UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

RECONOCEDOR ÓPTICO DE CARACTERES DE PLACAS DE AUTOMÓVILES EMPLEANDO TÉCNICAS DE PROCESAMIENTO DIGITAL DE IMÁGENES

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRÓNICO

PRESENTADO POR:

GIANCARLO AYALA CHARCA

PROMOCIÓN

2001-I

LIMA – PERÚ

2007

RECONOCEDOR ÓPTICO DE CARACTERES DE PLACAS DE AUTOMÓVILES EMPLEANDO TÉCNICAS DE PROCESAMIENTO DIGITAL DE IMÁGENES

A mi Madre, gracias por tu aliento y fuerzas
que me distes para seguir adelante.

A mi Padre, gracias por enseñarme a creer que la
perseverancia es el camino para llegar a
nuestros objetivos mas anhelados.

Al Doctor Jorge Alberto del Carpio Salinas,
gracias por creer en mi,
la confianza, el apoyo, todos sus consejos y
orientaciones que hicieron que este proyecto
se haga realidad.

A mis amigas y amigos, todos aquellos que estuvieron
conmigo, quienes me ayudaron, me dieron su consejo
y apoyo incondicional durante mi estancia en el CID-FIEE,
y de manera especial a Gustavo Samuel Huamán Bustamante,
José Antonio Huamán Layme, Amilcar Messco Mizares
y la Srta. Shirley Guillén Saravia.
Muchas gracias.

A la UNI, por darme la oportunidad de formarme
como profesional y poder contribuir con mi país.

SUMARIO

El presente trabajo propone el desarrollo de un sistema reconocedor de caracteres de placas de automóviles a partir de una imagen fija que puede provenir de una cámara fotográfica o de video. El esquema de trabajo está estructurado por etapas, por ello, luego de la adquisición y un preprocesado, el sistema procede a localizar la placa en la imagen. Una vez ubicada se corrige la perspectiva que pudiera tener, se calcula un umbral de gris para la binarización de la imagen y entonces se procede a la extracción de los caracteres a través de la segmentación. Es necesario reducir los caracteres a matrices de 16 x 16, para lo que empleamos Geometría Proyectiva. Para el reconocimiento de los caracteres primero extraemos sus patrones en vectores comprimidos con alta concentración de energía por medio de la Transformada de Karhunen-Lòeve, para finalmente a través de un Clasificador Neuronal (Red Neuronal Backpropagation) poder reconocer los caracteres de la placa. El sistema-prototipo implementado ha dado buenos resultados y fue desarrollado completamente en Matlab.

INDICE

INTRODUCCIÓN

Ante los problemas de seguridad, como robos, asaltos, secuestros, accidentes, que pueden haber en nuestro medio surge la necesidad de sistemas que proporcionen información inmediata de la identificación de un vehículo, de manera que se pueda tomar las acciones correspondientes en el momento en que ocurren los acontecimientos.

De esta manera se desarrolla el sistema de reconocimiento de placas de automóviles, con el propósito de dar una alternativa de solución a estos problemas que aquejan a nuestra sociedad. Este sistema que utiliza el procesamiento de imágenes viene a ser una aplicación de los Sistemas de Visión Artificial.

Los sistemas de Visión Artificial tienen una gran diversidad de aplicaciones como en automatización industrial, robótica, cartografía, análisis forense, imágenes de radar, sensado remoto, análisis de imágenes médicas, y reconocimiento de caracteres. Es en este último en el que se encuentra enmarcado el presente trabajo.

El reconocimiento de caracteres en imágenes procedentes de textos es un problema que se ha ido estudiando a lo largo de varios años atrás, con diversidad de trabajos publicados. El problema se presenta cuando estas imágenes provienen de escenas reales donde la iluminación, la distribución de los objetos y la cámara utilizada para la adquisición, constituyen factores muy importantes que influyen en la correcta adquisición de la imagen.

Es importante pues superar estos inconvenientes, tomando las previsiones a partir de la instalación, así como utilizar algoritmos robustos para superar aquellos problemas relacionados con la iluminación y la perspectiva.

En Norteamérica y Europa existen sistemas automatizados comerciales que a través de una cámara de video y un software de procesamiento de imágenes se extrae la información de la placa de automóvil para identificar el vehículo y realizar una acción en particular. A nivel de Latinoamérica se ha visto sistemas comerciales en Brasil. A nivel académico algunos trabajos relacionados a estos sistemas provenientes de Colombia, México y España. Es a partir de un capítulo de una Tesis Doctoral precisamente de éste último país **[4]**, que sirve de base para desarrollar el sistema, encontrando en el camino

otros inconvenientes que se fueron superando poco a poco. En ese sentido el presente trabajo se diferencia de aquel de la referencia en las diferentes etapas puesto que se hicieron cambios que consideramos ofrecerían mejores resultados, como por ejemplo, la utilización adicional de otro concepto como la densidad de gradientes para la ubicación de la placa, la corrección de perspectiva de la imagen y no una simple corrección de inclinación, la reestructuración del histograma para binarización de placas amarillas, azules o con zonas oscuras en general, la reducción de tamaño de los caracteres extraídos usando transformación proyectiva y no diezmado, y así también nuestra propia base de datos de fotografías tiene bastante variedad de casos de placas con caracteres despintados, marcos adicionales, los casos de distorsión de imagen en perspectiva, escenas complejas con otras figuras, etc.

El desarrollo del trabajo ha sido estructurado de la siguiente manera, inicialmente se estudiaron todos los aspectos a tener en cuenta para una buena adquisición de las imágenes considerando aquellos problemas de la iluminación y la perspectiva. Posteriormente se tomó una base de 145 fotografías, que sirvieron para el desarrollo de los algoritmos, con diversidad de distancias y ángulos todo esto se muestra en el Capítulo 1. Fue necesario hacer un pre-procesado de la imagen **[8]** para la aplicación del algoritmo de la siguiente etapa, Capítulo 2, y a continuación comenzamos por la ubicación de la placa en la fotografía, contemplando la complejidad de la imagen, como aquellos stickers, símbolos y figuras que representan perturbaciones y podrían confundir al algoritmo de localización de placa, su desarrollo lo vemos en el Capítulo 3.

La corrección de la inclinación es parte importante también aquí, ya que los caracteres deben quedar bien alineados con la horizontal, en esa fase se encontró el problema de la distorsión de la imagen por perspectiva. Fue necesario entonces utilizar la Transformación Proyectiva para corregir la distorsión y combinar con la Transformada de Hough para determinar las coordenadas de las esquinas de la placa, 4 puntos vitales para la aplicación de esta transformación de forma automática, como lo veremos en el Capítulo 4.

Cuando se tiene la placa corregida correctamente sobre un plano frontal, pasa por un proceso de “binarización” que consiste en convertir la imagen de escala de grises a dos niveles blanco y negro. Aquí se superan los problemas con placas oscuras (amarillas y azules) y de otro lado aquellos con una iluminación no uniforme que pueden conllevar a grandes problemas, Capítulo 5. A partir de aquí ya se encuentra lista para comenzar el proceso de segmentación de los caracteres, Capítulo 6, ello se realiza utilizando

proyecciones y por aproximaciones sucesivas, tomando algunas consideraciones para discriminar los caracteres de otros elementos diferentes.

En el Capítulo 7, una vez extraídos los caracteres serán acondicionados y pasan a un proceso de extracción de patrones de manera de obtener un patrón de cada carácter, y poder reconocerlos utilizando una red neuronal, para ello es necesario previamente que el clasificador de patrones (red neuronal) sea entrenado con los patrones extraídos.

Culminado el entrenamiento, el sistema queda listo para efectuar la identificación automática de los caracteres. Dependiendo de las aplicaciones que se le den, el sistema va a tener algunas variantes, ello se expone en la parte final del presente trabajo de Tesis como parte de las conclusiones.

En la figura 1 se muestra un diagrama de bloques en el que se ilustra el esquema básico de todo el procesamiento de la imagen de entrada hasta la obtención de los caracteres. En cada etapa hubo diversos problemas que se fueron superando con el curso de la investigación y el desarrollo de los algoritmos como veremos a continuación.

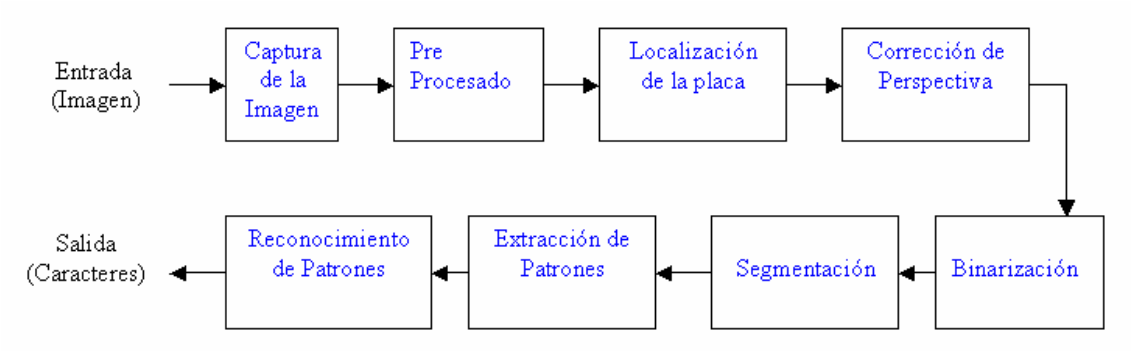


Figura 1. Diagrama de bloques del sistema

CAPÍTULO I
CAPTURA Y BASE DE DATOS DE IMÁGENES

En esta primera parte vamos a exponer todos los aspectos que se deben tener en cuenta para una adecuada captura de imágenes. Este tratado fue importante en el desarrollo del trabajo ya que nos brinda una mejor visión de todo el proceso de adquisición de la imagen considerando la cámara a utilizar, la distribución de la iluminación y la posición relativa más adecuada para tomar una fotografía, dependiendo pues del objeto que se quiere capturar en una imagen. En este apartado también tratamos los aspectos considerados para obtener las fotografías de la base de datos que sirvieron para el desarrollo del sistema y todos los algoritmos involucrados.

1.1 Adquisición de la Imagen

1.1.1 Consideraciones en cuanto a Cámaras

No es la intención hacer una descripción detallada del interior de una cámara fotográfica sino mas bien mostrar cuan importante es el proceso que se da al interior de la cámara en la captura de la imagen de manera que conociendo esto tenemos un mejor criterio para hacer una elección más adecuada con mejor conocimiento de los parámetros implicados.

a. Partes de una Cámara Fotográfica

La Cámara, es un mecanismo mediante el cual, una película (en cámaras analógicas) o un sensor CCD (en cámaras digitales) se expone a la luz reflejada por los objetos, generándose sobre aquel una imagen de éstos. Las cámaras se componen de 4 elementos básicos: el cuerpo, el objetivo, el obturador y el diafragma (Fig. 1.1).

El cuerpo es una pequeña cavidad hermética a la luz, donde se aloja la película (o el sensor CCD) para su exposición.

El objetivo es un conjunto de lentes ópticos de cristal, alojado en un anillo metálico, permite al fotógrafo enfocar una imagen sobre la película. Los objetos situados a diferentes distancias de la cámara pueden enfocarse con precisión al ajustar la distancia entre el objetivo y la película fotográfica.

El obturador es un dispositivo mecánico, dotado con un muelle, que sólo deja pasar la luz a la cámara durante el intervalo de exposición. La mayoría de las cámaras modernas poseen obturadores de diafragma o de plano focal. Algunas cámaras antiguas para aficionados utilizan el obturador de guillotina, que consiste en una pieza con bisagra que al disparar se abre y expone la película alrededor de 1/30 de segundo.

El diafragma es una abertura circular situada detrás del objetivo, funciona en sincronía con el obturador para dejar pasar la luz a la cámara oscura. Esta abertura puede ser fija, como en muchas cámaras para aficionados, o regulable. Los diafragmas regulables consisten en laminillas de metal o de plástico superpuestas, que cuando se separan por completo forman una abertura del mismo diámetro del objetivo, y cuando se cierran dejan un pequeño orificio detrás del centro del objetivo. Entre la máxima abertura y la mínima, la escala de diafragmas se corresponde con una clasificación numérica, llamada **f-stops**, situada en la cámara o en el objetivo.

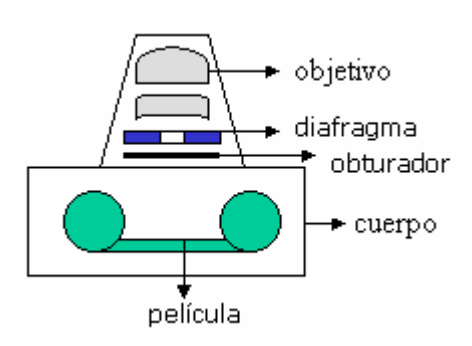


Figura 1.1 Partes principales de una cámara

b. Control de Exposición

Al regular la velocidad del obturador y la abertura del diafragma el fotógrafo consigue la cantidad exacta de luz para asegurar una correcta exposición de la película. La velocidad del obturador y la abertura son directamente proporcionales: si incrementamos la velocidad del obturador en una unidad, cambiará un **f-stop**. Al modificar en un punto la exposición se produce un cambio en la velocidad de obturación y en el diafragma, cuyo resultado será que la cantidad de luz que llegue a la película sea la misma. De esta manera, si se aumenta la velocidad del obturador el diafragma deberá ser aumentado en la misma medida para permitir que idéntica cantidad de luz llegue a la película. Los obturadores rápidos, de 1/125 segundo o menos, pueden captar objetos en movimiento.

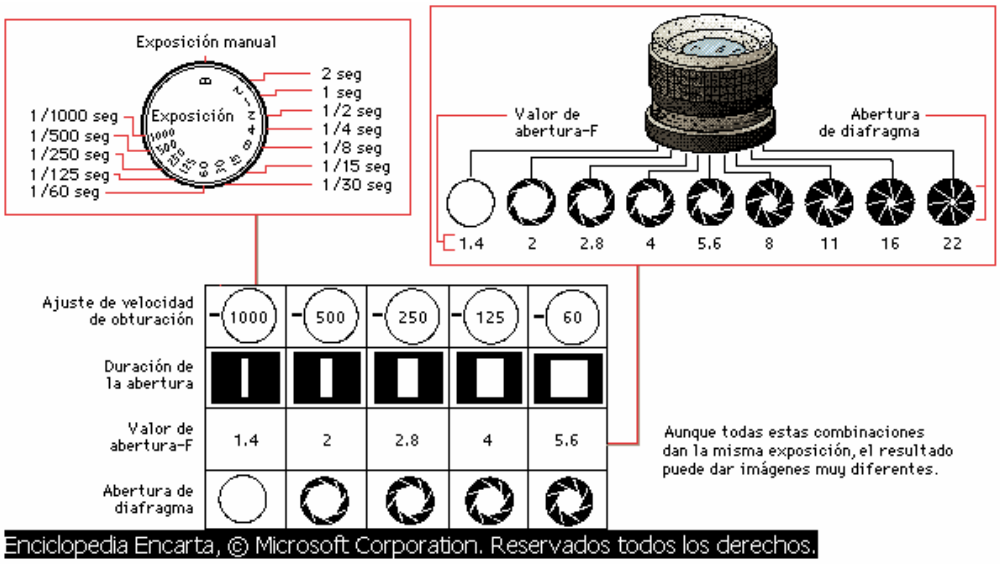


Figura 1.2. Control de la exposición **[1]**

Además de regular la intensidad de la luz que llega a la película, la abertura del diafragma se utiliza también para controlar la profundidad de campo **[2]**, también llamada zona de enfoque, que es la distancia entre el punto más cercano y más lejano del sujeto que aparece nítido en una posición determinada del enfoque. Al disminuir la abertura la profundidad de campo crece, y al aumentarla disminuye. Cuando se desea una gran profundidad de campo, es decir, la máxima nitidez de todos los puntos de la escena (desde el primer al último plano), se utiliza una abertura pequeña y una velocidad de obturación más lenta.

Como para captar el movimiento se necesita una gran velocidad de obturación, y en compensación una gran abertura, la profundidad de campo se reduce. En muchas cámaras el anillo del objetivo tiene una escala de profundidad de campo que muestra aproximadamente la zona de enfoque que se corresponde con las diferentes aberturas.

Debemos mencionar que la apertura del diafragma se mide mediante el valor de apertura **F** que viene a ser el cociente de la longitud focal (f) de la lente y el diámetro del diafragma (D), por lo que un valor como F = 1.4 es de mayor abertura que un valor de F = 16, que representa una abertura menor, ver Fig. 1.2.

$$F = \frac{f}{D} \qquad \text{Ec. 1.1}$$

Así también, en cuanto a la velocidad de obturación esta puede variar, de acuerdo a las cámaras, desde la posición Bulbo “b” a 1/500 seg. o más. En la posición Bulbo, el

obturador permanece abierto todo el tiempo que este presionado el disparador. Se pueden hacer exposiciones de varios minutos o también de horas, pero para eso se utiliza normalmente un cable disparador para no correr el riesgo de mover la cámara con la mano y además sería bastante incómodo estar horas presionando el disparador de la cámara. A continuación algunos valores comunes.

Escala de Velocidades:

b - 1 - 1/2 - 1/4 - 1/8 - 1/15 - 1/30 - 1/125 - 1/250 - 1/500

La siguiente figura es una aplicación que se le puede dar a la posición Bulbo.

Figura 1.3. Fotografía que muestra el efecto de mantener el obturador abierto durante un largo periodo de tiempo

En las 2 fotografías siguientes se puede ver la diferencia en cuanto a la profundidad de campo. En la primera, la apertura del diafragma es menor, la cantidad de luz se compensa con velocidad de obturación menor y se obtiene mayor profundidad de campo. En la segunda una apertura de diafragma mayor disminuye la profundidad de campo, la exposición se compensa con velocidad de obturación mayor.

Cabe mencionar que en las cámaras en general se suele tener una notación en cuanto al número **F** que acostumbran colocarlo por ejemplo como: **1:2.8** que representa **1/2.8**, a veces se coloca **f/2.8** y significa lo mismo son notaciones que utilizan algunos fabricantes todas ellas indican un F = 2.8. Así también la longitud focal de la lente se expresa como: f = 4.5mm, como se ve en la figura 1.5.

F = 16

Velocidad de Obturación = 1/30

F = 8

Velocidad de Obturación = 1/500

Figura 1.4. Control de la exposición combinando Apertura de Diafragma (F) y Velocidad de Obturación.

Figura 1.5. Cámara fotográfica mostrando el número **F** de la Apertura del diafragma y la longitud focal **f**.

Para las tomas fotográficas se decidió utilizar una cámara digital, que tenga flexibilidad en cuanto al control de la exposición, estas cámaras lo realizan automáticamente, así como el enfoque, que viene a ser el desplazamiento de la lente para modificar su longitud focal hasta que la imagen se haga nítida que también lo hacen automáticamente. Adicionalmente las cámaras digitales tienen la ventaja que éstas nos entregan la fotografía ya digitalizada en un archivo, para poder procesarla directamente, lo que nos evita utilizar un escáner si fuera con fotografías de cámaras analógicas.

Con todos estos criterios se eligió la cámara digital PowerShot A10, que tiene un shutter máximo de 1/1500 y permite tomar fotografías de autos con velocidad relativamente alta, adicionalmente el zoom óptico nos permite mayor flexibilidad durante

la toma fotográfica. El formato de la foto es en JPG. Las especificaciones se muestran en el Anexo C.

Figura 1.6. Cámara Digital PowerShot A10 utilizada para la obtención de las fotografías.

1.1.2 Consideraciones en cuanto a la Iluminación

En la toma de las imágenes comprendimos cuan importante y sensible es la imagen que se quiere captar a la iluminación. Como cuestiones importantes podríamos destacar las siguientes:

- Debemos establecer unas condiciones de iluminación que eviten gradientes de luz sobre la placa (eso invalidaría la elección de umbral). Otro problema a evitar es que la reflexión directa de la luz del foco en la placa alcance la cámara. Ese problema prácticamente borra un trozo de la placa. Una configuración ideal de foco y cámara sería la siguiente:

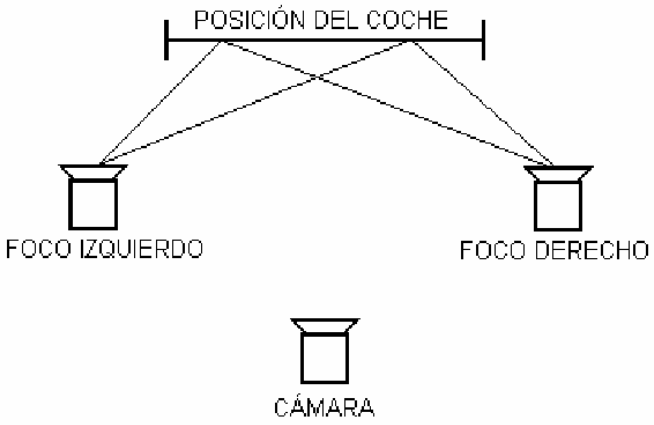


Figura 1.6. Distribución adecuada de iluminación y cámara.

- La superposición de luz de dos focos evita los gradientes. Además, las posiciones de los focos son tales que las reflexiones nunca llegan a la cámara. Las condiciones del lugar de instalación no siempre posibilitan la configuración óptima. Una colocación que casi siempre es posible es la siguiente:

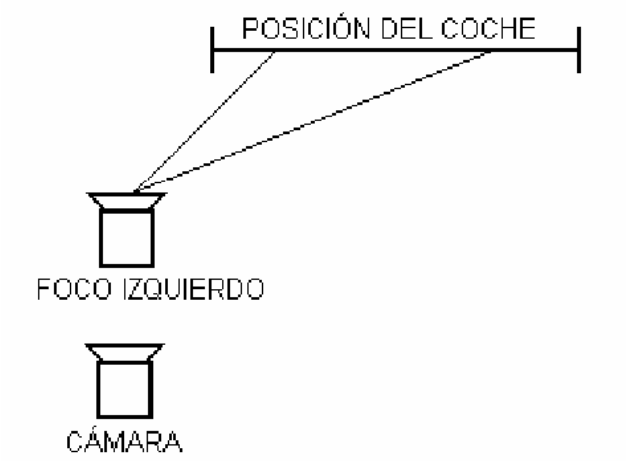


Figura 1.7. Distribución opcional de iluminación y cámara.

Figura 1.8. En esta fotografía se observa el brillo en diferentes partes del vehículo debido a que la cámara recibe directamente la luz reflejada, estos gradientes alteran la uniformidad de la imagen y por consiguiente su histograma.

- Para la toma de las fotografías se tuvo en cuenta durante el día la proyección de la luz solar evitando que se produzcan reflexiones o demasiado brillo sobre el automóvil o la placa, esto puede generar gradientes no uniformes de luz. En la noche la iluminación puede ser artificial utilizando un reflector o la otra opción sería usar una cámara con visor infrarrojo capaz de poder ver a total oscuridad. Así pues adicionalmente se hizo algunas pruebas usando una cámara de video B/N infrarroja. Esta señal fue capturada por un Frame Grabber o tarjeta de captura de video hacia la Computadora.
- Se debe considerar que a mayor velocidad de obturación (shutter elevado) se necesitará más luz para compensar.

1.1.3 Consideraciones en cuanto al Posicionamiento

Las fotografías fueron tomadas de la parte frontal o trasera de diferentes vehículos, con una cierta inclinación, respecto a la normal a su plano transversal, que se recomienda ser menor de 5°, esto es por ejemplo: a una distancia de 3.5m del plano transversal al vehículo y a 30cm aprox. de la normal en el punto más cercano como se aprecia en la siguiente figura. Se prefirió restringir ello, ya que con un mayor ángulo se corre el riesgo que se produzca distorsión en perspectiva. Como ya lo veremos más adelante en el capítulo 4 cuando planteamos una estrategia de solución a este problema.

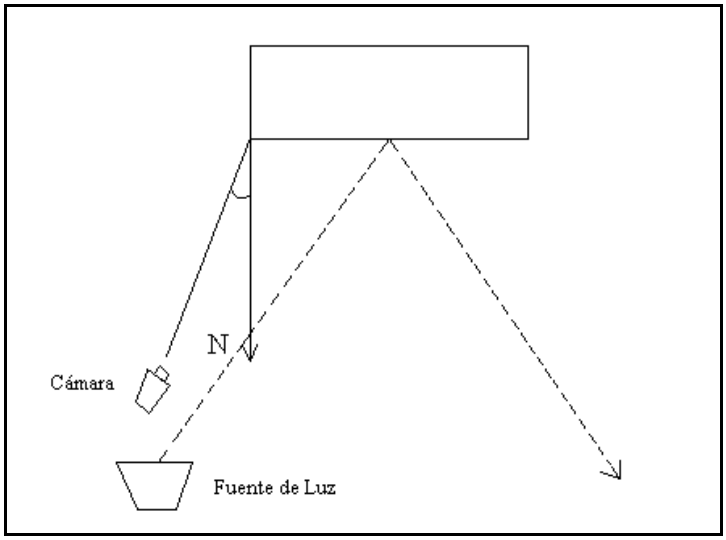


Figura 1.9. Ubicación relativa recomendada para la captura de imágenes

Nuevamente la iluminación se considera en la misma dirección como se analizó en el punto anterior.

La fotografías se tomaron de pie, se tuvieron que hacer muchas tomas para comprender la forma de obtener una buena fotografía esto se fue aprendiendo en el curso

de las tomas fotográficas. La posición ideal en cuanto a la altura que debería estar la cámara sería a la misma altura que la placa teniendo en cuenta la ubicación espacial de la figura 1.9, pero dado que esto sería demasiado bajo, la mayoría de las veces estos sistemas se colocan sobre postes por lo que se prefiere hacer un software más adaptable y robusto (figura 1.10).

Figura 1.10. Sistema comercial de identificación de vehículos utilizado en otros países, la cámara se encuentra alojada en una carcasa metálica como se indica en la figura.

1.1.4 Recolección de Base de datos de imágenes

La base de datos de imágenes consta de un total de 145 fotografías, adjuntas en un CD a este trabajo. Estas fueron tomadas dentro de la Universidad, en diferentes épocas del año durante todo el proceso de desarrollo de los algoritmos para someterlos a muchas pruebas, bajo diferentes condiciones climáticas, dado que se quería desarrollar un sistema robusto ante diferentes condiciones.

En el proceso de las tomas fotográficas se tuvo que separar otro grupo de fotografías que presentaban deficiencia en iluminación, ya que algunas veces la toma salía opaca o con demasiado brillo.

Así también como vimos anteriormente la ubicación de la cámara juega un papel importante, la distancia en primer caso, porque cuando la cámara se encuentra muy lejos pueden aparecer otros objetos alrededor que incrementan la complejidad de la escena, haciendo más difícil la ubicación de la placa, y en segundo caso de la posición vertical relativa, cuando la cámara está muy arriba y cerca del automóvil aparece la distorsión de

perspectiva deformando la placa, lo que complica en etapas posteriores la segmentación de los caracteres como veremos más adelante.

Así pues estas fotografías con ciertas deficiencias se encuentran separadas en otra carpeta: Fotos Restantes, como 170 fotografías adicionales que nos sirven de referencia para tener en cuenta. La lista de las 145 fotografías con los números de placa correspondientes de la Base de datos que sirvió para el desarrollo del sistema se encuentra en el Anexo B.

CAPÍTULO II
PRE PROCESADO DE LA IMAGEN

2.1 Objetivos

Acondicionar la imagen para una mejor aplicación del algoritmo de localización de la placa.

2.2 Conversión de Formato y "Decimación"

La cámara digital Canon PowerShot A10 nos permitió tomar fotografías a color en formato JPEG, con una resolución de 640x480 y transmitirlas a la PC con facilidad. Para comenzar se debe convertir la imagen adquirida de RGB conformada por 3 matrices (Red, Green, Blue), a una imagen en escala de grises (256 niveles), tomando la componente de luminancia del modelo de color (YIQ), y así simplificamos la imagen a una sola matriz conservando la información más relevante, en Matlab **[5]** esto se consigue utilizando el comando: **rgb2gray**.

Una vez que tenemos la imagen en escala de grises, la dividimos a la mitad de su resolución es decir hacemos una "decimación" de relación 2 a 1, esto se obtiene fácilmente eligiendo los píxeles impares en cada fila a través de todas las columnas, hecho en toda la matriz que forma la imagen. Así entonces pasamos de la resolución 640x480 a 320x240, esto servirá para reducir la cantidad de operaciones y por tanto el tiempo de procesamiento.

En esta etapa la "decimación" nos permitirá trabajar con menos píxeles y proceder con la siguiente etapa: la ubicación de la placa. Allí no se va a extraer información de la imagen sino que basándose en criterios del algoritmo ya no es crítico contar con una buena resolución. Luego de determinar aquella zona que encierra la placa regresaremos a la resolución original, pero entonces solo trabajaremos con aquella región reducida de la placa.

Figura 2.1 Imagen del auto a resolución 640 x 480, extraída de Archivo: IMG_0023.jpg

Figura 2.2 Imagen convertida a escala de grises y reducida a la mitad de resolución (320 x 240)
(Se puede apreciar que la imagen tiene menor resolución)

CAPÍTULO III
LOCALIZACIÓN DE LA PLACA DE UN VEHÍCULO

3.1 Introducción

La localización de caracteres de la placa consiste en determinar la ubicación de la misma dentro de la fotografía, es decir extraer aquella zona rectangular que encierre sólo la placa, ello implica poder discernir entre otras zonas que contengan figuras similares como la marca del auto, stickers, y otros símbolos que con cierta frecuencia suelen encontrarse aparte de la placa. Para ello se utilizó una técnica basada en los gradientes de grises, los cuales nos servirán para ir aproximándonos a la zona de la placa, utilizando, detectores de borde, histogramas y finalmente operadores morfológicos.

Dado que existe aún la posibilidad de encontrar más de una posible zona como potencial placa, para ello se implementó un código adicional que extrae las zonas, en caso que haya mas de una, en dos matrices para su posterior procesamiento e identificar la placa verdadera esto se expone en la penúltima parte del capitulo. En la parte final del capítulo mostramos el diagrama de flujo del mismo, para mayor comprensión del algoritmo y el código fuente comentado en el Anexo J.

3.2 Obtención de gradientes

Un problema de importancia fundamental en el análisis de imágenes es la detección de bordes. Los contornos caracterizan las fronteras de los objetos, y por tanto son de gran utilidad de cara a la segmentación e identificación de objetos en escenas.

Los puntos de contorno son como zonas de píxeles en las que existe un cambio brusco de nivel de gris, la teoría de detectores de bordes se explica con mayor detalle en el Anexo E.

La idea de este método es modelar los caracteres como una zona de imagen de altas derivadas [6]. La característica más importante de un carácter para este método es un valor alto y positivo de la derivada horizontal seguido de cerca por otro igualmente alto y negativo (o al revés: el negativo primero). La derivada se aproxima por las fórmulas de Sobel (figura 3.2) dado que mostró mejores valores de gradientes de gris que en la práctica

realzan más los bordes. Este método es capaz de localizar por igual caracteres negros sobre fondo blanco o blancos sobre fondo negro.

Como se sabe, los operadores gradientes son una aproximación discreta de las derivadas parciales de primer orden:

$$\frac{\partial G(x,y)}{\partial x} \approx \frac{G(x,y) - G(x-\Delta x, y)}{\Delta x} \qquad \text{Ec. 3.1}$$

$$\frac{\partial G(x,y)}{\partial x} \approx \frac{G(x-\Delta x, y) - G(x,y)}{\Delta x} \qquad \text{Ec. 3.2}$$

$$\frac{\partial G(x,y)}{\partial x} \approx \frac{G(x+\Delta x, y) - G(x-\Delta x, y)}{2\Delta x} \qquad \text{Ec. 3.3}$$

Con sus correspondientes operadores derivativos discretos:

$$^{-}\mathbf{D}_x = [1. \quad -1] \qquad \text{Ec. 3.1.1}$$

$$^{+}\mathbf{D}_x = [1 \quad -1.] \qquad \text{Ec. 3.2.1}$$

$$^{c}\mathbf{D}_x = \tfrac{1}{2}[1 \quad 0 \quad -1] \qquad \text{Ec. 3.3.1}$$

Donde el subíndice "**.**" denota al píxel central de la máscara.

Al aplicar una máscara de Sobel a una imagen de 3x3 (Figura 3.1), implica calcular la suma de los productos de los coeficientes por los niveles de grises correspondientes contenidos en la región encerrada por la máscara de Sobel. Así los gradientes direccionales E (Este), Oeste (O), Norte (N), Sur (S), aplicados al píxel (i,j) serán:

$$G\,x\,e = G\,E = (A2 + 2A3 + A4) - (A0 + 2A7 + A6) \qquad \text{Ec. 3.4}$$

$$G\,x\,o = G\,O = (A0 + 2A7 + A6) - (A2 + 2A3 + A4) \qquad \text{Ec. 3.5}$$

$$G\,y\,n = G\,N = (A4 + 2A5 + A6) - (A0 + 2A1 + A2) \qquad \text{Ec. 3.6}$$

$$G\,y\,s = G\,S = (A0 + 2A1 + A2) - (A4 + 2A5 + A6) \qquad \text{Ec. 3.7}$$

Donde los A_k son los píxeles que rodean al de interés. La notación queda aclarada con la siguiente figura:

A_0	A_1	A_2
A_7	(i,j)	A_3
A_6	A_5	A_4

Figura 3.1 Píxeles del entorno de un punto y su notación.

Esta matriz de 9 píxeles se aplica a la imagen a través de un filtro digital implementado en Matlab con el comando filter2, las matrices Este y Oeste tienen la forma:

$$h_E = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \qquad h_O = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Figura 3.2 Matrices de la máscara de Sobel

Considerando una imagen almacenada en la matriz **I_ef**, aplicamos el filtro digital usando Matlab de la sgte. manera:

```
hxE  = [ 1 0 -1 ; 2  0 -2  ; 1 0 -1 ];          % Definimos la máscara de Sobel
I2_E = filter2(hxE, I_ef);                      % Aplicamos la máscara a la imagen "I_ef"
hxO=[-1 0 1;-2 0 2 ;-1 0 1];
I2_O=filter2(hxO, I_ef);
I_sum=abs(I2_E)+ abs(I2_O);          % Superposición de los bordes
                                     % horizontales Este-Oeste
figure,imshow(I_sum,[])
```

Se utilizó sólo los gradientes en las direcciones horizontales por dos motivos: el primero es evitar mayor cantidad de operaciones innecesarias y el segundo es que es suficiente obtener los bordes verticales dado que los caracteres tienen mayor cantidad de transiciones en esta dirección. (Figuras 3.3 y 3.4).

A continuación se ilustra gráficamente el resultado del proceso descrito.

Figura 3.3 Resultado de aplicar el operador gradiente en las direcciones verticales (Norte y Sur). Se observa que aparecen más regiones aparte de la placa y ésta no se distingue bien.

Figura 3.4 Resultado de aplicar el gradiente en las dos direcciones horizontales (Este y Oeste). Aquí se aprecia mejor la placa y las otras regiones son menos diferenciadas.

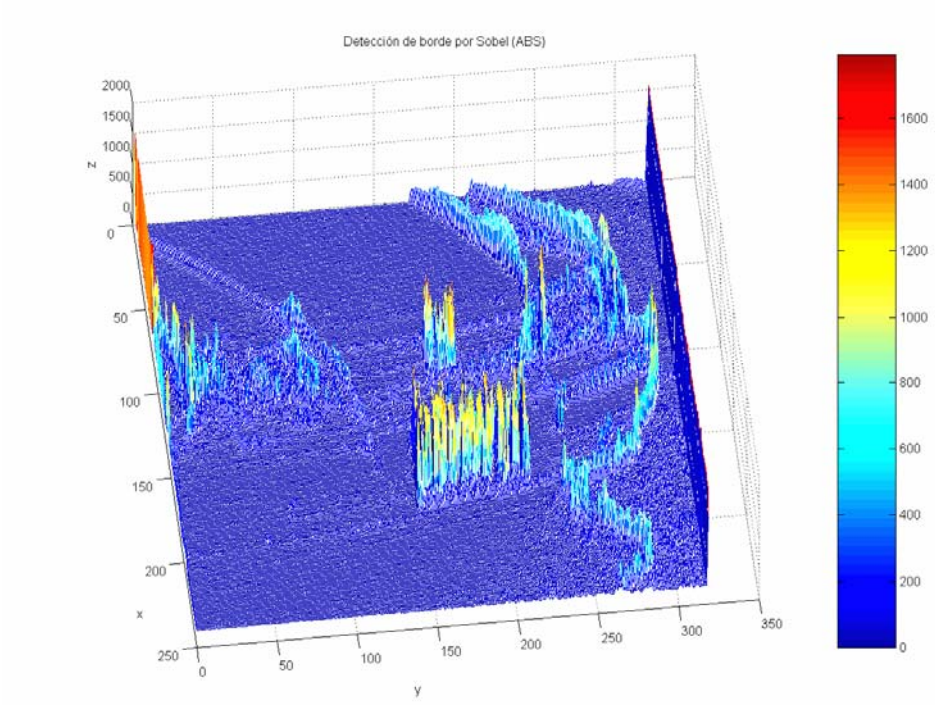


Figura 3.5 Distribución de gradientes en 3D a través de toda la imagen, así también se puede apreciar zonas de mayor densidad de gradientes que otras.

3.3 Umbral de Gradientes

Luego de tener una distribución de gradientes como en la figura 3.5 definimos un Umbral tal que sólo los gradientes a partir de ese valor queden en la matriz de gradientes de la imagen. La elección de un umbral fijo, lleva a grandes problemas cuando varía el contraste de las imágenes de entrada. Es preciso pues, elegir un umbral adaptivo para distinguir los valores altos del gradiente. Para ello, usamos el histograma del valor absoluto del gradiente. Su aspecto es el siguiente:

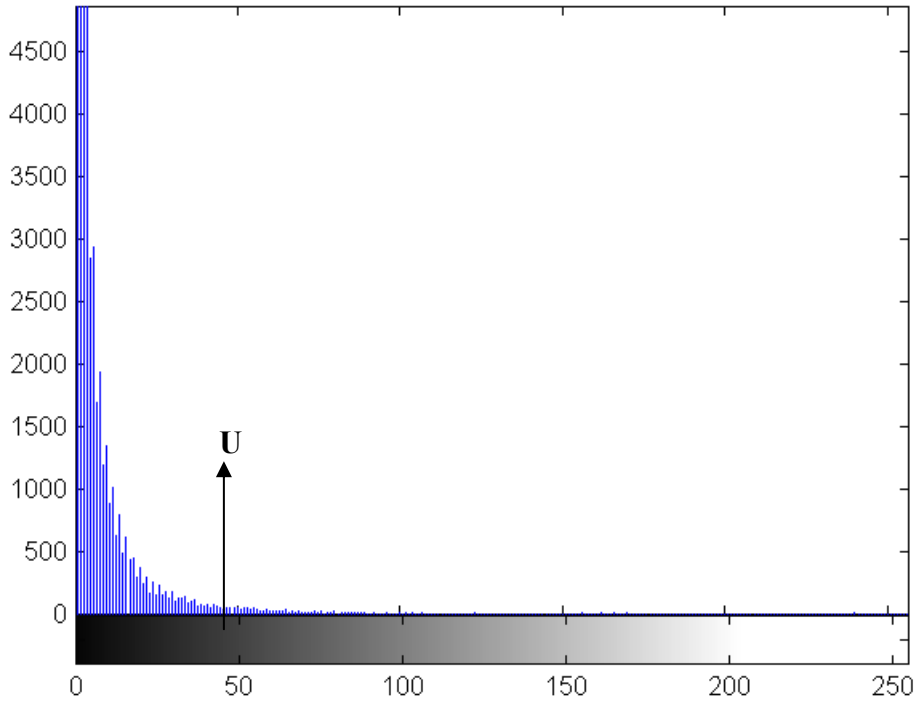


Figura 3.6 Búsqueda del umbral U en el histograma del valor absoluto de gradientes

Para elegir el valor adecuado de umbral recurrimos a un método muy simple: buscar el valor **U** tal que la "cola" del histograma a partir de U contenga un área mayor o igual que un porcentaje dado del total (experimentalmente, determinamos un valor de 3.5% para este umbral). Así, se consigue que el umbral se adapte a la imagen y la localización por gradiente funcione incluso en imágenes de muy bajo contraste. En pocas palabras el criterio consiste en quedarse sólo con un porcentaje de los gradientes más altos del total, este valor experimentalmente lo dejamos en un 3.5%.

Luego de ello podemos poner a "1" los valores altos de gradiente y a "0" aquellos por debajo de este umbral, por lo que tendríamos algo como la figura 3.7. (Estos valores los guardamos en una matriz que llamaremos 'I2_Umb')

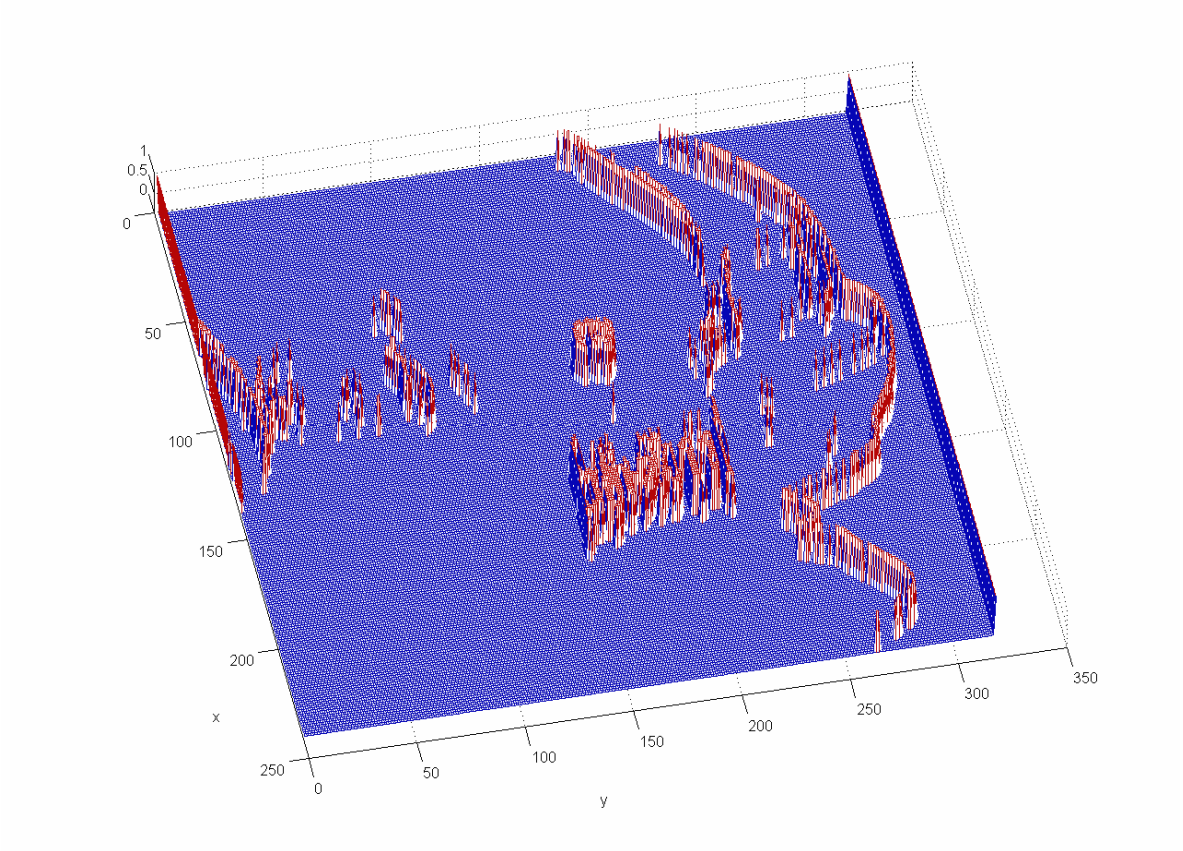


Figura 3.7 Valores de Gradiente mayores que el umbral "U"

Comparando con la figura 3.5 se ha reducido la cantidad de gradientes quedando solo los de valor 'alto', ello se aprecia en la escala de aquella imagen. Pese a la separación de gradientes altos aún se tiene otros gradientes a parte de los de la placa, será necesario aplicar otro criterio adicional y es lo que se hizo a continuación.

3.4 Densidad de Gradientes

Cuando nos encontramos ante la situación de la figura 3.7 tenía que idear la forma de separar los gradientes de la placa de los otros que pertenecían a otras figuras, por lo que siguiendo con los gradientes se pensó en que la zona de la placa contiene mayor "densidad de gradientes" que otras zonas por lo que se decidió aplicar un filtro digital cuyo máscara H (Núcleo de Convolución) sería una matriz cuadrada de coeficientes unitarios, de manera que sólo sume la cantidad de valores que se tenga en cada región que él encierre, es decir cuente los 1's por zonas.

Al principio se pensó en medir la densidad por "unidad de área cuadrada", ello se realizó de la sgte. manera en Matlab:

```
H = ones(19);                    %Defino la matriz de unos de 19x19
I2_Dens = filter2(H, I2_Umb);    %Aplico el filtro a la matriz 'I2_Umb'
```

La aplicación del filtro implica una convolución **[7]** de la matriz de gradientes binaria con el máscara (núcleo) **H**, lo que significa efectuar la suma de los productos de los coeficientes (hij = 1) por los valores binarios (zij) contenidos en la región encerrada por la máscara a través de toda la imagen, por lo que la respuesta de la máscara en un punto (x,y) cualquiera de la imagen sería:

$$R(x,y) = \sum_{i=1}^{M} \sum_{j=1}^{N} h_{ij} z_{ij} \qquad \text{Ec. 3.8}$$

$$H = \begin{pmatrix} 1 & 1... & 1 \\ .. & h_{ij} & .. \\ 1 & 1... & 1 \end{pmatrix} \qquad \text{Ec. 3.9}$$

M x N = 19 x 19

Las dimensiones de **H** se determinaron experimentalmente, al usar esta matriz implícitamente se normaliza la unidad de área al cuadrado de 19x19 elementos. Luego de aplicar el filtro graficamos la distribución de la densidad en 3D para poder apreciarla mejor.

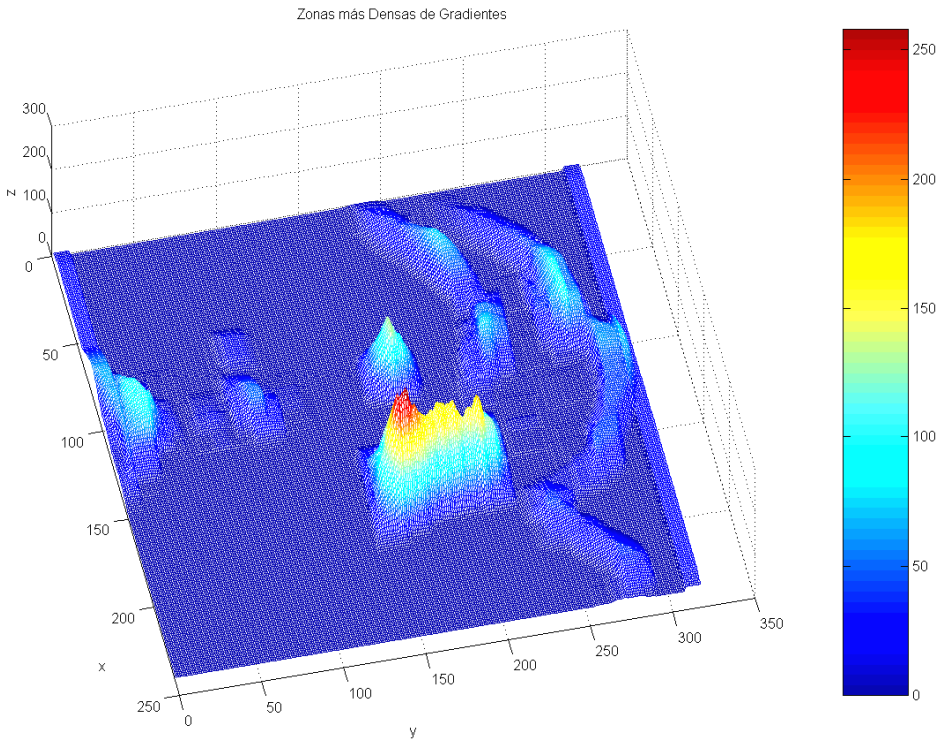


Figura 3.8 Distribución de la Densidad de 'altos' gradientes

De aquí ya podemos diferenciar la zona de la placa de cualquier otra zona, como era de esperarse, en la figura 3.8 se distingue la zona de interés con mayor densidad. En la figura 3.9 hacemos una proyección de esta gráfica sobre el plano XZ para poder observar la diferencia de los picos más elevados.

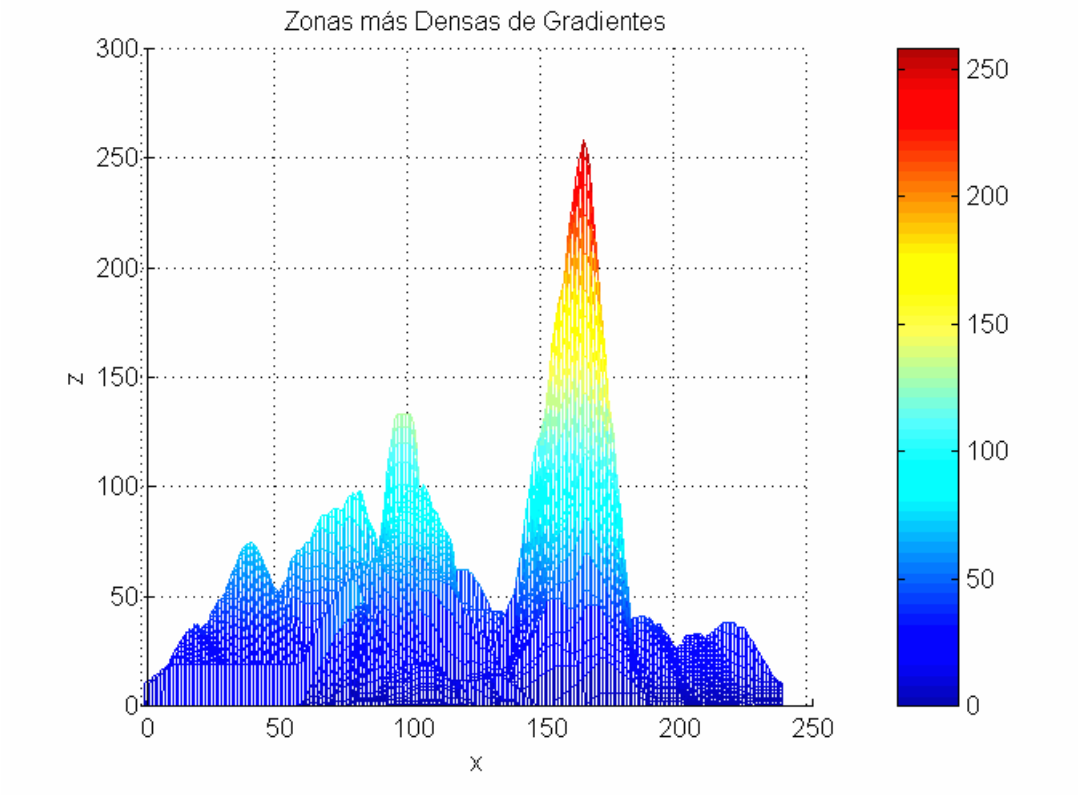


Figura 3.9 Proyección de la gráfica de distribución de densidad en el plano XZ.
El Segundo pico más alto pasa el valor referencial de 100.

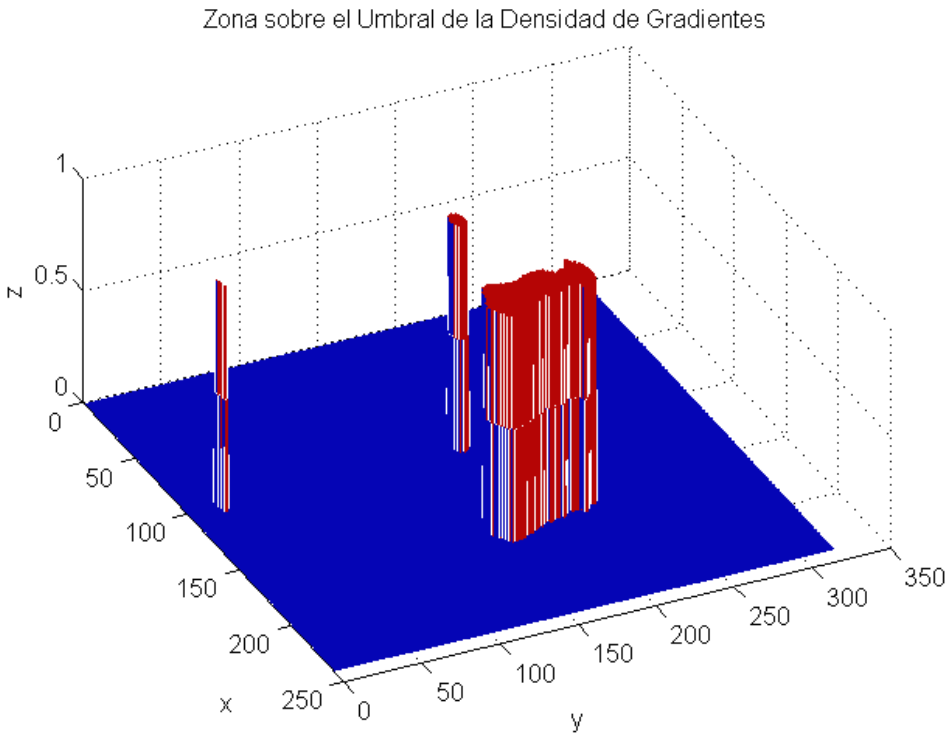


Figura 3.10 Regiones con valores mayores que el Umbral de Densidad

A partir de aquí queda separar la zona de la placa usando un nuevo umbral, en este caso "Umbral de Densidad". Éste se calcula de igual manera como se hizo con el umbral de gradientes, a partir del histograma, en este caso el histograma de la distribución de la densidad. Experimentalmente usamos un umbral de densidad de 2% que en algunas ocasiones aún mantiene otras pequeñas zonas después de aplicar el umbral, como se puede apreciar en la figura 3.10. Resulta que aún persisten pequeñas regiones, por lo que se pensó en usar una densidad que se ajuste a la forma de la placa, de manera de acentuar esa zona.

Es así como llegamos a utilizar una máscara o núcleo **H** ya no de 19x19 (cuadrada) sino de 5x91 (rectangular) para medir la densidad, lo que nos dio mejores resultados tal como se esperaba. Las dimensiones las obtuvimos experimentalmente y efectivamente como podemos ver en la figura 3.11 las regiones que no pertenecen a la placa sobresalen y las otras se reducen.

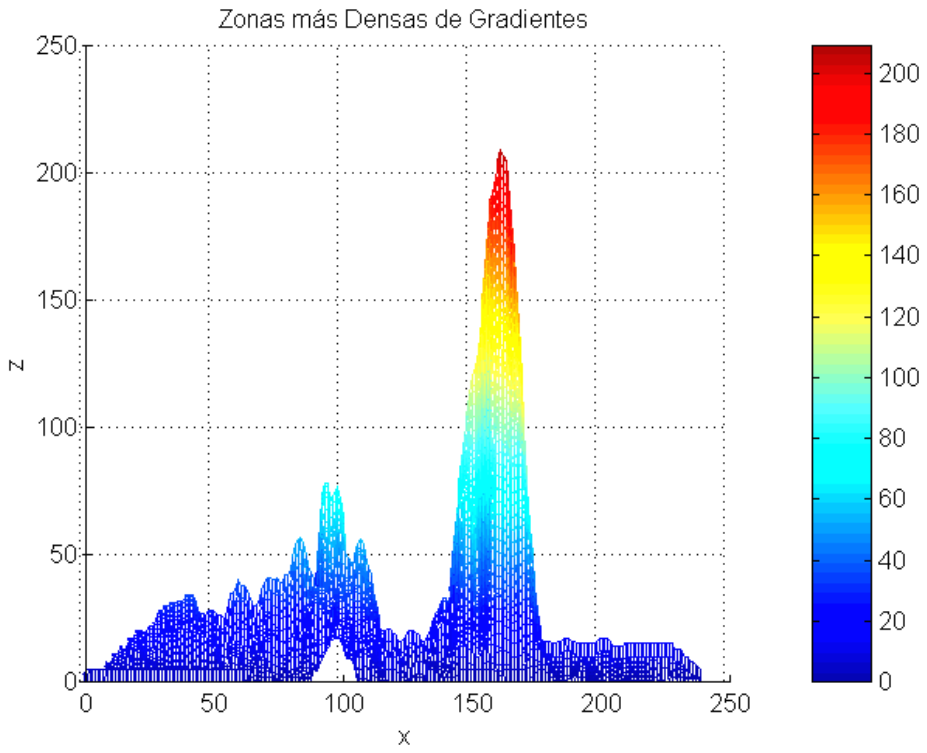


Figura 3.11 Proyección en XZ de la Densidad de Gradientes con H = ones(5,91).
El Segundo pico más alto está por debajo de los 100, ahora mucho más alejado del pico principal.

El código en Matlab es el mismo con la diferencia que la definición de la matriz es ahora:

```
H=ones(5,91);
I2_Dens=filter2(H,I2_Umb);
```

La diferencia relativa entre los picos más altos aumentó y se comprueba porque el menor se encuentra por debajo de la zona verde (Fig. 3.11) no como en la Fig. 3.9.

Esta situación nos permite ahora si, obtener con el Umbral de densidad la zona de la placa (Fig. 3.12). Sólo recortamos la imagen original con un rectángulo que encierra esa zona (Fig. 3.13).

Hasta el momento hemos ido reduciendo las zonas próximas a la placa fundamentalmente manteniendo como "Norte" la presencia de los gradientes en la zona de la placa, separamos con umbral de gradientes, discriminamos más aún con la "densidad de gradientes" y así, otro umbral de densidad, hasta obtener finalmente la zona de la placa. Pero resulta que los valores de Umbral se obtienen por prueba y error, es decir experimentalmente con la base de datos de las fotografías, de manera que en la mayoría de los casos puede funcionar bien un umbral (porcentaje U) pero en otros no, ello tanto para

los gradientes (umbral: 3.5%) como para la densidad de gradientes (umbral: 2%). Esto implica que cuando el umbral no es suficiente para discriminar la zona de la placa, aparecerán más "zonas extrañas" o ajenas a la placa. Debemos entonces aplicar un método para poder reducir por completo las pequeñas "zonas extrañas" que puedan persistir.

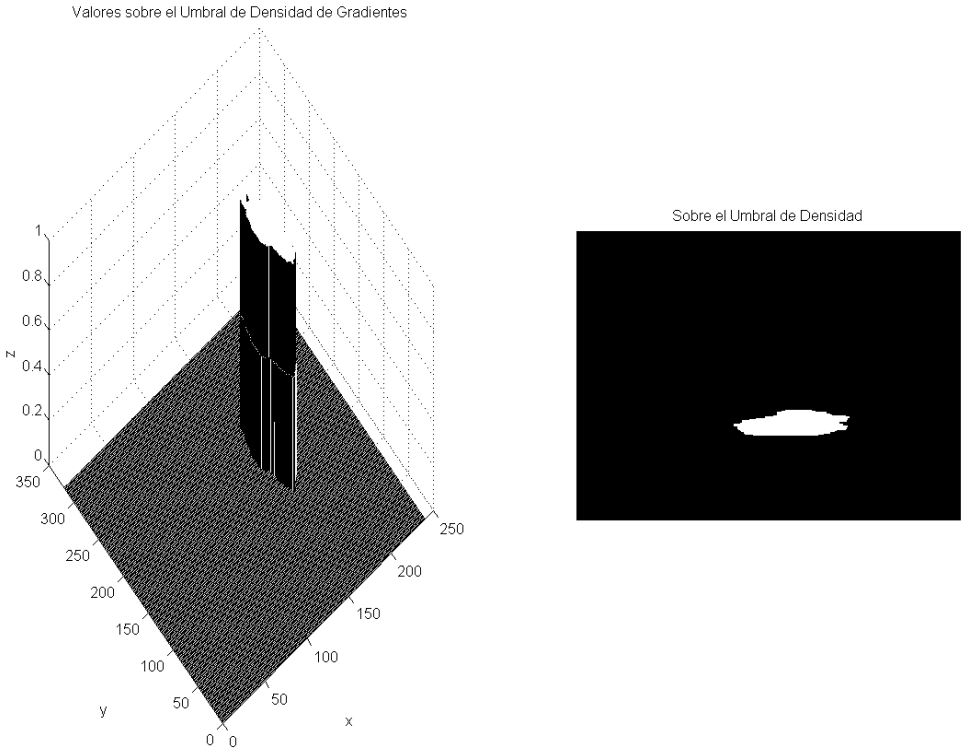


Figura 3.12 Zona sobre el umbral de densidad convertida a binario (1's y 0's)

Figura 3.13 Recorte de la placa ubicada con el método expuesto

3.5 Operadores Morfológicos

En algunas ocasiones, como se mencionó, las imágenes son más complejas que en el ejemplo anterior, principalmente cuando aparecen regiones que puedan tener gran concentración de gradientes, como la vegetación, algunos faros de vehículos, caracteres grandes relacionados a la marca del vehículo, etc, éstas generan en algunos casos regiones grandes comparables con la de la placa (Ver figura 3.14.a.).

Para la reducción total de las "zonas extrañas" utilizaremos entonces las operaciones primarias morfológicas de Erosión y Dilatación [6]. A continuación algunas nociones.

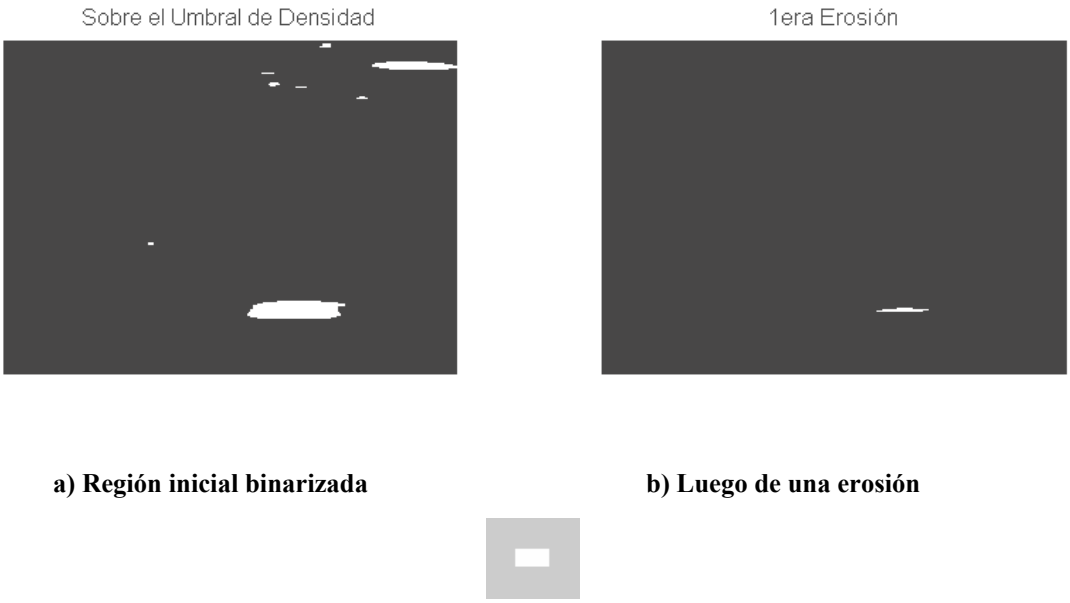
Las operaciones morfológicas a imágenes se definen como procedimientos en los cuales cada nuevo píxel de la imagen resultante es obtenido de una operación no lineal entre un conjunto de puntos de la imagen original F[x,y] y un conjunto de puntos conocido con el nombre de elemento estructurante S[x,y]. Este elemento estructurante recorre toda la imagen para obtener todos los puntos de la nueva imagen.

Dependiendo de los elementos estructurantes y de las operaciones utilizadas, los filtros morfológicos, son capaces de detectar bordes en las imágenes, filtrar objetos de tamaños menores a uno determinado, suavizar fondos de texturizados, detectar fallos en patrones de textura, etc.

El lenguaje de la morfología matemática es la teoría de conjuntos, una base teórica relacionada a la Morfología Matemática se presenta en el Anexo F.

3.5.1 Erosión

La operación morfológica de erosión para imágenes binarias consiste en tomar el elemento estructurante (ver figura 3.14.c), (uno de cuyos puntos se considera el origen, en este caso su centro) y superponerlo a la imagen (Fig. 3.14.a). Si el elemento estructurante está completamente contenido en la imagen, el punto de la imagen se mantiene, de lo contrario pasa a ser del color del fondo. Se aplica para reducir salientes de objetos, eliminar partes menores que el elemento estructurante, eliminar conexiones insignificantes entre objetos mayores, etc. El efecto de la erosión se puede apreciar en la siguiente figura.



a) Región inicial binarizada **b) Luego de una erosión**

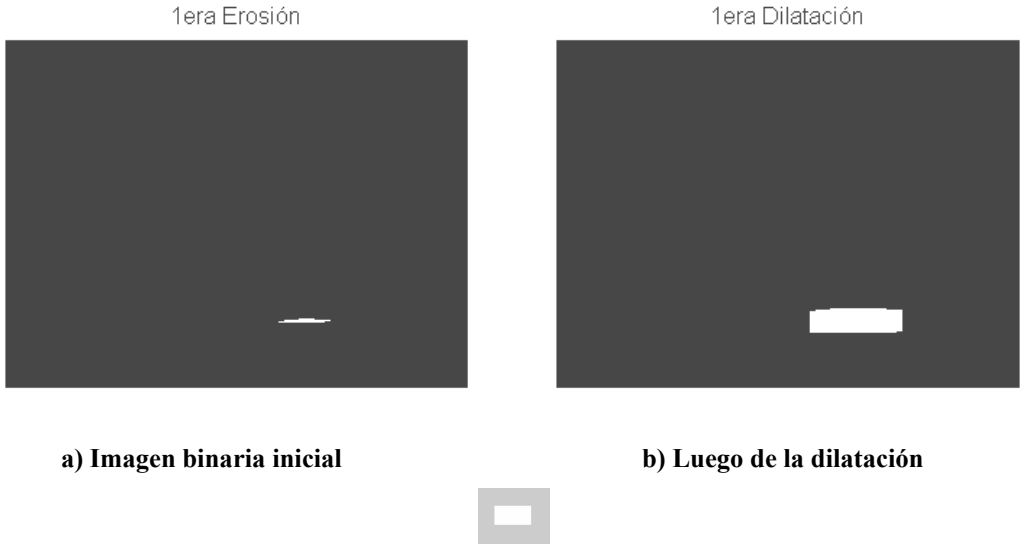
c) Elemento Estructurante

Figura 3.14 Ejemplo de Erosión

El tamaño del elemento estructurante se eligió también experimentalmente y en este caso para la erosión usamos una matriz de 11x21, como se aprecia en la Fig. 3.14.c.

3.5.2 Dilatación

La operación morfológica binaria de dilatación consiste en tomar el elemento estructurante (uno de cuyos puntos se considera el origen, el centro de la matriz en este caso) y superponerlo a la imagen. Si parte del elemento estructurante tiene intersección con el contenido de la imagen, el punto de la imagen pasa a ser igual al punto origen del elemento estructurante, de lo contrario pasa a ser del color del fondo. Se aplica para rellenar entrantes de objetos y unir pequeñas separaciones en las imágenes. El resultado de una dilatación a una imagen binaria se puede observar en la siguiente figura.



a) Imagen binaria inicial **b) Luego de la dilatación**

c) Elemento estructurante

Figura 3.15 Ejemplo de Dilatación

El elemento estructurante fue elegido luego de muchas pruebas, para el caso de la dilatación fue una matriz de 15x29.

Es así como se pudo encerrar la zona de la placa con mayor eficacia, los ejemplos de las figuras 3.14 y 3.15 fueron aplicados secuencialmente en el proceso a la fotografía de la figura. 3.17.

Figura 3.16 Ubicación de la placa de la figura 3.17, utilizando operadores morfológicos para discernir entre otros objetos.

Figura 3.17 Imagen de archivo IMG_0037.jpg

Aquí podemos apreciar la aparición de figuras algo complejas como la vegetación, piedras y otras que producen zonas con altos gradientes que finalmente pudimos discriminar.

3.6 Localización de las zonas que contengan una placa

En el Proceso de localización de la placa utilizando gradientes, existe la posibilidad que por la complejidad de la escena donde se ubique el vehículo, los algoritmos determinen más de una posible zona que podría contener una placa. Para estos casos se implementó un procedimiento que extrae dos zonas como potenciales placas, a partir de las proyecciones verticales y horizontales, que pasarían las siguientes etapas donde más adelante se determina cual es la verdadera.

A continuación expondremos la técnica que se ideó para superar el problema.

3.6.1 Separación de posibles placas en bloques

Un problema que se presenta con cierta frecuencia es la presencia de stickers u otros caracteres en los vehículos, lo que conlleva a que los algoritmos de ubicación de la placa encuentren más de un bloque, por esto fue necesario elaborar de manera adicional un programa que pueda separar estos bloques, pasen a las siguientes etapas, y se elimine la que no tenga información relevante.

El criterio básicamente consiste en determinar a partir de las proyecciones horizontales y verticales cuantos bloques tenemos (posibles placas). El código toma como entrada el resultado de la dilatación como se muestra en la Figura 3.19 (Imagen procedente de archivo: **IMG_0070.jpg**).

Figura 3.18 Imagen original

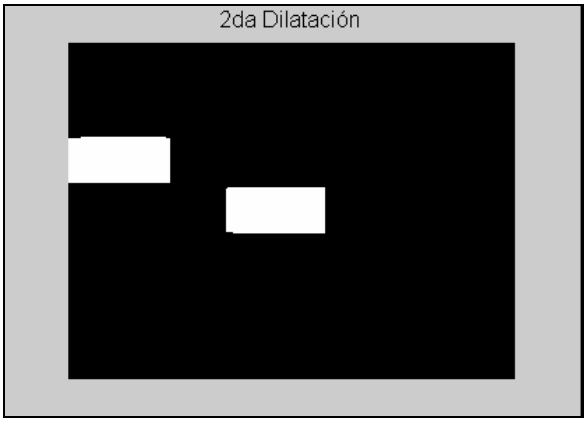


Figura 3.19 Resultado de la dilatación, en la etapa de localización.
El sistema muestra dos regiones con potenciales Placas

Las proyecciones se muestran en las figuras 3.20 y 3.21. El resultado de la separación de los bloques queda como se muestra en la figuras 3.22 y 3.23.

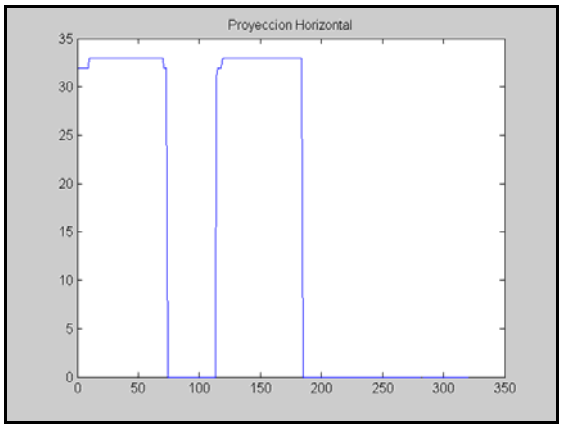


Figura 3.20 Proyección Horizontal

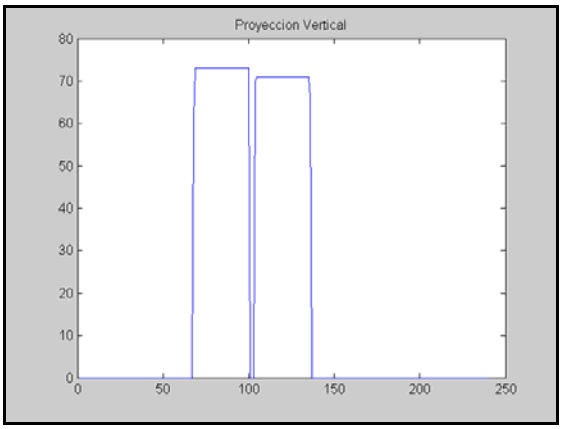


Figura 3.21 Proyección Vertical

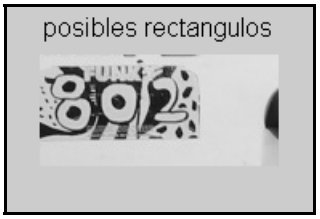


Figura 3.22

Figura 3.23

El código Matlab correspondiente, se muestra en el Anexo J.

3.7 Estructura del algoritmo de localización

El algoritmo de localización básicamente consiste en dos etapas diferenciadas, una primera, la de ubicación en sí de la placa a través del procesamiento de imágenes y la segunda, la extracción de uno o dos rectángulos que puedan ser placas para el análisis posterior, esta última esta representada por la sub-rutina llamada separa_rectángulos.m.

Éstas son las partes que componen esta etapa, ahora lo detallamos en un diagrama de flujo.

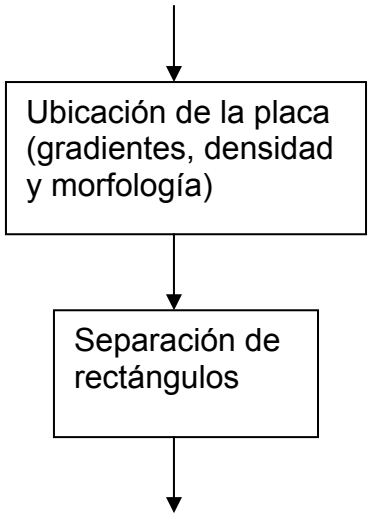


Figura 3.24 Etapas de la localización: ubicación y extracción de rectángulos

Luego, el primer bloque se compone de los siguientes:

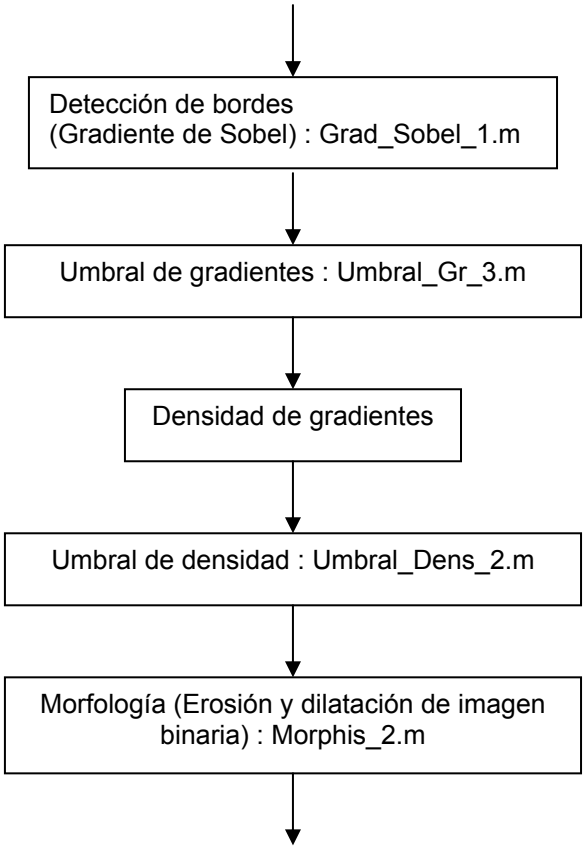


Figura 3.25 Etapas de la ubicación de la placa

El segundo bloque es una función, **separa_rectángulos.m**, que tiene por entrada el resultado de los operadores morfológicos y la salida son una o dos matrices conteniendo la zona de la placa:

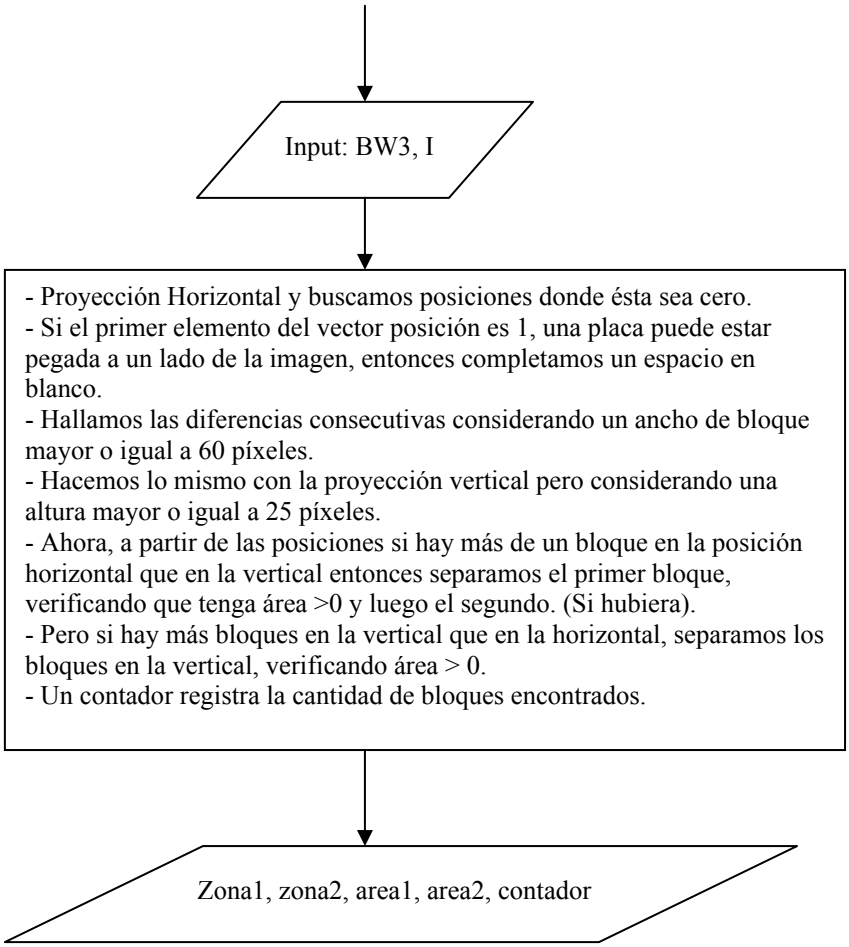


Figura 3.26 Etapas del bloque separa rectángulos

CAPÍTULO IV
CORRECCIÓN DE PERSPECTIVA
(PROYECCIÓN SOBRE PLANO FRONTAL)

4.1 Introducción

La ubicación relativa de la cámara con respecto al objeto puede producir una cierta inclinación de la placa, cuando la cámara se encuentre de frente. Pero el caso más general se presenta cuando la cámara está ubicada a un lado de la acera y con una altura determinada, lo que en algunos casos produce distorsión de perspectiva. Existen trabajos como en [11] que contemplan el problema en fotografías de documentos.

La corrección de inclinación de la placa, para el primer caso, parte por la determinación del ángulo primero, para luego efectuar el giro geométrico de la imagen de la placa a través de las fórmulas ya conocidas de Transformación Isométrica o Euclídea [10], estas técnicas han sido empleadas por otros autores como se aprecia en [4].

Sin embargo la corrección de perspectiva depende de la determinación de las coordenadas de los vértices de la imagen a corregir y consiste en proyectar una imagen deformada por perspectiva sobre un plano de frente [10] en el que definimos las nuevas coordenadas. Esta corrección geométrica se realiza a través de una Transformación Proyectiva que se estudia en el campo de la Geometría Proyectiva 2D.

La corrección de perspectiva matemáticamente se implementa con las 4 coordenadas de los vértices de la imagen a corregir que puede ser en general un trapezoide que encierre la placa, pero en la práctica éstas deben ser halladas previamente a partir de la fotografía y esto debe hacerlo el sistema automáticamente.

Para resolver este inconveniente se hizo una combinación de la transformada de Hough y la Transformación Proyectiva General, la primera para determinar las coordenadas de los vértices y la segunda para la corrección directa de la distorsión de perspectiva.

En este capítulo veremos la distorsión de perspectiva y Transformación Proyectiva como marco principal, y la Transformada de Hough, como herramienta complementaria en este caso, para la determinación de rectas y ángulos, así observaremos también como lo utilizamos convenientemente.

4.2 La Transformación Proyectiva General

Como mencionamos anteriormente la ubicación de la cámara a un lado de la pista, sobre una vereda, puede producir un efecto en perspectiva de la imagen, tal como lo muestra la figura 4.1. (Imagen de archivo: Foto_0010.jpg, **Fotos Restantes**), aquí la corrección de inclinación simple por rotación no resuelve el problema.

a) Imagen original **b)** La rotación simple no corrige el problema

Figura 4.1 Toma de imagen con proyección en perspectiva

En la figura **4.1.a** se tiene la placa con un efecto en perspectiva, nótese que el margen inferior es la línea de referencia que cualquier transformada de líneas rectas tomaría en cuenta (Transformada de Hough o Radón), por lo que en 4.1.b (luego de la "corrección" de la inclinación) esta línea de "referencia" se encuentra horizontal pero los caracteres quedan inclinados casi como un texto en letra ***cursiva***. Lo que complicaría la segmentación por técnicas de proyecciones. Este efecto se resuelve con la aplicación de la "Teoría de Geometría Proyectiva" a través de la Transformación Proyectiva General con ésta se puede obtener la proyección de la placa en un plano rectangular "corregido", el problema se reduce a hallar la matriz de transformación "**H**" a partir de 4 puntos correspondientes tanto de la imagen original como de la nueva (plano rectangular, ver figura 4.3.) tal que transforme la región X'Y' a otra en XY.

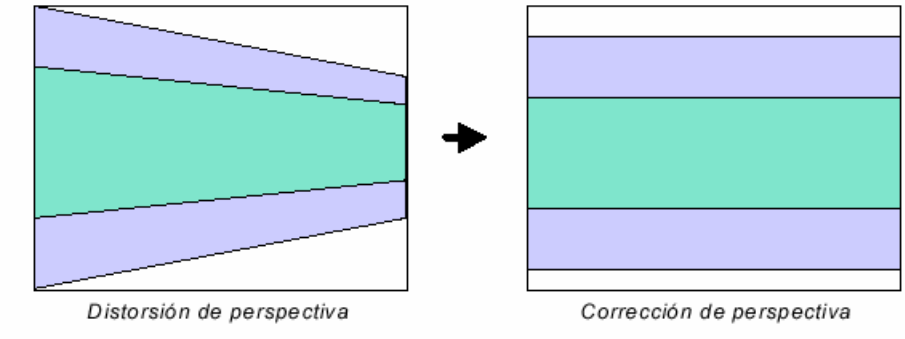


Figura 4.2 Plano en perspectiva y su correspondiente corrección

La Geometría Proyectiva es una herramienta muy utilizada en aplicaciones de visión artificial, ésta nos permite mediante las transformaciones proyectivas de un plano, modelar

la distorsión geométrica que le ocurre cuando es visto por una cámara perspectiva. Con estas transformaciones algunas propiedades se conservan como la colinearidad (líneas rectas son vistas como líneas rectas) mientras que otras propiedades no (las líneas paralelas por lo general no son vistas como líneas paralelas).

En un nivel elemental la geometría es el estudio de puntos y líneas, y sus relaciones. A lo largo de la historia la geometría ha sido concebida inicialmente como una disciplina netamente geométrica, en la que las líneas y puntos se estudian sin considerar un sistema de coordenadas. Posteriormente, mediante la introducción de un sistema de coordenadas cartesiano se logra algebrizar a la geometría. De esta manera, las entidades geométricas pueden ser descritas como coordenadas y entidades algebraicas. Por medio de las relaciones algebraicas se obtiene una representación matemática apropiada para implementar algoritmos y programar métodos computacionales de corrección de perspectiva en imágenes. Un compendio teórico se puede encontrar en la referencia [4].

La Transformación Proyectiva consiste básicamente en aplicar una Transformación a una imagen a través de una matriz de transformación H. Para la determinación de esta matriz se necesitan 4 puntos (coordenadas) de la imagen original y los correspondientes 4 puntos de la nueva imagen proyectada. Una vez determinada la matriz de transformación se usa un algoritmo para generar la nueva imagen aplicando la matriz H y una interpolación bilineal para determinar el valor de gris que se le asignará al punto de la nueva imagen. A continuación ilustramos este procedimiento en un ejemplo.

➢ ***Ejemplo 4.2.1***

Aquí tenemos la distorsión de perspectiva ilustrada en la figura 4.3 con ejes **x' y'**, el objetivo es encontrar la transformación proyectiva definida por la matriz **H** que realiza la corrección de la distorsión mostrada en la figura con ejes **x y**.

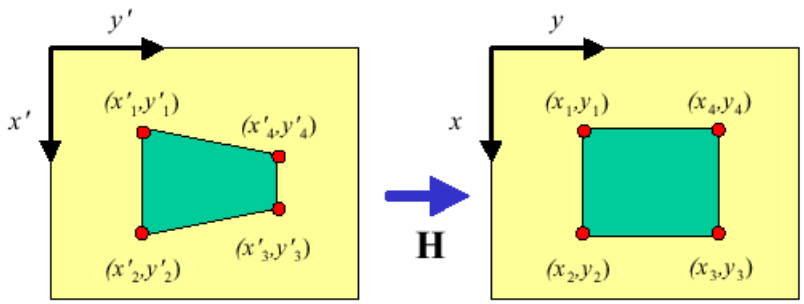


Figura 4.3 Para corregir un plano en perspectiva (izquierda), es necesario hallar la matriz de transformación lineal **H** que corrija la imagen en un plano frontal. Esta matriz se hallla a partir de los 4 vértices conrrespondientes en cada plano ***x'y'*** y ***xy***.

Se puede establecer "**H**" según la teoría de Geometría Proyectiva:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \qquad \text{Ec. 4.1}$$

De la referencia **[4]** Págs.13 -15, tenemos las siguientes relaciones para cada punto del plano **xy** y su correspondiente en el plano **x'y'**:

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \qquad \text{Ec. 4.2}$$

$$y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \qquad \text{Ec. 4.3}$$

Podemos ver que si dividimos por un factor a todos los elementos h_{ij}, x' ó y' no cambian su valor por lo que podemos hacer $h_{33} = 1$, con lo que solo quedan por hallar 8 valores h_{ij}.

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \qquad \text{Ec. 4.4}$$

Por cada par de puntos se tiene dos ecuaciones, como son 8 incógnitas, con 4 puntos es suficiente para definir la matriz **H**. Por ejemplo en la figura 4.3 si se tuviera:

$(x'_1; y'_1) = (1; 1)$; $(x'_2; y'_2) = (4; 1)$; $(x'_3; y'_3) = (3; 4)$; $(x'_4; y'_4) = (2; 4)$;

$(x_1; y_1) = (1; 1)$; $(x_2; y_2) = (4; 1)$; $(x3; y3) = (4; 4)$; $(x_4; y_4) = (1; 4)$.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 & 1 & 1 & -1 & -1 \\ 4 & 1 & 1 & 0 & 0 & 0 & -16 & -4 \\ 0 & 0 & 0 & 4 & 1 & 1 & -4 & -1 \\ 4 & 4 & 1 & 0 & 0 & 0 & -12 & -12 \\ 0 & 0 & 0 & 4 & 4 & 1 & -16 & -16 \\ 1 & 4 & 1 & 0 & 0 & 0 & -2 & -8 \\ 0 & 0 & 0 & 1 & 4 & 1 & -4 & -16 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 4 \\ 1 \\ 3 \\ 4 \\ 2 \\ 4 \end{bmatrix}$$

Resolviendo el sistema se obtiene:

$$\mathbf{H} = \begin{bmatrix} 3 & 5 & -5 \\ 0 & 11 & -8 \\ 0 & 2 & 1 \end{bmatrix}$$

Es importante recalcar que para determinar la matriz de transformación **H**, aquella que va a hacer la corrección de perspectiva, son suficientes 4 puntos (de la imagen original en **x'y'**) que en este caso pueden ser precisamente aquellos vértices que encierran la placa, una región trapezoidal en general. Los otros 4 de la imagen corregida, uno mismo las delimita para así definir las dimensiones de la nueva región en **xy** como se ve en la figura 4.3.

- ***Ejemplo 4.2.2***

En este ejemplo se tiene un caso real de la distorsión de perspectiva:

Figura 4.4 Imagen original

Ubicamos las coordenadas de las esquinas de la placa manualmente para este ejemplo y se le asignó a una nueva región rectangular de 50x90 (Fig. 4.5), con los 4 puntos correspondientes de uno y otro plano se determinó la matriz **H** y luego se utilizó un algoritmo para determinar la correspondencia de un píxel de la imagen corregida a la imagen deformada haciendo una interpolación bilineal para el valor de gris

correspondiente de los píxeles cercanos. Así entonces se consigue la imagen proyectada de la placa sobre un nuevo rectángulo dado.

Figura 4.5 Imagen corregida luego de la transformación con la matriz **H** obtenida a partir de los vértices de la imagen original y la nueva (rectángulo definido).

En general para generar la nueva imagen se aplica la transformada proyectiva mediante la matriz **H** entre una imagen original **I'** y una rectificada **I**, esto es que para cada píxel (x, y) en la imagen rectificada se busque su píxel correspondiente (x', y') en la imagen original utilizando la transformación **H, [10]** Pág. 27. El valor de gris en (x', y') tendría que ser interpolado de sus cuatro vecinos enteros, ya que al ser **I'** una imagen digitalizada **I'** (x', y') está definido sólo para valores enteros de (x', y'). De esta manera los píxeles van tomando el nivel de gris más adecuado y así se obtiene la imagen rectificada.

Este proceso se traduce en un programa donde los datos de entrada son **H** y la imagen original **I'** cuyos elementos **I'** (x', y') son los valores de gris en los píxeles denotados por (x', y') para $x' = 1, \ldots, N'$ y $y' = 1, \ldots, M'$. Se desea encontrar entonces la imagen rectificada **I** donde **I**(x, y) son los valores de gris en los píxeles (x, y) para $x = 1,\ldots,N$ y $y = 1,\ldots,M$. A continuación se presenta el algoritmo básico:

Algoritmo:

1. Para $x = 1,\ldots,N$ y para $y = 1,\ldots,M$:
2. Definir $m = [x\ y\ 1]^T$.
3. Evaluar **m'** = **Hm**.
4. Evaluar $x' = m_1 / m_3$ y $y' = m_2 / m_3$.
5. Definir $\tilde{x} = fix(x')$, $\tilde{y} = fix(y')$, $\Delta x' = x' - \tilde{x}$ y $\Delta y' = y' - \tilde{y}$.
6. Evaluar $I(x,y) = [I'(\tilde{x}+1,\tilde{y}) - I'(\tilde{x},\tilde{y})]\Delta x' + [I'(\tilde{x},\tilde{y}+1) - I'(\tilde{x},\tilde{y})]\Delta y' +$

$$[I'(\tilde{x}+1,\tilde{y}+1) + I'(\tilde{x},\tilde{y}) - I'(\tilde{x}+1,\tilde{y}) - I'(\tilde{x},\tilde{y}+1)]\Delta x'\Delta y' + I'(\tilde{x},\tilde{y})$$

En este algoritmo se ha utilizado la función ‘fix’ que extrae la parte entera de un número real. La interpolación bilineal tiene lugar en el punto 6 del algoritmo, con ésta se

obtiene el valor de gris que correspondería al nuevo píxel.

4.3 Descripción de la Transformada de Hough

Una línea recta a una distancia ρ y orientación θ puede ser representada como:

$$\rho = x\cos\theta + y\,sen\theta \qquad \text{Ec. 4.5}$$

La transformada de Hough de esta línea es sólo un punto en el plano (ρ, θ), ver Fig. 4.6. Este hecho puede ser usado para detectar líneas rectas en un conjunto de puntos que pueden formar fronteras (segmentos de rectas).

Así también la ecuación Ec. 4.5 convierte un punto del espacio cartesiano (*x,y*) en una curva en el espacio polar (ρ, θ), (Fig. 4.7) ecuación 4.6.

$$\rho = \sqrt{x^2 + y^2}\,sen(\theta + \varphi); \quad \text{donde} \quad \varphi = tg^{-1}(\frac{x}{y}) \qquad \text{Ec. 4.6}$$

Esa curva representa a todas las rectas que pasan por ese punto. Cada punto (ρ_o, θ_o) de la curva transformada corresponde con una recta que pasa por el punto, donde θ_o es el ángulo que la recta forma con el eje de las "**x**" y ρ_o la distancia de la recta al origen de coordenadas, Fig.4.6.

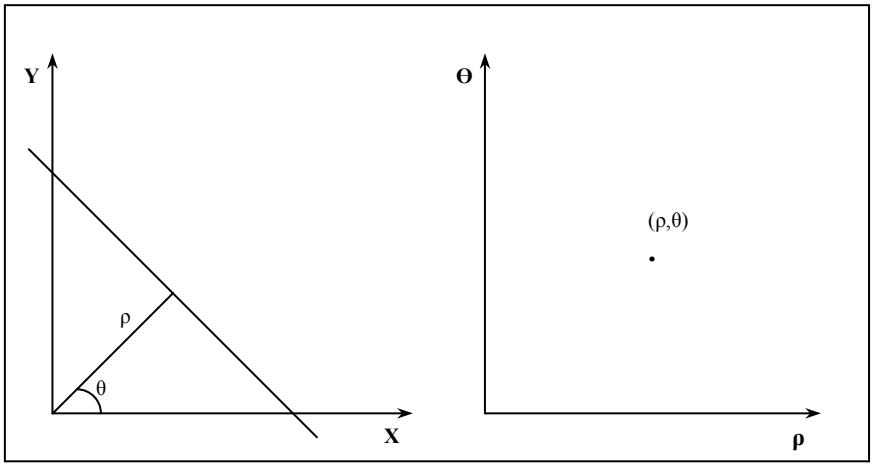


Figura 4.6 La transformada de Hough de una recta en el plano **xy** es un punto en el plano **ρθ** (Ec. 4.5)

Una consecuencia de esto es que cuando dos curvas transformadas de dos puntos se cortan en un punto (ρ_1, θ_1), esas coordenadas nos permiten dibujar la recta que los une (la recta que pasa por esos puntos). Al haber varios puntos con sus correspondientes curvas transformadas y éstas se interceptan en un solo punto común (ρ, θ) significa que todos estos puntos forman una misma recta representada con ángulo θ a una distancia ρ (Fig. 4.7).

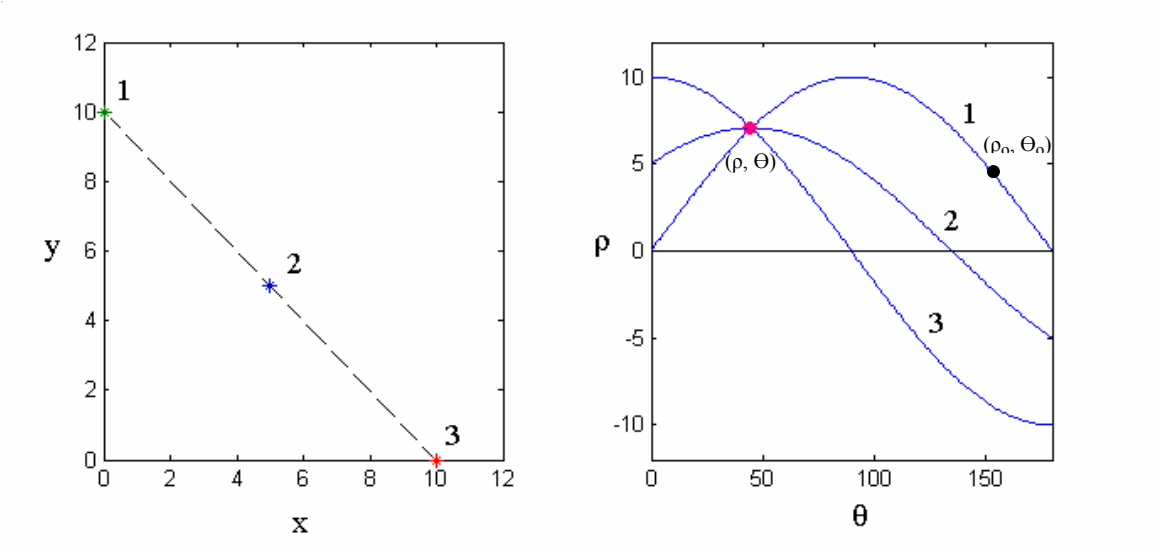


Figura 4.7. Los puntos 1, 2 y 3 del plano X-Y tienen sus correspondientes curvas en el plano ρ - θ. La intersección de aquellas en un punto (ρ , θ) coincide con la recta común que los une en el plano X-Y.

Dado que las imágenes se almacenan en matrices, el orden comienza desde la esquina superior izquierda para el conteo de las filas y columnas, por tanto se elige convenientemente el origen y los ejes coordenados en la misma posición como se aprecia en la figura 4.8. De esta manera el ángulo θ que arroja la transformada de Hough coincide precisamente con la inclinación de la recta con respecto a la "horizontal".

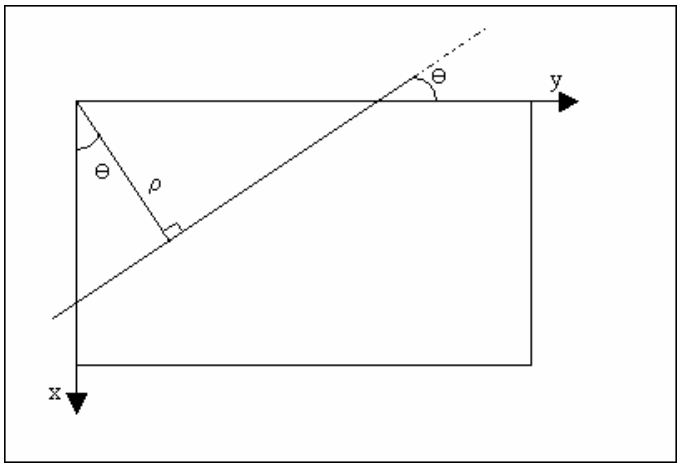


Figura 4.8. Ubicación adecuada de los ejes XY sobre la imagen para obtener directamente el ángulo de inclinación utilizando de la transformada de Hough

Para implementar la transformada discretizamos los dos ejes polares (ρ, θ) y construimos una matriz N(ρ, θ). Donde esa matriz de números enteros representan el número de curvas que pasan por ese punto.

Los máximos en la matriz representan las inclinaciones dominantes. Para el rango de los ángulos restringimos de -30^o a 30^o para determinar rectas horizontales, así reducimos la cantidad de operaciones que como sabemos toda transformada implica, el incremento es de 1^o en 1^o que será la precisión. Más adelante veremos como la utilizamos para determinar rectas verticales.

Una aplicación de la transformada es para determinar el ángulo de inclinación de la placa, esto se hace sobre un recorte de la misma proveniente de una fase anterior. Utilizaremos sólo un conjunto reducido de puntos, ya que si lo aplicáramos sobre toda la imagen (Fig. 4.9) demoraría mucho dado que estas operaciones demandan un uso intensivo del procesador. Así entonces utilizamos solamente los puntos de los bordes horizontales, luego de aplicar la máscara de Sobel. Se obtiene entonces una imagen binaria donde los puntos blancos tienen valor "1" y los negros "0" como en la Figura 4.10.

Figura 4.9. Imagen de archivo: Foto_0012.jpg, **Fotos Restantes**

Figura 4.10. Usamos sólo los puntos de los bordes para la transformada de Hough
En este ejemplo los bordes horizontales

En Matlab, la detección de los bordes sobre la imagen de la figura 4.9 lo hallamos con el siguiente comando:

```
edge(zona2,'sobel','horizontal')
```

Donde zona2 es la matriz que contiene la 'imagen original a escala de grises', así obtenemos la figura 4.10. A continuación la implementación de la transformada de Hough en Matlab a continuación.

```
%%%%%%%%%   TRANSFORMADA DE HOUGH   %%%%%%%%%%%%%%
A= edge(zona2,'sobel','horizontal');      % Aplicación de bordes sobre 'zona2'
figure,imshow(A)
[m,n]=size(A);
theta=-30:30;                  % Restringimos los ángulos convenientemente
[L1,L2]=size(theta);           % L2, tamaño de theta
p=(m^2+n^2)^0.5/100;
rho1=0:p:(m^2+n^2)^0.5;              % Discretización de "ρ"
[L3,L4]=size(rho1);              % L4, tamaño de rho
N=zeros(L4,L2);                  % Definimos la matriz que contará las intersecciones
for i=1:m,
   for j=1:n,
      if (A(i,j)==1)
         for k=1:L2,
           rho2=i*cos(theta(k)*pi/180)+j*sin(theta(k)*pi/180);    %usamos la ecuación
                                % con filas i, las "x" y columnas j, las "y", Fig. 4.8
          Q=1;
           while Q<L4
               if (rho1(Q)<rho2 & rho2<=rho1(Q+1)) % rho2 es mayor que el rho1(Q)
                                                % pero menor que el sgte.
                                                      % rho1(Q+1). Esta en el intervalo.
                  N(Q,k)=N(Q,k)+1;                    % Incrementa contador
               end
               Q=Q+1;
            end
         end
      end
   end
end;
```

N es la matriz cuyas filas representan los valores discretos de ρ_i y las columnas los de θ_i, el valor de cada elemento de esta matriz representa el número de puntos que pasan por una recta para un ρ_i y θ_i específicos.

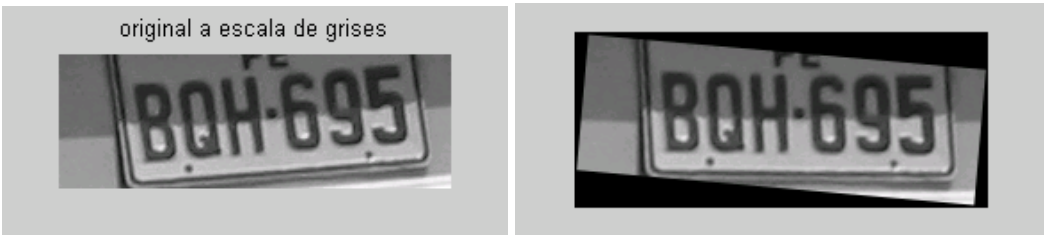
Luego de efectuada esta parte, podemos ubicar el máximo de la matriz, es decir la mayor cantidad de puntos que coinciden con la misma recta de parámetros (ρ,θ) que será la recta con ángulo de inclinación θ del borde inferior de la placa, Fig. 4.10. Para ello utilizamos el siguiente algoritmo.

```
%%%%% DETERMINANDO ÁNGULO DE INCLINACIÓN
recta = N(1,1);
for i=1:L4,
   for j=1:L2,
      if (N(i,j)>recta)
        recta=N(i,j);
        theta_recta=theta(j);
      end
  end
end;
```

El resultado se obtiene en la variable **theta_recta** en grados sexagesimales:

```
theta_recta =
 5
```

Esto quiere decir que la placa tenía 5 grados de inclinación, haciendo la rotación de la placa se obtiene la figura 4.11.b:



a) Placa original inclinada 5^o b) Placa corregida por rotación simple

Figura 4.11 Determinación de ángulo de inclinación y rotación geométrica

Lo anterior es una aplicación típica de la transformada de Hough, ahora lo usaré para determinar cuatro vértices que encierren los caracteres de la placa o la placa en si, estos

podrían formar no necesariamente un rectángulo, en muchos casos por la distorsión de perspectiva conforman un trapezoide. Pero como vimos en la sección anterior (4.2), es necesario contar con esos vértices para la corrección de perspectiva a través de la transformación proyectiva.

La estrategia elaborada consistió en:

1.- Determinar cuatro rectas una para cada lado de la placa, 2 "horizontales" y 2 "verticales".

2.- Una vez halladas las rectas que encierran la placa, estas generan una "malla" a partir de ahí hallamos las intersecciones de éstas, que constituirán precisamente los vértices deseados para la transformación proyectiva.

En la figura 4.12. de la Base de datos, se observa la placa a la que se aplicará el procedimiento descrito, como resultado de la aplicación del punto 1 se obtiene figura 4.13.

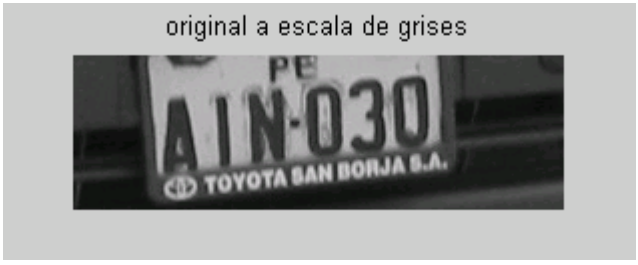


Figura 4.12. Imagen de archivo: Foto_0004.jpg **(Fotos Restantes)**

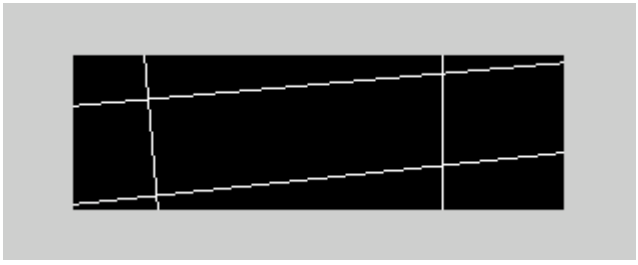


Figura 4.13. Aplicación de la Transformada de Hough a partir de los bordes sobre los cuatro lados de imagen de la placa.

Con esto se consigue determinar una región que encierra los lados de la placa a través de rectas que se cruzan. A partir de aquí el siguiente paso será hallar las intersecciones de estas rectas es decir los vértices. La implementación del algoritmo del punto 1 está en la función "corrige_perspectiva2" y del punto 2 se encuentra dentro de la función "corrección", ambos en el anexo K.

Con la aplicación de esta parte, se obtiene para las abscisas:

a =10 22 55 70

y para las ordenadas:

b =185 38 185 42

Tomando los pares ordenados en sentido horario a partir del vértice superior izquierdo tenemos:

(22,38) (10,185) (55,185) (70,42)

Es importante establecer un orden relativo entre los vértices, para efectuar el cálculo de la matriz de transformación H (Fig. 4.3, Ejemplo 4.2.1), de manera de hacer corresponder correctamente los puntos de la imagen original distorsionada y la nueva, para ello definimos por conveniencia que el orden sea a partir del vértice superior izquierdo y en sentido horario. Así entonces los vectores "**a**" y "**b**" pasan por un algoritmo para ordenarlos con esta convención.

Con esto los vértices se almacenan en esta última matriz, en el orden requerido:

vertices =

22 38

10 185

55 185

70 42

Así, ya estamos listos para la corrección de perspectiva, como veremos en el siguiente apartado.

4.4 Corrección de la Distorsión de Perspectiva

A partir de los vértices hallados se calcula la matriz de transformación H, con la correspondencia punto a punto entre las imágenes original **I'** y corregida **I**. Para generar las dimensiones de la nueva imagen corregida o proyectada, escogemos a partir del ancho y el largo de la placa original tomando como referencia sus dimensiones más grandes. Una vez que tenemos los 4 puntos **(*x*,*y*)** y sus correspondientes **(*x*', *y*')**, se obtiene un sistema de ecuaciones lineales o una ecuación matricial. Se resuelve el sistema y aplicamos píxel a píxel la Transformación Proyectiva generando el valor de gris correspondiente por interpolación y así la proyección de la imagen en un nuevo plano frontal rectangular.

El código que realiza todas estas operaciones, se encuentra en la función "corrección", Anexo K.

Con lo cual se obtiene la imagen corregida de la figura 4.14.a en una nueva matriz 4.14.b.

(a) Imagen Original **(b)** Imagen corregida

Figura 4.14. Corrección de perspectiva de la placa

4.5 Estructura del algoritmo de corrección de perspectiva

La corrección de perspectiva está implementada en la función **corrige_perspectiva2.m** elaborada en Matlab. El algoritmo ha sido dividido en tres etapas, la primera de localización de vértices de la placa, la segunda de verificación de la ubicación de los vértices, y la tercera finalmente de la proyección de la imagen en un plano frontal (Transformación Proyectiva).

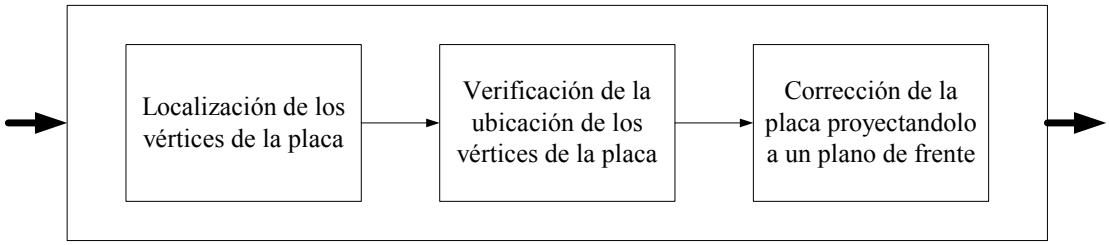


Figura 4.15. Etapas de la función **corrige_perspectiva2**

La función tiene como dato de entrada la imagen de la placa en una matriz, y da como resultado la imagen proyectada sobre un plano frontal, ya sin distorsión por perspectiva Fig. 4.16. La transformada de Hough y la transformación Proyectiva son partes esenciales de esta función.

(a) Imagen de Entrada **(b)** Imagen de Salida

Figura 4.16. Matrices de entrada y salida del bloque de corrección de perspectiva
(Imagen de archivo: IMG_0030.jpg)

En cada etapa hubo algunos problemas que se fueron superando con el curso del desarrollo de los algoritmos. A continuación se expone con más detalle cada etapa, con las soluciones a los problemas que se presentaron elaboradas para un desempeño más óptimo.

4.5.1 Etapa de la Localización de los Vértices de la Placa

El objetivo de esta etapa es localizar los vértices de la placa a través de las intersecciones de dos rectas horizontales y dos rectas verticales los cuales encierran correctamente la placa.

La imagen de entrada, para nuestro caso era la matriz **zona2**, que es sometida a la máscara de Sobel para utilizar sólo los puntos de los bordes, pero si utilizábamos los bordes de los cuatro lados esto inducía error, como se ve en las figuras 4.18 y 4.19.

Figura 4.17. Archivo: IMG_0030.jpg

Figura 4.18. Resultado de aplicar edge (zona2,'sobel')

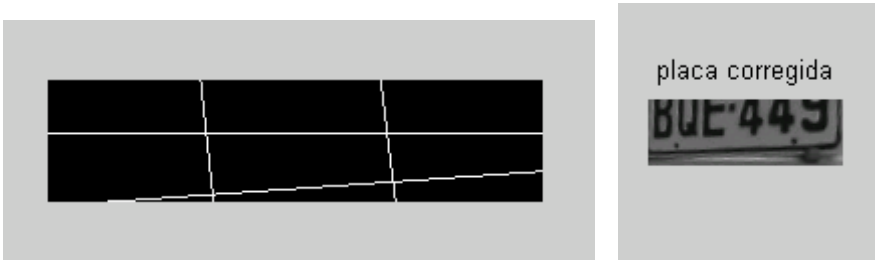


Figura 4.19. Aplicando la corrección en la figura 4.18.

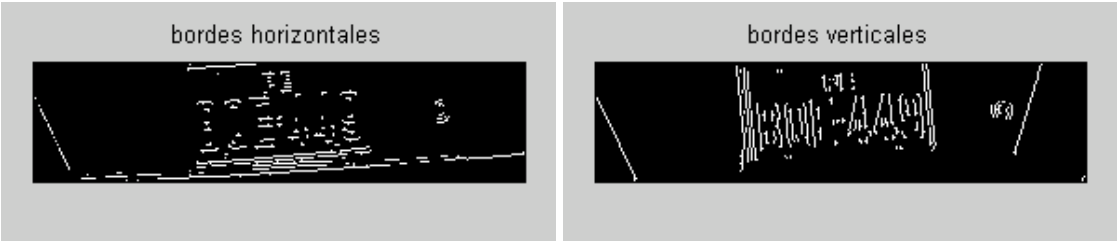


Figura 4.20. Separación de bordes para determinar cada tipo de recta

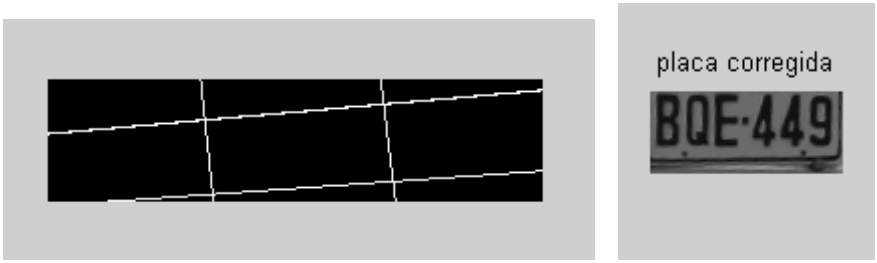


Figura 4.21. Al separar los bordes se obtiene un mejor resultado

Hallando por separado los bordes horizontales para determinar las rectas correspondientes y los bordes verticales para los laterales se obtuvo mejores resultados como en la Fig. 4.21.

Pero hubo casos en los que pese a estos cambios no se determinaban las rectas adecuadas como la siguiente figura de archivo Fig. 4.22 y 4.23

Figura 4.22. Archivo: IMG_0018.jpg

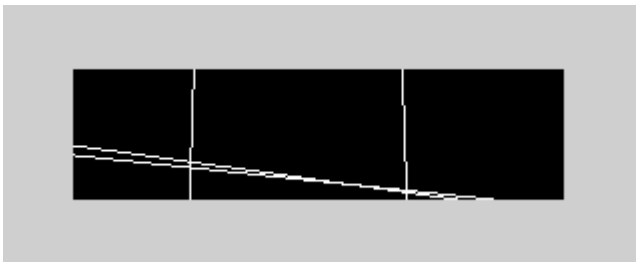


Figura 4.23. Las rectas halladas no encerraban la placa

En algunas placas las rectas horizontales y verticales no encerraban la placa, se cruzaban o coincidían, ante esta situación se planteó dividir la imagen en 2 partes, una parte superior y otra inferior llamadas **Mitad1** y **Mitad2** respectivamente, a cada una se le aplicó la transformada de Hough, y así se obtuvo una mejora muy significativa. El código quedó de la siguiente manera:

```
A=edge(zona2,'sobel','horizontal');
figure,imshow(A)
[m,n]=size(A);
Mitad1=A(1:round(m/2),1:n);
figure,imshow(Mitad1)
Mitad2=A(round(m/2):m,1:n);
figure,imshow(Mitad2)
```

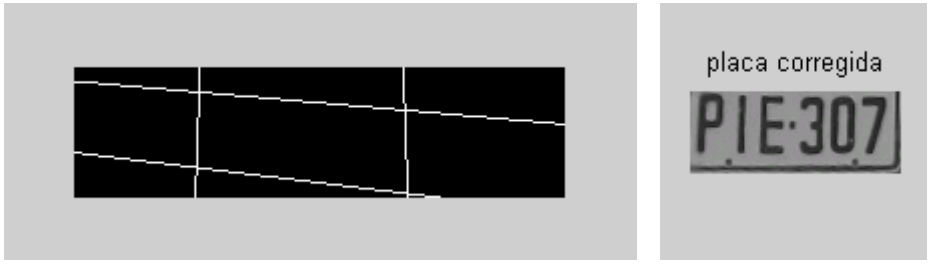


Figura 4.24. Dividiendo la matriz original en dos partes y aplicando Hough por separado se superó este problema

Pero dividir la imagen en dos partes, implica un cálculo computacional adicional debido a que ahora se tiene que hallar la transformada de Hough para cada una de las partes de la imagen, para compensar un poco esto se limitó el ángulo de trabajo entre -30 y 30 grados sexagesimales para las rectas horizontales, como figura en la línea del programa.

```
theta = -30:30;
```

Estos límites se tomaron de manera estadística tomando como elementos a cada una de las placas existentes en la base de datos.

Para los bordes verticales se hizo el cálculo de la transformada una sola vez en toda la imagen, y luego buscamos una recta vertical en la mitad izquierda y otra recta en la mitad derecha. Para esto utilizamos la matriz del espacio paramétrico discretizado $NV(\rho,\theta)$ que mantiene la cuenta de la cantidad de puntos alineados, cuyas filas corresponden a los valores discretos de ‘rho’ (eje vertical) y sus columnas a los diferentes ángulos ‘theta’ (eje

horizontal), en esta matriz tomamos la mitad que representan los valores más grandes de ‘rho’ y que corresponde a las rectas verticales del lado derecho:

NV2=NV(round(L4/2+1):L4,1:L2); % los mayores valores de ‘rho’

De la misma forma para la recta vertical izquierda tomamos los valores menores de ‘rho’, representados en las primeras filas de NV, y que corresponden a las rectas verticales del lado izquierdo:

NV3=NV(1:round(L4/2+1),1:L2); % los menores valores de ‘rho’

La diferencia con las rectas horizontales es que aquí no dividimos la imagen para aplicar dos veces la transformada sino simplemente aplicamos una vez la transformada de Hough y buscamos las rectas en el espacio paramétrico.

Figura 4.25. Las rectas verticales se buscan en cada mitad izquierda y derecha

En el caso de las rectas verticales también se limitó el ángulo para hallar la transformada de Hough en este caso tomamos el rango de **70º a 110º:**

theta = 70:110;

Reduciendo así la carga computacional y optimizando el programa.

En algunos casos se presentó poco contraste entre el color del vehículo y el marco de la placa, esto puede dificultar la determinación de los bordes laterales, dando un resultado inesperado. Para mejorar el nivel de bordes, se incrementa el umbral de gradientes, en el caso del Matlab se utilizó el comando:

B = edge(zona2,'sobel',0.045,'vertical');

Donde 4.5% indica el nivel de sensibilidad, es decir tomar los gradientes que superen este valor porcentual relativo al máximo. Con ello se obtienen más puntos de los bordes y la transformada de Hough puede determinar mejores rectas que se aproximen al marco. En las figuras siguientes se puede apreciar los cambios.

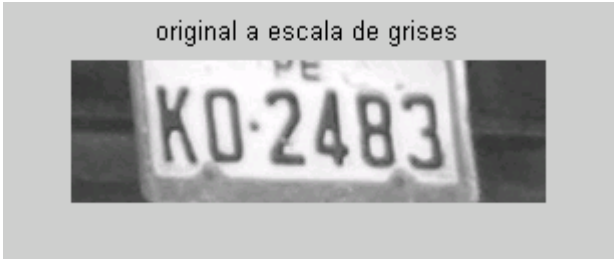


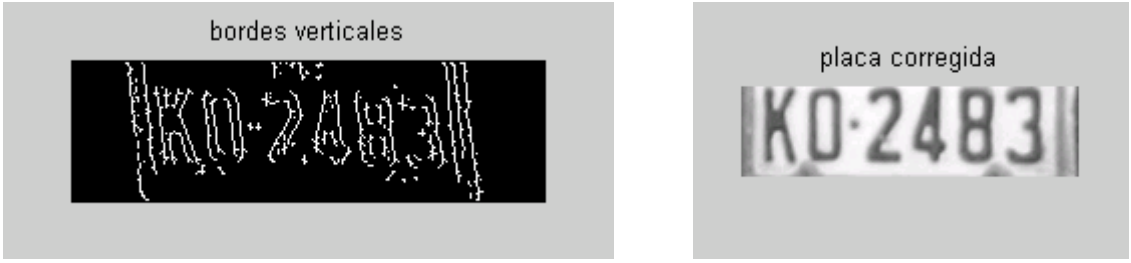
Figura 4.26. Imagen de archivo: IMG_0012.jpg

Sin hacer esta corrección tenemos la figura 4.27, con los bordes en la parte (a) y la placa corregida en la parte (b).

(a) Bordes verticales **(b)** Imagen recortada

Figura 4.27 Bordes verticales hallados con poca sensibilidad en Sobel y su correspondiente corte final

Después de la corrección de la sensibilidad se tiene lo siguiente:



(a) Bordes verticales corregidos **(b)** Imagen recortada

Figura 4.28 Bordes verticales hallados incrementando la sensibilidad en Sobel. El corte obtenido es mejor que en la figura anterior.

Como se podrá notar con la variación de la sensibilidad aumentan los bordes y se tiene un mejor corte de la placa corregida, lo cual ayuda en las etapas posteriores como por ejemplo la extracción de los caracteres, en este caso resalta más cuando queremos extraer el primer carácter de la placa cuya letra es la “K”.

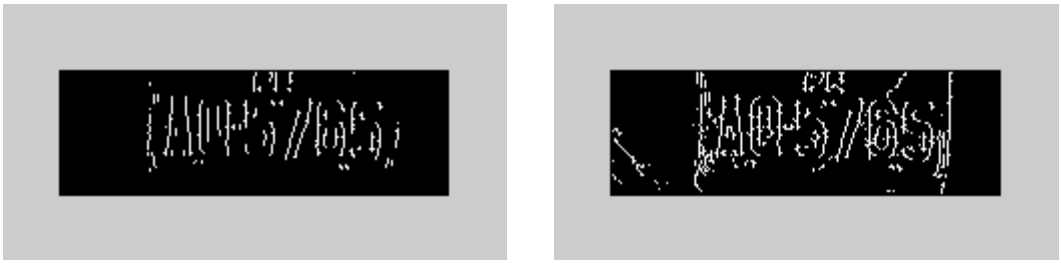
(a) **(b)**

Figura 4.29 Comparación en la extracción de un carácter, al incrementar la sensibilidad en Sobel

La imagen de la izquierda muestra la extracción del primer carácter (etapa de extracción de caracteres o segmentación) sin mejora de la sensibilidad y la imagen de la derecha es cuando se mejora la sensibilidad. Este proceso nos indica cuan importante son las etapas anteriores para disminuir los problemas en etapas posteriores.

Cuando los bordes representan algo mas crítico podrían perderse los caracteres, como se ve en las siguientes figuras.

Figura 4.30. Zona de la placa, Imagen de archivo: IMG_0026.jpg



(a) Poco umbral de gradientes **(b)** Imagen con mayor umbral de gradientes

Figura 4.31 Importancia de tener bordes laterales bien definidos

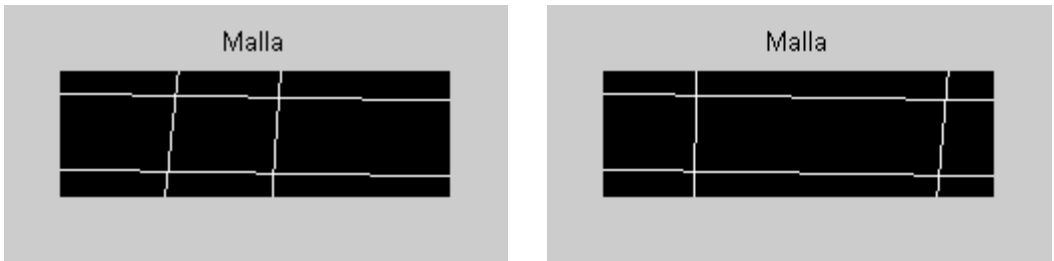


Figura 4.32. Las regiones definidas por las intersecciones de las rectas son críticas cuando no hay suficientes puntos

(a) **(b)**

Figura 4.33. Diferencias obtenidas en los cortes verticales.
El recorte mejoró con el incremento del umbral de gradientes.

Esta etapa entrega un conjunto de variables a la siguiente etapa, que constituyen sus variables de entrada:

Puntos,m,n,zona2,
theta_recta_1,theta_recta_2,
rho_recta_1,rho_recta_2,
matriz_de_h0,matriz_de_h1,matriz_de_h2,matriz_de_h3,
visualizacion

Donde:

matriz_de_h0, matriz_de_h1, matriz_de_h2, matriz_de_h3, cada matriz contiene una recta: la recta horizontal superior, la recta horizontal inferior, recta vertical derecha y la recta vertical izquierda respectivamente.

Puntos es una matriz que es la suma de:

Puntos = matriz_de_h0 + matriz_de_h1+ matriz_de_h2 + matriz_de_h3.

theta_recta_1, rho_recta_1, theta_recta_2, rho_recta_2, definen la recta horizontal superior e inferior respectivamente.

zona2 es la matriz conteniendo la imagen de la zona de la placa titulada “original a escala de grises”.

m y **n** nos representa al número de filas y el número de columnas de la imagen “original a escala de grises” (Figura 4.26).

Finalmente, la matriz **visualización** es la que nos da el gráfico de las rectas horizontales y verticales que encierran a la placa.

4.5.2 Etapa de Verificación de la Localización de los Vértices de la Placa

Como su nombre lo dice, en esta etapa se verifican los puntos de intersección de las rectas horizontales y verticales para luego pasar esos puntos de intersección a la siguiente etapa. Es importante verificar esto dado que si no existen los puntos de intersección

significa que anteriormente no se ha obtenido una buena imagen de la placa o que simplemente no es una placa lo que nos sirve para descartar entre otras zonas detectadas.

Las variables de entrada para esta etapa son:

Puntos,m,n,zona2,theta_recta_1,theta_recta_2,rho_recta_1,rho_recta_2,matriz_de_h0 ,matriz_de_h1,matriz_de_h2,matriz_de_h3,visualizacion

La matriz **Puntos**, se divide en cuatro partes, Fig. 4.34, en cada una de ellas se buscará una intersección para cada vértice, ya que hay casos en que las rectas horizontales y verticales no se interceptan como se muestra en el caso de la figura 4.35.

Parte_1	Parte_2
Parte_4	Parte_3

Figura 4.34. Matriz Puntos particionada

Figura 4.35. Imagen de archivo: Foto_0035.jpg (Fotos Restantes). El parachoque del vehículo en la imagen presenta más contrastes en la parte superior e inferior que los bordes de la placa.

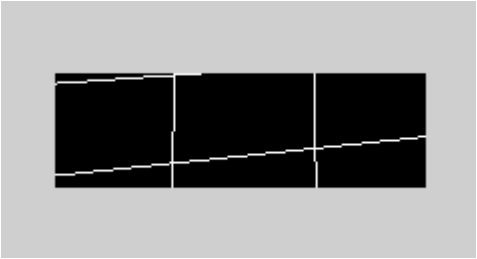


Figura 4.36 Las rectas obtenidas de la figura 4.35 no encierran la región adecuadamente, debido a la presencia de contrastes más definidos en zonas más alejadas de la placa.

Al aplicar los algoritmos a esta imagen, la recta horizontal superior no se corta con una de las verticales Fig. 4.36, como sabemos es necesario contar con cuatro vértices para hallar la matriz de transformación. Para superar esta contrariedad se añadió al algoritmo la posibilidad que la recta horizontal baje o suba de una unidad en una unidad, manteniendo su inclinación, hasta que encuentre un punto de intersección con la recta vertical que falta. Esto podemos apreciarlo en la figura 4.37.

Figura 4.37 Proceso iterativo automático de búsqueda de intersecciones,
el proceso termina al encontrar los dos puntos de corte

De esta manera una vez que encuentra la intersección que falta ya tiene los 4 vértices y puede continuar con la siguiente etapa de corrección. Al aplicar toda la función de corrección sobre la imagen original (Fig. 4.35) queda como se muestra en la figura 4.38.

Figura 4.38 Imagen corregida de la figura 4.35, con la técnica descrita.

Un ejemplo de una placa cuya recta horizontal inferior no intercepta con una recta vertical se muestra a continuación.

Figura 4.39. Imagen de archivo: Foto_0013.jpg (Fotos Restantes)

Figura 4.40. Las rectas no pudieron cerrar debido a un recorte en los bordes de la placa.

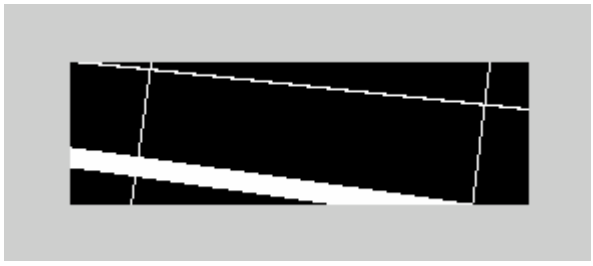


Figura 4.41. La recta horizontal inferior comienza a subir hasta encontrar intersección con la vertical derecha.

Figura 4.42. Imagen corregida con la Transformación Proyectiva y la contribución de la Transformada de Hough

Los datos de salida de esta etapa para la etapa para la proyección en un plano de frente son: Puntos, m, n, zona2.

4.5.3 Etapa de corrección de la placa proyectándolo a un plano de frente

En esta parte los datos de entrada son las variables: **Puntos**, **m**, **n, zona2**, la salida es la imagen de la placa corregida, esta parte básicamente es la aplicación de la transformación proyectiva.

De la matriz **Puntos** se busca las intersecciones de las rectas verticales y horizontales guardando el número de fila en que se encuentran en una variable denominada ‘aa’ y el número de columna en una variable denominada ‘bb’ luego se ordenan los vértices de tal manera que aparezcan en orden comenzando por el extremo superior izquierdo y en sentido horario. Se definió una matriz de dimensiones 4x2 que contiene el número de fila y

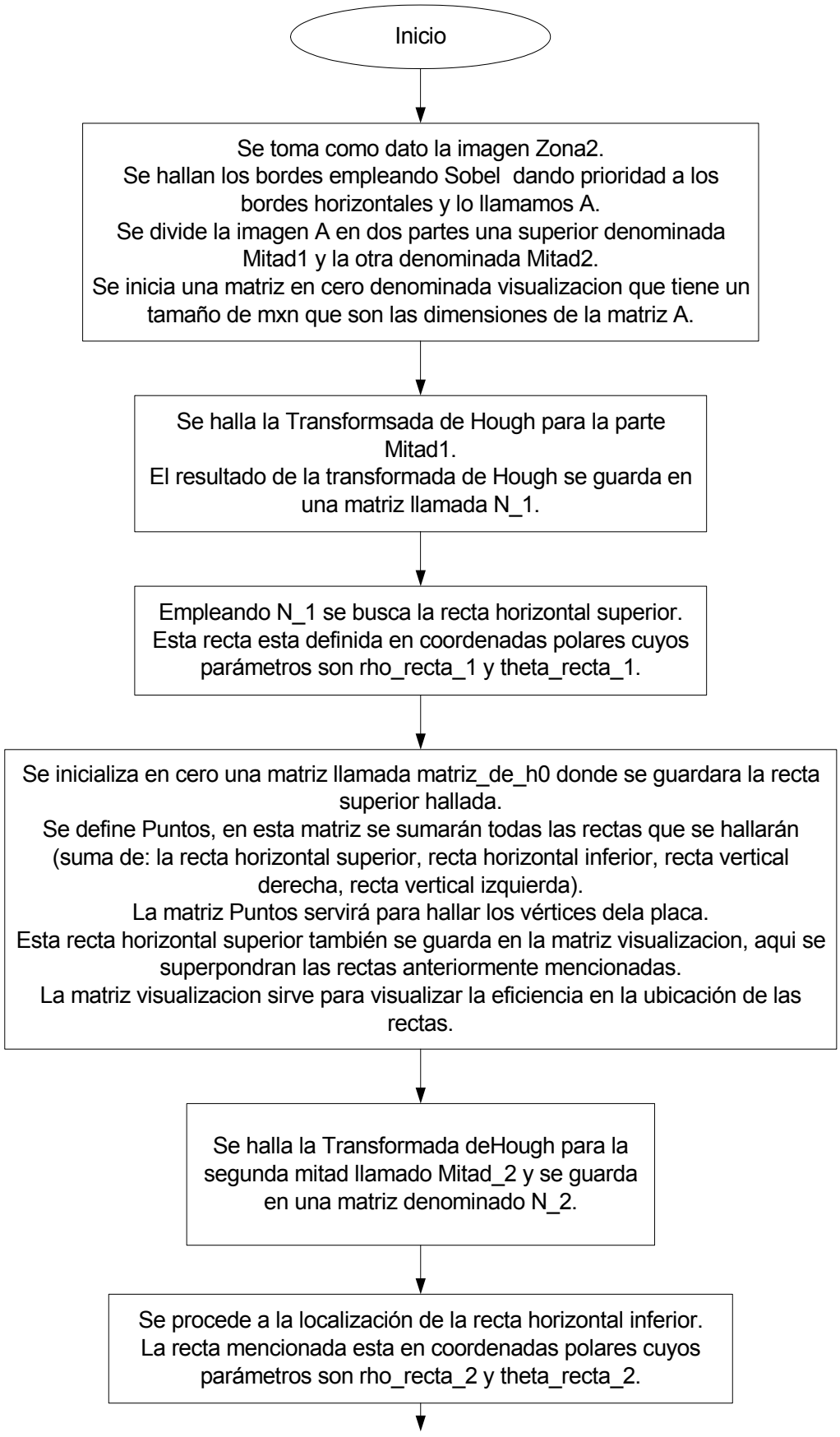
el número de columna en el orden mencionado, ésta se denominó “vértices”. Estos valores se asignarán a las coordenadas (X,Y) de la imagen original como en la sección 4.4.

$$vertices = \begin{bmatrix} aa(1) & bb(1) \\ aa(2) & bb(2) \\ aa(3) & bb(3) \\ aa(4) & bb(4) \end{bmatrix} \Rightarrow \begin{bmatrix} X1 & Y1 \\ X2 & Y2 \\ X3 & Y3 \\ X4 & Y4 \end{bmatrix} \qquad \text{Ec. 4.7}$$

Con las coordenadas (x,y) asignadas de las máximas dimensiones de la imagen original, se forma el nuevo rectángulo (plano de frente) ya corregido, y se procede a hallar la matriz H. Luego simplemente aplicamos el algoritmo de proyección píxel a píxel como se vio en el capitulo 4.4. Con esto la imagen ya corregida esta lista para seguir con el próximo paso, su Binarización.

Para terminar con esta parte mostramos en diagramas de flujo la descripción del algoritmo. El código fuente se encuentra en el **Anexo K**.

DIAGRAMA DE BLOQUES PARA EL PROGRAMA DE LOCALIZACION DE VERTICES



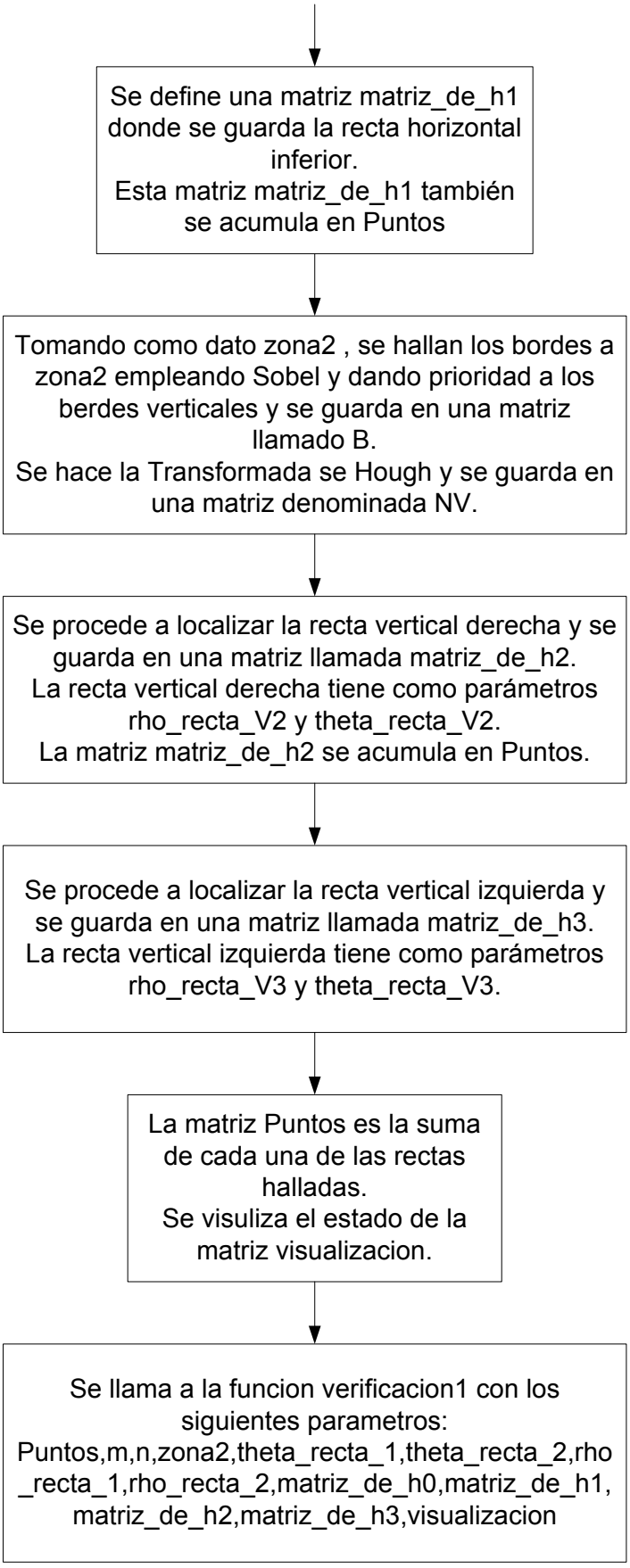
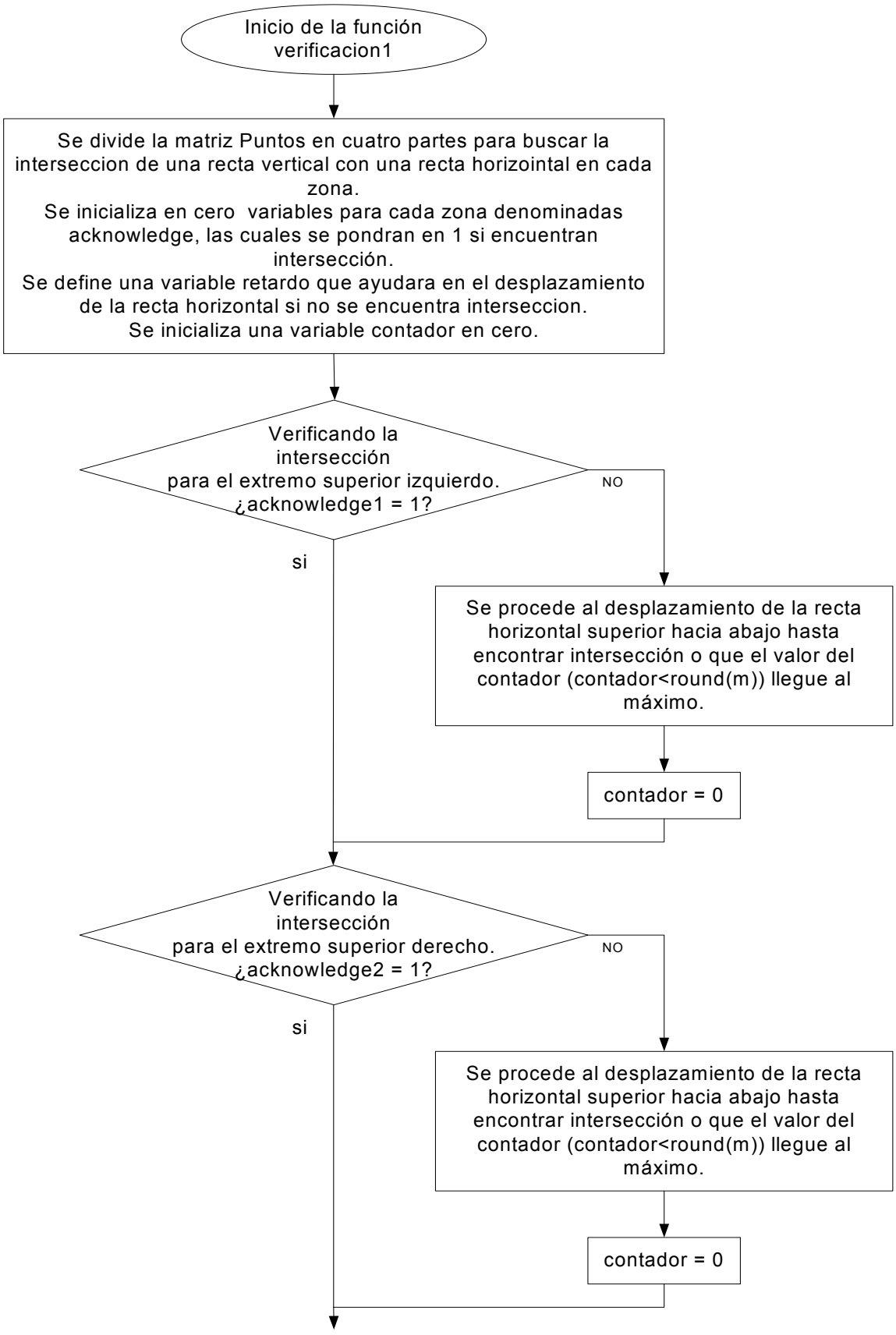


Figura 4.43 Proceso de localización de vértices

DIAGRAMA DE BLOQUES PARA LA FUNCION VERIFICACION1



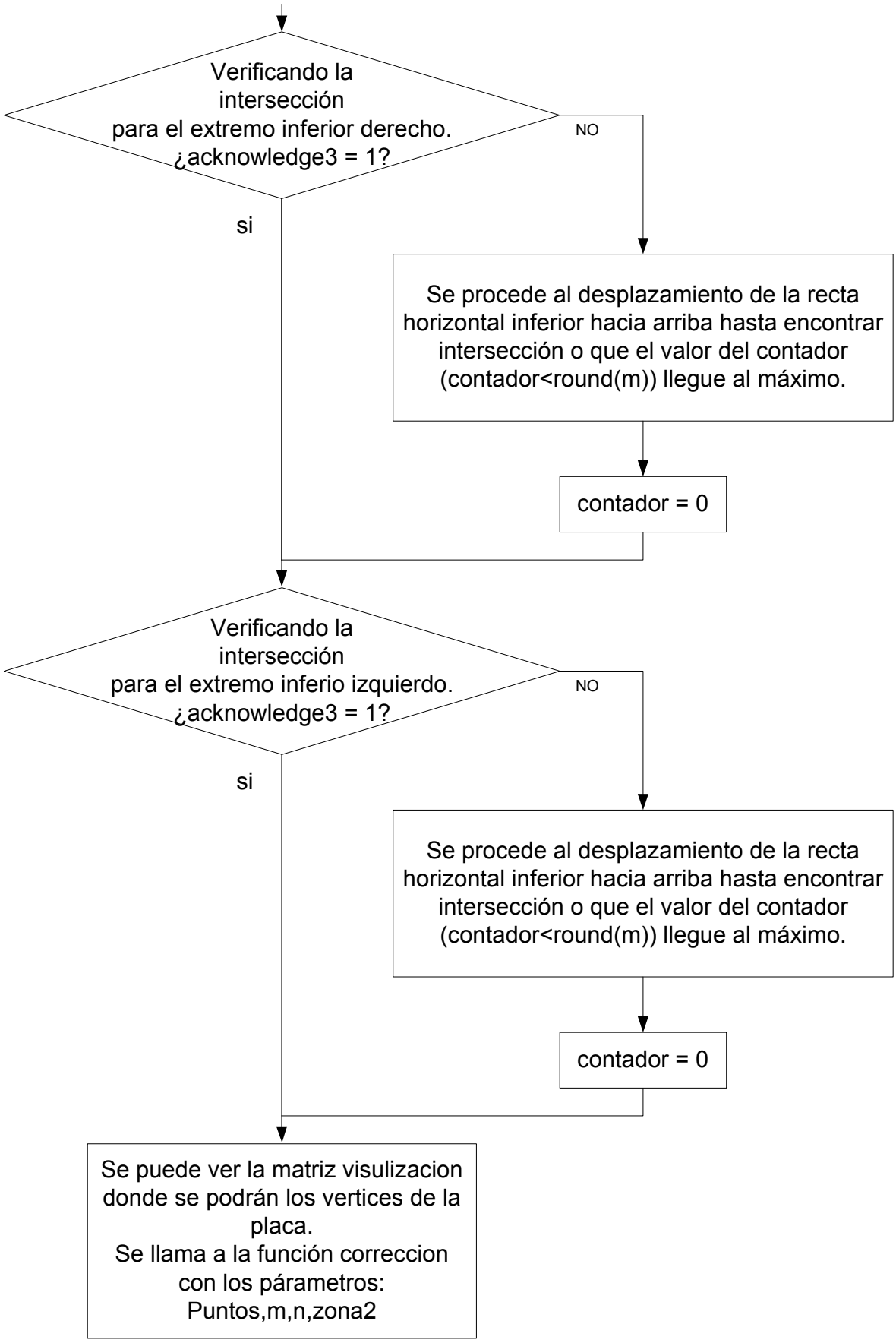


Figura 4.44 Proceso de verificación de los vértices apropiados

DIAGRAMA DE BLOQUES PARA LA FUNCION CORRECCION

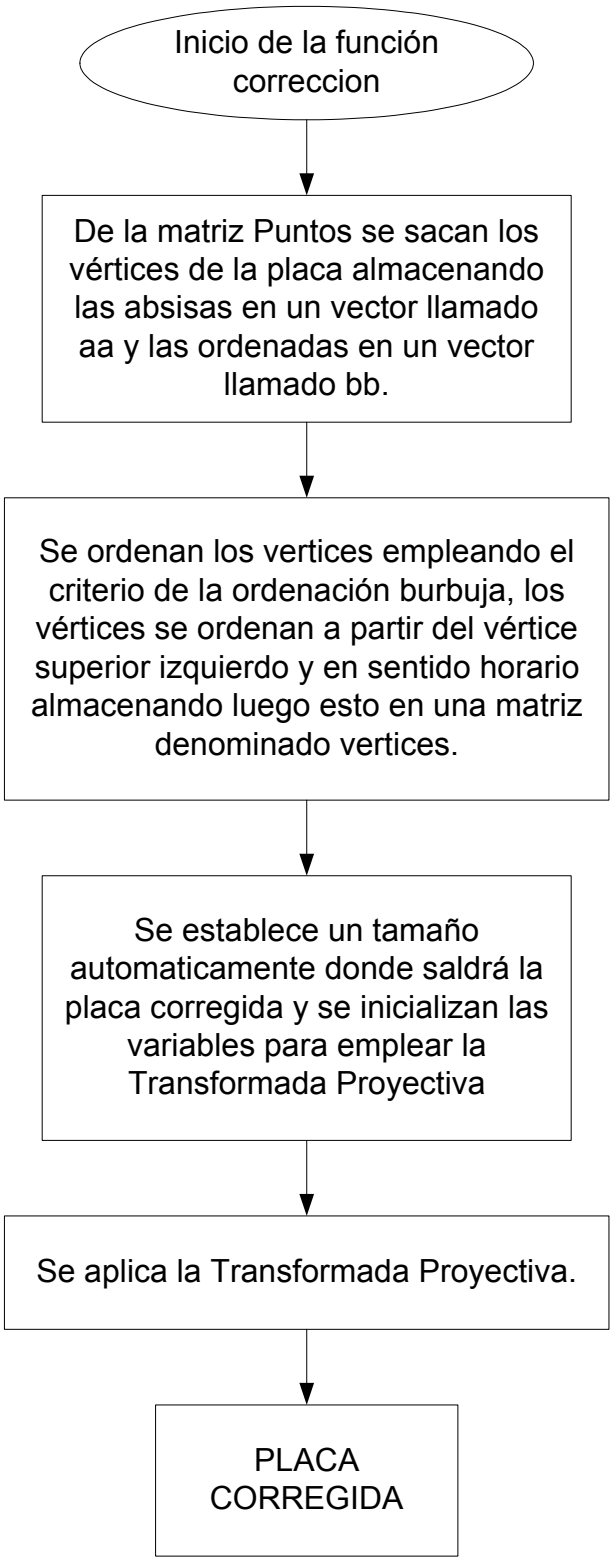


Figura 4.45 Proceso de correción de la imagen (función “corrección”).

CAPÍTULO V
BINARIZACIÓN (THRESHOLDING)

5.1 Introducción

La Binarización consiste en agrupar los píxeles de una imagen en escala de grises en 2 clases separadas, **Blanco** (‘**1**’) para “píxeles claros” y **Negro** (‘**0**’) para los “oscuros”. Esto responde a la necesidad de simplificar los datos, ya que manejar datos binarios (0 y 1) para las posteriores etapas es más sencillo que trabajar con 256 niveles. Esto debe llevarse a cabo de manera que la forma de los caracteres no se afecte, es decir sin distorsionarlos o en el peor de los casos perderlos. Este procedimiento tiene varias denominaciones en la literatura: Separación de niveles, Binarización, Umbralización, o en inglés: Thresholding.

El proceso parece simple pero cuando tomamos los métodos basados en el criterio de elegir un umbral de gris que separa los píxeles en dos clases (píxeles claros y oscuros a valores blanco y negro respectivamente) **[9]**, esto no siempre da los resultados esperados, pues algunas veces se presentan imágenes algo opacas, con mucha luz ó con alguna pequeña sombra que puede invalidar el umbral elegido. Existen pues dos criterios, aquellos métodos basados en la clasificación de píxeles en dos categorías y aquellos métodos que no solo consideran el valor de gris de los píxeles sino su grado de conectividad entre ellos (como el método split and merge, o region growing), otros basados en gradientes y la dirección de éstos para determinar la relación entre píxeles. Fueron estudiados varios métodos y sometidos a pruebas con la base de datos. Primero exponemos los conceptos de segmentación de imágenes en un sentido general, y luego enfocamos en la segmentación de bajo nivel haciendo un recuento de sus técnicas de segmentación que nos sirvieron para estudiar y afrontar el problema. Así pues se mostrarán los resultados obtenidos.

5.2 Segmentación de Imágenes

Las técnicas para extraer información de una imagen están dentro del campo del procesamiento que se denomina *“análisis de imágenes”*. El primer paso en el análisis de imágenes consiste primero en segmentar la imagen. **La segmentación subdivide una imagen en sus partes constituyentes u objetos.** El nivel de subdivisión depende del

problema a resolver, así entonces el proceso se detiene cuando los objetos de interés de una aplicación hayan sido **aislados**. **[12]**

En general la segmentación autónoma es una de las tareas más difíciles del procesamiento de imágenes. Por ello se debe poner un considerable cuidado en aumentar la probabilidad de tener una segmentación robusta.

5.2.1 Niveles de Segmentación

La segmentación de imágenes tiene su origen en numerosos estudios psicológicos que indican la preferencia de los humanos por agrupar regiones visuales en términos de proximidad, similitud y continuidad, para construir un conjunto de unidades significativas. Es pues la segmentación un proceso que se clasifica según su nivel de abstracción. Un primer nivel estaría relacionado con solo marcar los puntos de la imagen (píxeles) con un valor indicativo de su pertenencia a determinada región o clase. La segmentación a un nivel de abstracción medio implica además de lo anterior, proveer un mecanismo que permita una representación simbólica de las relaciones topológicas existentes entre las distintas unidades. En esta etapa entramos a una segmentación de bajo nivel por lo que a continuación se expone los métodos existentes más conocidos.

5.3 Segmentación de Bajo Nivel

5.3.1 Métodos Orientados a Píxel

Estos métodos asocian un píxel de una imagen a una determinada clase según su nivel de gris, pero ignoran las nociones de proximidad y conectividad suponiendo que los objetos se distinguen únicamente por su valor de intensidad. En este grupo se encuentran los métodos como el de Otsu, el de Selección Iterativa, aquellos basados en la reestructuración del histograma, otros utilizando “lógica difusa”, algunos basados en la entropía de la imagen, otros diseñados especialmente para el caso de placas (Método Específico) y los métodos adaptivos que binarizan la imagen por bloques. Si bien la implantación de estos métodos suele llevar a soluciones relativamente rápidas y sencillas aún así están limitados en general, y en particular aquellos casos en que la imagen tiene una iluminación no uniforme (con presencia de sombras) que pueden invalidar el valor umbral de gris elegido.

En general el proceso de binarización consiste en determinar un valor umbral de gris U, de la forma:

$$U = U(\mathrm{x}, \mathrm{y}, p(\mathrm{x}, \mathrm{y}), f(\mathrm{x}, \mathrm{y})) \qquad \text{Ec. 5.1}$$

Donde *f*(x, y) es el nivel de gris del punto (x,y), y *p*(x,y) representa alguna propiedad local de este punto, aquel criterio adicional para poder elegir entre un nivel u otro, en esa vecindad centrada en (x,y). Una imagen binarizada *g*(x, y) **[9]** se define como:

$$g(x,y) = \begin{cases} 1 & \text{si} \quad f(x,y) > U \\ 0 & \text{si} \quad f(x,y) \leq U \end{cases} \qquad \text{Ec. 5.2}$$

De esta manera los pixeles marcados con 1, corresponden al fondo (color blanco) y aquellos marcados con 0, serán los caracteres y probablemente algunas otras zonas como el marco de la placa.

a) Método de Otsu y sus variantes

Este método descrito en **[13]** se basa en el histograma de la imagen en escala de grises y se basa en la suposición que en la distribución del histograma existen 2 grupos naturales predominantes de valores de intensidad (Distribución Bimodal) que llamamos "clases". Después de calcular el histograma podemos inferir algunos parámetros estadísticos. Suponemos un umbral U que define dos clases C1 con niveles de gris [0,…U] y C2 con niveles de gris [U+1,….L]. Entonces se calcula la media y varianza dentro de cada clase lo que supone aplicar el concepto de probabilidad condicional a los casos $l \leq U$ y $l > U$.

$$w_1 = \sum_{l=0}^{U} p(l) \qquad \text{Ec. 5.3} \qquad\qquad w_2 = \sum_{l=U+1}^{255} p(l) \qquad \text{Ec. 5.4}$$

$$\mu_1 = \sum_{l=0}^{U} lp(l)/w_1 \quad \text{Ec. 5.5} \qquad\qquad \mu_2 = \sum_{l=U+1}^{255} lp(l)/w_2 \quad \text{Ec. 5.6}$$

$$\sigma_1^2 = \sum_{0}^{U} (l-\mu_1)^2 p(l)/w_1 \quad \text{Ec. 5.7} \qquad \sigma_2^2 = \sum_{U+1}^{255} (l-\mu_2)^2 p(l)/w_2 \quad \text{Ec. 5.8}$$

Así tenemos:

$p(l)$ = Probabilidad de nivel de gris l.

w_1 = Probabilidad estimada que el valor del píxel caiga en el grupo 1 ($l \leq$ U).

w_2 = Probabilidad estimada que el valor del píxel caiga en el grupo 2 ($l >$ U).

μ_1, μ_2 = Valores medios correspondientes.

$\sigma_1{}^2$, $\sigma_2{}^2$ = Varianzas correspondientes.

Después de eso podemos definir la "**varianza conjunta** después de aplicar el umbral U":

$$\psi^2 = w_1\sigma_1^2 + w_2\sigma_2^2 \qquad \text{Ec. 5.9}$$

El algoritmo consiste simplemente en calcular $\boldsymbol{\Psi^2}$ para todos los posibles umbrales, se hace el cálculo para todos los niveles de 0 a 255 y se usa el que produce el máximo. Podemos ver que las fórmulas son adecuadas para un cálculo iterativo y rápido ya que se puede calcular las sumas para un valor de umbral a partir de las anteriores: sumando o restando un valor.

Este método es bastante robusto ante cualquier tipo de histograma y de hecho es el más referenciado y utilizado de la literatura.

Existen variantes como la que nos muestra **[14]**, que básicamente es la simplificación de la varianza conjunta a partir de redefinir la expresión anterior como la "**varianza entre clases**":

$$\sigma_B{}^2 = w_1(\mu_1 - \mu_T)^2 + w_2(\mu_2 - \mu_T)^2 \qquad \text{Ec. 5.10}$$

Donde: $\mu_T = w_1\mu_1 + w_2\mu_2$, es la media total.

De manera que ahora se debe hallar el valor de umbral de "U", tal que la "varianza entre clases" sea máxima. Esta varianza se puede simplificar aún más, desarrollando se obtiene:

$$\sigma_B{}^2 = w_1\mu_1^2 + w_2\mu_2^2 - \mu_T^2 \qquad \text{Ec. 5.11}$$

Pero maximizar la varianza entre clases es maximizar la expresión:

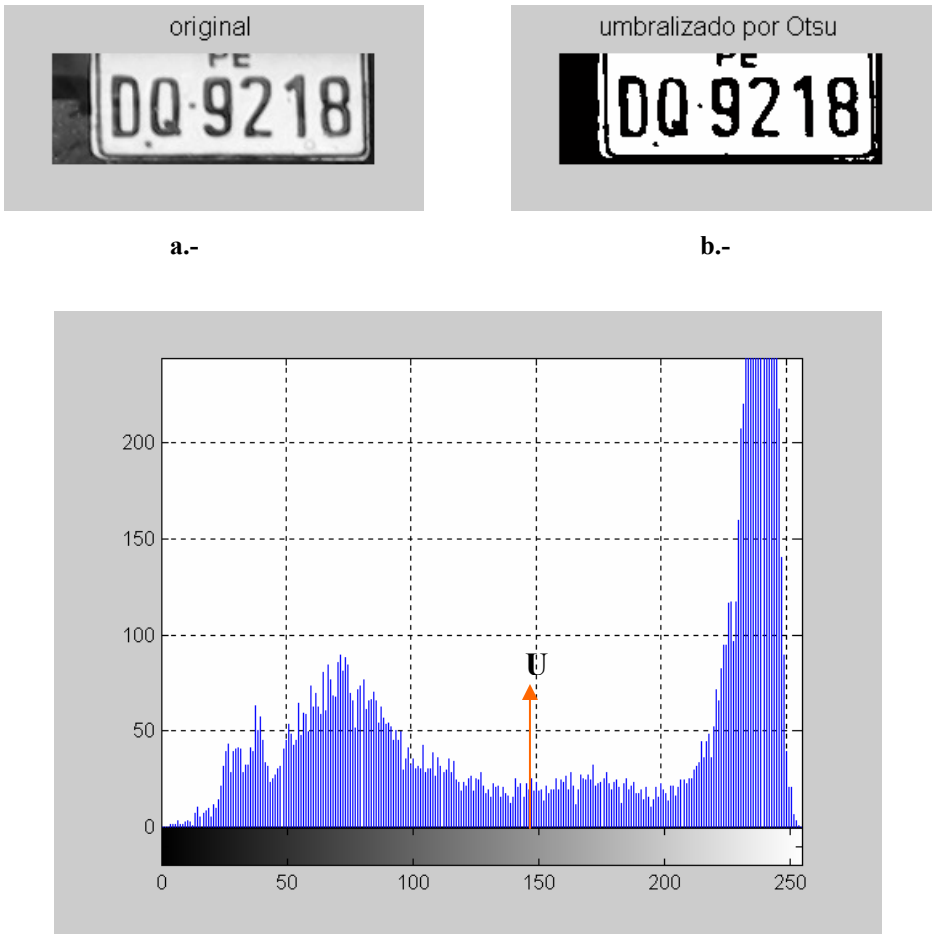
$$\sigma_C{}^2 = w_1\mu_1^2 + w_2\mu_2^2 \qquad \text{Ec. 5.12}$$

Dado que μ_T es una constante y que sólo nos interesa hallar el valor U para el cual la varianza entre clases es máxima.

Aunque la teoría es coherente para un histograma que tienda a distribución bimodal de imágenes, lo hace bien para histogramas multimodales. Pero existen casos de imágenes de placas con fondo amarillo que en escala de grises se tornan algo oscuras lo que al hacer el cálculo por este método falla sustancialmente, y es por que el umbral adecuado no se encuentra en un valle como sería de esperar normalmente. En los siguientes ejemplos podemos notarlo con mayor claridad.

➢ ***Ejemplo 5.3.1***

En este caso se puede ver claramente del histograma que se trata de una distribución bimodal, el método de Otsu nos dio un **Umbral de 153**, lo que significa que a partir de ese valor aquellos píxeles menores al umbral serán negros (**'0'**) y los mayores serán blancos (**'1'**). Se aprecia fácilmente en la **figura 5.1.c** que el umbral se encuentra en el valle entre los picos más altos de la distribución.



c.- Histograma de la Imagen original: IMG_0005.jpg, se muestra el umbral hallado U = 153.

Figura 5.1 Binarización de la imagen (a) usando el método de Otsu, se obtiene la figura (b)

- ***Ejemplo 5.3.2***

Figura 5.2 Imagen de archivo: IMG_0014.jpg

Figura 5.3 Placa localizada con algoritmo del Capítulo de Localización

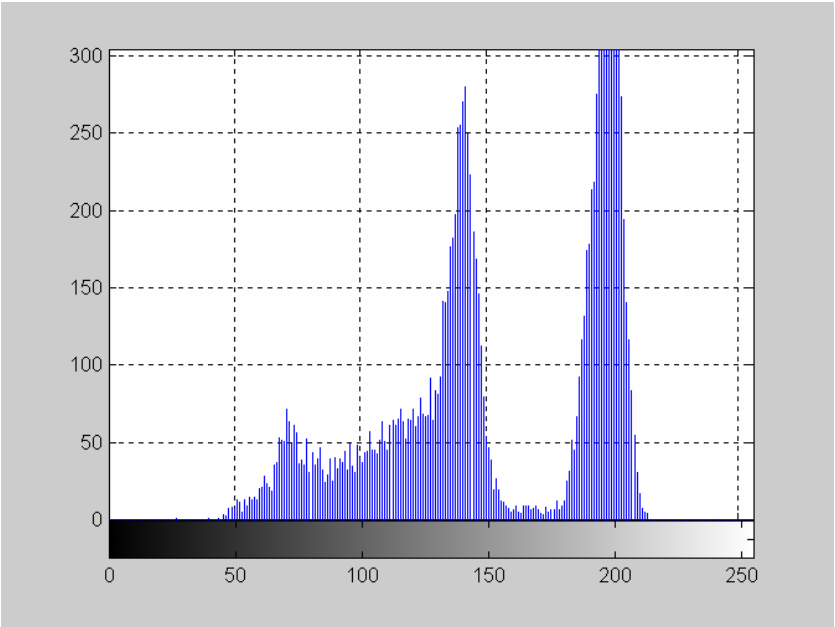


Figura 5.4 Histograma de la imagen original

El método de Otsu nos dio un valor de umbral: 158 como es lógico se encuentra en el valle del histograma, pero el resultado de la Binarización se ve en la figura 5.5.

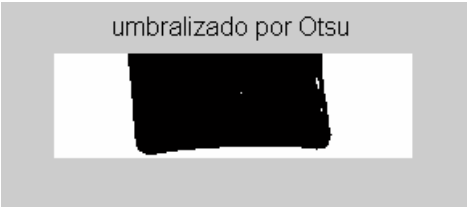


Figura 5.5 Imagen Binarizada por el método de Otsu

La imagen se oscureció sensiblemente, y no se pudo hacer nada para remediarlo ya que el método no tiene ningún otro parámetro que pueda modificar el resultado. Fuera de estos casos el método trabaja aceptablemente bien.

b) Método de Selección Iterativa

El método de Selección Iterativa parte de un valor inicial aproximado del umbral, el cual se recalcula y mejora en sucesivos pasos. Este método no emplea el histograma, sino que segmenta repetidas veces la imagen entre las dos clases (fondo-objeto) y emplea los niveles de cada clase (en la imagen original) para mejorar el valor del umbral. El valor aproximado inicial (T_i) es el valor medio de gris de la imagen original. A partir de este valor se estiman los estadísticos de las dos regiones (blanca y negra) obtenidas. Así, se calcula primero el valor medio de los píxeles por debajo del umbral inicial T_i (T_N) y, después, el valor medio de los píxeles de valores igual o superiores a T_i (T_B). El valor del nuevo umbral se recalcula como $(T_N + T_B)/2$. Este proceso se repite sucesivas veces hasta que el nuevo valor calculado sea el mismo en dos iteraciones consecutivas **[15]**.

Este método tuvo un rendimiento regular similar al de Otsu, que también adolece de fallar en aquellos casos como el del ejemplo 5.3.2 mostrado anteriormente.

c) Método Específico

Este es un método diseñado especialmente para este problema. La base del método es que hemos calculado el gradiente de la imagen en la etapa previa de localización. Si nos movemos a lo largo de la imagen, el gradiente nos dirá donde hay cambios del nivel de gris de fondo al de texto y viceversa (de hecho, ésta fue la forma de localizar los caracteres). **Algoritmo**: elíjase una línea de la imagen donde la probabilidad de tener muchas transiciones es alta (por ejemplo, en el centro). Entonces, nos moveremos a lo largo de la línea y detectaremos todas las transiciones. Cada transición significará un cambio en el estado de una variable con sólo dos posibles valores. Llamaremos Q a esta variable, y sus

valores serán 0 y 1. Cuando no haya transiciones estaremos en un píxel con un nivel de gris dado al que llamaremos L. Calcúlese:

$$Sum_Q = Sum_Q + L \qquad \text{(la suma de valores para cada estado).} \qquad \text{Ec. 5.13}$$

$$Number_Q = Number_Q + 1 \qquad \text{(el número de píxeles para cada estado).} \qquad \text{Ec. 5.14}$$

Después de la línea entera calcularemos:

$$Mean_Q = Sum_Q / Number_Q \qquad \text{(la media para cada estado).} \qquad \text{Ec. 5.15}$$

El umbral es:

$$U = (Mean_0 Number_1 + Mean_1 Number_0)/(Number_0 + Number_1) \qquad \text{Ec. 5.16}$$

Este método puede ser extendido a varias líneas. La base del algoritmo es la división de píxeles en dos estados que representan, obviamente, el blanco y el negro. Entonces podemos calcular una suma global para los píxeles blancos y otra para los negros. También debemos contar los píxeles de cada categoría. La fórmula final para el umbral es la misma. En la práctica aplicamos este método extendido al 80% de las líneas de la imagen (despreciamos el primer y último 10%, por suponer que esa zona no contiene caracteres). Aquí también se tiene un rendimiento regular y aún no se supera los casos como el *ejemplo 5.3.2* o con algunas sombras de color similar.

Debemos mencionar que este método tal como se describe en [4] halla un solo umbral del análisis línea por línea, pero nosotros calculamos un umbral para cada línea generando un vector de posibles umbrales, para luego evaluar su calidad mediante un criterio también aplicado en [4] y que pasamos a describir.

Para medir la "calidad" de la binarización medimos el porcentaje de píxeles situados en la "zona dudosa" y lo comparamos con un umbral. Supongamos que el histograma es como el de la figura 5.6.

En la figura se han marcado: el umbral calculado (U), la media de los píxeles negros (M_n) y la media de los píxeles blancos (M_b). Considerando como píxeles dudosos aquellos situados en un intervalo centrado en U y de longitud X. Se calcula X como un porcentaje de la diferencia ente ambas medias, un buen valor es $X = 0.15 (M_b - M_n)$. Aquel umbral U que nos dé la menor cantidad de píxeles dentro del segmento X, será el elegido. Nótese que para cada umbral U, se debe calcular sus respectivas medias M_n y M_b.

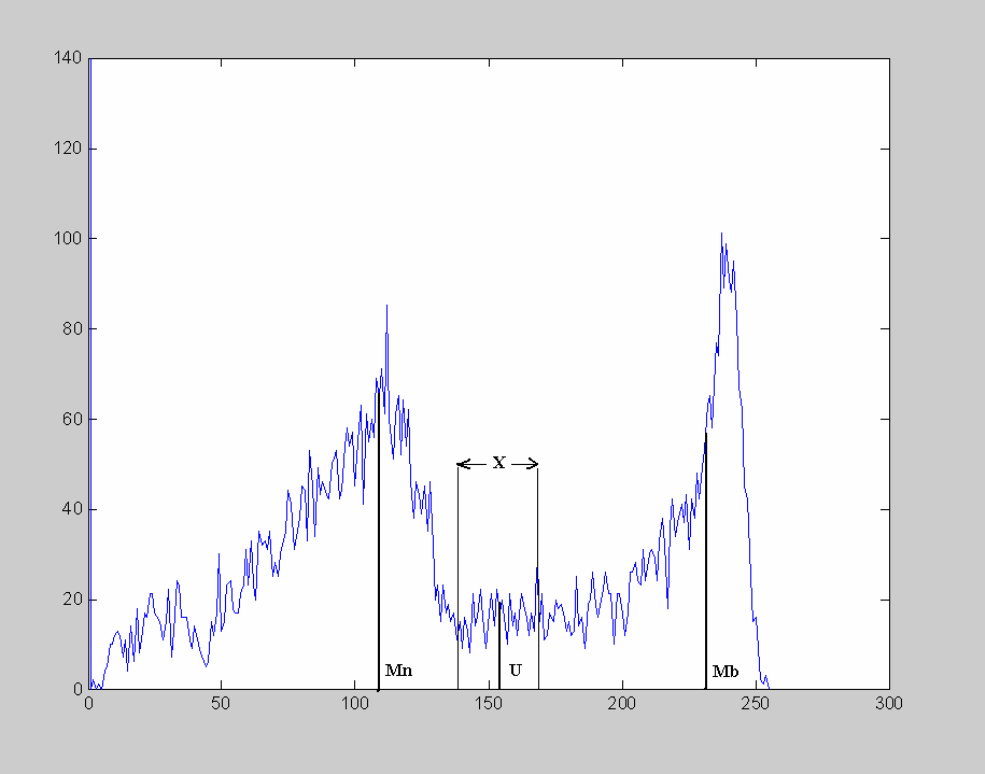


Figura 5.6 Histograma Bimodal

Hasta el momento hemos visto métodos basados en el histograma y la estadística de los píxeles considerando la media y varianza en general. Pero todavía no se resuelve el problema de casos con placas opacas. El problema se debe a que las sombras o zonas oscuras en una placa generan mayor cantidad de píxeles oscuros, **alterando la distribución de píxeles** en el histograma, esto hace que píxeles que realmente son claros "se agrupen con los oscuros" terminando finalmente en el nivel '0', por consiguiente oscureciendo el fondo de la placa y perdiendo los caracteres. Algo similar ocurre con placas con mucho brillo y al final toda la placa (caracteres y fondo) terminan de un solo color. Es evidente que en estas circunstancias con **distribuciones de grises alteradas** los métodos basados en medias y varianzas aplicados "ciegamente" a toda la imagen fallen irremediablemente "sin saberlo".

d. Método de Reestructuración del Histograma basado en características del limite para selección del Umbral

Dada la no uniformidad de las distribuciones de histogramas que pueden generarse debido a sombras o demasiada iluminación en la imagen, es necesario uniformizarlas [9]. Un método de mejorar la forma de los histogramas es considerar solamente aquellos

píxeles que están situados en el límite entre los objetos y el fondo o cerca de él. Una de las evidentes e inmediatas mejoras es que los histogramas podrán ser menos dependientes de los tamaños relativos de los objetos y el fondo. Por ejemplo, el histograma de intensidad de una imagen compuesta por una gran zona de fondo aproximadamente constante y un objeto pequeño podría estar dominado por un gran pico como consecuencia de la alta concentración de los píxeles del fondo. Pero, si solamente se utilizan los píxeles que están sobre o cerca del límite entre el objeto y el fondo, el histograma resultante tendría picos de aproximadamente la misma altura. Además la probabilidad de que cualquiera de los píxeles dados esté situado en un objeto podría ser aproximadamente igual a la probabilidad que este situado en el fondo y de este modo se mejora la simetría de los picos del histograma. El empleo de píxeles que satisfacen alguna de las medidas sencillas basadas en los operadores Gradiente tiene la tendencia a hacer más profundos los valles entre los picos del Histograma.

Una forma de clasificarlos sería usando el gradiente para detectar los píxeles que se encuentran en los bordes, para ello establecemos un **nivel mínimo de gradientes** (**T**) con lo que separamos las zonas de píxeles de color uniforme en la imagen y nos quedamos con los de los bordes que involucran necesariamente a los caracteres. Sea $s(x, y)$ la nueva distribución de gris, a partir de la imagen $f(x, y)$ y $\nabla f(x, y)$ su gradiente, así tenemos:

$$s(x,y) = \begin{cases} 0 & si \ \ \nabla f(x,y) < T \\ f(x,y) & si \ \ \nabla f(x,y) \geq T \end{cases} \qquad \text{Ec. 5.17}$$

Una vez realizado el proceso de clasificación, el histograma que presentan los píxeles del borde será más regular que el histograma de toda la imagen, ahora entonces calculamos un valor umbral U que será el que segmente la imagen original. Esto se puede apreciar en el siguiente ejemplo.

- ***Ejemplo 5.3.3***

Para aplicar este método en la figura del ejemplo 5.3.2, debemos establecer un umbral de gradiente en nuestro caso primero, normalizamos los gradientes al intervalo [0-255] y establecemos un valor de T = 7.

Figura 5.7 Imagen original a la que aplicaremos la reestructuración del histograma antes de la binarización

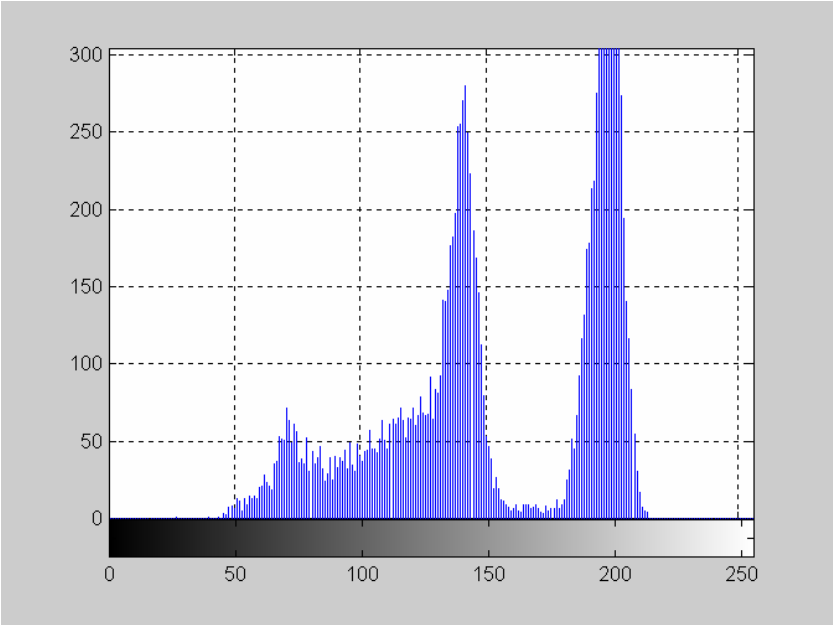


Figura 5. 8 Histograma Original

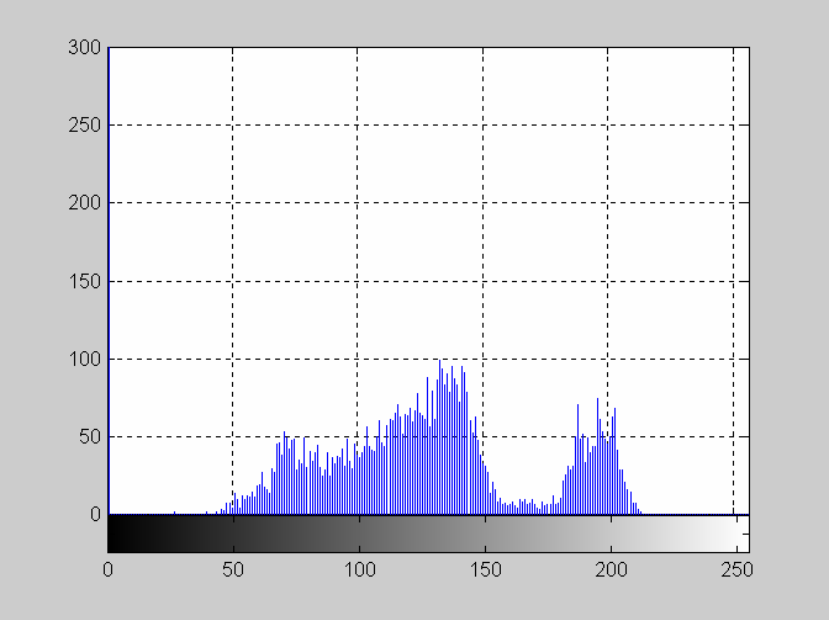


Figura 5.9 Histograma Reestructurado con solo los píxeles de los bordes

Como puede verse en la Figura 5.9 el histograma se uniformizó un poco y esto depende del nivel de umbral de gradiente "T" que se elija.

Con este método de obtuvo el umbral de 92 utilizando luego de la reestructuración del histograma el método de Específico y con la evaluación del umbral óptimo al medir su calidad, valor muy distinto de los 158 que arroja normalmente el método de Otsu o cualquier otro.

Figura 5.10 Resultado del método aplicado

Se puede notar que en el histograma original (Figura 5.8) el Nivel de Gris = 92 está lejos del valle en una zona que no es tan evidente.

Es importante señalar que en este método la reestructuración del histograma primero es un paso previo a la segmentación, pero ese paso previo depende del nivel que se le dé a "**T**", por lo que éste se convierte en un parámetro que normalmente trabaja en T = 30 para la mayoría pero en casos extremos como éste: T = 7 es un valor más adecuado.
Este parámetro se puede apreciar en el código fuente, archivo: pixels_bordes.m adjunto en el **Anexo L**.

e) Método Adaptivo

Este método básicamente consiste en subdividir la imagen en bloques pequeños y calcular su umbral respectivo con cualquiera de los métodos descritos antes (Otsu, Selección Iterativa, etc) [4]. En algunos casos da buenos resultados, sobretodo aquellos que no tienen iluminación uniforme. El Problema de este método es por un lado, la elección adecuada del tamaño de los bloques y por otro, que consume mucho más tiempo que cualquier otro método debido a que se debe aplicar el algoritmo a todos los bloques es decir varias veces, lo que lo hace demasiado lento.

5.3.2 Métodos Orientados a Regiones

Explotan la noción de conectividad para agrupar zonas de la imagen con puntos de intensidad similar en entidades discernidas. En este grupo destacan, el algoritmo "split & merge" o "división y fusión" [16], y sus variantes piramidales. Cabe destacar algoritmos basados en Programación Dinámica así como la reciente aplicación, al problema, de la Teoría de Campos Aleatorios de Markov (Markov Random Fields) y el uso de la Transformada Watershed dentro del ámbito de la Morfología Matemática. Existe una pequeña dificultad a la hora de establecer el criterio por el cual una región pequeña es engullida por otra grande, exigiendo un compromiso entre la eliminación de ruido y la pérdida de detalle. Otra dificultad es la de determinar con precisión la frontera entre regiones cuando las variaciones de intensidad son graduales, produciendo a veces un efecto de ambigüedad (bias) [12].

a) Método Split & Merge (División y Fusión)

Este método descrito en **[5]** Tiene dos reglas básicas:

- **Divide el Bloque si C(Ri) = Falso**; Que quiere decir si el criterio elegido no se cumple (que las características de homogeneidad no se mantengan, puede ser gradientes de grises menores que un umbral, o varianza, etc.) el bloque se divide en 4.
- **Fusiona dos Bloques si C(Ri U Rj) = Verdadero**; Significa que se unen dos bloques adyacentes si tienen características similares homogéneas.

Donde Ri y Rj son regiones correspondientes a bloques a analizar. El algoritmo se detiene cuando ya no es posible dividir ni fusionar. No se implementó éste método ya que requiere imágenes cuadradas (de preferencia con número de píxeles por lado potencia de 2) que puedan dividirse sucesivamente lo que no se presenta en las imágenes rectangulares de las placas. Se hizo más complejo de implementar sacrificando la línea de simplicidad que se requiere para una aplicación en tiempo real.

b) Método Region Growing (Crecimiento de Regiones)

El crecimiento de regiones es un procedimiento que agrupa píxeles o sub-regiones dentro de regiones más grandes **[16]**. Básicamente consiste en la agregación de píxeles que comienza con un conjunto de puntos generadores a partir de los que van creciendo las regiones al agregar a cada uno de estos los píxeles próximos que tienen propiedades similares, como nivel de gris, textura, color. La selección de criterios de similitud depende no solamente del problema que se está considerando, sino también del tipo de datos que se dispone. Los descriptores pueden proporcionar resultados falsos si no se utiliza la información de conectividad o adyacencia en el proceso de crecimiento de regiones.

5.3.3 Métodos orientados a contorno

Estos se basan en suponer que existe una correspondencia entre las discontinuidades en intensidad de una imagen y la frontera de los objetos contenidos en la misma. En muchos casos dicha información no es suficiente requiriendo otro tratamiento posterior.

5.4 Notas acerca de la segmentación

La segmentación, es un componente crítico en sistemas de visión artificial porque los errores en este proceso serán propagados a los procesos de análisis de nivel superior e incrementa la complejidad de las subsecuentes tareas. Idealmente las regiones segmentadas dentro de una imagen deben tener las siguientes características:

a) Las regiones deberían ser uniformes y homogéneas con respecto a alguna característica particular.

b) El interior de las regiones debería ser simple y sin muchos hoyos pequeños.
c) Las regiones adyacentes deberían tener valores significativamente diferentes con respecto a la característica sobre el cual ellos son uniformes.
d) Las fronteras de cada segmento deberían ser simples, no rasgados y deben ser espacialmente precisos.

El problema de la segmentación de imágenes naturales es básicamente una **emulación de la percepción psicológica** y por ello **no se presta para soluciones puramente analíticas**. Cualquier algoritmo matemático debe ser suplido por heurísticos usualmente involucra semántica o descripciones acerca de las clases de las imágenes bajo consideración. Algunas veces es apropiado ir mas allá de una simple heurística e introducir un conocimiento a priori de la imagen. En tal caso la segmentación de la imagen procede simultáneamente con una comprensión de la imagen. En segmentación un conocimiento a priori se refiere a obligaciones implícitas o explicitas sobre la probabilidad de un grupo de píxeles dados. Tales suposiciones frecuentemente surgen de restricciones colocadas sobre la imagen como una consecuencia de consideraciones dominio-dependiente. (**[11]**)

5.5 Evaluación de Métodos de Binarización

Los métodos orientados a píxel antes expuestos fueron evaluados, para ello se tuvo en cuenta el criterio que la segmentación es un proceso subjetivo al tener como propósito la separación de un objeto en particular de una imagen, por lo que es un proceso de interpretación visual.

Así también se toma en cuenta que los caracteres en la imagen binarizada no tengan conexiones entre ellos o con los bordes de la placa. Del mismo modo se considera que no se genere ruido o pequeñas zonas que puedan complicar la segmentación en las siguientes etapas.

Para la evaluación se hizo la prueba de los siguientes métodos aplicados a la base de datos:

- Método de Otsu
- Método Específico (Umbral Óptimo)
- Método de Selección Iterativa

Método de Reestructuración del Histograma para selección del Umbral usando:

- Método de Otsu
- Método Específico (Umbral Óptimo)
- Método de Selección Iterativa

Siguiendo esa línea se evaluó con el siguiente puntaje:

Tabla 5.1

Binarización	Excelente*	Buena	Regular	Mala
Puntaje	1.5	1	0.5	0

(*) Aunque se presentan pocos casos se decidió recompensar aquellos en que ningún método pudo ser capaz y sólo aquel lo consiguió y de buena forma.

Se utilizó una base de 147 fotografías para la evaluación obteniendo la siguiente calificación relativa:

Tabla 5.2

				Con Reestructuración del histograma		
	Óptimo	**Otsu**	**Iterativa**	**Óptimo**	**Otsu**	**Iterativa**
Puntaje	101.5	97.5	94.5	101.5	98	72.5

Se decidió utilizar el método umbral óptimo con reestructuración del histograma, dado que es parametrizable, con parámetro "**T**" utilizado como umbral de gradiente de la imagen, esto permitiría adecuar la elección del umbral en función de algunas características de la imagen como son el brillo, el contraste u otros.

A continuación mostramos un ejemplo para ilustrar el método.

- ***Ejemplo 5.5.1***

En la Figura 5.11 se puede apreciar una placa oscura con caracteres negros, a continuación notamos que su histograma presenta una distribución bimodal Fig. 5.12, si utilizamos cualquier método común basado en el histograma, como el de Otsu o de Selección Iterativa seleccionará el umbral en el valle que se encuentra entre los dos picos, (es decir un valor aprox. menor que 150 en la Figura 5.12) y efectivamente aplicando: OTSU => 134, UMBRAL_OPTIMO => 140. Todos estos umbrales generan una binarización como la de la Figura 5.13.

Figura 5.11 Imagen original en escala de grises de fondo amarillo

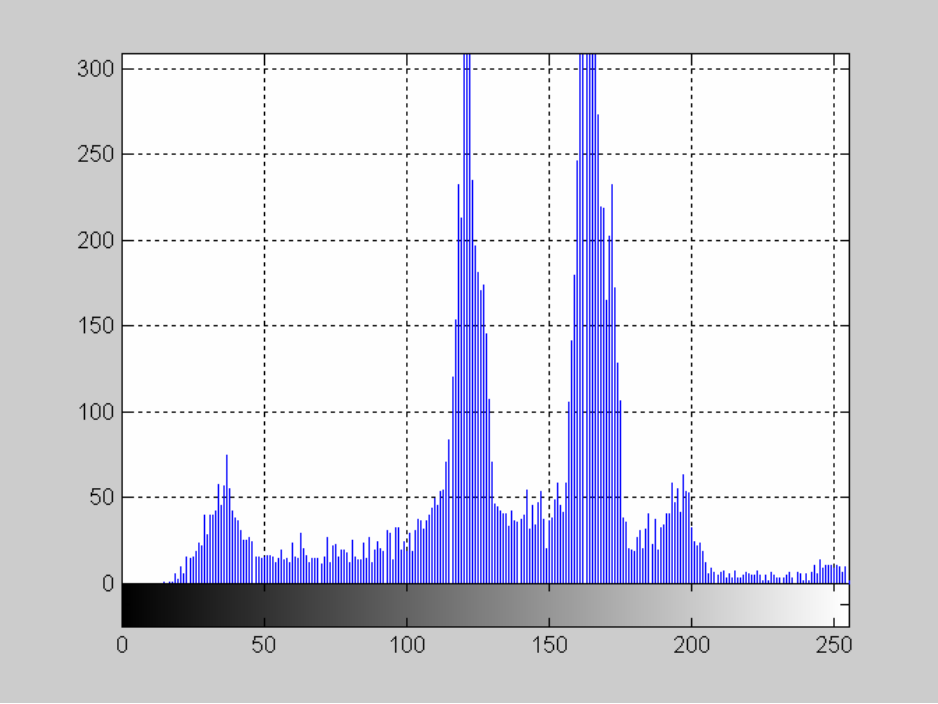


Figura 5.12 Histograma de la figura 5.11

Figura 5.13 Binarización usando cualquiera de los otros métodos (Optimo, Otsu, Iterativo)

Al hacer la reestructuración del histograma (Figura 5.14) se puede distinguir que notoriamente se ha "suavizado", los picos disminuyeron de tamaño, y esto porque el proceso consiste en tomar en cuenta sólo los píxeles de los bordes que son aquellos involucrados más con los caracteres y ya no tanto con el fondo, ya que como se ve en la figura 5.11 hay buena cantidad de píxeles oscuros en la placa y otros fuera de ella, lo que hace deformar el histograma.

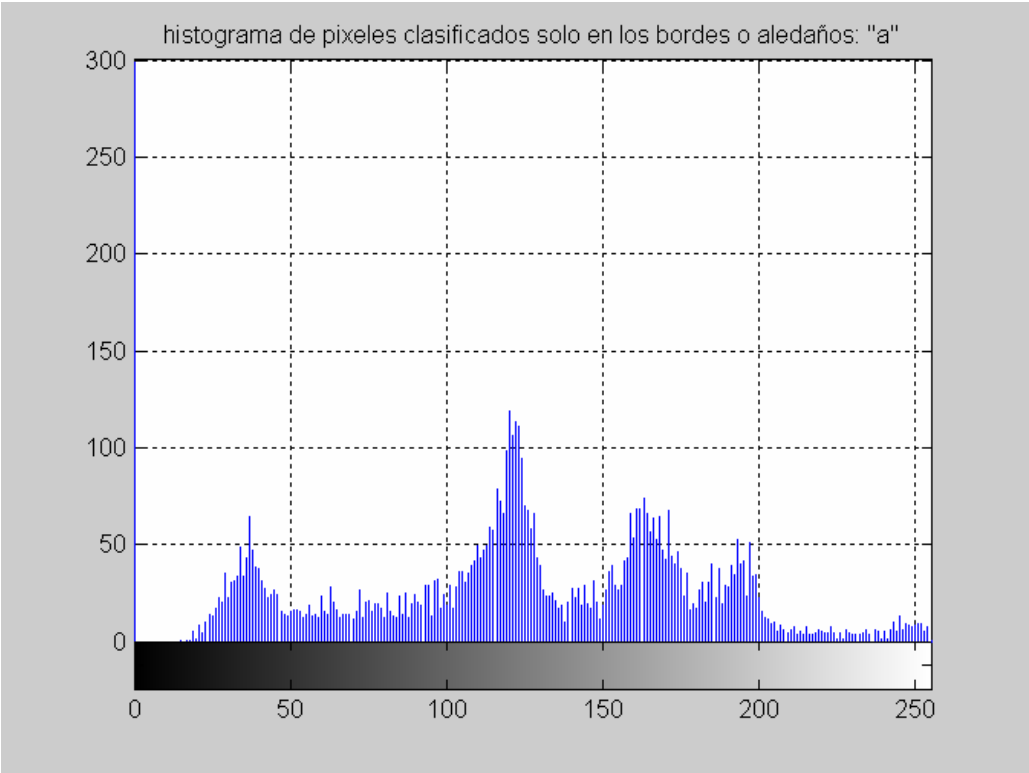


Figura 5.14 Histograma resultante, luego de su reestructuraci\on considerando sólo los pixeles de los bordes, el resto pasan a ser ceros.

La reestructuración toma en cuenta un umbral de gradiente "T" que en este caso se ha fijado para aquellos píxeles con T ≥ 7 (representa un gradiente normalizado) que funciona para la mayoría de los casos. Significa que los píxeles que tienen un gradiente normalizado mayor o igual a 7 pasan al análisis. Finalmente la aplicación del método del umbral óptimo nos da => 101, un valor que no se "esperaba" por la forma de su histograma y sin embargo obtenemos un resultado interesante, Figura 5.16.

En la figura 5.15 mostramos la imagen que resulta de la reestructuración del histograma, donde los píxeles de fondo se agrupan por igual (color negro), y sólo los píxeles de los bordes mantienen su nivel de gris, esta nueva distribución sirvió para hallar el umbral que se aplicó a la imagen original de la figura. 5.11.

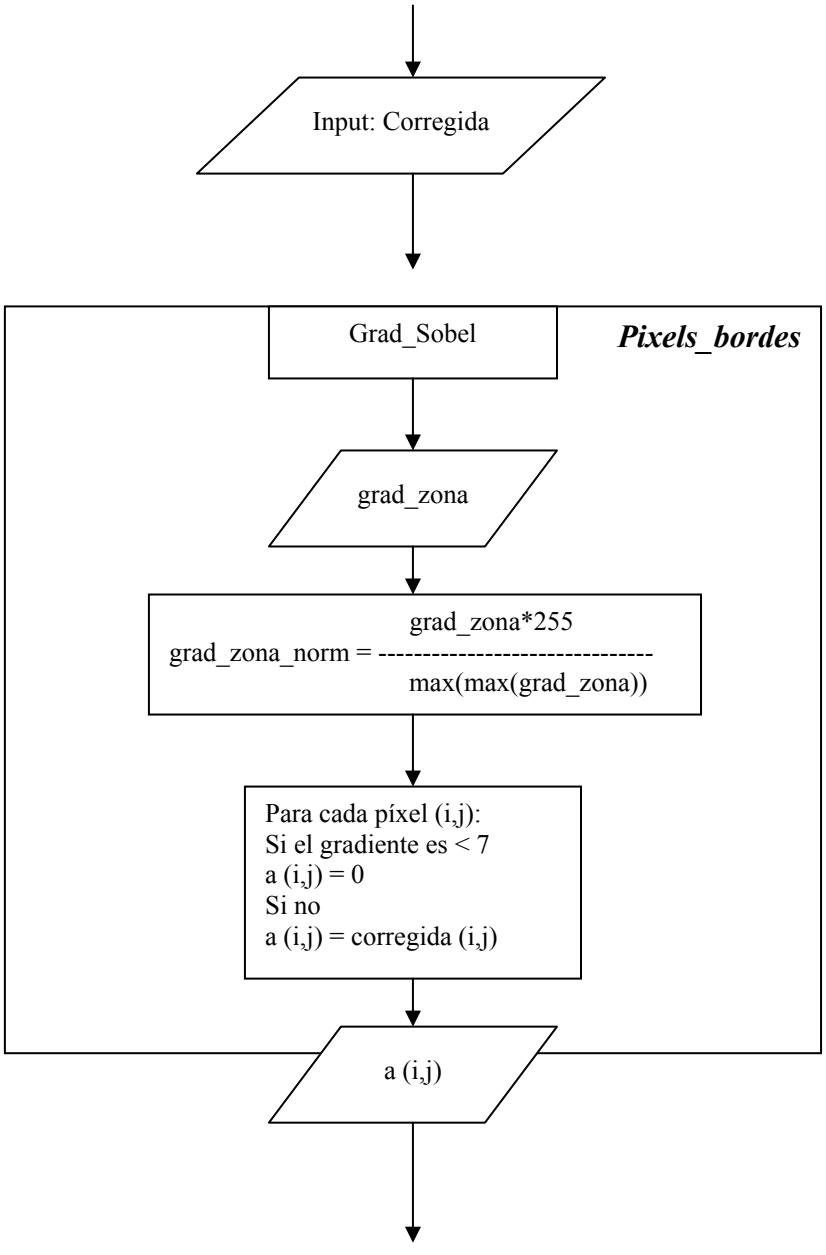
Figura 5.15 Imagen resultante luego de reestructurar el histograma. De aquí obtenemos el nuevo umbral

Figura 5.16 Imagen binarizada utilizando el umbral obtenido de la figura 5.15 sobre la imagen original fig. 5.11

Las líneas de código para esta reestructuración del histograma son como sigue.

```
function a=pixels_bordes(zona2)
grad_zona=Grad_Sobel(zona2);
grad_zona_norm=grad_zona*(255)/(max(max(grad_zona))); % gradiente normalizado
[fs,cs]=size(zona2);
for i=1:fs,
  for j=1:cs,
        if grad_zona_norm(i,j) < 7  % Parametro T = 7
        a(i,j)=0; %
        else
        a(i,j)= zona2(i,j);
    end
  end
end;
```

5.6 Estructura del algoritmo de Binarización



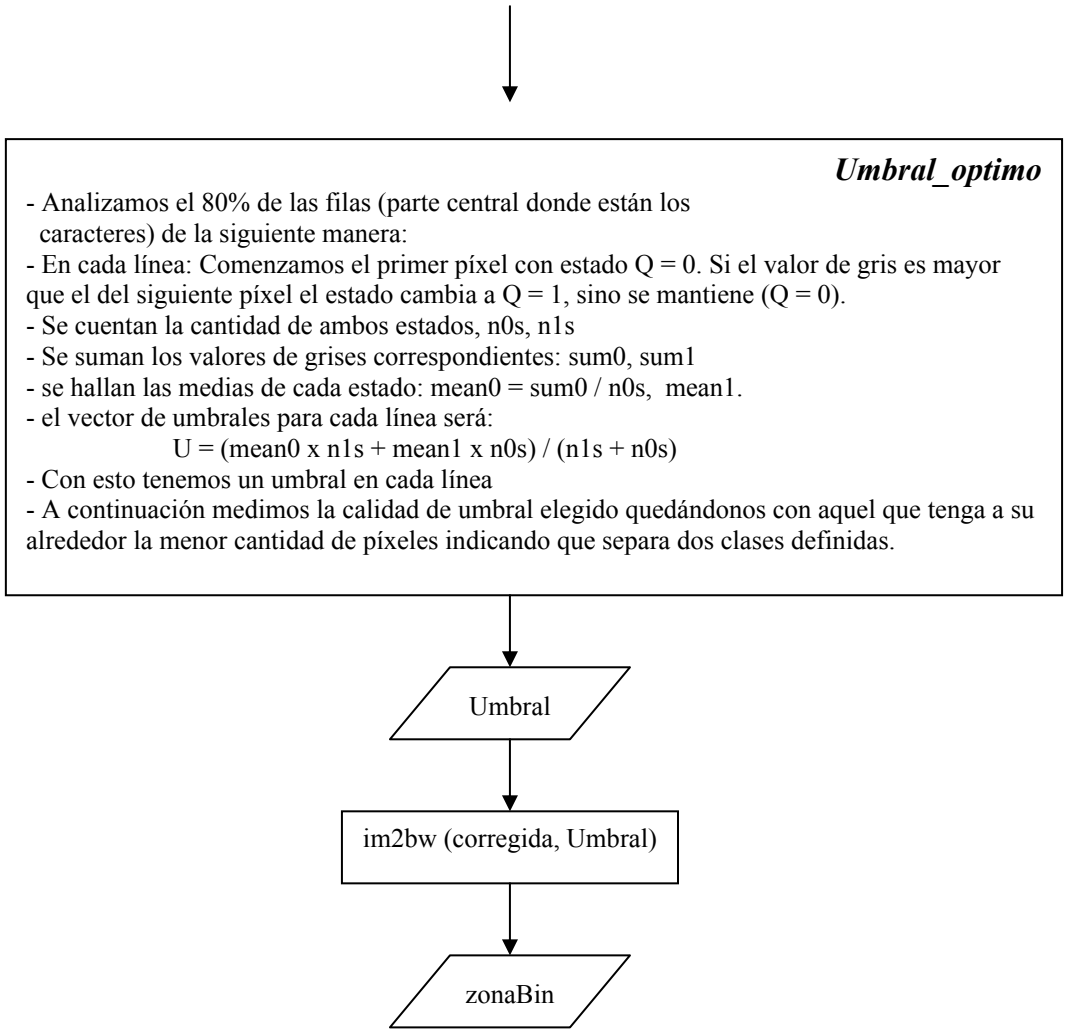


Figura 5.17 Proceso de Binarización

CAPÍTULO VI
SEGMENTACIÓN

6.1 Introducción

La segmentación consiste en extraer de un texto los diferentes caracteres. La extracción de caracteres es un proceso que pasa por aproximaciones, cada vez más cerca de los caracteres es decir a través de segmentaciones consecutivas hasta la obtención de los mismos en matrices, preparándose así para la siguiente fase de extracción de características. Este proceso puede complicarse ya que en la placa suelen aparecer otros elementos como suciedad, y aquellos que podemos considerarlos como ruido, para evitarlos aplicamos algunas reglas heurísticas de forma de discriminar un caracter de cualquier otra mancha o ruido. A esta parte se le denominó segmentación y sería propiamente de caracteres, explícitamente es la extracción de los mismos en matrices separadas. A continuación exponemos la segmentación de los caracteres utilizando la técnica de las proyecciones, aunque es un poco limitada nos proporcionó resultados aceptables para este sistema prototipo.

6.2 Extracción de Caracteres a Través de Proyecciones

La segmentación de los caracteres pasa por la separación de ellos de manera individual, para esto será necesario primero hacer una primera aproximación hacia ellos debido a que normalmente están rodeados de algunos pequeños detalles así como del marco de la placa. Esto se consigue a través de las proyecciones horizontales y verticales [4].

Suponemos una imagen binarizada (sólo dos niveles: el blanco y el negro) y con la inclinación corregida. A cada píxel le asociaremos un número que será cero para los píxeles blancos y uno para los píxeles negros. Después de esto, representaremos la imagen como una matriz de ceros y unos. En ese momento podemos calcular la suma de sus columnas (proyección horizontal) y la suma de sus filas (proyección vertical).

Figura 6.1. Imagen de archivo: IMG_0085.jpg (Binarizada)

De la imagen de ejemplo se hace claro como funciona el método. A partir de los mínimos de la proyección horizontal (Fig. 6.2) se pueden detectar las columnas que separan los caracteres de otros elementos. Para esto imponemos un nivel dado por:

$$nivel\_h = \min(Ph) + 0.25*(\max(Ph) - \min(Ph)); \qquad \text{Ec. 6.1}$$

Donde **Ph**, es la proyección Horizontal de la imagen binaria.

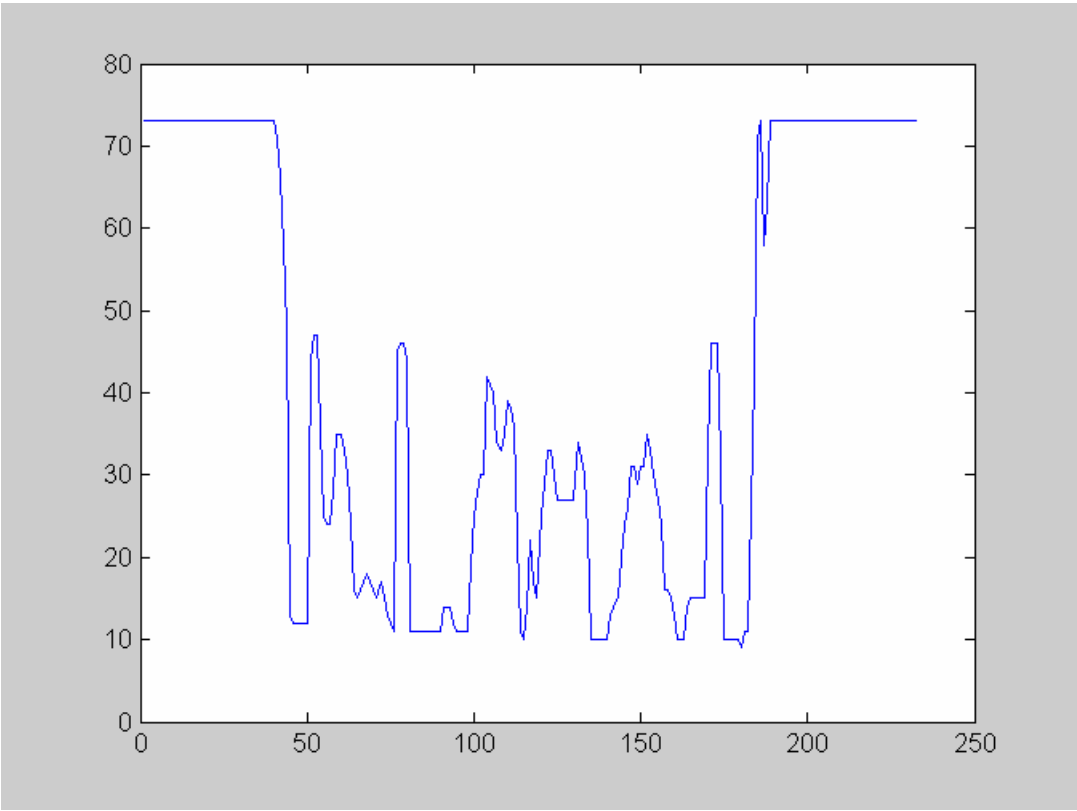


Figura 6.2. Proyección Horizontal de la imagen en blanco y negro (**Ph**), Fig. 6.1

Con este nivel hacemos la diferencia a toda la proyección y luego nos quedamos con el primer negativo o cero de la izquierda y el último de la derecha.

% La Matriz J contiene el negativo de la imagen

```
nPh = Ph - nivel_h;   % Nuevo nivel para la proyeccion Horizontal
[Cc] = find(nPh<=0);
plkN = ~J(1:fs,min(Cc):max(Cc)); figure, imshow(~plkN) % fs es el número de filas
```

Figura 6.3.- Recorte a partir de la proyección horizontal

De manera similar ahora, una vez recortada la imagen, hacemos la proyección vertical **Pv**:

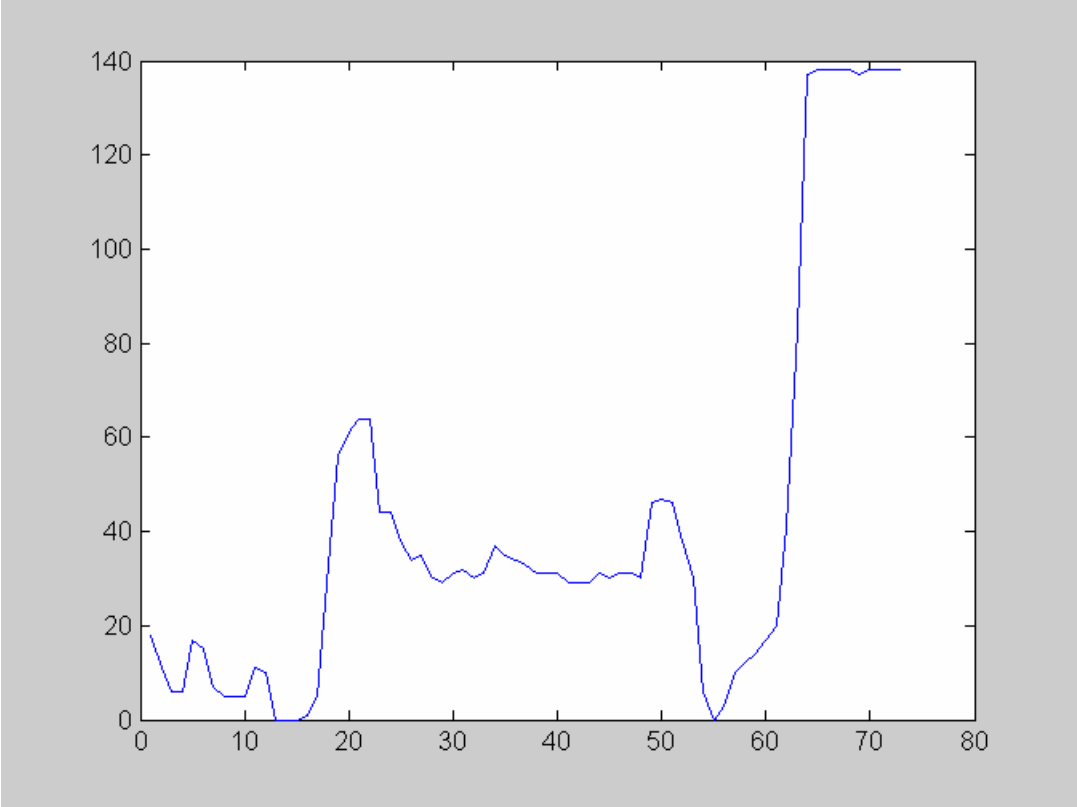


Figura 6.4. Proyección Vertical de la figura 6.3. El bloque de la izquierda proviene de las letras PE, el bloque central viene de los caracteres de la placa y la parte derecha es la parte inferior de la placa.

```
% El valor porcentual 0.2, se determina experimentalmente
nivel_v = min(Pv)+0.20*(max(Pv)-min(Pv));
nPv = Pv - nivel_v;   %Nuevo nivel para la proyección Vertical
[Fc] = find(nPv<=0);
```

```
Lim_C=J(min(Fc):max(Fc),min(Cc):max(Cc));
figure, imshow(Lim_C)
```

Figura 6.5. Recorte de la figura 6.3 a partir de la proyección vertical

A partir de aquí debemos hacer una segmentación más próxima a los caracteres, para ello tomamos la proyección vertical nuevamente pero ahora nos aseguramos con la 1era derivada (**Fig. 6.7**) que nos da valores más altos a distinguir.

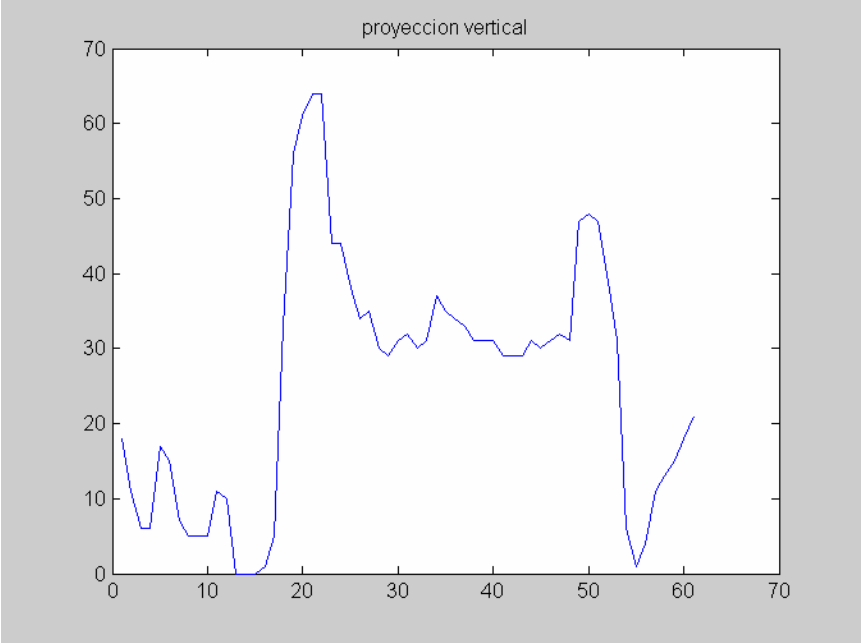


Figura 6.6. Proyección Vertical de la figura 6.5

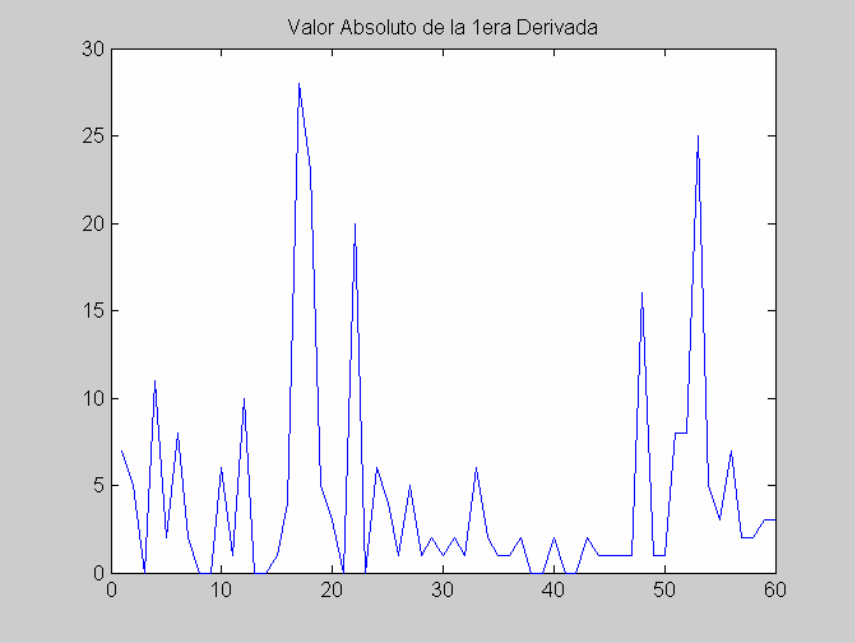


Figura 6.7. La derivada de la figura 6.6 nos da pendientes altas en las transiciones

Ahora se pueden tomar el primer y el último valor más altos, estos se dan según la Figura 6.7 en las filas 17 y 53 lo que vendría a ser el nuevo intervalo para la altura y la figura queda:

Figura 6.8 Nuevo recorte de la figura 6.5 luego de usar
la proyección vertical y sus derivadas

Ahora nos queda hacer la extracción de los caracteres lo cual es una tarea más exhaustiva.

Para esto limpiaremos primero algunos restos de ruido que pudieron haber quedado de etapas previas, utilizando una erosión con una máscara de unos de tamaño 2x2, Fig. 6.9. Luego nos valemos de la proyección horizontal nuevamente, Fig. 6.10.

Figura 6.9. Erosión de la figura 6.8

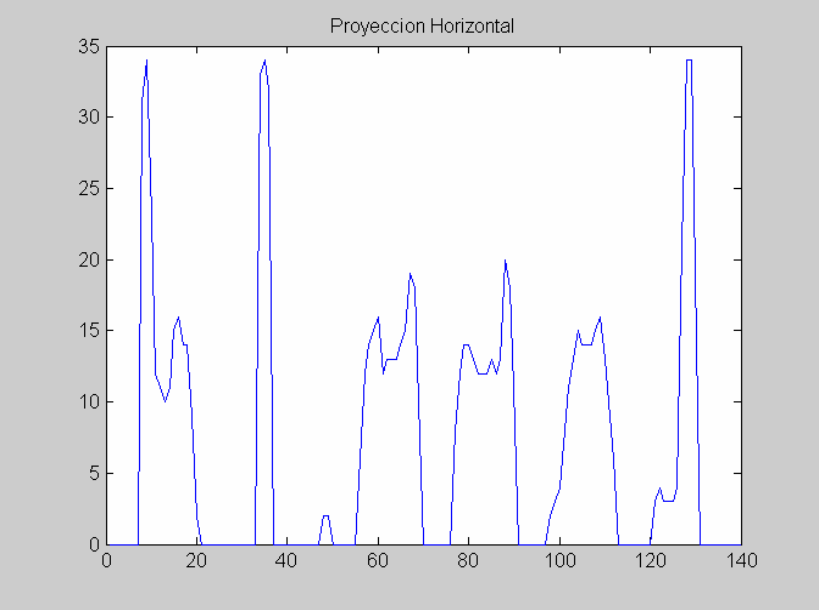


Figura 6.10 Proyección horizontal de la figura 6.9 para hallar las posiciones de los caracteres

El algoritmo consiste en:

1- Ubicar en un vector **A** todas las posiciones que contienen ceros.

2- Hallar las diferencias consecutivas y guardarlas en otro vector **B**.

3- Usando el comando ***find*** ubicar las posiciones del vector **B** en los que la diferencia sea mayor que 3 (viene a ser el mínimo ancho en píxeles de un caracter) y guardarlos en otro vector **C**.

4- Con esto, ahora comenzaremos a guardar en un arreglo de matrices todos los caracteres haciendo corresponder de la siguiente forma:

CharsNeg(**:**,1**:**superior-inferior+1,i) = ~Lim_Hn(**:**, inferior **:**superior)

Donde:

Inferior = **A**(**C** (i)) % *Contiene la posición del inicio del carácter i-ésimo*

Superior = **A**(**C**(i)) + **B**(**C**(i)) % *A lo anterior le sumamos el ancho del caracter correspondiente*

Lim_Hn: es la matriz de la imagen

Una vez ubicados los caracteres se obtiene un arreglo de matrices de dimensiones (37,17,6)

Es decir 6 matrices de 37x17 (se obtuvieron 6 caracteres como efectivamente se ve en la figura 6.9) y estos son:

Figura 6.11 Caracteres extraídos almacenados en matrices

Ahora estas matrices están listas para pasar a la siguiente etapa de clasificación para su posterior reconocimiento.

6.3 Mejoras en la segmentación

El proceso de segmentación de caracteres básicamente consistía en las aproximaciones a través de la proyección horizontal y vertical, dándole algunas condiciones para descartar posibles marcos. Lo que sucede es que existen placas con marcos más gruesos o como en la figura **6.12.b** la recuperación de la imagen en un plano frontal implicó mantener el marco completo a su alrededor. Esto se puede notar con mayor énfasis en la binarización. (figura **6.12.c**).

(a) Placa original: IMG_0025.jpg

(b) Placa proyectada en un plano frontal

(c) Placa Binarizada

Figura 6.12 Recuperación de placa y su binarización correspondiente

Lo que se hizo inicialmente fue establecer un código para que cuando existan píxeles negros (1's) en los extremos o bordes de la matriz, se incrementen las filas

correspondientes con píxeles blancos (0's). Esto conlleva a que mientras antes se obtenía una proyección vertical como la figura 6.13 con valores altos en los extremos.

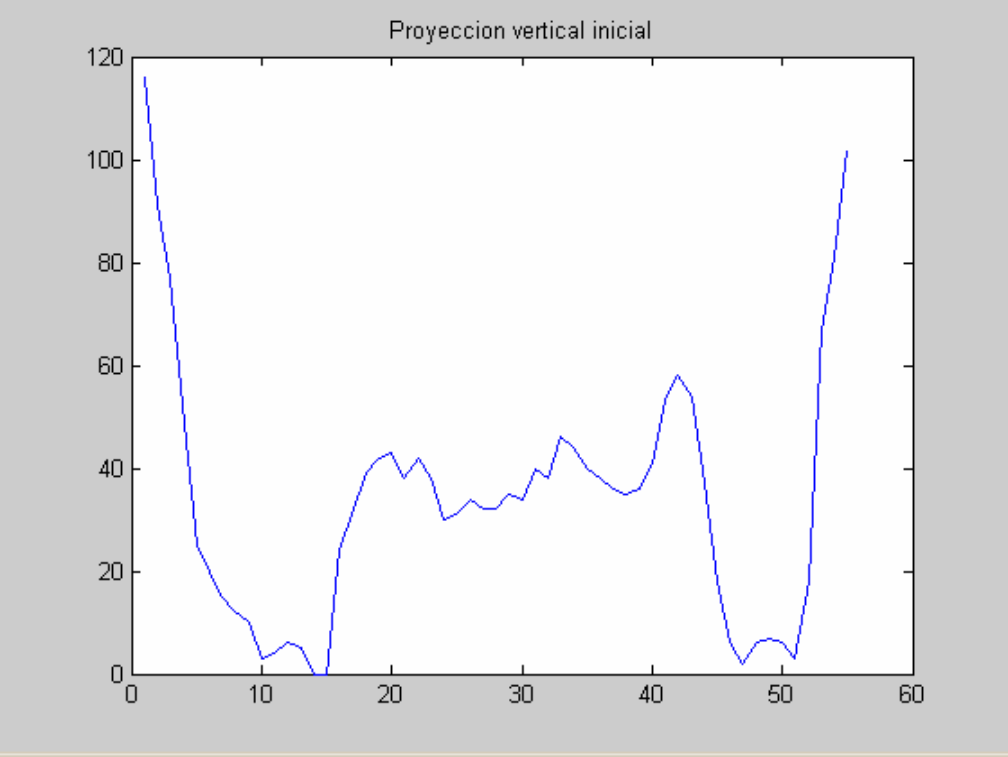


Figura 6.13 Proyección Vertical antes de los cambios.

Al completar filas los extremos "bajan" hasta cruzar la línea del cero (Figura 6.14), esto proporciona cruces por cero que nos ayudan a trabajar mejor en la búsqueda de la altura correcta del carácter. El resultado se aprecia en la figura 6.15.

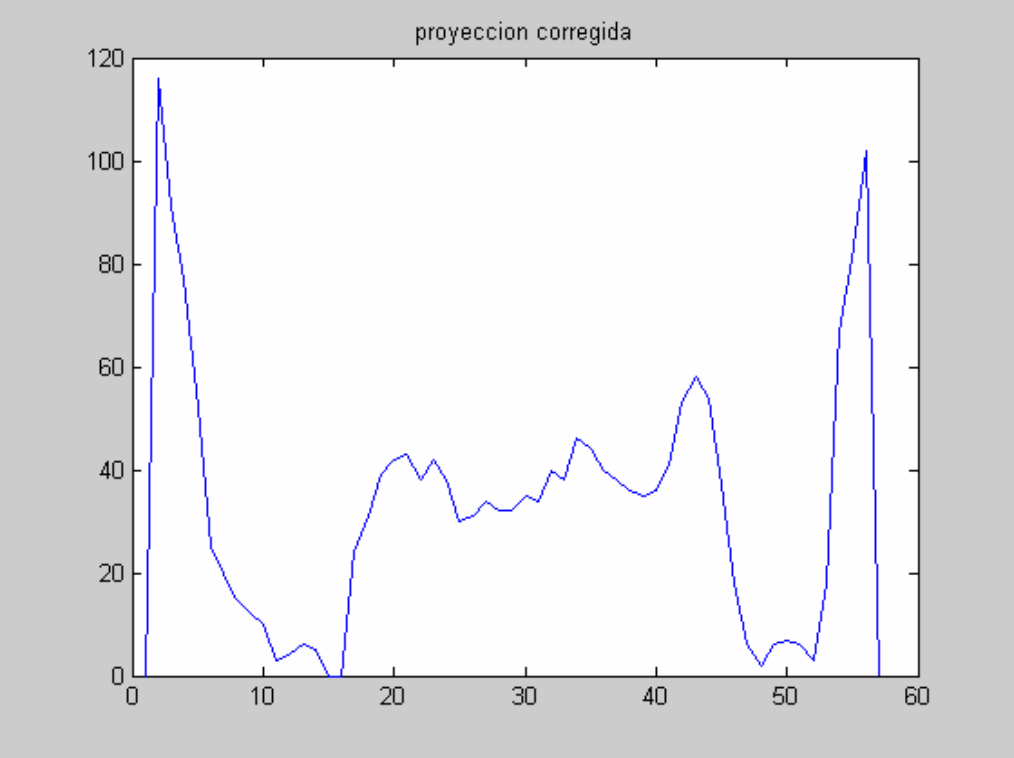


Figura 6.14 Proyección Vertical de Imagen completada con "Filas Blancas" en los extremos.

Figura 6.15 Aproximación a los caracteres con método mejorado.

Un ejemplo donde se puede apreciar mejor la utilidad del método puede ser en la siguiente figura y su Binarización.

Figura 6.16 Recorte de placa

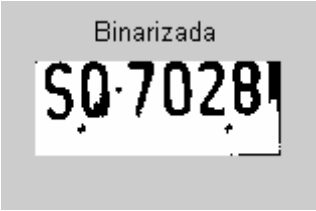


Figura 6.17 Luego de la binarización la imagen mantiene zonas irregulares

Como se ve en su proyección vertical (Figura 6.18) los extremos no tienen cruces con el eje horizontal, principalmente en este caso el extremo izquierdo, lo que hace que no se pueda determinar los límites de la altura del carácter.

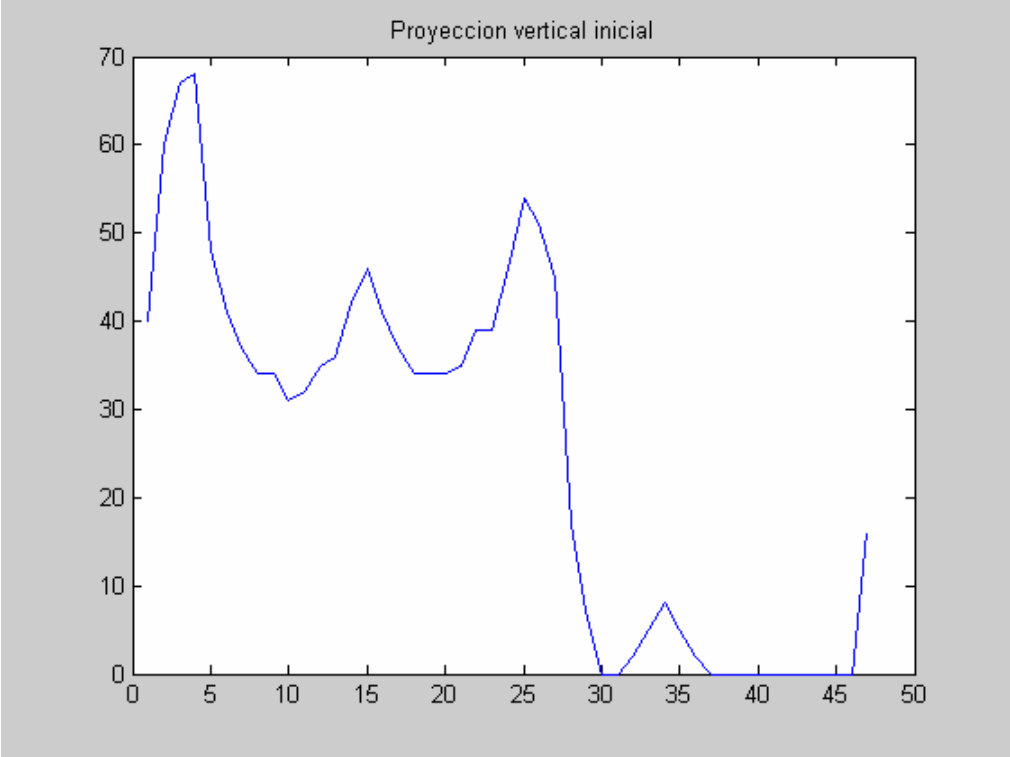


Figura 6.18 Proyección vertical de la figura 6.17. Se aprecia que dado que los caracteres están próximos al borde superior, en esta condición, su proyección vertical no permite definir claramente los limites de los caracteres.

Sin embargo aplicando el método se obtiene la figura 6.19, aquí si se pueden determinar fácilmente los límites con esto finalmente se obtuvo un recorte más preciso como se puede ver en la figura 6.20.

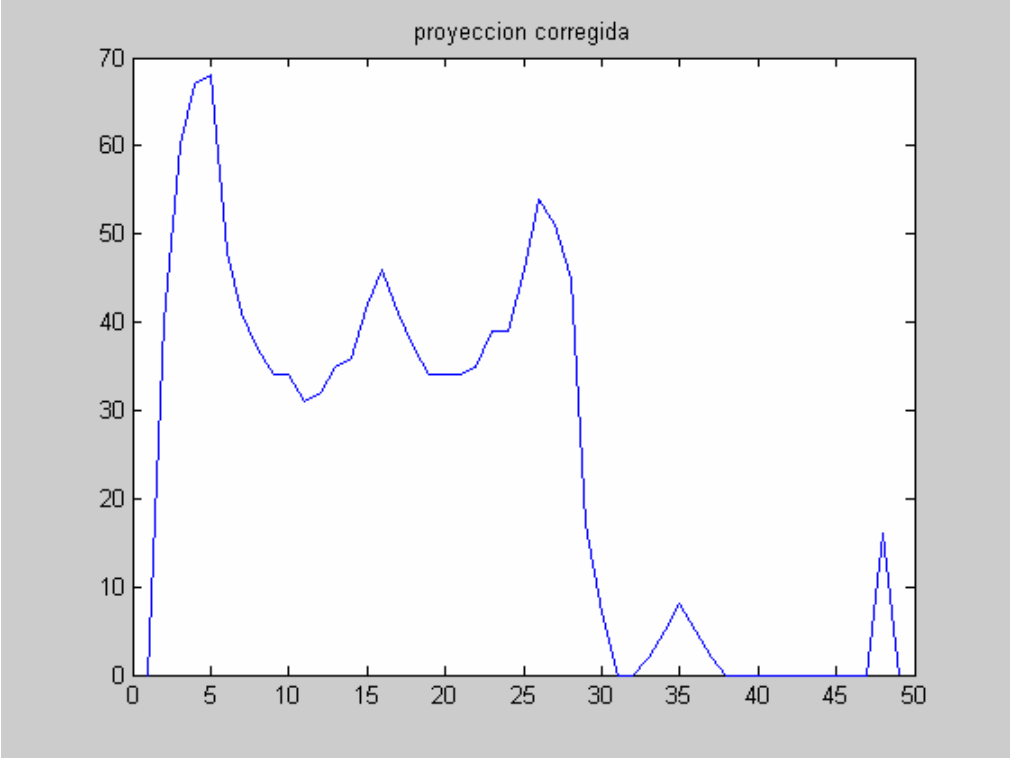


Figura 6.19 La nueva proyección vertical nos permite delimitar los caracteres.

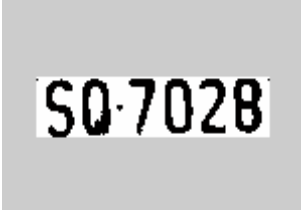
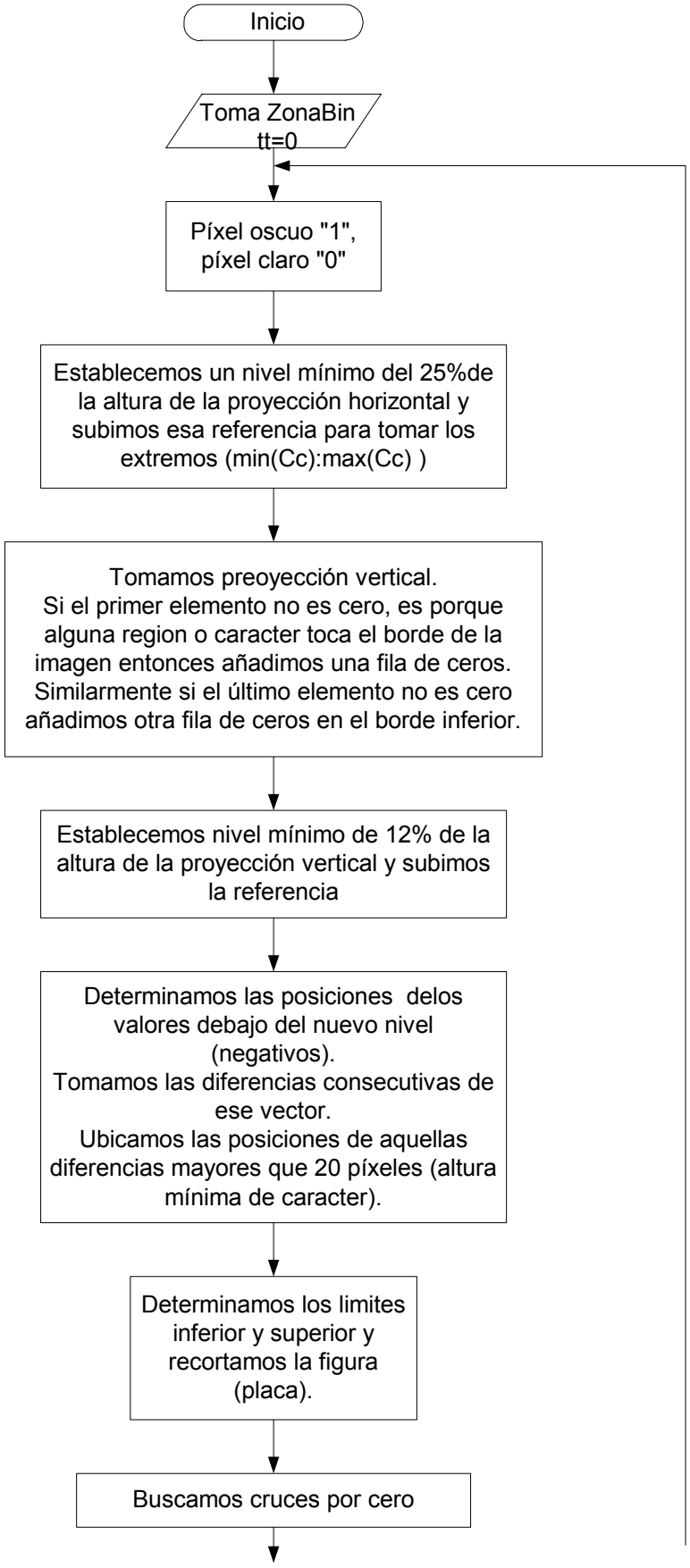
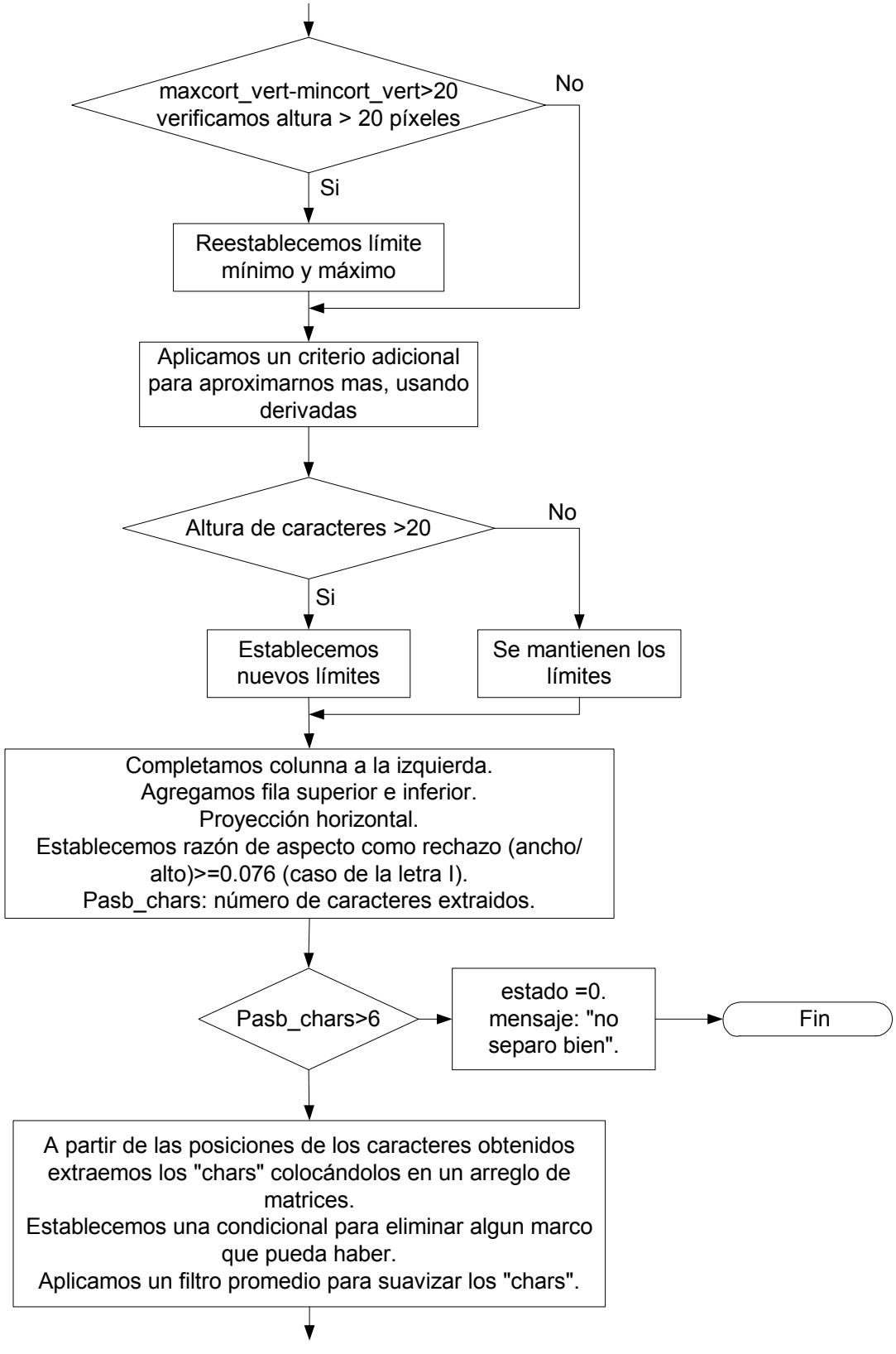


Figura 6.20 Nueva aproximación a los caracteres

Por otro lado se aplicó un lazo realimentado luego de la extracción de los caracteres, sólo para el caso en que se encuentren 5 caracteres, la función **Extraccion.m** puede llamar a **Segmentacion.m** solo una vez más para intentar una segunda vez luego de agregar columnas laterales convenientemente. Esto se puede apreciar en el diagrama de flujo de todo el algoritmo, que exponemos a continuación.

6.4 Estructura del algoritmo de segmentación




maxcort_vert-mincort_vert>20
verificamos altura > 20 píxeles
No
Si
Reestablecemos límite mínimo y máximo
Aplicamos un criterio adicional para aproximarnos mas, usando derivadas
Altura de caracteres >20
No
Si
Establecemos nuevos límites
Se mantienen los límites
Completamos colunna a la izquierda.
Agregamos fila superior e inferior.
Proyección horizontal.
Establecemos razón de aspecto como rechazo (ancho/ alto)>=0.076 (caso de la letra I).
Pasb_chars: número de caracteres extraidos.
Pasb_chars>6
estado =0.
mensaje: "no separo bien".
Fin
A partir de las posiciones de los caracteres obtenidos extraemos los "chars" colocándolos en un arreglo de matrices.
Establecemos una condicional para eliminar algun marco que pueda haber.
Aplicamos un filtro promedio para suavizar los "chars".

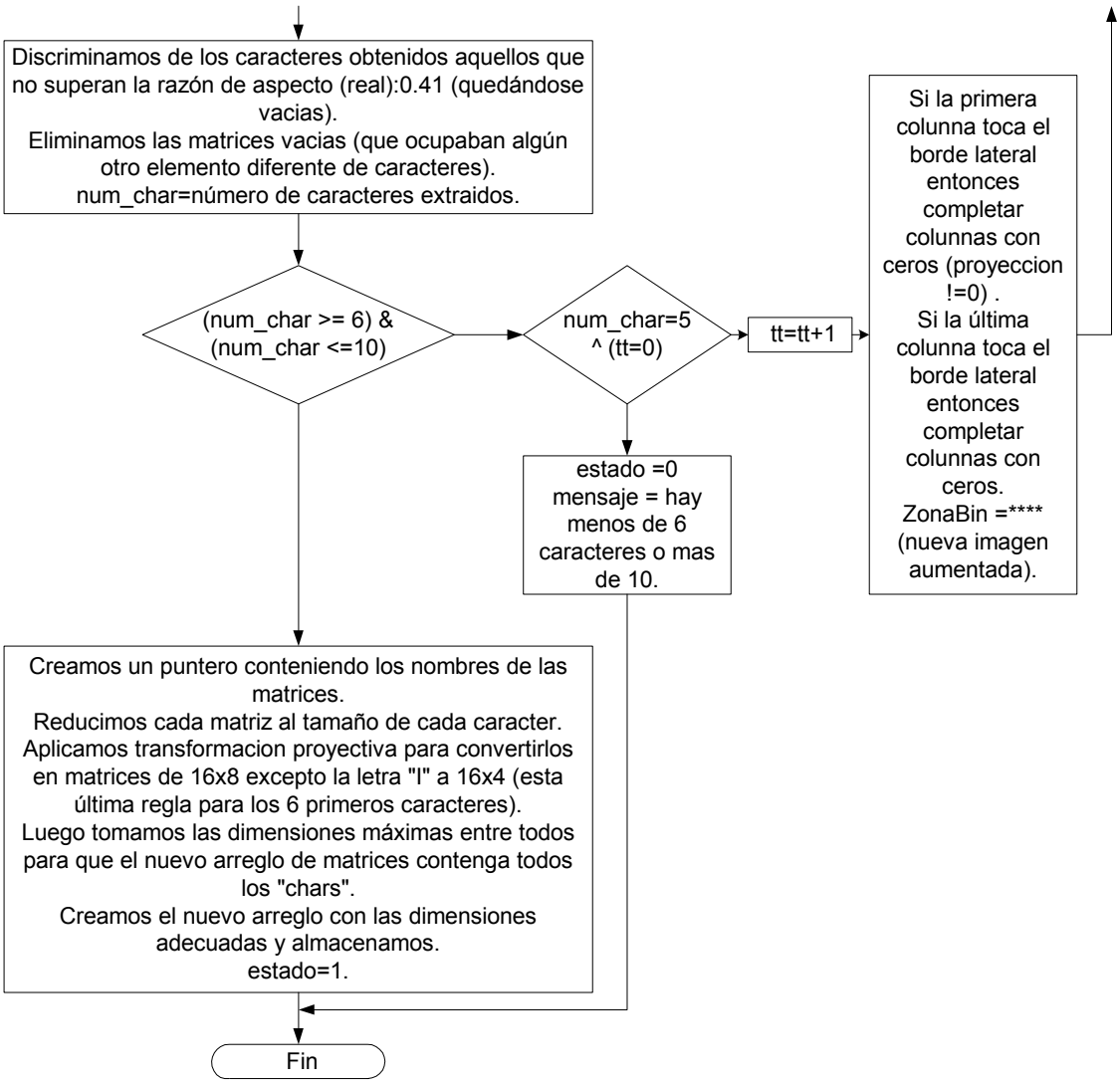


Figura 6.21 Proceso de Segmentación

6.5 Proceso de Decisión para la aplicación de todos los algoritmos

Para aquellos casos en los que el sistema detecta dos zonas luego de la localización, como potenciales placas, es necesario establecer una regla de decisión que sea óptima de tal manera que aplique los algoritmos para el reconocimiento a una zona y en caso no encuentre caracteres, en alguna etapa, inmediatamente cambie por aplicarlos a la otra zona. Esto se puede apreciar de manera gráfica mediante el diagrama de flujo del algoritmo, que mostramos en la figura 6.22. En la gráfica se muestra un diagrama desde el punto de vista de la programación. Al terminar la localización se utiliza un contador de zonas obtenidas, por lo que si es "1" se procesa la única, en caso contrario existen 2 y se procesará la de mayor área, ya que la zona más grande es más probable que contenga la placa. El algoritmo de localización determina dos zonas como posibles placas estas son: "Zona1" que tiene "Area1" y la "Zona2" que tiene "Area2".

```
[zona1,zona2,area1,area2,CONTADOR]=Localizacion(ii);
                    % Esta función arroja las posibles zonas con placas  y sus áreas.
```

Luego pasa a un bloque denominado: "Procesa", éste engloba todas las etapas siguientes: corrección de perspectiva, binarización, segmentación, y reconocimiento de caracteres.

Al hacer la corrección de la perspectiva utilizamos la transformada de Hough para diferenciar los lados de la placa esto permite discernir entre zonas que sean placas y las que no sean. El criterio entonces consiste en establecer una condicional después de la corrección de perspectiva de manera que luego ésta verifique que el área obtenida de la nueva zona sea lo suficientemente grande como para contener una placa, de lo contrario sale del lazo y continua analizando la otra zona. En las líneas de código siguientes vemos ese análisis dentro del bloque "Procesa".

```
corregida=corrige_perspectiva2(zona);          % Función de corrección de perspectiva.
dimen=size(corregida);              % Luego a partir de sus dimensiones medimos su área.
 if (dimen(1,1)*dimen(1,2))<(0.01)*area_aux,
    estado=0;
 else
    estado=1;
 end;
```

Luego que procesa una de las zonas en caso no segmente bien el indicador de estado se pone a cero, se descarta esta zona y se procede con la otra. Si no segmentó bien puede deberse a que no consiguió extraer caracteres de esa zona, que por lo general son zonas de vegetación u otros caracteres que no corresponden a una placa vehicular, salvo las excepciones en que ya sea demasiado complicado extraer los caracteres como en placas que pueden tener caracteres despintados, otros marcos que los unen, etc. Al término del proceso se obtienen los caracteres, se arroja un indicador de estado con valor uno.

Este esquema se diseñó de esta manera para poder descartar a tiempo sin tener que llegar hasta el reconocimiento para hacer el descarte, evitando así operaciones innecesarias. Adicionalmente dentro del proceso de reconocimiento se hace un descarte de caracteres analizando la razón de aspecto. El diagrama de flujo se muestra en la figuras 6.22 y 6.23.

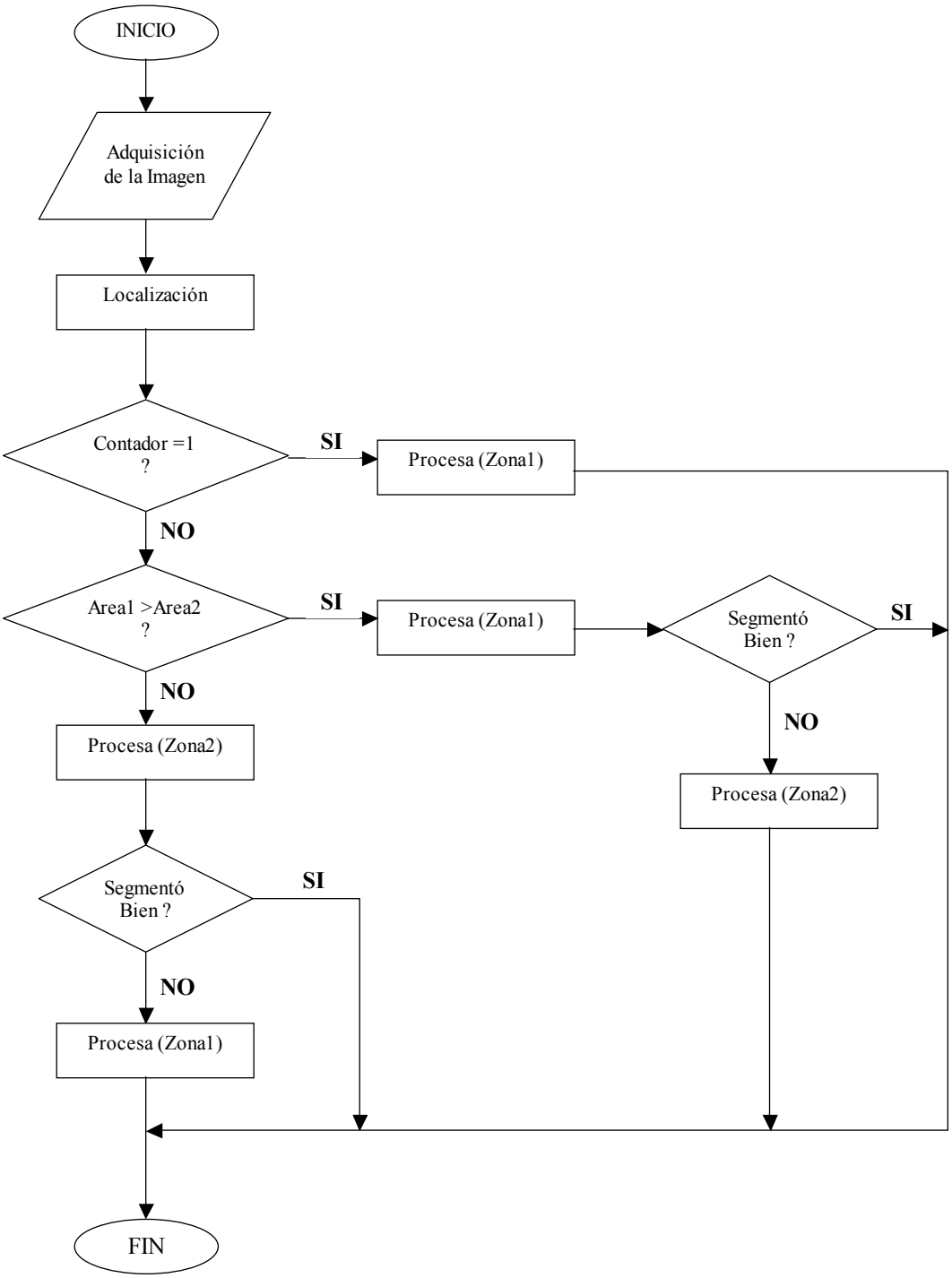


Figura 6.22 Proceso de desición para la aplicación de los algoritmos. Como se puede observar los criterios han sido, ¿Cuantas placas se localizaron?, ¿Cuál de ellas tienen el tamaño adecuado?, ¿Se segmentó bien?. Evaluando estos aspectos descartamos a tiempo algun problema ahorrando tiempo de procesamiento.

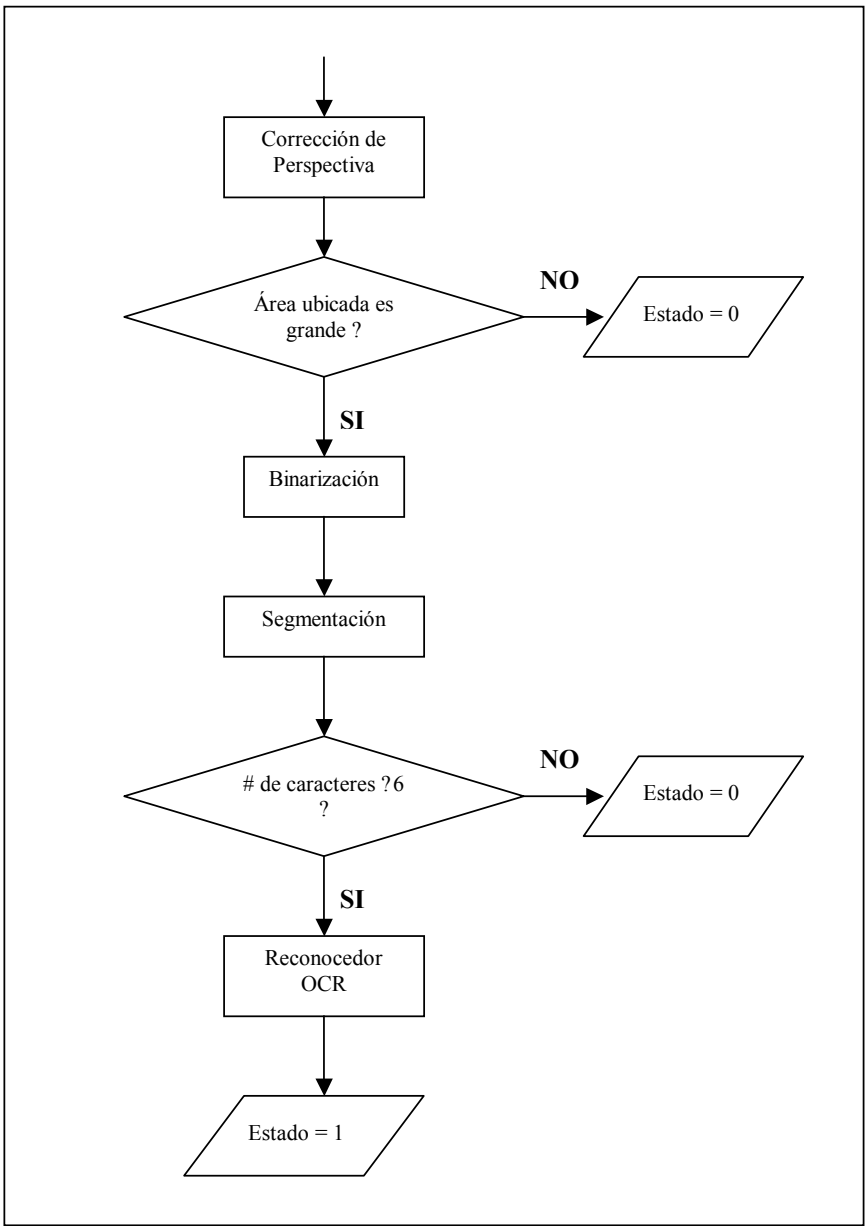


Figura 6.23 Bloque: Procesa(Zona, area).

Este bloque utiliza dos condicionales la primera sobre el tamaño del área, la segunda en la cantidad de caracteres extraídos. En cualquiera de los casos luego de tener un estado 0 ó 1, es suficiente para que se termine el proceso.

CAPÍTULO VII
EXTRACCIÓN DE PATRONES

7.1 Introducción

Para extraer los patrones de los caracteres, es necesario un paso previo, que viene a ser el de la normalización, este consiste en reducir de tamaño las matrices de caracteres extraídos en la etapa anterior, (en promedio de 37 x 17 a matrices cuadradas de 16 x 16). Es importante reducirlas evitando la distorsión de los caracteres, ya que esto podría inducir a error en la etapa de reconocimiento. Una vez reducidas estas matrices se procede a extraer las características en vectores de menor longitud, es decir que cada vector, con pocos coeficientes es capaz de representar la imagen de un caracter. Estos nuevos vectores servirán en la siguiente etapa para entrenar nuestro clasificador y finalmente obtener el reconocedor de patrones.

7.2 Normalización

Para la extracción de características es necesario previamente reducir las dimensiones de los caracteres obtenidos en la etapa previa, ver figura. 7.2, para esto utilizaremos la transformación proyectiva **[10]**.

Figura 7.1 Placa original: IMG_0057.jpg

Figura 7.2 Resultado de la extracción de caracteres

Como se había contemplado en estudios anteriores, el uso de la transformación proyectiva en este caso se convierte en una aplicación directa y sencilla dado que sería sólo la proyección de la matriz del caracter en otro plano "alejado" con un tamaño reducido, además debe conservar sus características como la relación de altura y ancho del caracter (de 2 a 1) denominada relación de aspecto, por ello inicialmente la transformación será a una matriz de dimensiones 16 x 8. Se implementó entonces una función utilizando el algoritmo referenciado en el capítulo 2 de **[10]**, que realiza la proyección (reducción) de una matriz imagen, (en este caso de 37x17) ingresando como parámetros las dimensiones iniciales y las finales que se deseen, en este caso serían a 16 x 8 (Figura 7.3).

Figura 7.3 Matrices de 16 x 8 reducidas por Transformación Proyectiva

La normalización consiste en llevar estas matrices de 16 x 8 a matrices de 16 x 16, con los caracteres centrados. También se implementó un algoritmo para realizar esta tarea (Normalización). Básicamente consiste en determinar el ancho del caracter y completar con columnas "blancas" (1's) a los costados de manera simétrica. También debemos tener en cuenta que no siempre se van a obtener matrices exactas de 16 x 8, luego de la transformación, en algunos casos pueden ser de 15 x 7, debido a que según la imagen original la relación de aspecto exactamente no es de 2 a 1 y pueden variar en decimales. De todas maneras la Normalización corrige estos pequeños inconvenientes, el resultado podemos apreciarlo en la figura 7.4.

Figura 7.4 Caracteres normalizados a matrices de 16 x 16

7.2.1 Mejoras Adicionales en la Proyección de Caracteres sin Distorsión

Una aspecto importante antes de la extracción de características de los caracteres es la reducción de los mismos o proyección de ellos en matrices mas pequeñas (16 x 8 y luego normalizados en 16 x 16). Una vez extraídos los caracteres desde la primera parte de la segmentación (Figura 7.5), estos eran tratados en un arreglo de matrices, todas del mismo tamaño ya que el Matlab solo concibe arreglos de este tipo con matrices de las mismas dimensiones. Hasta cierto punto proporcionaba cierta facilidad en el manejo del arreglo ya que solo se manejaban sub-índices para referirse a una de las matrices. El problema fue que estas matrices no siempre mantenían una proporción adecuada (2:1) como se esperaba para poder transformarla a 16 x 8 (con relación 2:1), podían resultar de 39 x 17 por ejemplo lo que no guardaba la proporción, como se puede ver en la figura 7.6, al proyectarlas en matrices más pequeñas los caracteres podían resultar más estrechos de lo normal, esto en algunos casos podría resultar perjudicial, sobre todo cuando los caracteres provienen de una reconstrucción con transformación proyectiva de una placa un poco alejada, al reconstruirla los caracteres pueden quedar un poco mas delgados de lo normal y su proyección aun más, perdiendo resolución e información.

Figura 7.5 Caracteres extraídos en la etapa de segmentación

Figura 7.6 La reducción directa de las matrices de los caracteres, en matrices de 16 x 8, podia en algunos casos, deformar los caracteres.

Para superar este inconveniente primero se decidió guardar las matrices de forma separada ya que estas tenían diferentes dimensiones, no necesariamente iguales debido a la forma de cada caracter, y se utilizó para ello una matriz auxiliar que contenía los nombres de las variables de los caracteres:

index = ['A01';'A02';'A03';'A04';'A05';'A06';'A07';'A08';'A09';'A10'];

Donde cada cadena contiene un caracter, utilizamos pues esta matriz como puntero. Posteriormente se definen uno a uno cada caracter y ya se les puede llamar a través de "index". Luego estas matrices son ajustadas a su entorno, es decir sólo el tamaño que ocupan, no más, así entonces mantienen su proporción, para luego proyectarlas y normalizarlas. El resultado se puede ver en la figura 7.8, que a diferencia del anterior los caracteres conservan mejor su forma sin distorsionarse mucho.

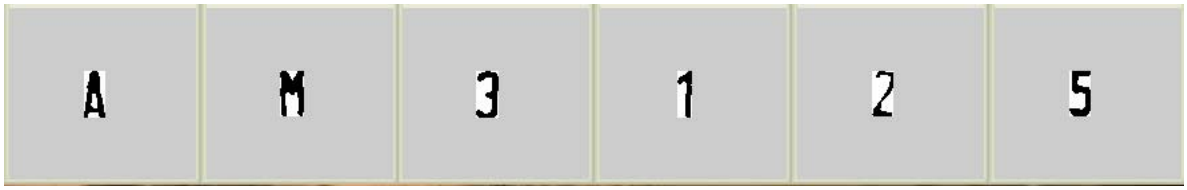


Figura 7.7 Tomamos la matriz justa al tamaño y forma del carácter.

Figura 7.8 Luego de hacer la proyección en matrices más pequeñas, se observa que mantienen una mejor forma en sus proporciones, a diferencia de la Fig. 7.6.

Así podemos ver otro ejemplo particular como el de la figura 7.9 con la letra "I".

Figura 7.9 Caracteres originales reducidos a su entorno

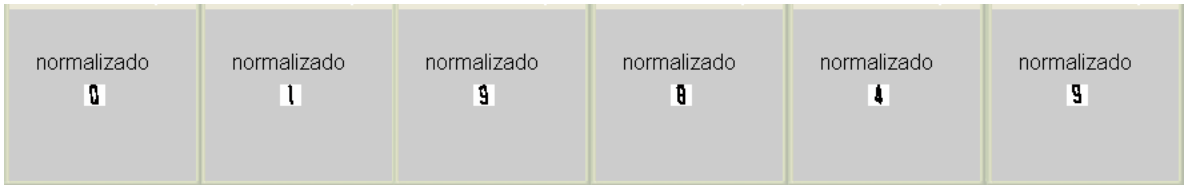
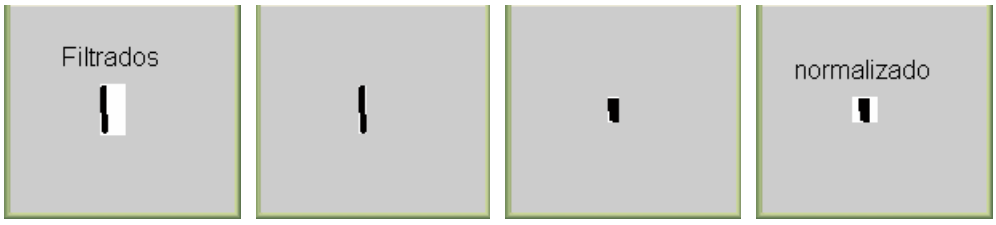


Figura 7.10 Caracteres obtenidos con la técnica anterior en arreglo de matrices

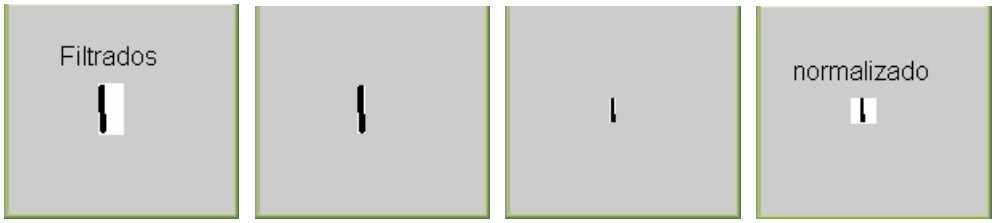
Figura 7.11 Caracteres obtenidos con la nueva técnica manteniendo las proporciones

Se tuvo que añadir código adicional para evitar que aquellos caracteres que no tengan la proporción de 2 a 1, como la letra I, se transformen a matrices de proporción 4 a 1 (16 x 4). Ya que si se mantenía la regla para todos de 2 a 1 éste carácter se deformaba demasiado, como se puede apreciar en la siguiente figura.



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

Figura 7.12 Proceso que se aplicaba en la proyección de caracteres: (a) Caracter extraído en un arreglo de matrices. (b) Reducido a su entorno, no mantenía proporción 2 a 1. (c) Carácter proyectado en matriz 16 x 8. (d) Caracter normalizado a 16 x 16.



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

Figura 7.13 Nuevo proceso que se aplica a caracteres para mantener proporciones, la secuencia es similar a diferencia de la parte (c) donde se mantuvo la relación de 4 a 1 de manera que la normalización muestra un mejor caracter.

Definitivamente se optó por mejorar la forma de los caracteres y evitar la distorsión, ya que esto induce error. Si queremos mejorar la calidad del reconocimiento, será pues necesario disminuir la preponderancia de aquellos factores que influyan negativamente. En este caso se optó sacrificar un poco el tiempo de procesamiento agregando más líneas que cuiden las proporciones de los caracteres y así mantener su forma a fin de que antes de la extracción de características se tenga un caracter bien formado.

7.3 Generación de una Base Vectorial de imágenes

Luego de la Normalización de los caracteres se usó la Transformada Karhunen-Lòeve (KL) para poder representar las matrices que contienen los caracteres en vectores de tamaño reducido. Para ello primero es necesario generar una base ortonormal con la cual cualquier caracter pueda ser representado como una combinación lineal de los vectores propios que generan esta base (Fig. 7.14).

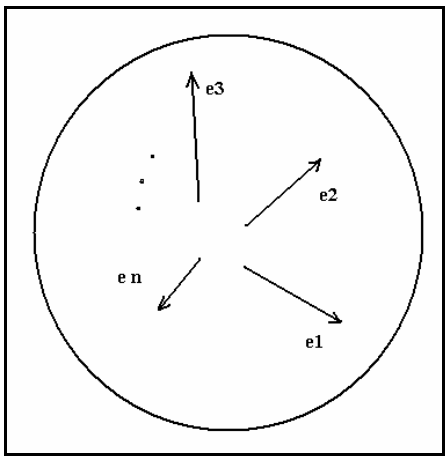


Figura 7.14 Base Ortonormal de vectores e_i

Es decir, si los elementos de los caracteres que formaban matrices de 16x16 (Fig. 7.11) se ordenan formando vectores columna de 16x16 = 256 elementos, cada uno de estos vectores será un vector imagen V_i (columna), que podrá ser representado como una combinación lineal de la base ortogonal de vectores de la figura 7.14:

$$V = w_1 \times e_1 + w_2 \times e_2 + w_3 \times e_3 + ... + w_n \times e_n \qquad \text{Ec. 7.1}$$

$$= \begin{bmatrix} e_1 & e_2 & e_3 & ... & e_n \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ ... \\ w_n \end{bmatrix} \qquad \text{Ec.7.2}$$

$$V = \Psi W \qquad \text{Ec. 7.3}$$

Donde los vectores e_i son columnas que forman la matriz Ψ y los w_i son números reales. Para generar la base de vectores ortonormales definida por la matriz Ψ, se utiliza la matriz de covarianza **R** (matriz real y simétrica), dado que esta matriz es diagonalizada por una matriz ortogonal (Ψ) (para su cálculo previamente se redefinen los píxeles: Píxel negro → +1, Píxel blanco → -1) (**[4] y [21]**).

$$R\Psi = \Psi\Lambda \qquad \text{Ec. 7.4}$$

Donde Λ es una matriz diagonal que contiene los valores propios λ_i en la diagonal principal, además los vectores propios son las direcciones de máxima varianza. La matriz de covarianza se puede calcular como:

$$R = (V - \bar{V})(V - \bar{V})^{\mathrm{T}} \qquad \text{Ec. 7.5}$$

Donde V, será la matriz que contiene todas las imágenes de los caracteres representados en vectores columnas de 256 = N^2 elementos. Si consideramos "M" imágenes, la matriz V será:

$$V = [v_1\ v_2\ v_3 \ldots\ldots v_M] \qquad \text{Ec. 7.6}$$

De dimensión N^2 x M, por lo que la matriz de covarianza **R** será de dimensiones: N^2 x N^2. Por tanto la matriz Ψ será una matriz cuadrada de orden N^2, en nuestro caso significa una base de 256 vectores ortogonales (de 256 elementos cada uno: $\Psi_{256\ x\ 256}$). Graficando los autovalores obtenidos en orden decreciente se tiene (Fig. 7.15) que estos decrecen monótonamente y de forma rápida, se puede ver entonces que la energía está altamente concentrada en los "primeros" coeficientes, entonces se obtiene que tomando los 41 primeros de los 256 en total, se tiene el 90% de la energía.

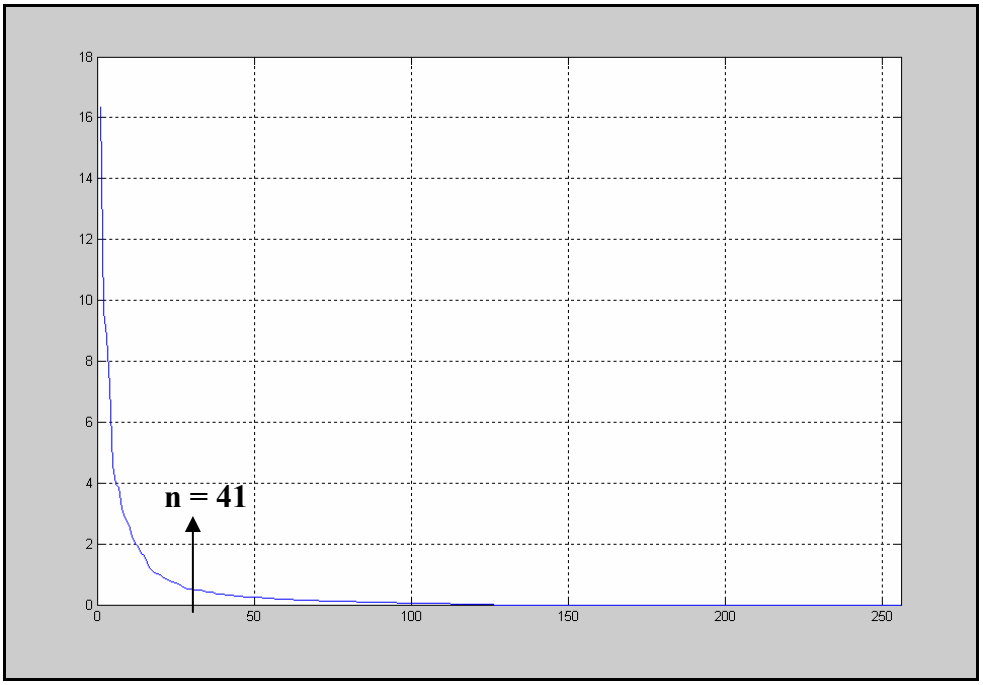


Figura 7.15 Espectro de autovalores. La energía está concentrada en los primeros coeficientes

Así ya tenemos la base ortonormal Ψ_{256x41} (reducida) formada por 41 vectores ortogonales correspondientes a los 41 autovalores que concentran el 90% de la energía.

De esta manera cualquier imagen (V_{256x1}) de un carácter de 16x16, puede ser representada como una combinación lineal de los 41 autovectores que forman la base vectorial.

A partir de la base de datos de 145 fotografías, para la generación de las Bases Ortogonales, se usaron 485 muestras (imágenes binarias de los caracteres).

7.4. Extracción de patrones

A través de la Transformada KL se puede obtener los valores reales de esa combinación lineal. De la Ec. 7.3 obtenemos:

$$W = \Psi^T V \qquad \text{Ec. 7.7}$$

Dado que $\Psi^{-1} = \Psi^T$ se cumple para matrices ortogonales. En el vector W_{41x1} ya tenemos representada la imagen con pocos elementos, lo que constituye la extracción de sus características o su correspondiente patrón.

En el Anexo N mostramos el código fuente de este capitulo.

CAPÍTULO VIII
CLASIFICACIÓN Y RECONOCIMIENTO DE PATRONES USANDO UNA RED NEURONAL

Una vez extraídas las características ya podemos clasificar los caracteres en diferentes categorías. Existen varias técnicas de clasificación entre las cuales tenemos: Técnicas estadísticas, support vector machine, redes neuronales, etc.

En general un clasificador es aquel que toma un conjunto de características como una entrada y le asigna una determinada clase. Los clasificadores como las redes neuronales, se construyen tomando un conjunto de patrones de entrada en vectores de R^n, que sirven de entrenamiento para ajustar los parámetros de la función de decisión, este proceso se repite para cada elemento del conjunto de entrenamiento asignándoles los puntos correspondientes de las diferentes clases. El proceso termina cuando se encuentran los parámetros de la función de decisión que producen un error mínimo determinado. Hemos elegido como clasificador una red neuronal, ya que en la literatura se sabe que es uno de los que procesa la información de manera más rápida obteniéndose buenos resultados.

Para la clasificación es necesario establecer conjuntos de muestras de cada clase con las que se van a obtener los parámetros del clasificador. A continuación vamos a desarrollar con más detalle primero la formación de las clases y luego la implementación de la red neuronal en Matlab, describiendo el proceso de obtención de los parámetros de la red a través de su entrenamiento.

8.1 Formación de Clases (Conjunto de Entrenamiento)

Para obtener los parámetros de la red neuronal debemos formar previamente conjuntos de entrenamiento constituidos por los caracteres que representarán las diferentes clases. Nuestro primer bloque fue una matriz que contenía los 10 primeros números y las 26 letras del alfabeto representados en vectores columnas V_i (ya comprimidos utilizando como extractor de características la Transformada KL) de 41 elementos:

$$\text{Alfa\_num}_{41xM} = [V_1\ V_2\ \ldots\ V_M]_{41xM} \qquad \text{Ec. 8.1}$$

Este primer bloque, *Muestras Medias*$_{256x36}$, entrena la red con los mejores bien formados caracteres, extraídos de la base de fotografías, que vendrían a ser 10 números + 26 letras = **36** caracteres, entonces así obtenemos la matriz:

load Training_Basic Muestras_medias *% cargamos el bloque de imágenes base*

Componentes_principales=KLT(Bases_reduc,Muestras_medias); *% extracc. de caract.*

Componentes_principales $_{41x36}$ =Alfa_num $_{41x36}$. *% finalmente obtenemos la matriz*

Esta es la entrada al clasificador, la salida es una matriz formada por vectores columna donde el valor "1" en una fila indica la posición de una clase determinada (número o letra) y el resto de valores deben ser "0". Como es lógico, estos vectores tienen 36 filas dada esa cantidad de clases, 10 números + 26 letras.

load Output_Basic Vect_Obj_Basicos

El segundo bloque de entrenamiento se forma de manera similar con la diferencia que escogemos caracteres con ruido o algo deformes, son 485 patrones y sirven para entrenar al clasificador y hacerlo más robusto.

8.2 Definición de la Red Neuronal

La teoría de Redes Neuronales tiene un pasado de aproximadamente 5 décadas, pero solo en los últimos 20 años se han encontrado sus sólidas aplicaciones, y el campo aún sigue en desarrollo.

Las redes Neuronales están compuestas por simples elementos operando en paralelo. Fue inspirado por los sistemas nerviosos biológicos. Como en la naturaleza, la función de la red es determinada principalmente por las conexiones entre los elementos, nosotros podemos entrenar una red neuronal para efectuar una función particular ajustando los valores de las conexiones (pesos) entre los elementos. Comúnmente las redes neuronales son entrenadas para una entrada particular que conduzca a una salida específica u objetivo. Este se puede apreciar en la figura 8.1. Los pesos de la red son ajustados basados en la comparación de la salida y el objetivo, hasta que la salida de la red corresponda con el objetivo. Típicamente tales pares entrada/objetivo son usados en un aprendizaje supervisado, para entrenar la red.

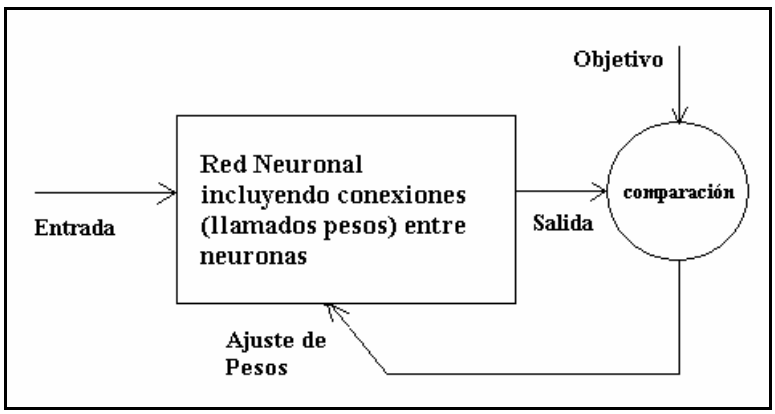


Figura 8.1 Proceso de entrenamiento: Aprendizaje supervisado

En la figura 8.2 se muestra el modelo de neurona elemental con R entradas, cada entrada tiene un peso determinado por su correspondiente w. La suma de las entradas pesadas y el bias (**b**) forman la entrada a la función de transferencia *f*.

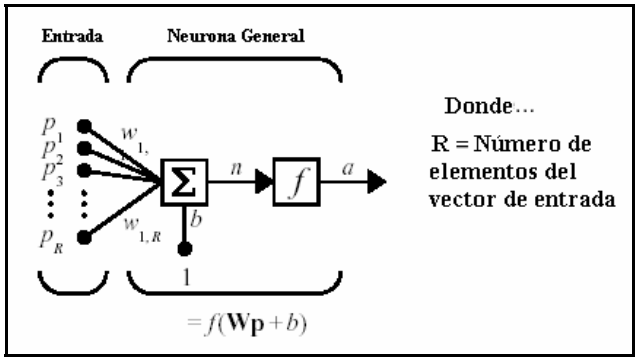


Figura 8.2 Modelo de neurona elemental

Las neuronas pueden usar cualquier función de transferencia diferenciable como salida. Las funciones utilizadas ***f*** en este trabajo se muestran en la figura 8.3.

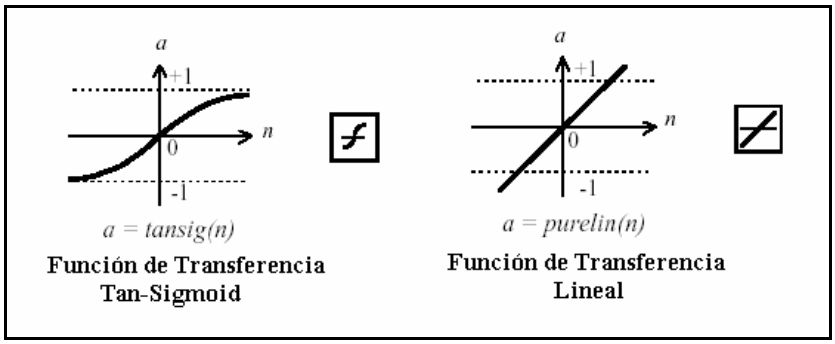


Figura 8.3 Funciones de Transferencia utilizados

A continuación se puede apreciar una red detallada con **R** entradas y de una sola capa de **S** neuronas con función de transferencia **logsig.**

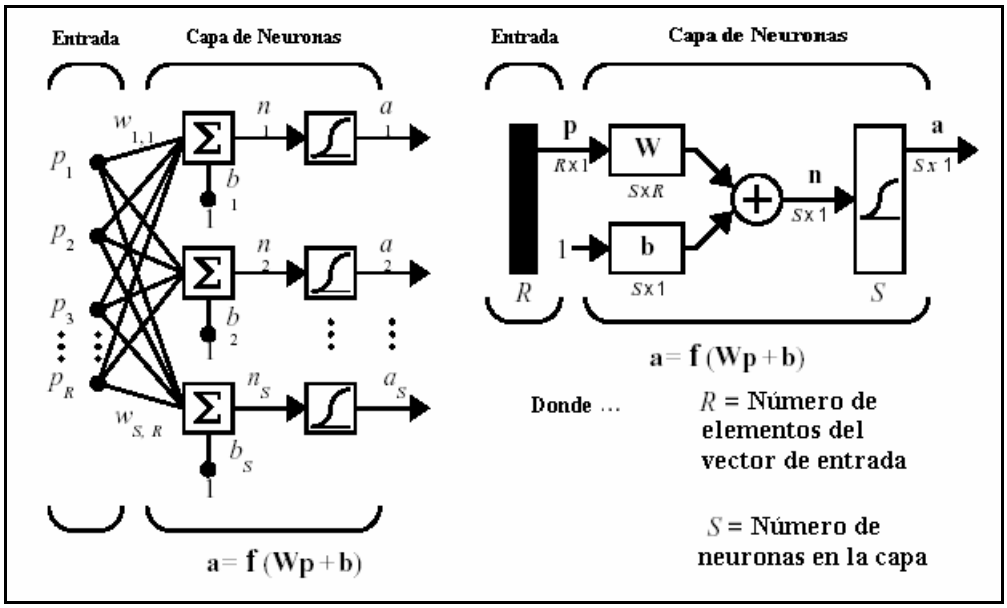


Figura 8.4 Red de una sola capa de 'S' neuronas y su representación simplificada

El clasificador Neuronal utilizado fue una Red multicapa Feedforward, ya que esta red nos permite clasificar vectores de entrada de una manera adecuada asociando su correspondiente salida. Las neuronas con funciones no lineales (sigmoidea) de la capa oculta y las de salida con funciones lineales, nos permiten aproximar cualquier función. Se utilizó el algoritmo de entrenamiento fue el gradiente escalado conjugado el cual es uno de los algoritmos de convergencia de error más rápido. La estructura de la red la definimos con 41 entradas (provenientes de la transformada KL), 41 neuronas en la capa intermedia y 36 neuronas en la capa de salida (ver Fig. 8.5).

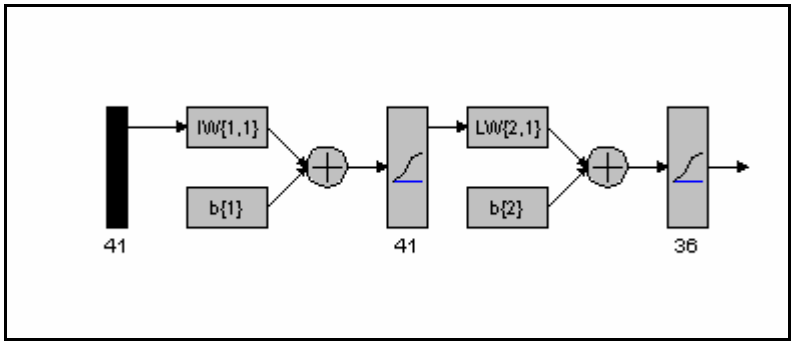


Figura 8.5 Estructura de la red neuronal utilizada

La determinación del número de neuronas en la capa oculta es en parte basada en la experiencia y en pruebas. Se comenzó probando con la mitad de neuronas de entrada hasta obtener una buena performance con 41, por lo que ya no fue necesario aumentar más.

8.3 Entrenamiento de la Red Neuronal

El entrenamiento de la red neuronal es un proceso que consiste en determinar los pesos de la red que producen un error menor que un máximo dado por nosotros. Finalmente es la optimización de una función de decisión en el que se busca los parámetros de la red que hacen mínima esa función error.

Este proceso con Matlab se vuelve una tarea sencilla, una vez definida la matriz de entrenamiento se debe también hacer otra que contendrá los correspondientes objetivos para cada clase, por ejemplo para el caracter "1" (uno), su correspondiente salida será una matriz columna de 36 elementos con un "1" en la primera posición y el resto todos ceros (la décima posición es para el cero). Así se tienen definidas todas las clases donde: las 10 primeras son los números del **1** al **9** con el **0** y las 26 clases restantes son los caracteres de la "**A**" a la "**Z**". En nuestro caso el objetivo fue una matriz de 36 filas (salidas correspondientes a las clases) y 36 columnas (número caracteres de prueba).

En Matlab definimos la matriz de entrada en P_{41x36} y el objetivo en T_{36x36} a partir de las matrices formadas en el capítulo anterior, el procedimiento sería:

>> load Training_Basic Muestras_medias

>> load bases_reducidas_41 Bases_reduc

>> Componentes_principales=KLT(Bases_reduc,Muestras_medias);

% hasta aquí tenemos las componentes principales de la entrada

>> load Output_Basic Vect_Obj_Básicos

% cargamos los vectores objetivos para el primer grupo en Vect_Obj_Básicos

>> P = Componentes_principales;

>> T = Vect_Obj_Básicos;

Luego definimos la red en la variable **"net"**:

>> net=newff(minmax(P),[41,36], {'logsig','logsig'},'trainscg');

Donde:

- **minmax(P)**: Establece el rango de entrada.
- **[41,36]**: define el número de neuronas de la capa oculta y la de salida.
- **{'logsig','logsig'}**: Son las funciones de transferencia (Fig. 8.3) respectivas de la capa oculta y la de salida.
- **'trainscg'**: Es la función de entrenamiento, básicamente define el algoritmo a utilizar en el entrenamiento de la red neuronal, para este caso es el de Levenberg-Marquardt (trainlm).

Algunos parámetros más:

>> net.trainParam.show = 100;

>> net.trainParam.epochs = 100000;

>> net.trainParam.goal = 1e-3;

Las dos primeras líneas indican que se mostrarán los resultados del error alcanzado cada 100 épocas hasta las 100000 que se indican, cada época equivale a una iteración, la cantidad depende del algoritmo de entrenamiento usado, si es uno rápido serán menos épocas. La última línea indica alcanzar un error menor de 0.001.

Antes de continuar con el entrenamiento quiero indicar que el proceso de entrenamiento es un proceso de prueba y error en busca de la mejor estructura que de la mejor performance, entonces los parámetros más importantes que se pueden variar para el entrenamiento son:

- Las funciones de transferencia de las neuronas (tansig, pureline, logsig, etc.)
- El algoritmo escogido para el entrenamiento (trainlm, trainscg, etc.)

- Número de neuronas en la capa de entrada (35, 41, etc.), éste esta en función a la cantidad de coeficientes escogidos en la etapa de extracción de patrones, pues son los mismos.
- Número de neuronas en la capa oculta, éste es el valor que más se va a probar. (17, 18, 19,… 35, 36, 37, … 40, 41, etc.).

Sólo la experiencia sobre la base de otros trabajos nos puede indicar el camino más corto en esta búsqueda.

Continuando con el entrenamiento, ahora procedemos a entrenar la red con el comando "**train**", que hace un entrenamiento por lotes y supervisado. El resultado en Matlab será:

>> [net,tr]=train(net,P,T);

TRAINSCG, Epoch 0/100000, MSE 0.354857/0.001, Gradient 0.113023/1e-006

TRAINSCG, Epoch 100/100000, MSE 0.00562824/0.001, Gradient 0.000204972/1e-006

TRAINSCG, Epoch 183/100000, MSE 0.000277201/0.001, Gradient 0.000804381/1e-006

TRAINSCG, Performance goal met.

Se aprecia que en la iteración o época 183 se alcanza el objetivo con un error de 0.0002772 que es menor a 0.001.

La curva de entrenamiento para este primer lote se puede apreciar en la figura 8.6, donde se alcanzó un error menor de 0.001 aproximadamente en 5 min, ya que se utilizó uno de los algoritmos más rápidos de entrenamiento.

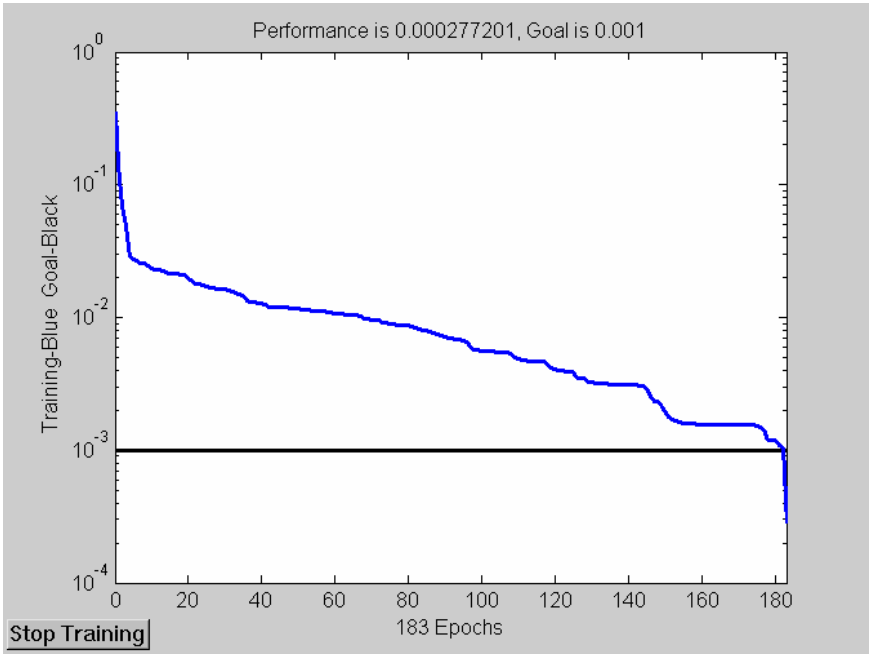


Figura 8.6 Curva de entrenamiento Error Vs. Épocas

Luego del primer entrenamiento formaremos un segundo lote de entrenamiento ahora de mayor tamaño y con caracteres mas deformados, producto del procesamiento, interpolación bilineal, ruido, etc. que se presentó en las etapas anteriores.

Esta segunda matriz contiene 485 caracteres extraídos de 145 placas. Obtenemos las componentes principales de cada caracter utilizando la transformada KL, formando la matriz: Componentes_principales $_{41 \text{ x } 485}$, además se debe construir la matriz de los correspondientes objetivos, nuevamente vectores de 36 elementos con un solo valor "1" para la clase correspondiente y el resto todos "0's". El proceso en Matlab sería:

>> load Matrix_training generador_de_base

>> load bases_reducidas_41 Bases_reduc

>> Componentes_principales=KLT(Bases_reduc,generador_de_base);

>> load Matrix_output Vect_Obj_AlfaNum

>> P = Componentes_principales;

>> T = Vect_Obj_AlfaNum;

Algunos parámetros más:

>> net.trainParam.show = 100;

net.trainParam.epochs = 100000;

net.trainParam.goal = 5e-3;

Aquí es importante recalcar lo siguiente, el error para este segundo bloque es mayor dado que son caracteres con ruido y deformaciones, entonces debemos indicarle a la red una mayor tolerancia con estos patrones, ya que si le diéramos el mismo valor de error como al grupo anterior estaríamos entrenándolo como si fueran patrones ideales, como consecuencia estaría aprendiendo un ruido en particular lo cual no favorece pues le restaría su capacidad para generalizar.

Continuando con el entrenamiento, ya no es necesario definir la red porque usaremos la misma dado que ya tiene los pesos adecuados producto de su primer entrenamiento. Entonces simplemente la entrenamos nuevamente con el segundo lote:

>> [net,tr]=train(net,P,T);

TRAINSCG, Epoch 0/100000, MSE 0.0155956/0.005, Gradient 0.00376583/1e-006

TRAINSCG, Epoch 10/100000, MSE 0.00473813/0.005, Gradient 0.00151386/1e-006

TRAINSCG, Performance goal met.

Efectivamente consiguió el objetivo con un error menor a 0.005. La curva de entrenamiento se aprecia en la figura 8.7.

Luego de este segundo entrenamiento la red pudo haber "olvidado" los patrones

correctos de los caracteres bien formados, para evitar eso se hace un último entrenamiento nuevamente con el grupo del primer bloque y así aseguramos que no haya distorsionado mucho los patrones correctos en su aprendizaje. Este procedimiento se sugiere para reconocimiento de caracteres en la referencia **[22],** Capitulo de aplicaciones (Character Recognition).

Entonces finalmente hacemos:

>> load Training_Basic Muestras_medias

>> load bases_reducidas_41 Bases_reduc

>> Componentes_principales=KLT(Bases_reduc,Muestras_medias);

>> load Output_Basic Vect_Obj_Basicos

>> P= Componentes_principales;

>> T= Vect_Obj_Basicos;

>> net.trainParam.show = 100;

>> net.trainParam.epochs = 100000;

>> net.trainParam.goal = 1e-3; % nuevamente el error en menor

>> [net,tr]=train(net,P,T);

TRAINSCG, Epoch 0/100000, MSE 0.000961215/0.001, Gradient 0.00157427/1e-006

TRAINSCG, Performance goal met.

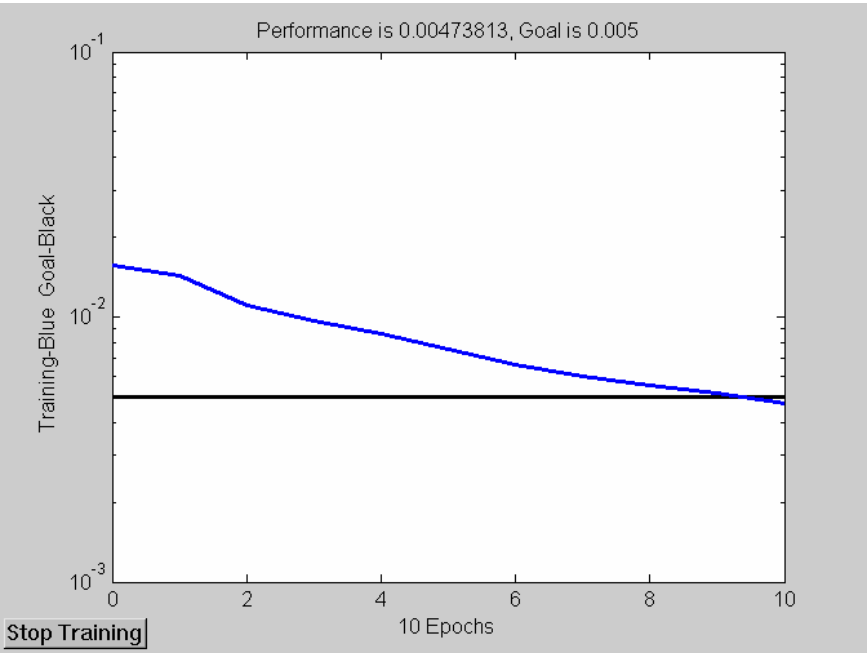


Figura 8.7 Curva de entrenamiento para el segundo lote

En resumen el entrenamiento para este sistema debe ser como sigue:

- ✓ 1er entrenamiento, error <0.001 (lote de patrones sin ruido)
- ✓ 2do entrenamiento, error<0.005 (lote de patrones con ruido)

- ✓ 3er entrenamiento, error <0.001 (lote inicial sin ruido)

Debemos anotar que para hallar la cantidad de neuronas a utilizar en la capa oculta, solo podemos saber el óptimo por prueba y error, así repetir y repetir haciendo pruebas y evaluando el rendimiento de la red ya en la simulación. La evaluación del comportamiento de la red lo veremos en el capítulo que sigue a continuación.

CAPÍTULO IX
RESULTADOS OBTENIDOS

9.1 Simulación

Hasta aquí ya se consiguió clasificar los patrones y la red ya debe haber aprendido los diferentes patrones de los caracteres, para verificarlo se implementó un código en Matlab que haga lo siguiente:

- Los caracteres normalizados en el arreglo de (16x16x6) deben pasar a formar las columnas de una nueva matriz que se llamará "caracteres_col $_{256 \times 6}$".
- A ésta matriz le aplicaremos la transformada KL para la extracción de sus características usando la base reducida de 35 autovectores (Ec. 7.7), con ello se obtiene una nueva matriz que denominaremos:

$$A_{35x6} = \text{"Componentes\_principales\_placa}_{35 \times 6}\text{"}.$$

- Ahora, la simulación se efectúa con el comando **"sim"** de la siguiente manera:

```
>> simula=sim(net,A);
>> A2=compet(simula);
```

Donde la función "compet" es una función de transferencia competitiva del Matlab, que asegura que el vector de salida correspondiente al caracter más parecido al vector de entrada ruidoso tome un valor de "1" y todos los otros 35 elementos tengan un valor de "0".

- Ubicamos la posición de los valores 1 en los vectores de salida:

```
for J=1:6
answer(J)=find(compet(A2(:,J))==1);
end
```

- Generamos un vector que contiene todos los caracteres almacenados como en un arreglo tipo "char":

```
C = [ '1' ; '2' ; '3' ; '4' ; '5' ; '6' ; '7' ; '8' ; '9'; '0' ;...
    'A' ; 'B' ; 'C' ; 'D' ; 'E' ; 'F' ; 'G' ; 'H' ; 'I' ; 'J';...
    'K' ; 'L' ; 'M' ; 'N' ; 'O' ; 'P' ; 'Q' ; 'R' ; 'S' ; 'T';...
    'U' ; 'V' ; 'W' ; 'X' ; 'Y' ; 'Z'];
```

- Añadimos unas reglas de decisión adicionales en los caracteres difíciles:

```
%% INTRODUCIMOS UN CODIGO PARA SELECCIONAR NUMEROS Y LETRAS
%% LOS 2 PRIMEROS DEBEN SER LETRAS
%% LOS 3 ULTIMOS DEBEN SER NUMEROS
for jj=1:2
 if answer(jj)== 8,
     answer(jj)=12;           % Para la letra "B"
  end;

 if answer(jj)==10,
     answer(jj)=25;           % Para la letra "O"
 end;

 if answer(jj)==5,
     answer(jj)=29;           % Para la letra "S"
 end;
end

for kk=4:6
 if answer(kk)== 12,
      answer(kk)=8;           % Para el número "8"
 end;

 if answer(kk)==25,
     answer(kk)=10;           % Para el número "0"
 end;

 if answer(kk)==29,
     answer(kk)=5;            % Para el número "5"
 end;

end
```

Finalmente como "**answer**" nos da las posiciones, la siguiente línea mostrará los caracteres en pantalla:

C(answer)

En el Anexo O se muestra todo el código fuente de esta parte.

9.2 Evaluación y Resultados Obtenidos

Para el entrenamiento y evaluación se utilizó una base primaria de 145 placas de donde se extrajo 485 caracteres utilizados en el entrenamiento, considerando 6 caracteres por placa que es lo normal, esto representa:

$$485 / 6 \approx 80.83 \text{ placas}$$

Es decir se utilizó aproximadamente 81 placas para el entrenamiento y se evaluó con todo el bloque de 145 placas todas mezcladas.

Luego de muchas pruebas y cambios en los parámetros que se indicaron en el apartado 8.3 para encontrar la estructura con mejor performance, se llegó a una red con 41 entradas (neuronas) y 37 neuronas en la capa oculta, con un porcentaje de acierto del 65%. Este nivel bajo se debió principalmente a que se estaba utilizando el mismo valor de error medio cuadrático de entrenamiento (MSE < 0.001) tanto para los lotes con ruido como aquellos sin ruido (como se explica en la parte final del Cap. 8). Se corrigió esto y se volvió a probar con nuevas estructuras como se aprecia en la siguiente tabla.

Número de entradas	No. de neuronas en la capa oculta	No. de neuronas en la capa de salida	Metodo de entrenamiento	Error (sin ruido y con ruido)	Convergencia	No. de placas que se tomaron de muestra para probar la tasa de reconocimiento	Tasa de Reconocimiento
41	26	36	Gradiente escalado conjugado	0.001 0.006	SI	27	77.78%
41	27	36	"	"	"	29	78.74%
41	28	36	"	"	"	44	76.89%
41	30	36	"	0.001 0.005	"		
41	32	36	"	"	"	51	83.01%
41	34	36	"	"	"		
41	41	36	"	0.001 0.005	"	139	81.89%

Tabla 9.1 Evaluación de la Red Neuronal

Se aprecia que los niveles de acierto aumentaron significativamente con el nuevo entrenamiento. Finalmente añadiendo unas reglas de decisión adicionales para establecer que los primeros caracteres deben ser letras y los últimos números, se probó la red con las 145 placas obteniéndose el siguiente porcentaje de acierto:

Extracción de Características	Clasificador Backpropagation
KLT	**83.22%**

Tabla 9.2 Porcentaje de acierto

Esto significa una tasa de reconocimiento global por placa del 83.22%, y representa aproximadamente fallar en un solo caracter de los 6 que normalmente tienen las placas de automóviles.

Esta tasa de acierto se debe a diversos factores:

- Están los casos de placas con marco, en las que éste en la etapa de binarización termina uniendo todos los caracteres, o en algunos, uno de los caracteres se encuentra pegado al marco, para estos casos se necesitaría desarrollar algoritmos más elaborados de segmentación, considerando la conectividad, razón de aspecto u otros parámetros.

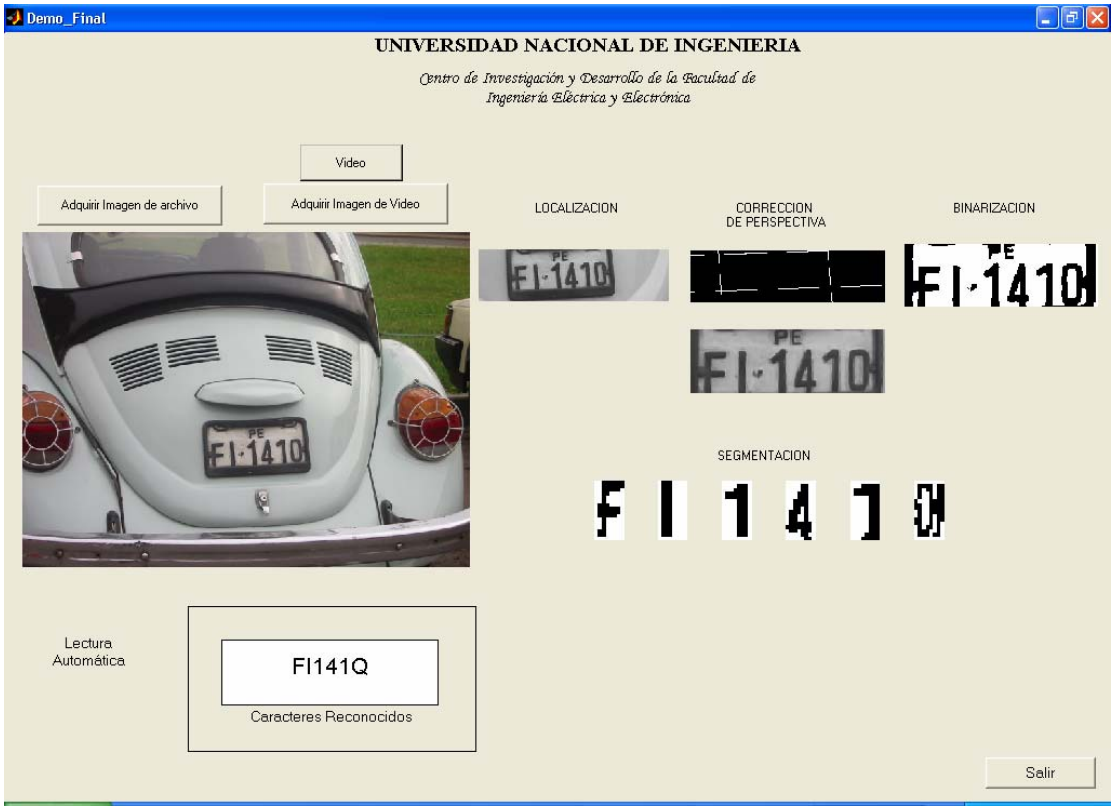


Figura 9.1 Marcos pegados a caracteres impiden un correcto reconocimiento

- En algunos casos en el proceso de determinar el marco que encierra la placa, para la corrección de perspectiva, otras líneas externas próximas a la placa pueden confundir el algoritmo. Esto se puede evitar haciendo el algoritmo de localización más preciso para evitar otras figuras.

Figura 9.2 En algunos casos otras rectas pueden confundir como bordes de la placa a la transformada de Hough

Figura 9.3 En la parte inferior, a los lados de la placa estos segmentos confunden el marco de la placa a la transformada de Hough

- Los casos con placas despintadas influyen en la binarización y en la segmentación.

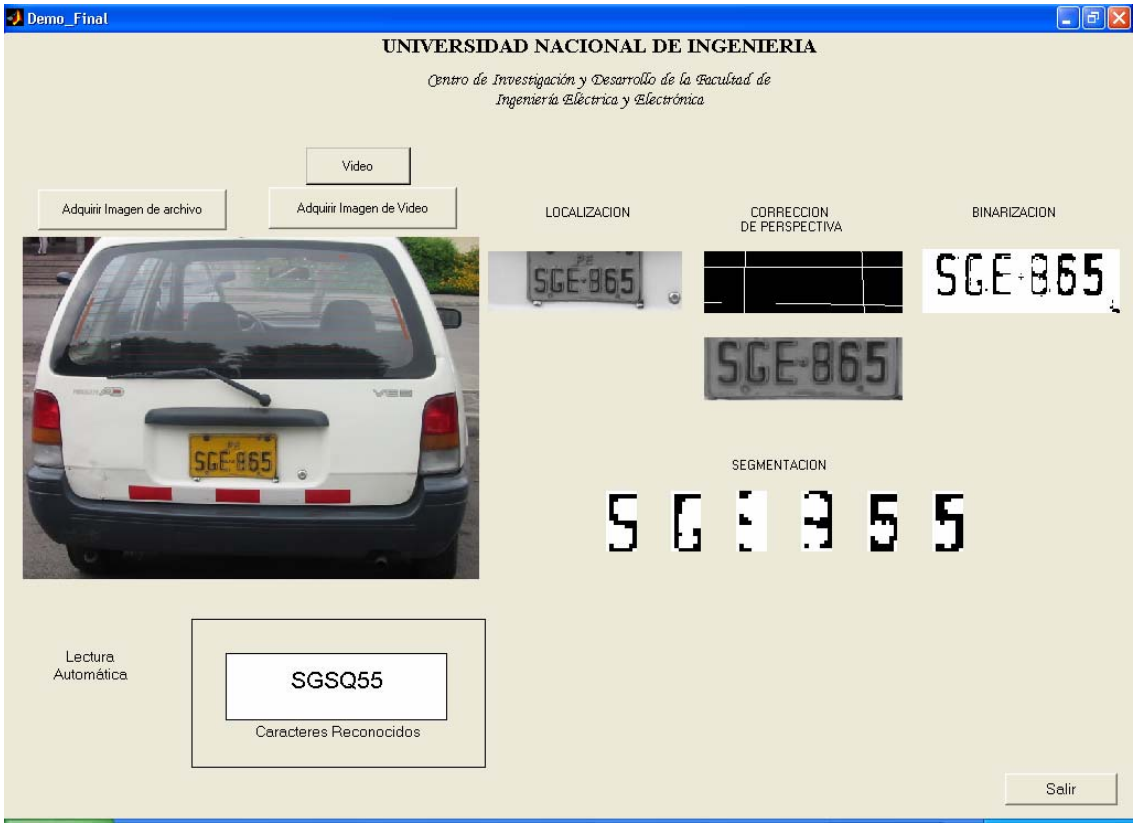


Figura 9.4 Caso de placa despintada

- Aún están los casos que por la naturaleza del caracter son difíciles de identificar inclusive para un operador humano. Como es el caso de las letras “O”, “D”, “Q”.

Figura 9.5 Caracteres difíciles de distinguir por naturaleza

Y en el caso de placas con 3 letras, diferenciar la “B” con el “8” como tercer carácter puesto que otras placas tienen solo 2 letras.

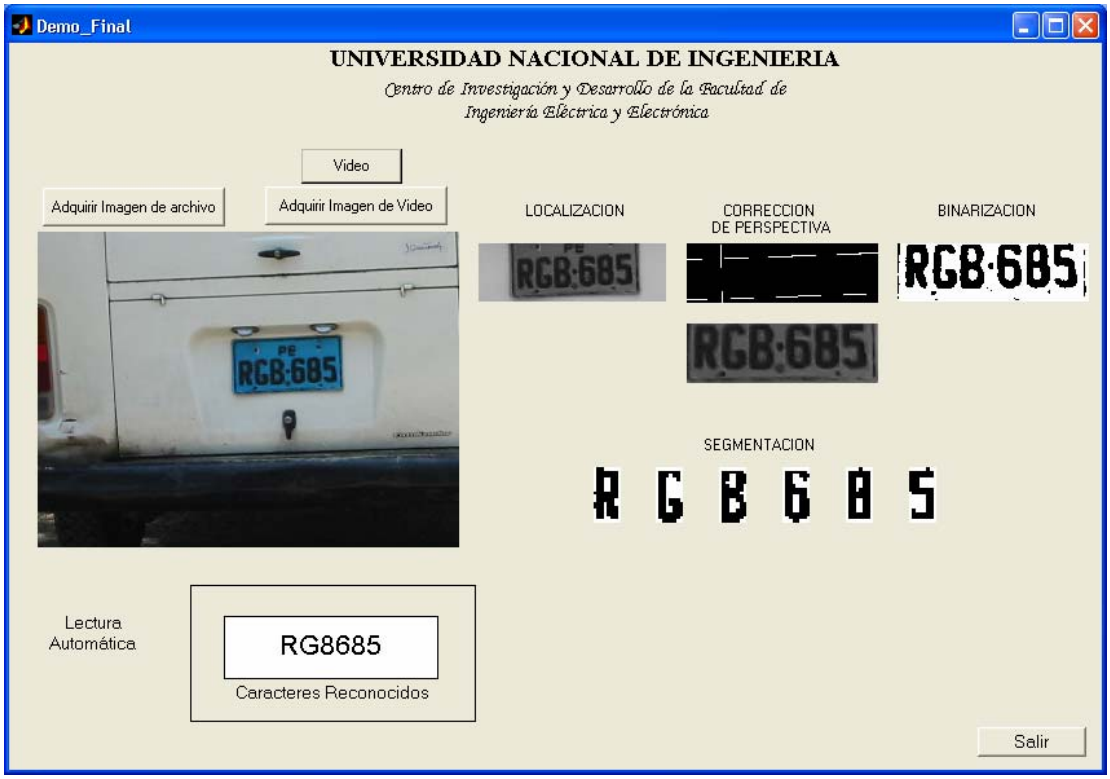


Figura 9.6 Caracteres parecidos “8” y “B”

Sin embargo pese a estos inconvenientes la gran mayoría respondió positivamente, de los cuales podemos resaltar:

- Los casos de perspectiva, fueron superados con la transformación proyectiva.

Figura 9.7 Caso de perspectiva con placa amarilla

Inclusive algunos casos un poco extremos, donde la placa logra recuperarse frontalmente.

Figura 9.8 Caso de perspectiva bien pronunciado

- Pese a la complejidad de la escena el algoritmo de localización trabajó muy bien, y a la vez la binarización de placas azules.

Figura 9.9 Pese a la complejidad de la imagen se extrajo y procesó la placa

Otras escenas complejas,

Figura 9.10 Otra escena compleja

localiza la placa pese a otras inscripciones,

Figura 9.11 Caso con inscripciones adicionales, símbolos, stickers, etc.

Y logra desechar otras figuras con muchos bordes que podrían generar bastantes gradientes y confundirse con una placa. Adicionalmente la red neuronal logró reconstruir algunos caracteres gracias a su capacidad de generalización.

Figura 9.12 Otro caso con simbolos

- La base de datos contiene algunas fotografías complejas, por lo que si se desea elevar esta tasa de reconocimiento sería necesario hacer un entrenamiento con una mayor cantidad de patrones buscando de la mejor estructura, de manera de hacer más robusta la red.

Por otra parte, en cuanto al tiempo de procesamiento, se evaluó el sistema en un computador personal AMD Athlon y un Pentium IV obteniéndose los siguientes resultados:

Hardware Utilizado		Tiempo de Ejecución Total Promedio (seg)
Procesador	Memoria	
AMD Athlon 900Mhz	256 MB	8.0
Intel Pentium IV 2.4 Ghz	512 MB	6.0

Tabla 9.3 Evaluación de los tiempos de ejecución de todo el proceso

Donde el mayor consumo de procesamiento se produce en los algoritmos de localización y el de corrección de perspectiva en la siguiente proporción promedio:

Algoritmo	Consumo de tiempo de procesamiento (%)
Localización	17.4%
Corrige_perspectiva2	59.2%
Total	**76.6%**

Tabla 9.4 Evaluación de las etapas que consumen mayor tiempo de procesamiento

Hace un consumo total de 76.6% del tiempo total de procesamiento y el 23.4% restante se reparte en los diferentes bloques y algoritmos en menores proporciones. Esto nos da una referencia en cuanto al cálculo intensivo de las etapas y las mejoras futuras que se pueden hacer para obtener un sistema más óptimo y comercial.

CAPÍTULO X
INTERFAZ CON SENSOR LASÉRICO Y COMUNICACIÓN SERIAL

10.1 Introducción

El sistema reconocedor de placas, se complementó con el diseño e implementación de un sensor lasérico, que le indique la presencia del vehículo cuando se aproxime, de manera que tome la fotografía de forma automática y realice el procesamiento de la imagen.

A continuación describiremos el diseño de los circuitos utilizados teniendo en cuenta consideraciones de diseño puntuales.

10.2 Objetivos

Diseñar circuitos electrónicos de detección de señales para control vehicular.

10.3 Diseño de los Circuitos

10.3.1 Especificaciones de Diseño

Nuestro sensor lasérico debe:

- ✓ Sensar la presencia de un objeto que atraviesa una región
- ✓ Cubrir largas distancias
- ✓ Tener bajo consumo de potencia
- ✓ Ser de bajo costo

10.3.2 Criterios de Diseño

Los dispositivos electrónicos ópticos tienen por naturaleza un rango de operación, esto es algunos diodos láser emiten luz en el espectro del color rojo, en un rango de frecuencias determinado, es por ello que el dispositivo a escoger como receptor debe poder capturar la luz emitida por el diodo láser a esa frecuencia de trabajo.

Por otro lado, para evitar el recalentamiento del dispositivo si estuviera permanentemente encendido, este debe llevar una señal alternante, para evitar así también un mayor consumo de potencia. El diodo láser al llevar una señal pulsante cumple la misma función de transmitir la luz, mientras el receptor capte la misma el sensor se encuentra enganchado.

En el lado del receptor, dado que para sensar la presencia de un objeto no necesitamos transmitir información en el haz de luz láser, no es necesario detectar la forma de los pulsos sino simplemente un valor medio de tensión. Si necesitáramos transmitir información se usaría un fototransistor, pero en este caso podemos simplificar la recepción utilizando un dispositivo electrónico que tenga una mayor inercia de respuesta en la recepción generando un valor DC ante la presencia de los pulsos y en ausencia 0V, elegimos pues entonces una LDR o fotocelda.

10.3.3 Análisis de la Naturaleza de las Señales a Tratar

a. Luz Láser

La luz láser es una radiación electromagnética con la propiedad de ser luz coherente y monocromática, es decir que tenemos un haz luminoso en el que todos los fotones tienen la misma frecuencia y están sincronizados a la misma fase, por lo que este tipo de luz puede llegar a cubrir largas distancias con menor atenuación que otra luz normal.

Esta propiedad en nuestro caso nos sirve para hacer una barrera de luz e inclusive se puede aprovechar las reflexiones generando zonas o áreas de seguridad. La aplicación de la luz láser se ha llevado fundamentalmente a las comunicaciones, instrumentación, control y la medicina. Esto ha sido posible gracias al diseño de dispositivos electrónicos capaces de generar este tipo de luz con distintas intensidades, de acuerdo a su aplicación.

b. Diodos Láser

Un diodo láser se fabrica colocando la unión de un diodo emisor de luz fabricado con un semiconductor ópticamente activo, entre dos superficies terminales pulidas parcialmente reflejantes que forman una cavidad ópticamente resonante llamada interferómetro, tal como se muestra en la figura 1. Cuando tiene polarización directa, la unión LED emite luz que interactúa con el resonador óptico para estimular una emisión adicional de luz de la unión, solo a una única longitud de onda dependiente del material. La resonancia óptica del interferómetro hace que la luz que sale del diodo láser sea coherente, así como extremadamente monocromática. [24]

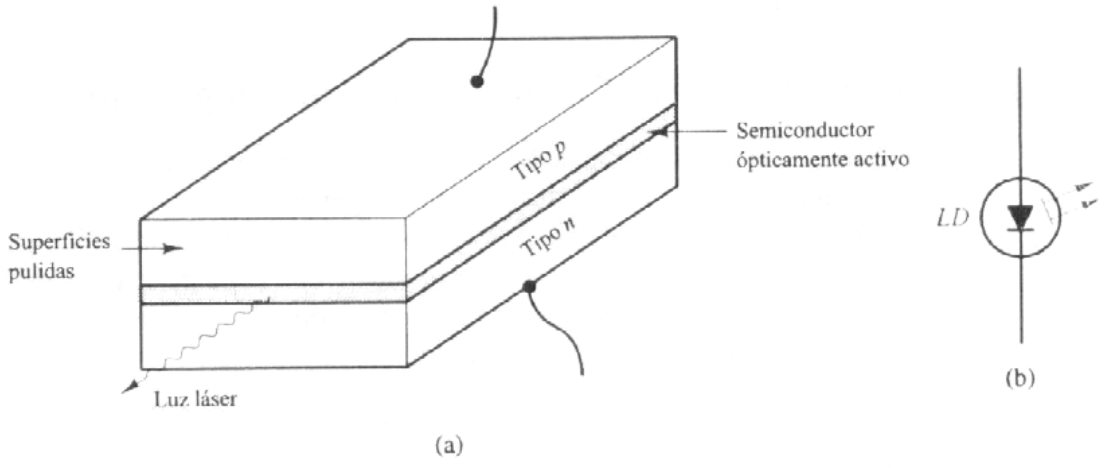


Figura 10.1.- Diodo láser semiconductor. **a)** Representación física. Las superficies pulidas forman una cavidad interferométrica. **b)** Símbolo del circuito. (Extraído de **[24]** Pág. 283)

La operación física del diodo láser semiconductor es idéntica en teoría a la de los láser gaseosos. Aunque muchos láser gaseosos emiten luz visible, los diodos láser emiten primordialmente luz en el infrarrojo. Esta característica es consecuencia de las propiedades físicas del arseniuro de galio y de otros semiconductores utilizados para la fabricación del diodo láser. El símbolo de circuito del diodo láser semiconductor aparece en la figura 1.b.

El diodo láser es un componente electrónico óptico capaz de generar este tipo de luz y nos permite controlarlo en función a su curva de trabajo luego de superar su voltaje umbral y teniendo en cuenta la potencia que disipa. Según esta última propiedad los diodos láser se clasifican en CLASE I, II , III y IV.

El diodo ha utilizar será de potencia menor a 1mW, según el rango de aplicación permitido y por que se ajusta a los requerimientos. Estos pertenecen a la clase II.

Por otro lado la eficiencia y la potencia óptica del diodo láser aumentan con la disminución de la temperatura, por lo que es deseable tener un ambiente de temperatura moderado.

10.3.4 Circuito Transmisor

El dispositivo consta de un diodo láser comercial de 1mW (aprox.) de potencia que emitirá luz de forma intermitente. De manera de no someterlo a demasiado esfuerzo. Veremos todas las consideraciones que se han tenido en cuenta para obtener el transmisor operativo.

1 Acondicionamiento de puntero láser para el transmisor

Los diodos láser comunes son de 1mW o 3mW de potencia, estos se venden por Internet a precios un poco altos aún, pero en el mercado nacional encontramos punteros láser Taiwaneses, si bien es cierto no son fabricados con insumos de buena calidad, pero se pueden acondicionar estos para realizar experiencias, siempre que tengamos en cuenta algunos aspectos para su acondicionamiento.

- Cortar parte del tubo metálico hasta llegar cerca del diodo láser, mantener el diodo en su sitio sin despegarlo. Si es posible retirar el switch o pulsador y así está listo para el siguiente paso.
- Soldar dos puntos con cables para poder extraerlos hacia fuera e independizarlo del switch. El polo negativo es el más cercano al resorte y el positivo se debe soldar en la plaquita impresa que se encuentra en la ranura. Aquí hay un punto crítico importante: Las soldaduras, especialmente la del polo positivo, deben ser en un instante, es decir colocar un punto muy rápido y si fuera posible soplar luego de la unión, ya que si se demora mucho se recalienta y se malogra el diodo. Para que pegue colocar a ambas superficies algo de soldadura previamente.
- Adicionalmente si se desea se pueden extraer los cables por el agujero que tiene en un lado del cuerpo del puntero (donde estaba el switch). Y eso es todo, ya tenemos un diodo láser para las pruebas. Debe quedar mas o menos como los que venden por Internet. (Fig. 2).

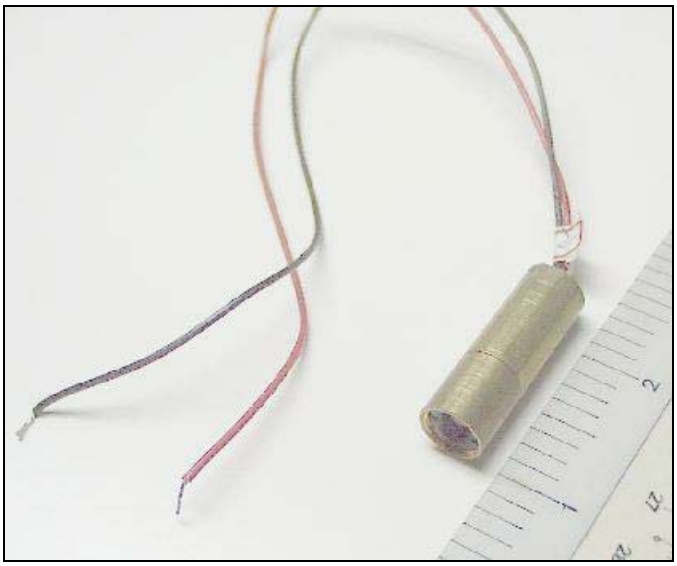


Figura 10.2.- Puntero láser comercial

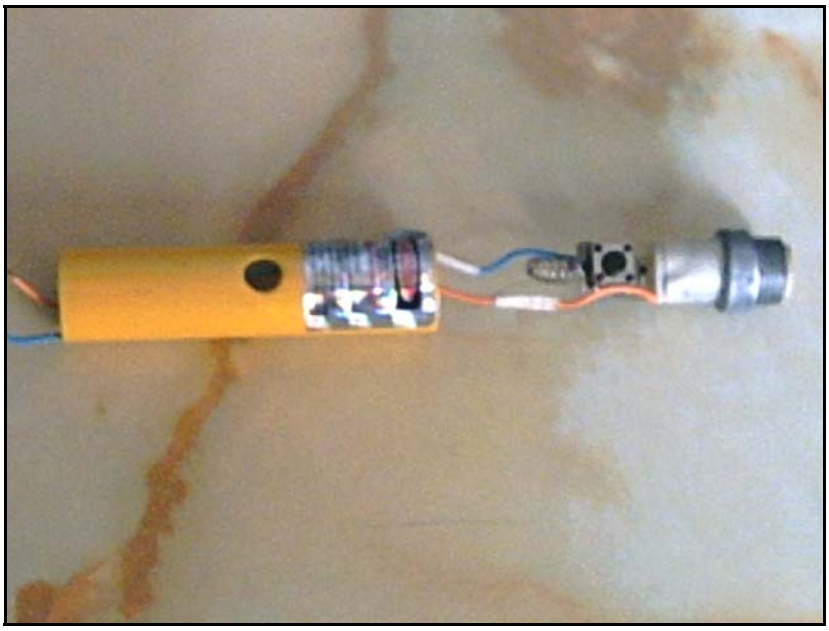


Figura 10.3.- Puntero Láser adaptado para las pruebas, mostrando su parte interna.

2 Diseño del Transmisor

Para el transmisor utilizaremos un generador de onda cuadrada de 10Khz, aproximadamente, esto lo conseguimos utilizando un Timer. En el circuito de la Figura 2. elegimos los valores de resistencias a partir de la fórmula:

$$f = \frac{1.44}{(R1+2*R2)*C} \qquad \text{EC. 10.1}$$

Eligiendo: **C = 11nF** con **f =10kHz**

tenemos: $R1+2*R2 \approx 13k\Omega$ tomando **R1 = 4.7kΩ** (valor comercial)

Así entonces obtenemos: $R2 = 4.15k\Omega$ colocando un potenciómetro de 10k cubrimos el valor. → **R2 = 10kΩ**

Con esto tenemos diseñado nuestro generador del transmisor que alimenta a un diodo láser de 1mW, adaptado a partir de un puntero láser común que se encuentra en el mercado actual como se vio el acondicionamiento del diodo láser en la parte 1.3.3.

El valor de frecuencia elegido se ha escogido en base a algunas pruebas y considerando que la oscilación en orden de los kHz permite que el dispositivo descanse mejor que cuando conmuta a baja frecuencia.

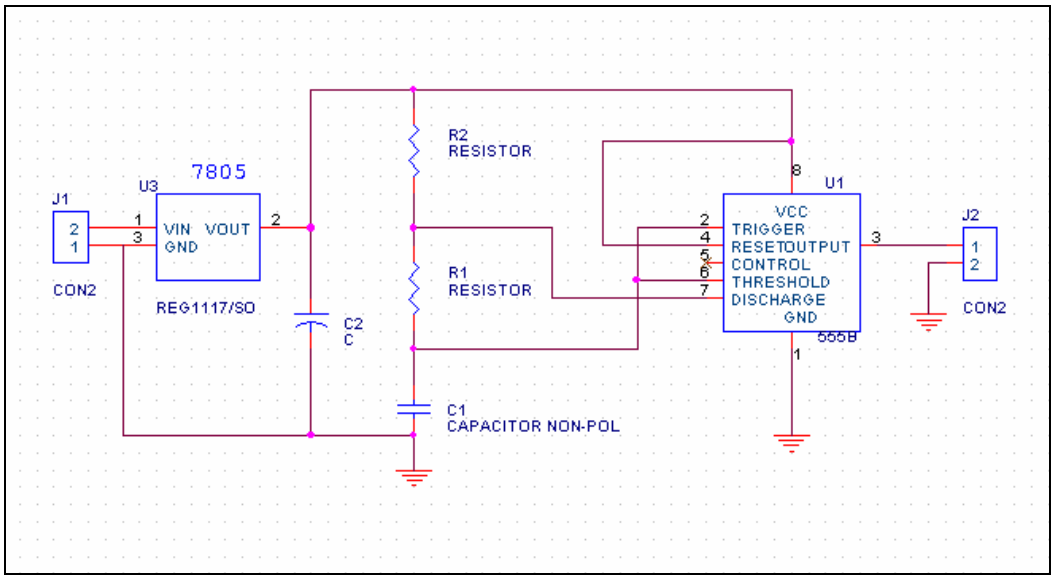


Figura 10.4.- Diseño del circuito transmisor

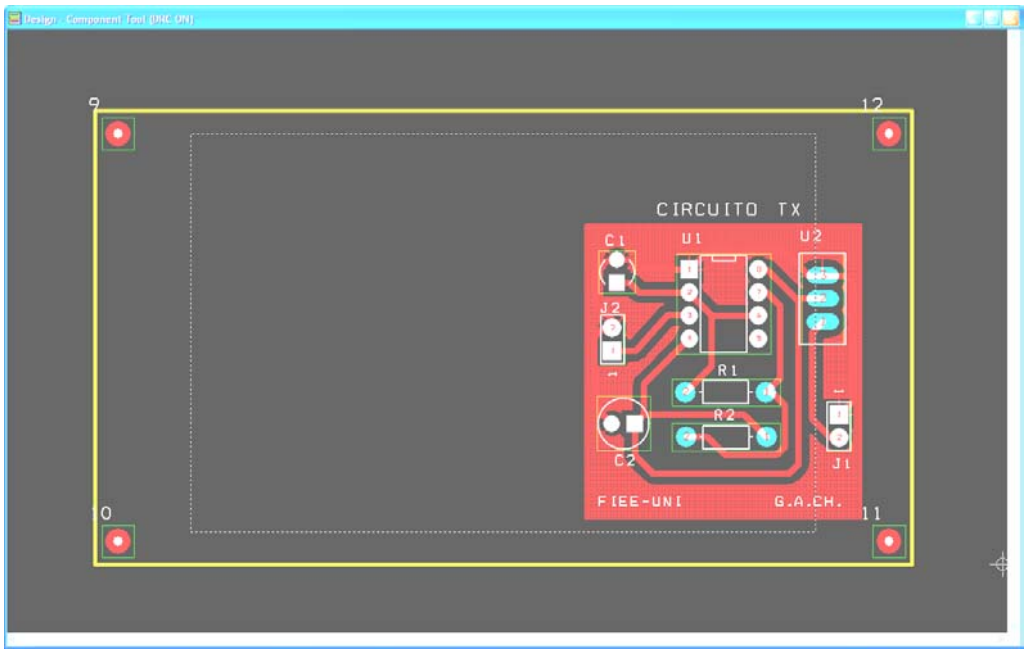


Figura 10.5.- Layout del circuito Transmisor en Orcad

10.3.5 Circuito Receptor

1 Foto Resistor (LDR)

Cuando el LDR recibe luz, éste disminuye su resistencia y conforme llegue menor cantidad de luz sobre él, su resistencia se incrementa en función a la ecuación:

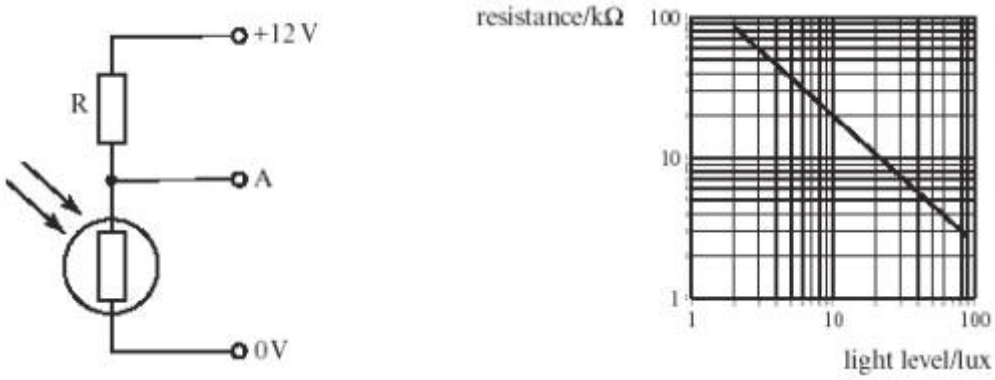
$$R = R_0 L^{\alpha} \qquad \text{EC. 10.2}$$

Donde:

R: Representa la resistencia en ohms (Ω).

L: Flujo luminoso sobre el área en expresado en lux.

R0 y α : constantes, con α < 0.



(a) **(b)**

Figura 10.7.- a) Símbolo del dispositivo. b) Dependencia de la resistencia con la iluminación

Esto significa que debemos ensamblar un circuito que detecte esa variación de resistencia, lo que se puede conseguir utilizando comparadores sencillos en este caso si utilizamos un Amplificador Operacional podemos conseguir el objetivo. Dado que es más práctico utilizar la configuración de un comparador usando una sola fuente, entonces se puede utilizar un LM324, que cumple con este requisito. Ahora, el LDR ha utilizar es uno de los más comerciales que se encuentran en el mercado, sus valores resistivos varían de la siguiente forma:

Con total brillo solar : $R \approx 200\Omega$.

En casi oscuridad Total : $R \approx 10M\Omega$.

Una vez armados los circuitos, el LDR del circuito receptor debe ser protegido con algún tipo de cubierta de manera que solo vea la luz láser a través de un agujero, puede ser un pequeño tubo oscuro de preferencia para que absorba otras emisiones y no confunda las señales con los cambios de luz ambiental.

2 Diseño del Receptor

Para el receptor existen diversas configuraciones utilizando transistores, amplificadores operacionales y/o filtros. Éste inicialmente se basó en receptores con transistor y timer en configuración monoestable (Figura 10.6), pero el único detalle es que tienen una cierta inercia para detectar objetos de movimiento rápido.

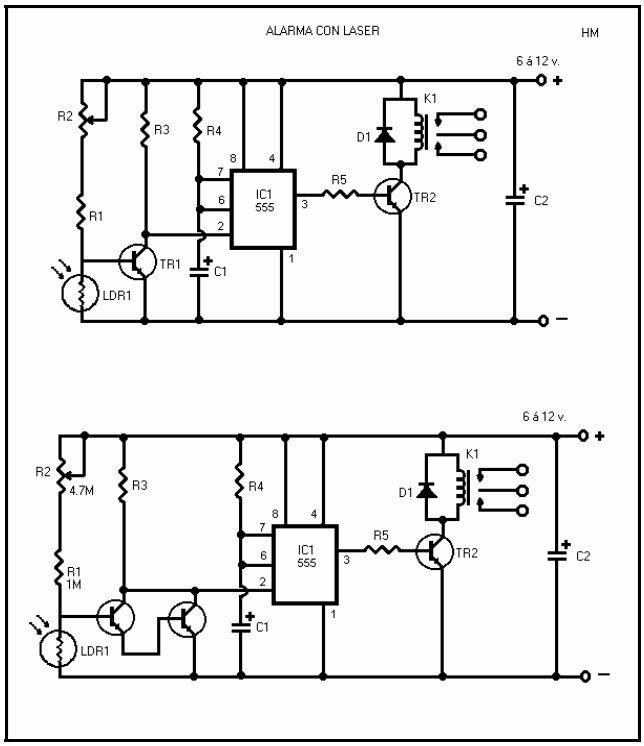


Figura 10.6.- Diseños de circuitos receptores disponibles en Internet

Estos circuitos reaccionan lentamente y no al instante. Hemos preferido simplificar el diseño manteniendo la confiabilidad y sensibilidad del mismo.

El receptor básicamente consta de un comparador en el que colocamos en una de las ramas al LDR, que producirá el desbalance en ausencia de luz al incrementar su resistencia. Fig. 10.8**.** Por facilidad decidimos colocar una bornera para poder conectar o cambiar el diodo láser. Una vez que se detectó el cambio, el OPAM arrojará un nivel alto, que en el caso del LM324, es el valor de fuente: 5V.

Para poder indicar al computador el cambio de estado del sensor decidimos utilizar un microcontrolador PIC, el cual detecta el cambio, incrementa un contador e inmediatamente envía el valor a través de su puerto de comunicación serie RS-232 en cuestión de microsegundos. Finalmente adaptamos los valores TTL provenientes del PIC a niveles RS-232 usando el driver MAX-232. Fig. 10.9.

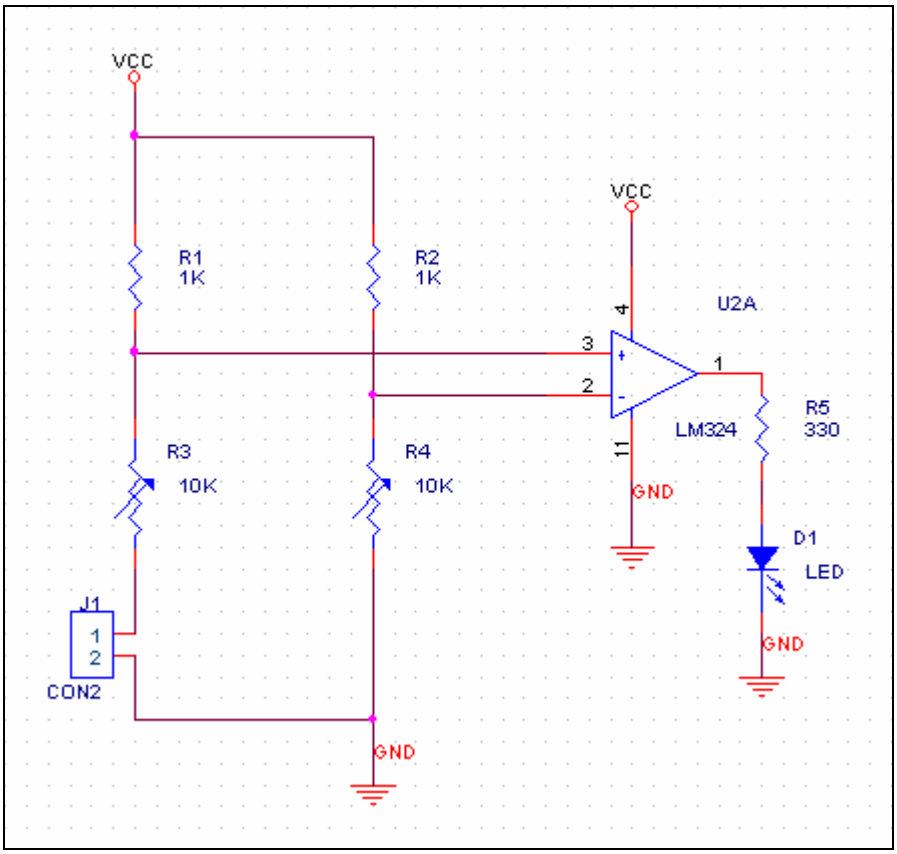


Figura 10. 8.- Etapa de Recepción de señal con OPAM

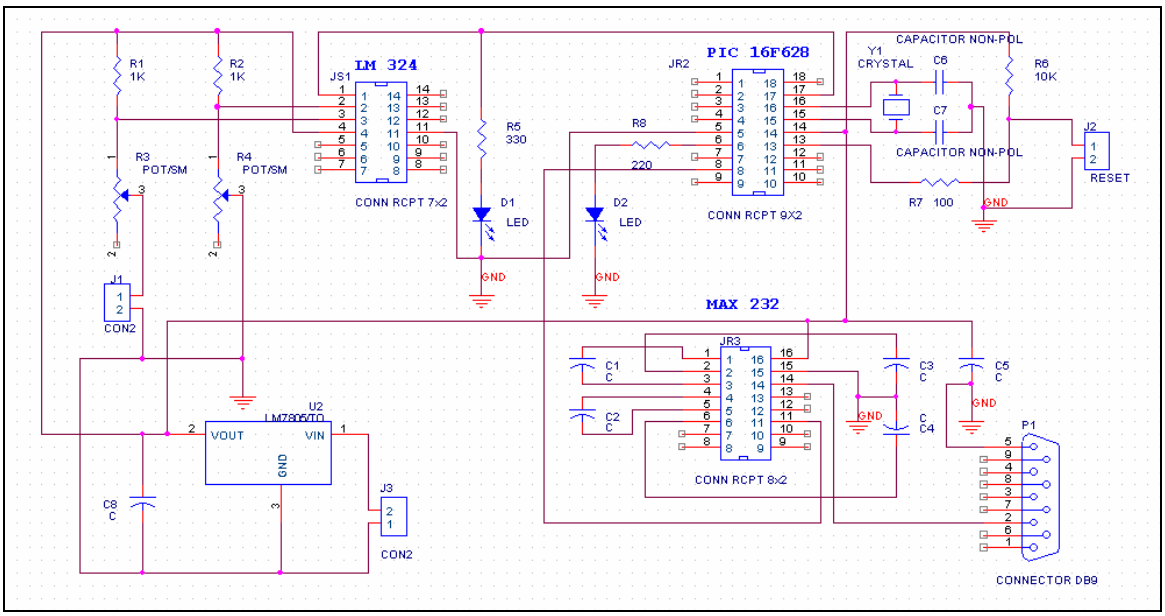


Figura 10. 9.- Diseño del Circuito Receptor

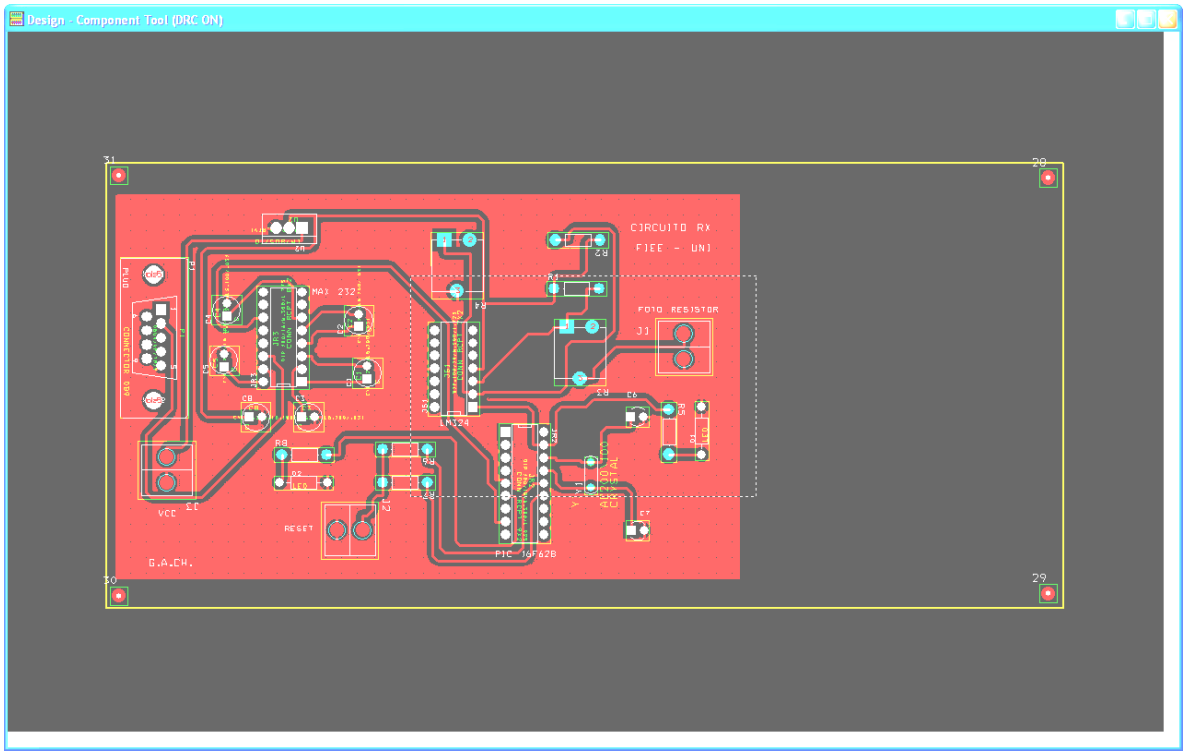


Figura 10.10.- Layout del Circuito Receptor en Orcad

10.4 Programación del Microcontrolador PIC

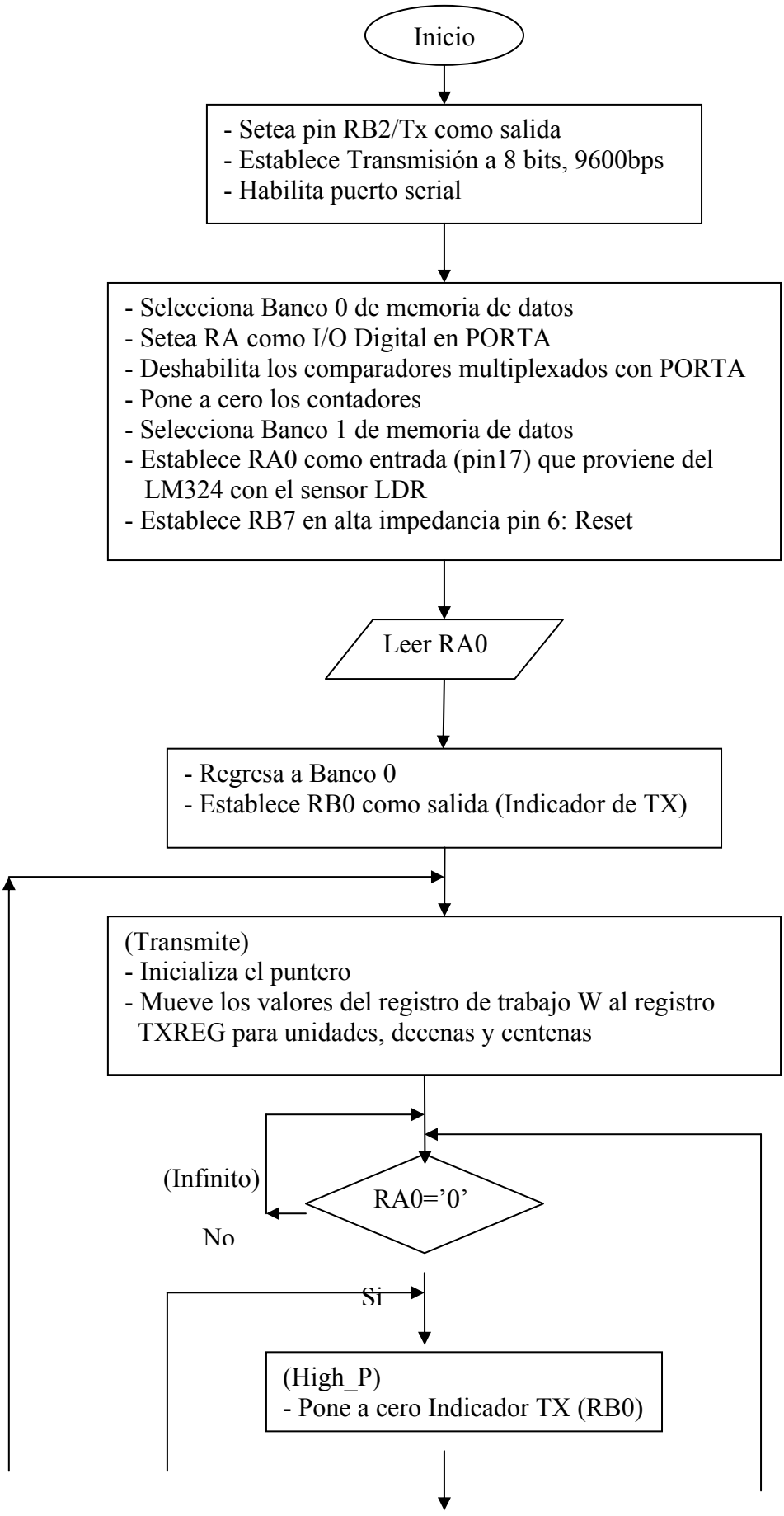
La función del Microcontrolador PIC viene a ser la de recepcionar, en el pin 17, la señal proveniente del sensor a través del LM324 y su pin de salida '1' ver figura 10.9 (el sensor LDR se coloca en el conector J1) y realizar una cuenta para cada activación del sensor debido a la presencia de un vehículo.

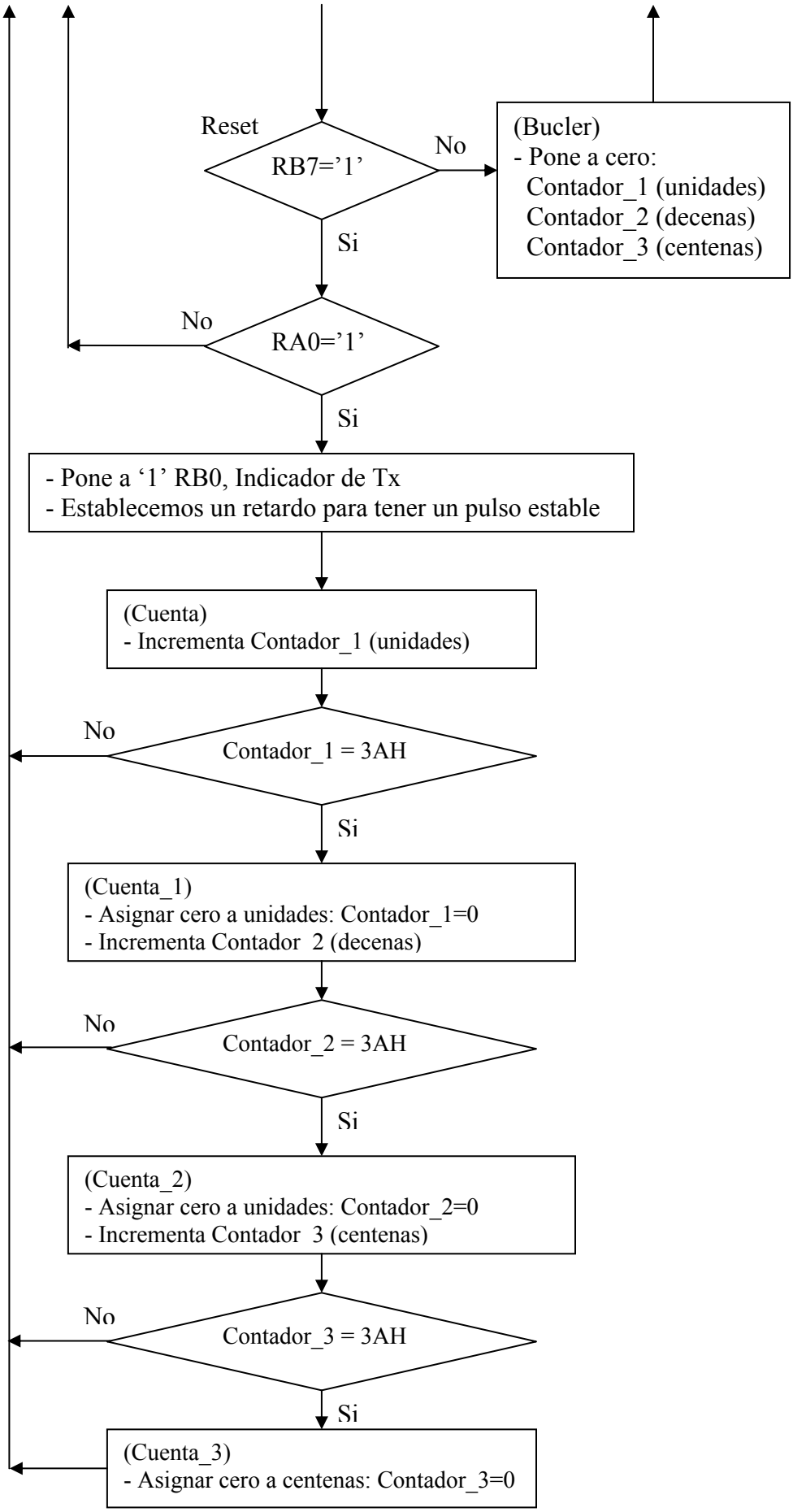
Para cada cuenta en el microcontrolador, éste envía al PC via comunicación serial RS-232, el valor de la cuenta y mostrando a través de un led la conformidad de la transmisión.

Cabe mencionar que el microcontrolador en sus pines se obtiene RS-232 a nivel TTL y no a los nivels de tensión del protocolo EIA-232, por lo que se utiliza el driver MAX232 que convierte los niveles de tensión adecuados para la transmisión serial RS-232.

10.4.1 Estructura del Algoritmo Implementado en el Microcontrolador

A continuación describiremos el algoritmo implementado en el microcontrolador en un diagrama de flujo.


Inicio
- Setea pin RB2/Tx como salida
- Establece Transmisión a 8 bits, 9600bps
- Habilita puerto serial
- Selecciona Banco 0 de memoria de datos
- Setea RA como I/O Digital en PORTA
- Deshabilita los comparadores multiplexados con PORTA
- Pone a cero los contadores
- Selecciona Banco 1 de memoria de datos
- Establece RA0 como entrada (pin17) que proviene del LM324 con el sensor LDR
- Establece RB7 en alta impedancia pin 6: Reset
Leer RA0
- Regresa a Banco 0
- Establece RB0 como salida (Indicador de TX)
(Transmite)
- Inicializa el puntero
- Mueve los valores del registro de trabajo W al registro TXREG para unidades, decenas y centenas
(Infinito)
RA0='0'
No
Si
(High_P)
- Pone a cero Indicador TX (RB0)


Reset
RB7='1'
No
(Bucler)
- Pone a cero:
Contador_1 (unidades)
Contador_2 (decenas)
Contador_3 (centenas)
Si
No
RA0='1'
Si
- Pone a '1' RB0, Indicador de Tx
- Establecemos un retardo para tener un pulso estable
(Cuenta)
- Incrementa Contador_1 (unidades)
No
Contador_1 = 3AH
Si
(Cuenta_1)
- Asignar cero a unidades: Contador_1=0
- Incrementa Contador 2 (decenas)
No
Contador_2 = 3AH
Si
(Cuenta_2)
- Asignar cero a unidades: Contador_2=0
- Incrementa Contador 3 (centenas)
No
Contador_3 = 3AH
Si
(Cuenta_3)
- Asignar cero a centenas: Contador_3=0

10.5 Comunicación Serial con Matlab

Para la comunicación a través del puerto serie se tuvo que añadir código que básicamente defina el objeto de tipo “puerto serial”, para abrirlo y estar a la espera de los datos escaneando o leyendo permanentemente el buffer de recepción. En el momento de recibir algún dato el sistema toma la fotografía y la procesa. Establecemos un número máximo de autos a sensar, en un parámetro sólo con fines demostrativos. Sin embargo puede contar hasta 999 autos y resetear la cuenta automáticamente en el Microcontrolador.

```
function togglebutton2_Callback(hObject, eventdata, handles)
button2_state = get(hObject,'Value');
%%% El sgte. codigo captura una imagen, a la señal del PIC
%%% desde una camara de video y la muestra.
% toggle button is pressed
  if button2_state == get(hObject,'Max')
  s=serial('COM1');
  fopen(s);
  set(s,'Timeout',0.10);
  warning off MATLAB:serial:fscanf:unsuccessfulRead
    scan=1;
  end
  n_cap=0;
  n=1;  %numero maximo de autos
  while (scan==1),
    buffer=fscanf(s);
    b=str2num(buffer); %version 7.0//Para 6.5 usar:wstr2num
    fprintf('leyendo..\n')
    if b>=1
      n_cap = n_cap + 1;
    if (n_cap<=n)        %b>0
      ii=vfm('grab',1);
      set(handles.text4,'string','')
      axes(handles.axes2)
      imshow(ii)
      set(handles.axes2,'XMinorTick','on')    %set(handles.figure1,)
      figure(1), imshow(ii)
      b
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Aplicacion del reconocimento de los
%%% caracteres. La salida es el numero de la placa
    num_placa=Reconocedor_Final(ii,handles);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Mostramos los caracteres
    set(handles.text4,'string',num_placa')
%%%
    pause(0.05)
    end
    end
    pause(0.05)
    if n_cap>=n
      scan=0;
      set(handles.togglebutton2,'Value',0)
    end
    if button2_state == get(hObject,'Min')
      scan=0;
```

```
        set(handles.togglebutton2,'Value',0)
        pause(0.05)
        fclose(s)
        delete(s)
        clear s
      end
   end
out=get(s,'Status');
if out=='open'
   fclose(s)
   delete(s)
   clear s
end
```

Cabe añadir que la interfaz de comunicación serial se implementó en Matlab 7, dado que en la versión 6.5 presentaba inestabilidad y el computador se colgaba. Sin embargo los algoritmos de procesamiento digital de la imagen se mantienen excepto por algunos cambios de nombre como erode por imerode y dilate por imdilate, etc.

Finalmente se corrigió aquellos comandos y el sistema está adaptado a la Versión de Matlab 7, en la que trabaja bien la captura de imagen de la cámara y el puerto serie, sin problemas.

CONCLUSIONES

En el proceso de recolección de fotografías se tuvieron en cuenta todos los aspectos estudiados, pero fueron la iluminación y la distancia al vehículo, los aspectos más críticos. En algunos casos la iluminación deficiente complica el proceso de Binarización, o una distancia muy alejada del vehículo resta resolución y en el peor de los casos no se podrían extraer los suficientes píxeles para los caracteres. A pesar de ello si se superaron otros problemas en cuanto a la posición de la cámara como el de la distorsión por perspectiva o la Binarización de placas oscuras en escala de grises como las de fondo azul o amarillo.

Por otro lado, se utilizó la cámara digital para tomar las fotografías que formaron la base de datos, y se diseñó una interfaz gráfica para la adquisición de imágenes a través de video utilizando la librería VFM, Video For Matlab. Esta interfaz se describe en el Anexo H.

La etapa de preprocesado no fue demasiada extensa, principalmente fue obtener una imagen fácil de procesar que siga conteniendo la información de la imagen. Algo muy importante que anotar es que la "decimación" en esta etapa, sirve para aplicar los algoritmos de localización a una imagen reducida de tamaño, esto reduce el tiempo de procesamiento y sirve para determinar solo las coordenadas de una zona que encierre la placa, luego de ello con las coordenadas, **se regresa a la imagen original en escala de grises y se trabaja con solo esa zona, pero con la resolución original, para no perder resolución en la zona de la placa.** El código fuente de esta parte se adjunta en el **Anexo J**, se encuentra incluido (por fines prácticos) dentro de la función Localización, de manera que siempre la Localización tiene este pre-tratamiento.

La localización de la placa ha sido una etapa algo complicada pero que es fundamental dentro de todo el proceso de Reconocimiento Óptico de Caracteres. Decimos esto porque me he basado en otros trabajos realizados como en **[4]** o **[6]**, pero no fue suficiente, porque nos encontramos ante una gama diversa de fotografías tomadas con cierto ángulo, en diversos escenarios es decir diversidad de fondos como ladrillos, vegetación, personas, etc. que como se ha visto podrían generar zonas con altos gradientes o en el peor de los casos de gran densidad de gradientes. En aquellos trabajos muchos

ejemplos son fotografías tomadas de frente con pocas figuras que podrían complicar la escena. Sin embargo se pudo superar estos inconvenientes combinando algunas cosas de uno y otro, con criterio y algo de imaginación. Es importante porque es el punto de partida para facilitar la labor de las etapas siguientes, por lo que la placa debe ser aislada con eficacia.

Hubo cosas que cayeron por su propio peso como la "decimación", pero otras no tan evidentes como la elección de un umbral determinado o el cálculo de una "densidad rectangular" adaptada al problema y que dio buenos resultados.

Es importante mencionar que fue esencial una base de datos variada pero tomada con cuidado, aún tenemos fotografías que son tan complicadas que el sistema encuentra una zona aparte de la placa. Por esto debemos tener una base de datos adecuada hasta con un cierto margen de tolerancia pero sin exagerar. Se ha utilizado esta base de fotografías porque se quiso hacer un sistema robusto para una variedad de escenarios como parqueos, cruces de calles, avenidas, etc. Tal vez sea necesario que algunos parámetros varíen según las necesidades. Éste al ser un sistema general básico aún tiene algunas limitaciones en ese sentido. El algoritmo no puede ubicar una placa sólo en aquellos casos que tengan menor contraste relativo (placas azules) los caracteres de la placa que en otras partes de la fotografía , como el caso de la fotografía **Foto_0014,** en el grupo de carpeta **Fotos Restantes**. Las letras "**NISSAN**" tienen mayor contraste que los de la placa azul y por tanto generan gradientes más grandes.

Figura 10.1 Caso de altos contrastes realtivos en otras zonas de caracteres (“NISSAN”), a diferencia de la placa de fondo azul

Otro factor importante es que las rectas que encierran la placa sean halladas de una manera tal que ésta encierre la placa de una manera efectiva. El programa para esta etapa ha sido previsto para que no caiga en bucles indeterminados de búsqueda de rectas que harían colgar al sistema.

En el proceso de Binarización se comenzó utilizando el método de Otsu, como en **[4]**, pero nuevamente la situación se complica, en este caso debido a la no-uniformidad de la iluminación que puede presentarse en muchos casos reales. Así que se tuvo que estudiar a fondo otras técnicas hasta llegar a las clasificaciones generales dentro del ámbito de la Segmentación de imágenes y sus niveles. Existe una gran variedad de métodos (algunos basados en el análisis de la entropía de la imagen, otros usando Lógica Difusa **[17]**, y otro también basada en la Transformada Karhunen-Loeve como mecanismo de de-correlación de datos y generación de la matriz de co-ocurrencia) implementarlos y probarlos todos sería impracticable ya que dilataría el tiempo en esta etapa, puesto que el criterio de simplicidad y rendimiento a la vez debe mantenerse, por esto se intenta parametrizar la

umbralización de manera de cumplir con el buen rendimiento de los algoritmos en la mayoría de los casos.

Se realizó una combinación de métodos de algunos autores, la reestructuración del histograma **[9]** Págs. 489-492, el método específico y la medición de la calidad en la Binarización **[4]**, fueron inspiración para adecuarlos convenientemente al problema en cuestión y de esta manera superar principalmente el inconveniente de las placas con fondo oscuro en escala de grises (placas amarillas, azules o con sombras).

En la extracción de caracteres se ha utilizado el método de las proyecciones porque es más sencillo de aplicar a los caracteres que no tienen traslapes entre sí, letras cursivas, etc. En general la segmentación de caracteres es un proceso complicado ya que muchas veces se presentan figuras con bastante ruido, algún caracter muy junto al borde de la placa, o aquellos casos con placas que tienen doble marco y en la parte inferior una barra que une todos los caracteres como se muestra en la Figura 2, estos casos que se presentan con cierta frecuencia son más complejos y el método de simples proyecciones no funciona, aquí se debe extraer los caracteres cuidadosamente analizando la conectividad y otros aspectos como sugieren algunos autores, esto implica un estudio más profundo.

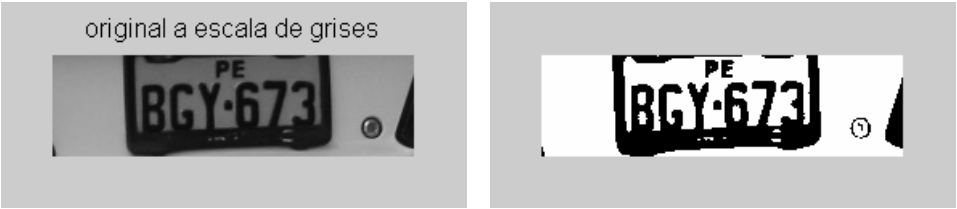


Figura 10.2 Marco que une los caracteres en la parte inferior.
En estos casos la segmentación debe ser más elaborada

La extracción de características a través de la transformada Karhunen-Lòeve demostró ser una forma efectiva de comprimir significativamente imágenes y extraer patrones en vectores con alta concentración de energía.

La Transformada KL es una transformada óptima en cuanto al menor error cuadrático medio y compactación de energía por tener la mayor covarianza en el menor número de coeficientes transformados, decorrela completamente la señal en el dominio transformado pero su implementación implica el cálculo de la matriz de covarianza, su diagonalización y la construcción de los vectores base, por lo que la incapacidad de predeterminar los vectores base, la convirtió en una herramienta en teoría ideal pero poco práctica. Sin embargo su uso puede ser de buena aplicación para la generación de la base ortogonal de manera “Off Line” y con ella la representación de las imágenes en el espacio transformado.

Gracias a las características de la transformada KL, otros autores han diseñado diversas técnicas para obtener una transformada rápida KL aunque son algo limitadas pueden proporcionar soluciones para compresión de información en ciertos casos.

El proceso de extracción de características se realiza off-line, y tiene, como pudimos apreciar su correspondiente capítulo, todo un procedimiento. Ese procedimiento se encuentra detallado en el archivo **extracción_de_características.m,** donde se usan otros como **generación_de_bases.m**, ahí también se encuentra el entrenamiento de la red neuronal.

Para el entrenamiento de la red neuronal, fue más fácil hacerlo con una menor cantidad de muestras (20), al incrementar el número a 485, este proceso se complicó, puesto que al haber más muestras hay más variedad, por tanto la red necesita una mayor capacidad para procesar más información, en consecuencia el número de coeficientes para la extracción de características tenía que aumentar, entonces la estructura necesaria de neuronas en la capa oculta es un número mayor y eso implica mayor cantidad de pruebas.
Una conclusión muy importante en el proceso de entrenamiento de la red neuronal, es que los patrones deben ser correspondidos con un error determinado (Cap. 8), para que la red "aprenda" a distinguir si los patrones tienen ruido o no. Puesto que si entrenamos la red indicando el mismo error para todos los patrones con ruido y sin ruido se corre el riesgo de enseñarle un ruido determinado y por tanto la red pierde su capacidad de generalizar.

Algo que podemos rescatar es que la característica de las redes feedforward de utilizar neuronas con funciones de transferencia sigmoidea en la capa oculta y funciones lineales en la capa final, permiten aproximar (ajuste) cualquier función. Ya sea lineal o no lineal.

La interfaz gráfica de usuario permite una mejor presentación y practicidad en el manejo del sistema **[23]**. Fue necesario utilizar la librería VFM para tener disponible la fuente de video en el Matlab, pero nos sirvió para desarrollar en poco tiempo la interfaz.

Finalmente se ha establecido las bases para el reconocimiento de caracteres de automóviles, estudiando los aspectos y consideraciones que son importantes desde la adquisición de la imagen y a través de cada etapa, ya que cada problema que se presentaba podía inducir a error en la siguiente lo que complicaba en sí el sistema.

El reconocimiento de caracteres a partir de imágenes fijas como fotografías o de una fuente de video es más complicado que de un texto impreso ya que por ser una fotografía de algo real, el objeto puede encontrarse rodeado de suciedad, estar inclinado, despintado,

etc. Y si añadimos las condiciones del clima como un día nublado, o con excesivo brillo solar, la imagen se complica. Así pues cuando se llega a la etapa de reconocimiento de los caracteres estos pueden tornarse complejos y en algunos casos asemejarse a caracteres escritos a mano. Fue importante pues mantener el objetivo de aproximarse cada vez más hacia los caracteres sin distorsionarlos y evitar un excesivo procesamiento, es por ello que se deben apostar en el futuro por técnicas efectivas pero no tan complicadas, manteniendo efectividad y buen rendimiento.

Debemos de tener en cuenta que en el presente trabajo, en principio se ha querido estudiar todos los aspectos y problemas que se pueden presentar en estos sistemas a la vez de proponer soluciones las cuales pueden ser optimizadas en trabajos futuros.

Finalmente se elaboró una interfaz gráfica en Matlab para adquirir las imágenes a partir de fotografías o de una cámara de video, el acondicionamiento de la cámara al sistema y otros aspectos complementarios se pueden ver en el Anexo G.

RECOMENDACIONES

Dado que el presente sistema fue desarrollado teniendo en cuenta consideraciones generales básicas para generar un prototipo, son importantes los aspectos particulares que se deben considerar según la aplicación que se le va a dar al sistema. Ya que si por ejemplo éste se implementa en una carretera, las cámaras a utilizar deben tener un alto valor de shutter, así como adecuar la iluminación. En cambio en un parqueo dentro de un recinto las velocidades de los vehículos no son mayores, así también no hay muchos inconvenientes con la iluminación en comparación a un ambiente natural. Así pues hay aspectos relacionados a la instalación así como algunas variaciones en los algoritmos según los requerimientos de la aplicación en cuestión. A continuación se exponen los diversos aspectos que se deben tener en cuenta a futuro.

I. Iluminación independiente de la luz ambiental.

Sería recomendable que adicionalmente a la cámara, se utilice un reflector infrarrojo potente, de manera que contribuya a la iluminación del objetivo sin que los conductores se percaten de la eventual iluminación y así capturar imágenes con independencia de la iluminación ambiental.

II. Cámara de video apropiada a determinada aplicación

Uno de los parámetros para elegir una determinada cámara está en función a la velocidad de captura, si es para carreteras, se necesitará un alto valor de shutter. Por ejemplo, un vehículo que se desplace a 80 Km/h tendrá un desplazamiento en cm/s equivalente de 2222.22 cm/s, lo que significa que por centímetro demorará 0.00045 seg/cm, con un valor de shutter de 1/100000 seg, la cámara fotográfica puede capturarlo. Si es una cámara de video se necesitará tomar el vehículo a distancia para tener varios cuadros esto implica que la cámara de video pueda enfocar la imagen desde lejos y para ello la lente debe tener una buena longitud focal (50mm mínimo), ahora si es variable mucho mejor, con lo que se podría hacer zoom analógico en la imagen. Es importante también que la cámara sea de escaneo progresivo y no entrelazado para tener una imagen bien definida. Algunos

fabricantes de estos sistemas, como A5 Security Consulting Group S.L., sugieren los siguientes modelos de cámaras:

Instalaciones interiores: PE 100

Instalaciones exteriores: PE 300 y SONY XC-55

De manera complementaria se puede utilizar la imagen a color del vehículo para establecer información extra de sus características como el color del vehículo, el modelo, etc.

III. Posicionamiento de la cámara

Algunos fabricantes sugieren colocar la cámara a unos 6m del vehículo, con un ángulo recomendable de 45^{o}, esto está en función de que la imagen tenga los suficientes píxeles para el procesamiento y reconocimiento, es decir establecer la relación entre longitud focal de la lente y cantidad de píxeles que se obtienen para la placa a una cierta distancia. La cantidad de píxeles mínimo que deben tener los caracteres de la placa debe ser aproximadamente 25. Si es menor no se podría garantizar que la reducción de tamaño (normalización, etapa de extracción de características) sea sin distorsión y por tanto haya un buen reconocimiento.

IV. Parametrización del umbral de gradientes con características de la imagen

El umbral de gradientes que se estableció para la localización de la placa estuvo basado en el histograma de gradientes, y fue una medida relativa de los gradientes más grandes, pero esta técnica puede fallar cuando existe alguna zona con altos gradientes como consecuencia de algún faro o exceso de brillo, así también puede fallar cuando existen en la escena otros elementos como plantas en cantidad que tienden a confundirse con gradientes altos. En ese sentido parametrizar el umbral de gradientes con características como el nivel de brillo, contraste, entropía en la imagen podría significar un conocimiento a priori para determinar un mejor umbral, y así parametrizarlo.

V. Técnica de Localización de caracteres

En la localización de caracteres se utilizó como base los gradientes de la imagen, estos nos sirvieron para aproximarnos a la placa, pero el criterio fue basado en la región de mayor densidad de gradientes. Después se optó por discriminar usando operadores morfológicos, en realidad se puede cambiar la técnica utilizando probablemente algunas reglas heurísticas de forma de ahorrarse operaciones morfológicas que como sabemos para implementarlas

en hardware no son óptimas ya que realizan operaciones en forma secuencial y lo mejor sería utilizar operaciones matemáticas que se implementen en paralelo. Finalmente en objetivo es aproximarse a la placa con el menor número de operaciones posibles y de manera efectiva, que sea capaz de discriminar una placa y no coger cualquier otro objeto.

VI. Conservación de proporciones en la Transformación Proyectiva

La corrección de perspectiva ayuda en forma general a obtener una mejor imagen de la placa ó varias en caso de vías con mayor cantidad de carriles en casos debido a la posición de la cámara si se tendrá mayor perspectiva. Sin embargo es importante que esta corrección se dé sobre un plano con dimensiones definidas, proporcionales a una placa convencional, dado que en nuestro algoritmo solo se consideró tomar como referencia la altura que se disponía en cada placa esto no garantiza que los caracteres tengan las proporciones adecuadas, por ello sería recomendable hacer la proyección de la transformación proyectiva sobre un rectángulo basado en sus proporciones reales, este puede obtenerse experimentalmente con la cámara y no debe ser de un tamaño exageradamente mayor ya que se estaría extrapolando y generando una imagen que probablemente no se asemeje a lo real.

VII. Parametrización del parámetro T en la Reestructuración de Histograma

En la etapa de Binarización se utilizó finalmente la reestructuración del histograma, y la aplicación del umbral óptimo, pero en el proceso de la reestructuración se utilizó un parámetro "**T**" de umbral de gradientes, este valor se fijó en 7, pero en algunos casos era necesario bajarlo hasta 1 ó 3 y en otros subirlo. Por tanto este parámetro también puede ponerse en función de características de la imagen como el nivel de brillo, contraste, entropía de la imagen, etc. Estableciendo una relación entre ellos y el parámetro **T**, la determinación del umbral sería adaptiva y de mejor calidad en función a la imagen y de manera automática. Debemos recordar que una buena Binarización permite una buena segmentación y caracteres mejor definidos.

VIII. Segmentación de caracteres basados en conectividad entre píxeles, dirección de gradientes, y reglas heurísticas

Como se pudo apreciar en el capítulo de segmentación aún queda otro desafío y es el caso de las placas con marcos que suelen unirse a los caracteres tanto en la parte inferior como

lateral, lo que dificulta su segmentación. Es por ello que para superar estos inconvenientes se necesitará considerar el grado de conectividad entre píxeles dentro de un caracter, así como medir la dirección de los gradientes en los mismos para determinar si un conjunto de píxeles adjuntos tienen la misma tendencia en orientación (dirección) o no. Esto se puede revisar en las referencias **[9]**, **[20]**, del capítulo de segmentación (3.8), y otras técnicas con mayor cantidad de reglas heurísticas y utilizando "thinning" de los caracteres se encuentran en **[4]**, **[5]**, del mismo, o cualquier otra bibliografía que trate sobre segmentación de caracteres para OCR's.

IX. Otras técnicas de clasificación de patrones

Sería importante probar otros clasificadores de patrones basados en bases de datos ó probabilísticos, tal vez una PNN (red neuronal probabilística) pueda dar mejores resultados así como combinar otras técnicas de extracción de características.

X. Grado de Confiabilidad en el reconocimiento

Se puede medir la confiabilidad en el reconocimiento, de manera que se puede afirmar que un caracter ha sido reconocido con un nivel de confianza, así si el caracter reconocido no supera un cierto margen existe la duda, y podría intervenir un operador humano. Esta técnica de medición se explica en **[1]**, del capítulo 3.10.

A N E X O S

ANEXO A
ESTRUCTURA GENERAL DEL SISTEMA

El sistema está constituido por bloques, cada uno es una función en Matlab y constituyen las diferentes etapas del sistema, algunos de estos bloques hacen llamadas a otras funciones para realizar tareas especificas.

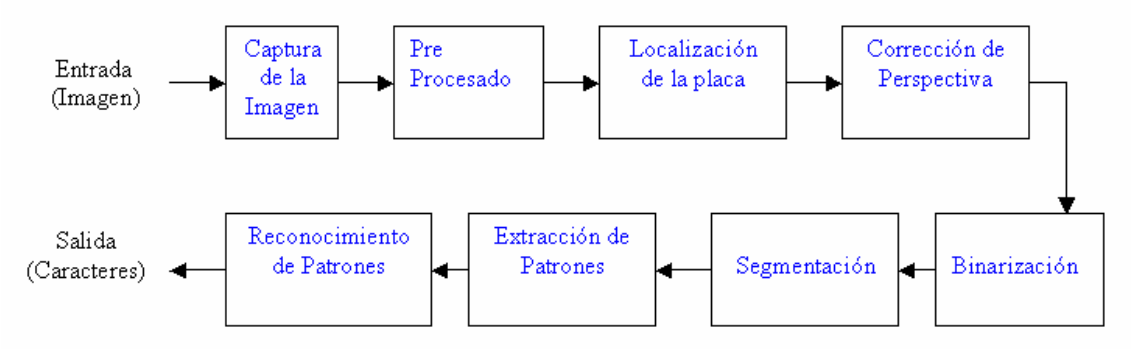
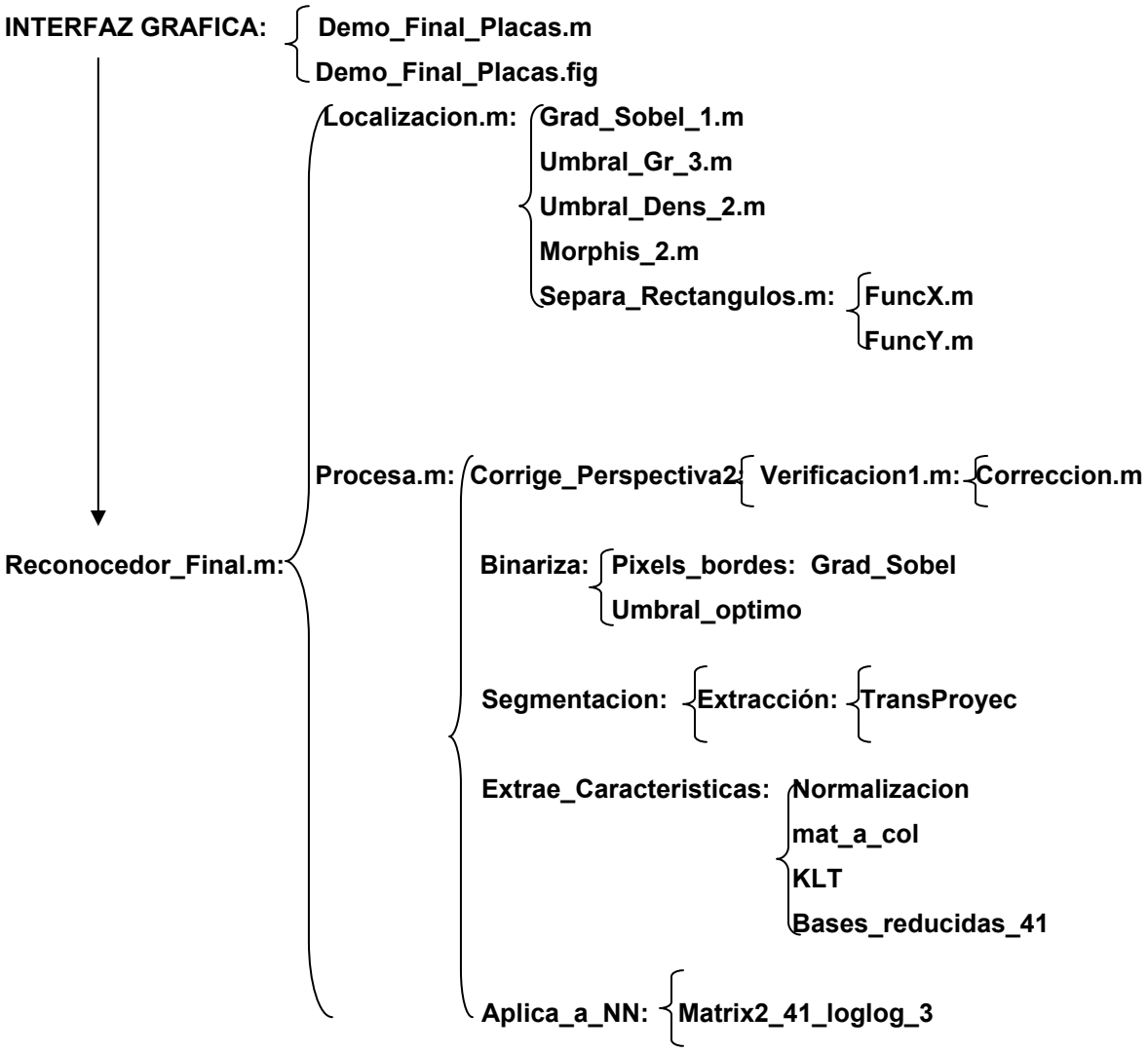


Figura A.1

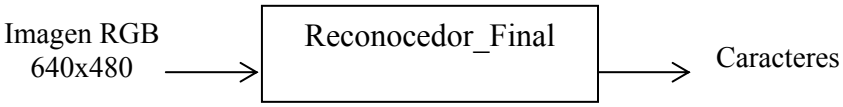
Desde el punto de vista jerárquico tenemos la siguiente estructura:



Esta viene a ser la distribución de las llamadas a todas las funciones y a otras variables involucradas como el arreglo que contiene la red neuronal.

Vamos a describir a continuación cada bloque desde el punto de vista funcional, sus entradas y salidas de manera que estos pueden ser reemplazados como módulos independientes o a la vez ser utilizados independientemente para alguna otra tarea en otras aplicaciones.

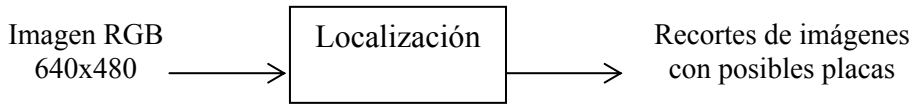
Tenemos un bloque principal que engloba todo el sistema llamado **Reconocedor_Final** éste tiene como entrada una imagen almacenada en una variable (ii), procesa la imagen a través de las llamadas a los diferentes bloques del sistema y finalmente arroja como salida los caracteres de la placa.



[num_placa] = Reconocedor_Final(ii)

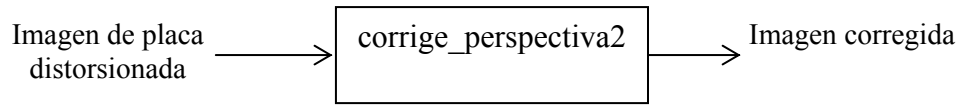
El bloque principal interiormente llama a dos funciones, estas son **Localización** y **Procesa**. "Localización" contiene el código del pre-procesado y la localización, mientras que "Procesa" contiene todas las llamadas al resto de bloques que conforman el sistema y adicionalmente sentencias condicionales para descartar con anticipación cualquier error y que no tenga que pasar la imagen por todo el sistema para descartar.

El bloque Localización tiene como entrada la imagen almacenada en una variable (ii) y como salidas, dos variables (zona1, zona2) para almacenar las posibles placas, otras dos (area1, area2) conteniendo sus respectivas áreas y una variable adicional de referencia (contador) con el número de "placas" encontradas.



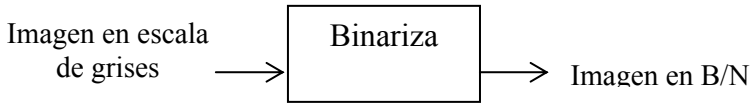

```
[zona1,zona2,area1,area2,contador] = Localización(ii);    % Puede usarse como función
                                                          % independiente o ser
                                                          % reemplazada

[num_placa,estado] = Procesa(zona1,area1);  % Este bloque llama al resto de bloques
                                            % del sistema luego de la localización.
```

Procesa llama primero a la función **corrige_perspectiva2** que es el bloque que sigue, la entrada a esta función es la imagen de la placa distorsionada en escala de grises (zona) y la salida es la imagen corregida almacenada en una variable (corregida).




```
corregida = corrige_perspectiva2(zona);   % Puede usarse como función independiente
                                          % o ser reemplazada
```

A continuación viene la Binarización, la llamada es a la función **Binariza**, cuya entrada es la imagen en escala de grises (corregida) y su salida es la imagen en blanco y negro almacenada en un variable (ZonaBin).

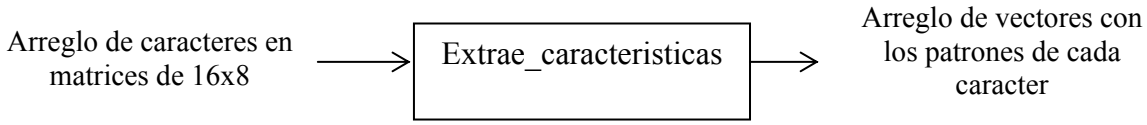



```
zonaBin = Binariza(corregida);          % Puede usarse como función independiente
                                        % o ser reemplazada
```

El bloque que sigue es la **segmentación**, esta función tiene como entradas la imagen de la placa corregida en B/N (zonaBin) lista para extraer los caracteres de la placa y una variable adicional (tt), donde se indica el número de veces que ha sido utilizada la función con una misma imagen, este parámetro sirve para que la función en caso de haber extraído sólo 5 caracteres se llame así mismo una segunda vez e intente extraer los 6, es por ello que la función debe iniciar con tt = 0 siempre. La salida a esta función es un arreglo de (6 o 7) matrices de 16x8 conteniendo los caracteres extraídos.

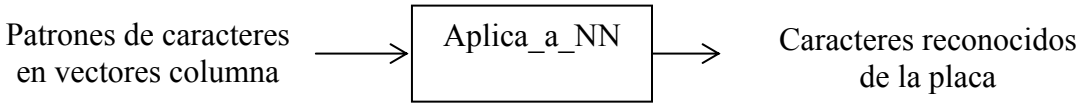

```
[CaractDec,estado] = SEGMENTACION(zonaBin,tt);
```

La extracción de características se realiza con la función **Extrae_características** la entrada es un arreglo de matrices (CaractDec) que contienen los caracteres extraídos, la salida es un arreglo de vectores columna conteniendo los patrones extraídos de cada carácter.



[Componentes_principales_placa,Ns] = Extrae_caracteristicas(CaractDec);

Finalmente teniendo los patrones en vectores columna en una sola matriz, se puede llamar a la función **Aplica_a_NN** que va a realizar el reconocimiento utilizando la red neuronal. La entrada es la matriz de componentes principales en vectores columna (Componentes_principales_placa), de todos los caracteres, además de un parámetro (Ns) que indica el número de caracteres a reconocer y la salida son los caracteres reconocidos.



[num_placa] = **Aplica_a_NN**(Componentes_principales_placa,Ns);

ANEXO B
BASE DE DATOS DE IMAGENES

Nombre	Número
IMG_0001	SQ-7028
los IMG_0002	BGR-774
IMG_0003	AM-3125
IMG_0004	QI-9849
IMG_0005	DQ-9218
IMG_0006	KO-1123
IMG_0007	TO-9878
IMG_0008	AGA-481
IMG_0009	LI-1752
IMG_0010	AA-1417
IMG_0011	AOV-386
IMG_0012	KO-2483
IMG_0013	RGB-685
IMG_0014	SGO-417
IMG_0015	CG-3077
IMG_0016	IQ-2542
IMG_0017	HI-2153
IMG_0018	PIE-307
IMG_0019	GG-8540
IMG_0020	BO-9028
IMG_0021	HO-9424
IMG_0022	FD-1274
IMG_0023	HO-2411
IMG_0024	SO-8444
IMG_0025	GG-8185
IMG_0026	AQ-5765
IMG_0027	AGB-515
IMG_0028	LQ-8719
IMG_0029	AO-2066
IMG_0030	BQE-449
IMG_0031	BI-1876
IMG_0032	AQV-382
IMG_0033	AQR-522
IMG_0034	JI-3333
IMG_0035	AO-4379
IMG_0036	FO-4798
IMG_0037	SGJ-704
IMG_0038	BGY-642
IMG_0039	SGJ-859
IMG_0040	AGE-651
IMG_0041	SO-9750
IMG_0042	TI-6337
IMG_0043	AQ-8425
IMG_0044	RH-9145
IMG_0045	SO-8444
IMG_0046	AQK-672
IMG_0047	LQ-5073
IMG_0048	PGP-233
IMG_0049	BGR-902
IMG_0050	AQK-146
IMG_0051	AQX-128
IMG_0052	AY-2773
IMG_0053	TI-8331
IMG_0054	JO-3585
IMG_0055	RG-3814
IMG_0056	CQ-6660
IMG_0057	OQ-7597
IMG_0058	AON-054
IMG_0059	RQ-8673
IMG_0060	CQ-4118
IMG_0061	CF-1758
IMG_0062	DO-6750
IMG_0063	AQX-128
IMG_0064	BIB-335
IMG_0065	PGN-513
IMG_0066	AOD-100
IMG_0067	BQA-083
IMG_0068	AGX-965
IMG_0069	OO-6085
IMG_0070	SGC-644
IMG_0071	CH-7460
IMG_0072	AIQ-432
IMG_0073	TI-4327
IMG_0074	SQI-888
IMG_0075	AQY-778
IMG_0076	AGN-532
IMG_0077	HO-8105
IMG_0078	CG-1377
IMG_0079	AE-5855
IMG_0080	AGQ-794
IMG_0081	IO-6680
IMG_0082	FQ-8442
IMG_0083	RB-3100
IMG_0084	OD-2288
IMG_0085	KI-2271
IMG_0086	BIF-615
IMG_0087	DO-8978
IMG_0088	LO-2874
IMG_0089	BGF-694
IMG_0090	CO-3584
IMG_0091	FQ-6381
IMG_0092	SGH-860
IMG_0093	LO-5501
IMG_0094	RIR-018
IMG_0095	FI-1410
IMG_0096	DO-7323
IMG_0097	CP-3859
IMG_0098	SGE-865
IMG_0099	AGC-813
IMG_0100	IO-8860
IMG_0101	AI-8545
IMG_0102	AIJ-731
IMG_0103	BGD-712
IMG_0104	AOV-026
IMG_0105	KO-5053
IMG_0106	SK-3583
IMG_0107	AQD-533
IMG_0108	JQ-7307
IMG_0109	AIL-420
IMG_0110	PGB-623
IMG_0111	BIV-215
IMG_0112	AOO-046
IMG_0113	GO-2759
IMG_0114	BQH-873
IMG_0115	IQ-5433
IMG_0116	KI-6683
IMG_0117	SO-4355
IMG_0118	LQ-5327
IMG_0119	SIC-462
IMG_0120	RH-3064
IMG_0121	SGT-713
IMG_0122	BO-9500
IMG_0123	SGR-311
IMG_0124	BQN-603
IMG_0125	CU-2334
IMG_0126	TE-1462
IMG_0127	CH-5241
IMG_0128	RIR-173
IMG_0129	FO-7538

IMG_0130	CQ-8699
IMG_0131	BO-6466
IMG_0132	GQ-5526
IMG_0133	TG-1305
IMG_0134	BGC-196
IMG_0135	OO-1118
IMG_0136	LO-6970
IMG_0137	LO-3182
IMG_0138	LO-8456
IMG_0139	SIA-184
IMG_0140	AOA-544
IMG_0141	SQO-634
IMG_0142	FQ-6381
IMG_0143	AGB-455
IMG_0144	BK-5259
IMG_0145	KO-5798

ANEXO C
CÁMARA POWERSHOT A10 ESPECIFICACIONES TÉCNICAS

General	
Product type	Digital photo camera
Digital zoom	2 x
Sensor resolution (effective)	1,320,000 pixels
Sensor resolution (gross)	1,320,000 pixels
Optical sensor size	1/2.7"
Optical sensor type	CCD
Light sensitivity	ISO 100, ISO 150
Still image format	JPEG
Color support	Color
Min shutter speed	1 sec
Max shutter speed	1/1500 sec
Continuous shooting speed	2.5 frames per second
White balance	Automatic, manual
White balance presets	Fine, fluorescent, cloudy, tungsten light
Lens System	
Lens aperture	F/2.8-4.8
Optical zoom	2.5 x
Lens type	Zoom lens
Auto focus	TTL phase detection
Focal length equivalent to 35mm camera	35 - 105mm
Min focal length	5.4 Mm
Max focal length	16.2 Mm
Focus adjustment	Automatic

Min focus range	29.9 in
Macro focus range	16-76cm
Filter size	52 mm
Zoom adjustment	Automatic

Flash	
Camera flash type	Built-in flash
Effective flash range	10 in - 14 ft
Flash modes	Flash ON mode, auto mode, flash OFF mode, red-eye reduction

Exposure	
Exposure modes	Automatic, manual
Exposure metering	Evaluative
Exposure compensation	±2 EV range, in 1/3 EV steps

Memory / Storage	
Media type	CompactFlash
Memory included	1 x 8 MB flash - CompactFlash Card
Image storage	Fine 1280 x 960 : 10 - with 8MB card, Normal 1280 x 960 : 16 - with 8MB card, Basic 1280 x 960 : 32 - with 8MB card, Fine 1024 x 768 : 16 - with 8MB card, Normal 1024 x 768 : 24 - with 8MB card

Viewfinder / Display	
Display type	LCD display - TFT active matrix 1.5" - color
Diagonal size	1.5"
Technology	TFT active matrix
Display form factor	Built-in
Display resolution	117,600 pixels
Optical viewfinder type	Real-image zoom

Expansion / Connectivity	
Connector type	1 x USB
Expansion slot(s)	1 x CompactFlash Card - type I
Cable(s) included	1 x video cable - external

Additional Features	
Additional features	Date/time stamp, DPOF support, AE/FE lock
Software included	Drivers & Utilities
Included accessories	Hand strap

Physical Characteristics	
Battery	4 x battery - AA type - alkaline
Depth	1.5 in
Height	2.8 in
Width	4.3 in
Weight	8.8 oz

ANEXO D
CÁMARA INFRARROJA AVC 307R ESPECIFICACIONES TECNICAS

Model	AVC307R
Pick up Element	1/3 inch Samsung B/W CCD image sensor
Number of Pixel	510(H) x 492(V)(EIA) / 500(H) x 582(V)(CCIR)
Resolution	380 TV lines
Min. Illumination	0.5 Lux / F2.0; 0 Lux (10 meter IR ON)
S/N Ratio	More than 48dB (AGC off)
Electronic Shutter	1/60(1/50) to 1/100,000 sec.
Gamma Correction	0.45
AGC	Yes
Standard Board Lens	F4.0mm / F2.0
Lens Angle	80¢X
Video Output	1.0 Vp-p composite, 75£[
Power Source	DC12V¡Ó10%
Current Consumption	120 mA (IR OFF), 280 mA (IR ON)
Water Resistance	IP 57
IR LED	12 units Infrared LED
Effective Range	Over 10 Meters
Dimension (mm)	64.6 (£r) x 105 (L)
Weight	550 g

ANEXO E
EXTRACCIÓN DE BORDES

Un problema de importancia fundamental en el análisis de imagen es la detección de bordes. Los contornos caracterizan las fronteras de los objetos, y por tanto son de gran utilidad de cara a la segmentación e identificación de objetos en escenas. Los puntos de contorno son como zonas de píxel en las que existe un cambio brusco de nivel de gris. Si pensamos en una imagen como una función continua f(x,y), vemos que su derivada tiene un máximo local en la dirección del contorno. Por esto las técnicas más usadas en la detección de contornos se basen en la medida del gradiente de f a lo largo de r en una dirección θ (fig.1).

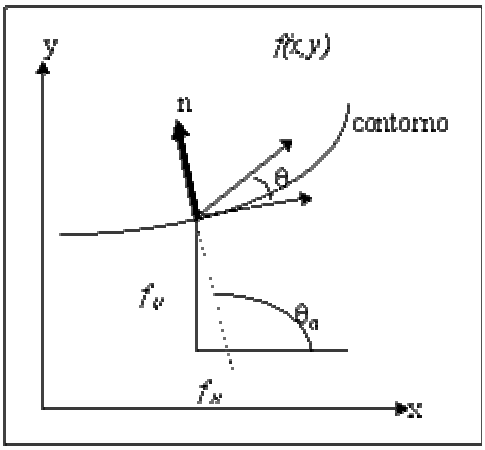


Fig. 1. Gradiente de f(x,y)

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} = f_x \cos\theta + f_y \sin\theta$$

El máximo valor de $\frac{\partial f}{\partial r}$ se obtiene cuando $\frac{\partial}{\partial\theta}\frac{\partial f}{\partial r} = 0$

$$-f_x \sin\theta_g + f_y \cos\theta_g = 0 \Rightarrow \theta_g = \tan^{-1}\left(\frac{f_y}{f_x}\right)$$
$$\left(\frac{\partial f}{\partial r}\right)_{max} = \sqrt{f_x^2 + f_{xy}^2}$$

Donde θg es la dirección del contorno.

En el tratamiento de imágenes, se trabaja con píxeles, y en un ambiente discreto. Basándose en los conceptos vistos anteriormente, se han ideado operadores detectores de contornos basados en máscaras. que representan aproximaciones en diferencias finitas de los gradientes ortogonales f_x, f_y .

Si llamamos H a una máscara de tamaño p x p, se define el producto interno con una imagen U, en una poición (m,n) como:

$$\langle U, H \rangle_{m,n} = \sum_i \sum_j h(i,j)u(i+m, j+m) = u(m,n) \otimes h(-m,-n)$$

OPERADORES GRADIENTE

La forma habitual de establecer el gradiente de una imagen u(m,n) en un punto dado, es mediante el producto de la imagen por dos máscaras H1, H2 que representan la magnitud del gradiente en dos direcciones perpendiculares:

$$g_1(m,n) = \langle U, H_1 \rangle_{m,n} \quad g(m,n) = \sqrt{g_1^2(m,n) + g_2^2(m,n)}$$
$$g_2(m,n) = \langle U, H_2 \rangle_{m,n} \quad \theta_g(m,n) = tan^{-1}\frac{g_2(m,n)}{g_1(m,n)}$$

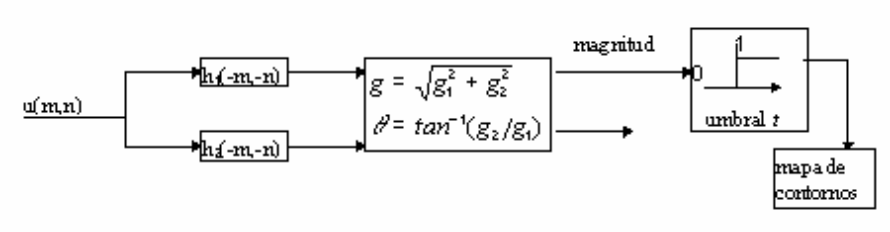


Fig. 2. Detección de bordes vía operadores gradiente

Por razones computacionales, casi siempre, la magnitud del gradiente se calcula como:

$$g(m,n) = |g_1(m,n)| + |g_2(m,n)|$$

Un píxel es declarado como perteneciente a un borde cuando g(m,n) excede un determinado valor umbral t. En función de este valor t, se tendrá un mayor o menor número de puntos de gradiente. Habitualmente se suele escoger este valor en función del histograma acumulado de g(m,n) de forma que sólo del 5 al 10% de los puntos de máximo gradiente sea declarado como borde.

Son famosas una serie de máscaras por su uso frecuente en análisis de imagen, sobre todo las de Prewitt o Sobel por su ventaja computacional.

Tabla 1.1. Operadores gradiente más comunes	**Dirección horizontal**	**Dirección vertical**
a) Roberts	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
b) Smoothed (o Prewitt)	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
c) Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
d) Isotrópico	$\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$

ANEXO F

APUNTES DE MORFOLOGÍA

Extraído del libro R.C. González y R.e. Wood, "Digital Image Processing" Addison-Wesley Publishing Company, Inc., 1993.

1. Operaciones básicas sobre conjuntos

Sean A y B son conjuntos en un n-espacio E^n con elementos $a = (a_1,..., a_n)$ y $b = (b_1,..., b_n)$ respectivamente siendo ambos n-tuplas.

1. La traslación de A por $x \in E^n$ que se nota A_x se define como

$$A_x = \{c | \ c = a + x, \text{ para algún } a \in A\}.$$

2. La reflexión de B notada $\hat{B}$ se define como

$$\hat{B} = \{x | \ x = -b, \text{ para algún } b \in B\}.$$

3. Por último la diferencia de dos conjuntos A y B, notada $A - B$, se define mediante

$$A - B = \{x | \ x \in A, x \notin B\}$$

2. Morfología binaria

El lenguaje de la morfología matemática binaria es el de la teoría de conjuntos. Los conjuntos en morfología matemática representan las formas presentes en imágenes binarias o de niveles de gris. El conjunto de todos los píxeles blancos en una imagen en blanco y negro (binaria) constituye una descripción completa de la imagen.

Los puntos en un conjunto sobre los que se aplica la transformación son el conjunto de puntos seleccionado y el complementario el no seleccionado. En las imágenes binarias los puntos seleccionados son los que no pertenecen al fondo.

Las operaciones primarias morfológicas son la erosión y la dilatación. A partir de ellas podemos componer las operaciones de apertura y clausura. Son estas dos operaciones las que tienen mucha relación con la representación de formas, la descomposición y la extracción de primitivas.

2.1. Dilatación binaria

La dilatación es la transformación morfológica que combina dos vectores utilizando la suma. La dilatación binaria fue usada primero por Minkowski, y en la literatura

matemática recibe el nombre de suma de Minkowski. Si A y B son conjuntos en un n-espacio E^n con elementos $a = (a_1,..., a_n)$ y $b = (b_1,..., b_n)$, respectivamente, siendo ambos n-tuplas, entonces la dilatación de A por B es el conjunto de todos los posibles vectores que son suma de pares de elementos, uno de A y otro de B.

Más formalmente, la dilatación de A por B se nota $A \oplus B$ y se define mediante

$$A \oplus B = \{c \in E^n | \; c = a + b \text{ para algún } a \in A \text{ y } b \in B\}$$

al ser la suma conmutativa, la dilatación también lo es: $A \oplus B = B \oplus A$.

Se puede probar que las siguientes definiciones de la dilatación son equivalentes

$$A \oplus B = \{x|(\square)_x \square A \neq \varnothing\} = \square A_b$$

En la práctica los conjuntos A y B no son simétricos. El primer elemento de la dilatación, A, está asociado con la imagen que se está procesando y el segundo recibe el nombre de elemento estructural, la forma que actúa sobre A en la dilatación para producir $A \oplus B$.

Cuando se realiza una dilatación con un elemento estructural que contiene el cero, lo que realizamos es la expansión de una imagen y es fácil pensar en una implementación paralela. Algunos ejemplos se muestran en las figuras 1 , 2 . Es importante tener en cuenta que el sistema de coordenadas que se usará en este tema es (fila,columna).

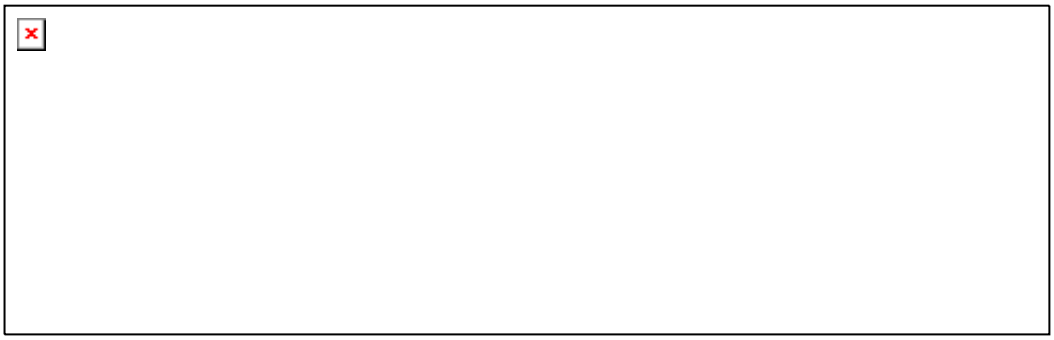


Figura 1: Ejemplo de dilatación. (a) Elemento estructural, *B*. (b) Imagen, *A*. (c) Resultado de la dilatación $A \oplus B$

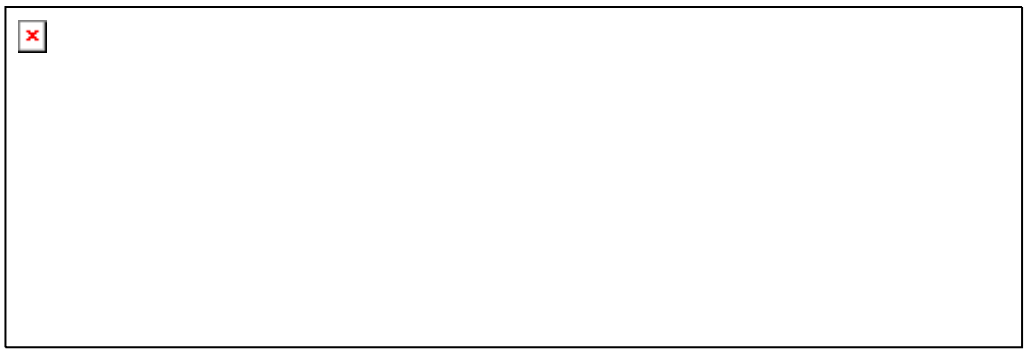


Figura 2: Otro ejemplo de dilatación. (a) Elemento estructural, *B*. (b) Imagen, *A* . (c) Resultado de la dilatación $A \oplus B$

La dilatación tiene las siguientes propiedades

1. La dilatación por el trasladado de un elemento estructural es el trasladado de la dilatación:

   $A \oplus B_t = (A \oplus B)_t$.

2. Propiedad distributiva:

   $A \oplus (B \cup C) = (A \oplus B) \cup (A \oplus C)$.

3. Asociatividad (iteración):

   $A \oplus (B \oplus C) = (A \oplus B) \oplus C$.

4. Crecimiento:

   $A \subseteq B \cup A \oplus K \subseteq B \oplus K$ *1$cm \cup K$.

2.2. Erosión binaria

La erosión es la operación morfológica dual, un concepto que definiremos formalmente a continuación, de la dilatación. Es la transformación morfológica que combina dos conjuntos usando el concepto de inclusión. Si A y B son conjuntos en el espacio euclídeo n-dimensional, entonces la erosión de A por B es el conjunto de todos los elementos x para los que $x + b \in A$ para todo $b \in B$. La sustracción de Minkowski está muy relacionada con la erosión.

La erosión de A por B se nota $A \cup B$ y su definición es

$$A \cup B = \{x \in E^n | \, x + b \in A \text{ para todo } b \in B\}$$

un ejemplo de erosión se encuentra en la figura 3 .

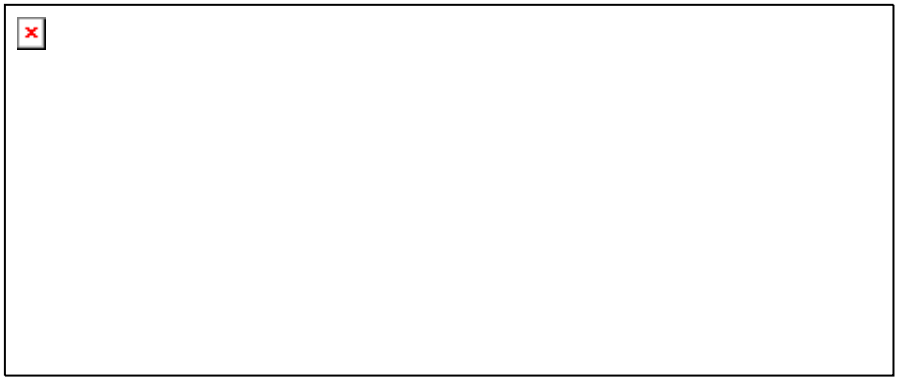


Figura 3: Ejemplo de erosión. (a) Elemento estructural, *B*. (b) Imagen, *A*. (c) Resultado de la erosión $A \ominus B$

La utilidad de la erosión puede apreciarse mejor cuando ésta se expresa de forma diferente. La erosión de una imagen, A , por un elemento estructural, B, es el conjunto de todos los elementos $x \in E^n$ para los cuales B trasladado por x está contenido en A. La demostración es inmediata y se tiene

$$A \ominus B = \{x \in E^n | B_x \subseteq A\}$$

Mientras que la dilatación puede representarse como la unión de los trasladados, la erosión puede representarse como la intersección de los trasladados negativos.

$$A \ominus B = \bigcap_{b \in B} A_{-b}$$

La erosión se concibe usualmente como una reducción de la imagen original. En términos de teoría de conjuntos, el conjunto erosionado se suele pensar que siempre está contenido en el original. Una transformación que cumple esta propiedad se dice antiextensiva. Sin embargo, la erosión es antiextensiva cuando el origen pertenece al elemento estructural. Esto es, si $0 \in B$, entonces $A \ominus B \subseteq A$, la demostración de esta propiedad, al igual que de las anteriores, es obvia.

Al igual que la dilatación, la erosión es también creciente: si $A \subseteq B$ entonces $A \ominus K \subseteq B \ominus K$. Además la erosión por un elemento estructural mayor produce un resultado menor si $K \subseteq L$ entonces $A \ominus L \subseteq A \ominus K$.

La dilatación y la erosión son muy similares en el sentido de que lo que uno hace al objeto el otro lo hace al fondo. Esta relación puede formularse como una relación de dualidad.

Dos operadores son duales cuando la negación de una formulación empleado en el primero es igual a la misma formulación empleando el segundo operador en la variable

negada. Un ejemplo es la ley de DeMorgan que establece la dualidad entre la unión y la intersección $(A \cup B)^c = A^c \cap B^c$, siendo la negación el complementario.

En morfología la negación de un conjunto puede ocurrir de dos formas diferentes: en un sentido lógico, siendo la negación la complementación o en un sentido geométrico, siendo la negación el cambio de la orientación de un conjunto con respecto a los ejes de coordenadas. Esta negación recibe el nombre de reflexión.

Sea $B \subseteq E^n$, como ya sabemos, la reflexión de B se nota y se define mediante

$$= \{x | \text{ para algún } b \in B, x = -b\}$$

Como se ve en el siguiente teorema, el complemento de la erosión es la dilatación de un complemento por reflexión. La dualidad de dilatación y erosión usa negaciones lógicas y geométricas por los diferentes papeles jugados por la imagen y el elemento estructural.

Teorema . Dualidad de la erosión y la dilatación.

$$(A \cap B)^c = A^c \oplus$$

Como corolario tenemos

Corolario

$$(A \oplus B)^c = A^c$$

Algunas de las propiedades de la erosión se resumen en la lista siguiente

1. Propiedad Distributiva:

$$A \cap (K \cap L) = (A \cap K) \cap (A \cap L)$$

2. Localización:

$$(A \cap B) \cap K = (A \cap K) \cap (B \cap K)$$

Finalmente, con respecto a la descomposición de elementos estructurales, una regla de la cadena para la erosión se verifica cuando el elemento estructural se puede descomponer mediante dilatación.

$$A \cap (B \oplus C) = (A \cap B) \cap C$$

ANEXO G
ACONDICIONAMIENTO DEL SISTEMA CON CÁMARA DE VIDEO

En esta parte el objetivo es encontrar a que distancia se debe colocar una cámara 2 para obtener la misma cantidad de píxeles, dada una posición conocida para una cámara 1 que enfoca un objeto. Para ello debemos tener en cuenta los parámetros de las cámaras que intervienen **[3]**.

Para comenzar en la parte exterior donde se encuentra la imagen real del objeto tenemos una región que va a tener una representación en píxeles en la imagen digital esto viene a ser que la altura total (h) que pueda abarcar la cámara le asignará 480 píxeles de altura y 640 píxeles a todo el ancho (w), esto a toda el área que abarque Fig. 1.

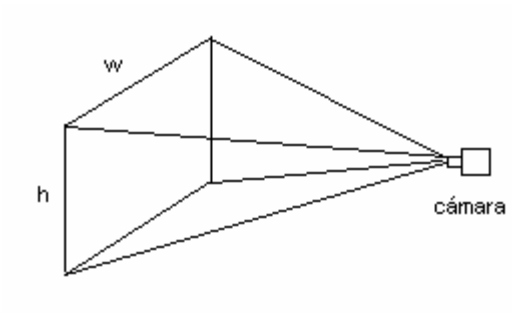


Figura 1.

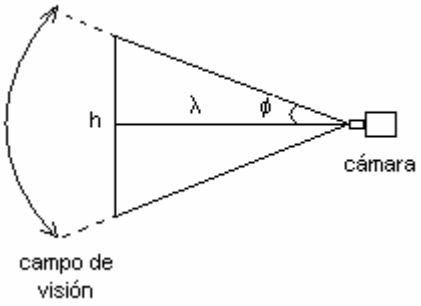


Figura 2.

En el plano sería como la figura 2, aquí observamos el ángulo de visión que tiene la cámara y las distancias involucradas, guardando sus proporciones podemos hacer la siguiente relación:

$$h = 2\lambda\, tg\phi \qquad (1)$$

Por otro lado en la parte interna de la cámara tenemos un parámetro denominado Field of View (FOV) o "Campo de Visión" y viene a ser el ángulo que ilumina el sensor CCD o elemento que capta la imagen (película) teniendo como punto de partida la lente de la cámara Fig.3.

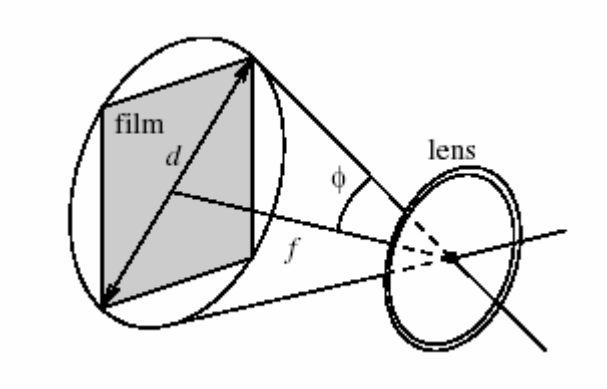


Figura 3. Campo de visión de una cámara (FOV)

Field of View = 2ϕ

De la figura fácilmente se tiene la siguiente definición:

$$\phi = Arctg\left(\frac{d}{2f}\right) \qquad (2)$$

Donde:

d: diámetro del sensor (film o CCD chip)

f: longitud focal de la cámara

A partir de estas dos relaciones podemos ver que se corresponden pues en la parte externa la cámara tiene un ángulo máximo de visión, éste ángulo precisamente es el campo de visión al cual siempre la cámara digital le va asignar la misma cantidad de píxeles, esto es importante para mantener el número de píxeles de la imagen y no trabajar con otra resolución sobre un mismo objeto en el procesamiento.

En la parte interna igualmente la luz que ingresa va a cubrir una determinada área ya definida y precisamente aquí es donde se define la imagen, sobre el sensor.

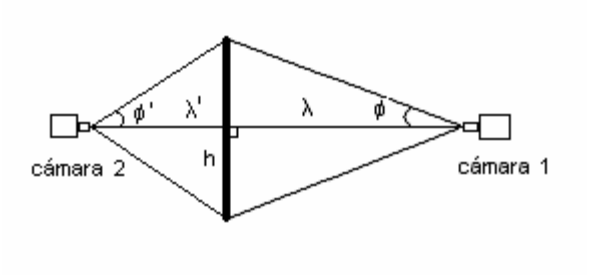


Figura 4. Dos cámaras de distintas características enfocando un mismo objeto.

De la relación 1 podemos decir que para mantener la misma cantidad de píxeles sobre un objeto de longitud h con 2 cámaras diferentes a distancias diferentes debe cumplirse:

$$h = 2\lambda Tg\phi = 2\lambda' Tg\phi' \qquad (3)$$

De las dos últimas despejamos la nueva distancia a la que debería ubicarse una segunda cámara para obtener el mismo resultado:

$$\lambda' = \lambda\left(\frac{Tg\phi}{Tg\phi'}\right) \qquad (4)$$

Utilizando la definición 2 llegamos a la siguiente relación,

$$\lambda' = \lambda\left(\frac{d}{d'}\right)\left(\frac{f'}{f}\right) \qquad (5)$$

Donde d, d', f y f' son los parámetros intrínsecos (Fig. 3) de las cámaras 1 y 2 (Fig. 4).

Tomando los valores de la primera cámara digital PowerShot A10 (cámara 1), Anexo C:

Diámetro del sensor: d = (1/2.7) "

Longitud focal de la cámara: f = 16mm

Considerando una distancia promedio al vehículo: λ = 3.5m

Para la nueva cámara Infrarroja a adaptar (cámara 2), Anexo D:

CCD image sensor: d' = 1/3 "

Longitud focal: f ' ≈ 4mm

Reemplazando estos valores en la ecuación (5):

$$d = 0.9722m$$

Quiere decir que colocando la **cámara infrarroja** aproximadamente a 97.2cm se va a obtener la misma cantidad de píxeles del mismo objeto que usando la **cámara digital** a 3.5m.

ANEXO H
INTERFAZ GRÁFICA CON CÁMARA DE VIDEO EN MATLAB

Elaboración de la Interfaz utilizando GUIDE

Matlab nos permite crear fácilmente interfases gráficas de usuario con objetos como ventanas y botones que con drag and drop se van ordenando y aplicando propiedades, así como dándoles las funciones que actuarán ante un evento determinado, como un clic en un botón. En la figura 1 podemos apreciar la forma de cómo se construye esta interfase.

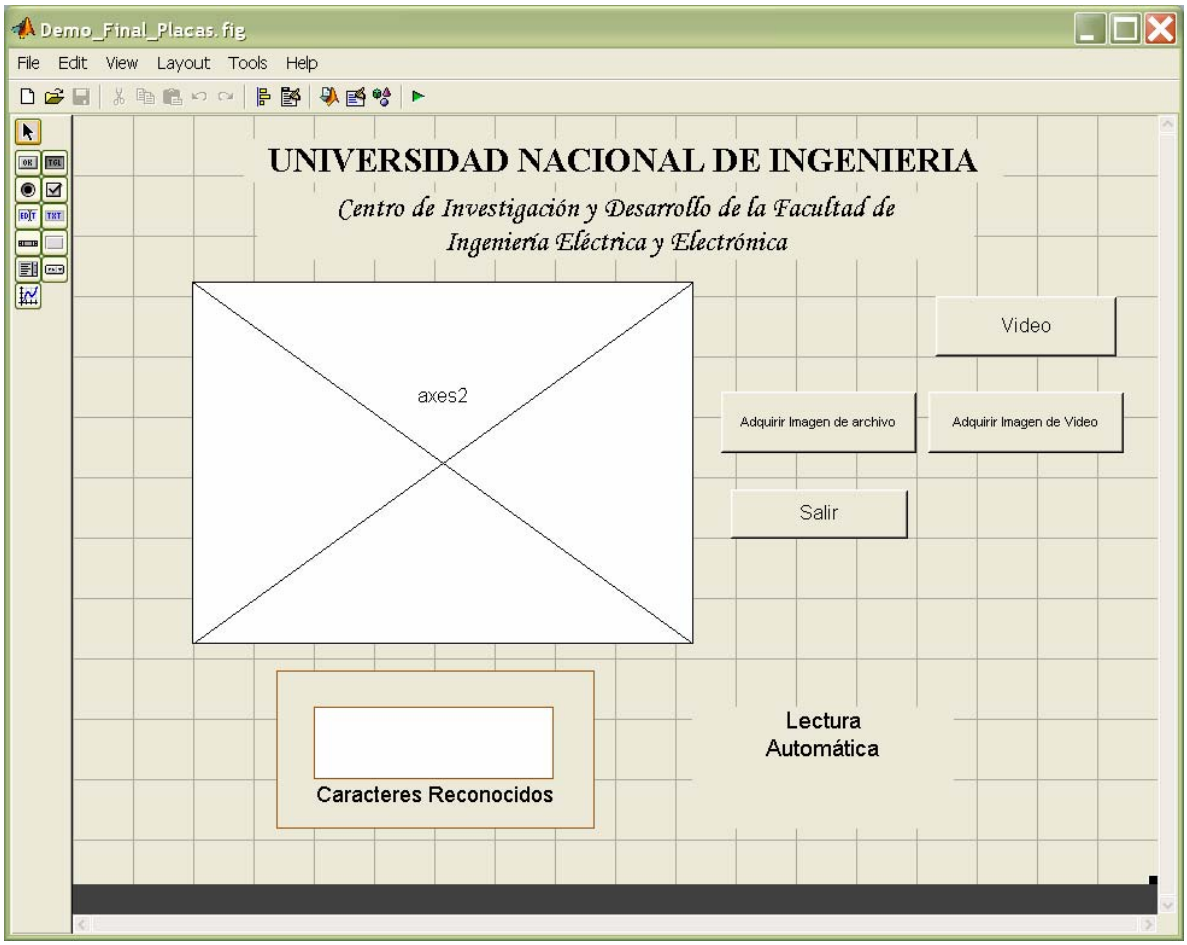


Figura 1. Entorno gráfico de la Herramienta GUIDE del Matlab

Por otro lado para la utilización de la cámara de video se utilizó Video for Matlab (VFM), éste fue un proyecto realizado en el exterior, que permite a los usuarios de Matlab capturar una secuencia de imágenes desde una tarjeta de captura de video conectada a una video cámara. Video for Matlab es distribuido bajo el GNU General Public License. Utilizando esta librería podemos hacer la captura de video, que al invocarla, directamente nos comunica con los drivers de la cámara (tarjeta) o nos permite cambiar entre las fuentes de video instaladas (cámara de video y webcam).

La vista inicial se muestra en la figura 2, aquí se agregaron dos botones mas a la interfaz original, el primero (**Video**) es para escoger la fuente de video, que como se puede apreciar en la figura 3, nos envía a los drivers correspondientes de la tarjeta de captura por defecto, aquí también se puede modificar el brillo o contraste manualmente. En casos haya una webcam instalada podríamos elegir cambiar a la webcam, aunque lo recomendable es trabajar con la cámara B/N originalmente adquirida para el proceso por su mejor resolución y mayor estabilidad en cuanto a los cambios de iluminación **[3]**.

En la figura 4 se muestra la pantalla con la fuente de video, que arroja la librería VFM (Video for Matlab). Se puede ver que en la barra de menú la opción **Driver** permite el cambio a alguna otra tarjeta de captura de video (como puede ser la webcam).

Figura 2. Interfaz gráfica del Sistema en Matlab

Figura 3. Ventana de configuración de la fuente de video

Figura 4. Desde esta ventana se puede monitorear permanentemente

Aquí podemos monitorear permanentemente el video en tiempo real, de manera que cuando se necesite adquirir una imagen se procede a accionar el segundo botón (figura 5: **Adquirir Imagen de Video**). Con ello se captura un solo cuadro y el sistema trabaja de inmediato con la imagen adquirida.

En el ejemplo mostrado dada una imagen algo complicada, el sistema consigue extraer los caracteres para su reconocimiento. Como se aprecia en la figura 6.

Figura 5. Luego de adquirir un cuadro desde la fuente de video

Figura.6. Caracteres extraídos automáticamente desde la fuente de video por el sistema

ANEXO I
CÓDIGO FUENTE DE LA CAPTURA DE VIDEO Y LA INTERFAZ GRÁFICA

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
UNIVERSIDAD NACIONAL DE INGENIERIA                                              %%
%%    CENTRO DE INVESTIGACION Y DESARROLLO DE LA FACULTAD DE INGENIERIA         %%
%%    ELECTRICA Y ELECTRONICA                                                   %%
%%                                                                              %%
%%    LIMA - PERU                                                               %%
%%                                                                              %%
%%                                                                              %%
%%    INTERFAZ GRAFICA PARA LA ADQUISICION DE LA IMAGEN.                        %%
%%                                                                              %%
%%     Sirve para mostrar una interfaz de usuario adquirir la imagen y          %%
%%     acceder a la funcion de mayor jeraquia del sistema: RECONOCEDOR_FINAL    %%
%%                                                                              %%
%%                                                                              %%
%%     Funcion pushbutton1_Callback   - Adquiere Imagen de Archivo              %%
%%         Contiene rutinas para la adquisicion de imagenes de archivo          %%
%%                                                                              %%
%%     Funcion togglebutton1_Callback - Configura la fuente de video            %%
%%                                                                              %%
%%     Funcion pushbutton3_Callback   - Adquiere Imagen de Video                %%
%%         Para la captura de imagen proveniente de video es necesario          %%
%%         configurar previamente la fuente de video con el boton VIDEO         %%
%%                                                                              %%
%%     Reconocedor_Final - Funcion que engloba todos los bloques del sistema.   %%
%%         Su entrada es una matriz que contiene la imagen RGB (640 X 480)      %%
%%         y la salida es la serie de caracteres reconocidos por la red         %%
%%         neuronal. Esta  función se usa luego de adquirir la imagen que       %%
%%         procede de un archivo o video.                                       %%
%%                                                                              %%
%%                                                                              %%
%%   ASESOR: DR. JORGE ALBERTO DEL CARPIO SALINAS    jdelcarpio@uni.edu.pe      %%
%%   GRADUANDO:    GIANCARLO AYALA CHARCA     giancarlo_ayala@dacafe.com        %%
%%                                                                              %%
%%                                                                              %%
%%                                                                              %%
%%    Matlab Version 6.5                                                        %%
%%    Image Processing Toolbox   Version 2.2 (R.11) o Superior                  %%
%%                                                                              %%
%%                                      Enero 2004                              %%
%%                                                                              %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function varargout = Demo_Final_Placas(varargin)
% DEMO_FINAL_PLACAS M-file for Demo_Final_Placas.fig
%      DEMO_FINAL_PLACAS, by itself, creates a new DEMO_FINAL_PLACAS or raises the existing
%      singleton*.
```

```
%
%     H = DEMO_FINAL_PLACAS returns the handle to a new DEMO_FINAL_PLACAS or the handle to
%     the existing singleton*.
%
%     DEMO_FINAL_PLACAS('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in DEMO_FINAL_PLACAS.M with the given input arguments.
%
%     DEMO_FINAL_PLACAS('Property','Value',...) creates a new DEMO_FINAL_PLACAS or raises the
%     existing singleton*.  Starting from the left, property value pairs are
%     applied to the GUI before Demo_Final_Placas_OpeningFunction gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop.  All inputs are passed to Demo_Final_Placas_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Demo_Final_Placas

% Last Modified by GUIDE v2.5 24-Oct-2003 16:50:29


% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Demo_Final_Placas_OpeningFcn, ...
                   'gui_OutputFcn',  @Demo_Final_Placas_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Demo_Final_Placas is made visible.
function Demo_Final_Placas_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Demo_Final_Placas (see VARARGIN)

% Choose default command line output for Demo_Final_Placas
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Demo_Final_Placas wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = Demo_Final_Placas_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%    BOTON ADQUIR IMAGEN DE ARCHIVO     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% El sgte. codigo abre un file y arroja una cadena
  [nombre,ruta]=uigetfile('*.bmp;*.jpg','Abrir Archivos de Imagen',0,0);
  nombre1=[ruta nombre];
  nombre=setstr(lower(nombre1)); %Esta variable 'nombre', tiene el
                               %nombre del archivo en una cadena
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Aqui se lee el archivo y se muestra la imagen
  ii=imread(nombre);
  imshow(ii)
  set(handles.axes2,'XMinorTick','on')   %set(handles.figure1,)
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Aplicacion del Reconocimiento
%%% la salida es el numero de la placa
  num_placa=Reconocedor_Final(ii)
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Mostramos los caracteres
  set(handles.text4,'string',num_placa')
%%%



% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.figure1)



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
%%%%%%%%%%     BOTON ADQUIRIR IMAGEN DE VIDEO     %%%%%%%%%%%%%%%%%%
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% El sgte. codigo captura una imagen desde
%%% una camara de video y muestra la imagen
  ii=vfm('grab',1);
  set(handles.text4,'string','')
  imshow(ii)
  set(handles.axes2,'XMinorTick','on')       %set(handles.figure1,)
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Aplicacion del reconocimento de los
%%% caracteres. La salida es el numero de la placa
  num_placa=Reconocedor_Final(ii)
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Mostramos los caracteres
  set(handles.text4,'string',num_placa')
%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%%%%%%%%%     BOTON CONFIGURAR LA FUENTE DE VIDEO     %%%%%%%%%%%%%%
% --- Executes on button press in togglebutton1.
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%% BOTON DE CAPTURA DE VIDEO
%%% Se utiliza la libreria VFM
button_state = get(hObject,'Value');
if button_state == get(hObject,'Max')
% toggle button is pressed
  vfm('configsource')  % esta opcion configura la fuente de video
  vfm('preview',1)
  vfm('show',1)        % muestra el video proveniente de una camara en una ventana

elseif button_state == get(hObject,'Min')
% toggle button is not pressed
  vfm('preview',0)     % oculta la ventana de video
  vfm('show',0)

end
```

ANEXO J
CÓDIGO FUENTE DEL PRE-PROCESADO Y LA LOCALIZACIÓN DE LA PLACA

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   UNIVERSIDAD NACIONAL DE INGENIERIA                                    %%
%%   CENTRO DE INVESTIGACION Y DESARROLLO DE LA FACULTAD DE INGENIERIA     %%
%%   ELECTRICA Y ELECTRONICA                                               %%
%%                                                                         %%
%%   LIMA - PERU                                                           %%
%%                                                                         %%
%%         [num_placa] = Reconocedor_Final(ii)                             %%
%%                                                                         %%
%%     FUNCION QUE ENGLOBA TODOS LOS BLOQUES DEL SISTEMA                   %%
%%                                                                         %%
%%     La entrada de esta funcion es una matriz (ii) que contiene una      %%
%%     imagen a color en RGB (640x480) y la salida es la serie de          %%
%%     caracteres de la placa. Esta funcion asi tambien efectua una serie  %%
%%     de condicionales para procesar primero la zona con mayor            %%
%%     probabilidad de ser placa, en caso de no serlo procesa la segunda.  %%
%%                                                                         %%
%%                                                                         %%
%%     Localizacion - Funcion que efectua la Localizacion de la placa      %%
%%         dentro de una imagen RGB de 640 x 480 de Resolucion.            %%
%%         La entrada es la matriz que contiene una imagen cargada en      %%
%%         memoria (puede ser con el comando 'imread'). Su salida es       %%
%%         una o dos matrices que contienen posibles placas, sus áreas y   %%
%%         un indicador (CONTADOR) de cuantas placas efectivas encontró.   %%
%%                                                                         %%
%%     Procesa - Funcion que abarca todo el resto de bloques del sistema   %%
%%         a partir de la corrección de perspectiva hasta finalmente la    %%
%%         identificación de los caracteres de la placa. Cada bloque a su  %%
%%         vez es un modulo separable. La entrada de esta función es       %%
%%         matriz (zona1 o zona2) que encierre una placa de automóvil y    %%
%%         un parámetro adicional: su área (el producto de sus dimensiones) %%
%%         La salida de la función es la secuencia de caracteres y un      %%
%%         indicador de 'estado' de si efectuó bien el trabajo o no        %%
%%                                                                         %%
%%                                                                         %%
%%    ASESOR: DR. JORGE ALBERTO DEL CARPIO SALINAS    jdelcarpio@uni.edu.pe   %%
%%    GRADUANDO:    GIANCARLO AYALA CHARCA     giancarlo_ayala@dacafe.com     %%
%%                                                                         %%
%%                                                                         %%
%%                                                                         %%
%%   Matlab Version 6.5                                                    %%
%%   Image Processing Toolbox   Version 2.2 (R.11) o Superior              %%
%%                                                                         %%
%%                                Enero 2004                               %%
%%                                                                         %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [num_placa] = Reconocedor_Final(ii)
%--> clear
%  profile on %-detail builtin
%--> ii=imread('IMG_0028.JPG');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Este bloque halla una o dos zonas con posibles placas %%%
[zona1,zona2,area1,area2,CONTADOR]=Localizacion(ii);
        %Da como resultado una o dos matrices: zona1 o zona2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% A CONTINUACION, LA REGLA DE DECISION PARA TRATAR UNA O DOS PLACAS.
%% ESTADO ES UN REGISTRO QUE MUESTRA SI RECONOCIO O NO ('1' o '0') LOS CARACTERES
if CONTADOR==1,
   [num_placa,estado]=Procesa(zona1,area1); %Hay una sola placa
%       area_aux=area1;
%       zona=zona1; Procesa;  %mientras para las pruebas
else
   if area1>area2,          %La mas probable es la de mayor area
      [num_placa,estado]=Procesa(zona1,area1);  %DEBE SER FUNCION DEL AREA
                                          %PARA TENER REFERENCIA!
%       area_aux=area1;
%       zona=zona1; Procesa;  %mientras para las pruebas
      if estado==0,
         fprintf(' "FALLO EL PRIMER BLOQUE" ')
      [num_placa,estado]=Procesa(zona2,area2);
%          area_aux=area2;
%          zona=zona2; Procesa;
      end
   else
      [num_placa,estado]=Procesa(zona2,area2);
%       area_aux=area2;
%       zona=zona2; Procesa;  %Para pruebas, 'PROCESA' de NO como funcion
      if estado==0,
         fprintf(' "FALLO EL PRIMER BLOQUE" ')
      [num_placa,estado]=Procesa(zona1,area1);
%       area_aux=area1;
%       zona=zona1; Procesa;
      end
   end
end;
%   profile report %viewer
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   UNIVERSIDAD NACIONAL DE INGENIERIA                                                    %%
%%   CENTRO DE INVESTIGACION Y DESARROLLO DE LA FACULTAD DE INGENIERIA                     %%
%%   ELECTRICA Y ELECTRONICA                                                               %%
%%                                                                                         %%
%%   LIMA - PERU                                                                           %%
%%                                                                                         %%
%%                  LOCALIZACION                                                           %%
%%                                                                                         %%
%%     LOCALIZA DE LA PLACA DEL VEHICULO DENTRO DE UNA IMAGEN                              %%
%%                                                                                         %%
%%     Pre-Procesado - Acondicionamiento basico de la imagen                               %%
%%                                                                                         %%
%%     Grad_Sobel_1 -  Sub rutina que obtiene el gradiente de Sobel en las                 %%
%%         direcciones horizontales Este y Oeste.                                          %%
%%                                                                                         %%
%%     Umbral_Gr_3 -  Sub rutina que establece un umbral de gradiente                      %%
%%         relativo a partir del histograma de gradientes.                                 %%
%%                                                                                         %%
%%     Umbral_Dens_2 - Sub rutina que establece un umbral de densidad de                   %%
%%         gradientes, despues de determinar la 'densidad' de los mismos.                  %%
%%                                                                                         %%
%%     Morphis_2 - Sub rutina que aplica los operadores de erosion y                       %%
%%         dilatacion.                                                                     %%
%%                                                                                         %%
%%     separa_rectangulos - Funcion que obtiene a paritr de la imagen una                  %%
%%         o dos zonas con mas probabilidad de encerrar una placa. Las                     %%
%%         entradas son las imagenes de 320x240(decimada) y la de 640x480.                 %%
%%                                                                                         %%
%%                                                                                         %%
%%    ASESOR: DR. JORGE ALBERTO DEL CARPIO SALINAS    jdelcarpio@uni.edu.pe                %%
%%    GRADUANDO:    GIANCARLO AYALA CHARCA     giancarlo_ayala@dacafe.com                  %%
%%                                                                                         %%
%%                                                                                         %%
%%   Matlab Version 6.5                                                                    %%
%%   Image Processing Toolbox   Version 2.2 (R.11) o Superior                              %%
%%                                                                                         %%
%%                                     Enero 2004                                          %%
%%                                                                                         %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [zona1,zona2,area1,area2,CONTADOR]=Localizacion(ii)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%  PRE PROCESADO  %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
I=rgb2gray(ii);
[nF,nC]=size(I);
if (nF==480) & (nC==640)
%%%%%   DECIMAMOS A LA MITAD
I_ef=I(1:2:nF,1:2:nC);
[nF,nC]=size(I_ef);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
elseif (nF==240) & (nC==320)
   I_ef=I;
   [nF,nC]=size(I_ef);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
x=[1:nF];
y=[1:nC];
[yg,xg]=meshgrid(y,x);

%%%%%
Grad_Sobel_1; % También puede graficar la imagen.
% en escala de grises en 3D

%%%%%%%%%%%% DESHABILITAMOS LA GRAFICA %%%%%%%%%%%%%
%figure, mesh(xg,yg,abs(double(I2))),xlabel('x'),ylabel('y'),
%zlabel('z');title('Detección de borde por Sobel (ABS)')
%colorbar

Umbral_Gr_3

      for m = 1:nF,
        for n = 1:nC,
          if I2(m,n)> Umb_grad
            I2_Umb(m,n) = 1;
          else
            I2_Umb(m,n) = 0;
          end
        end
      end

%-> figure, imshow(I2_Umb,[]),title('Valores sobre el Umbral de Gradientes')

%figure, mesh(xg,yg,(double(I2_Umb))),xlabel('x'),ylabel('y'),
%zlabel('z');title('Bordes mayores que el Umbral de Gradiente')

%%%%%%%%%  DENSIDAD DE GRADIENTES
h2=ones(5,91);
I2_Dens=filter2(h2,I2_Umb);
%%%%%%   Grafica en escala de grises de la densidad
%-> figure, imshow(I2_Dens,[]), title('Densidad de gradientes')
%%%%%%
          %% DESHABILITAMOS LAS GRAFICAS
%% Graficamos en 3D:
%figure, mesh(xg,yg,double(I2_Dens)),xlabel('x'),ylabel('y'),
%zlabel('z');title('Zonas más Densas de Gradientes')
%colorbar

%%%%%%
%%%% Nuevamente calculamos un nivel mínimo (Umbral) ahora para la densidad.
Umbral_Dens_2;

      for m = 1:nF,
        for n = 1:nC,
          if I2_Dens(m,n)> Umb_Dens
            I2_Dens_Umb(m,n) = 1;
          else
            I2_Dens_Umb(m,n) = 0;
          end
        end
      end

%figure, imshow(I2_Dens_Umb,[]),title('Sobre el Umbral de Densidad')

BW=I2_Dens_Umb;
%Morphis_2;
```

```
BW3=Morphis_2(BW);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Este codigo separa posibles rectangulos para que pasen por las sgtes. etapas
% Arroja posibles placas: zona1 y zona2, adicionalmente los parametros
% CONTADOR asi tambien, area1 y area2.
%  separa_rectangulos;

  [zona1,zona2,area1,area2,CONTADOR]=separa_rectangulos(BW3,I);
```

ANEXO K
CÓDIGO FUENTE DE LA CORRECIÓN DE PERSPECTIVA

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
UNIVERSIDAD NACIONAL DE INGENIERIA                                      %%
%%   CENTRO DE INVESTIGACION Y DESARROLLO DE LA FACULTAD DE INGENIERIA   %%
%%   ELECTRICA Y ELECTRONICA                                              %%
%%                                                                        %%
%%   LIMA - PERU                                                          %%
%%                                                                        %%
%%          corregida=corrige_perspectiva2(zona2)                         %%
%%                                                                        %%
%%        FUNCION QUE CORRIGE LA PERSPECTIVA DE LA PLACA                  %%
%%                                                                        %%
%%    Esta funcion implementa la transformada de Hough, y la transformacion %%
%%    proyectiva para hacer la correccion de distorsion en perspectiva de %%
%%    la imagen. Este archivo utiliza dos funciones mas: verificacion1 y  %%
%%    correccion, el primero llama al 2do y este arroja un resultado que se %%
%%    propaga hasta el primer bloque: corregida.                          %%
%%                                                                        %%
%%    zona2 - Es la entrada principal al bloque, es una matriz que tiene la %%
%%        imagen de la placa distorsionada. En esta primera parte se      %%
%%        utiliza la transformada de Hough para determinar, utilizando    %%
%%        rectas, el marco de la placa y con ello sus vertices, estos nos %%
%%        permiten hallar la matriz de transformacion para corregir la    %%
%%        imagen.                                                         %%
%%                                                                        %%
%%    corregida - Matriz que contiene la imagen corregida de la placa.    %%
%%                                                                        %%
%%    verificacion1 - Funcion que verifica que los vertices hallados      %%
%%        encierren la region de los caracteres y asi poder hacer la      %%
%%        transformacion. Esta funcion llama a otra llamada 'correccion'. %%
%%                                                                        %%
%%    corrección – Función que realiza la ubicación de los vértices,      %%
%%         el ordenamiento de los mismos y la aplicación de la transformación %%
%%         proyectiva para proyectar la placa en un plano frontal.        %%
%%                                                                        %%
%%                                                                        %%
%%   ASESOR: DR. JORGE ALBERTO DEL CARPIO SALINAS    jdelcarpio@uni.edu.pe %%
%%   GRADUANDO:    GIANCARLO AYALA CHARCA     giancarlo_ayala@dacafe.com   %%
%%                                                                        %%
%%                                                                        %%
%%   Matlab Version 6.5                                                   %%
%%   Image Processing Toolbox   Version 2.2 (R.11) o                      %%
%%                                                                        %%
%%                             Enero 2004                                 %%
%%                                                                        %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function corregida=corrige_perspectiva2(zona2)
%Corrigiendo el programa para las rectas horizontales

A=edge(zona2,'sobel','horizontal');       % Detección de bordes horizontales sobre 'zona2'
[m,n]=size(A);
Mitad1=A(1:round(m/2),1:n);               % Dividimos la matriz en dos mitades superior e inferior
Mitad2=A(round(m/2):m,1:n);
visualizacion=zeros(m,n);


% TRANSFORMADA DE HOUGH PARA EL CASO 1     %  (Recta horizontal superior)
A1=Mitad1;
[m_1,n_1]=size(A1);
theta=-30:30;                   % Restringimos el ángulo de inclinación de las placas
[L1_1,L2_1]=size(theta);   % L2, tamaño de theta
p=(m_1^2+n_1^2)^0.5/100;
rho1=0:p:(m_1^2+n_1^2)^0.5;    % Discretización de rho
[L3_1,L4_1]=size(rho1);   % Definimos la matriz que contará las intersecciones
N_1=zeros(L4_1,L2_1);
for i=1:m_1,
   for j=1:n_1,
      if (A1(i,j)==1)
         for k=1:L2_1,
           rho2=i*cos(theta(k)*pi/180)+j*sin(theta(k)*pi/180);   % usamos la ecuación con las filas i, las 'x'
                                                                 % y las columnas j las 'y'. Fig. 4.8
           Q=1;
           while Q<L4_1
               if (rho1(Q)<rho2 & rho2<=rho1(Q+1))      % rho2 es mayor que rho1(Q) pero menor que
                                                        % el siguiente rho1(Q+1). Si está en el intervalo
                                                        % incrementa contador.
                  N_1(Q,k)=N_1(Q,k)+1;
               end
               Q=Q+1;
            end
         end
       end
   end
end;

% LOCALIZACION DE PUNTOS: ANGULO DE INCLINACION %
recta=N_1(1,1);
for i=1:L4_1,
   for j=1:L2_1,
      if (N_1(i,j)>recta)
        recta=N_1(i,j);
        rho_recta_1=rho1(i);
        theta_recta_1=theta(j);
      end
  end
end;

%GRaFICO DE LA RECTA HORIZONTAL%
matriz_de_h0=zeros(m,n);
for y=1:n;
   if (theta_recta_1~=0)
        x=(y-rho_recta_1*(1/sin(theta_recta_1*pi/180+eps)))/(-(1/tan(theta_recta_1*pi/180+eps)));
        new_x=round(x);
       if (new_x>=1 & new_x<=m)
        matriz_de_h0(new_x,y)=1;
        visualizacion(new_x,y)=1;
```

```
      end
   else
      new_x=round(rho_recta_1);
      matriz_de_h0(new_x,y)=1;
      visualizacion(new_x,y)=1;
  end
end;
Puntos=matriz_de_h0;

% TRANSFORMADA DE HOUGH PARA EL CASO 2%    ( Recta horizontal inferior)

A2=Mitad2;                    % Utilizamos la segunda mitad
[m_2,n_2]=size(A2);
theta=-30:30;
[L1_2,L2_2]=size(theta);
p=(m_2^2+n_2^2)^0.5/100;
rho1=0:p:(m_2^2+n_2^2)^0.5;
[L3_2,L4_2]=size(rho1);
N_2=zeros(L4_2,L2_2);
for i=1:m_2,
   for j=1:n_2,
      if (A2(i,j)==1)
         for k=1:L2_2,
           rho2=i*cos(theta(k)*pi/180)+j*sin(theta(k)*pi/180);
           Q=1;
           while Q<L4_2
               if (rho1(Q)<rho2 & rho2<=rho1(Q+1))
                  N_2(Q,k)=N_2(Q,k)+1;
               end
               Q=Q+1;
            end
         end
       end
   end
end;

% LOCALIZACION DE PUNTOS 2%

recta=N_2(1,1);
for i=1:L4_2,
   for j=1:L2_2,
      if (N_2(i,j)>recta)
        recta=N_2(i,j);
        rho_recta_2=rho1(i);
        theta_recta_2=theta(j);
     end
  end
end;

%GRÁFICO DE LA RECTA HORIZONTAL%

matriz_de_h1=zeros(m,n);
for y=1:n;
   if (theta_recta_2~=0)
        x=(y-rho_recta_2*(1/sin(theta_recta_2*pi/180+eps)))/(-(1/tan(theta_recta_2*pi/180+eps)));
        new_x=round(x);
       if (new_x>=1 & new_x+round(m/2)<=m)
        matriz_de_h1(new_x+round(m/2),y)=1;
        visualizacion(new_x+round(m/2),y)=1;
       end
```

```
    else
       new_x=round(rho_recta_2);
       matriz_de_h1(new_x+round(m/2),y)=1;
       visualizacion(new_x+round(m/2),y)=1;
   end
end;
Puntos=matriz_de_h0+matriz_de_h1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% TRANSFORMADA DE HOUGH %          (vertical)

B=edge(zona2,'sobel',0.045,'vertical');       % Determinamos bordes verticales
[m,n]=size(B);
theta=70:110;                                 % Restringimos las inclinaciones verticales
[L1,L2]=size(theta);
p=(m^2+n^2)^0.5/100;
rho1=0:p:(m^2+n^2)^0.5;
[L3,L4]=size(rho1);
NV=zeros(L4,L2);
for i=1:m,
    for j=1:n,
       if (B(i,j)==1)
          for k=1:L2,
            rho2=i*cos(theta(k)*pi/180+eps)+j*sin(theta(k)*pi/180+eps);
            Q=1;
            while Q<L4
                if (rho1(Q)<rho2 & rho2<=rho1(Q+1))
                   NV(Q,k)=NV(Q,k)+1;
                end
                Q=Q+1;
             end

          end
        end
    end
end;

% LOCALIZACION DE PUNTOS % (para la vertical)

matriz_de_h2=zeros(m,n);
NV2=NV(round(L4/2+1):L4,1:L2);            % Recta vertical derecha
[L4_2,L2_2]=size(NV2);
recta_V2=NV2(1,1);
for i=1:L4_2,
    for j=1:L2_2,
       if (NV2(i,j)>recta_V2)
          recta_V2=NV2(i,j);
          rho_recta_V2=rho1(i+round(2*L4/4+1));
          theta_recta_V2=theta(j);
       end
    end
end;
        for x_V=1:m;
           y_V=(-(1/tan(theta_recta_V2*pi/180+eps)))*x_V+rho_recta_V2*(1/sin(theta_recta_V2*pi/180+eps));
           new_y_V=round(y_V);
           if (new_y_V>=1 & new_y_V<=n)
              matriz_de_h2(x_V,new_y_V)=1;
              visualizacion(x_V,new_y_V)=1;
           end
        end;
```

```
Puntos=matriz_de_h0+matriz_de_h1+matriz_de_h2;

%LA OTRA RECTA VERTICAL%

matriz_de_h3=zeros(m,n);
NV3=NV(1:round(L4/2+1),1:L2);        % Recta vertical izquierda
[L4_2,L2_2]=size(NV3);
recta_V3=NV3(1,1);
for i=1:L4_2,
   for j=1:L2_2,
      if (NV3(i,j)>recta_V3)
         recta_V3=NV3(i,j);
         rho_recta_V3=rho1(i+1);
         theta_recta_V3=theta(j);
      end
   end
end;
      for x_V=1:m;
         y_V=(-(1/tan(theta_recta_V3*pi/180+eps)))*x_V+rho_recta_V3*(1/sin(theta_recta_V3*pi/180+eps));
         new_y_V=round(y_V);
         if (new_y_V>=1 & new_y_V<=n)
            matriz_de_h3(x_V,new_y_V)=1;
            visualizacion(x_V,new_y_V)=1;
         end
      end;
Puntos=matriz_de_h0+matriz_de_h1+matriz_de_h2+matriz_de_h3;
%figure,imshow(visualizacion)
corregida=verificacion1(Puntos,m,n,zona2,theta_recta_1,theta_recta_2,rho_recta_1,rho_recta_2,matriz_de_h0,mat
riz_de_h1,matriz_de_h2,matriz_de_h3,visualizacion);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %Función "verificación1"
function
corregida=verificacion1(Puntos,m,n,zona2,theta_recta_1,theta_recta_2,rho_recta_1,rho_recta_2,matriz_de_h0,mat
riz_de_h1,matriz_de_h2,matriz_de_h3,visualizacion,handles)
Puntos=Puntos-matriz_de_h0;
parte_1=Puntos(1:round(m/2),1:round(n/2));
parte_2=Puntos(1:round(m/2),round(n/2):n);
parte_3=Puntos(round(m/2):m,round(n/2):n);
parte_4=Puntos(round(m/2):m,1:round(n/2));
[m1,n1]=size(parte_1);
[m2,n2]=size(parte_2);
[m3,n3]=size(parte_3);
[m4,n4]=size(parte_4);
acknowledge1=0;
acknowledge2=0;
acknowledge3=0;
acknowledge4=0;
retardo=1;
contador=0;
%extremo superior izquierdo = parte_1
for i=1:m1,
   for j=1:n1,
      if (parte_1(i,j)==2)
         acknowledge1=1;
      end
   end
end
if (acknowledge1==0)
```

```
   while (acknowledge1~=1 & contador<round(m))
        matriz_de_h0=zeros(m,n);
        for y=1:n;
           if (theta_recta_1~=0)
             x=(y-retardo-rho_recta_1*(1/sin(theta_recta_1*pi/180+eps)))/(-(1/tan(theta_recta_1*pi/180+eps)));
             new_x=round(x);
             if (new_x>=1 & new_x<=m)
             matriz_de_h0(new_x,y)=1;
             visualizacion(new_x,y)=1;
             end
           else
            new_x=round(rho_recta_1);
            matriz_de_h0(new_x,y)=1;
            visualizacion(new_x,y)=1;
           end
        end
        Puntos=matriz_de_h0+matriz_de_h1+matriz_de_h2+matriz_de_h3;
        parte_1=Puntos(1:round(m/2),1:round(n/2));
        for i=1:m1,
          for j=1:n1,
            if (parte_1(i,j)==2)
              acknowledge1=1;
           end
          end
        end
        retardo=retardo+1;
        contador=contador+1;
    end
end
contador=0;
%extremo superior derecho = parte_2
for i=1:m2,
   for j=1:n2,
      if (parte_2(i,j)==2)
         acknowledge2=1;
      end
   end
end
if (acknowledge2==0)
   while (acknowledge2~=1 & contador<round(m))
        matriz_de_h0=zeros(m,n);
        for y=1:n;
           if (theta_recta_1~=0)
             x=(y-retardo-rho_recta_1*(1/sin(theta_recta_1*pi/180+eps)))/(-(1/tan(theta_recta_1*pi/180+eps)));
             new_x=round(x);
             if (new_x>=1 & new_x<=m)
             matriz_de_h0(new_x,y)=1;
             visualizacion(new_x,y)=1;
             end
           else
            new_x=round(rho_recta_1);
            matriz_de_h0(new_x,y)=1;
            visualizacion(new_x,y)=1;
           end
        end
        Puntos=matriz_de_h0+matriz_de_h1+matriz_de_h2+matriz_de_h3;
        parte_2=Puntos(1:round(m/2),round(n/2):n);
        for i=1:m2,
          for j=1:n2,
            if (parte_2(i,j)==2)
```

```
                acknowledge2=1;
              end
            end
          end
          retardo=retardo+1;
          contador=contador+1;
     end
end
contador=0;
%extremo inferior derecho = parte_3
for i=1:m3,
   for j=1:n3,
      if (parte_3(i,j)==2)
         acknowledge3=1;
      end
   end
end
if (acknowledge3==0)
   while (acknowledge3~=1 & contador<round(m))
         matriz_de_h1=zeros(m,n);
         for y=1:n;
            if (theta_recta_2~=0)
              x=(y-retardo-rho_recta_2*(1/sin(theta_recta_2*pi/180+eps)))/(-(1/tan(theta_recta_2*pi/180+eps)));
              new_x=round(x);
              if (new_x>=1 & new_x+round(m/2)<=m)
                 matriz_de_h1(new_x+round(m/2),y)=1;
                 visualizacion(new_x+round(m/2),y)=1;
              end
            else
               new_x=round(rho_recta_2);
               matriz_de_h1(new_x+round(m/2),y)=1;
               visualizacion(new_x+round(m/2),y)=1;
            end
          end
         Puntos=matriz_de_h0+matriz_de_h1+matriz_de_h2+matriz_de_h3;
         parte_3=Puntos(round(m/2):m,round(n/2):n);
         for i=1:m3,
            for j=1:n3,
              if (parte_3(i,j)==2)
                 acknowledge3=1;
              end
            end
         end
         retardo=retardo+1;
         contador=contador+1;
     end
end
contador=0;
%extremo inferior izquierdo = parte_4
for i=1:m4,
   for j=1:n4,
      if (parte_4(i,j)==2)
         acknowledge4=1;
      end
   end
end
if (acknowledge4==0)
   while (acknowledge4~=1 & contador<round(m))
         matriz_de_h1=zeros(m,n);
         for y=1:n;
```

```
            if (theta_recta_2~=0)
              x=(y+retardo-rho_recta_2*(1/sin(theta_recta_2*pi/180+eps)))/(-(1/tan(theta_recta_2*pi/180+eps)));
              new_x=round(x);
              if (new_x>=1 & new_x+round(m/2)<=m)
                 matriz_de_h1(new_x+round(m/2),y)=1;
                 visualizacion(new_x+round(m/2),y)=1;
              end
            else
               new_x=round(rho_recta_2);
               matriz_de_h1(new_x+round(m/2),y)=1;
               visualizacion(new_x+round(m/2),y)=1;
            end
          end
         Puntos=matriz_de_h0+matriz_de_h1+matriz_de_h2+matriz_de_h3;
         parte_4=Puntos(round(m/2):m,1:round(n/2));
         for i=1:m4,
            for j=1:n4,
               if (parte_4(i,j)==2)
                  acknowledge4=1;
               end
            end
         end
         retardo=retardo+1;
         contador=contador+1;
    end
end
axes(handles.axes4)
%ww=(visualizacion)*2;
%map=[0 0 0;1 0 0];
%imshow(ww,map)
imshow(visualizacion)
set(handles.axes4,'XMinorTick','on')

corregida=correccion(Puntos,m,n,zona2);

axes(handles.axes5)
imshow(corregida)
set(handles.axes5,'XMinorTick','on')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Función "corrección"
function corregida=correccion(Puntos,m,n,zona2)
%%%LOCALIZACION DE LOS VERTICES%%%
coordenadas=1;
for x=1:m,
   for y=1:n,
      if (Puntos(x,y)==2)      % Se busca valores iguales a 2, dado que serían las intersecciones luego de la
                               % suma de las matrices binarias.
         aa(coordenadas)=x;
         bb(coordenadas)=y;
         coordenadas=coordenadas+1;
      end
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%FUNCION PARA ORDENAR LOS VERTICES%%%
for i=1:4
   for j=1:4
      if (aa(i)<aa(j))
```

```
            tmp1=aa(i);
            tmp2=bb(i);
            aa(i)=aa(j);
            bb(i)=bb(j);
            aa(j)=tmp1;
            bb(j)=tmp2;
        end
    end
end
for i=1:2
    for j=1:2
        if (bb(i)<bb(j))
            tmp1=aa(i);
            tmp2=bb(i);
            aa(i)=aa(j);
            bb(i)=bb(j);
            aa(j)=tmp1;
            bb(j)=tmp2;
        end
    end
end
for i=3:4
    for j=3:4
        if (bb(i)>bb(j))
            tmp1=aa(i);
            tmp2=bb(i);
            aa(i)=aa(j);
            bb(i)=bb(j);
            aa(j)=tmp1;
            bb(j)=tmp2;
        end
    end
end
vertices=[aa(1) bb(1);aa(2) bb(2);aa(3) bb(3);aa(4) bb(4)]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Puntos1=[aa bb];
U=double(zona2);
[m_1,n_1]=size(zona2);
x1=1;
x2=1;
x3=max(aa(4)-aa(1),aa(3)-aa(2));   % Seleccionamos las dimensiones de la nueva imagen
x4=max(aa(4)-aa(1),aa(3)-aa(2));
y1=1;
y2=max(bb(2)-bb(1),bb(3)-bb(4));
y3=max(bb(2)-bb(1),bb(3)-bb(4));
y4=1;
%%%  Tomamos los vértices de la nueva imagen original
X1=vertices(1,1);
X2=vertices(2,1);
X3=vertices(3,1);
X4=vertices(4,1);
Y1=vertices(1,2);
Y2=vertices(2,2);
Y3=vertices(3,2);
Y4=vertices(4,2);
%%%   Resolvemos el sistema y hallamos la matriz de transformación H
a=[x1 y1 1 0 0 0 -1*X1*x1 -1*X1*y1;0 0 0 x1 y1 1 -1*Y1*x1 -1*Y1*y1;x2 y2 1 0 0 0 -1*X2*x2 -1*X2*y2;0 0 0
x2 y2 1 -1*Y2*x2 -1*Y2*y2;x3 y3 1 0 0 0 -1*X3*x3 -1*X3*y3;0 0 0 x3 y3 1 -1*Y3*x3 -1*Y3*y3;x4 y4 1 0 0 0 -
1*X4*x4 -1*X4*y4;0 0 0 x4 y4 1 -1*Y4*x4 -1*Y4*y4 ];
```

```
H=inv(a)*[X1 ;Y1; X2; Y2; X3; Y3; X4; Y4];
h11=H(1,1);
h12=H(2,1);
h13=H(3,1);
h21=H(4,1);
h22=H(5,1);
h23=H(6,1);
h31=H(7,1);
h32=H(8,1);

for x=1:max(aa(4)-aa(1),aa(3)-aa(2)),
   for y=1:max(bb(2)-bb(1),bb(3)-bb(4)),

         X=(h11*x+h12*y+h13)/(h31*x+h32*y+1);
         Y=(h21*x+h22*y+h23)/(h31*x+h32*y+1);
         m=fix(X);
         n=fix(Y);
             if (m>0&m<m_1)
                if(n>0&n<n_1)
                  AAAA(x,y)=(U(m+1,n)-U(m,n))*(X-m)+(U(m,n+1)-U(m,n))*(Y-n)+(U(m+1,n+1)+U(m,n)-
U(m+1,n)-U(m,n+1))*(X-m)*(Y-n)+U(m,n);
                  AAAA(x,y)=fix(AAAA(x,y));
                end
            end
    end
end
%figure,imshow(uint8(AAAA))
corregida=uint8(AAAA);
```

ANEXO L
CÓDIGO FUENTE DE LA BINARIZACIÓN

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%    UNIVERSIDAD NACIONAL DE INGENIERIA                                                    %%
%%    CENTRO DE INVESTIGACION Y DESARROLLO DE LA FACULTAD DE INGENIERIA     %%
%%    ELECTRICA Y ELECTRONICA                                                                     %%
%%                                                                                                                   %%
%%    LIMA - PERU                                                                                              %%
%%                                                                                                                   %%
%%                   zonaBin = Binariza(corregida)                                                  %%
%%                                                                                                                   %%
%%     FUNCION QUE CONVIERTE LA IMAGEN DE ESCALA DE GRISES A BINARIO (B/N)    %%
%%                                                                                                                   %%
%%     corregida - Matriz de una imagen en escala de grises, proviene de la                 %%
%%         etapa anterior                                                                                      %%
%%                                                                                                                   %%
%%     zonaBin - Matriz que contitne la imagen ya convertida en blanco y                    %%
%%         negro (B/N).                                                                                         %%
%%                                                                                                                   %%
%%     pixels_bordes - Funcion que realiza la reestructuración del                            %%
%%         histograma.                                                                                         %%
%%                                                                                                                   %%
%%     umbral_optimo - Funcion que determina el valor umbral de gris para la               %%
%%         binarizacion.                                                                                       %%
%%                                                                                                                   %%
%%                                                                                                                   %%
%%    ASESOR: DR. JORGE ALBERTO DEL CARPIO SALINAS    jdelcarpio@uni.edu.pe           %%
%%    GRADUANDO:    GIANCARLO AYALA CHARCA     giancarlo_ayala@dacafe.com           %%
%%                                                                                                                   %%
%%                                                                                                                   %%
%%                                                                                                                   %%
%%    Matlab Version 6.5                                                                                    %%
%%    Image Processing Toolbox   Version 2.2 (R.11) o Superior                                 %%
%%                                                                                                                   %%
%%                                             Enero 2004                                                        %%
%%                                                                                                                   %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function zonaBin=Binariza(corregida)

  a=pixels_bordes(corregida);
%   umbral_optimo(uint8(a))
  zonaBin=im2bw(corregida,umbral_optimo(uint8(a)));
      figure,imshow(zonaBin), title('Binarizada');


%% A continuación las funciones involucradas, estas se encuentran en archivos diferentes
```

```
function a=pixels_bordes(zona2)

grad_zona=Grad_Sobel(zona2);
grad_zona_norm=grad_zona*(255)/(max(max(grad_zona)));
[fs,cs]=size(zona2);
for i=1:fs,
  for j=1:cs,
              if grad_zona_norm(i,j) < 7 %25 o 3 cuando es opaca, 7 gral.
                a(i,j)=0; %
              else
         a(i,j)= zona2(i,j);%grad_zona_norm(i,j);

    end
  end
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function umbral=umbral_optimo(zona2)

[fz,cz]=size(zona2);
inicio=round((fz-(fz*0.8))/2); %0.98
final=inicio+round(fz*0.8);

for p=inicio:final,
L=double(zona2(p,:));
Qs(1)=0; n1s=0;
for j=1:cz-1,
  if L(j+1)-L(j)>0
    Qs(j+1)=1;
    n1s=n1s+1;
  else
    Qs(j+1)=0;
  end
end
sum1=sum(Qs.*L);
sum0=sum(L)-sum1;
n0s=cz-n1s;
%% Las medias para cada estado
mean0=sum0/n0s;
mean1=(sum1+eps)/(n1s+eps);
%% El Vector de Umbrales será..
U(p-inicio+1)=(mean0*n1s+mean1*n0s)/(n0s+n1s);
end
U=round(U);
%u=max(U);

%%%%%%%% MIDIENDO LA CALIDAD DEL UMBRAL ELEGIDO:
%%%% Deshabilito grafica de histograma
[ni,X]=imhist(zona2);
%X=X-1; %V5.3
%%% Medición..
%% Para cada valor de "U"

for i=1:length(U)
 %%Media de pixeles negros : mu_N
 Nn=sum(ni(1:U(i)));
  mu_N=0;
  for j=1:U(i),
    mu_N = mu_N + X(j)*(ni(j)+eps)/(Nn+eps);
  end
```

```
  %%Media de pixeles blancos : mu_B
 Nb=sum(ni(U(i)+1:length(ni)));
  mu_B=0;
  for j= U(i)+1:length(ni),
     mu_B = mu_B + X(j)*(ni(j)+eps)/(Nb+eps);
  end

   Perc(i)=0.15*(mu_B-mu_N);

 if ( ((U(i)-round(Perc/2)+1)>0) & (U(i)+round(Perc/2)+1)<=length(ni) )
   Pixels_vecindad_u(i)=sum(ni( U(i)- round(Perc/2)+1 : U(i)+round(Perc/2)+1 ) );
 end
%   Pixels_vecindad_u(i)=sum(ni( U(i)- round(0.15*(mu_B-mu_N)/2)+1 : U(i)+round(0.15*(mu_B-mu_N)/2))+1
);
end

%%% Estas 3 lineas para diagnostico:
%figure, stem(X,ni),title('Histograma de zona2'),grid
%hold on
%stem(U,ni(U),'r')

%figure, subplot(2,1,1),stem(U),title('Valores del vector U')
%subplot(2,1,2),stem(Pixels_vecindad_u), title('Cantidad de pixels en vecindad del umbral "u"')

[pixels,val]= min(Pixels_vecindad_u);
umbral=U(val)/255;

%figure,im2bw(zona2,umbral/256), title('umbralizado usando el optimo')
```

ANEXO M
CÓDIGO FUENTE DE LA SEGMENTACIÓN

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    UNIVERSIDAD NACIONAL DE INGENIERIA                                    %%
%%    CENTRO DE INVESTIGACION Y DESARROLLO DE LA FACULTAD DE INGENIERIA    %%
%%    ELECTRICA Y ELECTRONICA                                              %%
%%                                                                         %%
%%    LIMA - PERU                                                          %%
%%                                                                         %%
%%              [CaractDec,estado]= SEGMENTACION(zonaBin,tt)               %%
%%                                                                         %%
%%     FUNCION ENCARGADA DE LA SEGMENTACION DE LOS CARACTERES DE LA PLACA  %%
%%                                                                         %%
%%     Esta funcion realiza primero una aproximacion hacia los caracteres  %%
%%     utilizando proyecciones verticales y horizontales, cuando esta lo   %%
%%     suficientemente cerca extrae los caracteres.                        %%
%%                                                                         %%
%%     zonaBin - Matriz que contiene la imagen de la placa ya corregida, en %%
%%          blanco y negro, para extraer los caracteres.                   %%
%%                                                                         %%
%%     tt - Parametro que indica el numero de veces que se esta efectuando %%
%%          la funcion.                                                    %%
%%                                                                         %%
%%     CaractDec - Arreglo de matrices que contienen las imagenes de los   %%
%%          caracteres, ya reducidos de tamaño (16x8).                     %%
%%                                                                         %%
%%     estado - Indicador de estado de exito ('1') o fracaso ('0') en la   %%
%%          segmentacion.                                                  %%
%%                                                                         %%
%%     Extraccion - Funcion encargada de realizar la extraccion de los     %%
%%          caracteres a traves de un algoritmo basado en la proyeccion    %%
%%          horizontal.                                                    %%
%%                                                                         %%
%%                                                                         %%
%%    ASESOR: DR. JORGE ALBERTO DEL CARPIO SALINAS    jdelcarpio@uni.edu.pe %%
%%    GRADUANDO:    GIANCARLO AYALA CHARCA     giancarlo_ayala@dacafe.com  %%
%%                                                                         %%
%%                                                                         %%
%%                                                                         %%
%%    Matlab Version 6.5                                                   %%
%%    Image Processing Toolbox   Version 2.2 (R.11) o Superior             %%
%%                                                                         %%
%%                                   Enero 2004                            %%
%%                                                                         %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [CaractDec,estado]= SEGMENTACION(zonaBin,tt)
```

```
[fs,cs]=size(zonaBin);
plk=~zonaBin;
%Pv=sum(plk');     %proyeccion vertical
Ph=sum(plk);      %proyeccion horizontal
%%%%%%figure, plot(Pv);
% figure, plot(Ph);,title('Proyeccion Horizontal')

%%% Es importante una buena binarizacion !! (0.25 en nivel horizontal)
nivel_h=min(Ph)+0.25*(max(Ph)-min(Ph)); %si se aumenta mucho puede ser perjudicial
nPh=Ph-nivel_h;   %Nuevo nivel para la proyeccion Horizontal
[Cc]=find(nPh<=0); %Columnas de Caracteres
plkN=plk(1:fs,min(Cc):max(Cc)); %figure, imshow(~plkN)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Del recorte de la Proyeccion horizontal hacemos la vertical

%%%%AQUI SE PUEDE BUSCAR MEJOR EL MINIMO DE LA IZQUIERDA Y DERECHA !!!!

Pv=sum(plkN');
%--> figure, plot(Pv), title('antes')
if Pv(1)~=0,
   linea_sup=zeros(1,length(plkN(1,:)));
   plkN_mod=cat(1,linea_sup,plkN);  %agregamos una fila arriba
   Pv=sum(plkN_mod');
   plkN=plkN_mod;
end;
if Pv(length(Pv))~=0,
   linea_inf=zeros(1,length(plkN(1,:)));
   plkN_mod=cat(1,plkN,linea_inf);  %agregamos una fila abajo
   Pv=sum(plkN_mod');
   plkN=plkN_mod;
end

%--> figure, plot(Pv), title('proyeccion corregida') %Proyeccion Vertical
 nivel_v=min(Pv)+0.12*(max(Pv)-min(Pv)); %0.05-0.12 (0.15)-->0.14 (0.2)
 nPv=Pv-nivel_v;   %Nuevo nivel para la proyeccion Vertical

  [cuts]=find(nPv<=0);
  cut_aux1=diff(cuts);     % genera diferencias uno a uno consecutivas
  cut_aux2=find(cut_aux1>20); %-->>(20)  % mayores que la diferencia 20, la altura del caracter

   infe=cuts(cut_aux2);
   supe=cuts(cut_aux2) + cut_aux1(cut_aux2);

Lim_C=~plkN(infe:supe,:);
%figure, imshow(Lim_C)

[cort_vert]=find(sum(~Lim_C')==0);

if (max(cort_vert)-min(cort_vert))>20
Lim_C=Lim_C(min(cort_vert):max(cort_vert),:);
%figure, imshow(Lim_C)
end


[h,w]=size(Lim_C);
Neg=~Lim_C;
Pv=sum(Neg');
```

```
%-> figure, plot(Pv),title('proyeccion vertical')

%%%%%%%%%%%%%%%%%%  Criterio de derivada
der1=diff(Pv,1);
AbsDer1=abs(der1); %figure, plot(AbsDer1), title('abs(1era derivada)')

nivel_v=max(AbsDer1)-0.4*(max(AbsDer1)-min(AbsDer1));
range=AbsDer1-nivel_v;
[Fc]=find(range>0);

if (max(Fc)-min(Fc))>20
Lim_H=~Neg(min(Fc):max(Fc),1:w); %figure, imshow(Lim_H),
%title('usando criterio de derivada "Lim _ H"')
else
Lim_H=Lim_C;
end

%Extraccion;
[CaractDec,estado]= Extraccion(Lim_H,plk,cs,tt);
```

ANEXO N
CÓDIGO FUENTE DE LA EXTRACCIÓN DE PATRONES

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%   UNIVERSIDAD NACIONAL DE INGENIERIA                                       %%
%%   CENTRO DE INVESTIGACION Y DESARROLLO DE LA FACULTAD DE INGENIERIA        %%
%%   ELECTRICA Y ELECTRONICA                                                  %%
%%                                                                            %%
%%   LIMA - PERU                                                              %%
%%                                                                            %%
%%  [Componentes_principales_placa,Ns] = Extrae_caracteristicas(CaractDec)    %%
%%                                                                            %%
%%     FUNCION QUE APLICA LA TRANSFORMADA KL PARA EXTRAER CARACTERÍSTICAS     %%
%%                                                                            %%
%%      La entrada a esta funcion es el arreglo de matrices CaractDec,        %%
%%      previamente a la extraccion de caracteristicas se hace una            %%
%%      normalizacion de todos los caracteres a un arreglo de matrices de     %%
%%      16x16, y luego de ello se procede a aplicar la transformada KL para   %%
%%      la extraccion de caracteristicas. La salida  son vectores columna     %%
%%      'Componentes_principales_placa' que contienen el 85% de energia, en   %%
%%      este caso son vectores con 41 elementos en vez de los 256 (16x16)     %%
%%      que tenian las matrices originales.                                   %%
%%                                                                            %%
%%      Componentes_principales_placa - Arreglo de vectores que contienen     %%
%%          el 85% de energia en componentes principales de las imagenes de   %%
%%          los caracteres, ahora comprimidos a vectores de 41 elementos,     %%
%%          esto como resultado de aplicar la transformada KL.                %%
%%                                                                            %%
%%      Ns - Numero de caracters que se estan utilizando.                     %%
%%                                                                            %%
%%      Normalizacion - Funcion que aplica la normalizacion de las imagenes   %%
%%          de los caracteres centrandolos en matrices cuadradas de 16x16,    %%
%%          completando columnas en cualquiera de los lados si es necesario.  %%
%%                                                                            %%
%%      KLT - Transformada KL (Karhunen-Loeve), a traves de esta funcion se   %%
%%          implementa esta transformada, basicamente utliza el conjunto de   %%
%%          Bases Reducidas que se encuentran almacenadas en el archivo de    %%
%%          nombre bases_reducidas_41 y  halla las componentes de la imagen   %%
%%          en esta nueva Base Ortogonal. Utiliza como entrada la funcion     %%
%%          mat_a_col, que convierte de matriz cuadrada a matriz columna.     %%
%%                                                                            %%
%%                                                                            %%
%%     ASESOR: DR. JORGE ALBERTO DEL CARPIO SALINAS    jdelcarpio@uni.edu.pe  %%
%%     GRADUANDO:    GIANCARLO AYALA CHARCA     giancarlo_ayala@dacafe.com    %%
%%                                                                            %%
%%                                                                            %%
%%                                                                            %%
%%   Matlab Version 6.5                                                       %%
%%   Image Processing Toolbox   Version 2.2 (R.11) o Superior                 %%
%%                                                                            %%
%%                                    Enero 2004                              %%
%%                                                                            %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [Componentes_principales_placa,Ns] = Extrae_caracteristicas(CaractDec)

%%%%%%%%%%%%%%%%%%%%%%%        EXTRACCION        DE        CARACTERISTICAS
%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Para Normalizacion la entrada es CaractDec
    char_norm = Normalizacion(CaractDec); % Arroja : char_norm

   %% TRABAJA CON EL FILE: bases_reducidas_41
%%%%%% Para convertir todos en columnas
caracteres_col=mat_a_col(char_norm(:,:,1));;
Ns=length(char_norm(1,1,:));
for L=2:Ns,
   chs=mat_a_col(char_norm(:,:,L));
   caracteres_col=[caracteres_col,chs];
end
%%%%%%
%load bases_reducidas Bases_reduc
% load bases_reducidas_N Bases_reduc % de 35 elementos
 load bases_reducidas_41 Bases_reduc

Componentes_principales_placa=KLT(Bases_reduc,caracteres_col);
```

ANEXO O
CÓDIGO FUENTE DEL RECONOCIMIENTO DE PATRONES

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%    UNIVERSIDAD NACIONAL DE INGENIERIA                                 %%
%%    CENTRO DE INVESTIGACION Y DESARROLLO DE LA FACULTAD DE INGENIERIA  %%
%%    ELECTRICA Y ELECTRONICA                                            %%
%%                                                                       %%
%%    LIMA - PERU                                                        %%
%%                                                                       %%
%%        num_placa = Aplica_a_NN(Componentes_principales_placa,Ns)      %%
%%                                                                       %%
%%      FUNCION QUE IMPLEMENTA LA RED NEURONAL PARA RECONOCER CARACTRES  %%
%%                                                                       %%
%%      Esta funcion implementa una red neuronal asignando la salida a una %%
%%      determinada clase o caracter, previamente la red ha sido entrenada %%
%%      con una cantidad de patrones para hacer la clasificacion.       %%
%%                                                                       %%
%%      Componentes_principales_placa - Este es un conjunto de vectores  %%
%%          columna que contienen las componentes principales de las     %%
%%          imagenes de los caracteres y sirven de entrada a la red      %%
%%          neuronal.                                                    %%
%%                                                                       %%
%%      Ns - Numero de caracteres que se estan utilizando.               %%
%%                                                                       %%
%%      num_placa - Es la salida de la funcion y arroja un arreglo de    %%
%%          los caracteres reconocidos por la red neuronal.              %%
%%                                                                       %%
%%                                                                       %%
%%     ASESOR: DR. JORGE ALBERTO DEL CARPIO SALINAS    jdelcarpio@uni.edu.pe %%
%%     GRADUANDO:    GIANCARLO AYALA CHARCA     giancarlo_ayala@dacafe.com   %%
%%                                                                       %%
%%                                                                       %%
%%                                                                       %%
%%    Matlab Version 6.5                                                 %%
%%    Image Processing Toolbox   Version 2.2 (R.11) o Superior           %%
%%                                                                       %%
%%                                 Enero 2004                            %%
%%                                                                       %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function num_placa = Aplica_a_NN(Componentes_principales_placa,Ns)

   load matrix2_41_loglog_3
A=Componentes_principales_placa;
simula=sim(net,A);
A2=compet(simula);
for J=1:Ns
answer(J)=find(compet(A2(:,J))==1);
end
```

```
C=['1';'2';'3';'4';'5';'6';'7';'8';'9';'0';...
   'A';'B';'C';'D';'E';'F';'G';'H';'I';'J';
   'K';'L';'M';'N';'o';'P';'Q';'R';'S';'T';...
   'U';'V';'W';'X';'Y';'Z'];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% INTRODUCIMOS UN CODIGO PARA SELECCIONAR NUMEROS Y LETRAS
%% LOS 2 PRIMEROS DEBEN SER LETRAS
%% LOS 3 ULTIMOS DEBEN SER NUMEROS
for jj=1:2
 if answer(jj)== 8,
     answer(jj)=12;
  end;

 if answer(jj)==10,
     answer(jj)=25;
 end;

 if answer(jj)==5,
     answer(jj)=29;
 end;
end

for kk=4:6
 if answer(kk)== 12,
      answer(kk)=8;
 end;

 if answer(kk)==25,
     answer(kk)=10;
 end;

 if answer(kk)==29,
     answer(kk)=5;
 end;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

num_placa=C(answer);
```

ANEXO P
CÓDIGO FUENTE DEL MICROCONTROLADOR PIC16F628

```
;=====================================================
;PROGRAMA:
;       SENSOR INGRESO AUTOS
;       VERSION 2
;=====================================================
      LIST P=PIC16F628
      #include "P16F628.inc"
      #include "MACROPIC.inc"
      ERRORLEVEL -302,-202
  ;    __FUSES _CP_OFF&_HS_OSC&_WDT_OFF&_LVP_OFF&_BODEN_OFF

      CBLOCK H'20'
         CONTADOR_1
         CONTADOR_2
         CONTADOR_3
         PDel0
         PDel0E
         PDel1E
         PDel1
         DLY_CNT
      ENDC

      ORG 0X00                ; Indica al ensamblador la dirección de la memoria de
                              ; programa donde se sitúa la siguiente instrucción
       GOTO    MAIN           ; Let's get this puppy rolling
MAIN:
      CALL    INICIA_TX       ; Macro en UART.ASM, Pone RB2 como salida, establece
                              ; transmisión a 8 bits, 9600bps y habilita puerto serial.
      BANCO   0               ; Macro que indica seleccionar Banco 0 de memoria de datos
      MOVLW   0X07           ; Establecer PORTA I/O Digital.
      MOVWF   CMCON          ; CMCON<0:3>, Apaga los comparadores
      MOVLW   0X30           ; Pone a cero los contadores
      MOVWF   CONTADOR_1
      MOVWF   CONTADOR_2
      MOVWF   CONTADOR_3
      BANCO   1               ; Selecciona  Banco 1
      MOVLW   B'00000001'
      MOVWF   TRISA          ; Establece RA<0> como entrada (pin 17: proviene de la
                              ; comparación en el LM324, del cambio en el LDR)
      MOVLW   B'10000000'
```

```
      MOVWF   TRISB        ; Establece RB<7> en alta impedancia (pin 6: Reset)
      BANCO   0            ; Regresa a Banco 0
      BCF     PORTB,0      ; Establece RB0 como salida.
      GOTO    TRANSMITE
BUCLER:
     MOVLW   0X30                ; Pone a cero los contadores.
     MOVWF   CONTADOR_1
     MOVWF   CONTADOR_2
     MOVWF   CONTADOR_3
     GOTO    INFINITO

INFINITO:
      BTFSC   PORTA,0      ; Si Bit '0' de PORTA (RA0) es cero (pin 17) , salta la
                           ; próxima instrucción y realiza NOP. (Instrucción de 2 ciclos)
      GOTO    INFINITO
HIGH_P:
      BCF     PORTB,0      ; Pone a cero el bit '0' de PORTB (RB0), Indicador de TX.
      BTFSS   PORTB,7      ; Si el bit '7' de PORTB es uno, salta la siguiente instrucción
                           ; (Reset)
      GOTO    BUCLER       ; Cuando es cero, ejecuta la instrucción (Resetea contador).
      BTFSS   PORTA,0      ; Si el bit '0' de PORTA es uno, salta la siguiente instrucción
      GOTO    HIGH_P
      BSF     PORTB,0      ; Pone a uno el bit '0' de PORTB (RB0), Indicador de TX.
      CALL DEMORAE         ; Retardo
      CALL DEMORAE
      CALL DEMORAE
      CALL DEMORAE
      CALL DEMORAE
      CALL DEMORAE
      CALL DEMORAE
      CALL DEMORAE         ; Retardo
CUENTA:                    ; Cuenta unidades
      INCF    CONTADOR_1,1
      CJE     CONTADOR_1,0X3A,CUENTA_1    ; Compara y salta si es igual
      GOTO    TRANSMITE ; Si es <= 9, Transmite
CUENTA_1:                  ; Cuenta decenas
      MOVLW   0X30         ; Pone en cero unidades
      MOVWF   CONTADOR_1
      INCF    CONTADOR_2,1       ; Incrementa contador de decenas
      CJE     CONTADOR_2,0X3A,CUENTA_2    ; Compara y salta si es igual
      GOTO    TRANSMITE ; Si es <= 9, Transmite
CUENTA_2:                  ; Cuenta centenas
      MOVLW   0X30         ; Pone en cero decenas
      MOVWF   CONTADOR_2
      INCF    CONTADOR_3,1       ; Incrementa contador de centenas
      CJE     CONTADOR_3,0X3A,CUENTA_3    ; Compara y salta si es igual
      GOTO    TRANSMITE  ; Si es <= 9, Transmite
CUENTA_3:
      MOVLW   0X30         ; Pone en cero centenas
      MOVWF   CONTADOR_3
```

```
TRANSMITE:
    MOVLW 0X20          ; Inicializa puntero
    CALL TX_DATO        ; Mueve el valor 20H del registro de trabajo (W) al de
                        ; transmisión (TXREG).
    CALL DEMORAE        ; Retardo
    MOVF CONTADOR_3,0  ; Coloca 0 en contador 3
    CALL TX_DATO        ; Mueve el valor de W a TXREG para centenas.
    CALL DEMORAE        ; Retardo
    MOVF CONTADOR_2,0  ; Coloca 0 en contador 2
    CALL TX_DATO        ; Mueve el valor de W a TXREG para decenas.
    CALL DEMORAE        ; Retardo
    MOVF CONTADOR_1,0  ; Coloca 0 en contador 1
    CALL TX_DATO        ; Mueve el valor de W a TXREG para unidades.
    CALL DEMORAE        ; Retardo
    GOTO INFINITO

    INCLUDE <UART.ASM>
    INCLUDE <RETARDO.ASM>

    END
```

```
;*****************************************************************
;   FILENAME:          MACROPIC.INC                              *
;   DATE:        16 / 02 / 2004                                  *
;   FILE VERSION:  1.3                                           *
;   AUTHOR:       A.G. LY         *
;   COMPANY:      INICTEL                                        *
;*****************************************************************
;   FILES REQUIRED:                                              *
;                                                *
;*****************************************************************
;   NOTES:                                                       *
;              APLICACIONES AVANZADAS CON PIC                    *
;*****************************************************************

;***** PAGINA X *****
PAGINA      MACRO       X
        IF X == 0
                BCF    PCLATH,3
                BCF    PCLATH,4
        ENDIF
        IF X == 1
                BSF    PCLATH,3
                BCF    PCLATH,4
        ENDIF
        IF X == 2
                BCF    PCLATH,3
                BSF    PCLATH,4
        ENDIF
        IF X == 3
                BSF    PCLATH,3
                BSF    PCLATH,4
        ENDIF
        ENDM

;***** BANCO X *****
BANCO       MACRO       X
        IF X == 0
                BCF    STATUS,RP0
                BCF    STATUS,RP1
        ENDIF
        IF X == 1
                BSF    STATUS,RP0
                BCF    STATUS,RP1
        ENDIF
        IF X == 2
                BCF    STATUS,RP0
                BSF    STATUS,RP1
        ENDIF
        IF X == 3
                BSF    STATUS,RP0
```

```
            BSF    STATUS,RP1
        ENDIF
        ENDM

;***** MOV *****
MOV  MACRO      REGISTRO,LITERAL
        MOVLW      LITERAL
        MOVWF      REGISTRO
        ENDM

;***** MOVR *****
MOVR        MACRO      REGISTRO1,REGISTRO2
        MOVFREGISTRO2,0
        MOVWF      REGISTRO1
        ENDM

;***** COMPARA Y SALTA SI NO ES IGUAL *****
CJNE  MACRO      REGISTRO,DATO,DIRECC
        MOVLW      DATO
        SUBWF      REGISTRO,0
        BTFSSSTATUS,Z
        GOTO DIRECC
        ENDM

;***** COMPARA Y SALTA SI ES IGUAL *****
CJE    MACRO      REGISTRO,DATO,DIRECC
        MOVLW      DATO
        SUBWF      REGISTRO,0
        BTFSC      STATUS,Z
        GOTO DIRECC
        ENDM

;***** DECREMENTA, COMPARA Y SALTA SI NO ES IGUAL *****
DCJNE       MACRO      REGISTRO,DATO,DIRECC
        DECF  REGISTRO,1
        MOVLW      DATO
        SUBWF      REGISTRO,0
        BTFSSSTATUS,Z
        GOTO DIRECC
        ENDM

;***** COMPARA Y SALTA SI ES MENOR *****
CJL    MACRO      REGISTRO,DATO,DIRECC
        MOVLW      DATO
        SUBWF      REGISTRO,0 ; W = REGISTRO - W
        BTFSSSTATUS,C    ; ¿C='1'?
        GOTO DIRECC
        ENDM

;***** COMPARA Y SALTA SI ES MENOR IGUAL *****
```

```
CJLE  MACRO       REGISTRO,DATO,DIRECC
      MOVLW       DATO
      SUBWF       REGISTRO,0 ; W = REGISTRO - W
      BTFSSSTATUS,C    ; ¿C='1'?
      GOTO DIRECC
      BTFSC       STATUS,Z    ; ¿Z='0'?
      GOTO DIRECC
      ENDM

;***** COMPARA Y SALTA SI NO ES MENOR *****
CJNL  MACRO       REGISTRO,DATO,DIRECC
      MOVLW       DATO
      SUBWF       REGISTRO,0 ; W = REGISTRO - W
      BTFSC       STATUS,C    ; ¿C='0'?
      GOTO DIRECC
      ENDM

;***** COMPARA Y SALTA SI ES MAYOR O IGUAL *****
CJGE  MACRO       REGISTRO,DATO,DIRECC
      MOVLW       DATO
      SUBWF       REGISTRO,0 ; W = REGISTRO - W
      BTFSC       STATUS,C    ; ¿C='0'?
      GOTO DIRECC
      ENDM

;***** ESUNO *****
ESUNO       MACRO       REGISTRO,BIT
      LOCAL       NO_ES_UNO_1
NO_ES_UNO_1
      BTFSSREGISTRO,BIT
      GOTO NO_ES_UNO_1
      ENDM

;***** ESCERO *****
ESCERO      MACRO       REGISTRO,BIT
      LOCAL       NO_ES_CERO_1
NO_ES_CERO_1
      BTFSC       REGISTRO,BIT
      GOTO NO_ES_CERO_1
      ENDM

;***** SALTAR *****
JMP   MACRO       DIRECCION
      GOTO DIRECCION
      ENDM

;***** SALTA SI ES CERO *****
JZ    MACRO       DIRECCION
      BTFSC       STATUS,Z
      GOTO DIRECCION
```

```
        ENDM

;***** SALTA SI NO ES CERO *****
JNZ     MACRO       DIRECCION
        BTFSSSTATUS,Z
        GOTO DIRECCION
        ENDM

;***** SALTA SI CARRY *****
JC      MACRO       DIRECCION
        BTFSC       STATUS,C
        GOTO DIRECCION
        ENDM

;***** SALTA SI NO CARRY *****
JNC     MACRO       DIRECCION
        BTFSSSTATUS,C
        GOTO DIRECCION
        ENDM

;***** SALTAR SI BIT ESTA SETEADO *****
JBS     MACRO       REG, BIT, ADR
        BTFSC       REG, BIT
        GOTO ADR
        ENDM

;***** SALTAR SI BIT ESTA ACLARADO *****
JBC     MACRO       REG, BIT, ADR
        BTFSSREG, BIT
        GOTO ADR
        ENDM

;***** OBVIAR SI BIT ESTA SETEADO *****
SBS     MACRO REG, BIT
        BTFSSREG, BIT
        ENDM

;***** OBVIAR SI BIT ESTA ACLARADO *****
SBC     MACRO REG, BIT
        BTFSC       REG, BIT
        ENDM

;***** COMPARA REGISTRO *****
COMPARA   MACRO       REG1, CONSTANTE
        MOVLW       CONSTANTE
        XORWF       REG1,W
        ENDM

;*************************************************
UMUL1616L       MACRO
```

```
        MOVLW   0X08
        MOVWF   CONTADOR
LOOPUM1616A
        RRF     DATOA_H, F
        BTFSC   STATUS,C
        GOTO    ALUM1616NAP
        DECFSZ  CONTADOR, F
        GOTO    LOOPUM1616A
        MOVWF   CONTADOR
LOOPUM1616B
        RRF     DATOA_L, F
        BTFSC   STATUS,C
        GOTO    BLUM1616NAP
        DECFSZ  CONTADOR, F
        GOTO    LOOPUM1616B

        CLRF    DATOC_L
        CLRF    DATOC_H
        RETLW   0X00

BLUM1616NAP
        BCF     STATUS,C
        GOTO    BLUM1616NA
ALUM1616NAP
        BCF     STATUS,C
        GOTO    ALUM1616NA
ALOOPUM1616
        RRF     DATOA_H, F
        BTFSS   STATUS,C
        GOTO    ALUM1616NA
        MOVF    TEMPB1,W
        ADDWF   DATOC_H, F
        MOVF    TEMPB0,W
        BTFSC   STATUS,C
        INCFSZ  TEMPB0,W
        ADDWF   DATOC_L, F
ALUM1616NA
        RRF     DATOC_L, F
        RRF     DATOC_H, F
        RRF     DATOB_L, F
        DECFSZ  CONTADOR, F
        GOTO    ALOOPUM1616

        MOVLW   0X08
        MOVWF   CONTADOR
BLOOPUM1616
        RRF     DATOA_L, F
        BTFSS   STATUS,C
        GOTO    BLUM1616NA
        MOVF    TEMPB1,W
```

```
        ADDWF   DATOC_H, F
        MOVF    TEMPB0,W
        BTFSC   STATUS,C
        INCFSZ  TEMPB0,W
        ADDWF   DATOC_L, F
BLUM1616NA
        RRF     DATOC_L, F
        RRF     DATOC_H, F
        RRF     DATOB_L, F
        RRF     DATOB_H, F
        DECFSZ  CONTADOR, F
        GOTO    BLOOPUM1616
        ENDM
```

```
; ***************************************************************
; UART.ASM
; ***************************************************************

INICIA_TX
        BANCO       1
        BCF    TRISB,2                  ; RB2 COMO SALIDA
        MOV  TXSTA,B'00100100' ; MODO ASINCRONO / TX ENABLE / 8 BITS /
BRGH = 1
        MOV  SPBRG,.129          ; 9600 BPS
        BANCO       0
        BSF    RCSTA,7                  ; HABILITA PUERTO SERIAL
        BCF    PIR1,TXIF
        RETURN

INICIA_RX
        BANCO       1
        BSF    TRISB,1                  ; RB1 COMO ENTRADA
        BSF    TXSTA,BRGH               ; BRGH='1'
        MOV  SPBRG,.129          ; 9600 BPS
        BANCO       0
        MOV  RCSTA,B'10010000' ; MODO ASINCRONO / RX ENABLE / 8 BITS
        BCF    PIR1,RCIF
        RETURN
;*********************************
; TRANSMITE EL DATO QUE ESTÁ EN W *
;*********************************

TX_DATO
        MOVWF       TXREG
ESPERA_USART
        BTFSSPIR1,TXIF           ; ESPERA A QUE EL BUFFER ESTE VACIO.
        GOTO ESPERA_USART
        BCF    PIR1,TXIF
        RETURN
```

```
; ***************************************************************
;
; RETARDO.ASM
; ***************************************************************
;
        CBLOCK
                CONTADOR1       ;REGISTRO USADO POR RETARDO
                CONTADOR2       ;REGISTRO USADO POR RETARDO
        ENDC

;***************************************************************
;
T0              EQU   0X82  ;CONSTANTE PARA GENERAR APROX. 25MS
T4.6MS          EQU   0X06*4        ;CONSTANTE PARA GENERAR 4,6MS
T15.4MS         EQU   0X14*4        ;CONSTANTE PARA GENERAR 15,4MS
T200US          EQU   0X01  ;CONSTANTE PARA GENERAR APROX. 200US

;***************************************************************
;
RETARDO:
        CLRF  CONTADOR1         ;ACLARA CONTADOR1
RET     DECFSZ      CONTADOR1,F     ;
        GOTO        RET
        DECFSZ      CONTADOR2,F
        GOTO RET
        RETURN

RET50MS:                    ;RETARDO 50 MS
        MOVLW       T0
        MOVWF       CONTADOR2
        CALL RETARDO
        CALL RETARDO
        RETURN

RET100MS:                   ;RETARDO 100 MS
        CALL RET50MS
        CALL RET50MS
        RETURN

RET300MS:                   ;RETARDO 300 MS
        CALL RET100MS
        CALL RET100MS
        CALL RET100MS
        RETURN

RET600MS:                   ;RETARDO 600 MS
        CALL RET300MS
        CALL RET300MS
        RETURN

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DEMORA91  movlw    .112     ; 1 set numero de repeticion
      movwf    PDel0    ; 1 |
```

```
PLoop0  clrwdt              ; 1 clear watchdog
        decfsz    PDel0, 1  ; 1 + (1) es el tiempo 0  ?
        goto      PLoop0    ; 2 no, loop
PDelL1  goto PDelL2         ; 2 ciclos delay
PDelL2
        return              ; 2+2 Fin.

DEMORA96  movlw     .118      ; 1 set numero de repeticion
        movwf     PDel1     ; 1 |
PLoop1  clrwdt              ; 1 clear watchdog
        decfsz    PDel1, 1  ; 1 + (1) es el tiempo 0  ?
        goto      PLoop1    ; 2 no, loop
PDelL4  goto PDelL3         ; 2 ciclos delay
PDelL3  clrwdt              ; 1 ciclo delay
        return              ; 2+2 Fin.
;=============================================================
;
; Delay Loop
;
; This function is designed to delay for 100uS times the number
; of times through the loop
; Once through :
;               1+91+2+1+2+1+2 = 100
;
; Twice through
;               1+91+2+1+1+2+96+1+2+1+2 = 200
;                       ******
; Thrice through
;               1+91+2+1+1+2+96+1+1+2+96+1+2+1+2 = 300
;                       ******** ******
; "N" times through (n * 100) cycles.
;
; NOTE: This will have to be changed when you change the
; frequency from 4Mhz, but for the LAB-X3 it works great!
;
DELAY:
        MOVWF   DLY_CNT         ; Store delay count       (1)
        nop
        nop
        nop
        nop
        CALL DEMORA91
        GOTO    D2_LOOP         ; Check to see if it was 1 (2)
D1_LOOP:
        nop
        nop
        nop
        nop
        nop
        nop
```

```
        nop
        nop
        CALL DEMORA96
D2_LOOP:
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        DECF    DLY_CNT,F       ; Subtract 1            (1)
        nop
        nop
        nop
        nop

        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        INCFSZ  DLY_CNT,W       ; Check for underflow   (2)
        GOTO    D1_LOOP         ; Not zero so loop      (2)
        NOP
        nop
        nop
        nop
        nop
        RETURN

;-------------------------------------------------------------
DEMORAE  movlw    .6        ; 1 set numero de repeticion  (B)
        movwf    PDel0E     ; 1 |
PLoop1E  movlw    .207      ; 1 set numero de repeticion  (A)
        movwf    PDel1E     ; 1 |
PLoop2E  clrwdt             ; 1 clear watchdog
        decfsz   PDel1E, 1  ; 1 + (1) es el tiempo 0  ? (A)
        goto     PLoop2E    ; 2 no, loop
        decfsz   PDel0E,  1 ; 1 + (1) es el tiempo 0  ? (B)
        goto     PLoop1E    ; 2 no, loop
PDelL1E  goto PDelL2E       ; 2 ciclos delay
PDelL2E  clrwdt             ; 1 ciclo delay
        return              ; 2+2 Fin.
;-------------------------------------------------------------
```

BIBLIOGRAFÍA

[1] Enciclopedia Encarta 2001. Microsoft Corporation.

[2] http://www.fotohobby.com.ar/toma1.htm

[3] Especificaciones Técnicas de cámaras de video: http://www.avtech.com.tw/NewProduct/promotion/AVC667.asp?LanguageType=1

[4] Tesis Doctoral de F. Martín Rodríguez (Director: X. Fernández Hermida): Cap. 1 y Cap. 2 "Reconocedor Automático de Matrículas de Automóviles": http://wgpi.tsc.uvigo.es/wgpi_publicaciones.htm

[5] "Image Processing Toolbox User's Guide For use with MATLAB". Versión 2. COPYRIGHT 1993 - 1998 by The MathWorks, Inc.

[6] Fernando Martín Rodríguez, Xulio Fernández Hermida. "Un Nuevo Método, Basado en Morfología, para la Localización de Matrículas": http://wgpi.tsc.uvigo.es/wgpi_publicaciones.htm

[7] Aitzol Zuloaga Izaguirre, José Luis Martín González. "Visión artificial dinámica determinación de movimiento a partir de secuencias de imágenes": http://www.geocities.com/CapeCanaveral/8482/docu003.pdf

[8] "Introducción al Procesado de Imagen" : http://wgpi.tsc.uvigo.es/libro/procesim/node1.htm

[9] R.C. González y R.E. Wood, "Digital Image Processing" Addison-Wesley Publishing Company, Inc., 1993.

[10] Domingo Mery, "Visión Artificial". VISONA.pdf . Cap. 2: http://www.diinf.usach.cl/~dmery/imagenes/

[11] Maurizio Pilu, "Deskewing perspectively distorted documents: An approach based on perceptual organization", Hewlett-Packard Laboratories, May 2nd, 2001.

[12] "Estado del arte de la segmentación de imágenes". CAP2.pdf.

[13] N. Otsu. "A Threshold Selection Method for Gray Level Histograms". IEEE Transactions on System, Man and Cybernetics. Enero, 1979.

[14] Ping-Sung Liao, Tse-Sheng Chen And Pau-Choo Chung, “A Fast Algorithm for Multilevel Thresholding”. Journal Of Information Science And Engineering 17, 713-727 (2001).
http://www.iis.sinica.edu.tw/JISE/2001/200109_01.pdf

[15] Esmeralda Lozano Gómez , Javier Cardenal Escarcena , Jorge Delgado García, “Inventario Semiautomático Del Olivar A Partir De Ortofotografías Digitales”. XIV Congreso Internacional de Ingeniería Gráfica Santander, España – 5-7 junio de 2002.
http://departamentos.unican.es/digteg/ingegraf/cd/ponencias/173.PDF

[16] Segmentación de imágenes
http://gps-tsc.upc.es/imatge/Main/TEI/segmentacion.pdf.

[17] L. García-Pérez, M.C.García-Alegre, J. Marchant, T. Hague, “Dynamic Threshold Selection for Image Segmentation of Natural Structures Based Upon a Performance Criterion”
http://www.iai.csic.es/users/gpa/postscript/Montpellier_202.pdf

[18] M. D. Garris. "Component-Based Handprint Segmentation Using Adaptive Writing Style Model". N.I.S.T. Internal Report Number 5843.

[19] Chi Fang, “Deciphering Algorithms for Degraded Document Recognition”. Tesis Doctoral. Jul. 1997.

[20] Ruini Cao, Chew Lim Tan “Separation of Overlapping Text from Graphics”. School of Computing, National University of Singapore.

[21] P.J. Grother. "Karhunen-Loéve Extraction for Neural Handwritten Character Recognition". Actas de Applications of Artificial Neural Networks III, Vol 1709, pp 155-166. SPIE, Orlando (U.S.A.). Abril, 1992.

[22] “Neural Network Toolbox User’s Guide” COPYRIGHT 1992 - 2002 by The MathWorks, Inc.

[23] “Creating Grafical User Interfaces”. Versión 6. COPYRIGHT 2000 - 2002 by The MathWorks, Inc.

[24] Mark N. Horenstein, “Circuitos y Dispositivos Miroelectrónicos”. Prentice-Hall Latinoamericana, S.A., 1997.

[25] Data Sheet: “Low Power Quad Operational Amplifiers STMicroelectronics LM124, LM224, LM324”, Diciembre 2001.

[26] Data Sheet: “FLASH-Based 8-Bit CMOS Microcontroller”, PIC16F62X.
© Microchip Technology Inc., 2003.

[27] Data Sheet:”MAXIM +5V-Powered, Multichannel RS-232 Drivers/Receivers”. © Maxim Integrated Products, 2000.