior permite eliminar los casos ambiguos o difíciles de diferenciar como "8" y "B", "I" y "1", etc.

El capítulo 6 nos presenta cómo se comporta el programa sección por sección, y su funcionamiento global. Discutimos sus limitaciones y proponemos caminos de desarrollo a futuro.

CAPÍTUL0 2

Teoría de Reconocimiento de Patrones

Y

Procesamiento Digital de Imágenes

2.1 Aspectos generales de la teoría de reconocimiento de patrones.

Un sistema que reconozca automáticamente patrones o formas, trabaja siempre con un universo de trabajo previamente definido. Este universo de trabajo está formado por una muestra de los elementos que deben ser reconocidos (patrones, que es una traducción literal del término en inglés pattern).

Sin embargo en el mundo real, estos elementos están mezclados entre sí, formando relaciones e interacciones complejas; la operación para aislar los elementos de un objeto se llama segmentación.

Finalmente, para poder hacer la identificación del patrón específico, es necesario describirlo en función de sus características o rasgos diferenciadores, de tal manera que estos rasgos puedan agruparse en la forma de un vector $\boldsymbol{X}$:

$$X = \begin{pmatrix} x^1 \\ x^2 \\ \cdot \\ \cdot \\ xn \end{pmatrix} , \qquad (2.1)$$

(los vectores se indicarán mediante letras mayúsculas y en negritas). Cada vector de características se compara con un conjunto de valores preestablecido o diccionario, compuesto por los vectores de rasgos de todos los objetos del universo de trabajo; la comparación se basa en determinar su grado de semejanza con los vectores de características prototipos.

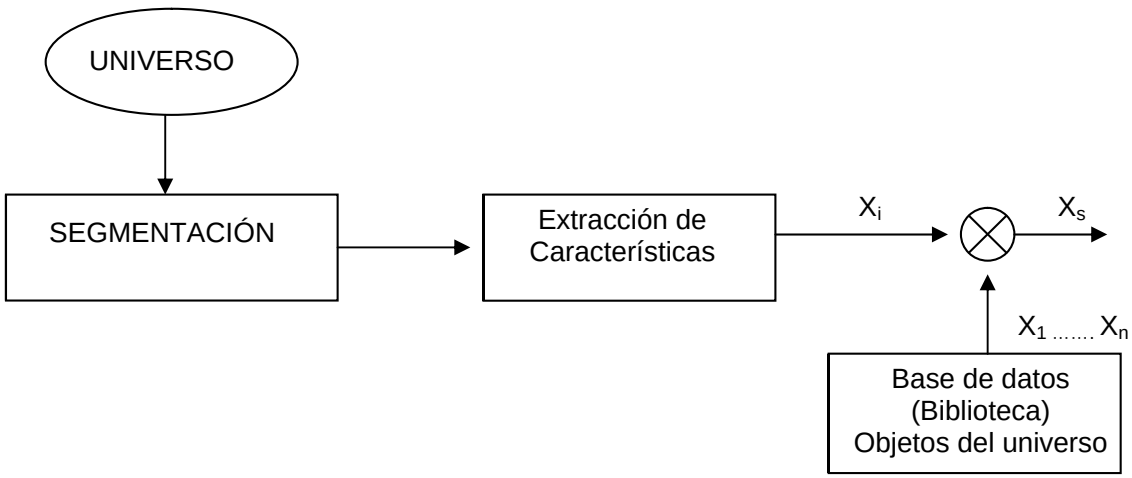


Fig. 2.1 Diagrama de bloques de un sistema de reconocimiento de patrones

La notación que se usa normalmente para los vectores de características es en la forma de un vector columna. Para indicar los vectores horizontales se usará la notación $\mathbf{X}^{\mathbf{T}}$.

Llegados a este punto nos enfrentamos con un problema ¿cómo determinar el grado de semejanza de los vectores de características con los vectores prototipos? Para contestar esta pregunta simplificaremos un poco el problema. Imaginemos que tenemos un grupo de objetos que deben ser clasificados en dos clases diferentes, y para hacerlo tomamos dos rasgos diferenciadores. Construyamos una gráfica bidimensional, donde los ejes *X* e *Y* sean las características diferenciadoras; el plano entonces puede dividirse en dos sectores con una curva, cuya ecuación es llamada función discriminante *fd(**X**)*, que generalmente es una recta. Los objetos a clasificar deben caer en una de las regiones en que se divide el plano, y de esta manera podemos saber a que clase pertenecen (ver figura 2.2).

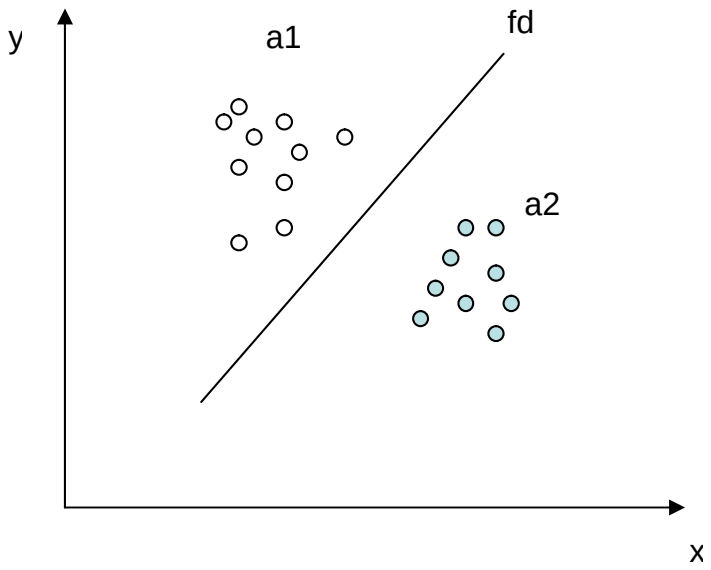


Fig. 2.2 Uso de recta discriminante para separar dos regiones distintas.

Es importante notar que los puntos situados en un semiplano dan lugar a un signo determinado de *fd* (positivo o negativo), y que los puntos sobre la recta discriminante producen *fd=0.*

La ecuación para la recta que divide al plano es:

$$fd(X) = W^T \bullet X = \begin{bmatrix} W_1 & W_2 & W_3 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} \quad , \qquad (2.2)$$

En caso de tener más clases, es necesario agregar más rectas.

Cuando las caraterísticas diferenciadoras son más de dos, en lugar de un plano tenemos un hiperespacio, dividido por hiperplanos.

La solución general consiste, pues, en encontrar un hiperplano $\boldsymbol{W^T X}$ que divida el espacio de las clases en dos regiones, asociadas cada una de ellas a diferentes clases. Obviamente, cuando el número de clases sea superior a dos, es preciso obtener más de un hiperplano.

Para *N* clases $\alpha_1,\alpha_2, \ldots,\alpha_N$ no es trivial establecer el número mínimo de funciones discriminantes que se necesitarán para dividir el espacio de las clases en *N* regiones distintas, cada una asociada con una sola clase. En el peor caso, se necesitarán $\binom{N}{2}$ funciones discriminantes, al requerirse una *fd* para cada una de las posibles combinaciones de dos clases que se pueden formar con *N* clases. En el caso óptimo, se necesitarán *N-1* funciones discriminantes, ya que se podría discriminar cada una de las clases con el resto, considerado éste como una sola clase.

El anterior planteamiento del problema de reconocimiento de patrones nos ha conducido a una solución lineal basada en una regionalización del espacio de las clases.

Existe otra solución lineal basada en el uso de medidas de distancia, también dentro del espacio abstracto de las clases.

Ésta nueva técnica consiste en asociar un vector $\boldsymbol{Z_i}$, único o prototipo de cada clase (habitualmente puede tomarse la media aritmética de los objetos, mientras no exista una dispersión elevada). Entonces, si tenemos un vector de características $\boldsymbol{X}$, la solución consiste en asociar $\boldsymbol{X}$ a la clase que esté más cercana.

El siguiente desarrollo demuestra que si se emplea la distancia euclídea, el clasificador tiene entonces una expresión matemática similar al clasificador por regiones. Esto se logra determinando la distancia Euclídea entre el vector de características y el vector prototipo de una clase genérica α_i :

$$d(X, Z_i) = \sqrt{(X - Z_i)^T (X - Z_i)} = \sqrt{\sum_{j=1}^{n} (X_j - Z_{ij})^2} \quad , \tag{2.3}$$

Tomando el cuadrado de la distancia euclídea y desarrollando:

$$d^2(X, Z_i) = (X - Z_i)^T (X - Z_i) = X^T \bullet X - 2X^T \bullet Z_i + Z_i^T \bullet Z_i \quad , \tag{2.4}$$

El factor $\boldsymbol{X^T * X}$ no es un discriminante, por lo que puede ignorarse. Cambiando los signos, se llega a la siguiente función discriminante de la clase α_i :

$$fd_i(X) = Z_i^T \bullet X - \frac{1}{2} Z_i^T \bullet Z_i \quad , \tag{2.5}$$

que es similar a la que aparecería en la discriminación por regiones, es decir, tiene la forma $\boldsymbol{W_i^T * X}$ siendo:

$$W_i^T = \left(Z_{i1} \quad Z_{i2} \qquad Z_{in} - \frac{1}{2} \sum_{j=1}^{n} Z_{ij}^2 \right), \tag{2.6}$$

Ahora, la clasificación no se basa en signos, que determinan las regiones, sino en la dimensión de un escalar fd_i , que al ser complementario a una distancia en virtud del cambio de signos operado, dará una pertenencia más acentuada de un vector $\boldsymbol{X}$ a una clase ω_i cuanto mayor sea en relación a los demás fd_j $(j \neq i, j=1,2,...N)$.

En ciertos casos, puede ocurrir que las clases sean desconocidas a priori, como por ejemplo en medicina y biología, donde no está bien clarificado el universo de las clases. En estas situaciones se recurre a las técnicas de agrupación (clustering), es decir, reconocimiento sin supervisión, y difieren de las técnicas anteriormente mostradas en que no existe una supervisión o reconocimiento externo que guíe el diseño de las funciones de discriminación, como ocurría en los dos métodos anteriores. Estas técnicas también pueden utilizarse en situaciones con conocimiento previo de las clases.

2.2 Técnicas de procesamiento digital de imágenes

2.2.1 Representación digital de imágenes.

El término *imagen* se refiere a una función bidimensional de intensidad de luz $f(x,y)$, donde x e y representan las coordenadas espaciales, y el valor de $f(x,y)$ es proporcional al brillo (nivel de gris) de la imagen en ese punto, que se representa mediante un tercer eje . La definición anterior implica que sólo trabajaremos con imágenes monocromáticas, ya que éstas son las más utilizadas para el procesamiento de imágenes, aunque también pueden

utilizarse imágenes a colores, utilizando por lo general tres funciones para indicar la intensidad de cada uno de los colores primarios (generalmente rojo, verde y azul) [10].

Una imagen digital es una imagen *f(x,y)* que se ha discretizado tanto en las coordenadas espaciales como en el brillo. Una imagen digital puede considerarse como una matriz cuyos índices de fila y columna identifican un punto de la imagen, y el valor del correspondiente elemento de la matriz indica el nivel de gris de ese punto o pixel. Las filas se identifican con el punto *x* y se cuentan desde la parte superior de la imagen; y las columnas se identifican con *y,* contadas desde el extremo izquierdo de la imagen.

2.2.2 Etapas fundamentales del procesado de imágenes.

El objetivo global de todo sistema de PDI es producir un resultado a partir de un determinado problema por medio del procesamiento de imágenes. Antes de comenzar el proceso, es necesario definir el dominio del problema, los elementos sobre los que se va a trabajar, y el objetivo del problema. Además, es necesario contar con una base de conocimiento, o conocimiento previo sobre el dominio del problema; este conocimiento puede ser simple, como detallar las regiones de una imagen donde se sabe que se única información de interés (limitando así la búsqueda que ha de realizarse para hallar tal información), o un archivo de información que va a ser comparada con las imágenes a ser procesadas (p.ej. en autentificación y reconocimiento de personas, información sobre defectos de materiales, etc.). La relación de esta base de conocimiento con el procesado de

la imagen no es sólo unidireccional, sino que puede existir una interacción entre ambos para enriquecer o modificar esta base.

La primera etapa del proceso es la adquisición y digitalización de la imagen. Este no es un problema menor, pues en muchas aplicaciones hay que escoger cuidadosamente tanto las cámaras, como las lentes, los sistemas de digitalización y la posición de la cámara respecto del objeto que mejor se ajusten a los requerimientos del problema, así que en muchos de los problemas que son investigados, se supone cubierto este problema y se trabaja ya con las imágenes obtenidas. En este trabajo también se supondrá que las imágenes ya existen en la memoria de la computadora.

La siguiente etapa es el preprocesamiento de la imagen, el cuál consiste en mejorar la imagen de forma que se faciliten los procesos posteriores. En esta etapa pueden incluirse la mejora del contraste, eliminar el ruido o aislar regiones de interés.

Sigue la segmentación. Consiste en partir una imagen de entrada en sus partes constituyentes u objetos, es decir, obtener de la imagen los elementos que nos interesan para resolver el problema.

A la salida del proceso de segmentación se tienen todos los pixeles en bruto, que constituyen bien el contorno de una región o bien todos los puntos de una región determinada. En cada caso es necesario convertir los datos a una forma adecuada para el

procesamiento por computadora. La representación como un contorno es adecuada cuando lo que nos interesa son características de la forma exterior como esquinas e inflexiones. La representación regional es adecuada cuando lo que interesa son propiedades internas como la estructura o la textura.

La elección de una representación, para ser útil, debe complementarse con una descripción (o selección de rasgos) que resalte los rasgos de interés, que tengan información cuantitativa del objeto o rasgos diferenciadores.

La última etapa incluye el reconocimiento y la interpretación. El reconocimiento es el proceso que asigna una etiqueta a un objeto basándose en la información proporcionada por sus descriptores. La interpretación implica asignar significado a un conjunto de objetos reconocidos [11].

2.2.3 Técnicas básicas de PDI.

Existen básicamente dos tipos de técnicas: métodos en el dominio espacial y métodos en el dominio de la frecuencia. El dominio espacial se refiere al propio plano de la imagen, y las técnicas de esta categoría se basan en la manipulación directa de los pixel de la imagen. El procesamiento en el dominio de la frecuencia se basa en la modificación de la transformada de Fourier de una imagen.

2.2.3.1 Métodos en el dominio espacial.

Las funciones de procesamiento de la imagen en el dominio espacial pueden expresarse como:

$$g(x, y) = T\left[f(x, y)\right] \ , \tag{2.7}$$

donde $f(x,y)$ es la imagen de entrada, $g(x,y)$ es la imagen procesada y T es un operador que actúa sobre f. T puede también operar sobre varias imágenes al mismo tiempo, como cuando se hace la suma pixel por pixel de M imágenes para reducir el ruido.

La aproximación principal para definir un entorno alrededor de (x,y) es emplear un área de de subimagen cuadrada o rectangular centrada en (x,y). El centro de la subimagen se mueve píxel a píxel comenzando, por ejemplo, en la esquina superior izquierda y aplicando el operador en cada posición $(x.y)$ para obtener g (a veces se utilizan otros entornos, como aproximaciones a un círculo).

La forma más simple de T corresponde a un entorno 1x1, lo que implica que g depende sólo del valor de f en (x,y), y T se convierte en una función de transformación del nivel de gris (correspondencia) de la forma:

$$s = T(r) \ , \tag{2.8}$$

donde, para simplificar la notación, r y s son variables que indican el nivel de gris de $f(x,y)$ y $g(x,y)$.

Las técnicas que funcionan con una función de transformación de nivel de gris se conocen como procesamiento de punto.

Los entornos de dimensiones diferentes tienen aún más aplicaciones, tales como el empleo de máscaras (también llamadas plantillas, ventanas o filtros). Una máscara es una pequeña distribución bidimensional (p.ej. 3x3) en la que los valores de los coeficientes determinan la naturaleza del proceso, como la acentuación de los bordes. Las técnicas de mejora basadas en este tipo de aproximación se conocen como procesamiento por máscaras o filtrado [12].

A continuación, se presentarán algunas técnicas básicas de PDI en el dominio espacial, por procesamiento por punto. Las imágenes procesadas están en formato BMP (mapa de bits) de 256x256 con 256 niveles de gris.

2.2.3.1.1 Alteración global del brillo.

Se logra sumando una constante al valor de niveles de gris de cada pixel:

$$g(x, y) = f(x, y) + K \ , \tag{2.9}$$

Si el nivel de gris excede el mayor nivel de gris, se deja el nivel en ese nivel. K es el brillo que se desea aumentar o disminuir.

a) b)

c)

Fig. 2.3 a) es la imagen original, b) es aumento de brillo, c) disminución del brillo.

2.2.3.1.2 Binarizado.

Los niveles de gris que estén sobre un cierto nivel se hacen iguales a 256 (blanco), y aquello que caigan debajo de ese nivel se hacen cero (negro).

Fig. 2.4 Binarizado de la imagen.

2.2.3.1.3 Cuantizado.

Es semejante al binarizado, sólo que aquí lo que se hace es definir varios niveles de gris y una región para cada nivel; aquellos niveles de gris que caigan dentro del rango especificado se igualan al nivel previamente definido.

a)

b)

Fig. 2.5 Reducción a diez niveles de gris.

2.2.3.1.4 Histograma.

Obtiene una gráfica de los niveles de gris contra número de pixeles. Se logra definiendo un vector con un número de espacios igual al número de tonos de gris que queremos. Si leemos un pixel y detectamos un nivel de tonos de gris x, en el lugar h[x] sumamos "1" a la cantidad anterior.

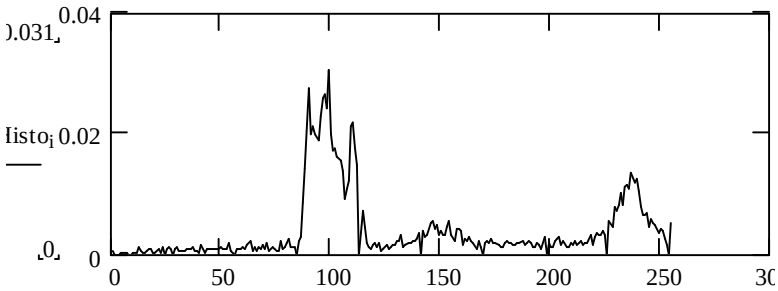


Fig. 2.6 Imagen original y su histograma (frecuencia de niveles de gris en una imagen).

2.2.3.1.5 Ecualizado.

Permite modificar el contraste de una imagen: si la imagen está oscura, aumenta el brillo, y si está muy clara, disminuye el brillo.

Para realizarlo, calculamos las frecuencias de cada tono de gris. Comenzando desde el menor nivel de gris, obtenemos la frecuencia acumulada de los tonos de gris. Calculamos una constante llamada *nipn*, cuyo valor es igual al número de renglones por número de columnas, entre el número de tonos de gris. Dividimos la frecuencia acumulada entre *nipn*, y el resultado nos relaciona el histograma de la imagen original con un nuevo histograma donde tenemos un aprovechamiento del rango dinámico más uniforme.

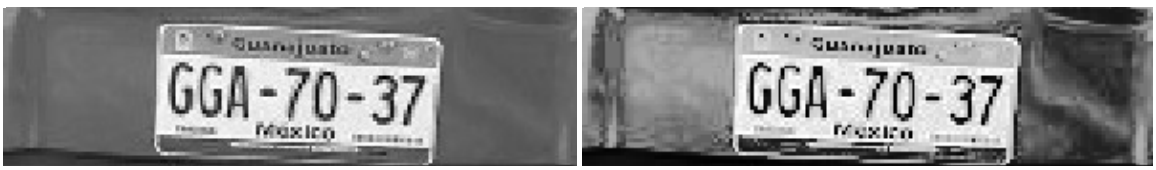


Fig. 2.7 El ecualizado permite que todos los niveles de gris se usen probabilidades similares.

2.2.3.1.6 Negativo de una imagen.

Permite hacer la inversión de los niveles de gris de una imagen. Se logra mediante:

$$g(x, y) = 256 - f(x, y) \ , \tag{2.10}$$

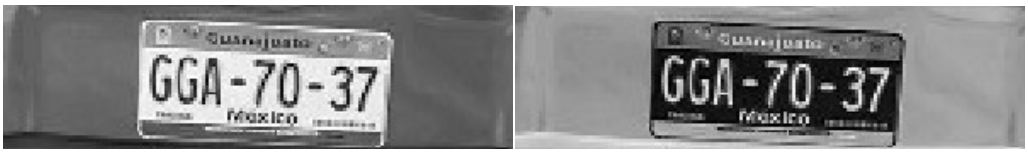


Fig. 2.8 Negativo de la imagen.

2.2.3.1.7 Pseudocolor.

Este método permite "agregar" colores a una imagen monocromática. Básicamente es el mismo algoritmo que para el cuantizado, sólo que en vez de usar niveles específicos de colores de gris, se utilizan colores.

2.2.3.2 Métodos en el dominio de la frecuencia.

La base de las técnicas en el dominio de la frecuencia [13] es el teorema de convolución. Sea *g(x,y)* una imagen formada por la convolución de una imagen *f(x,y)* y un operador lineal invariante de posición (aquél cuyo resultado depende sólo del valor de *f(x,y)* y no de su posición) *h(x,y)* :

$$g(x, y) = f(x, y) * h(x, y) = \int_s \int_t f(s,t)h(x-s, y-t)dsdt \ , \qquad (2.11)$$

Entonces, obteniendo la transformada de Fourier (para una introducción completa al tema de las transformadas de Fourier consultar [14]) de ambos términos y por el teorema de la convolución, podemos afirmar las relaciones 2.12 y 2.13:

$$f(x) * h(x) \Leftrightarrow F(u)H(u) \quad , \tag{2.12}$$

$$f(x)h(x) \Leftrightarrow F(u) * H(u) \quad , \tag{2.13}$$

Se cumple la siguiente relación en el dominio de la frecuencia:

$$G(u,v) = F(u,v)H(u,v) \ , \tag{2.14}$$

Donde $H(u,v)$ es la función de transferencia del sistema y está relacionada con las máscaras cuadradas vistas anteriormente, pues corresponde a la trasformada de Fourier de estas máscaras para un propósito específico. La utilidad de la transformada de Fourier es que nos permite simplificar los cálculos necesarios para obtener el resultado: lo que en el dominio del espacio es una convolución, en el domino del plano (u,v) es una multiplicación, más sencilla de realizar.

Antes de continuar, hay que aclarar dos conceptos: que la transformada de Fourier de una imagen es conceptualmente diferente a la trasformada de una función en el tiempo. La trasformada de Fourier de imágenes nos indica cómo se vería ésta desde un punto situado en el infinito, mientras que la transformada de Fourier de una función en el tiempo nos da las componentes de frecuencia de esa función. Aunque si interpretamos los cambios en los tonos de gris entre pixeles, podemos hablar de frecuencias altas cuando el cambio es grande, y de frecuencias bajas cuando el cambio de tonos de gris es pequeño. El otro concepto es que, al hacer la trasformada, aún cuando la imagen original no tenga partes imaginarias, el resultado se da como una parte real y una parte imaginaria, esto es:

$$F(u) = R(u) + jI(u) \ , \tag{2.15}$$

Esta función también puede representarse como una magnitud y una fase:

$$F(u) = |F(u)| e^{i\phi(u)} \ , \tag{2.16}$$

$$|F(u)| = \left[R^2(u) + I^2(u) \right]^{1/2} \ , \tag{2.17}$$

$$\phi(u) = \tan^{-1} \left[\frac{I(u)}{R(u)} \right] \ , \tag{2.18}$$

En las técnicas de procesamiento digital de imágenes tomamos sólo la magnitud de la función y despreciamos el desfase, pues el ojo humano sólo ve magnitudes. Esto significa que, de la transformada de Fourier de una imagen, sólo pondremos atención a su magnitud. En la imagen de magnitud, las frecuencias se distribuyen radialmente a partir del centro; las frecuencias bajas se encuentran cerca del centro, y las frecuencias altas están alejadas del centro.

Antes de comenzar a utilizar la trasformada de Fourier, es necesario familiarizarse con algunas de sus propiedades.

2.2.3.2.1 Traslación.

Las propiedades de traslación del par de transformadas de Fourier son:

$$f(x,y)\exp\left[j2\pi(uox+voy)/N\right] \Leftrightarrow F(u-u0,v-vo)\ , \tag{2.19}$$

$$f(x-xo,y-yo) \Leftrightarrow F(u,v)\exp\left[-j2\pi(uxo+vyo)/N\right], \tag{2.20}$$

Estas ecuaciones implican que el efecto de trasladar una imagen en el espacio es provocar un desfase en la transformada de Fourier, y viceversa. Pero este desfase no afecta al módulo o magnitud de la trasformada, así que la imagen original de la trasformada permanece sin cambios.

A menudo se utiliza la ec. 2.12 con $uo=vo=N/2$

$$\exp\left[j2\pi(uox+voy)/N\right] = e^{j\pi(x+y)}, \tag{2.21}$$
$$= (-1)^{x+y}$$

y

$$f(x,y)(-1)^{x+y} \Leftrightarrow F(u-N/2,v-N/2)\ , \tag{2.22}$$

Así, el origen de la trasformada de Fourier de *f(x,y)* puede ser desplazado hacia el centro de su correspondiente cuadrado de frecuencias NxN.

2.2.3.2.2 Periodicidad y simetría conjugada.

La trasformada discreta de Fourier y su inversa son funciones periódicas de período *N*, es decir:

$$F(u,v) = F(u+N,v) = F(u,v+N) = F(u+N,v+N)\ , \tag{2.23}$$

Esto nos indica que solamente son necesarios los N valores de cada variable para recuperar *f(x,y)* a partir de *F(u,v).*

Si *f(x,y)* es real, la transformada de Fourier presenta también simetría conjugada:

$$F(u, v) = F^*(-u, -v) \quad , \tag{2.24}$$

$$|F(u, v)| = |F(-u, -v)| \quad , \tag{2.25}$$

Donde $F^*(u,v)$ es el complejo conjugado de *F(u,v).*

Estas últimas ecuaciones implican que la transformada de Fourier es simétrica con respecto a su origen.

2.2.3.2.3 Rotación.

Si introducimos las coordenadas polares:

$$x=r\cos\theta \qquad y=r\,\text{sen}\,\theta \qquad u=\omega\cos\phi \qquad v=\omega\,\text{sen}\,\phi \quad , \tag{2.26}$$

entonces *f(x,y)* y *F(u,v)* se convierten en *f(r, θ)* y en *F(ω,ϕ)*, respectivamente. La sustitución directa en el par de transformadas de Fourier proporciona:

$$f(r, \theta + \theta o) = F(\omega, \phi + \theta o) \quad , \tag{2.27}$$

indicando que si *f(x,y)* se gira un cierto ángulo, su trasformada de Fourier también gira ese ángulo.

2.2.3.2.4 Distributividad y cambio de escala.

De la definición del par de transformadas de Fourier continuas o discretas:

$$F\{f_1(x,y) + f_2(x,y\} = F\{f_1(x,y)\} + F\{f_2(x,y)\} \ , \tag{2.28}$$

Y, en general:

$$F\{f_1(x,y) \bullet f_2(x,y\} \neq F\{f_1(x,y)\} \bullet F\{f_2(x,y)\} \ , \tag{2.29}$$

Es decir, la transformada de Fourier y su inversa son distributivas con respecto de la suma, pero no con respecto a la multiplicación.

Para dos escalares a y b:

$$af(x,y) \Leftrightarrow aF(u,v) \quad , \tag{2.30}$$

$$f(ax,by) \Leftrightarrow \frac{1}{|ab|} F(u/a, v/b) \ , \tag{2.31}$$

2.2.3.2.5 Valor medio.

El valor medio de una función discreta bidimensional se puede obtener como:

$$\overline{f}(x,y) = \frac{1}{N} F(0,0) \quad , \tag{2.32}$$

2.2.3.2.6 Laplaciano.

El laplaciano de una función de dos variables *f(x,y)* se define como:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad , \tag{2.33}$$

De la definición de la transformada de Fourier bidimensional:

$$F\{\nabla^2 f(x,y)\} \Leftrightarrow -(2\pi)^2(u^2+v^2)F(u,v) \quad , \tag{2.34}$$

El operador laplaciano es útil para delimitar los bordes de una imagen.

2.3 Conclusiones.

Los pasos que hay que seguir para la construcción de un sistema automático de reconocimiento de patrones son:

- Establecimiento de las clases (definición del universo de trabajo del sistema).
- Elección del vector de características.

Esta última etapa es la que menos se presta a formalización y aplicación de reglas generales, pues la elección de los rasgos es muy dependiente de la aplicación específica. Aunque puede aplicarse cierta metodología estadística una vez que se han determinado estas características.

Los conceptos y técnicas mostrados en este capítulo son solo la base de los algoritmos que mostraremos en los siguientes capítulos. Los algoritmos para extracción de bordes, suavizado y realzado pueden realizarse ya sea en el dominio del espacio o en el dominio de la frecuencia, aunque suele preferirse operar en el dominio del espacio.

CAPÍTULO 3

Captura de imagen y detección de la placa

En este capítulo explicaremos cómo obtener las posibles ubicaciones donde podemos encontrar una placa. Supondremos que ya tenemos una imagen capturada en escala de grises. El tamaño de la imagen no importa para la aplicación del algoritmo, pero en general se trabajará en imágenes de *640x480* pixeles (640 columnas por 480 filas). Este formato se escogió por ser el mismo de la cámara propuesta para el proyecto (Cohu).

Algunas de las imágenes del grupo de imágenes fueron tomadas con una cámara digital con una resolución mayor que la propuesta, así que tuvimos que reducir su resolución. Las pruebas se hicieron con la cámara Cohu usada en el laboratorio de procesamiento digital de imágenes.

3.1 Detección de placa.

Se han propuesto varios métodos de detección de la posición de la placa en una imagen. Todos hacen uso de alguna característica de las placas como la forma rectangular o la búsqueda de los caracteres.

Entre los métodos que buscan los caracteres existen los que usan la erosión o la dilatación sobre la imagen en escala de grises; la dilatación elimina los detalles oscuros de la imagen si el elemento estructurante tiene todos sus valores positivos [15] y si la placa tiene caracteres oscuros sobre un fondo claro, los caracteres se eliminan y podemos localizar su posición mediante la diferencia entre la imagen actual y la original. Si la placa tiene caracteres claros sobre un fondo oscuro, la erosión, que elimina los detalles claros de la imagen si el elemento estructurante tiene todos sus valores positivos [16], tendrá el mismo efecto sobre los caracteres que la dilatación en el caso anterior. La desventaja de este método es que no podemos saber de antemano cómo será la placa, así que es necesario aplicar ambos procesos.

Otros métodos reducen la información de la imagen ya sea mediante detección de bordes o binarización y después buscan rectángulos mediante la transformada de Hough, o se puede buscar regiones con zonas de píxeles 8-conectadas de tamaño y ubicación adecuados que puedan determinarse mediante algoritmos de optimización [17].

Finalmente, podemos aplicar correlación cruzada de imágenes, usando una imagen de placa como prototipo.

No siempre funciona buscar la placa buscando su forma. Las matrículas europeas son más alargadas que las mexicanas. Por tanto, la estrategia más efectiva para detectarlas es a través de la búsqueda de los caracteres.

El uso de operaciones morfológicas tiene la desventaja mencionada anteriormente. Otro camino que podemos tomar es encontrar una huella frecuencial específica de los caracteres de la placa, ya que notamos que los mismos tiene más cambios de nivel de gris que el resto del auto. Podemos intentar buscar esta huella usando correlación cruzada.

3.1.1 Emparejamiento de patrones (correlación).

Tenemos una subimagen *w(x,y)* de dimensiones *JxK* que queremos buscar en una imagen *f(x,y)* de dimensiones *MxN*, donde $J \leq M$ y $K \leq N$ [18].

Definimos que la imagen buscada se encuentra en la zona de *f(x,y)* donde el valor de la correlación es máxima:

$$c(s,t) = \sum_x \sum_y f(x,y)\, w(x-s, y-t) \quad , \tag{3.1}$$

que también puede escribirse como [19]:

$$c(s,t) = \sum_x \sum_y f(x+s, y+t)\, w(x,y) = f(x,y) \circ w(x,y) \quad , \tag{3.2}$$

donde $s=0,1,2,\ldots,M-1$, $t=0,1,2,\ldots,N-1$, y la sumatoria se toma sobre la región de la imagen superpuesta entre *f* y *w*. Es decir, desplazamos la subimagen *w(x,y)* sobre *f(x,y)* dándonos valores de *c* para cada *s,t* en *f(x,y)*. Suponemos que el origen de *f* está en la esquina superior izquierda y el origen de *w* está en su centro. Cuando *f* y *w* son la misma función, la operación resultante es una autocorrelación. Cuando la correlación se quiere hacer para señales finitas como las imágenes de *MxN*, las ec. 3.1 y 3.2 se multiplican por un factor *1/MN*, pero como no estamos interesados en el valor exacto de la correlación sino sólo en el valor máximo, podemos obviar este término.

La función de correlación dada en las ecuaciones anteriores tiene la desventaja de ser sensitiva a cambios en la amplitud de *f(x,y)* y *w(x,y)*. Una aproximación usada frecuentemente para evadir esta dificultad es usando el coeficente de correlación □:

$$\gamma(s,t) = \frac{\sum_x \sum_y \left[f(x,y) - \bar{f}(x,y) \right] \left[w(x-s, y-t) - \bar{w} \right]}{\left\{ \sum_x \sum_y \left[f(x,y) - \bar{f}(x,y) \right]^2 \sum_x \sum_y \left[w(x-s, y-t) - \bar{w} \right]^2 \right\}^{1/2}} \quad , \qquad (3.3)$$

donde $s=0,1,2,...,M-1$, $t=0,1,2,...,N-1$, $\bar{w}$ es el valor promedio de los píxeles en w, $\bar{f}$ es el valor promedio de f en la región coincidente con la ubicación actual de w, y las sumatorias son tomadas en la región de traslape. El coeficiente de correlación está escalado en el rango -1 a 1, independientemente de cambios de amplitud en las imágenes originales.

En caso de que hubiera cambios en rotación o escalamiento entre *w(x,y)* y la imagen que queremos encontrar en *f(x,y)*, se hace necesario encontrar una forma de normalización para estos cambios. Pero si el tamaño deseado o la rotación son desconocidos, requeriríamos calcular *c(s,t)* para diferentes tamaños y todas las rotaciones posibles, lo que es impráctico. Por eso, como ya se mencionó antes, nos restringiremos a imágenes sin rotación, aunque sería deseable que el algoritmo buscado tenga tolerancia al escalamiento.

La correlación cruzada también puede ser implementada en el dominio de las frecuencias usando la transformada de Fourier. Si *f* y *w* son de tamaños parecidos, la aproximación usando transformadas de Fourier es más eficiente que la implementación directa. Específicamente, si *w* tiene una resolución menor a 132 (una subimagen de *12x11*) la implementación directa de la ecuación (3.1) es más eficiente que la aproximación de

Fourier [20]. Nuestras imágenes f son de *640x480*, donde el tamaño del área de caracteres de la placa es de aproximadamente *100x26*, por tanto, es mejor implementar este método en el dominio de la frecuencia.

Si usamos la transformada de Fourier de las imágenes, la correlación toma la siguiente forma:

$$f(x,y) \circ g(x,y) \overset{F}{\Leftrightarrow} F^*(u,v)G(u,v) \ , \tag{3.4}$$

es decir, la operación de correlación entre dos señales se realiza obteniendo las trasformadas de Fourier de las señales originales, obteniendo el complejo conjugado de una de ellas y multiplicando las señales así obtenidas. Finalmente aplicamos la transformada inversa de Fourier para obtener el resultado en el dominio espacial.

Es necesario aclarar que la correlación no es conmutativa:

$$f(x,y) \circ g(x,y) \neq g(x,y) \circ f(x,y) \ , \tag{3.5}$$

La ec. 3.5 se demuestra si regresamos a la definición de correlación. Denotamos la función de correlación entre f y g como:

$$c_{fg}(s,t) = f \circ g = \sum_s \sum_t f(x+s, y+t)\, g(s,t) = \sum_s \sum_t f(x,y)\, g(x-s, y-t) \ , \tag{3.6}$$

y el cálculo de la correlación entre g y f es:

$$c_{gf}(s,t) = (g \circ f) = \sum_s \sum_t g(x+s, y+t)\, f(s,t) = \sum_s \sum_t g(x,y)\, f(x-s, y-t) \, , \tag{3.7}$$

A partir de las ecuaciones 3.6 y 3.7 podemos concluir que la igualdad correcta para la conmutación de funciones en la correlación cruzada bidimensional es:

$$c_{fg}(s,t) = c_{gf}(-s,-t) \ , \tag{3.8}$$

Usando la notación de las ec. 3.6 a 3.8, la ec. 3.1 que nos indica cómo se realiza el emparejamiento de patrones mediante correlación se expresa $c_{fg}(s,t)$.

Existen similitudes entre el cálculo de la correlación y la convolución [21]. En el cálculo de la convolución una de las señales se refleja, se desplaza y finalmente se multiplica por otra secuencia; finalmente se suman todos los valores de la secuencia producto:

$$f(x,y) * g(x,y) = \frac{1}{MN} \sum_m \sum_n f(m,n) g(x-m, y-n) \ , \tag{3.9}$$

Con excepción de la operación de reflexión, el cálculo de la correlación supone exactamente las mismas operaciones. Si tenemos un programa para el cálculo de la convolución, podemos usarlo para realizar la correlación proporcionando como entradas las funciones reales $f(x,y)$ y la secuencia reflejada de $g(x,y)$, es decir:

$$c_{fg}(s,t) = f(x,y) * g(-x,-y) \ , \tag{3.10}$$

La convolución también puede implementarse en el dominio de la frecuencia:

$$f(x,y) * g(x,y) \overset{F}{\Leftrightarrow} F(u,v) G(u,v) \ , \tag{3.11}$$

Por tanto, podemos convertir el problema de la correlación en un problema de convolución. Invertimos la imagen *w(x,y)* sobre su origen, obtenemos las transformadas de Fourier de f(x,y) y de w(-x,-y), las multiplicamos y finalmente calculamos la transformada inversa del resultado de la multiplicación. Como este proceso lo hacemos tomando como base la ec. 3.1 y no el coeficiente de correlación de la ec. 3.3, nuestro resultado no está normalizado en amplitud.

3.1.2 Aplicación del algoritmo de detección de placa.

El algoritmo de correlación consta de los siguientes pasos:

1) Obtener las líneas verticales de la imagen original.
2) Realizar la integración sobre esta imagen con una ventana de 3X13.
3) Obtener la transformada de Fourier de la imagen obtenida.
4) Multiplicar esta transformada por la transformada de Fourier de *w(-x,-y).*
5) Antitransformar la imagen resultante. Obtener su magnitud.

3.1.2.1 Obtención de líneas verticales.

El primer paso nos permite eliminar información no relevante. Usar la imagen en escala de grises tiene la desventaja de que las placas podrían ser oscuras con caracteres claros, o claras con caracteres oscuros, lo que nos obligaría a realizar la correlación con dos patrones diferentes. Podemos eliminar este problema si sólo operamos sobre una imagen de bordes de la imagen y no hay pérdida de información importante pues la forma de la placa se mantiene.

Antes de operar sobre la imagen podría parecer conveniente eliminar el ruido de la imagen que introduce la cámara. Lo anterior se hace ya sea convolucionando la imagen con un filtro de media, con un filtro gaussiano o con un filtro de mediana; la elección del filtro depende del tipo de ruido que aparezca en la imagen.

La eliminación del ruido es opcional. Aunque es conveniente realizarlo antes de aplicar un algoritmo de eliminación de bordes para evitar ruido en el resultado, existen imágenes con mala iluminación o bajo contraste (fig. 3.1 y 3.2) donde eliminar el ruido puede difuminar los caracteres de la placa, dificultando su detección.

Aún así, si en el proceso de captura de la imagen introduce demasiado ruido cuando extraemos los bordes, será necesario filtrar la imagen usando la matriz de media, y/o la mediana. Suponiendo que el ruido sea aditivo, puede describirse usando ya sea una distribución gaussiana o la distribución de Laplace . Si barremos la imagen con una ventana de dimensiones *NxN* siendo *N* impar, el problema se reduce a estimar el mejor valor para el píxel central, usando la información de la intensidad de los píxeles de la vecindad. Si el ruido es gaussiano, el mejor estimador es la media (promedio); para una distribución de Laplace, el mejor estimador es la mediana de la muestra [22].

Para el problema que nos ocupa, lo mejor es usar un filtro de mediana, pues respeta mejor los bordes que el filtro de media.

Los bordes son regiones de la imagen donde hay transición de niveles de grises [23]. Estas regiones pueden detectarse usando operadores de primera y segunda derivada. La primera derivada es cero excepto en regiones con cambio de niveles de gris; la segunda derivada en cero en todos lados excepto en el comienzo y final de la transición de intensidad (Fig. 3.3). Por tanto, un cambio de intensidad de manifiesta en un cambio brusco en la primera derivada y presenta un paso por cero en la segunda derivada.

Fig. 3.1 Imagen de auto con bajo contraste en la zona de la placa.

Fig. 3.2 Acercamiento a la placa de la fig. 3.1.

De los operadores basados en primera derivada los más utilizados son los de gradiente (Sobel, Prewitt, Roberts, Kirsch, Robinson, Frei-Chen).

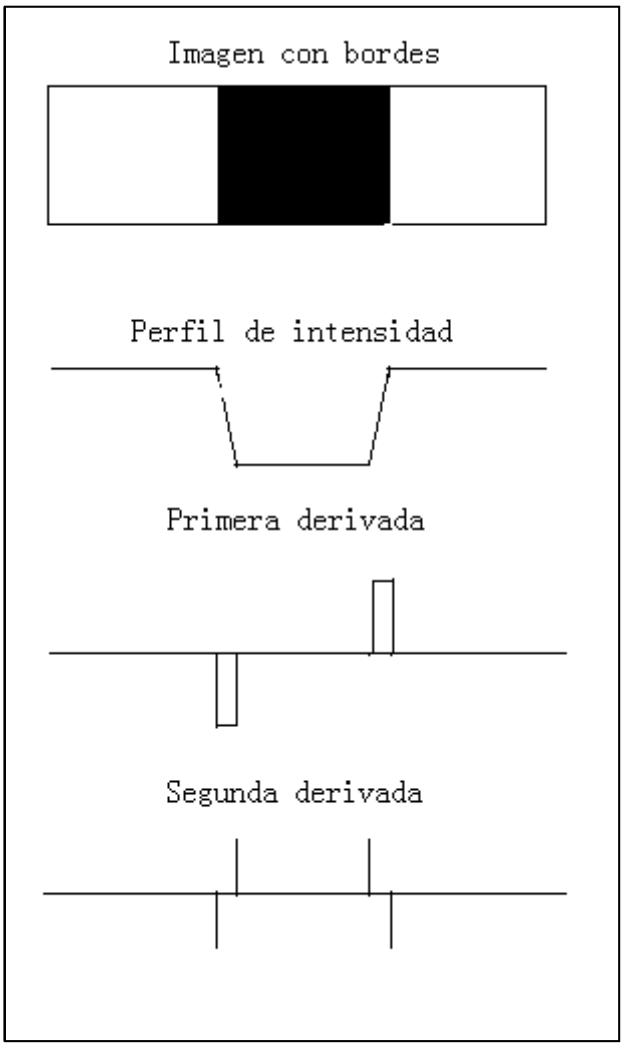


Fig. 3.3. Concepto de primera y segunda derivada para la extracción de bordes.

El gradiente de una imagen *f(x,y)* en un punto *(x,y)* se define como un vector bidimensional dado por la ecuación 3.12:

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix} , \tag{3.12}$$

El gradiente produce un vector que apunta en la dirección de variación máxima de f para cada punto (x,y). Al ser un vector, podemos representarlo como magnitud y fase dadas por:

$$\left|\nabla f(x,y)\right| = \sqrt{f_x^2 + f_y^2} , \tag{3.13}$$

$$\phi(x,y) = \arctan\frac{f_y}{f_x} , \tag{3.14}$$

donde f_x y f_y son las derivadas parciales con respecto a x e y mostradas en la ecuación 3.12 Un punto de borde es aquel que, cuando calculamos la magnitud del gradiente en él, supere cierto umbral; esta operación es idéntica a la binarización que vimos en el capítulo anterior.

Los valores de f_x y f_y pueden aproximarse por convolución con las máscaras 3x3 dadas en las ecuaciones 3.15 y 3.16, conocidas como los operadores de Sobel. Los operadores gradiente tienen el efecto de magnificar el ruido subyacente en la imagen, pero los operadores de Sobel tienen la propiedad añadida de suavizar la imagen, minimizando la aparición de falsos bordes [24]. La propiedad anterior hace aparecer a los operadores de Sobel deseables para la aplicación que estamos desarrollando.

$$f_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} , \tag{3.15}$$

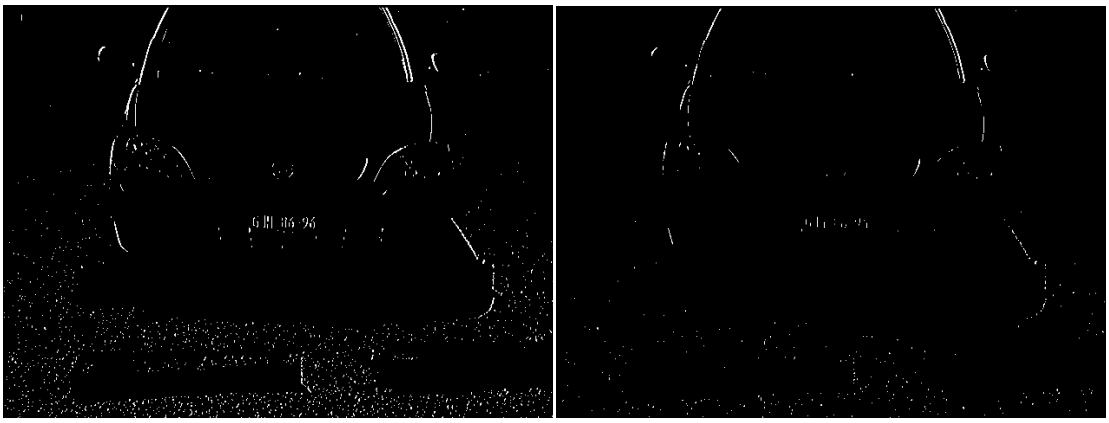
$$f_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} , \tag{3.16}$$

Si queremos localizar los bordes verticales, convolucionamos la imagen con la máscara de la ec. 3.15, obtenemos el valor absoluto del gradiente y aplicamos la binarización.

El umbral adecuado para poder eliminar el ruido y dejar sólo los bordes reales es variable; depende de factores como el método de obtención de las imágenes o la presencia de ruido, y es en este punto donde los operadores de Sobel muestran sus deficiencias.

Si partimos de la fig. 3.4 y aplicamos el operador de Sobel, obtenemos la imagen de la figura 3.5 (a), usando un umbral de 200. Como se puede ver, aún aparecen puntos producidos por la grava. Si intentamos subir el umbral para eliminar este ruido, los bordes de la placa se pierden 3.5 (b).

Fig. 3.4 Imagen frontal de un auto.



a) Umbral = 200. b) Umbral = 300.

Fig. 3.5 Imágenes de bordes de la fig. 3.4.

Seria deseable encontrar un operador que pudiera eliminar los puntos producidos por la grava, al mismo tiempo que respeta las líneas verticales de la placa. Con esto en mente definimos la matriz de la ec. 3.17. En cada renglón tenemos la suma de la aproximación discreta de la primera derivada, y la aproximación discreta de la primera derivada del siguiente punto con el signo invertido. Si tenemos un punto de nivel de gris diferente al de su entorno, éste será eliminado ya que ambas derivadas serán parecidas, pero seguirá respetando los bordes entre zonas con niveles de gris diferentes; si el píxel está en una zona clara y está junto a una zona oscura, el resultado será mayor a cero, en caso contrario, el resultado será menor a cero.

$$M_x = \begin{bmatrix} 1 & -2 & 1 \\ 1 & -2 & 1 \\ 1 & -2 & 1 \\ 2 & -4 & 2 \\ 1 & -2 & 1 \\ 1 & -2 & 1 \\ 1 & -2 & 1 \end{bmatrix} , \qquad (3.17)$$

Usando esta matriz, tenemos un rango más amplio para jugar con los umbrales. Empíricamente determinamos que el umbral adecuado puede fijarse tomando en cuenta cuál es el valor máximo que obtenemos al aplicar la convolución de M_x sobre *f(x,y)* (Tabla 3.1).

Valor del Máximo	Umbral
Máximo>1000	450
1000>Máximo>190	70
Máximo< 190	20

Tabla 3.1. Umbrales para la imagen de bordes obtenidas con M_x.

Usando M_x y aplicando un umbral de 450 (el máximo fue de 2508) obtenemos la imagen de la figura 3.6. Nótese como las líneas de la placa siguen visibles, mientras el perfil del auto está menos definido.

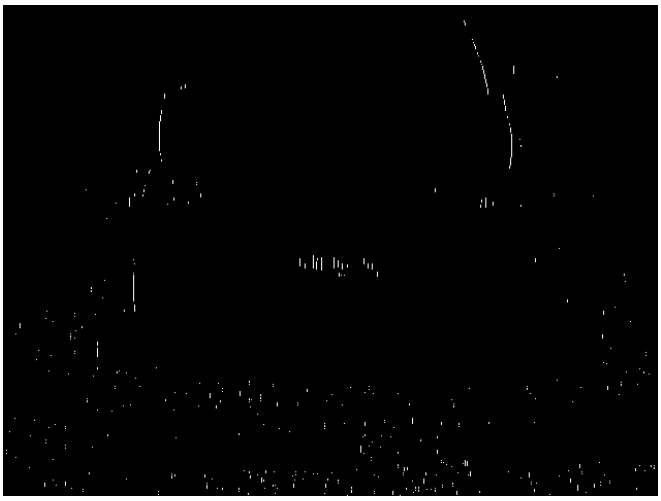


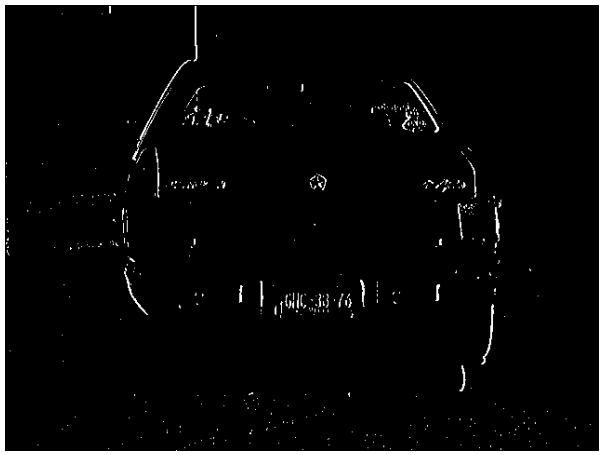
Fig. 3.6 Líneas verticales usando M_x.

3.1.2.2 Integración de la imagen y correlación en Fourier.

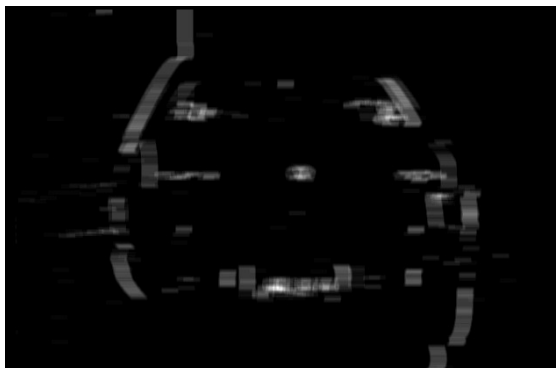
Una vez que hemos obtenido la imagen de bordes de la imagen, es necesario filtrarla, para eliminar los píxeles aislados y lograr que la imagen que obtengamos al finalizar el proceso sea una región sin huecos, cuya dimensión y posición sean las mismas de las de la placa.

La matriz de convolución que usaremos es de *13x3*, definida como *Me(x,y) = 1*, con origen en el centro de la matriz. En la figura 3.7 (c) tenemos el resultado de la convolución con la matriz de media que hemos definido, de la imagen de bordes de la figura 3.7(a).

a) Imagen original.



b) Imagen de líneas verticales.



c) Resultado de la convolución con *Me(x,y).*

Fig. 3.7

El siguiente paso es realizar la convolución entre *f(x,y)*, que en este caso es la imagen de la figura 3.7 (c), y *w(-x,-y)*, que sería la inversión de la imagen de la figura 3.8.

La transformada de Fourier discreta para señales finitas se define de la siguiente manera:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-j2\pi\left(ux/M + vy/N\right)\right] \ , \qquad (3.18)$$

Fig. 3.8 Imagen de $w(x,y)$.

y la antitransformada se define como:

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp\left[j2\pi\left(ux/M + vy/N\right)\right] \ , \tag{3.19}$$

Se necesitan muchos recursos de cómputo para calcular la transformada y la antitransformada de una imagen. Pero aún así podemos reacomodar los términos de la ec. 3.18 para aumentar la velocidad del proceso:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\exp\left(-j2\pi\frac{ux}{M}\right)\sum_{y=0}^{N-1} f(x,y)\exp\left(-j2\pi\frac{vy}{N}\right) , \tag{3.20}$$

Esto significa que la transformada de Fourier es separable, podemos calcular primero las transformadas unidimensionales de los renglones de la imagen, y a partir de esta imagen intermedia, calcular la transformada unidimensional de las columnas. También podemos intercambiar el orden de las operaciones.

Aún aplicando la separabilidad de la Transformada de Fourier, el proceso sigue siendo lento. Pero si las dimensiones de la imagen son potencias enteras de dos, podemos

aplicar el algoritmo de la Transformada Rápida de Fourier(FFT, Fast Fourier Transform). En [25] se explica la implementación de este algoritmo.

El resultado de la correlación puede verse en la figura 3.9. Comparando esta imagen con la imagen original en escala de grises, notamos que nuestro filtro respondió a la placa y a las calcomanías pegadas sobre el auto. Sin embargo, si nos fijamos en la zona con la respuesta más alta, la placa respondió con mayor fuerza (fig. 3.10).

Fig. 3.9 Resultado de la correlación.

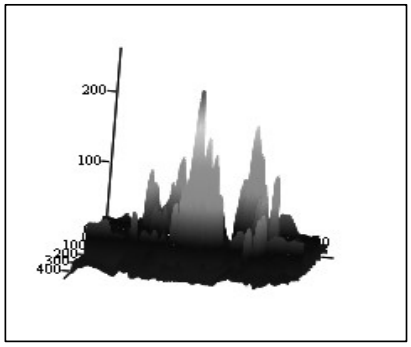


Fig. 3.10 Gráfica de la respuesta de la correlación al filtro.

Una vez con este resultado, corregimos el rango dinámico de la imagen para ajustarlo entre 0 y 255. Buscamos el máximo y el mínimo de la imagen, y aplicamos la ec. 3.21:

$$g(x,y) = \left(f(x,y) - \min(f)\right)\frac{255}{\max(f) - \min(f)} \quad , \tag{3.21}$$

Fig. 3.11 Imagen binaria mostrando posibles ubicaciones de la placa.

Finalmente, binarizamos usando un umbral de 90, obteniendo la imagen de la figura 3.11. A partir de aquí es fácil localizar los límites de la placa, sólo debemos localizar a qué mancha corresponde el máximo de la fig. 3.10. Si la mancha es continua es decir, si no tiene huecos, podemos hallar los límites de la placa contando cuántos píxeles, hacia arriba, abajo, izquierda y derecha, faltan para llegar al borde de la imagen. Si conocemos las coordenadas del punto máximo, solo debemos sumar o restar los valores que vamos obteniendo, y tenemos las coordenadas de las esquinas de un rectángulo que contiene a la placa (fig. 3.12).

3.1.2.3 Corrección de la imagen binarizada.

Si la imagen binarizada contiene "rasguños" en las zonas de alto nivel de gris, podemos resolver esta situación usando un proceso morfológico para cerrar los huecos.

Fig. 3.12 Placa del auto de la fig. 3.7 (a).

El proceso morfológico que queremos no debe modificar las formas de las zonas, sino sólo cerrar los huecos internos.

Operación lógica	Ejemplo
Conjunción	ab, (a)(b), a·b
Disyunción	a+b
Negación	$\bar{a}$,a'
Identidad	a = b

Tabla 3.2. Representación de operadores lógicos en el álgebra de Boole.

Recordemos que el índice *i* se asocia a la coordenada *y*, el índice *j* se asocia a *x*. Aplicamos las siguientes relaciones lógicas (usaremos la notación del álgebra de Boole que aparece en la tabla 3.2):

$$g(i,\ j) = \left(\overline{f(i,\ j)}\right)\left(\sum_{k=-5}^{5} f(i,\ j+k)\right)\ , \qquad (3.22)$$

$$g(i,\ j) = \left(\overline{f(i,\ j)}\right)\left(\sum_{k=-5}^{5} f(i+k,\ j)\right) \quad , \tag{3.23}$$

aplicadas en ese orden. Los resultados de aplicar estas fórmulas pueden verse en la figura 3.13. Para mayor información sobre algoritmos morfológicos consultar [26].

La técnica de correlación tiene una efectividad del 80%, en los cuáles podemos encontrar la placa y recortarla completa. Aún así, este porcentaje es muy bajo si queremos que nuestro sistema se use en aplicaciones delicadas como el control del tráfico; si estuviera en un cruce donde pasen tres mil autos por hora, en seiscientos autos no podría identificar la posición de la placa.

a) Imagen binaria con huecos en las regiones.

b) Resultado de la operación de cerrado de huecos.

Fig. 3.13

3.1.2.4 Esquema modificado del sistema.

Al esquema de la figura 1.3 nos pareció pertinente hacerle una modificación. Nuestra propuesta es agregar una retroalimentación a la salida del sistema, justo cuando se obtiene la cadena de caracteres ASCII (American Standard Code for Information Interchange). En este momento podemos decidir si el área es una placa o no usando varios métodos. Uno es comparando la cadena obtenida con el formato que esperábamos obtener, aunque esto tiene la desventaja de limitar el universo de imágenes que podemos tratar con nuestro sistema, pues tenemos que implementar sistemas de decisión diferente por cada formato que queramos incluir.

Otro sistema considerado implica hacer uso de las salidas de la red que usamos para reconocer los caracteres. Tomando en cuenta el valor de salida de cada caracter obtenemos una señal de energía asociada a la cadena ASCII, que nos permite diferenciar las placas de otras áreas que sean semejantes en frecuencia o en histograma, mediante la selección de un umbral adecuado.

Tomando en cuenta esta modificación, el sistema queda como se muestra en la figura 3.14.

Pero esto nos crea un nuevo problema. Ahora debemos contar las zonas que tenemos en nuestra imagen, y clasificarlas por prioridad. La clasificación no tiene problema; una vez que hayamos descartado la zona con la mayor respuesta, sólo debemos ir a la siguiente zona con la mayor respuesta.

3.1.2.5 Etiquetado de regiones conexas.

El conteo de zonas puede hacerse etiquetándolas. Existen muchos algoritmos de etiquetado de componentes conexas. Estos algoritmos lo que buscan es agrupar píxeles de la misma región asignándoles la misma etiqueta.

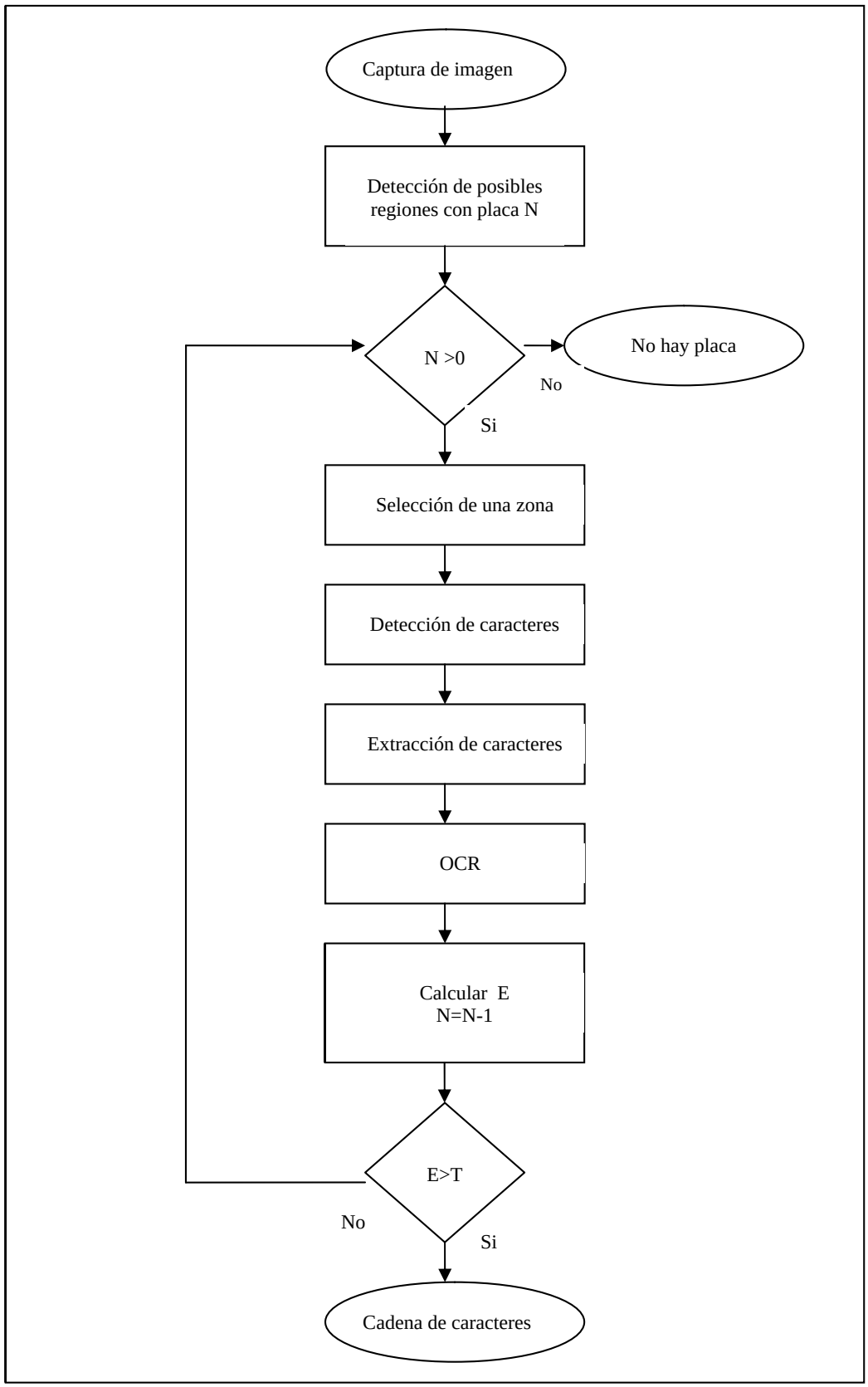


Fig. 3.14 Diagrama de flujo del algoritmo propuesto.

En este caso usamos un algoritmo iterativo consistente en tres pasos [27]:

- Etiquetado de todos los píxeles distintos de cero.
- Propagación de etiquetas de arriba-abajo y de izquierda-derecha.
- Propagación de etiquetas abajo-arriba e izquierda-derecha.

En el primer paso, etiquetamos todos los píxeles distintos de cero con números consecutivos. En el segundo paso revisamos la vecindad de cada píxel, tomando en cuenta que estén 8-conectados; si el píxel es diferente de cero, le asignamos el menor valor diferente de cero que encontremos en la región. El segundo paso lo ejecutamos píxel por píxel de arriba abajo y de izquierda a derecha. Finalmente, realizamos la misma propagación de etiquetas pero ahora desde abajo. En la fig. 3.15 encontramos un ejemplo de la aplicación de este algoritmo. Aunque puede usarse las etiquetas para encontrar los límites de las regiones, es más rápido contar los píxeles sobre los ejes vertical y horizontal desde el punto del máximo.

Ya que hemos etiquetado cada región, sólo es cuestión de contar cuantos valores diferentes de cero presenta la imagen. Este número, que llamaremos *N*, nos indica el número total de regiones donde pudiera estar la placa. En caso de que en la primera región no se encontrara la placa, disminuimos *N* en 1 y continuamos con la región con el segundo valor más alto en la imagen resultado de la correlación. Si ya no quedan regiones, consideramos que la imagen no contenía una placa.

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 2 & 0 & 3 & 4 & 0 \\ 0 & 5 & 6 & 0 & 7 & 8 & 0 \\ 0 & 9 & 10 & 11 & 12 & 13 & 0 \end{bmatrix}$$

a) Región conexa. b) Inicialización.

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 3 & 3 & 0 \\ 0 & 1 & 1 & 0 & 3 & 3 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

c) Propagación arriba-abajo. d) Propagación abajo-arriba

Fig. 3.15 Algoritmo iterativo para etiquetado de componentes conexas.

3.1.2.6 Algoritmo final de detección de placa.

Hemos visto cómo el algoritmo de correlación logra encontrar eficazmente la placa, y ya vimos un método para corregir su dificultad en detectar el 20% de las placas. Pero aún así sería deseable un método que no requiriera cambiar de dominio.

Si consideramos que la placa debe ser una región con una mayor cantidad de cambios de nivel de gris, podemos suponer que al obtener la imagen de bordes verticales la zona de la placa sobresaldrá con respecto al resto de la imagen.

En esta fase, nuestro objetivo es identificar las regiones donde podría estar la placa. Entonces, si sumamos todos los valores de los píxeles de una región debemos de obtener un valor máximo en el área de la placa. El tamaño de la región que debemos usar fue determinado empíricamente, de tamaño *101x27*. Este tamaño es el óptimo para imágenes de *640x480*; para imágenes de tamaño *MxN* podemos definir la siguiente relación:

$$m = 37\frac{M}{N}, \qquad \mathrm{n} = 10\frac{\mathrm{M}}{\mathrm{N}}, \tag{3.24}$$

Esta operación es semejante a una convolución con un filtro de media del tamaño definido anteriormente. Esta operación se representa como:

$$g(x, y) = \sum_{s=-(m-1)/2}^{(m-1)/2} \sum_{t=-(n-1)/2}^{(n-1)/2} f(x+s, y+t) \ , \tag{3.25}$$

Aunque esta operación es grande, indicando un proceso lento, la velocidad puede incrementarse si nos damos cuenta que la ec. 3. 25 también es separable, al igual que la Transformada de Fourier. Podemos escribir:

$$g(x,y) = \sum_{s=-(m-1)/2}^{(m-1)/2} \left(\sum_{t=-(n-1)/2}^{(n-1)/2} f(x+s, y+t) \right), \tag{3.26}$$

De esta manera, la operación de sumatoria sobre una región puede partirse y ejecutarse primero la suma sobre las columnas y después sobre las filas, o viceversa. La desventaja de este método es que disminuye el área efectiva que podemos aprovechar de una imagen, aunque si el auto está rodeado por un ambiente muy ruidoso, la eliminación de los bordes nos beneficiaría. De todas maneras debemos procurar que la placa se ubique al centro de la imagen.

3.2 Conclusiones.

En definitiva, los pasos del algoritmo para la detección de la placa son los siguientes:

1) Obtención de la imagen de bordes verticales, usando la ec. 3.17.
2) Convolución con el filtro de media de tamaño *mxn (101x27).*

3) Ajustar el rango dinámico de la imagen obtenida al rango 0-255. Guardar esta imagen.
4) Binarizar la imagen con un umbral de 90 (valor empírico).
5) Cerrar los huecos en las zonas.
6) Contar regiones.
7) Si el algoritmo regresó de un proceso anterior, eliminar la región ya revisada en la imagen guardada en el paso 3.
8) Si existen regiones, usar la imagen guardada anteriormente para localizar la posición del máximo.
9) Obtener las coordenadas de las esquinas de la imagen de la placa contando distancias a los bordes.

Este algoritmo modificado puede encontrar la placa en un 85% de lo casos al primer intento.

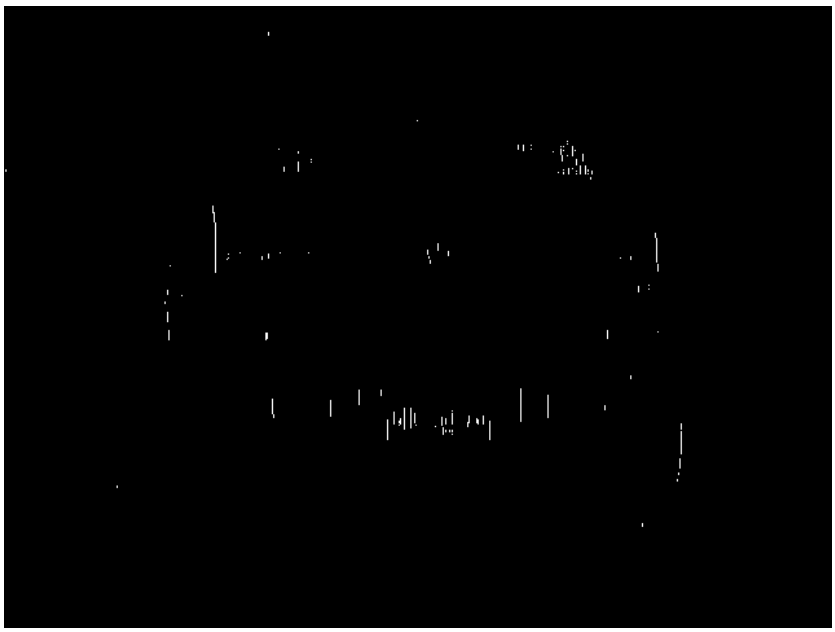


Fig. 3.16 Imagen de bordes de 3.7a usando M_x.

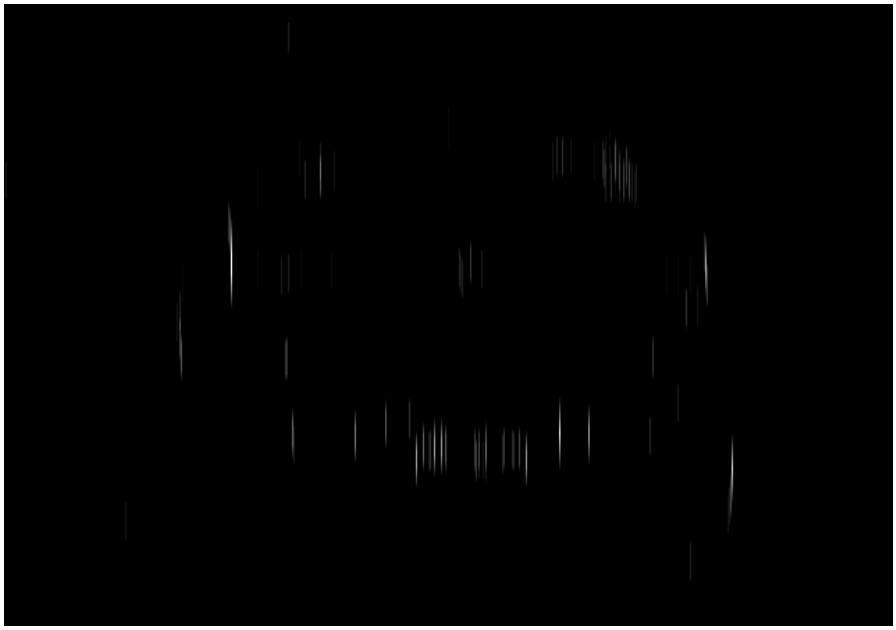


Fig. 3.17 Integración en *y* usando ec. 3.26.

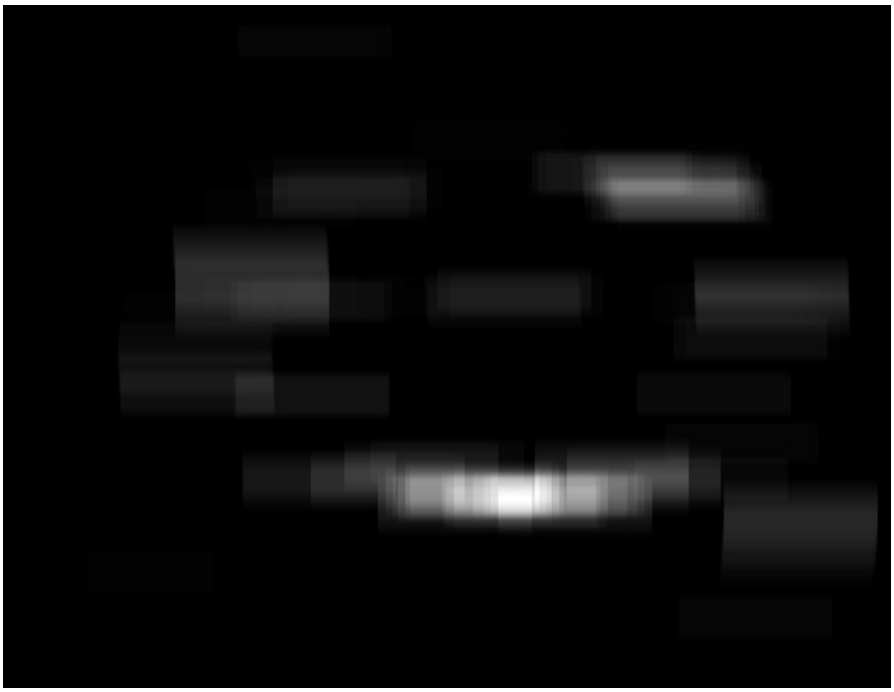


Fig. 3.18 Resultado de las sumatorias de la ec. 3.26.

CAPÍTULO 4

Detección y segmentación de los caracteres

4.1 Segmentación de caracteres.

En esta sección veremos los algoritmos necesarios para la detección del área específica de la placa donde se encuentran los caracteres. Este paso es necesario ya que en la mayoría de las placas los datos que nos interesan no vienen solos. Además, debido a problemas de iluminación, de suciedad de las placas, se hace necesario un proceso antes de la extracción de los caracteres.

4.1.1 Extracción de la imagen de la placa de la imagen original.

El proceso que explicamos en el capítulo 3 nos ha dado, como resultado final, las coordenadas de las esquinas de una imagen que posiblemente tenga la placa. Usaremos esta información para ampliar en una imagen de 300 columnas por *N* filas, *300xN*, donde *N* dependerá de las proporciones de la región que hemos recortado. La explicación del algoritmo de extracción de imágenes es importante pues lo necesitaremos utilizar al momento de extraer los caracteres ya segmentados.

El algoritmo que queremos describir se inscribe en el tema de las transformaciones geométricas. Estas transformaciones incluyen operaciones como magnificación o reducción de una imagen, rotación o traslación. Podemos interpretar una transformación geométrica, en forma general, como una transformación de un sistema de coordenadas a otro (ec. 4.1).

$$T[(x, y)] = (x', y') \ , \tag{4.1}$$

Podemos representar varios tipos de transformaciones de coordenadas mediante operaciones matriciales (traslación, rotación, cambio de escala). Si suponemos que las coordenadas (x,y) y (x',y') pueden representarse como vectores ***V*** y ***V'***, el operador que transforma de un sistema coordenado a otro se representa con una matriz ***T***:

$$V' = TV, \qquad V' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \quad \mathrm{V} = \begin{bmatrix} \mathrm{x} \\ \mathrm{y} \end{bmatrix} \ , \tag{4.2}$$

Así mismo podemos realizar la operación inversa para obtener ***V*** a partir de ***V'***, utilizando $\boldsymbol{T^{-1}}$, la matriz inversa de ***T***.

$$V = T^{-1}V', \tag{4.3}$$

Tenemos una imagen fuente *f(x,y)* de *MxN*, de donde queremos extraer cierta región de interés y mostrarla en otra función *g(x',y')* de *mxn.* La mejor forma de realizar la trasformación es verificar, para cada coordenada *(x',y')*, a qué coordenada *(x,y)* corresponde, es decir, aplicar un operador inverso T^{-1} sobre *(x',y');* de esta manera, para cada punto en *g* podemos asociarle un valor de *f* y evitamos dejar espacios vacíos. Pero ya que ambas funciones son discretas, no existen valores de intensidad en valores fraccionarios de coordenadas.

Como no existe garantía que al hacer la transformación de coordenadas no obtengamos un valor fraccionario, debemos encontrar una manera de aproximar el valor de intensidad que podría corresponder a ese punto, suponiendo que exista continuidad entre píxeles adyacentes.

La interpolación bilineal nos da la mejor aproximación. En la fig. 4.1 tenemos un esquema gráfico de la interpolación bilineal. Definimos los siguientes factores de ponderación:

$$a_1 = \left(1 - \frac{dx}{\Delta x}\right)\left(1 - \frac{dy}{\Delta y}\right) , \tag{4.4}$$

$$a_2 = \frac{dx}{\Delta x}\left(1 - \frac{dy}{\Delta y}\right) , \tag{4.5}$$

$$a_3 = \left(1 - \frac{dx}{\Delta x}\right)\frac{dy}{\Delta y} , \tag{4.6}$$

$$a_4 = \frac{dx}{\Delta x}\frac{dy}{\Delta y} , \tag{4.7}$$

Para el caso que estamos tratando $\Delta x, \Delta y = 1$. El valor del píxel interpolado queda:

$$p(x, y) = a_1 p(i, j) + a_2 p(i, j+1) + a_3 p(i+1, j) + a_4 p(i+1, j+1) , \tag{4.8}$$

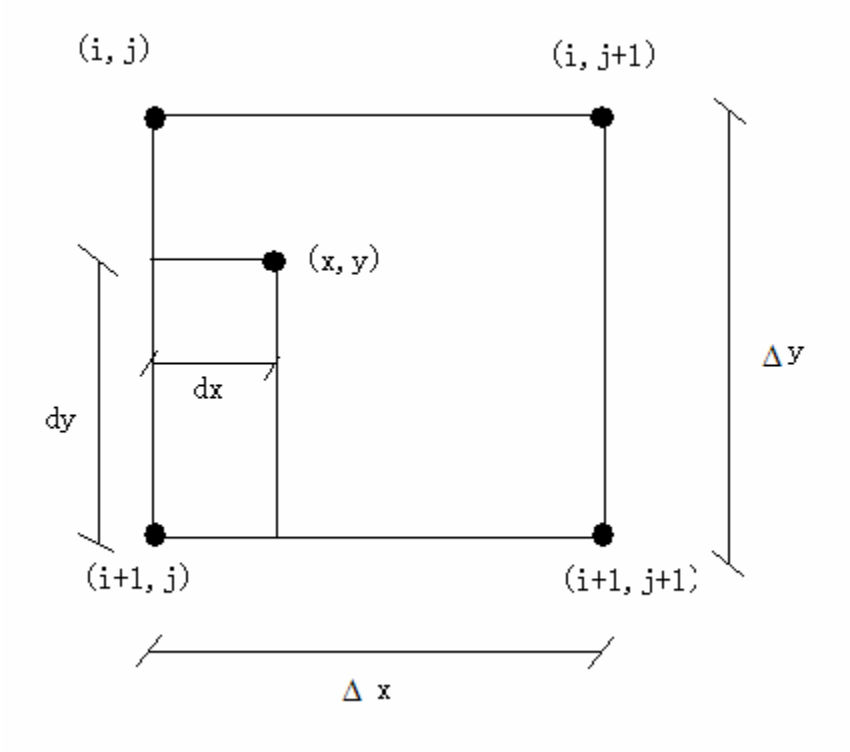


Fig. 4.1. Esquema de la interpolación bilineal.

Si la sección que queremos extraer tiene como coordenada superior izquierda (x_1, y_1) y la esquina inferior derecha es (x_2, y_2), la relación de transformación desde (x',y') a (x,y) es:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x1 & \frac{x2 - x1}{m-1} & 0 \\ y1 & 0 & \frac{y2 - y1}{n-1} \end{bmatrix} \begin{bmatrix} 1 \\ x' \\ y' \end{bmatrix}, \tag{4.9}$$

En este caso queremos que la imagen extraída ocupe todo el espacio disponible en $g(x',y')$. Por tanto, la ec. 4.9 representa una traslación del origen de coordenadas y un cambio de escala.

La operación mostrada en la ec. 4.9 tiene el problema de que si queremos invertir $\boldsymbol{T}$ obtenemos un sistema con mayor número de incógnitas que ecuaciones disponibles. La

situación anterior surgió al tener que modificar ***V'*** para poder realizar la suma del término constante. En base a lo anterior, podemos modificar la ec. 4.9 para que ***T*** pueda invertirse sin dificultades:

$$\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ x1 & \dfrac{x2 - x1}{m - 1} & 0 \\ y1 & 0 & \dfrac{y2 - y1}{n - 1} \end{bmatrix} \begin{bmatrix} 1 \\ x' \\ y' \end{bmatrix} , \qquad (4.10)$$

En la figura 4.2b tenemos un ejemplo de extracción de placa. La imagen de la placa se extrajo a partir de la fig. 4.2a .

a)

b)

Fig. 4.2 (a) Placa encontrada en (b).

4.1.2 Detección de caracteres.

En esta sección nos interesa la manera de delimitar específicamente la zona que presenta la tira de caracteres. Observemos la figura 4.3, en donde se muestra el resultado de calcular f_x de la figura 4.2b, usando los operadores de Sobel y adecuando el rango dinámico entre 0 y 255 niveles de gris.

Fig. 4.3 f_x obtenido a partir de la figura 4.2.

Calculando el valor absoluto de cada píxel de la fig. 4.3, y ajustando su rango dinámico, obtenemos la fig. 4.4.

Fig. 4.4 Valor absoluto de la imagen 4.3.

Este es el resultado que esperábamos. Esta imagen nos servirá para calcular los límites superior e inferior de la tira de caracteres.

Si la imagen 4.4 tiene dimensiones $m \times n$, podemos definir un vector $\boldsymbol{Vy}$ de n elementos. Si sumamos todos los valores de una fila y guardamos el resultado en el elemento correspondiente de $\boldsymbol{Vy}$ (ec. 4.11) obtendremos el perfil de la fig. 4.5.

$$Vy(i) = \sum_{j=0}^{m-1} M(i, j) \quad , \tag{4.11}$$

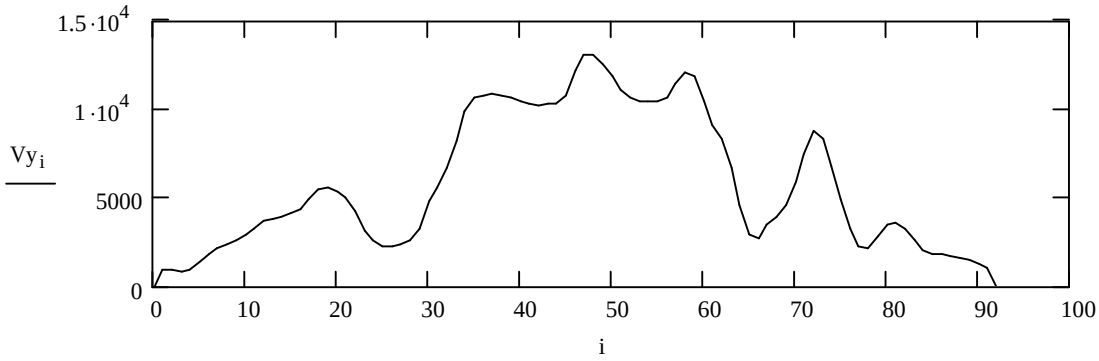


Fig. 4.5. Perfil vertical de la fig.4.4.

Podemos observar que esta función tiene cuatro zonas principales. La primera zona corresponde a las filas donde está el nombre del estado; la segunda zona corresponde al área de caracteres; la tercera zona corresponde al nombre del país (“México”). Vemos que estas zonas son fácilmente separables eligiendo un umbral adecuado; empíricamente determinamos que este umbral está en el 38% del valor máximo alcanzado por ***Vy***. Aplicando este umbral, obtenemos el perfil mostrado en la figura 4.6, denotado como ***Vy2***:

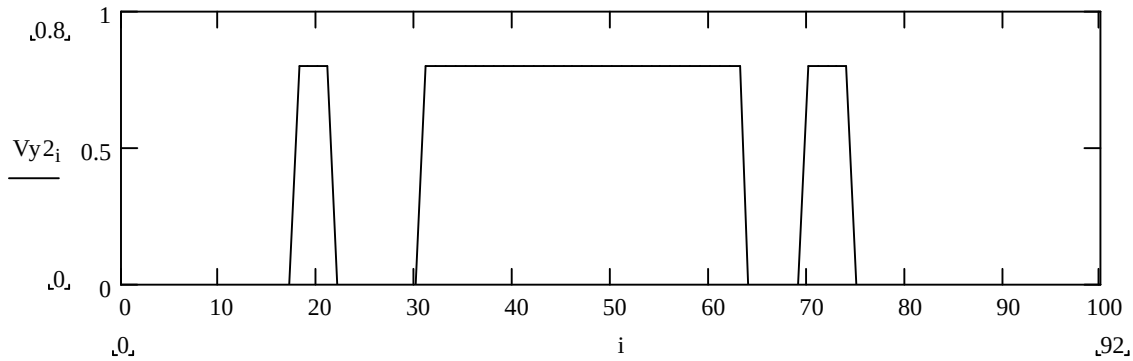


Fig. 4.6. Zonas de interés de la proyección vertical de la placa.

Observando la figura 4.2, podemos suponer que la zona donde están los caracteres que queremos recortar corresponde a la región más grande de ***Vy2***. Mediante un algoritmo

de etiquetado de regiones conexas y evaluando el tamaño de las regiones, podemos descartar las regiones más pequeñas. La figura 4.7 muestra el resultado de esta operación.

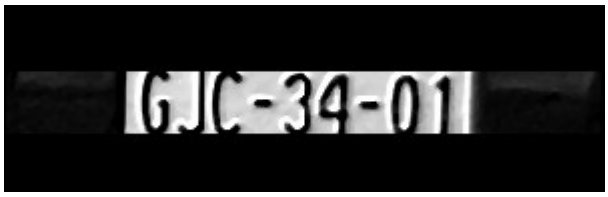


Fig. 4.7. Imagen de placa recortada en vertical.

La efectividad de este método depende de que la placa no esté girada, aunque tiene alguna tolerancia al giro (fig. 4.8), que aumenta si la imagen es de alta resolución.

a) Placa de auto inclinada.

b) Recorte de la zona de caracteres.

Fig. 4.8

Ahora debemos delimitar el área de los caracteres en el eje *X*. Para lograr esto seguiremos un procedimiento similar al usado para encontrar ***Vy2***. La imagen de partida será diferente; f_x ya no es suficiente, pues parte de la información de los bordes horizontales se pierde. Partiendo de la magnitud del gradiente de la figura 4.2, y recortando el área de caracteres en vertical, obtenemos la imagen de la figura 4.9.

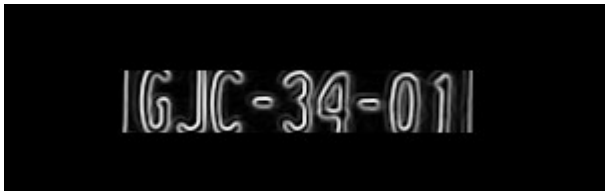


Fig. 4.9 Magnitud del gradiente de la fig. 4.8 (b).

Ahora obtendremos el vector ***Vx***, que contiene la proyección de esta imagen en el eje *X*:

$$Vx(j) = \sum_{i=0}^{n-1} M(i, j) \quad , \tag{4.12}$$

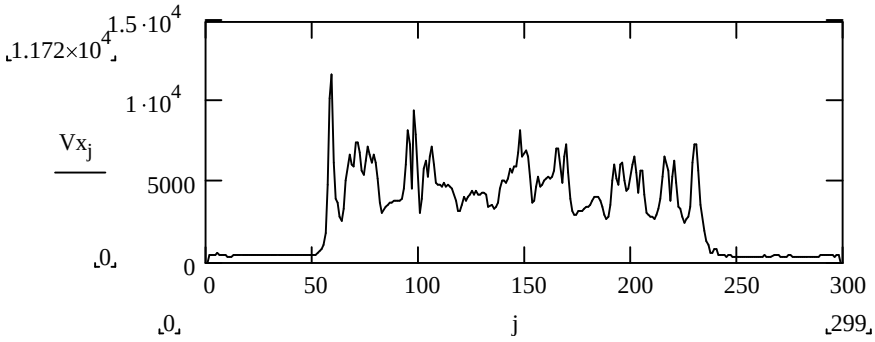


Fig. 4.10. Perfil de la proyección sobre el eje *X*.

Este perfil es binarizado usando como umbral el 0.38 del máximo (Fig. 4.11).

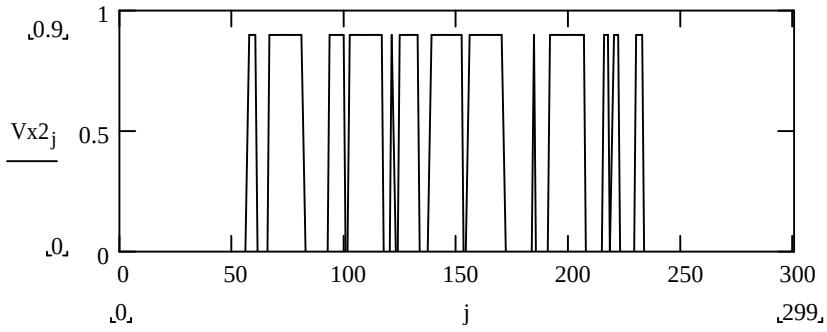


Fig. 4.11. Resultado de la umbralización.

Finalmente aplicaremos la misma idea que usamos para encontrar la posición de la placa, convolucionando ***Vx2*** con una ventana del tamaño aproximado de la placa que queremos encontrar. El número de columnas que ocupa la placa puede estimarse a partir del tamaño del caracter; si conocemos la altura del caracter, podemos saber que su base mide la mitad de su altura (excepto para "1" e "i"). Por tanto, si tenemos una placa de nueve caracteres, su ancho debe ser mayor a 4.5 veces la altura del caracter; esta altura se obtiene a partir de la figura 4.6, es el tamaño de la región principal. Empíricamente encontramos que la longitud estimada L de la placa es 5.8 veces su ancho H.

Como resultado de la convolución tenemos una función que aproximadamente tiene su máximo en el centro de la placa; usando este punto como centro de nuestro espacio estimado de longitud L podemos calcular los límites en horizontal de nuestra placa. El resultado final se muestra en la fig. 4.12.

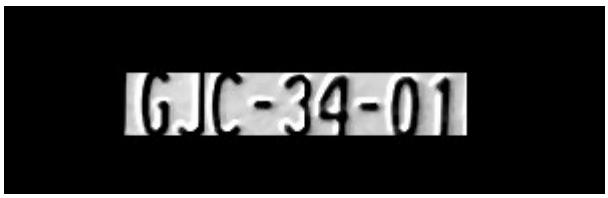


Fig. 4.12. Sección de caracteres recortada de la imagen de la placa.

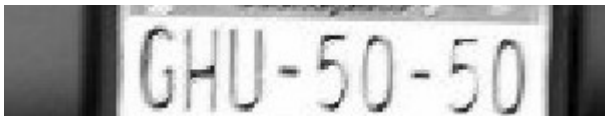
4.1.3 Binarización.

Idealmente la zona de caracteres de una placa tiene un histograma bimodal, donde aparecen separados los píxeles del fondo y los píxeles de caracteres, y esto nos permite separar fácilmente los caracteres del fondo eligiendo un umbral adecuado. Pero en ciertos casos podemos encontrarnos con varios problemas que dificultan la realización de esta tarea.

Desde una mala iluminación que ocasiona un bajo contraste o sombras, hasta imágenes que aparecen como fondo de la tira de caracteres, como en las placas del estado de Jalisco (fig. 4.13 e).

Comparamos el desempeño de tres métodos para la segmentación de los caracteres: el método de Otsu [28] , binarizado adaptativo [29] y el método que evaluaremos, basado en encontrar un umbral que nos de el porcentaje de nivel de gris de los caracteres predeterminado.

a)



b)

c)

d)

e)

Fig 4.13. Ejemplos de placas de automóvil problemáticas.

4.1.3.1 Métodos de umbralización.

4.1.3.1.1 Método de Otsu.

El histograma de una imagen nos da la probabilidad de que un determinado nivel de gris aparezca en la imagen, esto es:

$$P(i) = f(i)/N \ , \tag{4.13}$$

donde *P(i)* es la probabilidad para el nivel de gris *i*, *f(i)* es la frecuencia de píxeles de un mismo nivel de gris dentro de la imagen, y *N* es el número total de píxeles de la imagen.

Cuando tenemos una imagen con un objeto claramente diferenciable de su fondo por su nivel de gris, tenemos un histograma bimodal (fig. 4.14). En este caso tenemos dos grupos de niveles de grises, que podemos modelar como distribuciones gaussianas. Podemos entonces encontrar el umbral que minimice las varianzas de ambas distribuciones, lo que logramos maximizando la Varianza entre clases σ_B^2 para cada umbral que queramos probar:

$$\sigma_B^2(U) = \omega_1(U)\omega_2(U)\left[\mu_1(U) - \mu_2(U)\right]^2 \ , \tag{4.14}$$

$$\omega_1(U) = \sum_{j=0}^{U-1} P(i) \ , \tag{4.15}$$

$$\omega_2(U) = \sum_{j=U}^{L-1} P(i) \ , \tag{4.16}$$

$$\mu_1(U) = \sum_{j=0}^{U-1} iP(i)N \Big/ \sum_{j=0}^{U-1} P(i)N \ , \tag{4.17}$$

$$\mu_2(U) = \sum_{j=U}^{L-1} iP(i)N \Big/ \sum_{j=U}^{L-1} P(i)N \ , \tag{4.18}$$

donde *U* es el umbral bajo prueba y *L* el número máximo de niveles de grises. Por tanto, para encontrar el umbral adecuado debemos probar σ^2_B para cada nivel de gris que debamos probar. Para ahorrar tiempo de cálculo, en lugar de tener que recalcular los valores

cada vez que variamos el umbral bajo prueba, podemos actualizar los valores usando las siguientes ecuaciones:

$$\omega_1(U+1) = \omega_1(U) + P(U) \ , \tag{4.19}$$

$$\omega_2(U+1) = \omega_2(U) - P(U) \ , \tag{4.20}$$

$$\mu_1(U) = \mu_1(U)\omega_1(U) + UP(U)/\omega_1(U+1) \ , \tag{4.21}$$

$$\mu_2(U) = \mu_2(U)\omega_2(U) - UP(U)/\omega_2(U+1) \ , \tag{4.22}$$

El método de Otsu puede ser generalizado para encontrar umbrales con N clases diferentes, aunque su velocidad disminuye. En [30] se examina un método rápido para encontrar umbrales múltiples.

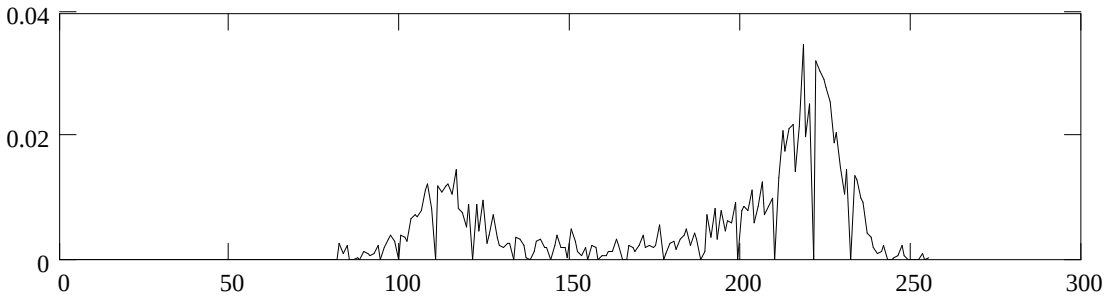


Fig. 4.14. Histograma bimodal.

4.1.3.1.1 Binarizado adaptativo.

Es un método de mejoramiento de imagen que se utiliza cuando es deseable una imagen de dos niveles (binarizada), tales como huellas digitales, texto, dibujos de líneas. Consiste en calcular un umbral para cada píxel, calculando el promedio de los píxeles en su

vecindad. En las imágenes que tratamos definimos una vecindad rectangular alrededor del píxel que estamos tratando en el momento, así que el umbral buscado queda:

$$U(x, y) = \sum_{i=-(M-1)/2}^{(M-1)/2} \sum_{j=-(N-1)/2}^{(N-1)/2} f(x+i, y+j) \Big/ MN \qquad (4.23)$$

Para M y N impares, siendo $f(x,y)$ el valor del píxel, y $U(x,y)$ el umbral calculado para ese píxel.

Este método es adecuado cuando el uso de un solo umbral para toda la imagen no es adecuado. Lo examinaremos debido a que nos veremos con imágenes con sombras, ruidos y fondos irregulares.

4.1.3.1.3 Porcentaje de nivel de gris de los caracteres.

Podemos suponer que, en promedio, los caracteres de una placa ocupan un porcentaje fijo del total de píxeles dentro de la imagen. Entonces, si queremos encontrar un umbral óptimo para separar los caracteres del fondo, lo que tenemos que hacer es calcular el histograma acumulado de la imagen, y tomar como umbral el nivel de gris donde el valor de la función sea mayor que el porcentaje predeterminado (fig. 4.15). El histograma acumulado se calcula:

$$PA(i) = \sum_{j=0}^{i} P(j) \quad , \qquad (4.24)$$

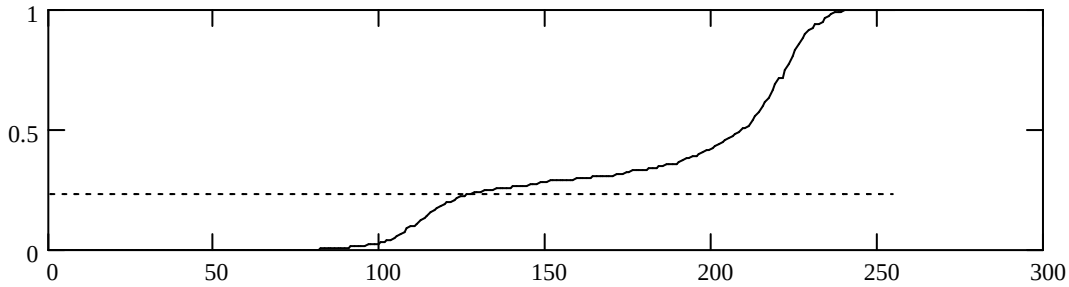
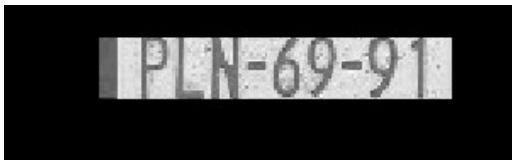


Fig. 4.15. Cálculo del umbral para un porcentaje de 24%, en este caso *U=129*.

4.1.3.1 Resultados.

El método de binarizado adaptativo no funcionó para la vecindad de 7x7, ya que el ruido afecta bastante su desempeño (5(d), 6(d), 7(d), 8(d), 9(d)). El mejor resultado se obtuvo al usar la vecindad de 19x19, pero el ruido sigue interfiriendo en el binarizado(5(f), 7(f)), además de que no logra mejorar las imágenes con sombras (8(f)), aunque logra un desempeño parecido a los métodos de binarizado adaptativo y de Otsu para la imagen donde la placa presenta un dibujo de fondo (figura 9). La figura 6(f) no presenta ruido pero los bordes de los caracteres se deforman.

El método de Otsu falla en tres de las cinco imágenes bajo prueba (7(c), 8(c), 9(c)). En la figura 7 falla debido a que los niveles de gris que corresponden a los caracteres no pueden separarse de los niveles de gris del fondo, y no podemos aplicar el modelo del histograma bimodal. En la figura 8 falla debido a la sombra; el histograma que obtenemos ya no corresponde a nuestro modelo pues ya tiene más de dos agrupamientos de píxeles. Como tenemos dos zonas con diferente brillantez en la misma imagen, el histograma de la imagen con sombra será bimodal, pero diferente de la porción de imagen sin sombra.



a) Imagen recortada de la placa.

b) Porcentaje de niveles de grises, U=129.

c) Método de Otsu, U=165.

d) Binarizado adaptativo, M,N=7.

e) Binarizado adaptativo, M,N=13.

f) Binarizado adaptativo, M,N=19.

g) Histograma de la zona de placa mostrada en (a).

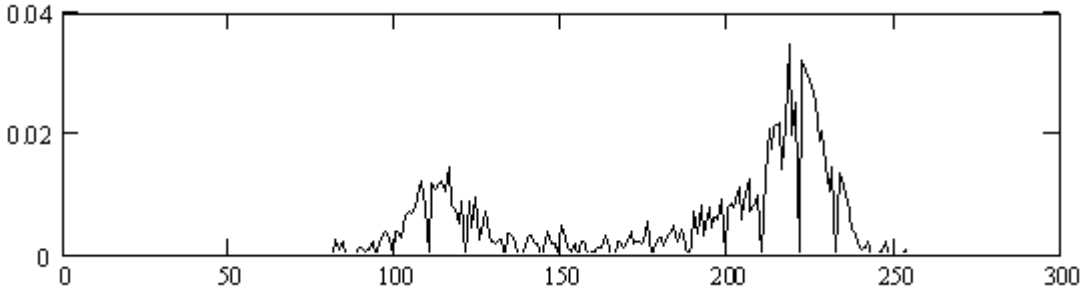
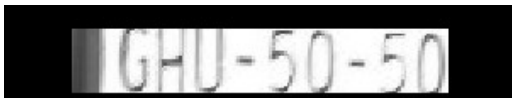


Fig. 4.16. Comparación de los tres métodos aplicados sobre la imagen de la figura 4.13a.



a) Imagen recortada de la placa.

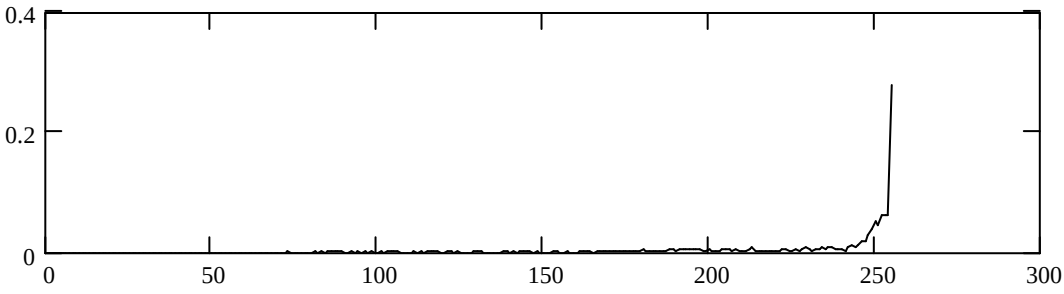
b) Porcentaje de niveles de grises, U=229.

c) Método de Otsu, U=212.

d) Binarizado adaptativo, M,N=7.

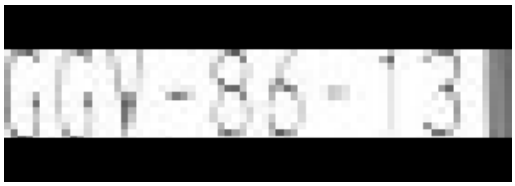
e) Binarizado adaptativo, M,N=13.

f) Binarizado adaptativo, M,N=19.

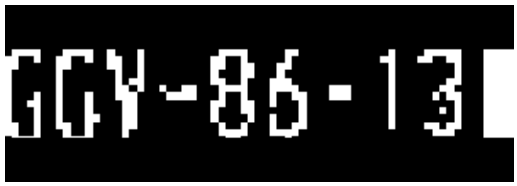


g) Histograma de la zona de placa mostrada en (a).

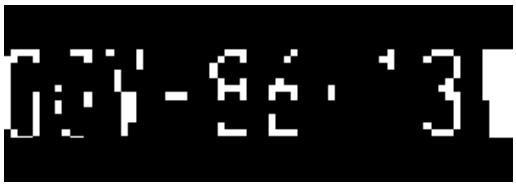
Fig. 4.17. Comparación de los tres métodos aplicados sobre la imagen de la figura 4.13b



a) Imagen recortada de la placa.



b) Porcentaje de niveles de grises, U=241.

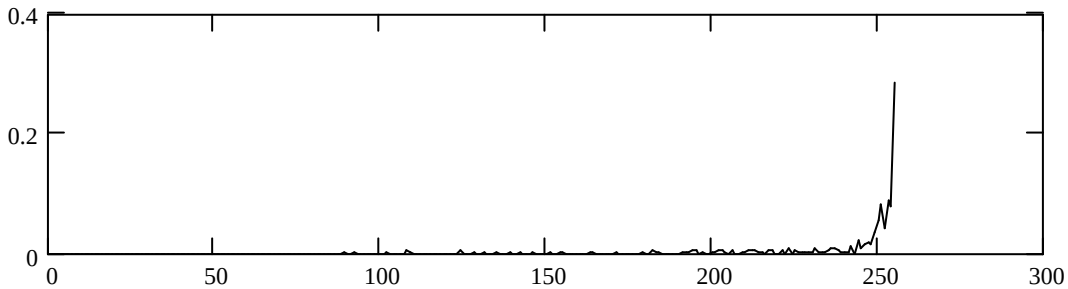


c) Método de Otsu, U=209.

d) Binarizado adaptativo, M,N=7.

e) Binarizado adaptativo, M,N=13.

f) Binarizado adaptativo, M,N=19.



g) Histograma de la zona de placa mostrada en (a).

Fig. 4.18. Comparación de los tres métodos aplicados sobre la imagen de la figura 4.13c.

a) Imagen recortada de la placa.

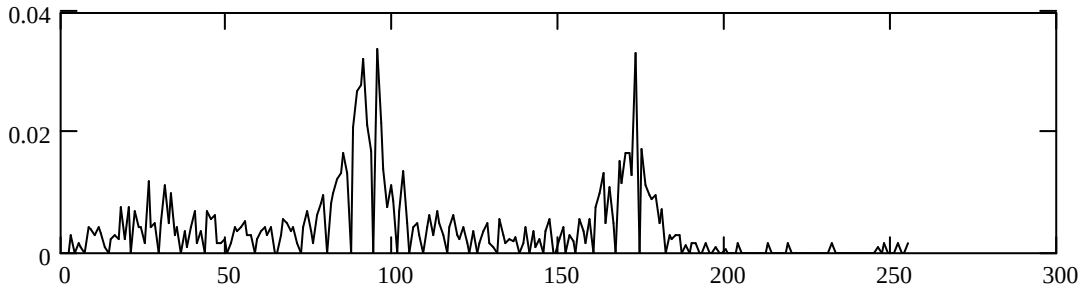
b) Porcentaje de niveles de grises, U=74.

c) Método de Otsu, U=120.

d) Binarizado adaptativo, M,N=7.

e) Binarizado adaptativo, M,N=13.

f) Binarizado adaptativo, M,N=19.



g) Histograma de la zona de placa mostrada en (a).

Figura 4.19. Comparación de los tres métodos aplicados sobre la imagen de la figura 4.13d.

a) Imagen recortada de la placa.

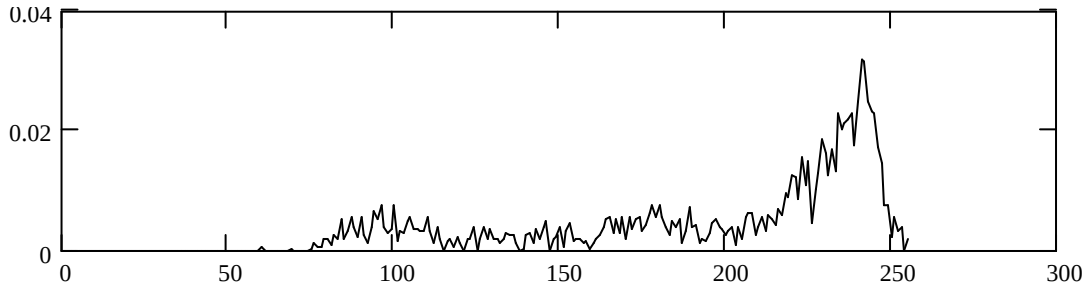
b)Porcentaje de niveles de grises, U=166.

c) Método de Otsu, U=176.

d) Binarizado adaptativo, M,N=7.

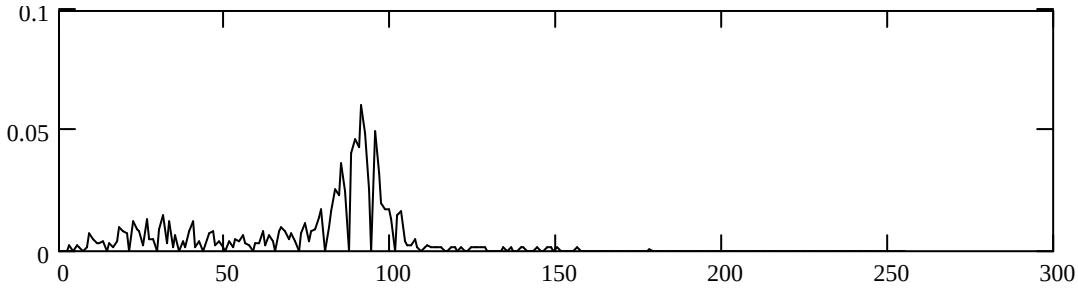
e) Binarizado adaptativo, M,N=13.

f) Binarizado adaptativo, M,N=19.

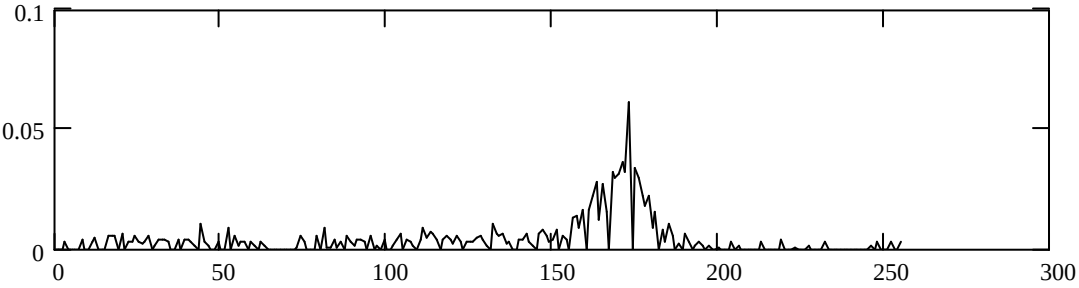


g) Histograma de la zona de placa mostrada en (a).

Fig. 4.20. Comparación de los tres métodos aplicados sobre la imagen de la figura4.13e.



a) Histograma de la zona con sombra de la figura 4.19(a).



b) Histograma de la zona sin sombra de la figura 4.19(a).

Fig. 4.21.

Como al calcular el histograma tomamos en cuenta las dos secciones, los histogramas se superponen (fig. 4.16).

Empíricamente determinamos el porcentaje de píxeles que ocupan los caracteres (20%), para un grupo de 73 imágenes de placas. Para las imágenes complicadas que estamos analizando ahora, el método del porcentaje de niveles de gris de caracteres logró resolver 4 de las cinco imágenes e incluso eliminar el ruido, pero aún tuvo problemas para resolver la última imagen.

Si tenemos imágenes tomadas con la cámara Cohu, como la de la fig. 4.12, el umbral debe de ser de 23.5%. Podemos diferenciar ambos casos mediante el valor máximo de la figura 3.17: Si es mayor a 200, usamos 20%, si es menor, usamos 23.5%. El resultado de la binarización se muestra en la fig. 4.22.

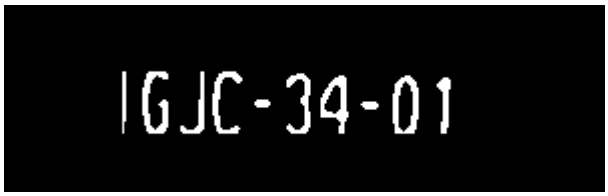
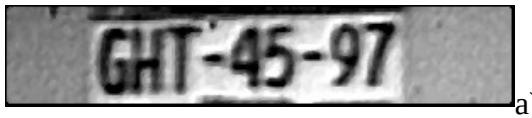


Fig. 4.22 Resultado de la binarización usando el método de porcentaje de escala de grises.

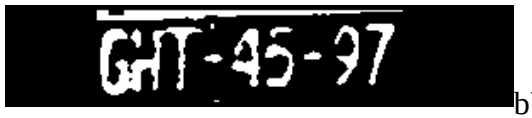
4.1.4 Segmentación de caracteres.

Ya que hemos logrado encontrar el umbral óptimo para la extraer los caracteres, ahora toca delimitar las coordenadas de las esquinas de cada caracter para extraerlas usando la ec. 4.10. hacia una imagen de *16x25*.

Partiendo de la imagen binaria, existen varias aproximaciones para obtener las posiciones de los caracteres. Podemos etiquetar las regiones y así cortarlas una a una; este método supone que la binarización se hizo correctamente y no tenemos caracteres cortados o enlazados. Cuando tenemos imágenes con buena iluminación y bien contrastadas no tenemos ese problema, pero ciertos tipos de imágenes tomadas con la cámara Cohu son complicadas (fig. 4.23).



a)



b)

Fig. 4.23. G,H y T están enlazados.

Cuando tenemos dos o tres caracteres unidos, podemos resolver el problema obteniendo la diferencia entre los puntos extremos en la dirección del eje *X*. Si esta diferencia es igual o mayor que *H*(altura de los caracteres), indica que dos o más caracteres están unidos. Si es aproximadamente *H*, con dividir la en dos se resuelve el problema; si está cerca de *1.5H* dividimos en tres.

Por otro lado tenemos el problema de caracteres cortados en dos o más secciones. Si el corte es vertical, no hay manera de diferenciar entre un caracter mal cortado, un “1” o “i”, o ruido hasta aplicar el reconocimiento de caracteres; si es horizontal podemos intentar localizar regiones que se traslapan.

El algoritmo que tenemos programado es un poco diferente. Se basa en calcular el perfil horizontal de la imagen binaria, por ej. 4.22. El perfil se calcula mediante la siguiente función lógica (ver tabla 3.2):

$$V(j) = \sum_{i=0}^{n-1} M(i, j) \quad , \tag{4.25}$$

Es decir, la ec. 4.25 detecta si en la columna tenemos un píxel en uno lógico, y si es así pone el correspondiente ***V(j)*** a uno; así creamos varias regiones que corresponden a las posiciones exactas de los caracteres. Este sistema corrige el problema de tener caracteres cortados horizontalmente, pero es sensible a ruidos que unan dos caracteres, o dar un falso valor para algún punto de ***V(j)***.

Los problemas introducidos por el ruido pueden disminuirse. En caso de tener caracteres unidos sólo dividimos las regiones grandes. Si la longitud de alguna región es menor a *0.125H* es muy probable que sea ruido, así que lo eliminamos. De esta manera, ahora solo detectamos las coordenadas *x* de inicio y final de cada región, mientras que las coordenadas y_1 e y_2 las obtenemos de ***Vy2*** (fig. 4.24).

Fig. 4.24. Caracteres segmentados de la Fig. 4.22.

- Las coordenadas *y* deben recalcularse para cada caracter, tomando como punto de inicio y_1 e y_2.
- Si ningún píxel en (x,y_1), con *x* variando entre x_1 y x_2, es "1" lógico y en el siguiente renglón tampoco tenemos "1", nos movemos al siguiente renglón. En caso contrario, paramos.
- Si estamos en (x,y_2) y el renglón anterior no encontramos "1", nos movemos al renglón anterior. En caso contrario, paramos.

Como se ve en la figura 4.24, el número "3" quedó cortado en la parte inferior. Para resolver esto, el umbral que hemos encontrado se aplica sobre la imagen de la placa, completa. En la figura 4.25 tenemos la imagen del auto y la imagen de su placa.

a) Imagen posterior de un auto.

b) Imagen binaria de la placa.

Fig. 4.25

A las instrucciones anteriores para ajustar las coordenadas, debemos agregar las siguientes:

- Si algún píxel en (x,y_1) , con x variando entre x_1 y x_2, es "1" lógico y en el renglón anterior también tenemos "1", nos movemos al renglón anterior. En caso contrario, paramos.
- Si estamos en (x,y_2) y el renglón posterior encontramos "1", nos movemos al renglón posterior. En caso contrario, detenemos el proceso.

Fig. 4.26. Caracteres segmentados de la fig. 4.25b.

Los casos de “1” e “I” son especiales. Para los demás caracteres, lo que queremos es que el caracter ocupe todo el espacio de *16x25*. Cuando la diferencia entre las coordenadas *x* de un caracter está entre *0.125H* y *0.25H*, se interpreta que es uno o “i”. El caracter se ajusta sólo en la mitad derecha del espacio de *16x25*, en un espacio de *7x25*, como se ve en el “1” de la fig. 4.22.

4.2 Conclusiones.

De todos los casos donde podemos hallar la posición de la placa, en 8.2% el algoritmo falla en delimitar el área de los caracteres. Aunque es un buen sistema, todavía pueden hacerse mejoras.

El método de porcentaje de niveles de gris mostró un desempeño superior a los otros métodos de binarización que fueron puestos a prueba. Otra ventaja es su rapidez, pues solo implica calcular el histograma acumulado, y esta característica es necesaria si queremos lograr un sistema de reconocimiento de caracteres de placas rápido para su uso en control de tráfico o en control de accesos con tráfico alto.

Los métodos de binarización y segmentación requieren más trabajo, pues hay varios tipos de imágenes de bajo contraste que no pueden resolver, aunque su principal problema

son las deformaciones en la placa o el ángulo de giro. En el capítulo 6 se abundará más sobre este punto.

CAPÍTULO 5

Reconocimiento Óptico de Caracteres (OCR)

5.1 Introducción.

Las diferentes áreas cubiertas bajo el término general “Reconocimiento de caracteres” pueden dividirse en aplicaciones “en línea” (On-line) y “fuera de línea”(Off-Line) [31]. En reconocimiento de caracteres en línea, la computadora reconoce los símbolos al tiempo que éstos son escritos. El reconocimiento fuera de línea es ejecutado después que el caracter ya haya sido impreso o escrito.

Un sistema típico de reconocimiento de caracteres tiene varios componentes. La figura 5.1 muestra un esquema ampliamente utilizado. El documento de entrada es digitalizado para producir una imagen binaria o en escala de grises. Un programa de control localiza la región en la imagen, y segmenta las palabras en caracteres aislados.

Después de la segmentación, los caracteres, en la forma de matrices binarias pasan por procesos de suavizado, eliminación del ruido, normalización de tamaño y otras operaciones, para facilitar la extracción de características en la fase posterior.

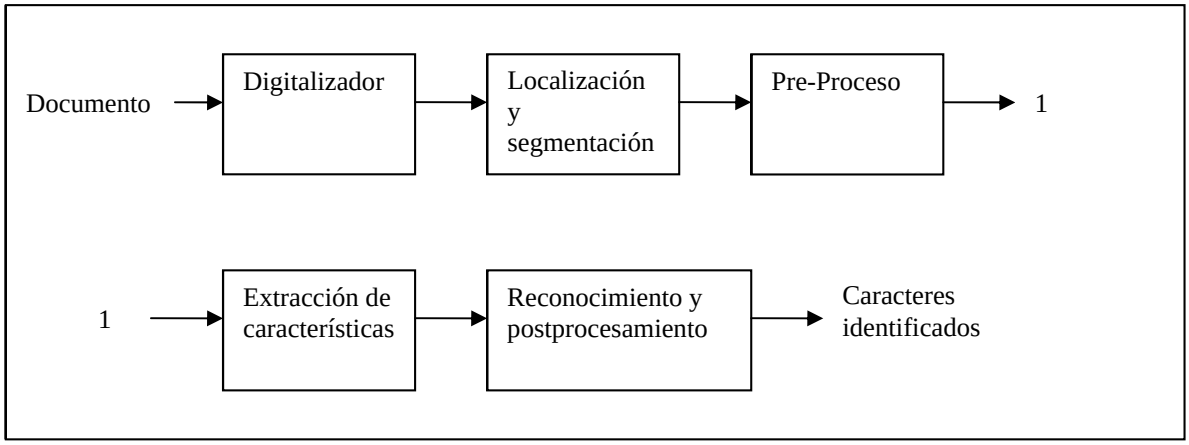


Fig. 5.1. Diagrama de bloque de un sistema de reconocimiento de caracteres típico.

La identificación de caracteres se realiza comparando las características extraídas con las estadísticas de características obtenidas del grupo de ejemplo usado en la fase de entrenamiento.

Finalmente, la información lingüística, contextual o estadística puede ser usada para resolver ambigüedades de caracteres que tienen perfiles similares, o para corregir palabras o frases.

En base a los conceptos expuestos, podemos ver el problema de reconocimiento de placas desde el punto de vista del reconocimiento de caracteres. Solo que partimos del punto en que ya tenemos una imagen y no nos preocupamos cómo fue tomada. Nuestro programa de control localiza la placa, selecciona la región de interés y segmenta los caracteres. Ajustamos las dimensiones de las imágenes de los caracteres a un tamaño de *16x25* pixeles.

Usamos una red neuronal para la identificación de los caracteres aislados. La literatura muestra varias ventajas de las redes neuronales sobre otros métodos para el reconocimiento de patrones [32]:

- Redes neuronales bien diseñadas (redes de doble capa) superan métodos clásicos en fiabilidad, mientras que requieren menos capacidad de almacenamiento [33].

- Son una herramienta universal que no requiere una cantidad considerable de programación detallada para una aplicación específica.

- Combinan las ventajas de los enfoques estadísticos y basados en conocimiento. Son complicadas de entrenar, pero conocimiento sobre el problema puede incorporarse a la arquitectura de la red. Esto permite disminuir la cantidad de datos necesarios para entrenar la red.

5.2 Conceptos básicos de neurofisiología.

El concepto de redes neuronales surgió a partir de los primeros trabajos de neurofisiología que intentaban explicar el funcionamiento de la neurona. Por tanto, primero veremos algunos conceptos básicos de neurofisiología.

5.2.1 Fisiología de una neurona.

En la figura se representan los componentes principales de una célula nerviosa. La membrana de la neurona separa el plasma intracelular del fluido intersticial que se encuentra fuera de la célula. La membrana es permeable para ciertos iones, de forma tal que

se crea una diferencia de potencial entre el interior y el exterior de la célula. Al mecanismo responsable de lograr esta diferencia de iones se le llama bomba de sodio-potasio.

Todos los tipos de iones pueden salir de la célula por difusión, excepto por lo iones orgánicos (carga negativa) que son demasiados grandes. Dado que los iones orgánicos no pueden salir de la célula, su carga negativa dificulta la entrada en la célula de iones de cloro (carga negativa).

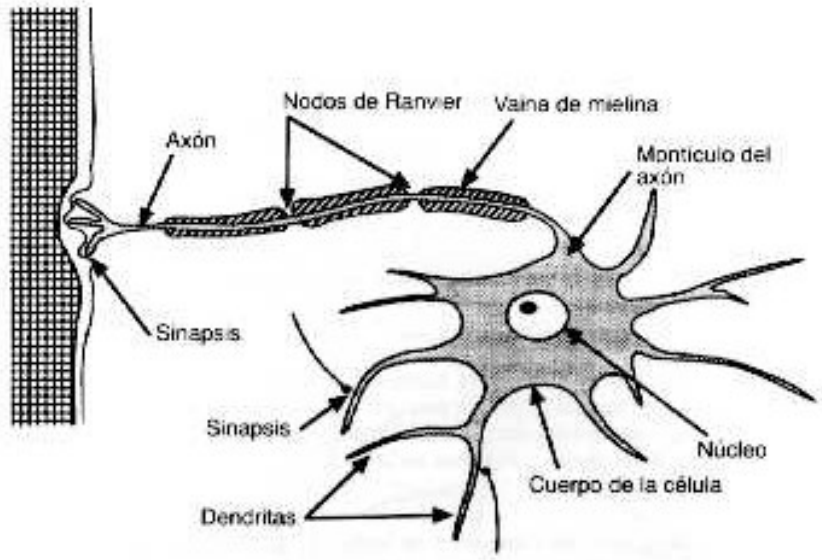


Fig. 5.2 Estructura de una neurona. Las sinapsis conectan el axón de la neurona con distintas partes de otras neuronas (Freeman, "Redes Neuronales").

La membrana celular es más permeable para los iones de potasio (carga positiva) que para los iones de sodio. El gradiente químico del potasio tiende a hacer que los iones de potasio salgan de la célula por difusión, pero la fuerte atracción de los iones orgánicos negativos tiende a mantener dentro el potasio.

Como resultado de los procesos anteriores, se tiene un exceso de iones negativos fuera de la célula, e iones positivos dentro de la célula. La diferencia de potencial producida es del orden de 70 a 100 mV. Este potencial se le llama *potencial de reposo* de la célula.

La figura 5.3 ilustra una neurona con varias conexiones de entrada, y los potenciales que se tienen en distintas posiciones. La cubierta del axón se llama vaina de mielina, la cuál es interrumpida en varios puntos por los nodos de Ranvier.

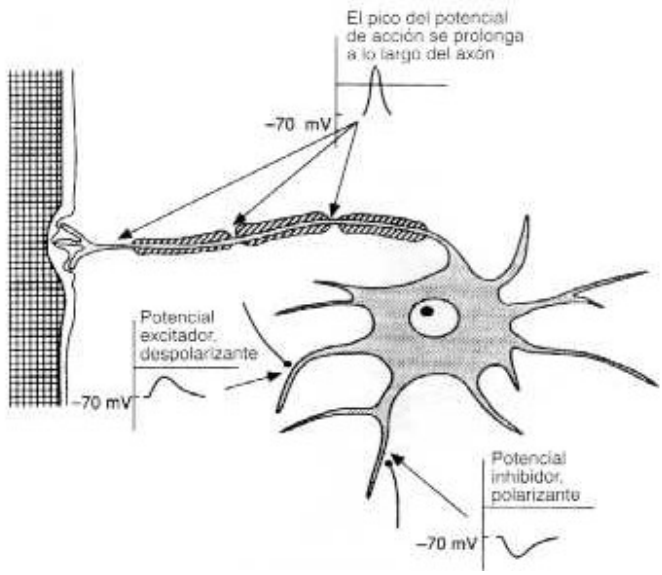


Fig. 5.3 Potenciales en la neurona (Freeman, "Redes Neuronales").

Las entradas excitatorias que llegan a la célula a través de las sinapsis, reducen la diferencia de potencial en la membrana. La despolarización resultante en el montículo del axón altera la permeabilidad de la membrana celular a efectos de los iones de sodio. Como resultado, los iones de sodio pueden entrar a la célula, contribuyendo más a la despolarización. Este efecto autogenerado da lugar al potencial de acción.

Las fibras nerviosas son malos conductores, pero el potencial de acción puede transmitirse gracias a una serie de despolarizaciones que tienen lugar en los nodos de Ranvier: cuando uno de los nodos se despolariza, se desencadena la despolarización del siguiente nodo. Una vez que el potencial de acción ha pasado por un nodo. Ese nodo no puede volver a ser excitado durante un milisegundo, que es el tiempo que tarda en volver a su potencial de reposo; este período refractario limita la frecuencia de transmisión de los impulsos nerviosos a 1 kHz. Esta frecuencia obviamente es menor a las frecuencias utilizadas en las computadoras actuales, entonces ¿por qué el cerebro humano es más eficiente para ciertas acciones como el reconocimiento de patrones, el aprendizaje, etc, que las computadoras? La respuesta parece hallarse en la forma en que están organizadas las neuronas. Pero antes de ver este tema, veamos primero como funciona la sinapsis en las neuronas, tema clave para comprender los modelos actuales de la organización neuronal en el cerebro.

5.2.2 Las sinapsis

Las sinapsis es el espacio existente en la unión entre dos neuronas. Ya que el impulso nervioso no puede transmitirse a través de este espacio, la comunicación entre neuronas se realiza gracias a sustancias llamadas neurotransmisores, que son liberados por la célula presináptica y absorbidos por la célula postsináptica: cuando el potencial de acción llega a la membrana presináptica, los cambios de permeabilidad de la membrana dan lugar a un flujo entrante de iones de calcio, los cuáles estimulan la liberación de los neurotransmisores.

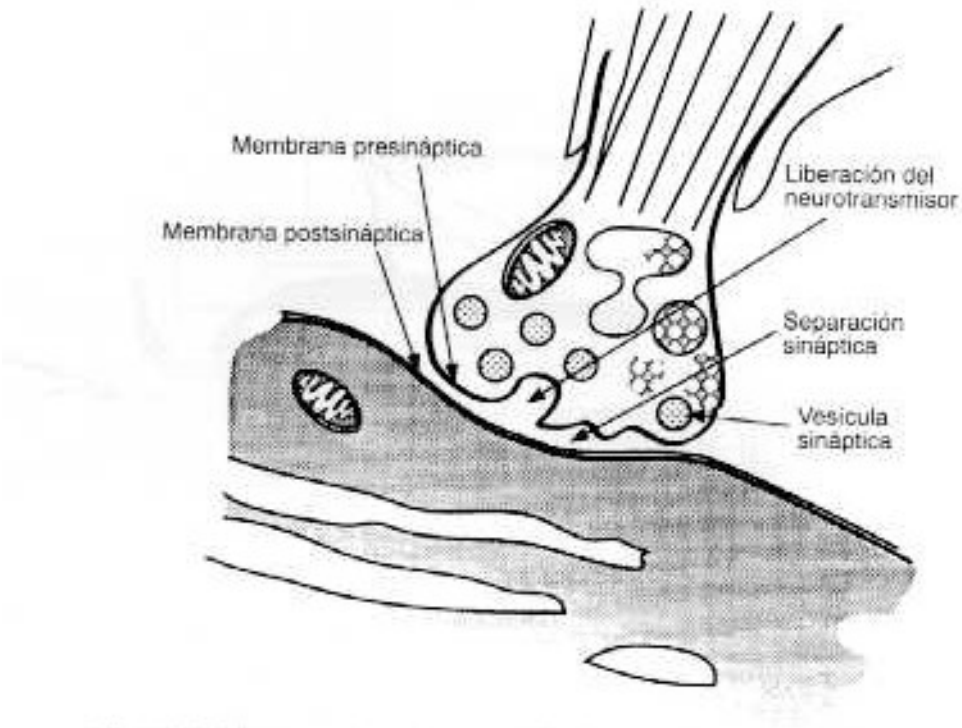


Fig. 5.4 Los neurotransmisores se almacenan en vesículas próximas a la sinapsis (Freeman, "Redes Neuronales").

Los neurotransmisores se difunden a través de la unión y se unen a la membrana postsináptica en ciertos lugares llamados receptores. La acción química que se produce en los receptores da lugar a cambios de permeabilidad de la membrana postsináptica: un flujo entrante de iones positivos producirá una despolarización (excitación), y si entran iones negativos se producirá un efecto hiperpolarizante (inhibición). Estos dos efectos son locales, y actúan sólo a lo largo de una pequeña distancia hacia el interior de la célula; se suman en el montículo del axón, y si la suma es mayor que un cierto valor de umbral se genera un potencial de acción.

5.2.3 Circuitos neuronales.

Una vez que ya hemos visto como funcionan cada neurona individual, debemos determinar cómo es que las neuronas se relacionan entre sí para dar al cerebro su enorme capacidad.

Existen varios modelos de circuitos neuronales, pero todos se basan en un sencillo principio: cada neurona envía impulsos a otras neuronas (divergencia) y recibe impulsos provenientes de otras (convergencia); estas señales pueden ser tanto excitatorias como inhibitorias, y ya que también se pude dar la retroalimentación de señales, ésta puede ser positiva o negativa.

El primer intento significativo para entender el funcionamiento del sistema nervioso a partir de los principios anteriormente mencionados, se hizo en 1943, a través del trabajo de McCulloch-Pitts [34]. Este fue le primer trabajo que trató al cerebro como un organismo computacional. La teoría de McCulloch-Pitts se basa en cinco suposiciones:

1. La actividad de una neurona es un proceso todo-nada (las neuronas están, o bien activas o desactivadas).
2. Es preciso que un número fijo de sinapsis (>1) sean excitadas dentro de un período de adición latente para que se excite una neurona.
3. El único retardo significativo dentro del sistema nervioso es el retardo sináptico.
4. La actividad de cualquier sinapsis inhibitoria impide por completo la excitación de la neurona en ese momento.

5. La estructura de la red de interconexiones no cambia con el transcurso del tiempo.

Aunque esta teoría ha resultado no ser un modelo preciso de la actividad cerebral, indican una idea fundamental que no estaba explícita en el trabajo en sí: Aunque las neuronas son dispositivos sencillos, se puede obtener una gran potencia de cálculo cuando se interconectan adecuadamente estas neuronas y se insertan dentro del sistema nervioso (Anderson y Rosenfeld)[35].

Los sistemas neuronales biológicos no nacen preprogramados con todo el conocimiento y las capacidades, sino que se tiene que dar un proceso de aprendizaje, pero, ¿cómo es el proceso por el cuál aprendemos, visto desde el punto de vista de las conexiones neuronales? La teoría básica procede de un libro de 1949 escrito por Hebb, Organization of Behavior. La idea principal es:

Cuando un axón de la célula A está suficientemente próximo para excitar a una célula B o toma parte en su disparo de forma persistente, tiene lugar algún proceso de crecimiento o algún cambio metabólico en una de las células, o en las dos, de tal modo que la eficiencia de A, como una de las células que desencadena el disparo de B, se ve incrementada [36].

Aunque esta ley de Aprendizaje no describe totalmente el proceso de aprendizaje, aparece de una u otra forma en muchos de los modelos neuronales que existen en la actualidad.

5.3 El elemento general de procesamiento.

El modelo de procesamiento que describimos aquí no pretende ser un modelo exacto del funcionamiento de una neurona, sino que sólo es una aproximación basada en nuestros conocimientos actuales sobre neurofisiología. Por este motivo, los elementos individuales de cálculo que forman las redes neuronales no suelen llamarse neuronas artificiales, sino que es más frecuente llamarlos nodos, unidades o elementos de procesamiento (PEs).

La figura 5.5 muestra el modelo general de PE. Cada PE está numerado, siendo el i -ésimo el que aparece en la figura. El PE tiene muchas entradas, pero una sola salida, que puede ser aplicada a muchos otros PEs en la red, y suele denominarse x_i, mientras que la entrada que recibe el PE del j-ésimo elemento es x_j. Cada conexión con el i-ésimo PE tiene asociada a él una magnitud llamada peso o intensidad de conexión; el peso de la conexión procedente del j-ésimo nodo y que llega al i-ésimo nodo se denota mediante w_{ij}.

Obsérvese que las entradas de un PE están divididas en varios tipos, lo que indica que cada división tiene diferentes efectos. Una entrada puede ser inhibitoria o excitatoria, o incluir términos de tendencia (concepto que será explicado más adelante). Existen otros efectos utilizados en varios tipos de redes neuronales, como la ganancia, amortiguamiento y disparo fortuito que por ahora no explicaremos por no ser utilizados en la red neuronal implementada [37].

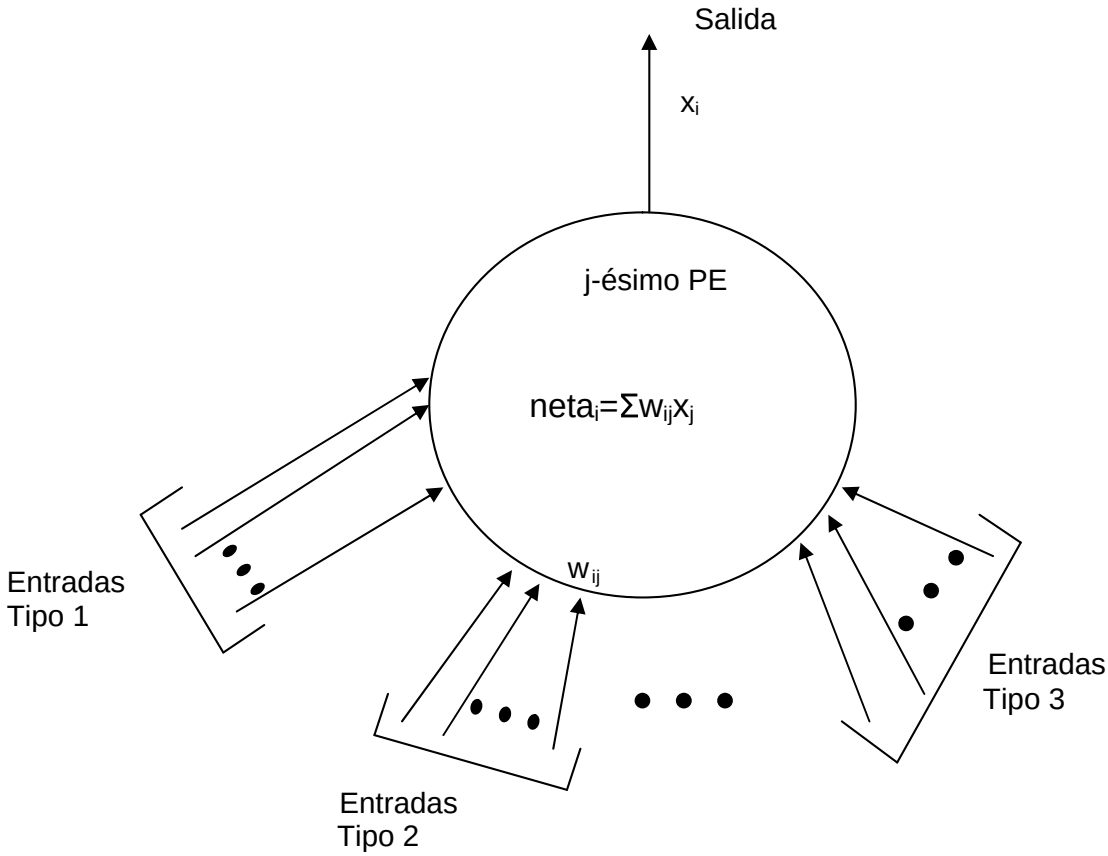


Fig. 5.5 Elemento general de procesamiento (Freeman, "Redes Neuronales").

Cada PE determina un valor de entrada neto basándose en todas las conexiones de entrada. En ausencia de conexiones especiales, lo típico es calcular el valor neto multiplicando cada entrada por su peso correspondiente, y después sumando los resultados:

$$RED_i = \sum_{j=1}^{n} x_j w_{ij} \quad , \tag{5.1}$$

La excitación y la inhibición se toman en cuenta según el signo de su peso correspondiente.

Una vez que la entrada neta ha sido calculada, se transforma en el valor de activación para ese PE. Se puede escribir el valor de activación como:

$$a_i(t)=F_i(a_i(t-1),RED_i(t)) \ , \qquad (5.2)$$

Como la simulación digital es la forma más común de implementar redes neuronales, se toma el tiempo discreto. El término $a_i(t-1)$ se incluye por generalidad, en caso de que la entrada posterior de una red sea función de una entrada anterior. En la mayoría de los casos, la activación (valor de activación) y la entrada neta son idénticas

U na vez que se ha calculado la activación del PE, se determina el valor de salida utilizando la función de salida:

$$x_i = f_i(a_i) \ , \qquad (5.3)$$

y dado que normalmente $a_i = RED_i$, podemos escribir:

$$x_i = f_i\,(RED_i) \ , \qquad (5.4)$$

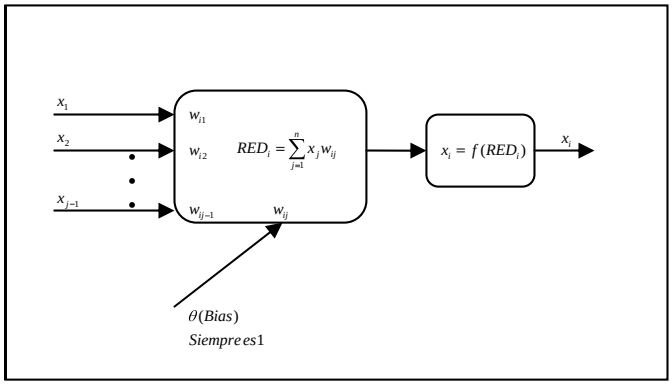


Fig. 5.6 Diagrama de bloques de un elemento general de procesamiento (PE).

Podemos ver la colección de valores de los pesos como un sistema dinámico, es decir, que cambia con el tiempo; recuérdese la descripción del proceso de aprendizaje de Hebb, en la que el aprendizaje es un resultado de la modificación de la fuerza entre las uniones sinápticas entre neuronas. Se puede escribir un sistema de ecuaciones diferenciales para los pesos:

$$\dot{w}_{ij} = G_i(w_{ij}, x_i, x_j, ...) \ , \qquad (5.5)$$

en donde G_i representa la ley de aprendizaje. El proceso de aprendizaje consiste en hallar los pesos que codifican ese conocimiento que deseamos que aprenda el sistema. Para la mayoría de los sistemas reales, no es fácil determinar una solución en forma cerrada para este sistema de ecuaciones, pero existen técnicas que dan una aproximación razonable de la solución, aunque la investigación en este aspecto sigue su marcha.

5.4 El perceptrón.

Ahora haremos una breve descripción de uno de los dispositivos basados en las ideas anteriores.

El perceptrón fue inventado por el psicólogo Frank Rosemblatt a finales de los años cincuenta [38]. El fotoperceptrón es un dispositivo que responde a señales ópticas.

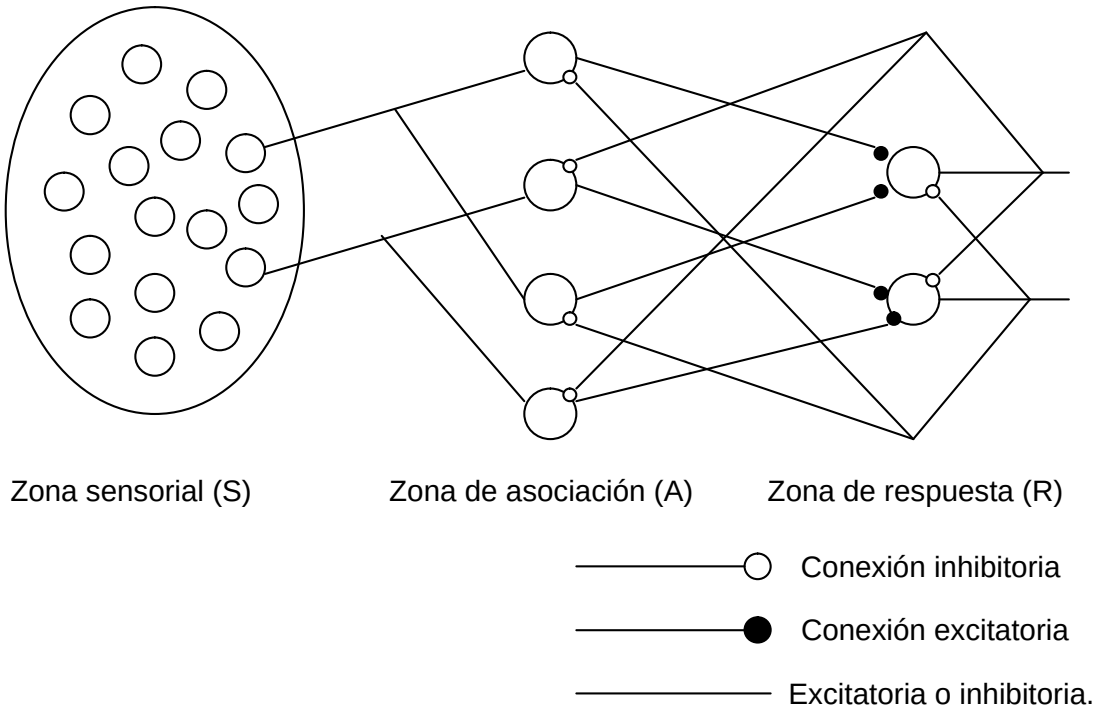
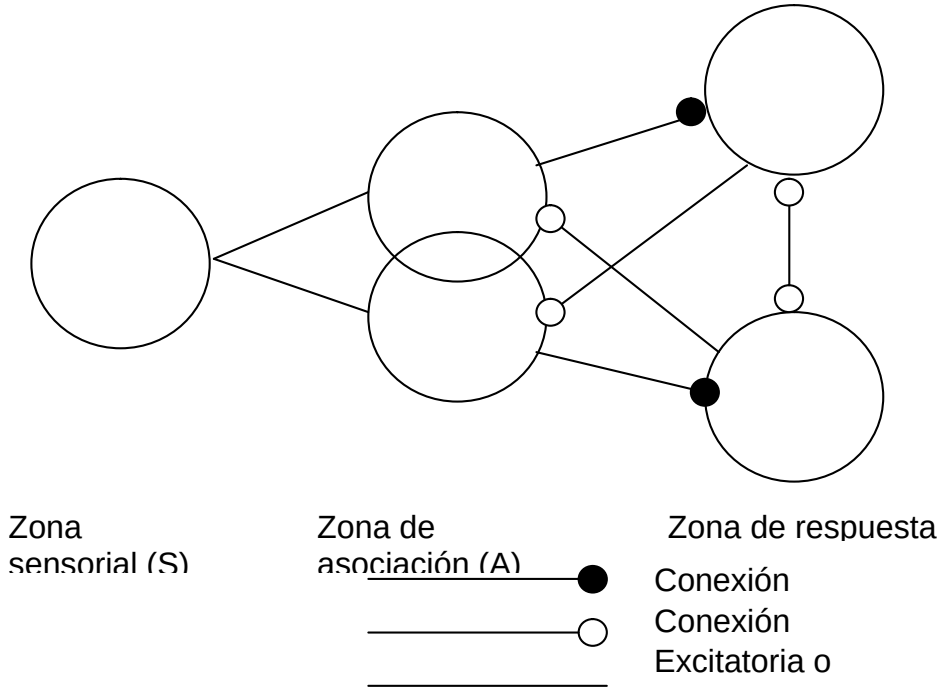


Fig. 5.7 Diagrama esquemático del perceptrón (Freeman, "Redes Neuronales").

La luz incide en los puntos sensibles *S* de la estructura de la retina. Cada punto *S* responde en forma binaria a la luz entrante, y los impulsos generados se transmiten a las unidades de asociación *A* de la capa de asociación. Cada unidad está conectada a un conjunto aleatorio de puntos *S*, denominado conjunto fuente de la unidad *A*, y las conexiones pueden ser tanto excitatorias como inhibitorias *(+1,-1,0).* Cuando aparece una trama de estímulos en la retina, una unidad *A* se activa si la suma de sus entradas sobrepasa algún valor umbral. Si está activada, la unidad *A* produce una salida, que se envía a la siguiente capa de unidades, la capa de respuesta. Las unidades *A* están conectadas a las unidades de respuesta *R* mediante una trama de conectividad aleatoria entre capas, añadiéndose conexiones inhibitorias de realimentación procedentes de la capa de respuesta y que llegan a la capa de asociación. Además, existen conexiones inhibitorias entre las unidades *R*.



Fig, 5.8 Representación de un perceptrón sencillo como diagramas de Venn (Freeman, "Redes Neuronales").

En la fig. 5.8 se describe un perceptrón sencillo con dos unidades *R*, en forma de diagramas de Venn. Cada una de las unidades *R* inhibe a las unidades *A* en el complemento de su propio conjunto fuente, además de inhibirse entre sí. Estos factores hacen que sólo una unidad *R* se active con los estímulos de la retina (las unidades *R* responden en forma similar a las unidades *A*).

Se puede utilizar el sistema anteriormente descrito para clasificar tramas que aparezcan en la retina por categorías, de acuerdo con el número de unidades de respuesta que haya en el sistema. Las tramas que sean suficientemente parecidas deberían excitar a la misma unidad *R*. Por lo tanto, se trata de un problema de separabilidad: ¿sería posible

construir un perceptrón tal que pueda distinguir con éxito entre distintas clases de tramas? La respuesta es sí, pero deben cubrirse ciertos requisitos que veremos después.

El perceptrón es un dispositivo de aprendizaje. Al inicio, no es capaz de reconocer ninguna trama, pero mediante un proceso de aprendizaje es capaz de hacerlo. En esencia., el entrenamiento implicaba un proceso de refuerzo mediante el cuál la salida de las unidades *A* se incrementan o decrementan dependiendo de si las salidas contribuían o no a las respuestas correctas del perceptrón para una trama dada.

Utilizando un sistema así, Rosemblatt pudo demostrar que el perceptrón era capaz de clasificar tramas correctamente, en lo que él denominaba un entorno diferenciado, en el cuál cada clase estaba formada por tramas que eran similares. El perceptrón también era capaz de responder de manera congruente frente a tramas aleatorias, pero su precisión disminuía mientras más tramas intentaba aprender.

En 1969 Marvin Minski y Seymour Papert, en su libro "Perceptrons: An introduction to Computational Geometry" [39] presentaron un análisis del perceptrón en términos de sus capacidades y limitaciones. Como parte de su análisis, mencionan las restricciones para las clases de problemas: los perceptrones sólo pueden distinguir tramas si éstas son linealmente separables. Para llegar a esta conclusión se basaron en un perceptrón idealizado (similar a un elemento general de procesamiento), como el que se muestra en la figura 5.9.

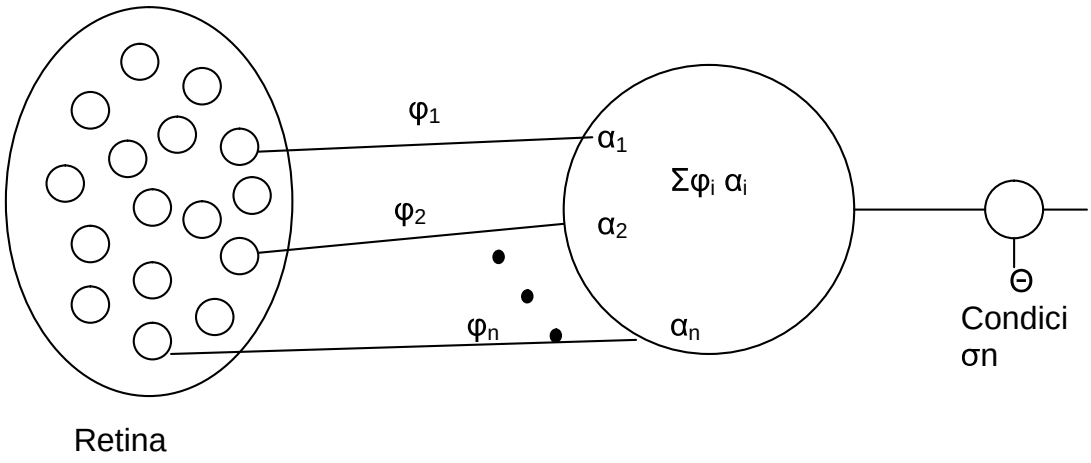


Fig. 5.9 Estructura de un perceptrón sencillo (Freeman, Redes Neuronales).

El conjunto $\Phi=\{\varphi_1, \varphi_2, ..., \varphi_n\}$ es un conjunto de predicados. En su forma más sencilla, $\varphi_i=1$ cuando el *i-ésimo* punto de la retina está activado, y $\varphi_i=0$ en caso contrario. Cada uno de los predicados está ponderado mediante un número del conjunto $\{\alpha_1, \alpha_2, ..., \alpha_n\}$. La salida Ψ es uno si $\Sigma_n \alpha_n \varphi_n > \Theta$, donde Θ es el valor de umbral.

Ahora debemos explorar el concepto de hiperplanos, para poder explicar cómo es que los perceptrones permiten la clasificación de tramas.

En el espacio tridimensional, un plano es un objeto de dos dimensiones. Un único plano puede descomponer el espacio tridimensional en dos regiones distintas, dos planos pueden dar lugar a tres o cuatro regiones distintas, dependiendo de sus orientaciones relativas, y así sucesivamente. En un espacio n-dimensional (hiperespacio) los hiperplanos son objetos de n-1 dimensiones; una colocación adecuada de los hiperplanos permite descomponer los espacios n-dimensionales en varias regiones diferentes.

La ecuación general de un hiperplano es:

$$\sum_{i=1}^{n} \alpha_i x_i = C \quad , \tag{5.6}$$

en donde α_i son constantes, con al menos un $\alpha_i \neq 0$, y x_i son las coordenadas del espacio n-dimensional. Como se puede ver, esta ecuación es idéntica a la ecuación de salida de un perceptrón sencillo, donde φ_i es semejante a x_i (dimensiones del hiperespacio) y α_i corresponde en ambos casos a las constantes; o en un caso más general, a la ecuación de un elemento general de procesamiento.

Existen muchos problemas reales que implican la separación de regiones de puntos del hiperespacio en categorías individuales o clases que deben distinguirse de otras clases. Una forma de hacer estas distinciones consiste en seleccionar hiperplanos que descompongan el espacio en regiones adecuadas, tarea que es difícil realizar en un espacio de muchas dimensiones, pero ciertas redes neuronales pueden aprender la descomposición adecuada. Todo lo que hay que hacer para dividir el espacio en regiones es añadir varios elementos generales de procesamiento, uno por cada hiperplano que se desee agregar; estas PE's se agregan en una nueva capa, llamada capa intermedia, que se encuentra entre las unidades de entrada, y uno o varios PE que sirven de salida.

5.5 Red multicapas y algoritmo de retropropagación.

En esta sección explicaremos las características básicas y el método de entrenamiento de la red neuronal utilizada para el reconocimiento de las tramas obtenidas durante el método de segmentado: la red de retropropagación.

La red de retropropagación es útil para problemas que requieren el reconocimiento de tramas complejas y la realización de funciones de correspondencia no triviales. Está diseñada para que funcione como red multicapa, con propagación hacia delante, empleando el modo supervisado de aprendizaje.

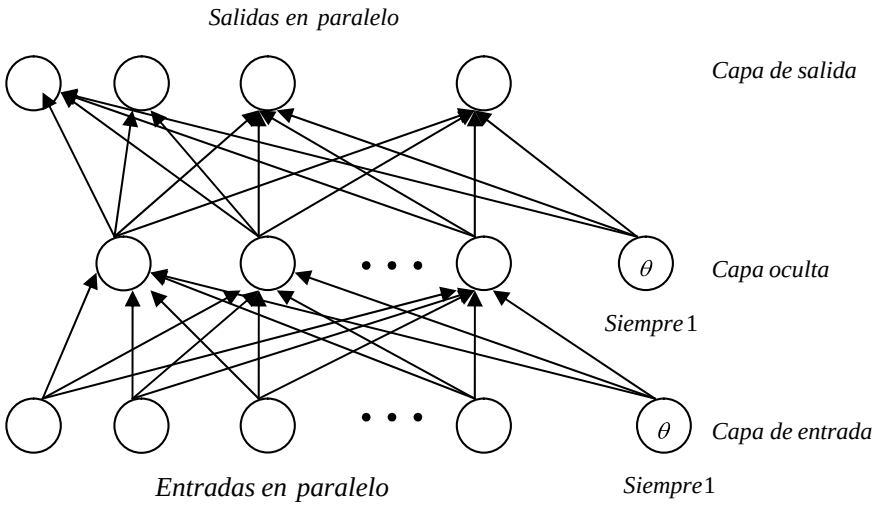


Fig. 5.10 Estructura general de una red de retropropagación.

En la figura 5.10 vemos la estructura general de una red de retropropagación. Tiene nodos de entrada que no tienen otra función mas que distribuir las entradas de la trama a la siguiente capa de elementos de procesamiento; además, en la capa de entrada tenemos una unidad de tendencia (que se agrega por razones prácticas que serán comentadas más adelante) cuyo valor es siempre 1, pero cuyos pesos varían durante el proceso de entrenamiento. La red BPN (Backpropagation Net, red de retropropagación) puede tener una o más capas ocultas; en este caso, sólo tenemos una capa oculta. Cada unidad de la capa oculta tiene una estructura semejante al elemento general de procesamiento, excepto por un detalle: sólo tiene un solo tipo de entrada, la cuál puede ser excitatoria o inhibitoria

de acuerdo al signo del peso asociado con esa entrada. La salida es una función de RED_i; en este caso en particular, se escogió una función sigmoide como función de salida:

$$f_k^o(RED_{jk}^o) = (1 + e^{-RED/T})^{-1}, \qquad (5.7)$$

Donde *T* es un factor que permite escalar la función en su respuesta a *RED* (permite ajustar una sensibilidad para *RED*, pudiendo asemejarse a una función escalón o, por el contrario, expandir la curva para tener una zona de respuesta lineal).

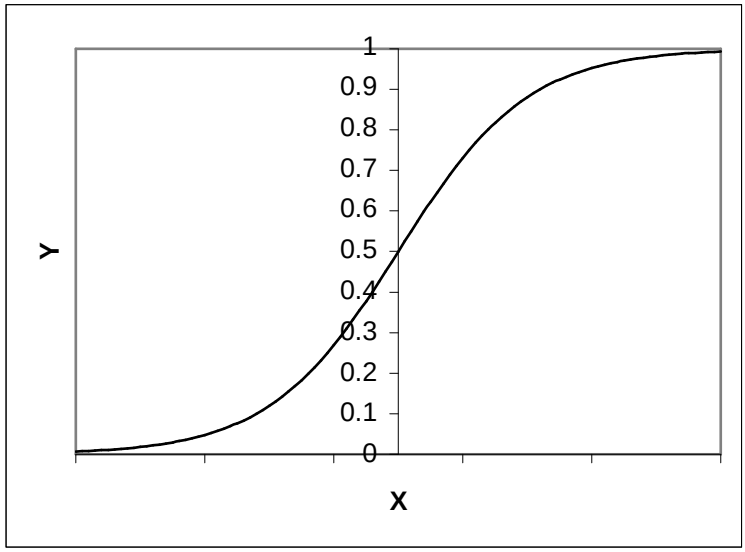


Fig. 5.11 Forma característica de la función sigmoide.

Tenemos una tercera capa, llamada capa de salida, cuyos elementos de procesamiento son idénticos en características a los elementos de la capa oculta. La capa de salida también presenta una unidad de tendencia análoga a la unidad de tendencia de la capa oculta.

En la red BPN no tenemos entradas de realimentación, y todas las unidades están completamente conectadas, es decir, cada salida de las distintas unidades está conectada a todas las unidades de la capa siguiente.

5.5.1 Funcionamiento de una BPN.

La red aprende un conjunto predefinido de pares de entrada y salidas dadas como ejemplo, empleando un ciclo propagación-adaptación de dos fases. Una vez que se ha aplicada una trama como estímulo para la primera capa de unidades de la red, ésta se va propagando a través de todas las capas superiores hasta generar una salida. La señal de salida se compara con la salida deseada, y se calcula una señal de error para cada unidad de salida.

Las señales de error se transmiten entonces hacia atrás, partiendo de la capa de salida, hacia todos los nodos de la capa intermedia que contribuyan directamente a la salida. Las unidades de la capa oculta o intermedia reciben sólo una fracción del error total, basándose en su contribución relativa al error total. Este proceso se repite capa por capa hasta que todos los nodos reciban una señal de error que describa su contribución relativa al error total, señal que se utiliza para la actualización de los pesos de cada unidad, para hacer que la red converja hasta un estado de error mínimo.

A medida que se entrena la red, los nodos de las capas intermedias se organizan a sí mismos de tal modo que aprenden a reconocer distintas características del espacio total de entradas. Después del entrenamiento, cuando se presente a la entrada una trama arbitraria

que contenga ruido o que esté incompleta, las unidades de las capas ocultas de la red responderán con una salida activa si las características de la trama se asemejan a aquellas que las unidades hayan aprendido a reconocer durante su entrenamiento. De este modo, obtenemos una trama de salida que constituye un mapa de características que puedan estar presentes en la trama de entrada. El comportamiento global de la BPN (backpropagation network) permite, entonces, reconocer tramas incompletas o con ruido.

Varios investigadores han demostrado que durante el entrenamiento, la red tiende a desarrollar relaciones internas entre nodos que organizan los datos de entrenamiento en clases de tramas [40]. Esto significa que la red ha encontrado una representación interna que le permite generar las salidas deseadas cuando se le dan las entradas de entrenamiento; esta misma representación interna puede aplicarse a entradas no utilizadas en el entrenamiento, las cuáles clasificará según las características que compartan con los ejemplos de entrenamiento [41].

5.5.2 Proceso de entrenamiento de la red BPN.

A continuación se presentan los pasos necesarios para entrenar la red BPN. Si se quiere conocer la demostración del algoritmo, puede consultarse el libro: "Redes neuronales: Algoritmos, aplicaciones y técnicas de programación", de Freeman y Skapura, en el capítulo III.

Los índices *i,j,k* corresponden a las capas de entrada, capa oculta y capa de salida respectivamente. Las dimensiones del vector de entrada es *N*; la capa oculta tiene *L*

unidades, y la capa de salida tiene M unidades (estas no son necesariamente las convenciones utilizadas en el programa; en caso de haber cambios, se indicarán en el mismo programa).

1.- Se aplica el vector de entrada $X_p=(x_{p1}, x_{p2}, ..., x_{pN})^t$ a las unidades de entrada.

2.- Se calculan los valores netos procedentes de las entradas para las unidades de la capa oculta:

$$RED^o_{pj} = \sum_{i=1}^{N} w^o_{ji} x_{pi} + \theta^o_j \quad , \tag{5.8}$$

3.- Se calculan las salidas de la capa oculta:

$$i_{pj} = f^o_j(RED^o_{pj}) \ , \tag{5.9}$$

4.- Se calculan los valores netos de las entradas para cada unidad:

$$RED^s_{pj} = \sum_{i=1}^{L} w^s_{kj} i_{pj} + \theta^s_k \tag{5.10}$$

5.- Se calculan las salidas:

$$y_{pk} = f^s_k(RED^s_{pk}) \tag{5.11}$$

6.- Se calculan los términos de error para las unidades de salida.

$$\delta^s_{pk} = (O_{pk} - y_{pk}) f^{s'}_k(RED^s_{pk}) \tag{5.12}$$

Donde O_{pk} es la salida deseada para ese elemento.

7.- Se calculan los términos de error para las unidades ocultas.

$$\delta^o_{pj} = f^{o'}_j(RED^o_{pj}) \sum_k \delta^s_{pj} w^s_{kj} \tag{5.13}$$

Los términos de error de las unidades ocultas se calculan antes de que hayan sido actualizados los pesos de conexión con las unidades de la capa de salida.

8.- Se actualizan los pesos de la capa de salida:

$$w_{kj}^{s}(t+1) = w_{kj}^{s}(t) + \eta \delta_{pk}^{s} i_{pj} \tag{5.14}$$

η es llamada parámetro de velocidad de aprendizaje. Suele ser positivo y es menor que 1.

9.- Se actualizan los pesos de la capa oculta:

$$w_{ji}^{o}(t+1) = w_{ji}^{s}(t) + \eta \delta_{pj}^{o} x_i \tag{5.15}$$

El orden de actualización de pesos de una capa individual no es importante.

Por último, debe calcularse el término del error:

$$E_p = \frac{1}{2} \sum_{k=1}^{M} \delta_{pk}^{2} \tag{5.16}$$

Puesto que esta magnitud es la medida de qué tan bien está aprendiendo la red. Cuando el error es aceptablemente pequeño para todos los pares de entrenamiento, éste puede concluirse.

La derivada de la función sigmoide es:

$$f' = y(1 - y) \tag{5.17}$$

donde y corresponde a la salida del correspondiente elemento de procesamiento, ya sea de la capa oculta o de la capa de salida.

5.6 Consideraciones prácticas.

5.6.1 Datos de entrenamiento.

Pueden utilizarse todos los datos disponibles para entrenar la red, aunque con frecuencia sólo se necesitan un pequeño subconjunto de los datos. Los vectores restantes pueden utilizarse para probar la red.

Si se está entrenando una red para que funcione en un entorno con ruido, entonces deben incluirse algunos vectores de entrada ruidosos. Algunas veces, la adición de ruido a los vectores de entrada durante el entrenamiento ayuda a la red a converger, aunque no se lo espere en las entradas durante el funcionamiento normal.

La BPN generaliza bien. Esto quiere decir que si tenemos varios vectores distintos, cada uno perteneciente a la misma clase, la BPN aprenderá a adaptarse a las similitudes de los vectores de entrada, eliminando los datos irrelevantes.

Si una BPN se entrena de modo inadecuado o insuficiente empleando una clase concreta de vectores de entrada, la posterior identificación de miembros de esa clase será imprecisa. Por lo tanto, debe asegurase que los datos de entrenamiento cubren todo el espacio de entradas esperado. Además, durante el entrenamiento, no debe entrenarse la red completamente con miembros de una clase para después pasar a otra pues la red olvidará el entrenamiento original.

Si la función de salida es una sigmoide, entonces es necesario aplicar una escala a los vectores de entrada, ya que esta función está limitada a valores entre cero y uno, por lo que las salidas de la red nunca pueden alcanzar el cero o el uno. Debido a lo anterior, debemos limitar los valores a niveles entre 0.1 y 0.9, o también puede desplazarse la sigmoide para que tenga valores de $\pm$ 0.4. Además, puede modificarse la pendiente de la parte lineal de la curva, incluyendo una constante multiplicativa en la exponencial, como ya se mencionó antes.

5.6.2 Dimensionamiento de la red.

En general, para la mayor parte de los problemas tres capas son suficientes, pero existen casos en que la convergencia se logra más rápido al incluir más de una capa oculta.

El tamaño de la capa de entrada es determinado por la naturaleza de la aplicación (por ej., la cantidad de características faciales tomadas en cuenta, el tamaño en pixeles de una imagen, etc). A menudo es posible determinar el número de nodos de salida dependiendo de si se desean valores analógicos o valores binarios en las unidades de salida.

No es fácil determinar la cantidad de unidades ocultas que deben utilizarse, aunque la idea es que deben utilizarse la menor cantidad de elementos como sea posible, ya que cada unidad supone una mayor carga en tiempo de cálculo sobre el CPU en las simulaciones de redes neuronales. Para redes de un tamaño considerable (cientos o miles de entradas) el tamaño de la capa oculta sólo necesita ser una fracción de la capa de entrada. Si

la red no llega a converger, es posible que se necesiten una mayor cantidad de nodos ocultos. Si converge, puede probarse con menor número de nodos ocultos.

También es posible eliminar unidades ocultas que resulten superfluas, es decir, aquellos nodos que prácticamente no varíen sus valores iniciales, lo que indicaría que no están participando en el proceso de aprendizaje.

5.6.3 Pesos y parámetros de aprendizaje.

Los pesos deben tener unos valores iniciales pequeños y aleatorios (por ejemplo, entre ± 0.5), al igual que los términos de tendencia, ya que si fueran todos los pesos iguales, la red no aprendería nada. Se suele tratar a este valor de tendencia como a un peso más, que está conectado a una unidad ficticia cuya salida es siempre 1; otra opción es eliminar los términos de tendencia, aunque se ha observado que su uso permite lograr la convergencia de una manera más rápida.

La elección del valor del parámetro de aprendizaje debe tener un valor pequeño (entre 0.05 y 0.25) para asegurar que la red llegue a asentarse en alguna solución. Un valor pequeño de η significa que la red tendrá que hacer un gran número de iteraciones, aunque es posible ir incrementándolo a medida que avance el aprendizaje, acelerando la convergencia; el peligro de esta técnica estriba en que la red puede rebotar, alejándose demasiado del valor mínimo verdadero.

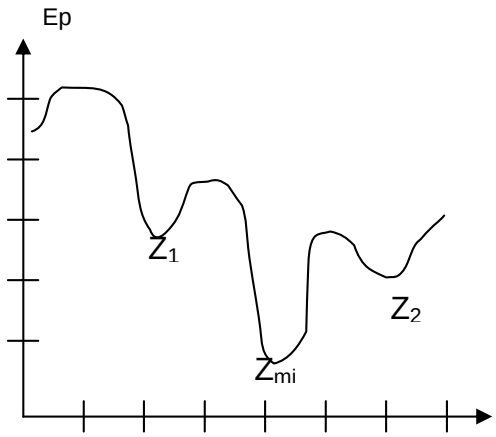


Fig. 5.12 Mínimos locales *(Z1,Z2)* y globales *(Zmin)*.

Finalmente, tenemos el problema de que la red converja a un mínimo local en lugar de un mínimo global. Una vez que la red cae en un mínimo, ya sea global o local, cesa el aprendizaje. Si se alcanza un mínimo local, el valor del error puede ser excesivamente alto, aunque esto puede resolverse cambiando el número de nodos ocultos, cambiando los parámetros del aprendizaje o cambiando los valores iniciales de los pesos; pero si el error obtenido es aceptable, no importa si se ha alcanzado un mínimo local o uno global.

Un factor que durante la elaboración del programa de BPN se encontró que afecta al valor del mínimo global, es el número de elemento ocultos: a mayor número de elementos ocultos, menor es el valor del error mínimo que se puede alcanzar (mínimo global).

5.7 Estructura del sistema de reconocimiento de caracteres.

La red de retropropagación que usamos tiene cuatrocientos elementos en la capa de entrada, cuarenta PE's en la capa oculta y 27 PE's de salida. No tiene términos de tendencia (bias).

Como aproximación inicial no entrenamos todos los caracteres del alfabeto. Comenzamos entrenando la red sólo para los 10 dígitos, y las letras más comunes que aparecen en las placas de Guanajuato, además del guión. Las letras se cortaron de las imágenes del conjunto de 73 placas. En total se entrenaron 16 letras: A, B, C, D, E, F, G, H, J, P, T, U, V, W, Y, Z.

Los cuatrocientos elementos de la capa de entrada corresponden a cada uno de los píxeles de la imagen donde tenemos el caracter segmentado. Los caracteres deben ser imágenes binarias. Desechamos la opción de utilizar imágenes en escala de grises pues la red no convergía adecuadamente.

En la capa oculta tenemos 40 PE's. Este número se encontró empíricamente, como un compromiso entre la velocidad de convergencia y la minimización del error de entrenamiento, para el entrenamiento de los números. El error de entrenamiento disminuye mientras mayor es el número de elementos de procesamiento en la capa oculta, pero no en forma monótona, sino que tenemos varios mínimos locales. Después de 40 PE's, el error sigue disminuyendo asintóticamente, a partir de este punto aumentar el número de elementos no consigue afectar en gran medida el error de entrenamiento.

Otra razón para fijar el número de elementos de procesamiento en la capa oculta en 40 es por los resultados obtenidos por [42], en donde con 50 neuronas ocultas logró un error de clasificación del 5% para los números. Sin embargo, en [43] se reporta una aplicación de reconocimiento de dígitos escritos a mano, donde tienen una capa de entrada de 400 elementos, una capa oculta de 300 unidades y una capa de salida de 10, obteniéndose una tasa de error del 1.6%.

El parámetro de la velocidad de aprendizaje η quedó fijado en 0.01.

Al comienzo de los trabajos se intentó entrenar juntos las letras y los dígitos, pero los resultados no fueron satisfactorios. Por tanto, se tomó la decisión de entrenar pesos diferentes según que lo que queramos reconocer sea una letra o un número. Este método limita la generalidad de nuestra solución, pues ahora debíamos establecer un criterio para juzgar si un caracter es una letra o un número. Aunque la respuesta de la red generalmente es menor para un caracter no entrenado, surgen ambigüedades cuando éste es similar a algún miembro del grupo considerado, como en los casos de "1" e "I", "8" y "B", "6" y "G", etc.

Como redujimos nuestro universo de trabajo a placas de autos particulares, cuya estructura es fija (tres letras, guión, dos dígitos, guión, dos dígitos), podemos esperar que las tres primeras imágenes sean letras. Esto no sucede así; si vemos la fig. 4.24 nos daremos cuenta que nuestro algoritmo de delimitación de la zona de caracteres en ocasiones deja

pasar regiones que no son letras o números, pero pueden producir una respuesta alta en la red, o por lo menos indistinguible de la respuesta a caracteres legítimos.

Este problema se solucionó al incluir una etapa de detección de guiones, que siempre están presentes en las placas de Guanajuato. Usando la misma red neuronal, buscamos que imágenes de posibles caracteres nos dan una respuesta más alta en la salida correspondiente al caracter, y el lugar que tiene es la hilera de imágenes nos permite determinar cuáles son letras y cuáles son números.

Aún separando letras y dígitos, nos encontramos con que la red aún tiene problemas para reconocer ciertas letras. El método que puede resolver esto es la aplicación de un segundo método de reconocimiento de caracteres que trate los casos difíciles. En [44] se presenta una introducción a estos métodos.

Lo más sencillo fue aplicar la misma red neuronal, entrenada con una menor cantidad de clases, como el método para evitar ambigüedades. Aparte de las redes para el reconocimiento de números y letras, se entrenaron otras dos, y siempre que la red de letras produzca algún resultado que se sospeche ambiguo se recurrirá al arbitraje de éstas. La primera red incluye: C,D,E,P,V,Z,B,F,Y. La segunda red incluye D,E,G,H,B,F. El proceso de decisión para usar las redes se muestra en la fig. 5.13.

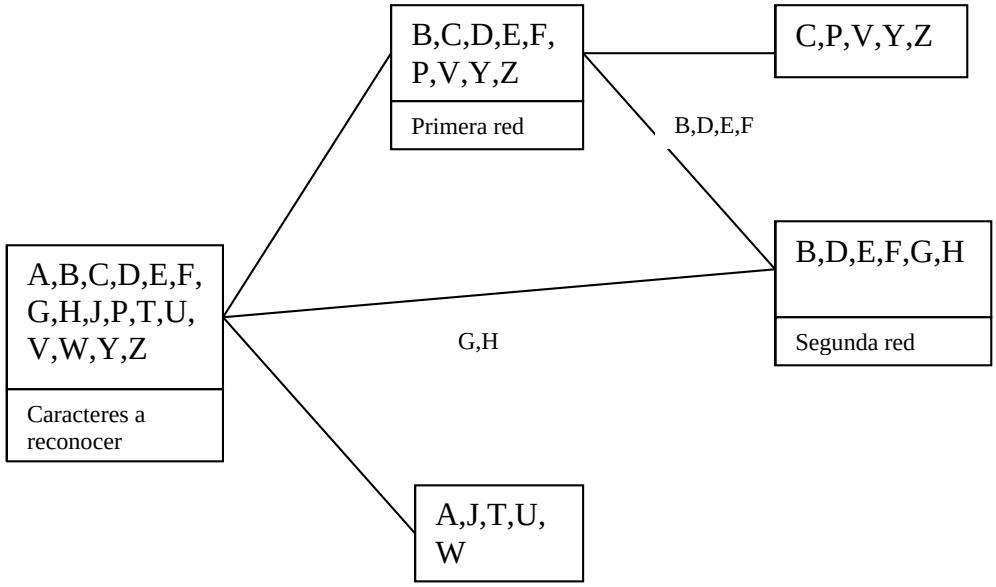


Fig. 5.13. Árbol de decisión del reconocedor de caracteres.

5.8 Cálculo del valor *E* para eliminación de zonas sin placas.

En capítulos anteriores se mencionó que podríamos tener varias zonas donde pudiera estar la placa. Para diferenciarlas, usaremos la respuesta de la red neuronal a cada posible caracter.

Al momento de decidir qué caracter está representado en la imagen a prueba actual, el resultado máximo de la red se guarda en un vector ***N***. Cuando ya hemos identificado todas las imágenes de posibles caracteres, calculamos el valor *E* de la siguiente manera:

$$E = \frac{\sum_i [N(i)]^2}{\sum_i 1} \quad , \tag{5.18}$$

Si E es mayor que cierto umbral, consideraremos que realmente hemos encontrado una placa, pero solo si el tamaño del vector $\boldsymbol{N}$ es mayor a 5.

Tomando el valor de E para 63 placas detectadas, y el valor de E para 11 no detectadas, suponiendo que la distribución es gaussiana, tenemos los resultados de la tabla 5.1.

	Placas	No placa
Promedio	0.568	0.15
σ	0.15	0.052

Tabla 5.1 Media y varianza de E para placas y zonas sin placa.

Estimamos un umbral óptimo resolviendo la ec 5.19:

$$AT^2 + BT + C = 0 \ , \tag{5.19}$$

$$A = \sigma_1^2 - \sigma_2^2 \ , \tag{5.20}$$

$$B = 2\left(m_1\sigma_2^2 - m_2\sigma_1^2\right) \ , \tag{5.21}$$

$$C = \sigma_1^2 m_2^2 - \sigma_2^2 m_1^2 + 2\sigma_1^2\sigma_2^2 \ln\frac{\sigma_2 n_1}{\sigma_1 n_2} \ , \tag{5.22}$$

donde $n_1=63^o$ y $n_2=11$. Con estos datos el umbral óptimo es T=0.24426. Usando este valor tenemos una probabilidad de 1.54% de declarar una placa como inexistente, y una probabilidad de 3.4% de declarar una zona sin placa como placa. En las figuras 5.14, 5.15 y 5.16 vemos un ejemplo de la aplicación de este método a la detección de placas.

Fig. 5.14 Imagen original.

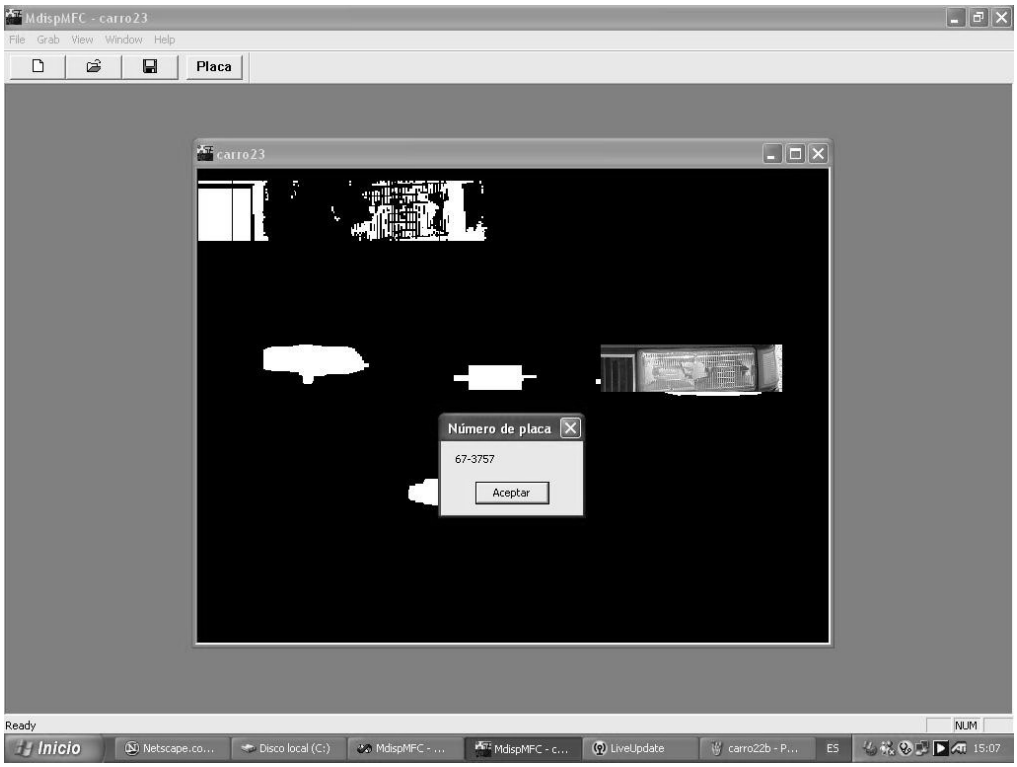


Fig 5.15 El programa falló en su primer intento de buscar la placa.

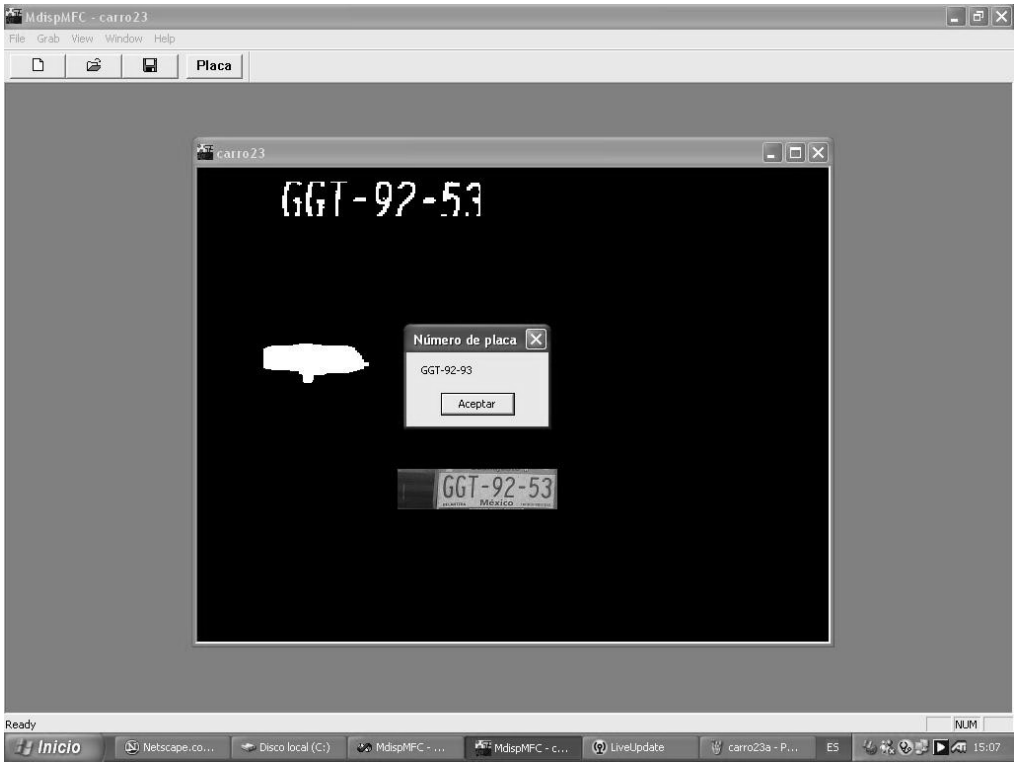


Fig. 5.16 La placa se encontró en el segundo máximo.

CAPÍTULO 6

Funcionamiento global del programa y conclusiones

6.1 Funcionamiento global del programa.

Todo el proceso para detectar la placa, segmentar los caracteres y reconocerlos, al correr en una computadora de 1.5 MHz de 256 Mb de RAM, tarda aproximadamente un segundo. Esto significa que puede manejar sin problema miles de placas por hora.

El algoritmo para la detección de la placa funciona en un 87.67% de los casos. Los casos donde el algoritmo falla, la placa no está bien iluminada y al querer buscar la placa, ésta no responde bien (fig. 6.1) y por tanto no pueden segmentarse los caracteres.

Otra fuente de error lo constituye el detector de guiones. Cuando no conseguimos ubicar correctamente los guiones, tomamos imágenes de ruido como verdaderos caracteres, o usamos la red de números para reconocer letras o viceversa. Por tanto, las salidas de la red tenderán a ser bajas haciendo que $\boldsymbol{E}$ sea menor al umbral establecido (fig. 6.2). El porcentaje de placas con los guiones correctamente localizados es de 91.67%.

a)


b)


c)

Fig. 6.1 En (b) tenemos la imagen de bordes de (a). En la imagen binaria sólo se aprecia parte de la zona de la placa.

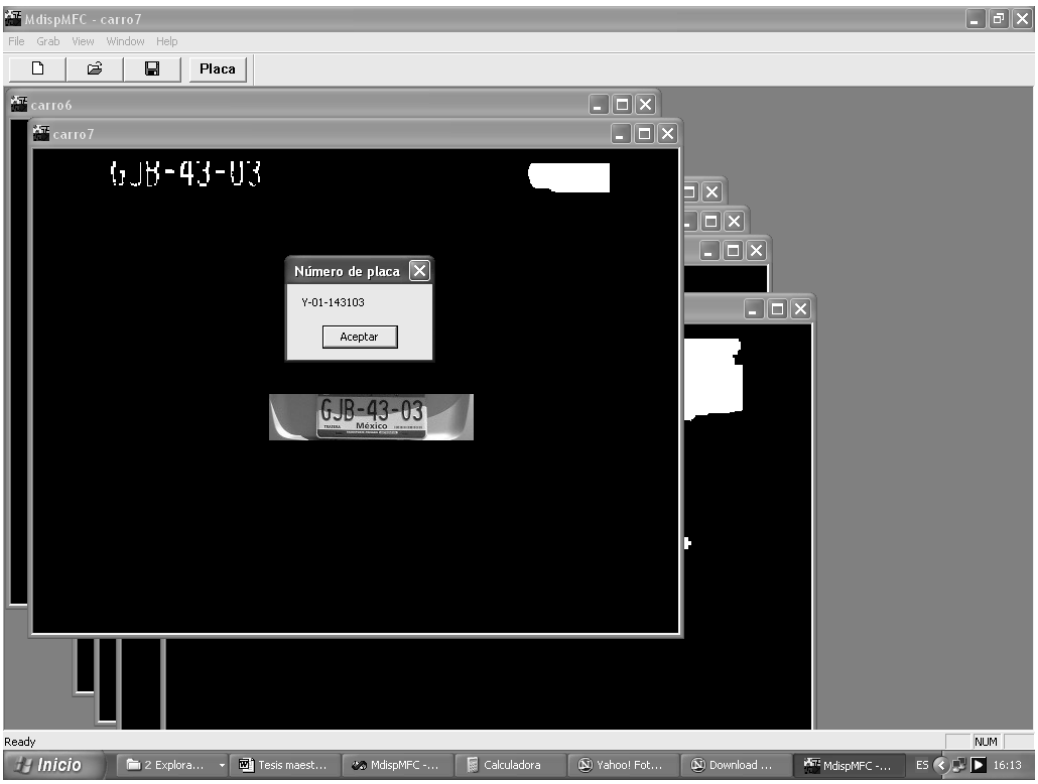


Fig. 6.2 Error en la detección de guiones.

Si definimos el éxito del segmentado como el porcentaje de placas cuyos caracteres son segmentados correctamente aunque sean ruidosos, el porcentaje está sobre 71.43%.

La red de números reconoce bien en el 86.91% de los casos. La red de letras es eficaz en 85.29% de los casos. Si definimos la probabilidad de reconocer todos los caracteres de una placa como:

$$R_p = \prod_i p_i \ , \tag{6.1}$$

donde p_i es la probabilidad de reconocer el carácter en la posición i.

Esto implica que una placa de tres letras y cuatro números tiene una probabilidad de 35.40% de ser reconocida perfectamente. Un sistema que reconozca 95% de los caracteres, sólo podrá reconocer 63.02% de las placas totales. Si quisiéramos tener un porcentaje de reconocimiento del 95%, el sistema debería tener ser capaz de reconocer 99.43% de los caracteres; para tratar exitosamente 90% de las placas, el sistema debe reconocer 98.83% de los caracteres, un porcentaje de error de 1.17%. Es decir, necesitamos un sistema como el descrito en [43].

El funcionamiento global del algoritmo se puede calcular usando la siguiente relación:

$$E_f = R_p R_s R_r \ , \tag{6.1}$$

donde E_f es la eficiencia total del algoritmo, R_p es el rendimiento del detector de placas, R_s es el rendimiento del segmentado y R_s es el rendimiento del sistema que clasifica las imágenes de caracteres y nos devuelve la cadena ASCII de los caracteres de la placa. De los datos expuestos en los párrafos anteriores, calculamos que $E_f = 22\%$. Una quinta parte de las imágenes serán correctamente reconocidas. Aunque parece un porcentaje muy bajo, hay que notar que estamos no estamos ya en condiciones ideales con imágenes bien contrastadas (fig. 63) , sino que tenemos imágenes del mundo real. (figs. 6.4 a 6.10).

Si calculamos E_f para R_p =90%, la eficiencia total se eleva a 56.36%. Aún mejorando el algoritmo de segmentado para que segmente todos los casos (o definiendo condiciones muy específicas de operación), y suponiendo 90% de eficacia en el reconocimiento de los caracteres, la eficiencia total llega hasta 78.9%.

Fig. 6.3 Algoritmo de segmentación y reconocimiento de caracteres para un caso ideal.

Los cálculos anteriores demuestran lo difícil que es diseñar un sistema eficaz para lograr leer un porcentaje razonable de placas en condiciones reales. Aunque es de esperarse que desarrollando nuevos algoritmo o mejorando los existentes podamos conseguir un mejor porcentaje.

6.2 Conclusiones y trabajo a futuro.

Es necesario seguir trabajando en varias direcciones diferentes:

- Modificar el algoritmo de detección de placa, para que pueda ser capaz de compensar variaciones locales de la imagen.

(ir a la página 134).

Fig. 6.4 Imagen original y resultado del algoritmo de reconocimiento de placas.

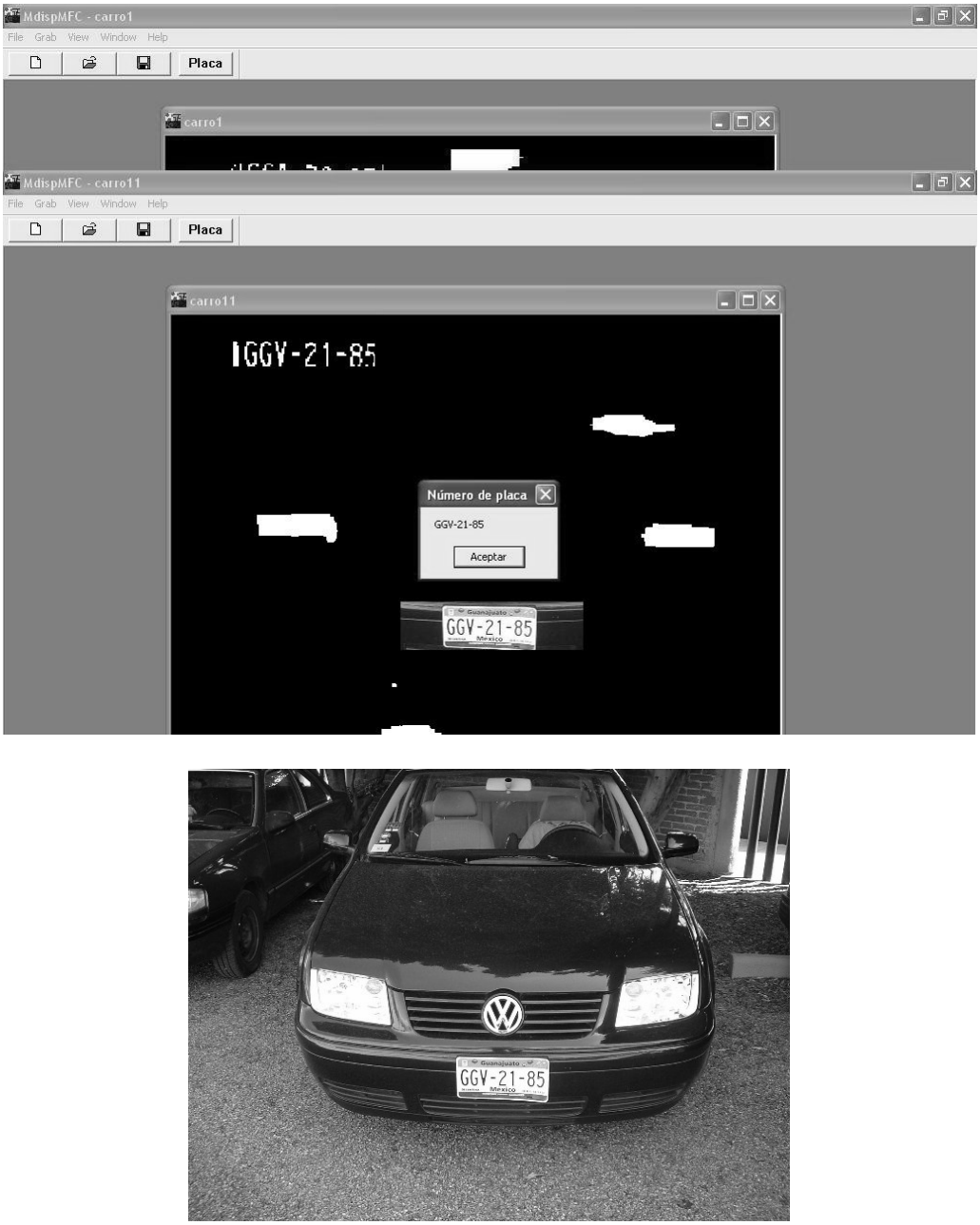


Fig. 6.5 Imagen original y resultado del algoritmo de reconocimiento de placas.

Fig. 6.6 Imagen original y resultado del algoritmo de reconocimiento de placas.

Fig. 6.7 Imagen original y resultado del algoritmo de reconocimiento de placas.

Fig. 6.8 Imagen original y resultado del algoritmo de reconocimiento de placas.

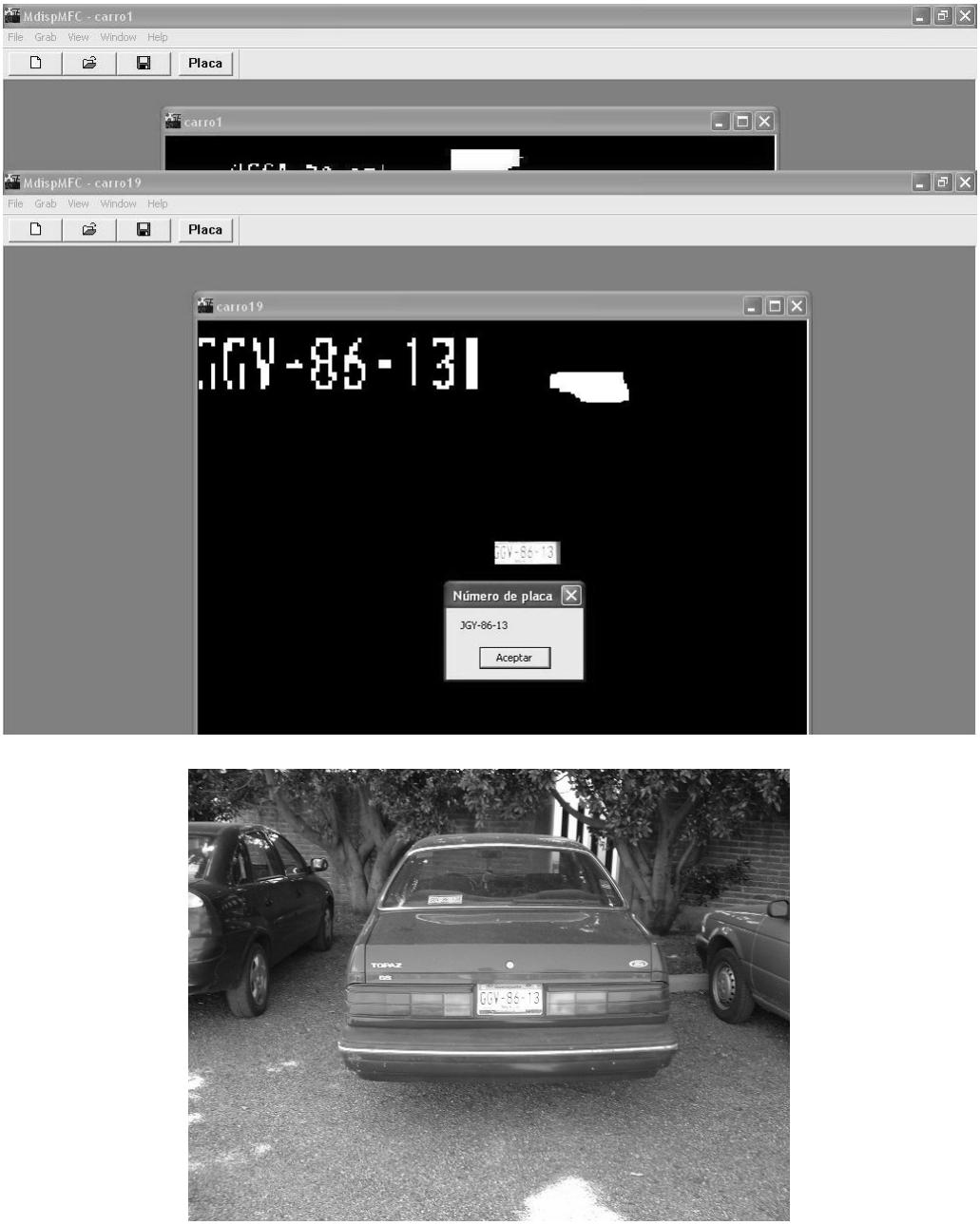


Fig. 6.9 Imagen original y resultado del algoritmo de reconocimiento de placas.

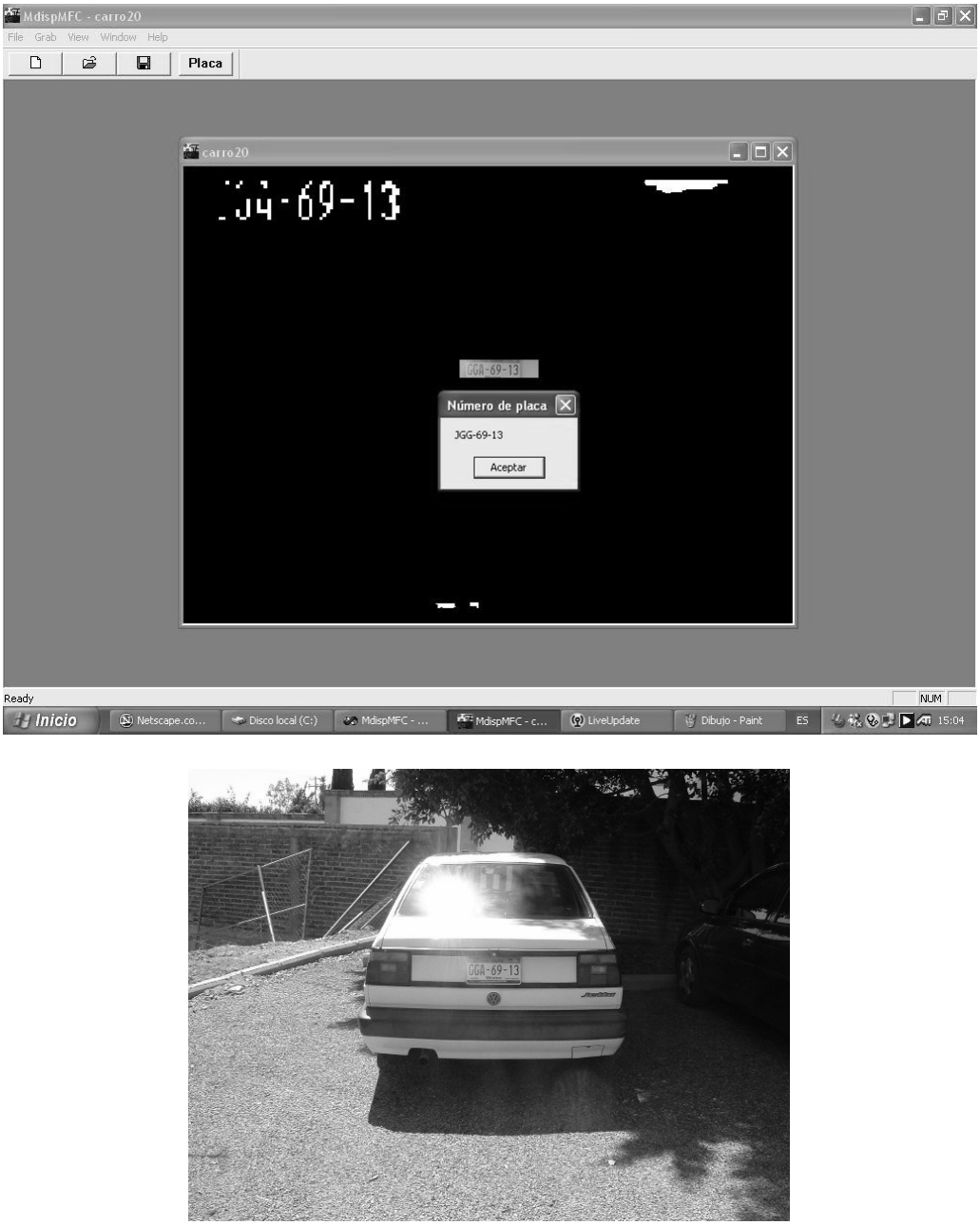


Fig. 6.10 Imagen original y resultado del algoritmo de reconocimiento de placas.

- Generalizar el algoritmo de selección de la tira de caracteres para que pueda trabajar sobre placas deformadas u observadas con perspectiva.
- Así mismo el algoritmo de segmentado. debe refinarse para tratar con cambios bruscos de intensidad como el mostrado en la figura 6.10.
- Generalizar el algoritmo de selección de la tira de caracteres para que pueda trabajar sobre placas deformadas u observadas con perspectiva.
- Así mismo el algoritmo de segmentado. debe refinarse para tratar con cambios bruscos de intensidad como el mostrado en la figura 6.10.
- Antes que introducir los caracteres a la red tal como llegan del proceso de segmentación, es conveniente buscar una representación más robusta. Podríamos intentar obtener el esqueleto y codificar el código de cadena; proyección en anillo [45], etc.
- El conjunto de caracteres usado para el entrenamiento de la red neuronal fue muy reducido, de sólo sesenta ejemplos. Se debe aumentar el conjunto de entrenamiento.
- Es posible que la arquitectura que estamos usando no es la adecuada para el problema. Con cuarenta elementos en la capa oculta podemos resolver perfectamente diez caracteres (números) y usando métodos de corrección de ambigüedades podemos obtener aproximadamente el mismo rendimiento para diecisiete letras incluyendo el guión. Pero si queremos llegar a un porcentaje de error de sólo 1%, para que al menos podamos reconocer todos los caracteres de una placa al menos en un 90% de los casos, es necesario modificar la arquitectura actual, ya sea aumentando el número de elementos en la capa oculta o cambiando de arquitectura.

Fig. 6.11 Imagen de placa, a gran distancia y en reposo.

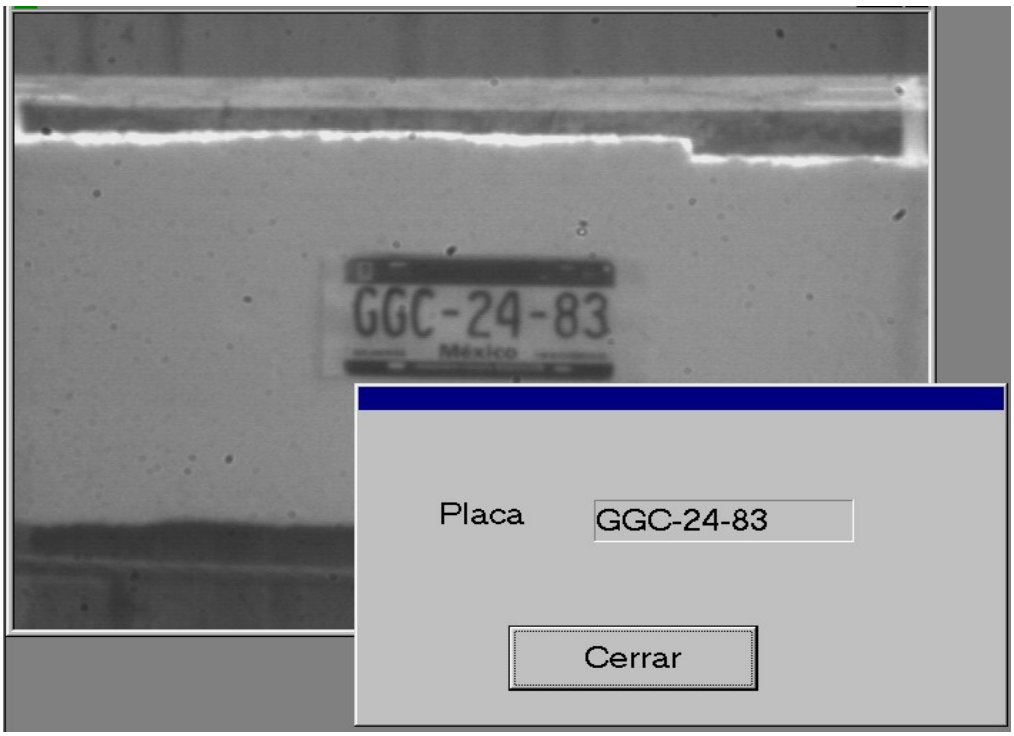


Fig. 6.12 Acercamiento y lectura de los caracteres.

En las figuras 6.11 y 6.12 mostramos el resultado obtenido con una versión anterior del programa para leer los caracteres en una placa colocada a gran distancia y sin movimiento. Aunque aún queda mucho trabajo por delante, con los resultados que tenemos ahora podemos dar el siguiente paso, y probar el sistema en condiciones reales, por ejemplo en la entrada de un estacionamiento y con autos moviéndose, por lo menos a bajas velocidades.

REFERENCIAS

[1] González, Woods, "Digital Image Processing", pag. 1, Addison Wesley, 1993.

[2] Idem.

[3] Ibid, pag. 2.

[4] Ibid, pag. 3.

[5] Plate Recognition, http://www.photocop.com/recognition.htm .

[6] "Number plate scan to be extended", BBC News, UK edition, 29 de mayo del 2003, http://news.bbc.co.uk/1/hi/uk_politics/2947136.stm

[7] Blake, Ashley, "Car Cloning", BBC Inside out, 10 de enero del 2005, http://www.bbc.co.uk/insideout/westmidlands/series7/car_cloning.shtml .

[8] Plate Recognition, http://www.photocop.com/recognition.htm .

[9] Fischer, Anne, "Vision System Performs License Plate Verification", Fotonics Spectra, Febrero del 2005, IEEE.

[10] Gonzalo Pajares, Jesús M. De la Cruz, "Visión por computador, Imágenes digitales y aplicaciones", pág. 8, Ed. Alfaomega – Ra-Ma, 2002.

[11] González, op. Cit. Pp. 8-10.

[12] Alcañiz y otros, "Procesamiento Digital de imagen", pp 66-69, Universidad Politécnica de Valencia, Servicio de Publicaciones, España, 1999.

[13] González, op. cit, pp. 81-111.

[14] Oppenheim, Willsky, Nawab, "Señales y Sistemas", 2da edición, pp. 284-422, Prentice Hall, México, 1997.

[15] Gonzalo Pajares, Jesús M. De la Cruz, op. cit., pp. 288-290.

[16] Idem.

[17] J.R. Parker, Pavol Federl, "An approach to License Plate Recognition", University of Calgary. http://pages.cpsc.ucalgary.ca/%7Efederl/Publications/LicencePlate1996/licence-plate-1996.pdf .

[18] González, Op. Cit, pp. 580-586.

[19] Proakis,Manolakis, "Tratamiento digital de señales", pág. 120, Prentice Hall, 1998.

[20] González, op. cit., p.585.

[21] Proakis,Manolakis, op.cit, pág. 121.

[22] Jaako Astola, Edward Dougherty, "Nonlinear filtres", *Digital image Processing Methods*, pp. 1-42, Editado por Edward R. Dougherty, Marcel Dekker Inc, EEUU, 1994.

[23] Pajares, De la Cruz, Op. Cit., pp. 146-158.

[24] Pajares, De la Cruz, Op. Cit., pág. 150.

[25] Proakis,Manolakis, op.cit, pp. 457-502.

[26] Díaz de Léon Santiago, Yáñez Márquez, "Introducción a la morfología matemática de conjuntos", Colección Ciencia de la Computación, Instituto Politécnico Nacional-Universidad Nacional A utónoma de México-Fondo de Cultura Económica, México, 2003.

[27] Pajares, De la Cruz, Op. Cit., pp. 188-190.

[28] Otsu, N., "A threshold selection method from grey-level histograms," IEEE Transactions Systems man, Cybernetics, Vol. 8, pp 62-66, 1978.

[29] Andrews, H., "Monochrome digital image enhancement," Optical Society of America, Applied optics, Vol. 15(2) pp. 495-503, 1976.

[30] Ping-Sung Liao, Tse-Sheng Chen, Pau-Choo Chung, "A fast algorithm for Multilevel Thresholding", Journal of information Science and Engineering 17, pp. 713-727, 2001.

[31] S. Impedovo, L. Ottaviano, S. Occhinegro, "Optical Character Recognition – A Survey ", pp. 353-382, Character & Handwritting Recognition, Editor P.S.P. Wang, World Scientific Series in Computer Science Vol. 30, 1991.

[32] I. Guyon. , "Applications of Neural Networks to Character Recognition", pp. 1-24, Character & Handwritting Recognition, Editor P.S.P. Wang, World Scientific Series in Computer Science Vol. 30, 1991.

[33] Guyon et al, "Comparing different neural networks architectures for classifying handwritting digits", *Proc. Int. Joint Conf. On Neural Networks , Volume II*, Washington DC, 1989, IEEE, pp. 127-132.

[34] Warren S. McCulloch y Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." Bulletin of Mathemathical Biophysics, 5:115-133,1943.

[35] James A. Anderson y Edward Rosenfeld (Editores), Neurocomputing:Foundations of Research. MIT Press, Cambridge, MA, 1988.

[36] Donald O. Hebb. "The Organization of Behavior", p. 50, Wiley, Nueva York, 1949.

[37] James A. Freeman, David M. Skapura, "Redes neuronales: algoritmos, aplicaciones y técnicas de programación", p. 20, Addison-Wesley Iberoamericana, 1993.

[38] Ibid. pág. 23.

[39] Marvin Minsky, Seymour Papert, "Perceptrons" MIT Press, Cambridge, MA, 1969.

[40] Geoffrey E. Hinton, Terrence J. Sejnowski. Neural network architectures for AI. Tutorial nº MP2, AAAI87, Seattle, WA, Julio de 1987.

[41] Freeman, Skapura, Op. Cit, pag. 96.

[42] Amescua Fonseca, Juan Carlos, "Redes Neuronales en el reconocimiento de patrones", pp. 76-78, Tesis de la Licenciatura en Ingeniería en computación y sistemas, Universidad del Bajío, León, Gto., 1995.

[43] Russel, Norvig, *Inteligencia artificial, un enfoque moderno;*2da edición, pág. 856, Pearson Prentice Hall , 2004.

[44] Zhou, Pavlidis, "Disambiguation of Characters by a Second Stage Classifier", Proceedings SPIE, *Character Recognition Technologies*, Vol. 1906, pp. 166-171.

[45] Tang, Cheng, Suen, "Transformation-Ring-Projection (TRP) algorithm ans its VLSI Implementation ", *Proc. Int. Joint Conf. On Neural Networks , Volume II*, Washington DC, 1989, IEEE, pp. 25-56.

BIBLIOGRAFÍA

- Alcañiz et al, *Procesamiento Digital de imagen*, Universidad Politécnica de Valencia, Servicio de Publicaciones, España, 1999.
- Amescua Fonseca, Juan Carlos, "Redes Neuronales en el reconocimiento de patrones", Tesis de la Licenciatura en Ingeniería en computación y sistemas, Universidad del Bajío, León, Gto., 1995.
- Andrews, H., "Monochrome digital image enhancement," Optical Society of America, *Applied optics, Vol. 15(2)* pp. 495-503, 1976.
- BBC news, http://news.bbc.co.uk/ .
- Díaz de León Santiago, Yáñez Márquez, *Introducción a la morfología matemática de conjuntos*, Colección Ciencia de la Computación, Instituto Politécnico Nacional-Universidad Nacional Autónoma de México-Fondo de Cultura Económica, México, 2003.
- *Digital image Processing Methods*, Editado por Edward R. Dougherty, Marcel Dekker Inc, EEUU, 1994
- Donald O. Hebb. *The Organization of Behavior*, p. 50, Wiley, Nueva York, 1949.
- Fotonics Spectra, Febrero del 2005, IEEE.
- González, Woods, *Digital Image Processing*, Addison-Wesley, septiembre 1993.
- Gonzalo Pajares, Jesús M. De la Cruz, *Visión por computador, Imágenes digitales y aplicaciones*, Ed. Alfaomega – Ra-Ma, 2002.
- Guyon et al, "Comparing different neural networks architectures for classifying handwritting digits", *Proc. Int. Joint Conf. On Neural Networks , Volume II*, Washington DC, 1989, IEEE, pp. 127-132.

- I. Guyon. , “Applications of Neural Networks to Character Recognition”, pp. 1-24, *Character & Handwriting Recognition*, Editor P.S.P. Wang, World Scientific Series in Computer Science Vol. 30, 1991.
- James A. Anderson y Edward Rosenfeld (Editores), *Neurocomputing:Foundations of Research.* MIT Press, Cambridge, MA, 1988.
- Ley de tránsito y transporte del Estado de Guanajuato.
- Marvin Minsky, Seymour .Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1969.
- Oppenheim, Willsky, Nawab, *Señales y Sistemas*, 2da edición, Prentice Hall, México, 1997.
- Otsu, N., “A threshold selection method from grey-level histograms,” IEEE Transactions Systems man, Cybernetics, Vol. 8, pp 62-66, 1978.
- Ping-Sung Liao, Tse-Sheng Chen, Pau-Choo Chung, “A fast algorithm for Multilevel Thresholding”, Journal of information Science and Engineering 17, pp. 713-727, 2001.
- Plate Recognition, http://www.photocop.com/recognition.htm.
- Proakis,Manolakis, *Tratamiento digital de señales*, Prentice Hall, 1998.
- Reglamento de Tránsito de la Ley de Tránsito y Transporte del Estado de Guanajuato.
- Russel, Norvig, *Inteligencia artificial, un enfoque moderno;*2da edición, Pearson Prentice Hall , 2004.
- S. Impedovo, L. Ottaviano, S. Occhinegro, “Optical Character Recognition – A Survey ”, pp. 1-24, *Character &Handwritting Recognition*, Editor P.S.P. Wang, World Scientific Series in Computer Science Vol. 30, 1991.

- Tang, Cheng, Suen, "Transformation-Ring-Projection (TRP) algorithm ans its VLSI Implementation", *Proc. Int. Joint Conf. On Neural Networks , Volume II*, Washington DC, 1989, IE EE, pp. 25-56.
- Warren S. McCulloch y Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." Bulletin of Mathemathical Biophysics, 5:115-133,1943.
- Zhou, Pavlidis, "Disambiguation of Characters by a Second Stage Classifier", Proceedings SPIE, *Character Recognition Technologies*, Vol. 1906, pp. 166-171.

APÉNDICE

Listado de programas para redes neuronales

Código de Borland C++ para la implementación y entrenamiento de una red neuronal.

```
//Definici¢n de las dimensiones de la red
//N£mero de vectores de entrenamiento
#define m 60
//tana¤o del vector de entrada
#define n 400
//Numero de elementos de procesamiento (neuronas) ocultas
#define noc 40
//Numero de neuronas de salida
#define nsal 24
//Parametro de la velocidad de aprendizaje
#define eta 0.01
//Error permitido. Est  en relaci¢n inversa con el tama¤o del vector de
//salida
#define permitido 0.5
//Bias de los elementos de procesamiento
#define bias 1
#define bias2 1
//Factor de expansion de la funcion sigmoide
#define T 1
```

//Declaraci¢n de variables

```
int h,i,j,k; /*indices del vector de entrada, de neuronas ocultas y
                de neuronas de salida, respectivamente*/
double iter; //Lleva la cuenta del n£mero de iteraciones.
double x[n],Yo[noc],Ysal[nsal];
/*Definici¢n de los vectores de entrada, vector de salida de las neuronas
ocultas y de las neuronas de salida, respectivamante*/
double Osal[nsal]; //Vector de valores de salida deseados
double Vent[m][n],Vsal[m][nsal]; //Vector de valores deseados de entrada
double Woc[n][noc],Wsal[noc][nsal]; //Pesos de las capas ocultas y de salida
double Ep,Ep3; //Error total del sistema
```

//Elemento de procesamiento (neurona).

```
void neurona ()
{   double REDoc[noc],REDsal[nsal]; /*Valor de RED para cada neurona de la capa oculta
                        y de la capa de salida, respectivamente*/

        //Calculando los valores de la capa oculta
        for(j=0;j<noc;j++)
        {       REDoc[j]=0;
                for(i=0;i<n;i++)
                        REDoc[j]=REDoc[j]+(Woc[i][j]*x[i])/300;/*Calculando la suma de
                        los productos de los pesos y las entradas*/
                REDoc[j]=REDoc[j]+ bias; //agregando el termino de tendencia
                Yo[j]=1/(1+exp((-1)*REDoc[j]/T));//aplicando la funcion sigmoide
        }

        //Calculando los valores de la capa de salida
        for(k=0;k<nsal;k++)
        {  REDsal[k]=0;
                for(j=0;j<noc;j++)
                        REDsal[k]=REDsal[k]+(Wsal[j][k]*Yo[j]);/*Calculando la suma de
                        los productos de los pesos y las entradas*/
                REDsal[k]=REDsal[k]+ bias2; //agregar el termino de tendencia 2
                Ysal[k]=1/(1+exp((-1)*REDsal[k]/T));//aplicando la funcion sigmoide
        }
}
```

//Algoritmo de retropropagación para el entrenamiento de la red.

```
void backpropagation(char *nombre1,char *nombre2,int lim,int iteraciones)
{  double dsal[nsal],doc[noc];/*vectores de error de la capa de salida y de la
                                capa oculta, respectivamente*/
        double acum,Ep2,Ep4,z;//Definici¢n del acumulador y error intermedio
        //Inicializacion aleatoria de los pesos de las capas ocultas y de salida
        for(i=0;i<n;i++)
                for(j=0;j<noc;j++)
                {       z=random(200);
                        Woc[i][j]=z/100 - 1;
                        //Valor aleatorio entre 0 y 1
                }
        for(j=0;j<noc;j++)
                for(k=0;k<nsal;k++)
                {       z=random(200);
                        Wsal[j][k]=z/100 - 1;
                        //Valor aleatorio entre 0 y 1
                }
```

```
        //Inicializando los vectores de errores en ceros
        for(j=0;j<noc;j++)
                doc[j]=0;
        for(k=0;k<nsal;k++)
                dsal[k]=0;
        iter=0;
        Ep3=100;
        Ep=0;

        do
{       Ep4=Ep3;
                Ep3=0;
        for(h=0;h<lim;h++)
{
        for(i=0;i<n;i++)
                x[i]=Vent[h][i];
        for(k=0;k<nsal;k++)
                Osal[k]=Vsal[h][k];

        neurona();
        //Calculo del error de salida
        for(k=0;k<nsal;k++)
                dsal[k]=(Osal[k]-Ysal[k])*Ysal[k]*(1-Ysal[k]);

        //Calculo del error de la capa oculta
        for(j=0;j<noc;j++)
        {  acum=0;
                for(k=0;k<nsal;k++)
                        acum=acum+(dsal[k]*Wsal[j][k]);
                doc[j]=Yo[j]*(1-Yo[j])*acum;
        }

        //Actualizaci¢n de los pesos de la capa de salida
        for(j=0;j<noc;j++)
                for(k=0;k<nsal;k++)
                        Wsal[j][k]=Wsal[j][k] + (eta*dsal[k]*Yo[j]);

        //Actualizaci¢n de los pesos de la capa oculta
        for(i=0;i<n;i++)
                for(j=0;j<noc;j++)
                        Woc[i][j]=Woc[i][j] + (eta*doc[j]*x[i]);
        Ep=0;
        //Calculo del error
        for(k=0;k<nsal;k++)
        {       Ep2=Osal[k]-Ysal[k];
                Ep=pow(Ep2,2)+Ep;
        }
```

```
        Ep=Ep/2;
        Ep3=Ep+Ep3;
        iter++;
}  w4. printf(10,105,"Ep4= %lf ",Ep4);
        Ep4=abs(Ep4-Ep3)*100/Ep4;
        iter++;
}while(iter<iteraciones);

        guardarbin(nombre1,nombre2);
        w4. printf(10,15,"Error= %lf",Ep3);
        w4. printf(10,45," Porcentaje Error= %lf",Ep4);
        w4. printf(10,75,"Num. iteraciones= %lf",iter);
}
```

//Reconocimiento de caracteres usando red neuronal.

```
void CMdispMFCView::ReconocerCaracter(double * I, char *resultado, int w)
{
        float mayor = 0;
        float x[400] ;

        double resul[27];

        int p,i,j;
        for(i=0;i<16;i++)
                for(j=0;j<25;j++)
                        x[25*i+j]=float(I[i+16*j]);

        if(w<G[0])
{
        Neurona(x,Ysal,2);

        for(i=0;i<10;i++)
                resul[i]=0;

        for(i=10;i<nsal;i++)
                resul[i]=Ysal[i];


         mayor=0;
         for(i=0;i<nsal;i++)
         {
                if(resul[i]>mayor)
                  {       mayor=float(resul[i]);
                          p=i;
                  }
         }
```

```
   if(p==10||p==11||p==12||p==17||p==18||p==20||p==22||p==24||p==26)
        {
                Neurona(x,Ysal,4);


        for(i=0;i<10;i++)
                resul[i]=0;


     for(i=10;i<nsal;i++)
                resul[i]=Ysal[i];


        mayor=0;
        for(i=0;i<nsal;i++)
                if(resul[i]>mayor)
                {       mayor=float(resul[i]);
                  p=i;
                 }
        }

        if(p==11||p==12||p==13||p==15||p==22||p==24)
  {
           Neurona(x,Ysal,3);

                for(i=0;i<10;i++)
                        resul[i]=0;


                 for(i=10;i<nsal;i++)
                        resul[i]=Ysal[i];


                  mayor=0;
                 for(i=0;i<nsal;i++)
                        if(resul[i]>mayor)
                        {       mayor=float(resul[i]);
                                p=i;
                        }
  }
}

        if(w>G[0])
{
        Neurona(x,Ysal,1);
```

```
  for(i=0;i<10;i++)
        resul[i]=Ysal[i];

        for(i=10;i<nsal;i++)
        resul[i]=0;


        mayor=0;
        for(i=0;i<nsal;i++)
        {       if(resul[i]>mayor)
                {       mayor=float(resul[i]);
                        p=i;
                }
        }
}
        resultado3[w-1] = mayor;

if(w==G[0]+3 ||  w==G[0])
        p=14;


  switch(p)
        {       case 0:
                resultado[w-1]='0';
        break;
                case 1:
        resultado[w-1]='1';
        break;
                case 2:
                resultado[w-1]='2';
        break;
                case 3:
                resultado[w-1]='3';
        break;
                case 4:
                resultado[w-1]='4';
        break;
                case 5:
                resultado[w-1]='5';
        break;
                case 6:
                resultado[w-1]='6';
        break;
                case 7:
                resultado[w-1]='7';
        break;
                case 8:
```

```
        resultado[w-1]='8';
break;
        case 9:
        resultado[w-1]='9';
break;
        case 10:
        resultado[w-1]='C';
break;
        case 11:
        resultado[w-1]='D';
break;
        case 12:
        resultado[w-1]='E';
break;
        case 13:
        resultado[w-1]='G';
break;
        case 14:
        resultado[w-1]='-';
break;
        case 15:
        resultado[w-1]='H';
break;
        case 16:
        resultado[w-1]='J';
break;
        case 17:
        resultado[w-1]='P';
break;
        case 18:
        resultado[w-1]='V';
break;
        case 19:
        resultado[w-1]='W';
break;
        case 20:
        resultado[w-1]='Z';
break;
        case 21:
        resultado[w-1]='A';
break;
        case 22:
        resultado[w-1]='B';
break;
        case 23:
        resultado[w-1]='T';
break;
```

```
            case 24:
        resultado[w-1]='F';
        break;
            case 25:
        resultado[w-1]='U';
        break;
            case 26:
        resultado[w-1]='Y';
        break;
            case 27:
        resultado[w-1]='?';
        break;
   }
}
```