arobably

TECNOLÓGICO DE MONTERREY®

Campus Ciudad de México

Escuela de Graduados en Ingeniería y Arquitectura

Maestría en Ciencias Computacionales

Reconocimiento de Placas de Automóvil en Secuencia de Imágenes

Autor:
Furahi

Director de la tesis:
Dr. Patricia Rayón Villela

Septiembre 2006

“Reconocimiento de placas de automóvil en secuencia de imágenes”

por

Furahi

ha sido aprobada

septiembre de 2006

Aprobada por la comisión de tesis

Dr. Alvaro de Albornoz, Presidente

Dr. Bárbaro Ferro, Sinodal

Dra Patricia Rayón Villela, Sinodal

Aceptada

Titular de la maestría

Director del Programa de maestría

Dedicatoria.

Al lucero que desde niño ha alumbrado mi camino, el báculo que orienta mis pasos, el bordón sobre el que siempre he dejado caer mi peso, para volar hasta aquí, hasta hoy: mi Uma

A mi Feba, quien como herrero ha forjado conmigo, más que un yunque pesado, la flecha que corta los tiempos, las distancias y las adversidades. Desde escribir mi nombre, hasta firmar mi título, a mi abuelita, por sus enseñanzas.

Agradecimientos.

En esta parte de mi tesis, debo dar las gracias a aquellos sin quienes no hubiera podido llegar hasta donde estoy; sin afán de llevar la contra, yo voy a pedirlas.

Le pido a Dios que colme de Gracias a mi mamá, ella mejor que nadie sabe -aunque a veces pretenda que no- que es el ser a quien más quiero en la vida. Nunca podré agradecerle lo suficiente por acogerme desde niño y hasta de grande entre su bondadoso e inigualable abrazo; por recibirme de un salto a cada encuentro; por su tesón incansable para sacarme siempre adelante -con o sin ayuda, ya se ha visto-; por transmitirme su inmensurable amor a los animales y con ello enseñarme cuan bello es el mundo; por tener siempre un lugar en su cama para quien ayer se acostó niño y hoy, en su honor, se levanta hombre; por ser, aún sin intención, el tierno retoño del que siempre me he prendido para no caer; por hacer de tripas corazón para sobreponerse a ella misma, y amarme y aceptarme e incluso tratar de orientarme intacto tal y como soy; por alumbrarme la vida con sus ojitos diáfanos: Mucho le he de pedir a Dios que no derramen demasiadas lágrimas por mi ahora que viviré lejos del cobijo de sus cuidados, pues aunque no quiera enterarse, mi vida, mi corazón y mis pensamientos son lo que en la casa dejo incondicionalmente....por supuesto que tendré que venir MUY seguido a reconectarme conmigo mismo, como pretexto para besar la frente de su único verdadero custodio: Umita, soy un hombre bueno.

Quisiera poder alcanzar suficientes bendiciones, para corresponder a todas aquellas con que toda mi vida mi Feba adorada me ha protegido desde recién nacido. Ya suficiente bendición es, tenerte conmigo por todo este cenagoso camino, más aún ahora es entregarte este nuevo título, y reconocerte que es gracias a tu esfuerzo, aunada a mi incontrolable hambre de conocimiento, que este nuevo camino se tiende frente a mis pies; seguirlo me llevará lejos de la seguridad de tu techo, pero sin lugar a dudas habrá de ser la senda para enaltecer la dedicación que has puesto en tu nieto. Feba, soy un hombre grande.

Aunque ya su suerte ha sido un derroche de virtudes, también anhelo que se eternice la felicidad de mi perro: En mis oraciones pido por Pulque. Porque es él el único que ha sabido escuchar mi llanto

y mi pesar en silencio, pues no me juzga ni recrimina mis actos, pues siempre me aturde con su discurso ladrado, pues soporta con ejemplar valentía mis cariños y mimos -por demás toscos-, porque su alma es inversamente proporcional a la longitud de sus patas y aún sin proponérselo es capaz de contagiarme de su radiante alegría, porque siempre me tiene guardados besos, porque me refresca los días con su cenagosa cauda, por aprender lo que he procurado enseñarle, siendo yo quien más de él se ha enriquecido. Pulque es mi mejor amigo.

Sin lugar a dudas no puedo dejar pasar esta ocasión para agradecerle a Paty Rayón, a quien tuve el placer de conocer desde que comenzaba yo ha hacer mis pininos en ésta subyugante área del saber, y me ha llevado de la mano, orientando mi pensamiento, impulsándome a llegar al conocimiento mismo a través de mis propios medios: Hoy hemos conquistado juntos éste reto, y entregamos en equipo un legado valioso a la Institución que ha sido plataforma para nuestro qué hacer científico. Gracias Paty, éste logro también es tuyo.

Resumen.

El proceso de reconocimiento de placas se divide en dos grandes etapas: la localización y el reconocimiento. En este trabajo de tesis se abordan estos dos problemas, y se propone una metodología de solución para ambos.

En este documento se propone una solución para localizar placas de automóviles del Distrito Federal, México. Las imágenes en las que se reconocerá la placa serán tomadas desde un vehículo en movimiento, y serán de vehículos que también estén en movimiento.

En la etapa de localización se realizaron varias pruebas, entre ellas la detección de lineas parea poder decidir si entre pares de lineas horizontales y verticales, y de acuerdo a la relación de alto y ancho, se decidía se encontraba presente una placa.

Otra prueba para la etapa de localización fue la de segmentar la imagen por color; por medio de la búsqueda de zonas en la misma que contuvieran los colores de las placas y que tuvieran una forma similar a la que debe tener una placa.

Para la etapa de reconocimiento, se utilizaron dos clasificadores, uno para letras y otro para números. Las características que utilizan los localizadores son la presencia de líneas verticales, horizontales y diagonales al igual que la distancia entre diferentes puntos y los caracteres.

El método basado en líneas para localizar las placas resultó ser muy sensible a ruido en la imagen y desperfectos en la misma placa. Se probaron dos métodos diferentes para la segmentación por color; uno utilizando redes neuronales y el otro geometría. El primero dio buenos resultados bajo diferentes condiciones de iluminación; el segundo generó aún mejores resultados, pero únicamente bajo condiciones de iluminación más o menos constantes.

Los métodos propuestos pueden ser utilizados por agencias de procuramiento de la ley para dar seguimiento a infracciones y delitos que actualmente permanecen impunes por la dificultad de rastrear un gran número de placas. Gracias a la baja sensibilidad del sistema al *motion blur*, éste se puede montar en vehículos policíacos con una cámara al frente para verificar las placas de todos los vehículos que crucen su camino.

Contenido

Índice de Figuras.

Índice de Tablas.

Índice de Fórmulas.

1. Introducción.

1.1. El Problema de Reconocimiento de Placas.

El problema de reconocimiento de placas de automóvil es un problema de actualidad en el área de la visión por computadora. Sus aplicaciones varían desde control y *monitoreo* de tráfico, *autenticación* para entrar a edificios privados; hasta sistemas de pago en estacionamientos automatizados.

El problema se resuelve típicamente en dos etapas principales: localización y detección de la placa y el reconocimiento de sus caracteres.

Las técnicas utilizadas hasta el momento para la detección y localización de la placa varían ampliamente; todas presentan sus ventajas y desventajas, y usualmente, son aplicadas a un país en específico. En general, los métodos funcionan correctamente únicamente bajo el ambiente de trabajo que los autores consideraron, y suelen ser menos eficientes en otros ambientes. En particular la forma, color, borde (o falta de) y contraste de la placa influyen enormemente en el desempeño de algún método.

Para el reconocimiento de caracteres las variantes suelen ser el tipo de símbolos que se intenta leer, tales como caracteres del alfabeto occidental, números arábigos, tipografías orientales, etc. Aún así, es común enfrentarse con retos similares tales como deformaciones por rotación y perspectiva, así como la presencia de caracteres incompletos o traslapados.

1.2. Problema a Resolver.

En este trabajo de tesis se desea abordar el problema de detección, localización y reconocimiento de placas.

El problema de detección y localización de placas se puede plantear de la siguiente manera:

Dada un imagen que captura a un coche en movimiento, determinar si en esa imagen esta presente una o varias placas, o no se encuentra ninguna placa, este problema aborda la detección de placas, que en muchos de los trabajos no es abordado, ya que parten del supuesto que en la imagen siempre esta presente una placa. En caso de que se hayan detectado placas, especificar la posición de éstas en la imagen, esto es, la localización de la misma.

Este problema se vuelve complejo principalmente por el ambiente en que se esta trabajando, como se menciona a continuación.

El método propuesto debe reconocer placas de la Ciudad de México desde un automóvil en movimiento, tomando imágenes de otro automóvil que también se está moviendo. Esto presenta algunas características particulares al problema.

Las placas de la ciudad de México son anchas, con una razón de alto contra ancho de aproximadamente 2:1. El fondo es de color beige, con una imagen de un ángel en la mitad. Las letras son de color verde y no presentan bordes.

Un número significativo aunque no mayoritario de los coches utilizan un portaplacas de ornato. La mayor parte de éstos es negra, pero hay algunos de otros colores, especialmente plateados. Dependiendo del portaplacas, el área visible de la placa puede reducirse hasta un a razón de 3:1

Dependiendo de las condiciones de iluminación, que debido a la naturaleza misma del problema, son variadas; en un gran número de casos el contraste entre los números y el fondo de la placa es muy bajo. Se llega a dar el caso de que un valor de intensidad con cierta luz sea del fondo, y con una variación muy pequeña de la luz se vuelva de frente. Esto descarta ciertos métodos de localización que se basan en la asunción de que la placa es una zona de alto contraste [ZUN00].

La falta de borde y el color poco saturado provocan que en casos en los que el color del vehículo sea similar al de la placa, se pierda o sea demasiado débil la información de los bordes de la placa, lo que hace que los métodos que se basan en la información de bordes de la placa, como [ZHE04] fallen en varios casos.

El movimiento de los vehículos, el que captura la imagen y el coche a ser identificado, se asume que será con velocidades similares para ambos. Por ésto, la distorsión por movimiento es mínima y no impacta el desempeño, como se probó de manera experimental.

Existe un gran número de placas en malas condiciones en la ciudad de México. El sistema no

contempla estos casos. La información de color le ayuda a compensar ligeras deformaciones en la placa; sin embargo, hay deformaciones tan grandes al igual que otras alteraciones como las caída de la pintura de la placa, instalación errónea, portaplacas rotos, así como en algunos casos donde la placa se encuentra traslapada con otros objetos que cubren parcialmente la placa que impiden el funcionamiento del sistema. En varios de los casos mencionados anteriormente, las placas no pueden ser leídas por personas a simple vista.

No se contempla en esta etapa un método para diferenciar placas de diferentes estados; todas se asumen ser del DF. En la mayoría de los casos el color es suficiente para que el sistema no intente leer placas que no debe; sin embargo, en ciertos casos como el de las placas del estado de Nuevo León, el sistema puede intentar leer la placa sin darse cuenta del origen de la misma.

Se espera que el método propuesto pueda funcionar en días despejados o nublados, siempre y cuanto haya suficiente luz solar. Durante la investigación, ciertas partes fueron optimizadas para días despejados y soleados, con el sol cerca de la parte más alta del cielo; sin embargo, dichas optimizaciones se pueden reproducir para otras condiciones siempre y cuando caigan en las descritas al comienzo de este párrafo.

Las condiciones adversas que no se contemplan son las de neblina excesiva, lluvias, tormentas, granizo u otros fenómenos naturales o artificiales que limiten la visibilidad de la cámara o de la placa (por ejemplo manchas de lodo sobre la placa o cámara).

Una vez detectada la placa el problema de reconocimiento se puede plantear de la siguiente forma:

Dada una subimagen de una placa que ha sido previamente detectada, identificar la placa, esto es, los dígitos y caracteres que tiene la placa.

Este problema a su vez es subdividido en dos etapas: la segmentación y el reconocimiento. El problema de segmentación se agrava debido principalmente a las sombras, así como a los portaplacas, que en varios casos ocasionan el traslape de los caracteres. Una vez que estos problemas han sido abordados, el reconocimiento de caracteres se lleva a cabo.

1.3. Objetivo.

El objetivo general del trabajo es llevar a cabo la detección de placas de automóvil en movimiento en la ciudad de México.

Los Objetivos Particulares:

a. Proponer una metodología para llevar a cabo la detección y localización de placas de automóviles en movimiento, en la Ciudad de México.

b. Proponer una metodología para llevar a cabo la segmentación y el reconocimiento de caracteres.

1.4. Metodología Propuesta.

En esta tesis se propone un método híbrido de detección, localización y reconocimiento de placas de automóvil. Para la etapa de localización se contemplaron tres métodos diferentes: El primero es la utilización de detección de bordes verticales similar a Zheng [ZHE04], combinada con información de la razón alto/ancho de la placa. Los otros dos métodos llevan a cabo la segmentación por color, donde una región de decisión de color (RDC) es definida y finalmente una red neuronal para identificar la forma de los diferentes objetos es utilizada. El segundo método utiliza una red neuronal para llevar a cabo la segmentación por color, mientras que el tercero utiliza un algoritmo genético para generar pirámides triangulares en el espacio RGB que definen regiones de decisión de color (RDC), que permitan llevar a cabo la segmentación.

La segmentación de caracteres de la placas utiliza segmentación por color, combinada con una función heurística para determinar el conjunto de colores que lleva a cabo una mejor segmentación de los caracteres y el fondo de la placa.

Finalmente, en la etapa de reconocimiento de caracteres, se detectan puntos de interés, se aplica una transformada de Hough reducida, y se obtiene un vector de 20 características que es alimentado a una red neuronal que permite clasificar letras y números

1.5. Contribuciones.

En la literatura presentada se asume que la imagen de entrada siempre tiene una placa. En este trabajo de tesis dada una imagen el sistema es capaz de decidir si hay o no una o varias placas en la imagen.

En la mayoría de los trabajos sólo es detectada una placa, aunque aparezcan más.

El reconocimiento para placas de automóvil en la Ciudad de México no se ha llevado a cabo.

Se desarrolló un algoritmo de *umbralización* con dos umbrales basado en el algoritmo óptimo de umbralización.

Se desarrolló un *kernel* para detectar bordes verticales basado en el operador de Sobel.

1.6. Organización del Reporte.

El capítulo 1 de este trabajo contiene una introducción al problema, tanto en general como en específico el que se va a resolver. Esto incluye objetivos generales y específicos al igual que contribuciones.

El segundo capítulo contiene el estado del arte. Presenta un sondeo amplio de técnicas encontradas en la literatura tanto para la localización como para la identificación de placas de automóviles al rededor del mundo; además de mencionar el ambiente bajo el que operan cada una de las técnicas revisadas en el capítulo.

El tercer capítulo contiene el marco teórico, es decir; los fundamentos sobre los que se construye la tesis. Contiene dos secciones principales: visión por computadora y reconocimiento de patrones. En la parte de visión por computadora explica algoritmos de mejora de contraste, *binarización*, filtros lineales y no lineales y detección de líneas. En la parte de reconocimiento de patrones explica el funcionamiento y teoría detrás de las redes neuronales y algoritmos genéticos.

El cuarto capítulo explica la metodología utilizada en la tesis. Explica a grandes rasgos las herramientas del marco teórico que se usaron en cada etapa.

El quinto capítulo describe a detalle los tres métodos propuestos para la localización de placas: Uno basado en la detección de líneas y los otros dos basados en segmentación por color. Uno usando redes neuronales y el otro algoritmos genéticos y geometría para la clasificación de los colores.

El sexto capítulo describe el método desarrollado para la segmentación de los caracteres de la placa y su lectura.

El séptimo capítulo contiene los resultados de los métodos desarrollados y probados en los capítulos 5 y 6.

2. Estado del Arte.

El problema de reconocimiento de placas de automóviles suele ser dividido en dos grandes partes: localización de placas y reconocimiento de caracteres; con una variedad de métodos descritos en la literatura para ambos.

2.1. Localización de Placas.

Para la localización, existen técnicas como detección de bordes verticales [ZHE04], clasificación de color usando redes neuronales [RYU94], *cuantización* de vectores [ZUN00], lógica difusa [ZIM97]; al igual que la búsqueda directa de secuencias de caracteres en la imagen fuente [LIM98].

Un gran número de las propuestas que se encuentran en la literatura, especialmente para la localización de las placas, no son comparables directamente debido a diferencias entre las placas de las diferentes localidades alrededor del mundo, y de los diferentes medio ambientes en los que se obtienen las imágenes que contienen las placas.

2.1.1. Extracción de Bordes Verticales.

Zheng *et al*. [ZHE04] propone un método de reconocimiento de placas utilizando información de bordes de la placa. Como se muestra en la tabla 1 en la sección 2.3, Zheng utiliza placas largas, con fondo negro y letras y bordes blancos.

El único paso de preprocesamiento que aplica a las placas es un algoritmo de mejora local de contraste de su propia creación en el que resalta el contraste de las zonas con poco contraste, dejando las de alto contraste sin modificación. El algoritmo se explica a detalle más adelante.

Zheng primero segmenta la imagen utilizando un operador de Sobel vertical convencional., *binariza* la imagen dividiendo el histograma en 75% fondo / 25% bordes.

A continuación reduce el ruido de la imagen de bordes analizando la extensión del borde en tres pasos: Primero recorre la imagen de arriba hacia abajo y de izquierda a derecha y registra el inicio de cada borde; después, recorre la imagen de abajo para arriba y derecha a izquierda, registrando de nuevo el largo de los bordes; finalmente calcula la longitud real del borde utilizando ambos valores. Elimina

cualquier borde que sea demasiado largo, asumiendo que es información del fondo de la imagen; o demasiado corto, asumiendo que es ruido. Después recorre la imagen dos veces más, de arriba a abajo y izquierda a derecha y de abajo hacia arriba y derecha a izquierda; , acumulando la información de bordes junto con la de sus vecinos. Finalmente recorre la imagen una vez más, dejando los bordes con valores mayores intactos, y únicamente preservando los menores en caso de que sus dimensiones estén dentro de un rango de dimensiones máximas y mínimas esperadas para las placas.

Para la localización de la placa recorre una ventana que es de un tamaño ligeramente superior al tamaño en el que se asume la placa será en las imágenes. Cuenta la cantidad de bordes dentro de la ventana y guarda esta información en una imagen adicional. Finalmente analiza esta imagen y determina que la placa se encuentra en una zona rectangular de cierto tamaño donde los valores de éste conteo sean altos.

2.1.2. Clasificación de Color Usando Redes Neuronales.

Ryung [RYU94] desarrolló un método para localizar placas basado en la suposición de que las placas son una única zona de las imágenes donde se combinan los colores del fondo de la placa con los de las letras.

Ryung procesa imágenes de 256 colores, convirtiéndolas al modelo HLS (*Hue, Lightness, Saturation*) para su trabajo con ellas.

Después, Ryung recorre la imagen y alimenta la información de HLS del píxel visitado, al igual que sus ocho vecinos, a una red neuronal con 24 nodos de entrada, 30 nodos en una capa oculta y 4 nodos de salida. Las salidas de la red neuronal representan los colores verde, rojo, blanco y otros. Elije la salida con el valor mayor para representar al píxel visitado.

A continuación calcula el histograma de rojo, verde y blanco línea por línea y columna por columna. Se utiliza información de las proporciones de la placa y de los caracteres en conjunto con estos histogramas para hacer la localización final de la placa.

2.1.3. *Cuantización* de Vectores.

Zunino [ZUN00] utiliza una muestra de imágenes de placas para minimizar el error en representar placas a través de *cuantización* de vectores.

Zunino utiliza un sistema externo que detecta vehículos en las imágenes y toma una fotografía,

con la posición y ángulo siendo conocidos. Las placas se asumen ser de cierto tamaño, tipografía, de colores de fondo muy brillantes o muy obscuros y se esperan condiciones de iluminación que sean lo más constante posible.

El primer paso es el codificar la imagen utilizando *cuantización* de vectores. Se asume que la placa es una región de alto contraste, así que debe generar bloques pequeños al codificarla. Una vez codificada toda la imagen se recorre para encontrar los bloques pequeños; y una vez localizados éstos se analiza su color de fondo para determinar si es muy brillante o muy obscuro y se clasifican como zonas de placa de así serlo. Se buscan regiones de zonas de placa que mantengan el tamaño y forma de la placa, a las que se llama tiras. Se asume que las tiras (y las placas) serán de un tamaño constante porque se asume que se conoce la distancia del sensor al vehículo. Cada tira es premiada o castigada de acuerdo a qué tanto se apega a la norma; es decir, a las condiciones esperadas de tamaño, proporciones y forma de las placas.

Finalmente se elije a la tira con mayor calificación como región de placa.

2.1.4. Lógica Difusa.

Zimic [ZIM97] utiliza reglas intuitivas de lógica difusa basadas en la percepción humana.

Las reglas que utiliza son:

- Área rectangular brillante dentro de las cuales hay algunas zonas obscuras
- Ubicada en la región media y baja de la imagen
- El borde de la placa es brillante
- La dimensión de la placa es 53x12cm

La imagen es particionada en bloques del tamaño esperado del que será la placa. Cada bloque es primero evaluado de acuerdo a la función “área rectangular brillante con zonas obscuras”. Para ésto, primero se encuentran los píxeles que entran en la clase “brillante” y después se analiza su tamaño (no por medio de un conteo de píxeles sino sumando su aptitud de acuerdo a la función de clase). Se repite el proceso para los píxeles obscuros.

A continuación se evalúa la calificación de los bloques a partir de la segunda regla: se ubica en la parte media baja de la imagen. La aptitud de cada bloque se ajusta en base a la que obtuvo de la

primera regla, junto con la que obtiene de la segunda.

Se elije al elemento con la mejor evaluación como posible zona de placa. Para determinar la posición exacta se utiliza la tercera regla: la placa tiene bordes brillantes.

2.1.5. Búsqueda Directa de Caracteres en la Imagen.

Lim [LIM98] realiza la ubicación de la placa por medio de la búsqueda directa en la imagen de los caracteres que la componen.

Inicialmente busca regiones separadas de la imagen, asumiendo que los caracteres no se traslapan y siempre están separados en una placa. Después ejecuta su algoritmo de reconocimiento de los mismos, basado en *esqueletización* e intersecciones y descrito en la siguiente sección. Finalmente analiza la posición de los caracteres identificados para determinar cuales podrían conforman una placa, por ejemplo por ser una secuencia en línea horizontal, uno tras el otro; y selecciona los más probables.

2.1.6. Información de Bordes de Color y Mapas Difusos.

Chang [CHA04] propone un método en el cual se utiliza información de color para obtener los bordes de las placas, y éstos para identificarlas.

Chang asume que las placas son regiones de alto contraste y ricas en información de bordes. Trabaja con placas que contienen los colores blanco, negro, rojo y verde. Utiliza, por lo tanto, un detector de bordes sensible únicamente a bordes de colores negro con blanco, rojo con blanco y verde con blanco.

Para detectar los bordes negro con blanco, Chang aprovecha la característica de que la diferencia de color entre el negro y el blanco da el vector (-1,-1,-1) o (1,1,1); que tiene la cualidad de que todos los elementos son del mismo signo. El borde queda entonces definido como aquel pixel que tiene al menos un vecino con el cual el vector diferencia tiene el mismo mismo en todos sus componentes. De manera similar, los bordes rojo con blanco son definidos de acuerdo a las siguientes características: El signo de todos los componentes de el vector diferencia es el mismo, y el valor absoluto de la diferencia en el canal rojo debe ser mayor a el valor absoluto de las diferencias tanto del canal azul como del verde. Finalmente, un borde verde con blanco es definido como sigue: El signo de todos los componentes del vector diferencia es igual y el valor absoluto de la diferencia en el canal verde debe ser mayor al valor absoluto tanto del canal rojo como del azul. La magnitud de los vectores

de diferencia, expresada como el mínimo de magnitud del valor absoluto de las diferencias del canal rojo, verde y azul; es guardado.

Se generan mapas difusos de los canales Tono, Saturación e Intensidad (H.S.I por sus siglas en inglés) y de la magnitud de las diferencias, llamado mapa Ê. almacenadas en el paso anterior. Los tres mapas son integrados en uno utilizando un *agregador* de dos etapas. En la primera etapa, los mapas son unidos celda por celda, aplicando el operador de unión difusa a las celdas de los mapas Ĥ, Ŝ e Î. En la segunda etapa son combinados de manera lineal con el mapa Ê.

La placa se detecta al encontrar las zonas con valores muy elevados en el mapa resultante.

2.2. Segmentación y Reconocimiento de Caracteres.

Al igual que el problema de reconocimiento de placas, el problema específico de su lectura es subdividido en dos partes [CHA04]: separación (segmentación) de caracteres e identificación (reconocimiento) de caracteres.

Algunas técnicas comúnmente usadas para la separación de caracteres incluyen redes neuronales celulares de tiempo discreto (DTCNN) [BRU98], acumulación de histogramas horizontales y verticales a través de pendientes de rango dinámico vertical [SAL99], y conteo de intersecciones horizontales y verticales de componentes.

Para el reconocimiento de caracteres, se ha propuesto usar plantillas basadas en el valor de Jaccard [RYU94], *esqueletización*, ubicación de intersecciones y trazo de líneas [LIM98] y mapas autoorganizados de Kohonen [CHA04], entre otros.

2.2.1. Redes Neuronales Celulares de Tiempo Discreto (DTCNN).

Brugge [BRU98] utiliza una DTCNN con un *template* de variación de tiempo para segmentar los caracteres. Inicialmente utiliza un algoritmo tradicional de *umbralización* variable. Almacena la imagen original de escala de grises y la imagen *umbralizada* en los primeros tiempos de la red.

Después efectúa una operación de apertura de la imagen, erosionándola tres veces para remover las letras, y después dilatándola tres veces. Con la resta de la imagen resultante de la imagen *umbralizada* original, remueve la zona negra al rededor de la placa. Cada paso es descompuesto y transformado en DTCNN y almacenado en un tiempo más de la red.

A continuación efectúa un *convex hull* de los caracteres, iterando la imagen y marcando de negro a todos los píxeles que tengan al menos dos vecinos en conexidad 4 de color negro hasta la convergencia.

Finalmente aplica un criterio de altura mínima a los caracteres, erosionando la imagen tres veces con un *kernel* de 1x3 píxeles.

Finalmente se asume que sólo quedan los bloques representantes de caracteres en la imagen. Se reconstruyen los rectángulos de la altura original al dilatar la imagen, y estas secciones son extraídas de la imagen original como caracteres individuales.

2.2.2. Pendientes de Rango Dinámico Vertical.

Salgado [SAL99] utiliza acumulaciones horizontales y verticales del histograma para segmentar los caracteres.

Las acumulaciones horizontales las utiliza para segmentar placas que tienen más de una línea de texto; analizando dónde se encuentran los cúmulos de información en éstos.

Las acumulaciones verticales son analizadas de acuerdo a pendientes de rango dinámico vertical para separar los caracteres, en las que busca zonas continuas donde el histograma sea mayor a cero.

Para el reconocimiento de los caracteres, propuso un método que considera el ruido *motion blurness*, bajo contraste y ruido ambiental. El método funciona al proyectar diferentes características de los caracteres en varias direcciones, complementado con una estructura jerárquica heurística. Genera tres candidatos con medidas de confianza por cada caracter.

2.2.3. Valor de Jaccard.

Ryung [RYU94] utiliza el análisis del histograma acumulado vertical y horizontal para la segmentación de caracteres como hace [SAL99].

Para el reconocimiento utiliza *templates* de 10 dígitos, 15 caracteres geográficos y 84 caracteres coreanos. Normaliza el caracter obtenido de la imagen a un cuadro de 40*40 y utiliza el valor de Jaccard (ecuación 1) para hacer la comparación con los *templates*.

$$Jaccard = \frac{n11}{\{n11 + n10 + n01\}} \quad (1)$$

2.2.4. *Esqueletización* y Conteo de Intersecciones.

Lim [LIM98] propone un algoritmo de reconocimiento óptico de caracteres basado en la *esqueletización* y el conteo de intersecciones horizontales y verticales de los mismos.

Inicialmente Lim recorre la imagen de arriba hacia abajo e izquierda a derecha, marcando los píxeles que se encuentran en el contorno de los caracteres. La imagen es recorrida una vez más, contemplando únicamente los píxeles remarcados en la primera corrida. A cada uno de éstos se le asignan ocho cantidades, P_0 a P_7 , que representan a sus píxeles vecinos.

Lim creó dos conjuntos de condiciones para borrar o conservar píxeles, uno con 12 elementos y otro con dos. Si un píxel cumple con al menos una condición en cada uno de ellos entonces se remueve del conjunto de píxeles a ser borrados. Posteriormente, Lim propuso un tercer conjunto de 8 condiciones. Si un píxel cumple con al menos una condición del tercer conjunto y ciertas condiciones del segundo, el píxel debe ser marcado para ser borrado.

A continuación Lim efectúa pruebas de conectividad en los píxeles marcados para ser borrados. Si el número de vecinos de un píxel es igual a uno, se mantiene el píxel en el conjunto para borrar. De lo contrario, se analizan los vecinos del mismo y en caso de haber dos o más vecinos de lados opuestos del centro, los píxeles son marcados para conservarse; de lo contrario continúan marcados para ser borrados. Finamente se borran todos los píxeles marcados.

Para reconocer los caracteres, Lim examina las imágenes *esqueletizadas* y encuentra todas las intersecciones. A continuación, se examinan las líneas que conectan a las intersecciones, por ejemplo si siguen una curva ascendente, si es una línea recta o da una vuelta.

Se generó una base de datos con tales características de los caracteres, y la identificación finalmente es realizada por comparación directa de los caracteres leídos con la base de datos.

2.2.5. Mapas Autoorganizados de Kohonen.

Chang [CHA04] propone un proceso basado en etiquetado de componentes conectados y mapas

autoorganizados de Kohonen para la lectura de los caracteres de la placa.

Las imágenes que utiliza Chang se presuponen tomadas directamente de enfrente de la placa, es decir, no pueden estar rotadas ni deformadas por perspectiva.

Las imágenes son primero *binarizadas* con un algoritmo de *umbralización* variable. A continuación se hace un análisis de los componentes conectados, comparando su ratio de alto contra ancho contra rangos predeterminados y eliminando los objetos que no caigan dentro de los rangos. Si el número de objetos sigue siendo mayor a un cierto límite, se eliminan los objetos restantes uno por uno comenzando por el más pequeño hasta llegar al límite, o alcanzar un cambio de tamaño considerable con respecto al último objeto borrado. Una segunda etapa analiza las características de los objetos restantes para ver si se ajustan a los parámetros definidos de caracter de placa de automóvil; trabajando en conjunto con operadores borrar, unir y separar para reconstruir la placa lo más fiel posible.

Una vez separados y reconstruidos los caracteres, éstos son clasificados como letras o números de acuerdo a las características conocidas de las placas. Después, se extraen las características topológicas de los caracteres, tales como el número de huecos, puntos terminales, nodos de tres vías y nodos de cuatro vías. Las características son invariantes a transformaciones espaciales, incluyendo traslación, escala y rotación. Para compensar por características erróneas que se puedan encontrar, o que puedan faltar debido a ruido, los caracteres son compatibles con un *template* sí la diferencia en el número de huecos está en el rango de [-1,1] y la diferencia en el número de nodos de cualquier tipo está en el rango [-2,2].

A continuación los *templates* del conjunto de entrenamiento son comparados con los del caracter a ser reconocido y éste es etiquetado como aquél con el que la comparación de el mejor resultado. La comparación se basa en la red neuronal autoorganizada de Kohonen como sigue: las características del caracter a identificar son codificadas en forma de pesos en las capas intermedias de la red neuronal. Se utilizan los *templates* de entrenamiento como estímulos para la red, lo que causa que los pesos cambien; se repite la estimulación hasta que los pesos se estabilicen. El número de veces que los pesos cambian se toma como una medida de similaridad entre el caracter a ser reconocido y el representado por el *template*. El caracter se etiqueta como el caracter por el cual se necesitaron hacer menos cambios a la red antes de que ésta se estabilizara.

2.3. Ambiente.

Pan [PAN05] utiliza placas negras, largas (razón de aproximadamente 4:1 entre sus lados), con letras blancas y un delgado marco blanco alrededor de la placa; su artículo trata del reconocimiento, no de la ubicación de las placas; así que no describe el medio ambiente en el que se capturaron las imágenes, aunque si considera pequeñas rotaciones y ruido. Lim *et al* [LIM98] utiliza el mismo tipo de placas: negro con blanco y un marco. Su técnica asume que la placa ya se encuentra ubicada y segmentada, pese a que su imagen de muestra tiene un acercamiento de una placa con parte del automóvil visible. Zheng [ZHE04] también reconoce el mismo tipo de placas. Su artículo incluye la localización y algo de ruido ambiental. Brugge [BRU98] utiliza una placa larga con un fondo amarillo reflejante y letras negras en pares. Las imágenes de muestra en su artículo incluyen algo de fondo; sin embargo, la cámara está fija así que el fondo es estático salvo por la intensidad de la iluminación. Chang [CHA04] lee placas anchas (razón de aproximadamente 2:1), blancas, con números negros; Chang utiliza una variedad de ángulos complejos y fondos reales y variables. Nijhuis [NIJ95] lee placas con fondos amarillos reflejantes, números negros, largas y con un borde negro delgado. En las imágenes que utiliza se ve el automóvil completo, aunque cubre casi completamente la fotografía, sin dejar que se vea gran parte del fondo. Park [PAR99] utiliza placas negras, anchas (ratio de aproximadamente 2:1), con letras y bordes blancos. En las fotografías que muestra, el carro ocupa casi todo el espacio. Ryung [RYU94) trabaja con el mismo tipo de placas: negras, con letras y bordes blancos y razón de 2:1. Sus imágenes de prueba tienen un fondo estático. Sirithinaphong [SIR99] utiliza placas con ratio 2:1, fondo blanco con letras negras sin mucho ruido ambiental.

***Tabla 1**: Comparativo de características de placas en la literatura*

Referencia	***Aplicación***	***Forma***	***Fondo***	***Letras***	***Borde***
PAN05	Identificación	Larga	Negro	Blancas	Blanco
LIM98	Identificación	Larga	Negro	Blancas	Blanco
ZHE04	Ambos	Larga	Negro	Blancas	Blanco
BRU98	Identificación	Larga	Amarillo reflejante	Negras	Negro
CHA04	Ambos	Ancha	Blanco	Negras	Ninguno
NIJ95	Ambos	Larga	Amarillo reflejante	Negras	Negro delgado
PAR99	Localización	Ancha	Negro	Blancas	Blanco

RYU94	Ambos	Ancha	Verde	Blancas	Blanco
SIR99	Localización	Ancha	Blanco	Negras	Negro
Método propuesto	Ambos	Ancha	Crema con dibujo	Verdes	Ninguno

El sistema a desarrollar funcionará con placas anchas, con un ratio de aproximadamente 2:1; con un fondo principalmente crema, pero que, además, tiene el dibujo de una victoria (a la cual se refiere como ángel, debido a que representa el "Ángel de la independencia"). Las letras son verde obscuras, pero su superficie reflejante hace que aparezcan como verde claro bajo ciertas condiciones de iluminación. No tienen borde en absoluto, excepto por los automovilistas que utilizan portaplacas. Las condiciones bajo las que se toman las imágenes son naturales; es decir, con fondos variables y complejos donde el vehículo o la placa pueden ocupar una región pequeña o grande de la imagen o, incluso, abarcarla casi en su totalidad.

En la tabla 1 se resumen las características principales de las placas de algunos autores que abordan el tema de reconocimiento y localización de placas: la aplicación, forma, color de fondo, de letras y del borde, si existe uno. En el caso de forma, las placas largas son aquellas con una proporción de 4:1 de alto contra ancho, mientras que las anchas presentan una proporción de 2:1. Como se puede apreciar en la tabla comparativa la mayor parte de las placas contienen un fondo uniforme, en el caso que se aborda en este trabajo de tesis, se tiene un fondo con un dibujo que hace que el problema sea un poco mas complicado.

Un primer acercamiento que se intentó para resolver el problema de detección y localización de placas fue el de utilizar bordes verticales como lo hace Zheng [ZHE04], combinado con información de la tasa de altura contra ancho de la placa. La complejidad de los fondos en las imágenes no permitió intentar el resto del método de Zheng, que consta en contar la cantidad de bordes con una ventana, ya que existen infinidad de zonas en la imagen que cumplen con la característica de tener cierto numero de bordes. La razón por la que se probó un método basado en bordes (que representan la figura) primero fue que se buscaba algo lo más independiente que se pudiera de las condiciones de iluminación, por lo que los métodos que utilizan color no eran atractivos, en vista de que varios autores mencionan cuan susceptibles a los cambios de iluminación son éstos.

Al evaluar los resultados del método que utiliza bordes verticales se decidió intentar segmentación por color. Aún se buscaba que se pudieran encontrar placas bajo distintas fuentes de

iluminación, así que se buscaron regiones de decisión de color que contemplaran diferentes condiciones de iluminación. El método de Ryung [RYU94] no da buenos resultados cuando la placa está muy cerca de la cámara, porque los colores de la vecindad son muy uniformes y el método está basado en los cambios de color. Por otro lado, el método que propone Chang [CHA04] es inadecuado por la falta de un borde en la placa. Chang reduce su espacio de búsqueda a bordes que presentan cambios de ciertos colores; sin embargo, para las placas del Distrito Federal los cambios pueden ser en un gran número de colores, tantos como colores de coches existan.

El método propuesto por Zunino [ZUN00] no funcionaría para las placas del D.F. porque presupone que las placas son zonas de alto contraste, pero en un gran número de imágenes tomadas para el trabajo de tesis se observó que el contraste de la zona de la placa es bastante bajo por la falta de un borde y porque bajo ciertas iluminaciones los colores de fondo y letras son muy similares.

El método de Zimic [ZIM97] asume que la distancia a la placa es conocida, por lo que no se adapta a las necesidades del problema a resolver en esta tesis; además, funciona para placas blancas con letras negras y bordes brillantes, pero las placas del D.F. no son blancas y más aún, suelen aparecer obscuras en las fotos.

El método de Brugge [BRU98] funciona para un rango de tamaños de la placa en la imagen muy pequeño, por lo que se debe conocer el tamaño de la placa de antemano, así que no fue posible utilizarlo para la resolución del problema a resolver.

Para la lectura de los caracteres, el primer método que se probó produjo resultados comparables a los encontrados en artículos en el estado del arte, incluso en presencia de una gran cantidad de ruido, mayor a la que contemplan la mayoría de los autores, por lo que la experimentación con otros métodos no fue exhaustiva.

Los métodos basados en *templates* son sensibles al ruido, así que los métodos propuestos por Ryung [RYU94] y Chang [CHA04] no funcionan para el problema a resolver. Lo mismo ocurre con el método propuesto por Lim [LIM98], que utiliza *esqueletización*, un proceso conocido por ser muy sensible al ruido.

3. Marco Teórico.

3.1. Visión por Computadora.

3.1.1. Mejora de Contraste.

3.1.1.1. Ecualización.

La ecualización es el proceso por el cual se ajusta el histograma de una imagen de tal manera que llegue tanto al valor mínimo como al máximo posible y se distribuya uniformemente entre los dos. El fin de este ajuste es mejorar el contraste en imágenes que lo tienen pobre, tal que otras herramientas de procesamiento de imágenes puedan operar mejor con éstas.

De manera formal, el histograma de una imagen puede ser visto como una aproximación discreta de la función $p_r(r)$; que es la función de densidad de probabilidad (PDF por sus siglas en inglés) de la variable aleatoria r, que representa la distribución de cada valor de intensidad en la imagen. La ecualización del histograma puede entonces definirse como una función T(r) que transforma a la función $p_r(r)$ en la función $p_s(s)$, que es la PDF del histograma después del proceso.

En la ecuación 2 se muestra, para una PDF cualquiera, como se puede elegir T(r). Donde *r* es la intensidad original de los valores de grises y *s* es la intensidad deseada

$$p_s(s) = p_r(r)\frac{\partial r}{\partial s},\ r = T^{-1}(s). \tag{2}$$

En vista de que se desea que la distribución después de la ecualización sea uniforme, es decir, $p_s(s) = 1$; la ecuación 3 presenta una solución posible.

$$\frac{\partial r}{\partial s} = [p_r(r)]^{-1} \tag{3}$$

La ecuación 3 no tiene sentido cuanto $p_r(r) = 0$, por lo que propone la función que se presenta en la ecuación 4, que representa la función de probabilidad acumulada de distribución de la variable aleatoria *r*.

$$s = T(r) = \int_0^r p_r(r)\,dr \tag{4}$$

Al calcular la diferencial de 4 utilizando la regla de Liebnitz se obtiene la ecuación 5, que define la ecualización del histograma.

$$\frac{\partial s}{\partial r} = p_r(r) \tag{5}$$

El algoritmo para ecualizar el histograma de una imagen consiste en:

1. Normalizar los valores r del histograma de la imagen Esto es, dividir cada valor en el histograma entre el número total de píxeles en la imagen para que cumplan con la ecuación 6

$$0 \leqslant r \leqslant 1 \tag{6}$$

2. Normalizar los índices del histograma de la imagen Esto es, dividir cada índice del histograma entre el índice máximo.
3. Obtener la probabilidad acumulada de cada valor del histograma por medio del cálculo de la integral de la función $p_r(r)$. Ya que el histograma es discreto, esto se logra mediante la suma de cada valor del histograma normalizado con todos los valores del mismo anteriores, como se muestra en la ecuación 7; donde i_i es el valor del índice *i* del histograma.

$$i_i = \sum_{n=0}^{i} i_n \tag{7}$$

4. Por cada valor de la probabilidad acumulada, buscar el valor más parecido al mismo en el índice normalizado, y asignar el índice no normalizado que corresponda a éste en el histograma.
5. Convertir cada píxel en la imagen que tuviera el índice anterior a este nuevo índice.

3.1.1.2. Algoritmo de Mejora Local de Contraste.

Algoritmo propuesto por Zheng [ZHE04] para facilitar la ubicación de la placa en una imagen, mediante la mejora del contraste únicamente en partes de la imagen donde éste es bajo

En las ecuaciones, "$I_{x,y}$" representa la intensidad del píxel en la posición "x", "y" ($P_{x,y}$); "$I'_{x,y}$" es la intensidad del píxel "$P_{x,y}$" de la imagen mejorada. "$W_{x,y}$" es una ventana centrada en "$P_{x,y}$", "$I_{wx,y}$" es la intensidad promedio de la ventana "$W_{x,y}$", mientras que "$\sigma W_{x,y}$" es su desviación estándar. "I_0" y "σ_0" son la media y desviación estándar esperadas, respectivamente.

La ecuación 8 muestra, en teoría, cual debe ser la intensidad de los píxeles en la imagen mejorada

$$I'_{x,y} = \frac{\sigma_0}{\sigma_{W_{x,y}}} (I_{x,y} - \bar{I}_{W_{x,y}}) + I_0. \tag{8}$$

No es recomendable calcular todos los valores de "$\sigma_{wx,y}$" debido a que toma demasiado tiempo; por lo que Zheng propone dividir la imagen en 8x8 ventanas, calcular la desviación estándar de cada una de estas e interpolar el resto.

En las zonas únicamente obscuras o brillantes, el valor de "σ" es cercano a 0, cosa que provoca que el valor de "$I'_{x,y}$" sea muy grande. Estas zonas, al igual que las zonas en las que "$\sigma_{Wx,y}$" sea grande, no necesitan mejora. Por lo que se propone la siguiente fórmula mejorada (ecuación 9)

$$I'_{x,y} = f(\sigma_{W_{x,y}}) \cdot (I_{x,y} - \bar{I}_{W_{x,y}}) + \bar{I}_{x,y}. \tag{9}$$

En la ecuación 2, "$f(\sigma_{wx,y})$" es un coeficiente de mejora que ajusta qué tanto se modificará la intensidad del píxel destino, dependiendo de la desviación estándar de la ventana, como se muestra en la ecuación 10

$$f(\sigma_{W_{x,y}}) = \begin{cases} \dfrac{3}{\frac{2}{400}(\sigma_{W_{x,y}} - 20)^2 + 1} & if\ 0 \leqslant \sigma_{W_{x,y}} < 20 \\ \dfrac{3}{\frac{2}{400}(\sigma_{W_{x,y}} - 20)^2 + 1} & if\ 20 \leqslant \sigma_{W_{x,y}} < 60 \\ 1 & if\ \sigma_{W_{x,y}} \geqslant 60 \end{cases} \tag{10}$$

3.1.2. *Binarización*.

La *binarización* es el proceso a través del cual se reduce el histograma de una imagen para contener únicamente dos valores; típicamente 0 y 1. Por lo general, el valor de intensidad de cada píxel se compara con un umbral y se convierte en uno si es mayor a éste, o cero de lo contrario.

Se pueden agrupar las técnicas de *binarización* en dos grandes grupos: los que calculan un umbral global para toda la imagen y los que calculan umbrales adaptativos, por sección de la imagen o incluso pixel por pixel.

3.1.2.1. Métodos de Umbralización Global.

a) P-tile.

Para el algoritmo P-tile, el tamaño *p* del objeto a buscar en una imagen se asume conocido. Entonces se selecciona el umbral más alto que convierta al menos *p* píxeles en objetos.

b) Óptimo.

El algoritmo óptimo encuentra el umbral para *binarizar* la imagen a través de la búsqueda de dos centroides en el histograma de la imagen y de el nivel de gris con la probabilidad más baja entre ambos centroides.

Para encontrar los centroides, el algoritmo óptimo comienza por encontrar un umbral inicial T, mismo que puede obtener de la media de la imagen, seleccionando un píxel del borde, etc... Se busca encontrar dos distribuciones normales alrededor de este umbral, por lo que será manipulado a lo largo del algoritmo.

A continuación, se divide la imagen en dos regiones R1 y R2 a partir de el umbral T. Se busca la media u1 y u2 de las regiones R1 y R2 respectivamente. Después se vuelve a asignar el valor del umbral T a la media de u1 y u2. Finalmente, se repite el proceso con el nuevo umbral de manera repetida hasta que se estabilice; es decir, el valor de T no cambie en iteraciones sucesivas.

3.1.2.2. Métodos de Umbralización Adaptativos.

a) Método de Chow y Kaneko.

El método de Chow y Kaneko consiste en dividir la imagen en una serie de ventanas que no se sobreponen, calcular el histograma de cada ventana, probar cada histograma para averiguar si son bimodales; calcular un umbral para la ventana si se encuentra que sea bimodal, o interpolar el de las ventanas contiguas de lo contrario y, finalmente, interpolar bilinealmente el histograma de la ventana para cada uno de los píxeles contenidos en ésta.

Como parte de las pruebas de bimodalidad, el histograma de cada ventana se aproxima a una combinación de dos distribuciones de Gauss.

b) Método de Bernsen.

En el método de Bernsen se analiza una ventana de tamaño r * r alrededor de cada píxel de la imagen. Se encuentra el valor más bajo (Z0)y el más alto (Z1) en la ventana. Se calcula un umbral que es igual al promedio de Z0 y Z1. Antes de evaluar al pixel de acuerdo al umbral, se compara la diferencia ente Z0 y Z1, y en caso de que el valor absoluto de ésta es menor a un parámetro *l*, entonces se asume que todos los píxeles de la ventana están realmente en la misma zona de la imagen.

Los píxeles para los cuales se determina que toda la ventana está en la misma zona deben ser tratados de manera especial:

- Pueden ser clasificados con una tercera etiqueta, por ejemplo “2”; y evaluados al final del proceso, determinando de que tipo son predominantemente sus vecinos para, entonces, recibir esa etiqueta.

- Para ciertas imágenes es seguro asumir que son parte del fondo, especialmente imágenes de bordes, ya que los objetos en estas suelen ser pequeños; más pequeños que el tamaño de la ventana.

3.1.3. Filtros.

3.1.3.1 Filtros de Suavizamiento.

a) Filtro Media.

El filtro media (figura 1) es un *kernel* de convolución usado para suavizar (reducir la diferencia de intensidad entre un píxel y sus vecinos) imágenes, usualmente con la intención de reducir el ruido en las mismas. El valor del píxel resultante es el promedio de los píxeles alrededor del píxel origen.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

***Figura 1**: Kernel del filtro media*

b) Filtro Mediana.

Similar al filtro Media, el Filtro Mediana es un *kernel* usado para reducir el ruido en las imágenes. El píxel resultante tiene el valor de la mediana de sus vecinos.

La mediana se calcula ordenando la intensidad de todos los píxeles vecinos de acuerdo a su valor numérico y después reemplazando el valor al centro de la ventana con el valor que se encuentre en la mitad de la lista.

c) Filtro de Suavizamiento Gaussiano.

El filtro de Suavizamiento Gaussiano (figura 2) sirve para hacer borrosa una imagen. El píxel resultante es la combinación de los píxeles a su rededor, utilizando una aproximación discreta de la función Gaussiana como peso para multiplicarlos. Reduce también el ruido de una imagen.

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

***Figura 2**: Kernel del filtro Gaussiano (multiplicado por 115)*

3.1.3.2. Filtros de Realce.

a) Filtro *Unsharp* (Nítido).

El Filtro *Unsharp* se usa para dar nitidez a una imagen. Lo hace al substraer una versión suavizada de la imagen de la imagen original, lo que produce una imagen de bordes. Posteriormente, se suma una versión con los píxeles ajustados a escala de la imagen de bordes a la imagen original.

Uno de los filtros que se pueden utilizar para obtener la imagen de bordes es un filtro negativo Laplaciano discreto y se muestra en la figura 3.

0	-1	0
-1	-4	-1
0	-1	0

***Figura 3**: Filtro para obtener la imagen de bordes para el filtro unsharp*

b) Detector de Roberts de Bordes Cruzados.

El detector de Roberts de bordes cruzados (figura 4) se utiliza para resaltar bordes de una imagen. Lo logra mediante un cómputo sencillo y rápido de la medida de gradiente bidireccional en una imagen.

1	0
0	-1

***Figura 4**: Ejemplo de kernel del detector de Roberts*

c) Detector de Bordes de Sobel.

El detector de bordes de Sobel (operador de Sobel) se utiliza para encontrar los bordes en una imagen. Calcula el gradiente vertical u horizontal de una imagen (que debe ser alto en presencia de bordes); o bien calcula la magnitud o dirección del gradiente espacial bidimensional.

La versión vertical del *kernel* del operador de Sobel se muestra en la figura 5

-1	0	1
-2	0	2
-1	0	1

***Figura 5**: Kernel de convolución vertical del detector de Sobel*

d) Detector de Bordes de Prewitt.

El detector de bordes de Prewitt (figura 6) es una alternativa a los de Sobel y Roberts. Regresa la magnitud y dirección del operador gradiente de la imagen en una dirección dada (típicamente 0º o -45º)

+1	+1	+1
-1	-2	+1
-1	-1	+1

***Figura 6**: Kernel del detector de bordes de Prewitt sensible a 45º*

e) Diferencia de Gaussiano.

Este es un caso especial en el cual no se utiliza únicamente un *kernel* de convolución, sino la imagen original. La diferencia de Gaussiano sirve para detectar bordes en imágenes con ruido; esto se logra primero aplicando el *kernel* Gaussiano a una imagen, y después encontrando la diferencia de la imagen generada con la original.

3.1.4. Detección de Líneas.

a) Transformada de Hough.

La transformada de Hough es un algoritmo propuesto por Hough en 1962 que consiste en parametrizar la descripción de una característica de una imagen para alguna posición dada.

De manera general, la transformada de Hough se utiliza para encontrar formas sencillas en una

imagen. Esto se logra mediante la iteración a través de los parámetros que describen a la forma para cada pixel activado dentro de una imagen. Cuando el pixel cumple las características de los parámetros para la forma a buscar, éste produce un voto para esta configuración que se guarda en una matriz. Finalmente se elijen las configuraciones de la tabla con más votos y se asume que esas son las formas encontradas.

En el caso más simple, la transformada de Hough se utiliza para encontrar líneas rectas en una imagen. Las líneas rectas se pueden representar utilizando dos parámetros, así que la matriz de votos es bidimensional; una tabla.

Una manera de parametrizar las líneas rectas es mediante la ecuación general de la línea recta, que expresa que la coordenada "y" es igual a la coordenada "x" por la pendiente más la coordenada al origen. Se puede iterar por cada pendiente y coordenada al origen que cruce por cada pixel activado en la imagen. Ésta no es una representación práctica por que la coordenada al origen tiende a infinito cuando la pendiente es muy vertical para los píxeles más retirados del origen en "x", por lo tanto se tendría que hacer una tabla muy grande.

Una representación más conveniente es la ecuación de la línea recta para el sistema de coordenadas polares; donde "x" por el coseno de teta más "y" por el seno de teta es igual a phi. Eligiendo una granularidad para incrementos de θ y φ se puede recorrer la imagen completa sin utilizar una tabla demasiado grande.

3.1.5. Reducción de Colores.

a) Nearest Color.

El método de reducción de color *nearest color* se utiliza para reducir el número de colores de una imagen a cualquier cantidad arbitraria. Lo hace mediante el cálculo de los colores más comunes que aparecen en una imagen, y el mapeo de el resto a ellos, asignando el color al que se parezca más.

El primer paso es calcular el histograma tridimensional "H" de la imagen (dimensiones "R", "G", y "B"). El cálculo se puede hacer de acuerdo a la fórmula 11, donde "I" representa la imagen.

$$\forall\ p \in I\ do\ H[p.r][p.g][p.b]++. \tag{11}$$

A continuación, para reducir el número de colores en una imagen a “N”, se seleccionan los “N” elementos de “H” con el mayor valor y se guardan en la paleta “P”

Finalmente, se calcula la distancia de cada uno de los colores de la imagen a todos los elementos de “P” utilizando la fórmula 12 (distancia euclidiana), y se convierten todas las instancias del color en la imagen por el color en “P” cuya distancia sea menor con respecto al color evaluado.

$$d = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2} \tag{12}$$

3.2. Reconocimiento de Patrones.

3.2.1. Redes Neuronales.

Las redes neuronales han sido una de las herramientas de la inteligencia artificial más importantes en los últimos 25 años, tanto así que han sido llamadas "más importantes que la bomba atómica" [SON89].

Las redes neuronales son un modelo computacional que tiene como principio la emulación de las estructuras biológicas del cerebro de los seres vivos, de una manera computacionalmente realizable.

a) Neuronas de McCulloch-Pitts.

Las redes neuronales de McCulloch-Pitts fueron el primer intento de modelar el funcionamiento del cerebro de manera matemática. Los autores las concibieron en 1943.

El foco de la investigación y modelo de McCulloch-Pitts era el funcionamiento del cerebro, sin embargo, las investigaciones más recientes se han apartado hasta cierto punto de este enfoque [AND95]. Aún así, los autores sabían que su modelo de neurona era una simplificación y tan sólo se preguntaban cómo este tipo de neuronas interactuaban en las intrincadas redes que formaban con ellas.

Las neuronas de McCulloch-Pitts tienen dos entradas y una salida. Las entradas pueden tomar los valores de uno o cero; y la salida puede igualmente ser uno o cero. Las entradas pueden activar o inhibir a la neurona. Para el caso de las neuronas que activan, su valor se suma para obtener un solo valor de entrada. Para el caso de las que inhiben, su valor se resta para obtener dicho valor. La neurona tiene un umbral; que, en caso de ser superado por el valor de entrada, activa la neurona (es decir, la salida se vuelve uno); de lo contrario la salida es cero. La figura 7 presenta un ejemplo de una neurona de McCulloch-Pitts

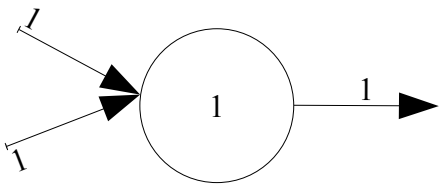

***Figura 7**: Neurona de McCulloch-Pitts*

b) Perceptrón.

El Perceptrón es un tipo de red neuronal computacional ideada por Frank Rossenblatt en el Laboratorio Aeronáutico Cornell en 1957.

El Perceptrón consiste en una capa de neuronas conectadas por ligas de entrada y salida con ciertos pesos asociados.

Cada neurona recibe una serie de estímulos; esto es, la suma de los valores de las ligas de entrada multiplicados por el peso de cada liga. Una vez estimulada, la neurona se "activa" si el valor de entrada se encuentra por encima de cierto umbral, en cuyo caso generará una salida de 1. Si el valor se encuentra por debajo del umbral, la neurona no se activa y ésto la hace generar una salida de -1 (función umbral o escalón).

De manera más formal, se puede decir que si la salida de una neurona es mayor a 0, un objeto pertenece a la clase "w_1" y que en caso de ser menor o igual a 0, éste pertenece a la clase "w_2". La salida de la neurona se define como la suma pesada de sus entradas, como se muestra en la ecuación 13, donde "R" representa la salida, "w_i" el peso de la liga de entrada "i", "x_i" la entrada "i".

$$R = \sum_{i=1}^{n+1} w_i x_i \tag{13}$$

La memoria del Perceptrón se encuentra en el valor de los pesos de las neuronas de las capas de entrada.

El aprendizaje se logra al proveer a la red de una muestra de objetos de entrenamiento de ambas clases y ajustar los pesos de acuerdo a la llamada "Regla Delta", donde el peso nuevo es igual a la suma del peso viejo más o menos un factor dependiendo de si la salida asociada con ese peso fue incorrecta y corresponde a la primera o segunda clase respectivamente; o se deja sin cambio en caso de la salida sea correcta sin importar la clase, como se muestra en la ecuación 14, donde "$w(\kappa+1)$" es el peso nuevo, "$w(\kappa)$" es el peso viejo; "$x(\kappa)$" es la salida del nodo y "c" es una constante positiva llamada "incremento de corrección" .

$$w(\kappa+1)=\begin{cases} w(\kappa)+cx(\kappa) & si\ x(\kappa)\in w_1\ \&\&\ w_i(\kappa)x(\kappa)\leqslant 0 \\ w(\kappa) & si\ x(\kappa)\in w_1\ \&\&\ w_i(\kappa)x(\kappa)>0 \\ w(\kappa)-cx(\kappa) & si\ x(\kappa)\in w_2\ \&\&\ w_i(\kappa)x(\kappa)\leqslant 0 \\ w(\kappa) & si\ x(\kappa)\in w_2\ \&\&\ w_i(\kappa)x(\kappa)>0 \end{cases} \quad (14)$$

Los pesos deben de ajustarse para todas los elementos del conjunto de entrenamiento hasta que la red converja; es decir, que todos los objetos de clase "w_1" sean clasificados como tal, y que todos los de clase "w_2" sean clasificados de esa manera. Esto constituye aprendizaje supervisado.

La regla delta tiene la desventaja de que, en caso de que las clases "w_1" y "w_2" no sean linealmente separables y que, por lo tanto, no converja la red; el algoritmo continuará iterando por tanto tiempo como se le permita correr. Es imposible saber *apriori* en cuántas iteraciones convergerá la red, y el algoritmo no presenta provisiones para saber si las clases son linealmente separables, por lo que es imposible determinar de manera determinística si el algoritmo aún no converge o si no lo hará nunca. Cuando el algoritmo ha corrido por mucho tiempo, se puede asumir que no convergerá.

La figura 8 contiene un ejemplo de Perceptrón

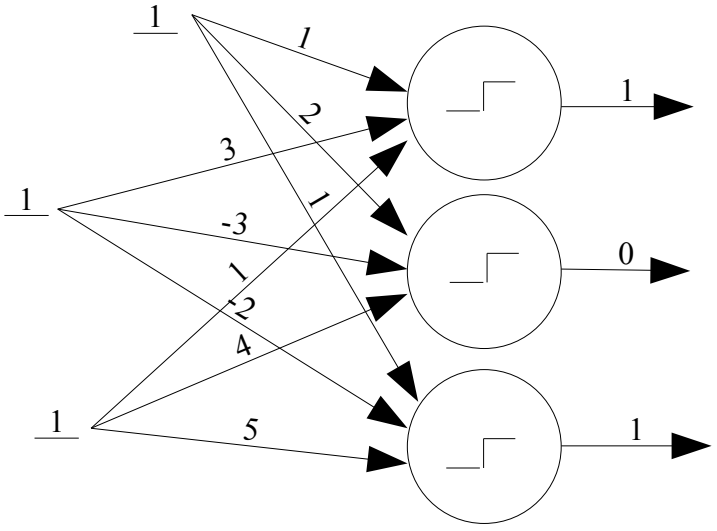

***Figura 8**: Perceptrón*

c) Adaline.

Las redes neuronales Adaline (Neurona Linear Adaptativa, por sus siglas en Inglés) fueron propuestas por Widrow y Hoff en 1985.

Adaline es una evolución del Perceptrón en la que se substituye la función umbral por una función continua de transferencia linear llamada sigmoide, que se muestra en la ecuación 15.

$$f(x)=\frac{i}{1+e^{x}} \tag{15}$$

Además se introdujo la regla de aprendizaje Widrow-Hoff, conocida como Media Mínima Cuadrada, (LMS por sus siglas en inglés) para ajustar los pesos de las ligas de acuerdo a la magnitud del error.

La regla de aprendizaje LMS se basa en otra función de aprendizaje que utiliza la gradiente de un vector para corregir las clases. El principio de la función de gradiente es la fórmula que se muestra en la ecuación 16, y utiliza un vector como el que se muestra en la ecuación 17. "X_w" corresponde a la gradiente del vector "x".

$$X_w > 0 \tag{16}$$

$$x=\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{pmatrix} \tag{17}$$

LMS utiliza una fórmula derivada de la ecuación 16, pero en vez de buscar un vector w, se buscan vectores "w" y "b" tal que se cumpla la expresión que se encuentra en la ecuación 18, donde los componentes de "b = $(b_1, b_2, b_3, ..., b_n)$' " son todos positivos; lo que hace que la ecuación 16 y 18 sean equivalentes.

$$X_w = b \tag{18}$$

La función de criterio en la ecuación 19 alcanza su valor mínimo cuando se cumple la ecuación 18, por lo que el objetivo es minimizar la función "J(w,x,b)" para "w" y "b".

$$J(w,x,b)=\frac{1}{2}\sum_{j=1}^{N}(w'x_j-b_j)=\frac{1}{2}\|x_w-b\|^2 \tag{19}$$

Los gradientes asociados con este problema son los presentados en las ecuaciones 20 y 21

$$\frac{\partial J}{\partial w}=X'(X_w-b) \tag{20}$$

$$\frac{\partial J}{\partial b}=-(X_w-b) \tag{21}$$

Debido a que "x" no tiene restricciones, la diferencial puede igualarse a cero y así obtener la ecuación 22, donde "$X^{\#}$" es es la inversa generalizada de "X".

$$w=(X'X)^{-1}X'b=X^{\#}b \tag{22}$$

Lo mismo no aplica para "b", en vista de que todos sus elementos deben ser positivos. Considerando esto, se llega a la ecuación 23, complementada por la ecuación 24; donde "κ" representa el índice de iteración, "*i*" el índice del vector y "c" es, de nuevo, un factor de corrección.

$$b(\kappa+1)=b(\kappa)+\partial b(\kappa) \tag{23}$$

$$\partial b_i(\kappa)=\begin{cases}2c[X_w(\kappa)-b(\kappa)]_i & si\,[X_w(\kappa)-b(\kappa)]_i>0\\ 0 & si\,[X_w(\kappa)-b(\kappa)]_i\leqslant 0\end{cases} \tag{24}$$

De las ecuaciones 22 y 23 se obtiene la 25

$$\begin{aligned} w(\kappa+1) &= X^{\#}b(\kappa+1)\\ &= X^{\#}[b(\kappa)+\partial b(\kappa)]\\ &= X^{\#}b(\kappa)+X^{\#}\partial b(\kappa)\\ &= w(\kappa)+X^{\#}\partial b(\kappa)\end{aligned} \tag{25}$$

Introduciendo un nuevo vector, definido como se muestra en la ecuación 26 se puede finalmente resolver el sistema para "w(κ+1)" y "b(κ+1)"; como se muestra en la ecuación 27

$$e(\kappa)=X_w(\kappa)-b(\kappa) \tag{26}$$

$$\begin{aligned} w(1) &= X^{\#}b(1) \qquad\qquad donde\ b(1)>0\\ e(\kappa) &= X_w(\kappa)-b(\kappa)\\ w(\kappa+1) &= w(\kappa)+cX^{\#}[e(\kappa)+|e(\kappa)|]\\ b(\kappa+1) &= b(\kappa)+c[e(\kappa)+|e(\kappa)|]\end{aligned} \tag{27}$$

Siempre que "$X_w>0$" tenga solución, este algoritmo converge utilizando un valor de "c" tal que "$0<c\leq 1$".

En el caso de que todos los elementos de "e(κ)" se vuelvan negativos en cualquier iteración, se puede comprobar que el algoritmo nunca convergerá. Por otro lado, se sabe que en caso de que "e(κ)≥0", "w(κ)" es una solución porque eso comprueba que "$X_w(\kappa)\geq b(\kappa)$", y se sabe que "b(κ)" es un vector positivo.

Al igual que el Perceptrón, Adaline es una red neuronal de una capa.

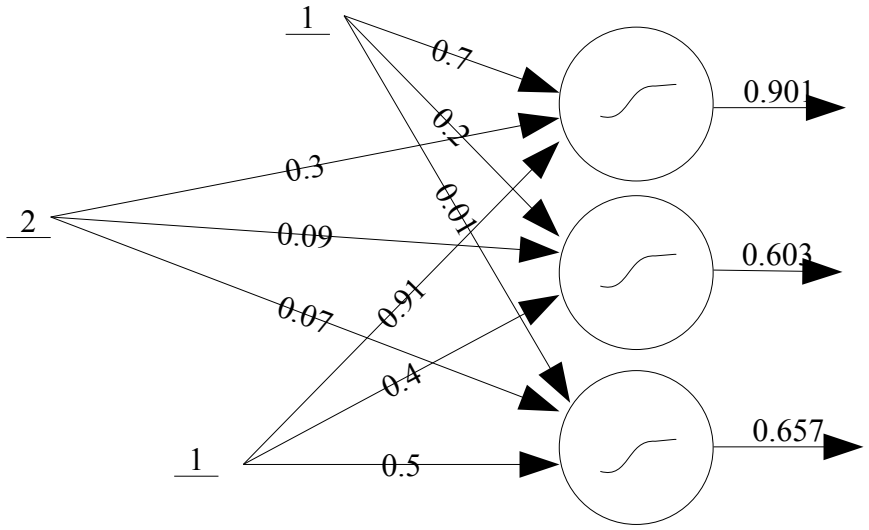

***Figura 9**: Red Adaline*

d) Redes Multicapa: Back Propagation.

Las redes multicapa son una extensión aplicable tanto al Perceptrón como a Adaline.

La principal razón para tener varias capas es que las redes de una capa sólo pueden resolver problemas linealmente separables, lo que las limita seriamente; por ejemplo una red de una capa no puede aprender la función XOR.

Al momento de reconocer patrones, las redes multicapa actúan de manera idéntica a la que lo harían una serie de redes neuronales de una capa colocadas una en frente de la otra. Las salidas de las redes de la capa n se convierten en las entradas de la capa n+1. Las entradas de la red son las entradas de la capa 1; las salidas de la red son las salidas de la última capa.

Una red de esta naturaleza no se puede entrenar de la misma manera que se entrena una red de una sola capa; tanto la regla delta para entrenar el Perceptrón, como LMS funcionan sólo en la última capa antes de la capa de salida debido a que necesitan saber el error correspondiente a cada salida de un nodo; información que no está disponible directamente en ninguna capa más que la última; así que es necesario encontrar una manera de ajustar los pesos de las neuronas ocultas. Al algoritmo que logra ésto se le llama Back Propagation, y consiste en calcular qué tanto del error que corresponde a una neurona en la capa de salida se debe a la liga que lo conecta con cada neurona de la capa anterior; o, en general, qué tanto del error que corresponde a cada neurona de la capa n, se debe a la liga que lo conecta con cada neurona de la capa n-1.

La función de error que se intenta minimizar con Back Propagation se muestra en la ecuación 28. E es el error, "t_k" es la salida esperada en el nodo "k", y "o_k" es la salida obtenida en el nodo "k".

$$E=\frac{1}{2}\sum_k (t_k-o_k)^2$$
$$\frac{\partial E}{\partial W_{jk}}=\frac{\partial}{\partial w_{jk}}\left(\frac{1}{2}\sum_k (t_k-o_k)^2\right) \tag{28}$$

En la ecuación 29 se muestra el proceso matemático para llegar a la fórmula para ajustar los pesos en la última capa, que se muestra en la ecuación 30. En las ecuaciones, "η" representa la ganancia, "w_{kj}" es el peso de la liga de entrada "k" al nodo "j" y "o_j" representa la salida del nodo "j".

$$\Delta w_{kj}=-\eta\frac{\partial E}{\partial w_{kj}}$$
$$o_k=f(net_k)$$
$$net_k=\sum w_{kj}o_j$$
$$Por\ \ regla\ \ de\ \ la\ \ cadena$$
$$\frac{\partial E}{\partial w_{kj}}=\frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}}$$
$$\frac{\partial net_k}{\partial w_{kj}}=\frac{\partial}{\partial w_{kj}}\sum w_{kj}o_j=o_j$$
$$\delta_k=-\frac{\partial E}{\partial net_k}$$
$$\Delta w_{kj}=\eta\delta_k o_j \tag{29}$$
$$Evaluando\ \ \delta_k=-\frac{\partial E}{\partial net_k}$$
$$=-\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k}$$
$$\frac{\partial E}{\partial o_k}=-(t_k-o_k)$$
$$\frac{\partial o_k}{\partial net_k}=f'_k(net_k)$$
$$\delta_k=(t_k-o_k)f'_k(net_k)$$

$$\begin{aligned} \Delta w_{kj} &= \eta\left(t_k - o_k\right) f'_k(net_k)\, o_j \\ &= \eta \delta_k o_j \end{aligned} \tag{30}$$

En la ecuación 31 se muestra el proceso matemático necesario para calcular el incremento de los pesos de las capas ocultas. En la ecuación, "f(net)" representa la función de activación que utilizan los nodos de la red.

$$\begin{aligned} \Delta w_{kj} &= -\eta \frac{\partial E}{\partial w_j} \\ &= -\eta \frac{\partial E}{\partial net_j} \frac{\partial net_j}{\partial w_j} \\ &= -\eta \frac{\partial E}{\partial net_j} o_i \\ &= \eta \left(-\frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \right) o_i \\ &= \eta \left(-\frac{\partial E}{\partial o_j} \right) f'_j(net_j) o_i \qquad ¿ \\ &= \eta \delta_j o_i \end{aligned} \tag{31}$$

$$\begin{aligned} \frac{-\partial E}{\partial o_j} &= -\frac{\sum_k \partial E}{\partial net_k} \frac{\partial net_k}{\partial o_j} \\ &= \sum_k \left(\frac{\partial E}{\partial net_k} \right) \frac{\partial}{\partial o_j} \sum_m w_{km} o_m \\ &= \sum_k \left(\frac{\partial E}{\partial net_k} \right) w_{kj} = \sum_k \delta_k w_{kj} \\ \delta_j &= f'_j\left(net_j\right) \sum_k \delta_k w_{kj} \end{aligned}$$

A partir de la ecuación de la función sigmoide (15), y de la ecuación para encontrar "δ_j" que se encuentra en (31), se obtiene la fórmula para calcular el incremento de pesos en una red neuronal que utilice la función sigmoide como función de activación. La fórmula resultante se muestra en la ecuación 32

$$
\begin{aligned}
o_j &= \frac{1}{1+e^{-I}} \\
\frac{\partial o_j}{\partial I} &= \frac{e^{-I}}{\left(1+e^{-I}\right)^2} \\
&= \frac{\left(\left(1+e^{-I}\right)-1\right)}{\left(1+e^{-I}\right)^2} \\
&= \frac{1}{\left(1+e^{-I}\right)}\left(1-\frac{1}{\left(1+e^{-I}\right)}\right) \\
&= o_j(1-o_j)
\end{aligned}
\tag{32}
$$

Substituyendo la ecuación (32) en la (31) se obtiene la fórmula para calcular el incremento en los pesos para las capas ocultas, que se muestra en la ecuación (33)

$$o_j = o_j(1-o_j) \tag{33}$$

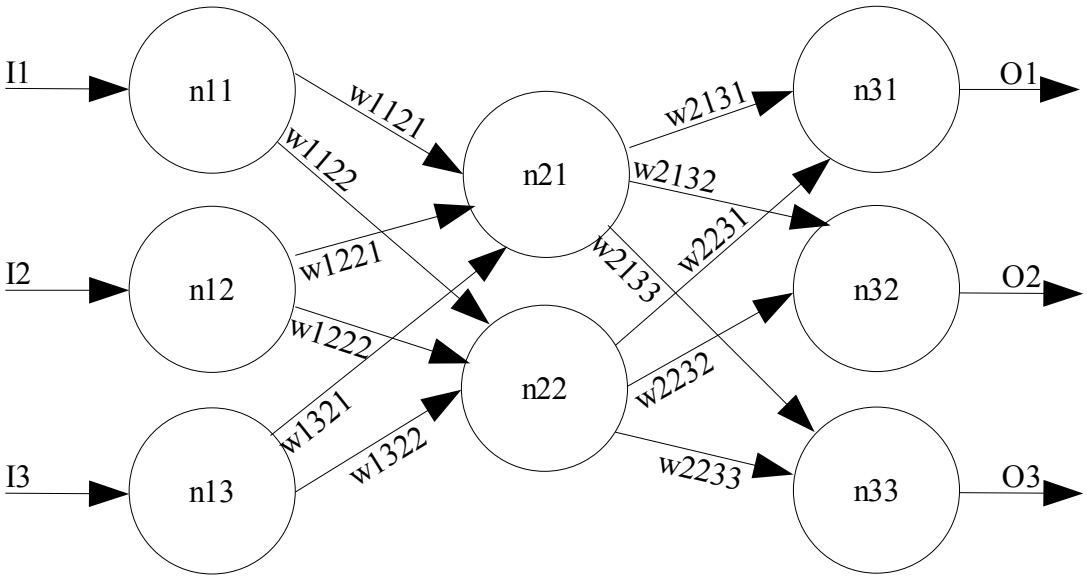

***Figura 10**: Red Back Propagation con una capa oculta*

e) Redes de Hopfield.

Las redes de Hopfield representan un tipo de memoria asociativa en la que se puede buscar de acuerdo a su contenido.

Estas redes consisten en una serie de neuronas o unidades de procesamiento conectadas de forma simétrica con sus vecinos a través de ligas pesadas.

Cada neurona puede estar en dos estados: Activa o inactiva, y su estado depende del peso de las ligas que la conecten a otras neuronas que también estén en estado activo. Un peso positivo hace que la neurona tienda a activarse, y un peso negativo la inclina a lo contrario.

De esta manera se realiza la suma de los pesos de las ligas que la conecten a sus vecinos activos (se puede considerar el que un vecino esté activo como un factor 1, e inactivo 0, y multiplicar este factor por el peso de la liga); si finalmente la suma es positiva la neurona queda en estado activo; de lo contrario, se inactiva.

Ésto se hace repetidamente en neuronas elegidas aleatoriamente hasta que el sistema converge; es decir, no ocurren más cambios en estados de las neuronas.

Cada configuración tiene un número limitado de estados hacia los cuales convergerá; esta propiedad se utiliza para que, alimentando a una red un patrón parcial, ésta converja hacia el completo. La figura 11 presenta un ejemplo de este tipo de red.

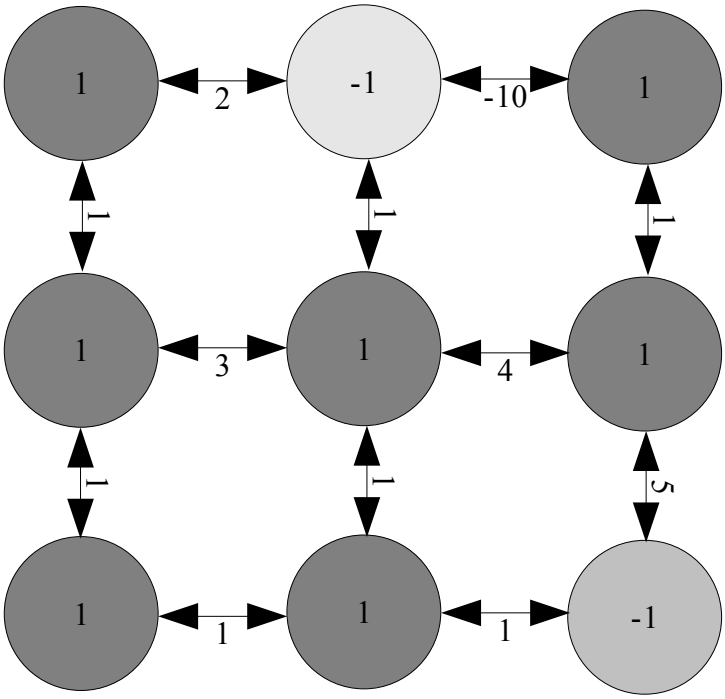

***Figura 11**: Configuración de una red de Hopfield*

3.2.2. Algoritmos Genéticos.

Los algoritmos genéticos son programas computacionales que emulan el proceso de evolución natural para buscar una solución nueva a un problema u optimizar una preexistente.

El algoritmo genético tradicional fue introducido originalmente por John Holland en 1975. Más recientemente se han propuesto modificaciones a los algoritmos genéticos para ampliar su aplicación o mejorar su desempeño, que difieren del modelo estricto que Holland propuso.

En términos generales, un algoritmo genético es un programa computacional cuyos datos están almacenados en estructuras de datos que son análogas a los cromosomas de la evolución natural.

Un algoritmo genético comienza con una población población inicial que, aunque de gran tamaño, es ínfima comparada con las posibles soluciones. Esta población usualmente es generada aleatoriamente. El algoritmo entonces selecciona individuos al azar de dicha población y crea una nueva población (población intermedia) más pequeña con éstos.

La aptitud de cada uno de los individuos de la nueva población es medida con el uso de una "función de aptitud "; ésta regresa un número representativo de qué tan bueno es el individuo al que evaluó.

Dependiendo de su aptitud, el algoritmo prosigue a asignar posibilidades de reproducción. Al igual que en la evolución natural, los individuos más aptos tienen mayores oportunidades de reproducirse que los individuos menos buenos.

Utilizando la antes mencionada posibilidad, el algoritmo crea parejas con el fin de cruzarlas para generar descendencia. Por lo general cada pareja genera uno o dos hijos, aunque esto puede variar.

Finalmente parte de la población "muere"; es decir, es desechada para conservar el mismo número de individuos que se tenía antes de la reproducción.

La evaluación de la aptitud de los individuos, asignación de probabilidades, creación de parejas, reproducción y muerte se repite hasta que se encuentra con una condición de paro; después de la cual se elije una nueva población intermedia a partir de otros individuos tomados de la población inicial, repitiéndose el resto de los pasos nuevamente hasta que se encuentre con la condición de paro otra vez; y de nuevo se repite el proceso continuamente hasta que se cumpla una segunda condición de paro, que indica la salida del programa.

Dentro de cualquiera de los pasos anteriores, es posible introducir un concepto llamado mutación. Cuando un individuo muta, se cambia de manera aleatoria uno o más de sus genes, introduciendo variedad en la población con soluciones que quizás no hayan sido vistas antes, que pueden ser mejores o peores.

Las mutaciones pueden ocurrir de forma probabilística durante la reproducción, después de cada cierto número de poblaciones intermedias, después de la reproducción, etc...

a)Codificación del Problema.

Tradicionalmente, se debe utilizar una estructura de datos para representar el problema que sea posible de ser convertida a una cadena binaria, manipulada de esta forma y después convertida de regreso a la representación que requiera el problema.

Por la manera en la que recorren los algoritmos genéticos en el espacio de búsqueda, es deseable que un cambio pequeño en la cadena binaria se refleje como un cambio pequeño en el individuo al que ésta representa, mientras que un cambio fuerte en una se transforme en uno radical en la otra. Debido a esto, no siempre la conversión de un número a su equivalente binario es la mejor representación que se puede usar para un algoritmo genético.

Preferentemente, todas (o al menos el mayor número posible de) las combinaciones de unos y ceros en la cadena deben generar soluciones válidas al problema. En caso de no ser todas, debe haber provisiones para evitar a toda costa que se generen individuos inválidos o, al menos, que se reproduzcan.

No siempre es posible cumplir con tal restricción, por lo que se han propuesto nuevas definiciones de algoritmos genéticos que son menos restrictivas. Bajo este pensamiento, en general, un algoritmo genético es todo algoritmo que utiliza operadores de selección y recombinación para generar nuevas muestras en un espacio de búsqueda [WHI94].

De esta manera, deja de ser estrictamente necesario codificar un problema de manera binaria. Al tipo de algoritmos genéticos que sí lo hacen se les llama entonces, canónicos.

b) Operador de Cruce.

En el caso más sencillo, para los algoritmos genéticos canónicos, el cruce consiste en combinar las cadenas genéticas (binarias) de dos individuos a partir de un punto llamado “punto de cruce”,

elegido al azar, tal que se generen dos hijos: Uno con la cadena genética del padre desde el inicio hasta el punto de cruce, y la de la madre del punto de cruce en adelante; y otro con la cadena de la madre del principio al punto de cruce y la del padre en adelante. A este operador de cruce se le conoce como "cruce de un solo punto"

Una extensión ampliamente usada al cruce antes descrito, es el cruce multipunto; donde se tienen varios puntos de cruce y se va alternando la cadena genética del padre y de la madre por cada uno de estos puntos. La efectividad de este tipo de cruce depende del tipo de problema, pero en general se llega más rápido a una solución con este cruce que con el de un solo punto; aunque la selección del número de puntos es crucial. En general no se recomienda un número elevado de puntos, especialmente para códigos binarios cortos.

Al momento de cruzar dos individuos es importante asegurarse de que los hijos sean individuos válidos; es decir, que se encuentren ubicados entre los rangos de la solución del problema y que todas sus partes sean válidas. En caso contrario se sugiere descartarlos inmediatamente o al menos hacer lo posible para que sus posibilidades de reproducción sean pequeñas.

Para los algoritmos genéticos no canónicos o en el caso de que la probabilidad de que se generen hijos inválidos al cruzar convencionalmente a dos individuos sea alta, no es posible el cruce por medio del intercambio de genes; por lo tanto se deben crear soluciones que se ajusten al tipo de problema específico en el que se está trabajando.

Una estrategia que se puede usar es la combinación lineal de los padres para la generación de los hijos, de manera similar a como se hace en la Programación Evolutiva [SCH97].

c) Operador de Mutación.

La mutación es, en general, una procedimiento opcional para los algoritmos genéticos, pero su uso es recomendado porque en una variedad de problemas ayuda a encontrar una solución mejor y más rápidamente.

El principio básico detrás de la mutación es la introducción de variedad a un acervo genético que probablemente se encuentre estancado después de cruzarse entre sí mismo un número de veces. Al introducirse mutaciones es probable que se visiten partes del espacio de búsqueda que no hayan sido explorados con anterioridad porque estuvieran fuera de la zona de búsqueda acotada por los individuos iniciales de la población intermedia, o porque se encuentren rodeados de mínimos locales que enfoquen

la exploración en otras direcciones.

Para los algoritmos genéticos canónicos, la mutación consiste en invertir, de manera aleatoria, un punto, varios puntos o un rango de puntos de la cadena genética de uno o más individuos de una generación.

Al igual que con el cruce, no debe descuidarse la posibilidad de la generación de individuos inválidos al alterar el código genético directamente.

c.i) Intensidad de la Mutación.

La intensidad de la mutación se refiere a que tanto se cambia un individuo al momento de mutar. Para los algoritmos genéticos canónicos, La intensidad de la mutación se controla por el número de genes que se alteran en la cadena binaria genética. Una mutación de un gen es débil y mientras más genes se modifiquen la intensidad de la mutación se incrementa.

d) Función de Aptitud.

La función de aptitud es la función que evalúa que tan bueno es un individuo para solucionar el problema que se está optimizando. Ésta, debe recibir un individuo y regresar un índice que indica su aptitud relativa a los demás.

Es importante que la función de aptitud regrese un resultado rápido, incluso si éste es una aproximación del verdadero *fitness* (aptitud) del individuo. La razón es que es necesario evaluar un número muy elevado de individuos por cada generación, además de que usualmente el algoritmo genético tiene un número también muy grande de generaciones; por lo que una función de aptitud que sea lenta puede resultar en que el problema no sea resuelto en un tiempo razonable. Por otro lado, ya que el único fin de el valor del *fitness* es comparativo, no suele resultar muy grave utilizar sólo una aproximación.

e) Criterio de Selección.

Al seleccionar individuos para reproducirse y sobrevivir se debe dar mayor posibilidades a los que sean más aptos. Varias técnicas han sido sugeridas en la literatura para lograr esto.

e.i) Selección por Ruleta.

La selección por ruleta consta de una rueda dividida en rebanadas que representan a los individuos de la población, donde el tamaño de cada rebanada es proporcional a la aptitud del individuo. Se hace girar la ruleta un número de veces y cada vez se elige al individuo que resulte seleccionado.

e.ii) *Stochastic Universal Selection.*

Stochastic Universal Selection es una variante de la selección por ruleta, en la que, después de formar la rueda con rebanadas proporcionales al *fitness* de los individuos, se colocan N marcadores equidistantes sobre ésta; donde N es el tamaño de la población. Si un individuo es seleccionado por más de un marcador, la población debe contener una copia de éste por cada marcador que lo seleccione.

e.iii) Selección por Torneo Binario.

Se eligen pares de individuos al azar y se selecciona al mejor de los dos.

En la variante de selección por torneo binario con reemplazo, los dos individuos regresan a la población y pueden volver a ser seleccionados individualmente. Se repite el proceso hasta que la nueva generación tenga el tamaño N (tamaño de población intermedia).

En la variante sin reemplazo, después de conservar al mejor; los dos individuos se guardan por separado sin regresar a la población de la que vinieron. Ya que siempre se toman dos y se elije a uno, al terminar con la población se tiene una población de tamaño N/2; por lo tanto, es necesario hacer otra ronda de torneos con todos los individuos de la población intermedia. Un efecto secundario de éste tipo de torneo es que el mejor individuo siempre se selecciona dos veces y el peor ninguna.

4. Metodología.

4.1. Métodos de Detección y Localización de Placas.

4.1.1. Método Basado en Bordes Verticales.

En el método basado en bordes verticales se utilizan técnicas como la ecualización de la imagen, un *kernel* de convolución de desarrollo propio, una versión modificada del algoritmo óptimo de *umbralización*, un algoritmo basado en ventanas de reducción de ruido, otro basado en segmentación y análisis de características de los bordes. La figura 12 contiene un diagrama que muestra las técnicas que se utilizan en este método y su secuencia.

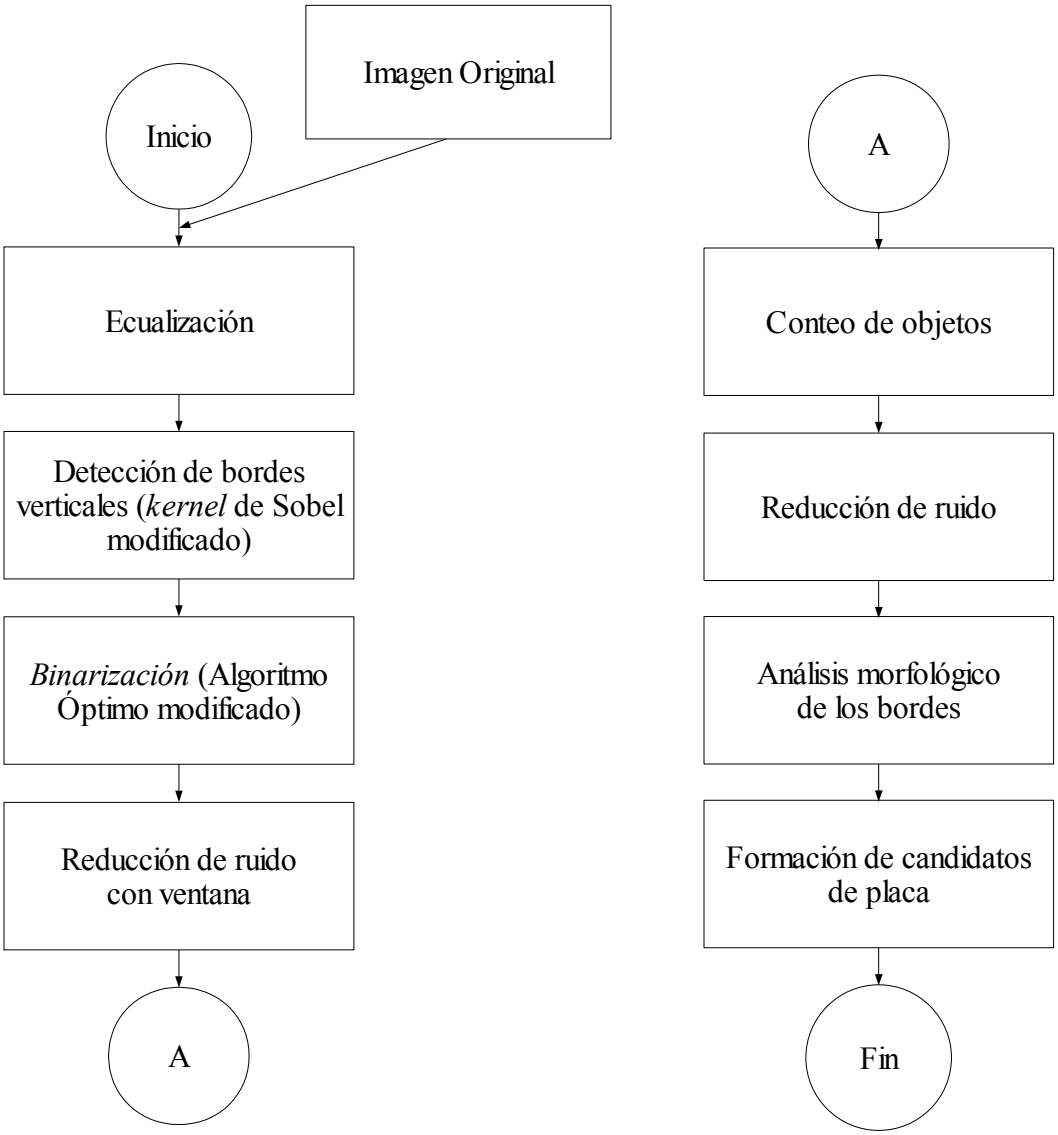

***Figura 12**: Método de detección y localización de placa basado en bordes verticales*

4.1.2. Métodos Basados en Color.

4.1.2.1. Método Basado en Redes Neuronales.

El método de localización basado en redes neuronales utiliza una red neuronal de Back Propagation para convertir los colores de cada píxel de la imagen en clases y después hace un conteo de objetos y determina los colores que son candidatos de placa a través de su clase. El proceso se muestra a manera de diagrama en la figura 13

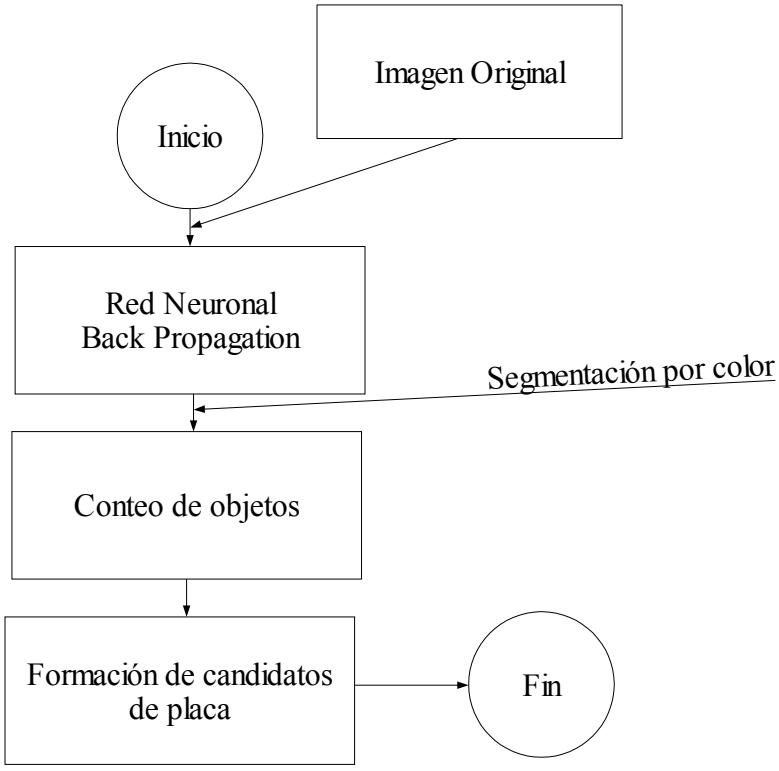

***Figura 13**: Método de detección y localización de placa basado en color y redes neuronales*

4.1.2.2. Método Basado en Algoritmos Genéticos.

El método basado en algoritmos genéticos utiliza un algoritmo genético para generar una región de decisión de color representada por objetos tridimensionales en el espacio RGB. La figura es después utilizada para segmentar la imagen por color. El método se muestra en la figura 14 , se muestran dos diagramas, uno representando la generación de la región de decisión de color y otro la segmentación.

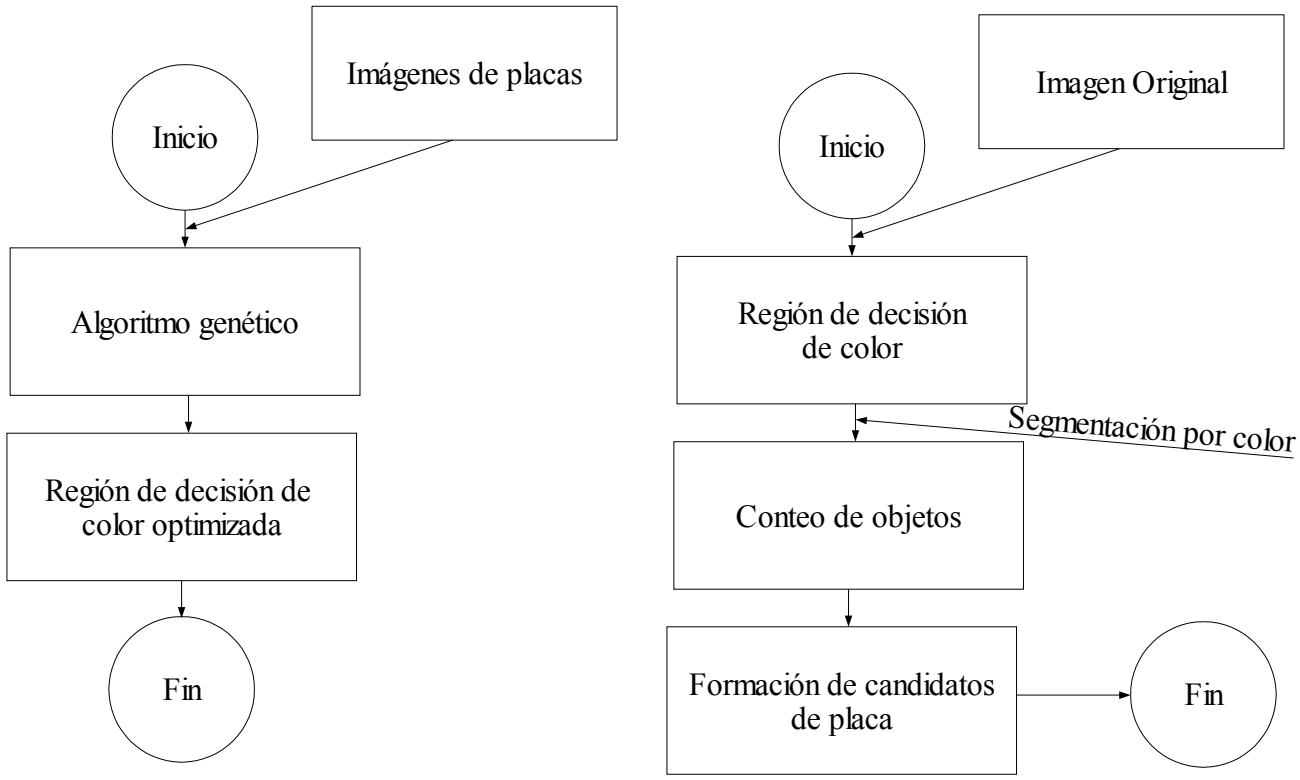

***Figura 14**: Método de detección y localización basado en color y algoritmos genéticos*

4.2. Reconocimiento de Placas.

Para el reconocimiento de placas primero se segmenta la imagen de la placa por color utilizando una tabla, se reconstruyen los caracteres utilizando una función heurística, se normaliza su tamaño y se extraen características para que una red neuronal los identifique. El proceso se muestra en la figura 15

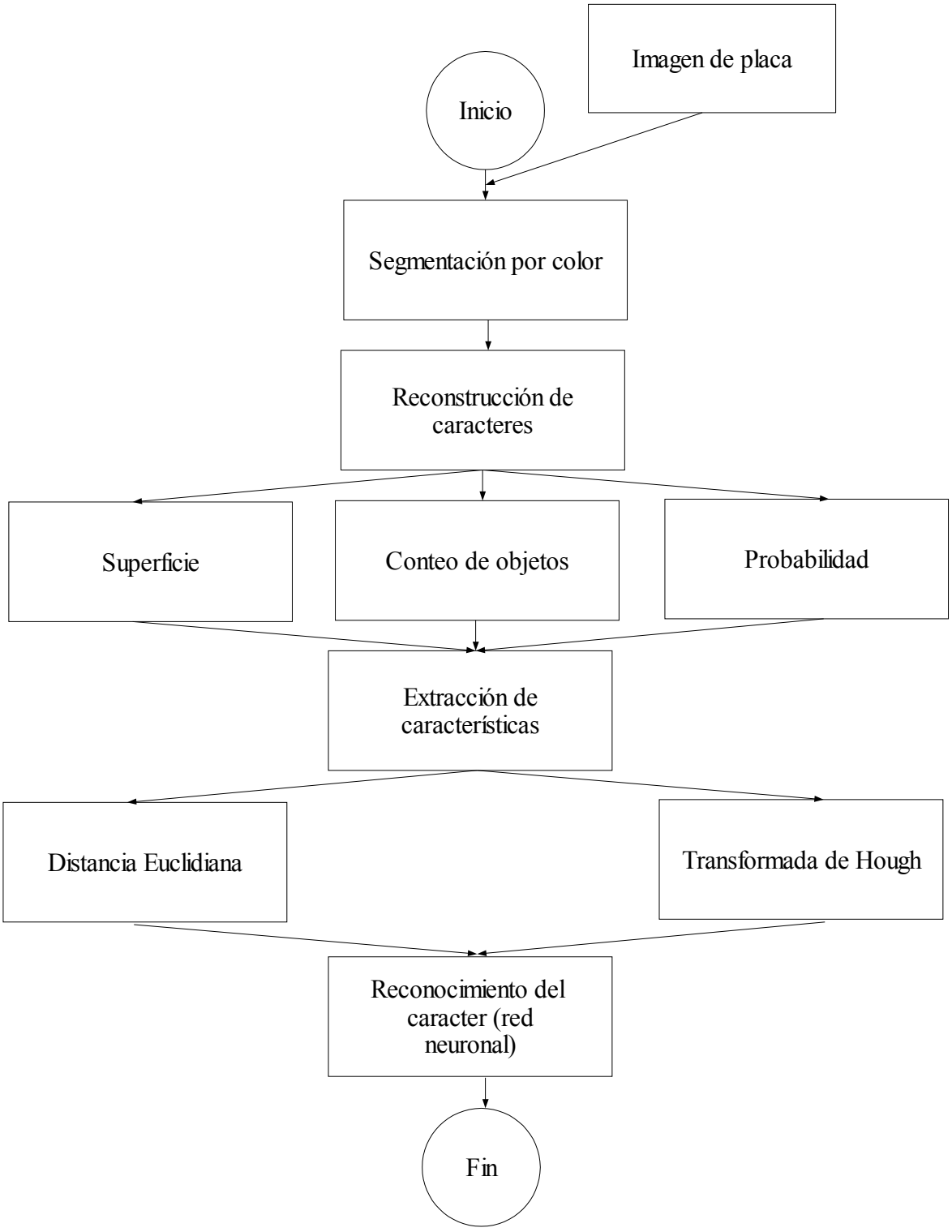

***Figura 15**: Método de reconocimiento de placas*

5. Detección y Localización de Placas.

5.1. Introducción.

En este capítulo se describe el proceso de detección y localización de placas. Inicialmente, el sistema recibe una imagen que puede contener cero o más placas en escenas complejas (coches al aire libre y en movimiento). Es importante resaltar que en los artículos mencionados en el estado del arte, no se mencionan casos donde la placa no ha sido localizada, esto es, en la mayoría de los casos se asume que dada una imagen, en esta siempre esta presente una placa.

En este capítulo se describe inicialmente la etapa de preprocesamiento, posteriormente se menciona el proceso de detección de placas, donde tres diferentes métodos fueron desarrolladas. El primero busca bordes verticales en la imagen, posteriormente busca conjuntos que tengan la altura y proporciones horizontales adecuados para ser candidatos de ser placas. Las otras dos técnicas utilizan regiones de decisión de color (RDC), esto es, segmentan la imagen en tres colores: color de fondo de placa, color de letras de placa y el resto; y posteriormente se utiliza una red neuronal para analizar la forma de los objetos de colores y obtener una clasificación final de color placa y otro.

El segundo método utiliza una red neuronal para llevar a cabo la segmentación por color. Se genera un conjunto de colores manualmente que después son entrenados por una red neuronal para que los discierna en las imágenes. Esta técnica se especializó en reconocer placas bajo las condiciones más variadas de iluminación

El tercer método genera regiones de decisión poligonales tridimensionales en el espacio RGB para segmentar la imagen por color. Las regiones de decisión son generadas de manera automática por un algoritmo genético a partir de una serie de imágenes de placas cortadas a mano. Esta técnica se especializó en una sola fuente de iluminación.

5.2. Preprocesamiento.

El preprocesamiento es el trabajo que se realiza en una imagen para mejorar su calidad con el fin de hacerla más adecuada para la tarea a realizar.

El contraste en las placas que localiza el sistema es muy bajo, por lo tanto, el contraste es una de las cosas que se intentó mejorar en las imágenes para aumentar el éxito del localizador de placas. La técnica que se eligió para esta tarea fue la ecualización.

Para algunos de los algoritmos que se aplicaron a las imágenes, es necesario extraer sus bordes. Ésto se hizo extrayendo el canal de intensidad de las imágenes primero y después usando un *kernel* de convolución para extraer los bordes del canal de intensidad de la imagen; reduciendo el ruido y engrosando los bordes simultáneamente.

El siguiente paso consiste en *binarizar* la imagen utilizando dos umbrales, el algoritmo utilizado en esta etapa es una modificación del método de *umbralización* óptimo, el cual fue desarrollado en este trabajo con el fin de separar la información de los bordes de las imágenes del fondo.

Finalmente se elimino ruido de la imagen *binarizada* mediante la eliminación de bordes pequeños e irrelevantes que suelen ser generados por información del ambiente en las imágenes, tales como irregularidades en el pavimento u hojas en los árboles.

5.2.1. Ecualización.

La imagen a color original es ecualizada con el fin de resaltar los bordes verticales en caso de que el contraste original no sea suficientemente bueno.

Con el fin de evitar el cambio en los colores de la imagen al momento de ecualizar, se ecualiza el histograma de el canal de intensidad de la imagen primero y después se aplica la función "$p_r(r)$" obtenida de esta ecualización a cada uno de los canales rojo, verde y azul; tal que todos sean modificados proporcionalmente sin alterar los tonos.

5.2.2. Detección de Bordes.

Una vez ecualizada la imagen, esta se transformo a niveles de gris para llevara a cabo la detección de bordes. En esta etapa se propuso una matriz de convolución propia, basada en el operador de Sobel para extraer los bordes verticales de la imagen. La modificación se hizo con el fin de reducir ruido y engordar ligeramente los bordes. La reducción de ruido es necesaria debido a la gran cantidad de información que se encuentra presente en las imágenes; los automóviles suelen estar rodeados de fondos complejos y llenos de detalle que hacen el trabajo de localización de la placa más difícil y tardado. El engrosamiento de los bordes, consecuencia inevitable de la reducción de ruido, es útil más adelante porque los objetos demasiado delgados serán eliminados.

El *kernel* modificado se obtuvo experimentalmente, y se muestra en la figura 16.

0	-1	0	1	0
0	-2	0	2	0
0	-4	0	4	0
0	-2	0	2	0
0	-1	0	1	0

***Figura 16**: Kernel del operador de Sobel modificado*

El resultado de recorrer la imagen con el *kernel* de convolución desarrollado contiene valores tanto positivos como negativos donde, en general, los valores extremadamente positivos y negativos contienen los bordes y los valores cercanos a cero contienen el fondo de la imagen.

La imagen es entonces normalizada al rango [0,255], donde los valores muy negativos se convierten en valores cercanos a cero, los valores muy positivos se vuelven cercanos a 255 y los cercanos a cero se contienen cerca de los valores medios del histograma.

5.2.3. *Binarización.*

Una vez que los bordes han sido detectados, se lleva a cabo la *binarización* de la imagen. Debido a la normalización de la imagen, presentada en el proceso de detección de bordes, se tuvo que hacer una modificación al algoritmo óptimo de *binarización*, para contemplar estos cambios.

La principal diferencia entre el algoritmo óptimo original y el modificado, es que éste regresa dos umbrales al final, tal que la imagen se pueda dividir en tres regiones (baja, media y alta) en vez de

dos (objetos y fondo), como se hace con el óptimo. Las regiones baja y alta son combinadas en una sola, que se convierte en la región de objetos del algoritmo óptimo convencional; mientras que la región media se vuelve la región fondo. Esta modificación surgió de la necesidad de separar todos los bordes de la imagen que, como ya se explicó, se encuentran tanto en las regiones bajas como en las altas del histograma.

La manera en la que el algoritmo modificado encuentra los umbrales es extendiendo el concepto del algoritmo óptimo original (que encuentra un centroide utilizando la media de todo el histograma; y después dos más utilizando la media de cada una de las regiones divididas por el primer centroide) para la búsqueda de tres centroides, que tengan dos medias entre ellos y por lo tanto dividan a la imagen en tres.

Al igual que en el algoritmo original, el primer paso es calcular la media de toda la imagen para encontrar un umbral inicial; después, se calcula la media de cada una de las regiones divididas por el umbral inicial, para obtener dos umbrales intermedios.

A continuación, se divide la imagen en tres regiones a partir de estos dos umbrales intermedios. Se calcula la media de cada una de estas regiones, que representa el centroide de cada una. Acto seguido, se encuentra la media del primer centroide con el segundo, y del segundo con el tercero; que representan la zona con menor probabilidad de pertenecer a cualquiera de las dos zonas contiguas en las que se encuentra; y éstos dos valores se utilizan como nuevos umbrales intermedios.

El proceso se repite hasta que el valor de los dos umbrales intermedios no cambie en iteraciones sucesivas. Cuando esto sucede, los dos umbrales intermedios son los umbrales finales y son utilizados para *binarizar* la imagen.

A continuación se muestra el proceso a manera de algoritmo:

Paso 1. Seleccionar un estimado del umbral “T”; por ejemplo, el promedio de intensidad de la imagen

Paso 2. Utilizando el umbral “T”, particionar la imagen en dos regiones: “R1” y “R2”.

Paso 3. Calcular los valores medios de gris “μ_1“ y “μ_2” de “R1” y “R2”, para estimar dos umbrales diferentes, “T1” y “T2”

Paso 4. Utilizando “T1” y “T2”, particionar la imagen en tres regiones: “Rf1”, “Rf2” y “Rf3”.

Paso 5. Calcular los valores medios de gris de “Rf1”, “Rf2” y “Rf3”, “μ_1”, “μ_2“ y “μ_3”

Paso 6. Seleccionar dos nuevos umbrales de acuerdo a la ecuación 34

$$T1 = \frac{\mu_1 + \mu_2}{2} \quad T2 = \frac{\mu_2 + \mu_3}{2} \tag{34}$$

Paso 7. Repetir los pasos 4-6 hasta que los umbrales no cambien en iteraciones sucesivas.

Una modificación adicional que se hizo al algoritmo de manera heurística, fue agregar un factor de ponderación a cada una de las regiones con el fin de reducir el ruido que aún se encontraba en las imágenes. El factor de peso extiende los bordes de una región de la misma manera que contrae los de sus vecinos. Se encontró que para la región central (que se convierte en fondo), un valor de 1.05 en sus límites inferiores y superiores dio los mejores resultados.

El resultado de aplicar este algoritmo a una imagen de bordes se puede ver en la figura 17.

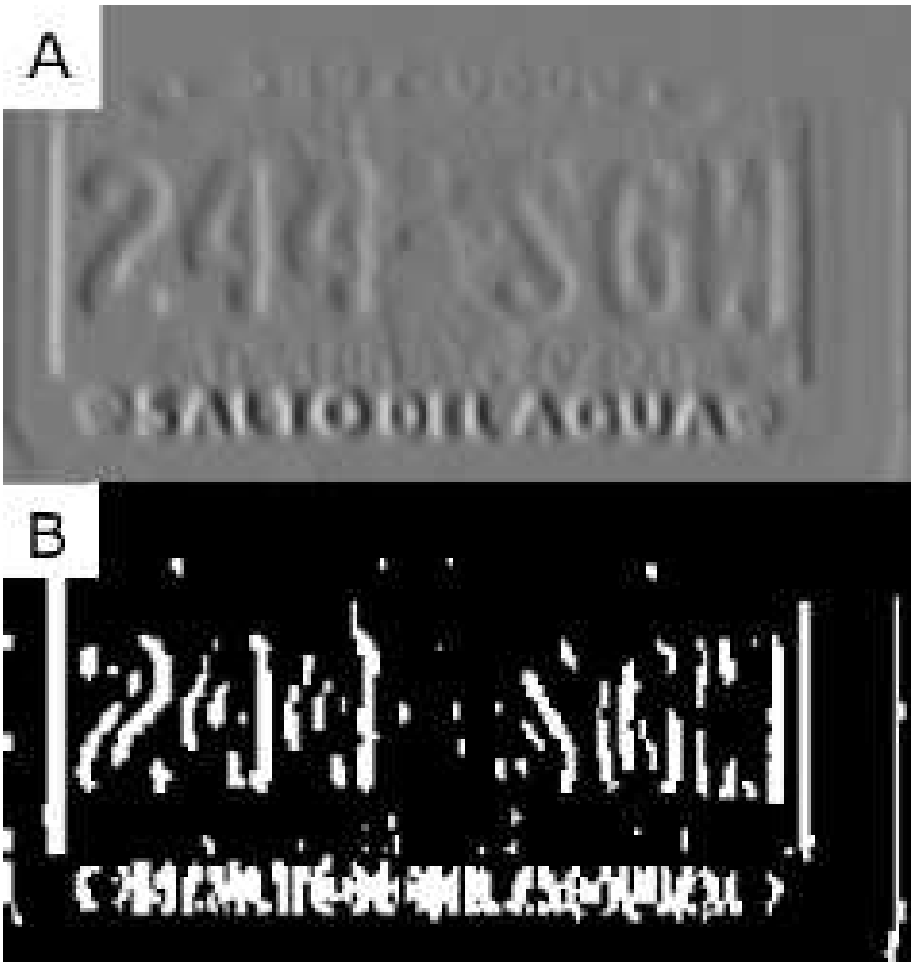

***Figura 17**: Binarización con el algoritmo desarrollado*
a)Imagen original b)Imagen binarizada

5.2.4. Reducción de Ruido.

En una gran cantidad de ocasiones, la imagen resultante de aplicar el algoritmo óptimo modificado aún contiene gran cantidad de ruido. Para fines de esta sección, el ruido se define como una gran cantidad de objetos pequeños, irrelevantes, que se derivan de las complejidades del ambiente natural en el que se toman las imágenes.

Se ideó un algoritmo para reducir de ruido a partir de una ventana que se desliza sobre la imagen. No se espera que este algoritmo elimine todo el ruido, sino que lo reduzca para evitar que se gaste un mayor tiempo computacional más adelante en el procesamiento de estos objetos que, ya se sabe, no sirven ningún propósito para la localización de la placa.

El algoritmo desliza una ventana que remueve cualquier objeto que encuentre que quede completamente contenido dentro de la misma.

La manera rápida de determinar si un objeto realmente está completamente contenido dentro de la ventana es revisar si todos los bordes de la ventana están libres de información (contienen únicamente ceros). Cuando esta condición se cumple, la ventana entera es llenada con ceros.

La condición de que los bordes deban estar completamente limpios tiene como efecto secundario que para una ventana de tamaño N, se removerán objetos de tamaño N-1 como máximo.

Se hicieron pruebas con diferentes tamaños de ventana y se determinó que se obtuvieron los mejores resultados utilizando una configuración en dos etapas, en la primera de las cuales se utilizó una ventana de 3×3, y en una segunda etapa una de 7×7.

Se observó que se los resultados podían mejorarse marginalmente utilizando una configuración de tres etapas, con ventanas de 3×3, 5×5 y 7×7 en ese orden; sin embargo, el impacto que tuvo en el desempeño del sistema no compensó la marginal mejora que ofreció, así que se decidió no usar esta configuración; especialmente en vista de que, como se especificó, este algoritmo tiene como fin mejorar el desempeño del programa más que liberar a la imagen de bordes completamente de ruido.

El algoritmo no garantiza remover todo el ruido, incluso si éste es de tamaño N-1 o menor; esto se debe a que cuando dos objetos se encuentran cercanos, sobreponiéndose en el eje “X” o “Y”, no permiten que los bordes de la ventana queden completamente limpios, incluso si ambos objetos son realmente pequeños. Es por ésto que es necesario recorrer la imagen con más de una ventana, y siempre con tamaños más pequeños primero.

Los resultados de esta etapa pueden verse en la figura 18, en las que se utilizaron las ventanas de tamaño 3 y 7.

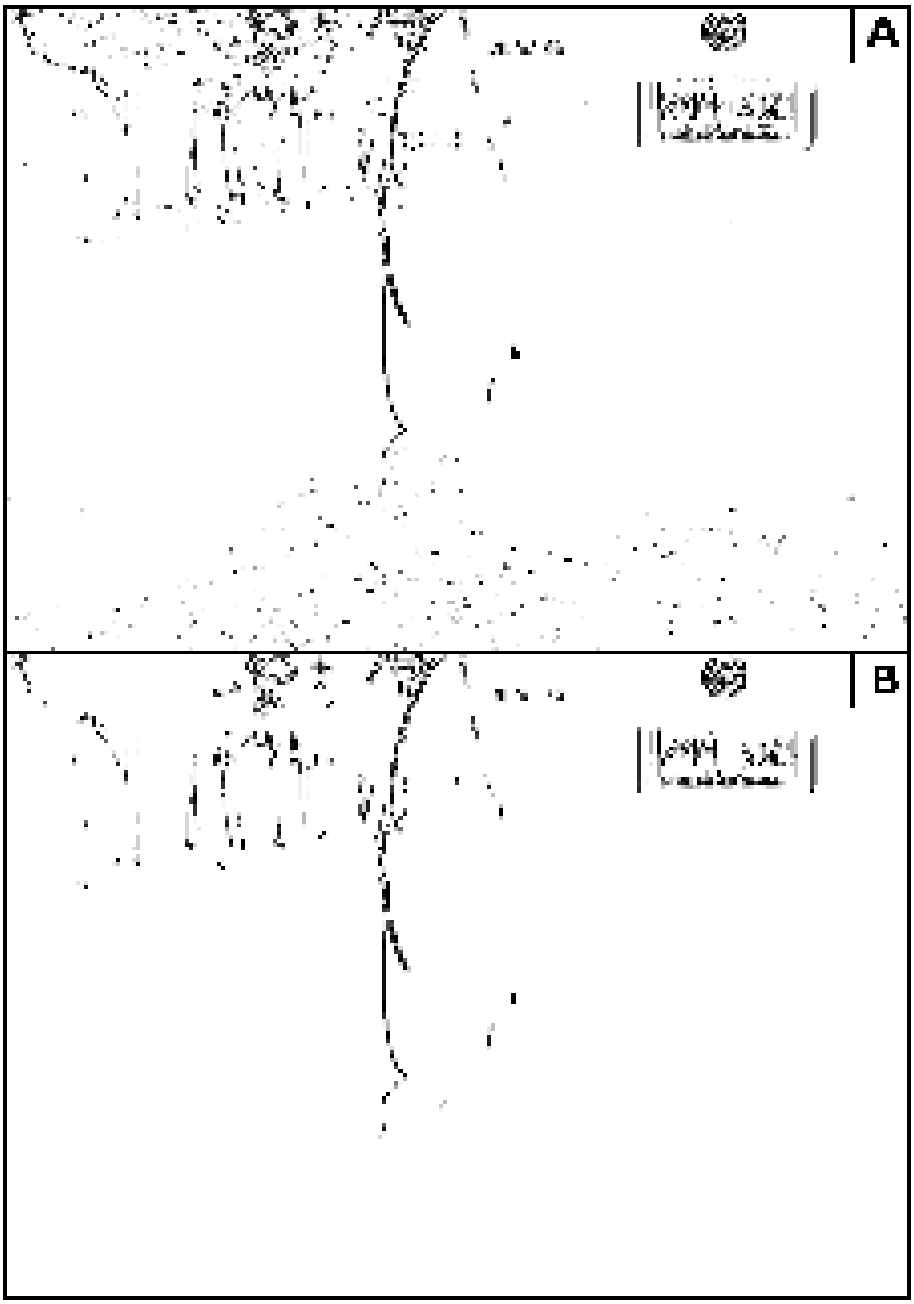

***Figura 18**: Reducción de ruido con ventana*
a)Imagen original b)Imagen sin ruido

5.3. Detección y Localización de Placas.

Se propusieron dos métodos para la localización de las placas. Uno es similar a [ZHE04] en que utiliza bordes verticales para la localización de las placas. Además de los bordes utiliza información de la razón de altura contra anchura de la región de interés.

El segundo lleva a cabo una segmentación por color para separar los objetos que puedan ser

placas del resto, y es complementado por una red neuronal que finalmente decide qué es una placa y qué no lo es.

5.3.1. Método Basado en Bordes Verticales.

Este método para encontrar placas a partir de la imagen de bordes con ruido reducido consiste en la búsqueda de líneas verticales que tengan aproximadamente la misma longitud, las mismas coordenadas "y", y se encuentren separadas por una razón de alto contra ancho de aproximadamente 2:1

Hay un número de factores que obligan a que se utilicen parámetros muy laxos para cada parte de este proceso, tales como el uso de portaplacas en los coches, placas que se encuentran dobladas, chuecas o despintadas y ruido en el proceso de adquisición de las imágenes; algunos de estos factores ocasionan que raramente se encuentran líneas realmente verticales; el uso de portaplacas deforman el radio de proporción, dejando un rango de 1.8:1 a 2.6:1.

El resultado de relajar los parámetros de detección para considerar estos problemas es que se encuentra un gran número de falsos positivos; es decir, objetos que parecieran ser placas pero no lo son.

5.3.1.1. Conteo de Objetos.

Todos los objetos en la imagen reducida de ruido son separados y guardados para su procesamiento individual. Para esto se utilizó conexidad cuatro para segmentar las regiones contiguas. Existen dos razones principales para utilizar conexidad cuatro sobre otros esquemas:

La conexidad cuatro es menos intensiva, computacionalmente hablando. Se tienen que visitar menos vecinos por píxel y, en total, menos píxeles por región. Esto repercute en que tenga menores requerimientos de tiempo de procesamiento y de memoria.

Debido a que se buscan líneas rectas verticales, no solo es innecesario sino indeseable visitar píxeles que se encuentren en la vecindad diagonal. Al visitar éstos es muy probable que en realidad se esté visitando algún otro objeto contiguo que se encuentre cerca del que originalmente se estaba segmentando, en vez de seguir con el mismo.

La conexidad cuatro tiene el riesgo de que si una línea es muy delgada y se encuentra posicionada a un ángulo muy pequeño (muy cercano a cero, o muy cercano a 90), el objeto se separe y

se clasifique como dos objetos diferentes. Esto no sucede en el caso de estas imágenes de bordes debido a que el *kernel* de convolución no sólo encuentra bordes verticales, sino que los hace un poco más gruesos, por lo tanto incluso para el caso de ángulos pequeños los bordes siguen siendo una sola región bajo conexidad 4.

Además de almacenar cada objeto individual en una lista, se guarda información acerca de los mismos, tales como posición y dimensiones; que se obtienen al momento de segmentar.

5.3.1.2. Segunda Reducción de Ruido.

El algoritmo de reducción de ruido en la etapa de preprocesamiento es insuficiente para librar a la imagen de ruido, como ya se mencionó, debido a su inhabilidad de remover ruido cuando dos patrones pequeños (o uno pequeño y uno grande) se encuentran traslapados. Su objetivo es agilizar la segmentación de la imagen, más que realmente eliminar el ruido de la imagen.

Una vez segmentada la imagen, teniendo cada objeto por separado junto con su descripción, es posible implementar de manera sencilla un algoritmo más efectivo de reducción de ruido

El segundo método utilizado para reducir el ruido de la imagen consiste en eliminar objetos cuya altura sea menor a un valor calculado a partir de un coeficiente ponderado del ancho del objeto. De manera experimental se determinó que el mejor coeficiente es 2.5; esto es, que los objetos deben ser al menos 2.5 veces más altos que anchos; de lo contrario deben ser eliminados. Esto es, sea h_i= altura de objeto i, a_i= ancho objeto i, el objeto i se elimina si $h_i < 2.5\ a_i$

Originalmente se consideraron coeficientes mayores; sin embargo, debido a las inconsistencias de las placas, especialmente por los portaplacas, tuvo que reducirse el número a 2.5, pese a que éste deja pasar también una buena cantidad de ruido.

Además del coeficiente, cualquier objeto cuya altura sea menor a un valor predeterminado, fijo en 15 píxeles, es borrado; independientemente de si su altura es o no 2.5 veces su ancho. Se observó que objetos con alturas menores a la establecida son ilegibles para la computadora.

5.3.1.3. Análisis de Grosor de Líneas.

Una vez eliminado el ruido, se analizan aquellos objetos restantes, eliminando aquellos que tiendan a ser mas gruesos y por lo tanto poco probables de ser líneas verticales. En general se observó que las líneas más horizontales tienden a ser más gruesas que las verticales, como se muestra en la figura 19

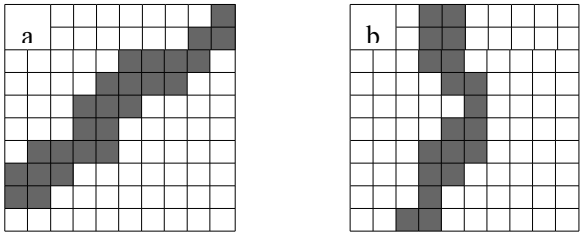

***Figura 19**: Grosor de lineas*
a) horizontales b)verticales

Se obtiene el histograma de proyección horizontal (figura 20) de cada uno de los objetos. De cada uno de los objetos seleccionados, se elimina el 5% de los valores superiores de su histograma de proyección horizontal. A partir del 95% restante se obtiene el ancho promedio de los objetos (líneas) restantes. Experimentalmente se obtuvo que los objetos de interés presentaron un grosor máximo de 7 píxeles; por lo que si el grosor medio de un objeto es menor al grosor máximo permitido, y la proporción del grosor medio es menor o igual al 5% de la altura, entonces el objeto permanece, en otro caso es eliminado.

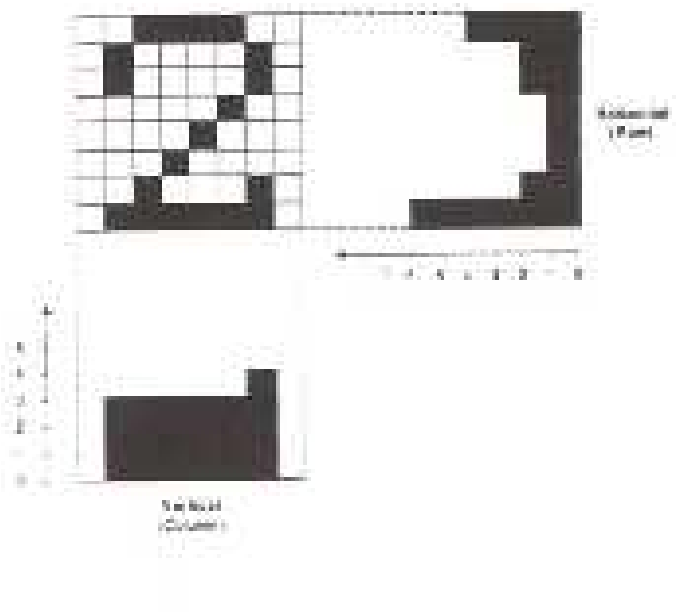

***Figura 20**: Histograma de proyección horizontal y vertical de imagen binarizada del dígito 2*

5.3.1.4. Adelgazamiento Horizontal.

Se recorre el objeto línea por línea y, en vista de que ya se sabe que es un solo objeto y no será necesario volverlo a extraer, por lo que no debe presentarse ningún problema si el objeto llegara a separarse; se reduce el grosor de las líneas a 1 píxel. El píxel que se deja activo es el que está en medio de los píxeles originales. A manera de algoritmo:

1. Se obtiene la proyección horizontal H del objeto i

2. Para cada valor h_i de H, borrar píxeles de acuerdo a 35

$$si\ h_i > 1, borrar\ píxeles\ laterales, \begin{cases} si\ h_i \% 2 = 0 & dejar\ píxel\ a\ la\ izquierda\ del\ centro \\ si\ h_i \% 2 \neq 0 & dejar\ el\ píxel\ central \end{cases} \tag{35}$$

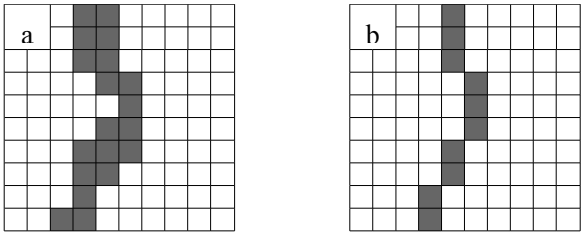

Figura 21*: Adelgazamiento de líneas verticales*
a) Imagen original b)Imagen adelgazada

Una vez que los objetos son adelgazados, se eliminan aquellos objetos cuyo ancho sea mayor al 5% de su altura, esto es aquellos objetos que no son candidatos a ser lineas verticales.

La figura 22 muestra un ejemplo de una imagen al llegar a esta etapa.

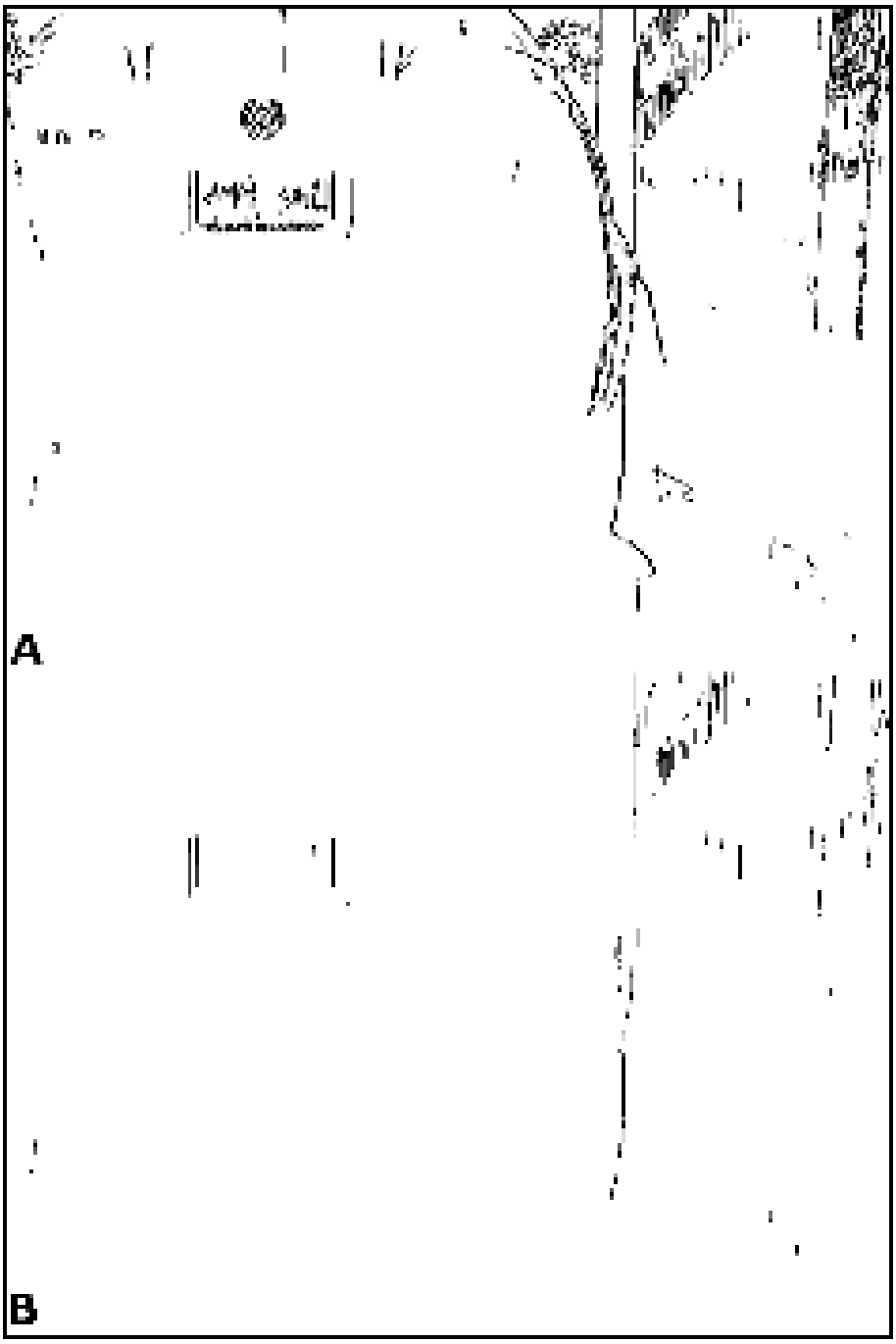

***Figura 22**: Preparación para localizar la placa*
a)Imagen original b)Imagen transformada

5.3.1.5. Selección y Reducción de Candidatos a Placas.

Los objetos que no han sido eliminados, son ordenados utilizando un índice basado en su ancho

y posición vertical (coordenada “y”), por lo que los elementos que se encuentren contiguos en la lista y además cuyas diferencias en altura y coordenada “y” no excedan cierto umbral, son agrupados para formar los objetos candidatos a placas.

Longitud	Coord. Y	Objeto
10	1	A
10	1	B
3	14	C
6	14	D

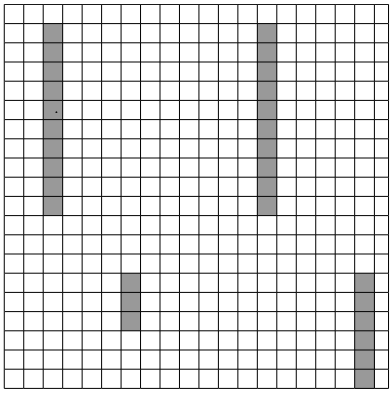

Figura 23: Selección de candidatos a placas. Los objetos A y B son un candidato

De los objetos candidatos se eliminan todos aquellos pares cuya relación ancho/alto, se encuentre fuera del rango 2:1.

La figura 24 muestra un ejemplo de los candidatos extraídos de una imagen

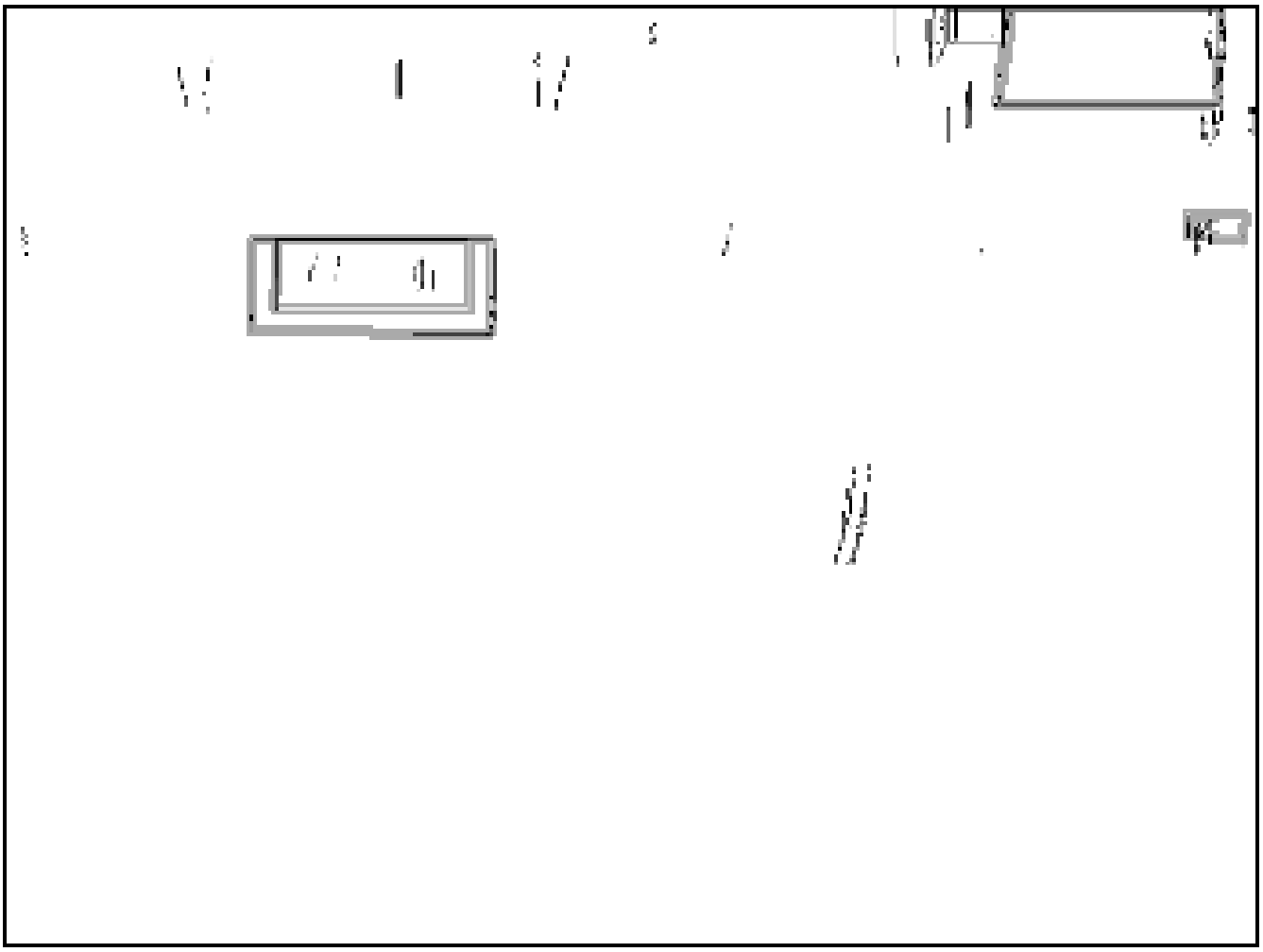

***Figura 24**: Candidatos de placa de automóvil*

5.3.2 Métodos Basados en Color.

Estos métodos se basan en los colores de las placas y la construcción de regiones de decisión de color (RDC) en el espacio RGB.

La imagen es segmentada por color de acuerdo a los rangos de color en que se determinó que pertenecen a las placas, y en aquellos en los que no. Cada región etiquetada como placa, es posteriormente clasificada por una red neuronal de acuerdo a le región de decisión generada para determinar si en efecto es una placa de automóvil o no.

Se desarrollaron y evaluaron dos métodos para la construcción de regiones de decisión de color (RDC). En el primer método se selecciono manualmente el conjunto de colores que pertenecen a placa y posteriormente una red neuronal fue utilizada para su clasificación. En el segundo método la región de decisión se construyo mediante un algoritmo genético.

Es importante mencionar que en el método basado en redes neuronales con el conjunto generado manualmente se utilizó una mayor cantidad de fuentes y tipos de iluminación posibles; mientras que para el método con algoritmos genéticos, se seleccionó una condición de iluminación mas uniforme bajo ambiente de luz natural en escenas complejas; esto es, en la calle.

5.3.2.1 Segmentación por Color Utilizando Redes Neuronales.

Para la segmentación por color utilizando redes neuronales, se extrajo un conjunto de colores pertenecientes a las placas manualmente; entonces, un conjunto de colores que no pertenecen a las mismas fue generado de manera automática y, finalmente, ambos fueron utilizados para entrenar una red neuronal con la capacidad de discernir entre ambos.

a) Construcción de la Región de Decisión de Color (RDC) y Conjuntos de Entrenamiento.

Los colores que forman parte de la placa se obtuvieron a partir de 63 imágenes de vehículos en los que se mostraba la placa posterior. Las imágenes fueron tomadas a diferentes distancias, de diferentes ángulos y con diferentes condiciones de iluminación. Algunas fueron tomadas con la luz directa del sol, otras con luz natural y cielo nublado, otras con lámparas incandescentes y otras aún con luz fluorescente.

De cada una de las imágenes, se extrajo una pequeña subimagen de la placa, para formar una imagen mosaico que contendrá muestras de los colores de placa. El histograma tridimensional de la imagen de las placas fue reducido a 255 colores utilizando el método *nearest color*. De este conjunto reducido de colores, se seleccionaron manualmente aquellos que pertenecen al fondo y a las letras de las placas, y se descartaron colores que no pertenecen a ninguno que pudieron haberse encontrado aún. La selección se hizo mediante el reemplazo directo en la paleta de la imagen de cada una de sus entradas por colores de prueba. Se hizo una examinación visual del resultado de reemplazar cada uno de estos colores y así se determinó su pertenencia a una de cuatro clases: colores de fondo de placa, colores de letras, colores principalmente de fondo y colores principalmente de letras.

A continuación se generó el conjunto de colores que no pertenecen a las placas. Para lograr ésto, se generaron dos regiones de decisión temporales, Rt1 y Rt2 mediante la combinación de las clases de colores de fondo con colores principalmente de fondo y de colores de letras con la de colores principalmente de letras. Se generó una gran muestra de colores de manera automática. Se evaluó si cada uno de los colores generados se encontraba contenido dentro de las regiones Rt1 y Rt2. Las

muestras que se encontraban contenidas dentro de la forma fueron descartadas; las muestras que cayeron fuera de la figura fueron agregadas al conjunto de colores de no placa, una quinta clase. Finalmente, las muestras que tocaron los bordes del polígono fueron descartadas también, ya que se buscó que la red neuronal determinara los bordes a través del entrenamiento; y el uso de éstos hubiera constituido sobreentrenamiento. El proceso se ilustra en la figura 25.

La construcción de las regiones R_{t1} y R_{t2} se efectuó de la siguiente manera:

1.- Se combinaron los conjuntos de fondo y principalmente de fondo en un conjunto temporal S_1, y los conjuntos de letras y principalmente de letras en un conjunto temporal S_2.

2.-Para los conjuntos S_1 y S_2:

3.- Se generaron todas las combinaciones posibles S' de 4 elementos sin orden de los conjuntos

4.- Cada " S' " fue evaluada con el resto de los elementos del conjunto para determinar si éstos se hallaban completamente contenidos dentro de la región tridimensional en el espacio RGB ubicada entre los 4 elementos, utilizando la fórmula descrita en la ecuación 36; donde "S'_1", "S'_2", "S'_3" y "S'_4" son los elementos 1, 2, 3 y 4 del conjunto " S' ", que además son puntos en el espacio RGB; sus componentes "r", "g" y "b" se expresan como "x", "y" y "z"; "p" es el elemento a evaluar, "p.x", "p.y" y "p.z" son sus componentes "r", "g", "b". "a_{123}", "b_{123}", "c_{123}" y "d_{123}" son los parámetros "a", "b", "c" y "d" del plano definido por los puntos "S'_1", "S'_2" y "S'_3" de acuerdo a la ecuación del plano (37). "sgn_{123}" representa el signo del vector normal al plano definido por los puntos "S'_1", "S'_2" y "S'_3" y que pasa por el punto "S'_4." "sgn_{p}" representa el signo del vector normal al plano definido por los puntos "S'_1", "S'_2" y "S'_3" y que pasa por el punto "p".

$$
\begin{aligned}
d_{123} &= a_{123}x + b_{123}y + c_{123}z \\
&\text{Se determina el valor de } a_{123}, b_{123} \text{ y } c_{123} \\
&\text{substituyendo x, y, z de } S'_1, S'_2 \text{ y } S'_3 \\
&\text{y resolviendo el sistema de ecuaciones} \\
sgn_{123} &= \begin{cases} 1 & si \quad d_{123} < (a_{123} \times S'_4.x) + (b_{123} \times S'_4.y) + (c_{123} \times S'_4.z) \\ 0 & si \quad d_{123} = (a_{123} \times S'_4.x) + (b_{123} \times S'_4.y) + (c_{123} \times S'_4.z) \\ -1 & si \quad d_{123} > (a_{123} \times S'_4.x) + (b_{123} \times S'_4.y) + (c_{123} \times S'_4.z) \end{cases} \\
sgn_{p_{123}} &= \begin{cases} 1 & si \quad d_{123} < (a_{123} \times p_4.x) + (b_{123} \times p_4.y) + (c_{123} \times p_4.z) \\ 0 & si \quad d_{123} = (a_{123} \times p_4.x) + (b_{123} \times p_4.y) + (c_{123} \times p_4.z) \\ -1 & si \quad d_{123} > (a_{123} \times p_4.x) + (b_{123} \times p_4.y) + (c_{123} \times p_4.z) \end{cases}
\end{aligned}
\tag{36}
$$

si sgn_{123} *es igual a 0, se descarta S' debido a que no define una figura de 3 dimensiones, sino un plano*

si sgn_p *es igual a 0, se descarta p debido a que está tocando el plano*

Se repite el proceso para obtener

$$sgn_{124},\ sgn_{134},\ sgn_{234} \text{ y } sgn_{p124},\ sgn_{p134},\ sgn_{p234}$$

Finalmente, p está contenido dentro de S' si

$$(sgn_{123} = sgn_{p123}) \wedge (sgn_{124} = sgn_{p124}) \wedge (sgn_{134} = sgn_{p134}) \wedge (sgn_{234} = sgn_{p234})$$

b) Entrenamiento y Localización de Candidatos.

Los 5 conjuntos de colores: colores de letras de las placa (letra y principalmente letra), colores de fondo de placa (placa y principalmente placa), y colores que no pertenecen a la placa fueron utilizados como entradas para entrenar una red neuronal multicapa entrenada con Back Propagation. La red lleva a cabo la clasificación de un color en letra, placa y otro. La configuración de la red fue de tres nodos de entrada, tres nodos de salida y 128 nodos en la capa oculta. La configuración fue obtenida por experimentación, se eligió por ser la configuración con menor número de neuronas que produjo resultados satisfactorios.

Como trabajo posterior se propone ajustar los parámetros de la red neuronal. Es probable que se pueda utilizar una con un menor número de neuronas intermedias; sin embargo, en la experimentación que se realizó se consideraron diferentes arquitecturas para la red y fue la de 128 nodos intermedios la que mejor separó todos los conjuntos de colores con un número mínimo de errores.

La red neuronal es entonces usada para clasificar los colores de las imágenes de entrada, para crear regiones de color contiguas que se convierten en candidatos que potencialmente pueden contener una placa. De estas regiones candidatas son eliminadas aquellas que no tienen una proporción alto ancho en un rango dado.

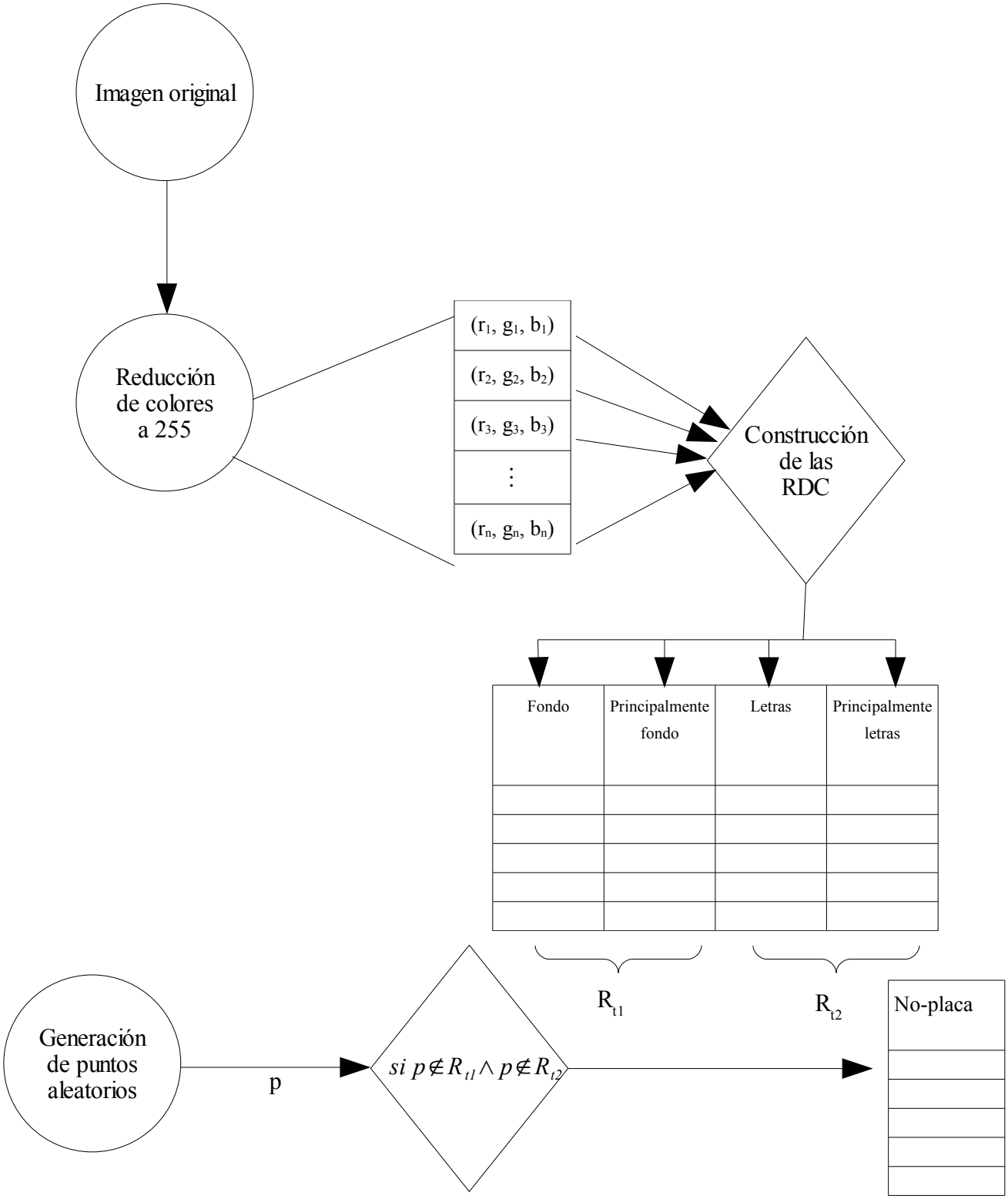

Figura 25: *Generación de conjuntos de entrenamiento*

5.3.2.2. Segmentación por Color Utilizando Algoritmos Genéticos.

Para la segmentación por color basada en regiones de decisión de color, se generaron 3 regiones de decisión, cada uno de 4 puntos en el espacio de color RGB. Las 3 regiones de decisión son: fondo de placa, letras de placa y colores que no pertenecen a las placas. Los conjuntos fueron afinados utilizando otra región de 4 puntos en el espacio RGB que intersectan a los conjuntos de fondo y placa y substraen los colores en los que se sobreponen; reasignando éstos al conjunto de colores que no pertenecen a las placas.

a) Construcción de la Región de Decisión de Color.

La base de datos de las imágenes utilizadas en esta etapa está formada por un conjunto de imágenes de entrenamiento conformado por 100 fotografías adquiridas bajo condiciones de iluminación similares, con luz solar intensa, tomadas entre las 11:00 y 13:00 horas, desde el mismo lugar para asegurar que la incidencia de los rayos de sol fuera homogénea a lo largo de las imágenes. Se tomó otro conjunto de fotografías bajo las mismas condiciones para conformar el conjunto de prueba.

Las placas de los vehículos fueron extraídas manualmente de las imágenes de entrenamiento. La extracción se hizo poniendo atención en dejar siempre una margen adicional alrededor de la placa para procurar incluir elementos del conjunto de colores que no pertenecen a las placas, pero están cerca de éstas.

Se utilizó un algoritmo genético para la generación de las regiones de decisión que se usarán para clasificar los colores de las imágenes. La meta del algoritmo genético es encontrar una configuración de regiones de decisión capaz de encontrar la placa en la mayor cantidad posible de las imágenes, y que al mismo tiempo, se asegure que las regiones de las placas no se extiendan más allá de sus bordes; es decir, que no invadan las zonas de la imagen al rededor de las placas donde la placa no se encuentra.

Para lograr separar la placa de esta manera, el algoritmo genético debe optimizar el conjunto de sólidos que logre:

- Maximizar el tamaño de la región de color más grande que encuentre en cada imagen. El objeto más grande siempre debe ser el fondo de la placa, incluyendo la imagen del “ángel” en el fondo. Puede o no incluir las letras; es mejor si no las incluye porque significa que el conjunto se ajusta

más cercanamente a el rango de colores que conforman el fondo de la placa; dando menos falsos positivos; sin embargo, no es grave si sí incluye el color de las letras en vista de que la segmentación de las mismas se realizará en una etapa posterior del proceso.

- Minimizar la cantidad de píxeles del objeto más grande (el fondo de la placa) que toquen los bordes de la imagen. Al lograr esto se asegura que la placa esté contenida en su totalidad en la imagen y no se extiende más allá de sus bordes (o, al menos lo hace lo menos posible). Es importante recalcar que no se trata de minimizar el número de píxeles que caen en la misma categoría en la que se clasifica al objeto más grande (definida por las regiones que la separen); es posible que haya píxeles del color del fondo de la placa que toquen las orillas, por ejemplo en automóviles dorados. Sin embargo, se desea que dichos píxeles no se encuentren conectados en ningún punto con el objeto que se reconoce como fondo de la placa, usando conexidad cuatro.
- Generar la menor cantidad de regiones que sea posible. De esta manera se intenta que tanto el "ángel" como las mitades derecha y izquierda del fondo de la placa se reconozcan como una sola región. Existe la posibilidad de que esta restricción resulte en que las letras se encuentren en la misma región que el fondo de la placa. Como se explicó con anterioridad, ése no sería un problema.

El algoritmo genético que se utilizó tiene las siguientes características

- Codificación

  Pese a que los componentes (rojo, verde y azul) de los colores que se utilizaron en cada paso del proceso son representados como enteros, se utilizaron números de punto flotante para construir las coordenadas de los cuatro vértices de los sólidos. La razón para hacerlo de esta manera es que la representación con únicamente 4 puntos por sólido de los rangos de color; pese a que se utilizan dos sólidos por categoría, es muy limitada. Puede provocar que los ángulos sean más agudos de lo que deberían en los extremos más alejados del centro. Por lo tanto, el uso de números reales debe ayudar al, por lo menos, incrementar la precisión de dónde exactamente deben estar los bordes de las secciones.

  Únicamente se utilizaron cuatro vértices de la región de decisión (el mínimo para definir un objeto en tercera dimensión) por cuestiones de desempeño. En vista de que el algoritmo genético tendrá que evaluar un gran número de píxeles para calificar la función objetivo de cada individuo, de que cada generación se compone de un gran número de individuos y de que el

sistema corre a lo largo de varias generaciones, existe una gran necesidad de un bajo costo computacional en ésta, que será la base de la función de aptitud.

Para la representación binaria, se codificaron los componentes de la posición de los 4 puntos por polígono, que son números de punto flotante, como cadenas binarias; una concatenada tras la otra.

Se utilizó la codificación del lenguaje de programación para representar números de punto flotante como cadenas binarias.

- Rango y precisión

  El rango de valores que se buscan es [-255,512].

  En vista de que los números binarios comprenden un rango muy alto de números, pero solo se necesitan números del 0 al 255, se convierten los números mayores al 512 o menores a -255 al rango [-255,512] en manera de *zig-zag*. Los números de -255 al 512 se toman como son, pero para números mayores se construye una serie de rangos que se decrementan e incrementan repetidamente de -255 a 512; de la siguiente manera: [512,-255][-255,512][512,-255][-255,512]...,etc.

  La precisión que se maneja es la precisión que proveen los números de punto flotante del lenguaje de programación.

  La precisión de un número de punto flotante disminuye al tiempo que éste crece, por lo tanto los rangos de [512,-255][-255,512] que se encuentren más cercanos a cero tendrán mayor precisión que los más alejados. No se observaron problemas por esta situación.

b) Segmentación.

Los sólidos resultantes son entonces utilizados para clasificar los colores de las imágenes del conjunto de prueba como colores de placa y de fondo. Con éstos, se agrupan las regiones y se forman candidatos de placas de acuerdo a los objetos que se encuentren con los sólidos que definen el color de fondo de placa.

Para determinar si el color de un píxel está o no contenido dentro de uno de los sólidos, se forma un plano con tres colores del sólido y se determina si el color restante es mayor o menor al plano de acuerdo a la ecuación general del plano (ecuación 37). Después, se determina si el color a evaluar es

también mayor o menor al plano. Si el color restante fue mayor, y el color a evaluar menor, o viceversa, se determina que el color no está contenido en el sólido; de lo contrario, el color se considera potencialmente dentro. Este proceso se repite para todas las combinaciones de planos y colores restantes del sólido, y únicamente se determina que el color está contenido en caso de estar "potencialmente dentro" para cada una de las combinaciones.

$$ax + by + cz = d \tag{37}$$

De manera más formal se puede describir al proceso de la siguiente manera:

Sea:

Fj: Número de pixels de la imagen j, que pertenecen a la Región de decisión del fondo.

Pj: Número de pixels de la imagen j, que pertenecen a la Región de decisión de la placa.

Tj: Número de píxeles de al imagen j, que no pertenecen al fondo ni a la placa.

Dj: Número de píxeles que son placa y tocan borde de la imagen.

Oj: Número de objetos detectados después de la segmentación.

La función objetivo para la segmentación de la imagen i, con las regiones de decisión de un individuo, esta dada por en la ecuación 38

$$S_i = W_1 Max\{F_j, P_j, T_j\} - W_2 D_j - W_3 O_j \tag{38}$$

La figura 26 muestra tres imágenes: una imagen original a color, la imagen segmentada utilizando el método basado en redes neuronales y la misma segmentada usando el método basado en geometría en el espacio RGB. La placa está remarcada en color rojo.

La figura 27 muestra tres imágenes: una imagen original, la imagen segmentada utilizando el método basado en redes neuronales y la imagen segmentada usando el método basado en geometría en el espacio RGB. El método basado en redes neuronales falló en encontrar la placa, pero el método basado en geometría sí la encontró.

Para las imágenes segmentadas, el color negro representa el conjunto de colores de placa y el blanco el de colores no de placa. La placa está remarcada en color rojo.

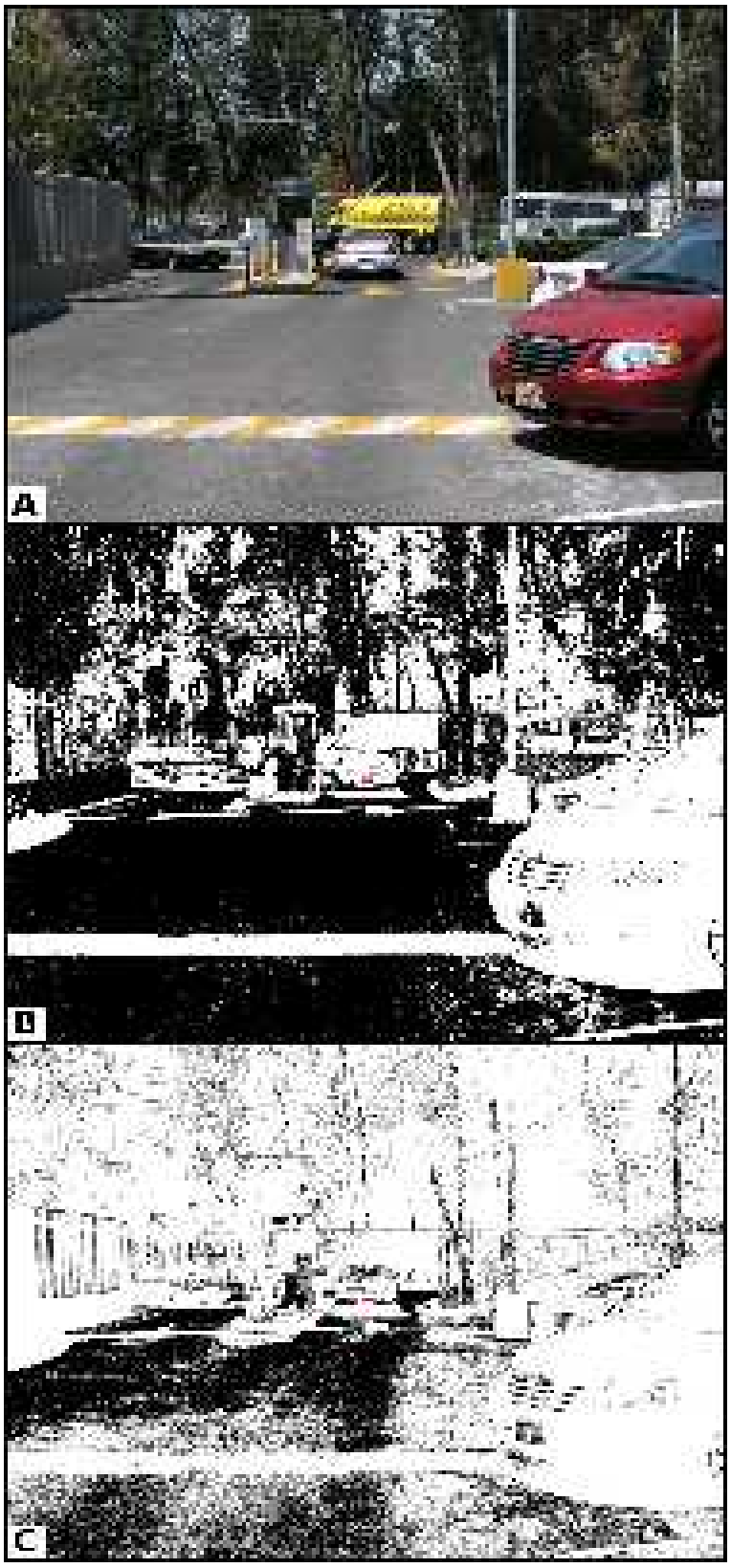

***Figura 26**: Segmentación por color.*

a)Imagen original b)Red neuronal c)Geometría

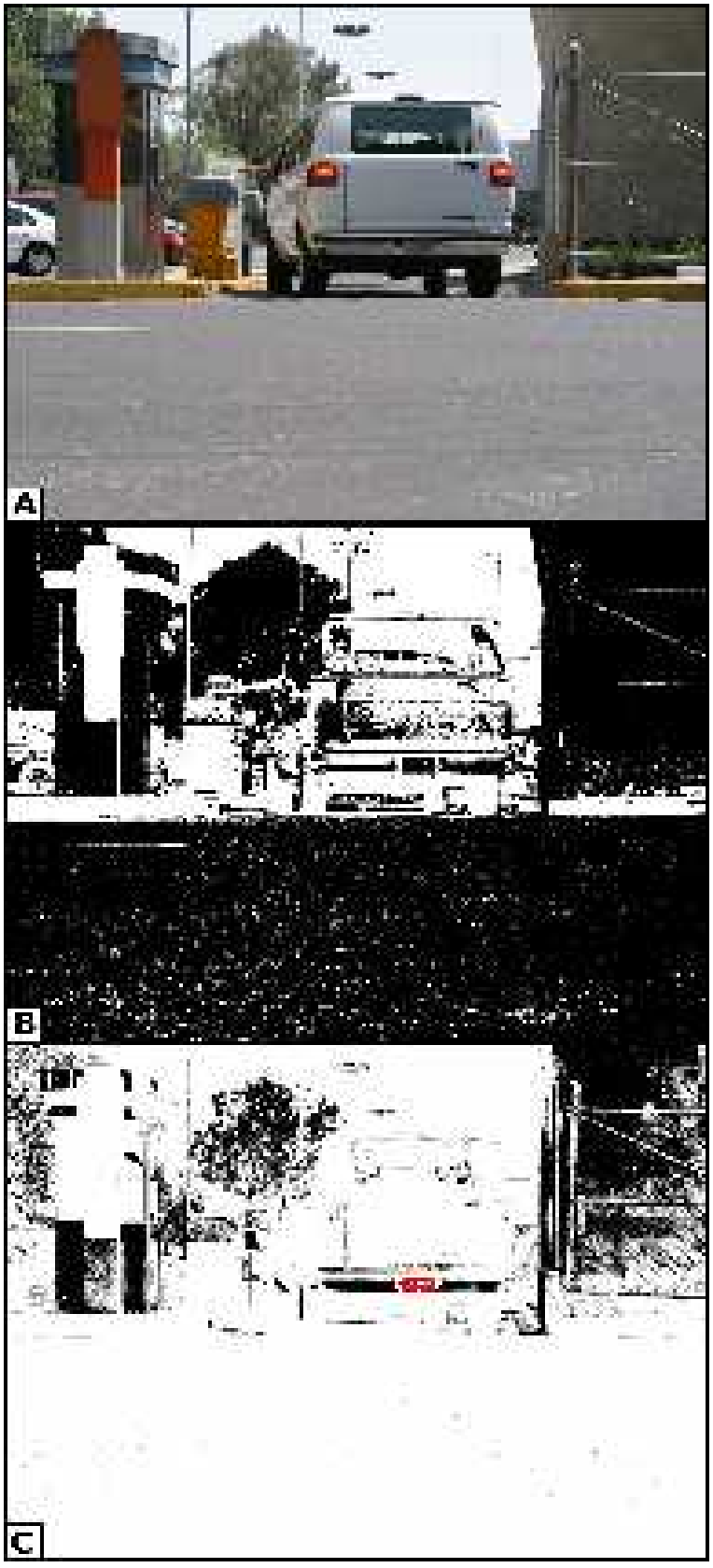

***Figura 27**: Segmentación por color.*

a)Imagen original b)Red neuronal (falló) c)Geometría

6. Reconocimiento de Placas.

6.1. Introducción.

Una vez localizada la placa, se lleva a cabo la segmentación de caracteres, los caracteres son segmentados utilizando información de color de la imagen original, en diferentes regiones que no necesariamente corresponden a caracteres completos por si solas. Después, mediante el uso de una función heurística, se reconstruyen los caracteres de manera que queden completos y con la menor cantidad de ruido posible.

En la etapa de identificación de caracteres, 20 características de los caracteres segmentados son extraídas y utilizadas como entrada a una red neuronal para su clasificación.

Se asume que los tres primeros caracteres son números y los últimos tres son letras, en vista de que así es el caso para el Distrito Federal, por lo que en la etapa de reconocimiento se utiliza una red para el reconocimiento de letras y otra para el reconocimiento de números.

6.2. Segmentación de Caracteres.

La segmentación de caracteres es el proceso que consiste en separar los caracteres de una placa de entre sí mismos y del fondo, dejando, típicamente, sólo una imagen binaria por caracter, lista para la extracción de características.

Como se mencionó en el proceso de segmentación de placas, se examinaron 60 imágenes de muestra de placas de automóvil, cortadas a mano, tomadas en diversas condiciones de iluminación; para generar una tabla de colores con las siguientes categorías:

- Colores de fondo
- Colores de letras (frente)
- Colores frecuentemente de fondo
- Colores frecuentemente de letras

El tamaño total de la tabla es de 156 entradas; es decir, 156 colores. La tabla después fue

optimizada para remover colores redundantes mediante la combinación de todos los colores de la tabla en conjuntos de 4, y la evaluación de inclusión en estos conjuntos de todos los demás colores en la tabla, como se muestra en la ecuación 36. quedando al final con 60 colores no repetidos, en el apéndice I se muestra la tabla resultante de los colores.

Con la imagen segmentada de la placa, a cada uno de los pixels de la imagen se la signa el color con la distancia euclidiana (que se mostró anteriormente en la ecuación 12) más corta.

Una vez que se encuentra el valor más cercano en la tabla, se asigna al píxel la misma etiqueta que tiene la celda correspondiente.; de esta manera se genera una imagen con las 4 categorías (colores) mencionadas anteriormente. La figura 28 muestra una imagen con los colores originales, y su resultante después de ser segmentada. En ella, el color negro representa el color de fondo de la placa, el blanco el color de frente (letras); el gris obscuro el color frecuentemente de fondo y el rojo el color frecuentemente de letras.

***Figura 28**: Segmentación de la placa por color.*
a) Imagen original b)Imagen segmentada

Las regiones “principalmente de letras” y “principalmente de fondo” son consideradas

ambiguas, ya que en algunos casos pertenecen a un conjunto y en otros a otro.

Con el fin de asignar las regiones ambiguas al frente o al fondo (ya que, como se mencionó, la imagen debe contener únicamente esas dos regiones), se presentan 3 casos, con las combinaciones de las categorías, las cuales son evaluadas por una función heurística para determinar cual aparenta ser mejor.

Las casos generados con las combinaciones son los siguientes:

- **Caso 1**. Se asignan las regiones "Letras" y "Principalmente letras" al frente y lo demás se vuelve fondo.
- **Caso 2**. Se deja la región "Fondo" como fondo y todo lo demás se vuelve frente.
- **Caso 3**. Se generan combinaciones de manera aleatoria-probabilística.

Las combinaciones aleatorias-probabilísticas fueron generadas asignando probabilidades ascendentes a cada una de las etiquetas: fondo, principalmente fondo, principalmente letras, letras respectivamente de convertirse en frente. Esto es, a cada uno de los objetos se les asignó una probabilidad de volverse frente dependiendo de su clase. Las probabilidades se muestran en la tabla 2.

***Tabla 2**: Probabilidades de que los objetos se vuelvan frente*

Clase	***Probabilidad de volverse frente***
Fondo	0.05
Principalmente fondo	0.25
Principalmente frente	0.60
Frente	0.95

Cabe mencionar que en el caso de objetos de tipo "fondo", únicamente se permitió la asignación a la clase "frente" si estaban rodeados completamente por objetos de otra clase; de lo contrario abarcarían la placa completa.

La figura muestra al número 0 segmentado con las 4 clases mostradas en la imagen en negro, gris, gris claro y blanco para fondo, principalmente fondo, principalmente letras y letras

respectivamente; y 3 combinaciones generadas probabilísticamente.

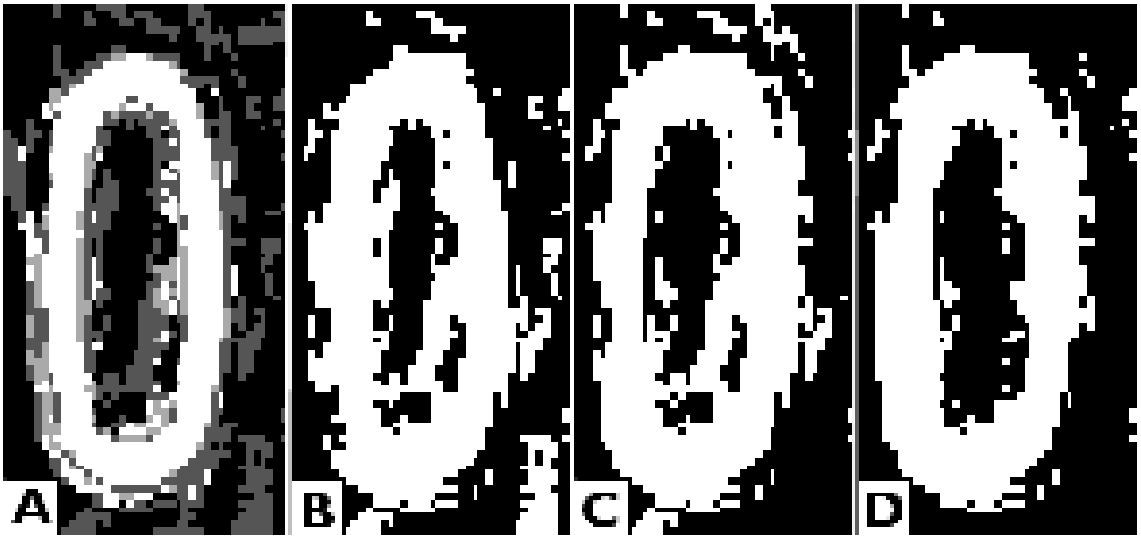

***Figura 29**: Combinaciones de las clases. a)imagen original a,c,d) combinaciones*

Se analizó cada región contigua en las imágenes de los caracteres y, con la probabilidad de su etiqueta, se decidió si se debe asignar al frente o al fondo.

La combinación con el mejor valor heurístico es elegida como representante del caracter.

La función heurística evalúa las siguientes características de una combinación para determinar cuál es mejor que cuál:

- Rangos mínimos y máximos de altura del caracter relativa a la placa. El mínimo de altura del caracter es el 30% de la altura de la placa; el máximo es el 60%.

- Rangos mínimos y máximos de ancho del caracter relativo a la placa. El ancho mínimo del caracter con respecto al ancho de la placa es 8.5% y el máximo 11%.

- Rangos mínimos y máximos de razón de superficie del caracter entre la superficie total de la placa (incluyendo al mismo caracter y otros). El mínimo es 0.005% de la superficie de la placa, el máximo es 0.038%.

- Además de verificar que la razón de superficie se encuentre dentro del rango, se favorece aún más a los caracteres cuya superficie se encuentre cerca del centro del rango. Esto es, se favorece a los caracteres cuya superficie se encuentra dentro de 0.014% y 0.0203

- Cantidad normalizada de píxeles activados en las diagonales del rectángulo que cubre al caracter, como se muestra en los píxeles de color rojo claro en la imagen 30. La cantidad en esa imagen es 0.5.

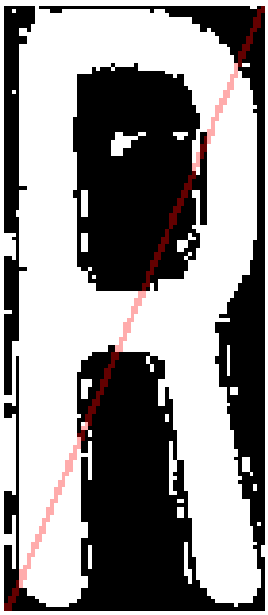

***Figura 30**: Medida de píxeles en la diagonal inferior izquierda de un caracter*

- Cantidad normalizada de píxeles cerca de las esquinas de la imagen. Las esquinas se definen como un cuadrado en la imagen con una longitud de lado de de un octavo del ancho del rectángulo que rodea al caracter tocando sus bordes. La figura 31 muestra un ejemplo de esquinas con la letra U.

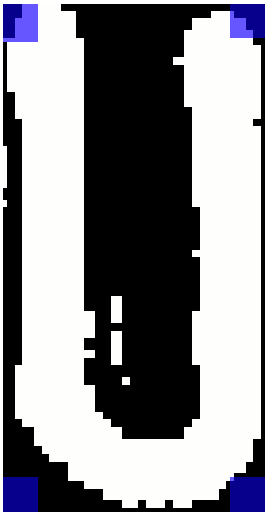

***Figura 31**: Medida de píxeles en las esquinas de un caracter*

- Número de objetos pequeños (menores a 3 píxeles) en la imagen, en cualquier dirección. Mientras menos de éstos haya, el índice mejora.

- Desfase horizontal del objeto con respecto a su centro. Se realiza buscando donde la extensión horizontal a ambos lados del centro del rectángulo que envuelve al objeto, como se muestra en la figura 32

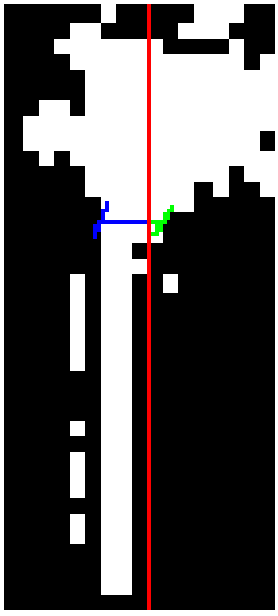

***Figura 32**: Medida de desface horizontal del caracter a partir del centro*

La figura 33 muestra algunas combinaciones generadas para la letra 'M'

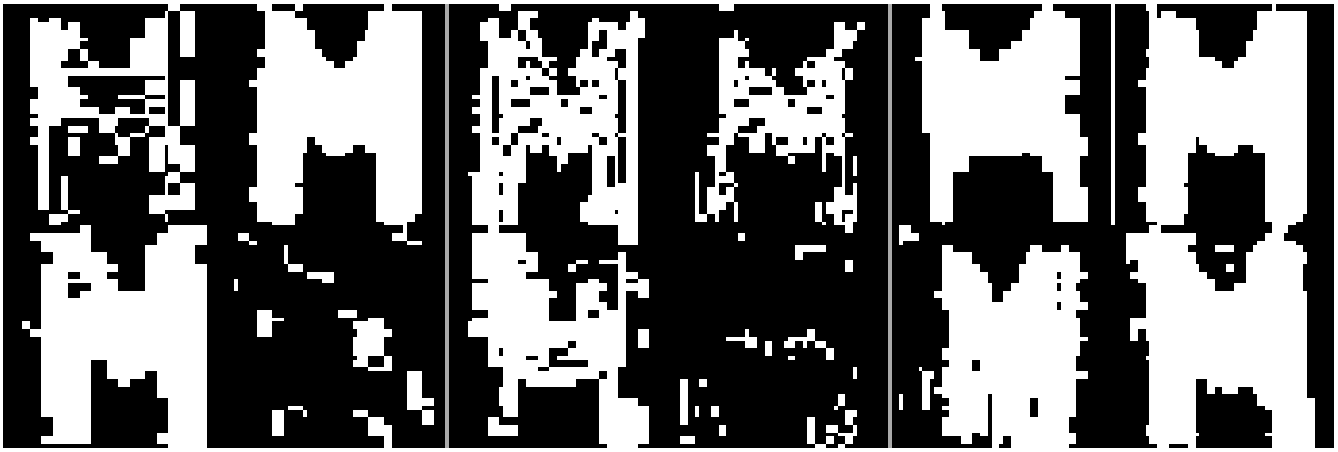

***Figura 33**: Pruebas de reconstrucción de la letra M.*

6.3. Extracción de Características.

Una vez segmentados los caracteres, se extrajo un conjunto de 20 características para que posteriormente sean clasificadas.

Cada caracter es primero normalizado a un tamaño fijo y su razón de altura contra ancho ajustada con *Continuous aspect ratio mapping*, lo cual, ha sido demostrado, da mejores resultados que ajustar el tamaño de los objetos proporcionalmente [LIN03]. La fórmula para ajustar el ratio de altura y ancho de la imagen usando c*ontinuous aspect ratio mapping* se muestra en la ecuación 5-1, donde "r" es la razón original de la imagen y " r' " la razón resultante.

$$r' = \sqrt{\sin(\frac{\pi}{2} r)} \tag{39}$$

Posteriormente, se utiliza una transformada de Hough reducida para identificar líneas horizontales, verticales, a 45 y a 135 grados. Las tres líneas que hayan conseguido la mayoría de votos

son seleccionadas y ordenadas para asegurar que siempre se encuentren en el mismo orden, a partir del resultado se consideran 3 características: origen, ángulo y número de votos como características

Una característica más es el cuadrado de la suma de la distancia Euclidiana, que se muestra en la ecuación (40), de cada píxel con información con respecto al centroide del objeto. Este valor también se normaliza, pero en vez de que el valor máximo (uno) después de normalizar corresponda al valor máximo posible de este índice (que sería el resultado de la suma de la distancia Euclidiana al cuadrado de cada píxel al centro); se fijó un porcentaje máximo de píxeles que pueden estar activados, es decir, que tienen un valor de 1; con esto se determinó un índice máximo posible para caracteres válidos y a éste valor se asignó el 1 después de la normalización. En caso de que el valor obtenido fuera mayor a éste, se corta el valor normalizado en uno.

Finalmente, la imagen es dividida en 6 regiones (dos regiones horizontales y tres verticales). Entonces, se obtiene el cuadrado de la distancia Euclidiana de cada píxel activado dentro de las regiones hacia ciertos puntos relevantes, ubicados al centro y bordes de éstas. Las distancias de todos los píxeles de una región a un punto en particular se suman y normalizan, y se regresan como características para la identificación del carácter. Los puntos se muestran el la figura 34

En resumen, las 20 características son:

1.- Ángulo de la línea a 0, 45, 90 o 135 grados con mayor número de votos de acuerdo a la transformada de Hough.

2.- Coordenada al origen de la línea a 0, 45, 90 o 135 grados con mayor número de votos de acuerdo a la transformada de Hough (cruce con el eje *y*, excepto para líneas verticales, en ese caso es con el eje “x”).

3.- Número de votos de la línea a 0, 45, 90 o 135 grados con mayor número de votos de acuerdo a la transformada de Hough.

4.- Ángulo de la línea a 0, 45, 90 o 135 grados con el segundo mayor número de votos de acuerdo a la transformada de Hough.

5.- Coordenada al origen de la línea a 0, 45, 90 o 135 grados con el segundo mayor número de votos de acuerdo a la transformada de Hough.

6.- Número de votos de la línea a 0, 45, 90 o 135 grados con el segundo mayor número de votos

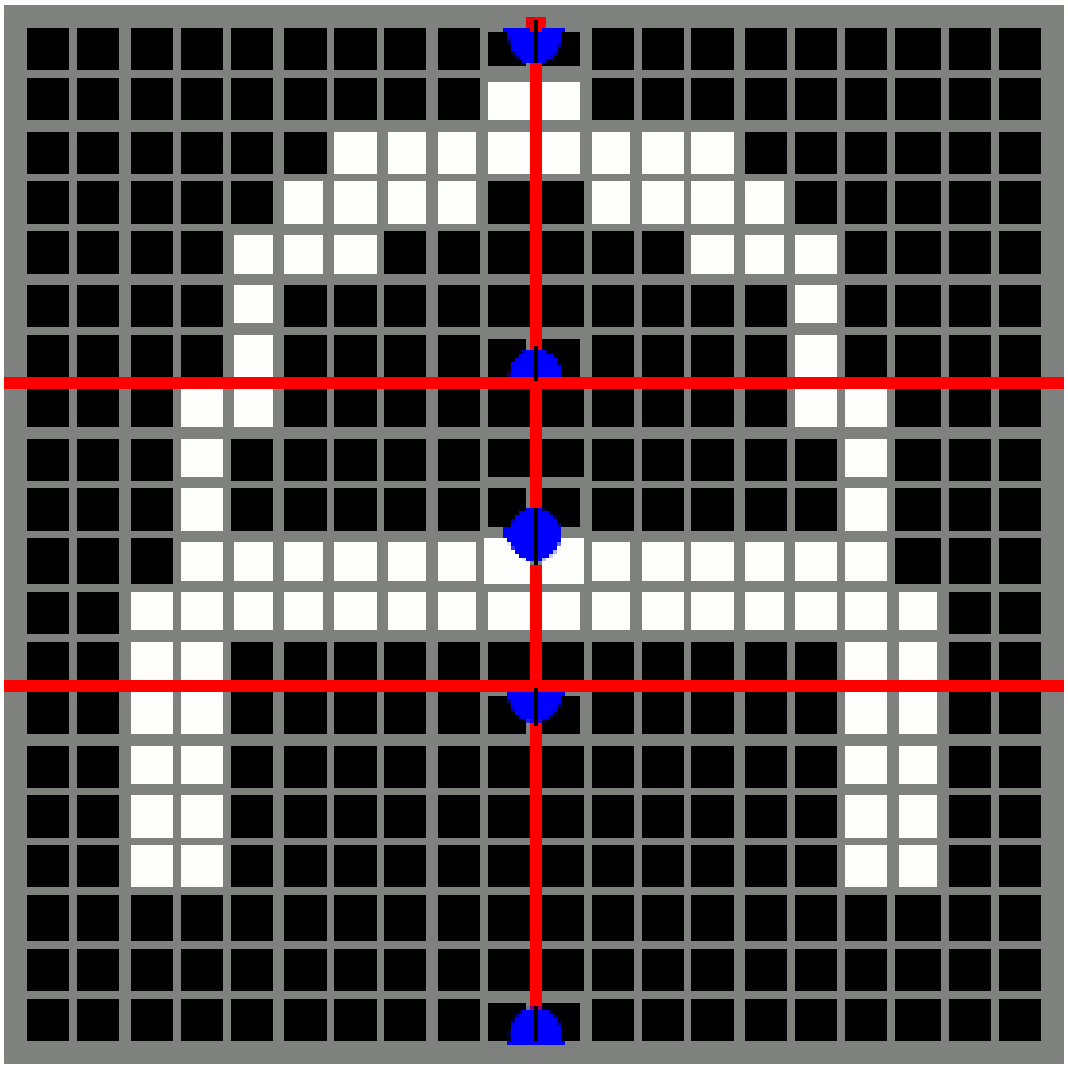

***Figura 34**: Puntos desde los que se mide la distancia como característica*

de acuerdo a la transformada de Hough.

7.- Ángulo de la línea a 0, 45, 90 o 135 grados con el tercer mayor número de votos de acuerdo a la transformada de Hough.

8.- Coordenada al origen de la línea a 0, 45, 90 o 135 grados con el tercer mayor número de votos de acuerdo a la transformada de Hough.

9.- Número de votos de la línea a 0, 45, 90 o 135 grados con el tercer mayor número de votos de acuerdo a la transformada de Hough.

10.- Cuadrado de la distancia euclidiana de cada píxel al centro.

Dividiendo la imagen por la mitad verticalmente, y en tercios horizontalmente, se forman las

regiones A, B, C, D, E, F de derecha a izquierda y arriba a abajo, como se muestra en la figura 35

A	B
C	D
E	F

***Figura 35**: División de la imagen para calcular pesos*

11.- Distancia de la esquina superior derecha de la región A a los píxeles de la región A.

12.- Distancia de la esquina inferior derecha de la región A a los píxeles de la región A.

13.- Distancia de la esquina superior izquierda de la región B a los píxeles de la región B.

14.- Distancia de la esquina inferior izquierda de la región B a los píxeles de la región B.

15.- Distancia de la media derecha de la región C a los píxeles de la región C.

16.- Distancia de la media izquierda de la región D a los píxeles de la región D.

17.- Distancia de la esquina superior derecha de la región E a los píxeles de la región E.

18.- Distancia de la esquina inferior derecha de la región E a los píxeles de la región E.

19.- Distancia de la esquina superior izquierda de la región F a los píxeles de la región F.

20.- Distancia de la esquina inferior izquierda de la región F a los píxeles de la región F.

La figura 36 contiene un ejemplo de las características extraídas a partir de una imagen de la letra “T”. La figura 37 incluye un ejemplo de las características extraídas del número 7.

0.666666	0.51428
0.942857	0.0
0.057142	0.571428
0.666666	0.571428
0.371428	0.261982
0.058219	0.082984
0.146835	0.290452
0.252899	0.402399
0.082005	0.094673
0.089181	0.150572

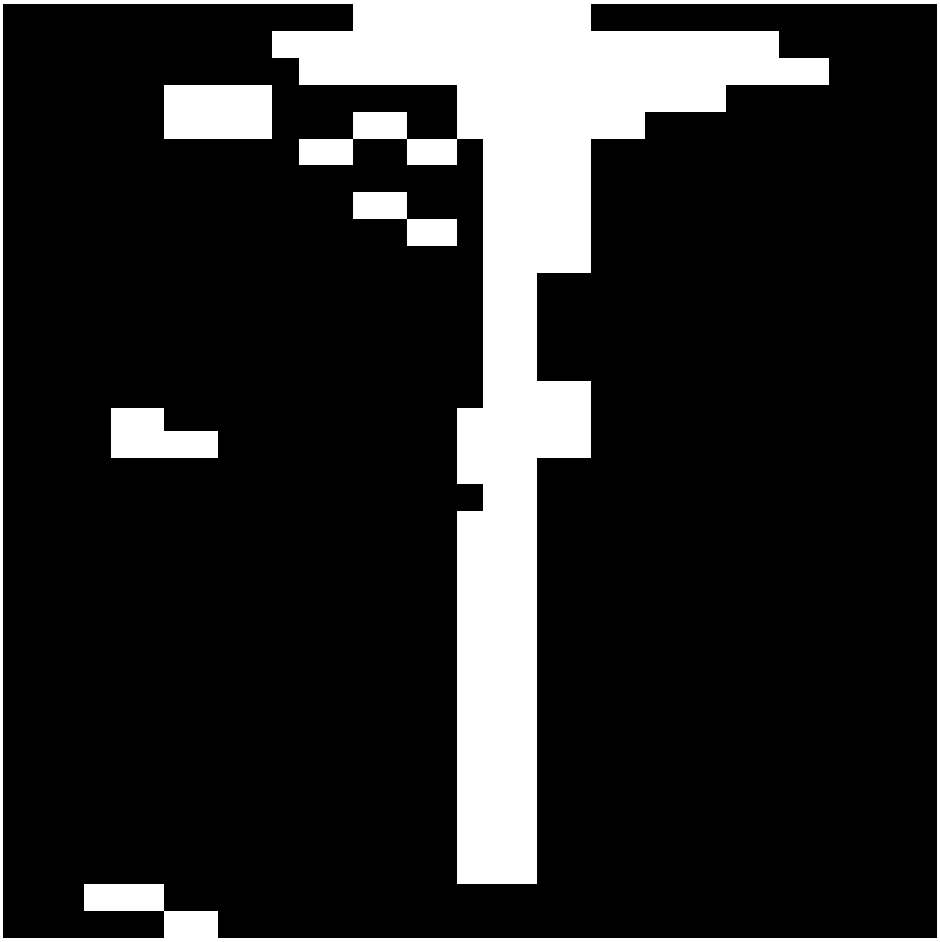

***Figura 36**: Extracción de características de letra T*

0.666666	0.4
0.742857	0.0
0.028571	0.685714
0.0	0.114285
0.657142	0.476662
0.185691	0.148090
0.262566	0.597427
0.418576	0.629231
0.452954	0.0010
0.528747	0.015816

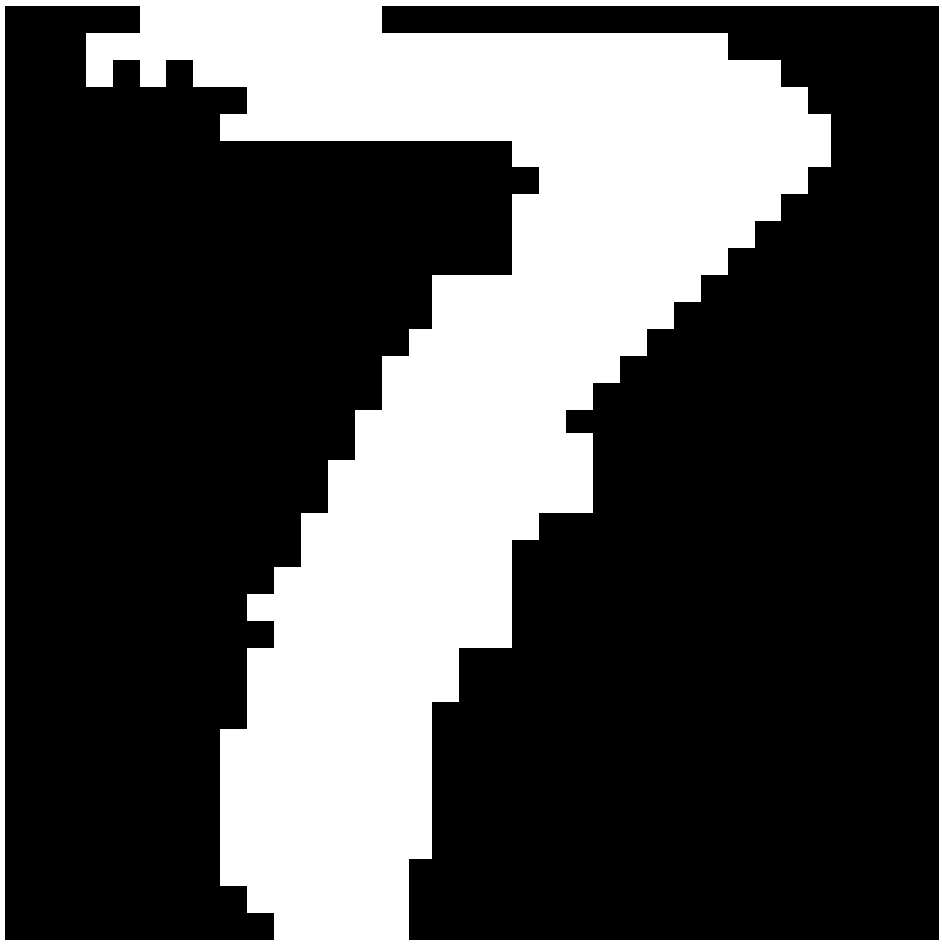

***Figura 37**: Extracción de características del número 7*

6.2.3. Identificación de Caracteres.

Debido a que las placas de automóviles particulares de la Ciudad de México siempre constan de tres números, seguidos por un guión y éste seguido por tres letras, se asumió que los tres primeros objetos grandes encontrados son números, y los últimos tres son letras. El guión es considerablemente más pequeño que los demás caracteres, entonces el tamaño sólo es suficiente para descartarlo.

Gracias a la suposición anterior, fue posible utilizar dos clasificadores diferentes; uno para letras y otro para números y, de ésta manera, obtener mejores resultados para ambos, ya que no le es posible al clasificador confundir letras y números.

6.4.1. Reconocimiento de Números.

Para el reconocimiento de los números se utilizó una Perceptrón multicapa entrenada con Back Propagation con la siguiente arquitectura:

- Dos capas ocultas
- 30 nodos en cada una de las capas ocultas
- 20 nodos de entrada (correspondientes a las 20 características extraídas)
- 10 nodos de salida

Se utilizaron 510 muestras seleccionadas aleatoriamente. La red se entrenó hasta obtener un error global de 0.03

6.4.2. Reconocimiento de Letras.

Para el reconocimiento de letras se utilizó una red neuronal con las siguientes características:

- Una capa oculta
- 100 nodos en la capa oculta
- 20 nodos de entrada
- 23 nodos de salida

Se utilizaron 516 muestras seleccionadas aleatoriamente del conjunto de entrenamiento para entrenar a la red; misma que se entrenó hasta que el error global fue menor a 0.01

7. Resultados.

7.1. Localización de Placas.

Mediante el método de localización de placas utilizando bordes verticales, el sistema encontró 90% de las placas en el conjunto de entrenamiento, sin embargo el desempeño en el conjunto de prueba cayó a 35%. Algunas de las placas no localizadas presentaban problemas por ser demasiado pequeñas u otros defectos.

El método de localización usando segmentación por regiones de decisión de color usando una red neuronal dio resultados del 80% en ambos conjuntos de imágenes. Reconoció 98.5% de las imágenes de coches con portaplacas. La mayor parte del porcentaje de error se produjo en vehículos de colores blanco, plata y, especialmente, dorados.

El método de localización con segmentación por regiones de decisión de color usando un algoritmo genético, obtuvo un porcentaje de éxito del 95% para el conjunto de entrenamiento, y 92% para un conjunto de prueba tomado con condiciones de iluminación similares.

7.2. Reconocimiento de Caracteres.

En el reconocimiento de números, el sistema obtuvo resultados de 98%, con varios números teniendo éxitos individuales del 100%. Los números más problemáticos fueron el 6 y el 9; que arrojaron la mayor cantidad de resultados inválidos, en los que la red informa que no pudo identificarlos. No hubo números mal reconocidos; es decir, números que hayan sido positivamente identificados como otro número. En reconocimiento de letras el éxito del sistema fue de 97.5%.

El reconocimiento de las placas tomadas en movimiento no mostró diferencia en desempeño con respecto a las que se tomaron sin movimiento.

8. Conclusiones.

Con un porcentaje de éxito de al menos 92% en la etapa de detección y localización, y 98% para la etapa de reconocimiento de caracteres, el sistema cumple con el objetivo de detectar, localizar y leer placas de la Ciudad de México.

El método de generación de regiones de decisión de color utilizando algoritmos genéticos presentó mejores resultados que el de redes neuronales, sin embargo, eso se debe en gran parte a que éste es más general que el otro. Sería posible, como trabajo posterior, generar un conjunto de regiones para segmentar colores con diferentes tipos de iluminación al igual que un sistema que discierna qué regiones funciona para una iluminación dada.

No era la intención del trabajo comparar el desempeño de las redes neuronales contra los sólidos tridimensionales. La especialización de uno y generalización del otro hacen que no sean comparables individualmente las redes contra los sólidos en este caso. Sólo se pueden comparar los métodos completos.

No se determinó con precisión la razón por la que el método basado en bordes verticales tuvo un desempeño tan pobre. Se teorizó que se debe al bajo contraste que hay entre las placas y los vehículos, debido en gran parte a que aquéllas no tienen bordes y no son de un color ni muy brillante ni muy obscuro (a diferencia, por ejemplo, de las placas hipotéticas propuestas por Zimic [ZIM97]). Además, el método es muy sensible a deformaciones en la imagen. Las placas deformes o los portaplacas rotos, aunque sea en lo más mínimo previenen a este método localizar los bordes verticales que requiere.

El método basado en redes neuronales presentó resultados aceptables, aunque bajos. Es muy robusto porque puede localizar placas bajo una variedad de condiciones de iluminación y es muy poco sensible a deformaciones. Prueba de su robustez es que obtuvo el mismo porcentaje de éxito en el conjunto de prueba que en el de entrenamiento. La razón por la que su porcentaje de éxitos no es mayor es que existen demasiados colores en el medio ambiente y en particular en los coches que son identificados como color de placa pese a que quizás bajo las condiciones de iluminación en un momento dado no correspondan con el color de las placas. Si estos colores se encuentran en las áreas contiguas a la placa los límites de ésta se desbordan y resulta imposible reconocerla como una placa.

El método que obtuvo mejores resultados fue el basado en algoritmos genéticos. Este método se

especializó en un sólo tipo de iluminación y las imágenes que se le alimentaron, tanto de entrenamiento como de prueba, cumplían con este muy estrecho tipo de iluminación (pese a que fueron tomadas en diferentes días).

Con el fin de lograr una comparación entre el método basado en redes neuronales y el basado en algoritmos genéticos sería necesario compararlos en igualdad de circunstancias. Podría entrenarse la red neuronal únicamente con las imágenes que se entrenó el método basado en algoritmos genéticos; podría hacerse lo opuesto y entrenar el método basado en geometría bajo todas las condiciones de iluminación.

Una mejor solución sería, como trabajo posterior, la generación de diferentes regiones de decisión para diferentes condiciones de iluminación con el algoritmo genético y, posteriormente, la elaboración de un algoritmo capaz de detectar las condiciones de iluminación en las que se capturó alguna imagen y a partir de ésto la selección automática del conjunto de regiones de decisiones que generarán una mejor segmentación de una imagen.

La generación de una sola región de decisión de color utilizando algoritmos genéticos tomó un gran número de días debido a que no se hicieron optimizaciones en la función heurística y a ésta le toma demasiado tiempo evaluar a cada individuo, además de que se usaron varias poblaciones iniciales e intermedias con el fin de explorar más ampliamente el espacio de búsqueda. Por esta razón, no se generó más que un juego de regiones de decisión de color y, por lo mismo, no se pudieron realizar más pruebas para lograr una comparación en igualdad de circunstancias con el método basado en redes neuronales.

El algoritmo de *umbralización* que se propuso en el documento dio buenos resultados con la mayoría de las imágenes de bordes con las que se probó. No es posible obtener un índice numérico para medir su desempeño de manera cuantitativa; sin embargo, de manera cualitativa se apreciaron las imágenes y se consideraron factores tales como qué tanto ruido fue convertido en información de bordes y, más importante, si los bordes de la placa se preservaron, de estar presentes (hay ocasiones en las que el borde no se aprecia siquiera en la imagen antes de *binarizar*).

El *kernel* de Sobel modificado segmentó 10% más placas que el kernel de Sobel sin modificar. Además, en vista de que el método basado en bordes verticales depende en que el grosor de los bordes sea mayor a un píxel, el desempeño del *kernel* modificado mejora aún más al llegar a la parte de conteo de objetos.

El algoritmo de reducción de ruido usando una ventana no generó resultados muy buenos en términos de qué tanto ruido remueve. Las imágenes aún contienen ruido después de que se les aplica tal algoritmo. Sin embargo, no era el objetivo del mismo eliminar el ruido de la imagen, sólo reducirlo con el fin de acelerar los procesos posteriores a esa etapa.

9. Referencias.

[AND88] Anderson, James A; Wisniewski, Edward J; Viscuso, Susan R. Software for Neural networks. ACM Press. 1988

[AND95] Andrew, Alex M. The decade of the Brain, some comments. Emerald group publishing limited. 1995

[BAB03] Babic, Zdenka; Mandic, Danilo. "An Efficient Noise Removal and Edge Preserving Convolution Filter". Telecommunications in Modern Satellite, Cable and Broadcasting Service, 6th International Conference on. 2003

[BAB90] Babaguchi, Naboru; Kise, Koichi; Tezuka, Yoshikazu. "Connectionist Model Binarization". Pattern Recognition, 1990. Proceedings, 10th International Conference on. 1990.

[BRU98] Brugge M, Stevens J, Nijhuis J, Spaanenburg J. "License plate recognition using DTCNNs". Fifth IEEE International Workshop on Cellular Neural Networks and their applications" 1998

[CHA04] Chang S, Chen L, Chung Y, Chen S. "Automatic License Plate Recognition". IEEE Transactions on intelligent transportations systems. 2004

[COL06] Coltuc, Dinu; Bolon, Philippe; Chassery, Jean-Marc. "Exact histogram Specification". 2006

[COO87] Cooper, A; Kahari, W; Such, R. "Image processing for electronic document storage". Computers and Digital Techniques, IEEE Proceedings. 1987

[DEJ89] De Jong, Kenneth A.; Spears, William M. "Using Genetic Algorithms to Solve NP-Complete Problems". Proceedings of the third international conference on genetic algorithms. 1989.

[DUD72] Duda, Richard; Hart, Peter. "Use of Hough transformation to detect lines and curves in pictures". Communications of the ACM Volume 15 Issue 1. 1972.

[ERE94] Erenshteyn, Roman; Foulds, Richard; Galuska, Scott. Is designing a neural network application an art or a science?. ACM Press. New York, 1994

[HUN75] Hunt, B.R. “Digital Image Processing”. Proceedings of IEEE. 1975.

[KIM01] Kim Joung-Youn; Kim Lee-Sup; Hwang, Seung-Ho. “An Advanced Contrast Enhancement Using Partially Overlapped Sub-Block Histogram Equalization”. Circuits and systems for Video Technology. 2001

[KIM97] Kim, Yeong-Taeg. “Quantized Bi-histogram equalization”. Acoustics, Speech and Signal Processing. 1997

[KNI90] Knight, Kevin. Connectionist ideas and algorithms. ACM Press. Pittsburg, PA. 1990

[LEE96] Lee, Kangsun; Fishwick, Paul A. Dynamic model abstraction. ACM Press. New York. 1996.

[LIM98] Lim B, Yeo W, Tan K, Teo C. “A novel DSP based real-time character classification and recognition algorithm for car plate detection and recognition”. Proceedings of ICSP, 1998.

[LIN03] Lin-Liu Ch, Nakashima K, Fujisawa H. “Handwritten digit recognition: benchmarking of state of the art techniques”. Pattern recognition. 2003

[NAS94] Nascimento Jr, Cairo L. Neural Networks in Control and Optimization. Tesis doctoral de la Universidad de Manchester. Manchester. 1994.

[NIJ95] Nijhuis J, Brugge M, Helmholt K, Plium J, Spaanenburg L. “Car license plate recognition with neural networks and fuzzy logic”. Proceedings, IEEE international conference on neural networks. 1995

[PAN05] Pan X, Ye X, Zhang S. “A hybrid method for robust car plate character recognition”. Engineering applications of artificial intelligence. 2005.

[PAR99] Park S H, Kim K I, Jung K, Kim H J. “Locating car license plates using Neural Networks”. Electronics Letters Online. 1999

[PER94] Perkins, Simon; Walker, Ashley; Wolfart, Erik. “HyperMedia Image Processing Reference”. http://www.cee.hw.ac.uk/hipr/html/hipr_top.html . 1994.

[REI90] Reichenbach, Stephen E.; Park, Stephen K.; Alter-Gartenberg, Rachel. “Optimal Small Kernels for Edge Detection”. Pattern Recognition, 1990. Proceedings. 10th International Conference on. 1990

[RUD97] Rudnick, Elizabeth M.; Patel, Janak H.; Greenstein, Gary S.; Niermann Thomas M. “Genetic Algorithm Framework for Test Generation”. Computer-Aided Design for Integrated Circuits and Systems, IEEE Transactions on. 1997

[RYU94] Ryung E, Pyeoung K, Joon H. “Automatic recognition of a car license plate using color image processing”. IEEE International Conference, 1994.

[SAL99] Salgado L, Menéndez J, Rendón E, García N. “Automatic car plate detection and recognition through intelligent vision engineering”. Proceedings 33rd annual 1999 international Carnahan conference on Security Technology. 1999

[SCH89] Schalkoff, Robert J. “Digital Image Processing and Computer Vision”. Wiley. 1989. (p. 169-172)

[SCH97] Schwefel, Hans-Paul. “Special Track on Computational Intelligence – Genetic Algorithms”. EUROMICRO 97. 'New Frontiers of Information Technology'., Proceedings of the 23rd EUROMICRO Conference. 1997

[SIR99] Sirithinaphong T, Chamongthai K. “The recognition of car license plate for automatic parking system”. Fifth international symposium on signal processing and its applications. 1999.

[SON89].Sondak, N.E, Neural Networks and Artificial Intelligence. ACM Press. Kentucky. 1989

[TOU74] Tou, J.C.; Gonzalez, R.C. “Pattern Recognition Principles”. Addison Wesley Pusblishing Co. 1974. (p. 158-179, 173-181)

[TRI95] Trier, Oivind Due; Jain, Anil K. “Goal-Directed Evaluation of Binarization Methods”. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1995.

[WHI94] Whitley, Darrel. “A Genetic Algorithm Tutorial”. Statistics and Computing. 1994

[ZHA06] Zhao, Mansuo. “Image Thresholding Technique Based on Fuzzy Partition and Entropy Maximization”. University of Sydney. School of Electrical and Information Engineering. 2006.

[ZHE04] Zheng D, Zhao Y, Wang J. “An efficient method on license plate location”. Pattern recognition letters. 2004

[ZIM97] Zimic N, Fickzo J, Mraz M, Virant J. “The fuzzy logic approach to the car number

plate locating problem". Intelligent information systems, proceedings. 1997

[ZIO98] Ziou, Djemel; Tabbone, Salvatore. "Edge Detection Techniques – An Overview". Technical report, No. 195. Dept Math & Informatique, Universit de Scherbrooke. 1998

[ZUN00] Zunino R, Rovetta. "Vector quantization for license plate location and image coding". IEEE Transactions on industrial electronics. 2000

Glosario

- Redes neuronales

  Sistemas computacionales que emulan el funcionamiento biológico del cerebro.

- Adaline

  Red Linear Adaptativa.

- LMS

  Medias mínimas cuadradas. Algoritmo de entrenamiento de las redes Adaline

- Neuronas Ocultas

  Neuronas en una red neuronal multicapa que no se encuentran en la capa de entrada ni en la de salida.

- Función Escalón

  Función que regresa 1 si el valor de entrada se encuentra por encima de cierto umbral o -1 de lo contrario

- Función Umbral

  Ver función Escalón

- Función XOR

  Función de lógica binaria con dos entradas y una salida. Regresa 1 si las dos entradas son diferentes y 0 si las dos entradas son iguales, independientemente de su valor.

- Regla Delta

  Función para entrenar redes neuronales de tipo Perceptrón. El peso de una liga es igual al peso anterior mas/menos uno dependiendo de si la salida asociada es correcta o incorrecta.

- Función Sigmoide

  Función de activación donde la salida es igual a 1 sobre 1 más la constante e elevada al valor de la entrada.

- Función *Piecewise Linear*

Función de activación donde se regresa un valor constante si la entrada está por debajo de un cierto rango; otro valor constante si está por encima, y una interpolación lineal de los dos valores anteriores si la entrada se encuentra dentro del rango.

- Algoritmo de mejora local de contraste

Algoritmo propuesto en [ZHE04] para mejorar el contraste únicamente de las zonas que tienen contraste pobre en una imagen.

- Algoritmo genético

Algoritmo evolutivo de optimización que se basa en en cruce de individuos a lo largo de un número de generaciones.

- Función de aptitud

Función para evaluar el *fitness* de un individuo (qué tan bueno es) dentro de un algoritmo genético.

- Cadena binaria

Secuencia de unos y ceros que representa un un número o valor.

- Histograma

Conjunto de pares índice contra valor que representa la distribución de los valores de intensidad en una imagen en escala de grises. Los índices representan los posibles valores de intensidad de cada píxel (por ejemplo 0-255 para una imagen con 256 valores de gris); los valores son el número de píxeles en la imagen que tienen el valor en particular de intensidad al que corresponde su índice.

- Transformada de Hough

Algoritmo utilizado para encontrar formas parametrizadas en una imagen.

- *Kernel* de Convolución

Matriz que se desliza sobre una imagen para aplicar filtros de procesamiento digital a las mismas. Típicamente el valor del píxel central de una ventana es la combinación lineal de los píxeles cubiertos por la ventana multiplicados por el valor en las celdas de la matriz correspondientes a cada

uno de estos.

- Operador de Sobel

Kernel de Convolución utilizado para encontrar bordes en una imagen. Calcula los gradientes horizontales y verticales de la imagen, o bien, la intensidad y dirección del gradiente bidireccional de la imagen.

- Distancia Euclidiana

Medida de distancia en la cual, al revolucionar un punto al rededor del centro manteniendo la distancia fija, dibuja un círculo. Para dos dimensiones $z_2 = z_1$, por lo tanto $z_2 - z_1 = 0$

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{40}$$

- Espacio RGB

Sistema de coordenadas cartesiano tridimensional donde cada eje representa la intensidad de los colores rojo, verde y azul.

- Conexidad cuatro

Cuando se usa un esquema de conexidad 4 conexo, se dice que dos píxeles son vecinos si su coordenada "x" es igual y su coordenada "y" varía solo en uno; o viceversa, como se muestra en los cuadros grises de la la figura 38.

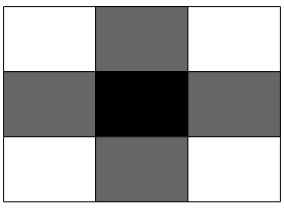

***Figura 38**: Píxeles vecinos bajo conexidad 4*

- Conexidad ocho

Bajo el esquema de conexidad 8 conexo, dos píxeles son vecinos si su coordenada "x" es igual o a lo más diferente en uno a su coordenada "y"; y si su coordenada "y" es igual o a lo más varía en uno a su coordenada "x", como se muestra en los píxeles grises de la figura 39.

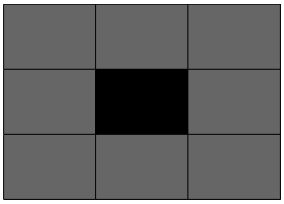

***Figura 39**: Píxeles vecinos bajo conexidad 8*

Apéndice I. Tabla de colores de placa

Tabla 3: Clases de colores de placa

Rojo	Verde	Azul	Clase
67	49	21	Color de letra
72	68	51	Color de letra
75	77	74	Color de letra
78	51	0	Color de letra
80	67	42	Color de letra
91	81	65	Color de letra
95	81	55	Color de letra
107	97	63	Color de letra
118	98	49	Color de letra
121	100	75	Color de letra
130	110	67	Color de letra
145	127	94	Color de letra
174	171	138	Color de letra
192	177	109	Color de letra
199	161	81	Color de letra
18	17	7	Color principalmente de letra
26	29	17	Color principalmente de letra
33	44	35	Color principalmente de letra
46	43	32	Color principalmente de letra
54	44	24	Color principalmente de letra
83	77	33	Color principalmente de letra
111	106	88	Color principalmente de letra
171	170	161	Color principalmente de letra
96	73	33	Color principalmente de fondo
105	75	16	Color principalmente de fondo
140	130	118	Color principalmente de fondo
142	104	27	Color principalmente de fondo
169	132	38	Color principalmente de fondo
215	165	42	Color principalmente de fondo
216	189	230	Color principalmente de fondo
248	252	215	Color principalmente de fondo
249	222	167	Color principalmente de fondo
249	244	182	Color principalmente de fondo
250	225	133	Color principalmente de fondo
171	162	112	Color de fondo
181	159	113	Color de fondo
183	158	107	Color de fondo
186	158	89	Color de fondo

186	159	96	Color de fondo
188	178	127	Color de fondo
190	156	95	Color de fondo
192	156	114	Color de fondo
198	174	128	Color de fondo
202	169	138	Color de fondo
202	174	110	Color de fondo
205	165	71	Color de fondo
208	172	129	Color de fondo
208	195	157	Color de fondo
215	165	42	Color de fondo
215	191	144	Color de fondo
216	189	230	Color de fondo
222	193	153	Color de fondo
230	192	123	Color de fondo
230	233	197	Color de fondo
233	197	100	Color de fondo
233	229	214	Color de fondo
247	248	205	Color de fondo
249	244	182	Color de fondo
250	231	154	Color de fondo