RECONOCIMIENTO DE PLACAS DE AUTOMÓVIL USANDO LA PLATAFORMA ECB_AT91.

LUIS MIGUEL ARIZA FLÓREZ.
SERGIO IVÁN MERCHÁN MEJÍA.

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA-FEBRERO 2009

RECONOCIMIENTO DE PLACAS DE AUTOMÓVIL USANDO LA PLATAFORMA ECB_AT91.

LUIS MIGUEL ARIZA FLÓREZ
SERGIO IVÁN MERCHÁN MEJÍA.

Trabajo de grado para optar al título de:
Ingeniero Electrónico

DIRECTOR:
Jorge Hernando Ramón Suárez, M.S.c.

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA-FEBRERO 2009

Dedicado a…

A Dios, porque siempre ha sido mi guía a lo largo de la vida.

A mis padres, Miguel Angel y Mary Luz por el apoyo, la confianza incondicional y quienes son mi fortaleza, mi fuente de protección y amor.

A mi hermano, porque a su manera me dio apoyo y animo en los momentos que sentí caer.

A todos mis compañeros por los buenos y malos momentos, por las tragedias y las alegrías, por los triunfos y las derrotas, en fin gracias.

A Sergio Merchán, por su empeño en la culminación de esta meta.

A la Universidad Industrial de Santander y sus profesores por brindarme su conocimiento.

Luis Miguel Ariza Flórez

Dedicado a…

A Dios, por ser mi fiel amigo y compañero incansable.

A mis padres, Nacho y Miriam, por su confianza, paciencia y comprensión durante todas y cada una de las etapas por las que ha transitado mi vida.

A mis hermanos, por su paciencia y comprensión, y por ser una inspiración más para continuar.

A mi familia, por su apoyo y motivación.

A mis amigos y compañeros, por esos grandes momentos que influyeron en mi formación personal y académica.

A la Universidad Industrial de Santander, quien por intermedio de sus docentes dio su mayor aporte a mi crecimiento intelectual.

Y finalmente a mi compañero de travesía, Miguel, por su amistad, dedicación y empeño por concluir con éxito este trabajo.

Sergio Merchán

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE ANEXOS

TITULO

RECONOCIMIENTO DE PLACAS DE AUTOMÓVIL USANDO LA PLATAFORMA ECB_AT91.[*]

AUTORES

LUIS MIGUEL ARIZA FLÓREZ
SERGIO IVÁN MERCHÁN MEJÍA.[†]

PALABRAS CLAVES: Automatización de procesos, ATR (reconocimiento automático de objetivos), LPR (license plate recognition), tratamiento digital de imágenes, lenguajes de programación, segmentación, filtrado, convolución.

DESCRIPCIÓN

En este documento se trata de describir un sistema de detección y reconocimiento de ciertas características presentes en una imagen, por medio de herramientas de análisis computacional que se aplican en la actualidad, haciendo parte el sistema ATR (de sus siglas en inglés Automatic Target Recognition, Reconocimiento Automático de Objetivos); cuyo objetivo es detectar, reconocer e identificar un objeto sin la ayuda del humano. Unas de las tantas aplicaciones que se encuentran dentro de estos sistemas, son los LPR (License Plate Recognition, Reconocimiento de placas de Automotores) los cuales localizan e identifican de manera automática las placas de vehículos automotores en imágenes fijas así como en videos; en la industria los sistemas LPR son empleados en parqueaderos, fronteras de algunos países, o para la identificación de autos robados.

El procedimiento luego del registro de la imagen, presenta la localización de la zona de interés (ROI); dicho proceso se inicia aplicando un cambio de espacio de color del espacio RGB a una modificación del espacio de Hering, con lo que se busca filtrar la mayor parte de los colores que no sean del color característico de las placas (en Colombia de color amarillo), siendo este el inicio de un proceso de segmentación cromática.

Aplicando seguidamente a la imagen, técnicas básicas en el procesamiento de imágenes, se segmentan los seis caracteres presentes en las placas de los vehículos. Para el proceso de reconocimiento de los caracteres, las técnicas mas utilizadas se basan en el campo de la inteligencia artificial: sistema de visión artificial y redes neuronales; pero debido a la capacidad de procesamiento de la plataforma en la cual se implementara nuestro código, se decidió implementar un método basado en el método de la correlación, modificándolo con el fin de requerir menos procesamiento a la hora de reconocer los caracteres.

[*] Trabajo de Grado

[†] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica, y de Telecomunicaciones. Director: Jorge Hernando Ramón Suárez, M.S.c.

TITLE

LICENSE PLATE RECOGNITION USING THE ECB_AT91[‡] PLATFORM.

AUTHORS

LUIS MIGUEL ARIZA FLOREZ
SERGIO IVÁN MEJÍA MERCHÁN.[§]

KEYWORDS: Process Automation, ATR (Automatic Target Recognition), LPR (license plate recognition), processing image digital, programming languages, segmentation, filtering, convolution.

DESCRIPTION

This document focuses on describing a detection and recognition system of some features presented in an image by computational analysis tools applied in nowadays, including the ATR System (Automatic Target Recognition); which objective is to detect, to recognize and to identify an object without the human's help. One of the many applications within this systems are the LPR (License Plate Recognition), the which locate and identify automatically the license plate of motorized vehicles in fixed images as in videos; In the industry the LPR systems are employed in parking lots, frontiers of some countries, or for identification of stolen cars.

The procedure after the image record, presents the localization of the region of interest (ROI), that process starts by applying a change of the color's space from RGB to a modification of the Hering space, looking forward to filtrate the most of the colors that doesn't belong to the characteristic color of license plates (In Colombia, that is yellow), being this, the start of a chromatic segmentation process.

Applying next, basic techniques in the image processing to segment the six characters presented in the license plates of vehicles. The Process to recognize them deals with applied techniques based on artificial intelligence field: artificial vision system and neuronal networks; but due to the capacity of the processing of the platform in which our code will run, it was decided of implement other method with less processing at recognizing the characters.

[‡] Degree Work

[§] Physics {Mechanics Engineering Faculty. Electrical, Electronic and Telecommunication School. Director: Jorge Hernando Ramón Suárez, M.S.c.

INTRODUCCIÓN

La aplicación del tratamiento de imágenes en sistemas de visión artificial es una herramienta que puede ser usada para la automatización de procesos que requieran la toma de decisiones basadas en características físicas de un objeto. Para esto se concibe un sistema de detección y reconocimiento de dichas características por medio de herramientas de análisis computacional.

Uno de estos ejemplos, que se aplican en la actualidad, son los sistemas **ATR** (de sus siglas en inglés Automatic Target Recognition, Reconocimiento Automático de Objetivos); cuya finalidad es detectar, reconocer e identificar un objeto sin la ayuda del humano.

Este tipo de sistemas tienen una gran variedad de aplicaciones en el mercado, como lo son identificación de objetivos con fines militares, hasta la interpretación de imágenes de placas de rayos X en medicina.

Unas de las aplicaciones que se encuentran dentro de estos sistemas, son los **LPR** (License Plate Recognition, Reconocimiento de placas de Automotores) cuyo objetivo es localizar e identificar de manera automática las placas de vehículos automotores de imágenes fijas o videos.

El registro y control del tráfico de automóviles, es de suma importancia en la entrada y salida en un parqueadero publico o privado, el cual se realiza diariamente y a cada minuto, esto dependiendo de la capacidad que tenga este. Para este tipo de problemas se utiliza como solución los LPR, para así suplir el personal, y realizar un control mas eficaz y rápido, como puede ser dar permisos a la entrada de un parqueadero privado, tal como una universidad, la entrada a profesores y personal autorizado, o en un parqueadero publico, guardar

información de la hora de entrada y salida del vehículo e imprimirla para que este al alcance del conductor.

Los sistemas LPR son un conjunto especial de componentes de hardware y software, con una señal de entrada gráfica como imágenes estáticas o secuencias de vídeo, donde se encuentra la placa para reconocer los caracteres de la misma. Una parte de hardware de los sistemas LPR consta de una cámara, procesador de imagen, la activación de la cámara, la comunicación y la unidad de almacenamiento.

El hardware de activación, controla directamente un sensor instalado en un carril. Cada vez que el sensor detecta un vehículo a una distancia de la cámara, se activa un mecanismo de reconocimiento.

El procesado de las imágenes instantáneas capturadas por la cámara, reconoce y devuelve un texto representación de la placa detectada.

En este documento se da una propuesta de implementar un algoritmo en una nueva plataforma basada en el sistema operativo Debian GNU/Linux, el cual tiene como propósito, tener un procesado de la imagen, simple y eficientemente, con un uso mínimo de un PC, con el fin de reducir los costos al implementar este tipo de sistemas , y demostrar que los estudiantes de la escuela están preparados para el cambio de tecnologías que están implementadas en la actualidad, y abrir el camino a la actualización de los sistemas de enseñanza empleados en las aulas de laboratorios en cualquiera de las asignaturas en las que se emplean instrumentos tecnológicos.

1. CONCEPTOS PRELIMINARES.

La segmentación de imágenes es un proceso basado en las características de los objetos presentes en la imagen, dichos objetos entonces se pueden clasificar según su color, textura, forma, etc.

Para mencionar algunos métodos de segmentación se debe dar una definición lo que es en si la segmentación, una segmentación completa de una imagen R, es un conjunto finito de regiones $R_1,\ldots,R_s$ tales que [5]:

$$R = \cup_{i=1}^{s} R_i \qquad R_i \cap R_j = \emptyset, \quad i \neq j \qquad (1)$$

Existe un gran número de técnicas empleadas para este objetivo, dentro de las mas comunes encontramos detección de discontinuidades, detección de fronteras, Umbralización, segmentación basada en regiones y segmentación morfológica con la transformada de Watersheds [2]. Teniendo en cuenta la clasificación de los objetos según la sensación de color y que esta nos permite diferenciar los objetos con mayor precisión, se incluye la segmentación por color como una técnica de segmentación de imágenes por regiones cromáticas [4] [9] [14].

Según el grado de interacción del usuario en el proceso de la segmentación, también se divide en manual, automática y semiautomática [3]:

- Manual: el usuario realiza todo el proceso de segmentación, gracias a una herramienta computacional.

- Automática: Un ordenador realiza según cierta programación todo el proceso de segmentación.
- Semiautomática: en algún momento del proceso de segmentación el usuario debe intervenir.

1.1. SEGMENTACIÓN POR REGIONES CROMÁTICAS

Se define como la separación de regiones de un mismo color, para entenderla mejor se debe tener claridad sobre lo que es el color y los espacios de color existentes.

1.1.1. El Color

La idea del color es básicamente simple, los colores que el ojo humano percibe tienen una gran dependencia de la naturaleza de la luz reflejada por el objeto y que es percibida e interpretada por las personas y animales a través de sus órganos de visión. Las ondas electromagnéticas reflejadas por los objetos son interpretadas como colores según las longitudes de onda correspondientes; si dicha luz no tiene color su único atributo es la intensidad, ahora para una luz con color, el espectro electromagnético de luz visible por el ojo humano va de 400 a 700nm [9].

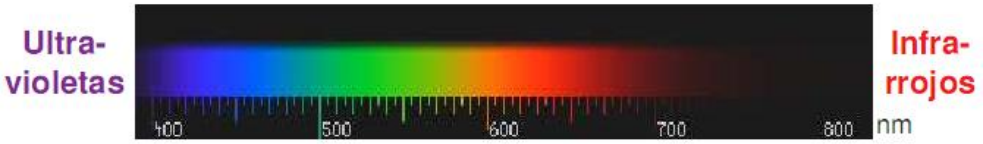


FIGURA 1. Espectro Electromagnético.
Fuente [9]

El ojo humano debido a su estructura, percibe los colores como una combinación de tres colores cromáticos llamados colores básicos rojo (R), verde (G) y azul (B); estandarizados por la Comisión Internacional de Iluminación (CIE) en el año de 1931, la cual designó las siguientes longitudes de onda para dichos colores básicos ó primarios de luz: 435.8nm (azul), 546.1nm (verde) y 700nm (rojo) [5]. Erróneamente se cree que con la combinación de los colores primarios podemos obtener todos los colores visibles, pero esto no es cierto puesto que no estamos permitiendo que las longitudes de onda varíen.

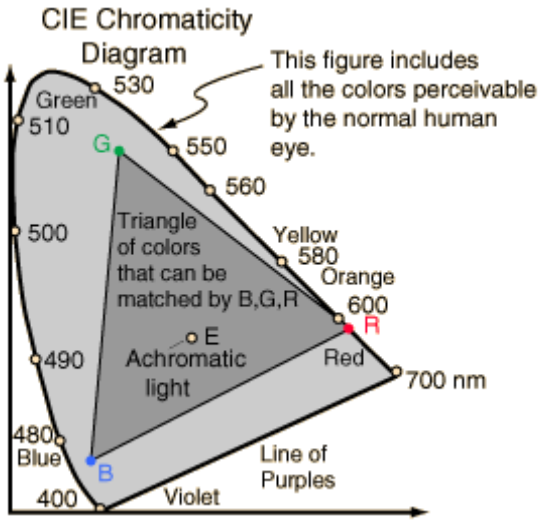


FIGURA 2. Representación del espacio RGB dentro del espectro visible. Fuente [11]

1.1.2. Espacios de color

Un modelo de color en un modelo matemático, que describe la forma en que se representan los colores mediante tuplas de 3 o 4 números; así mismo el conjunto formado por las combinaciones de dichas tuplas es lo que se conoce como espacios de color [10]. Los espacios de color son empleados por los sistemas de visión artificial para una representación digital de los colores.

1.1.2.1. RGB

Descrito anteriormente, basado en un sistema de coordenadas cartesiano, la mayoría de las cámaras que se emplean para adquirir imágenes digitales emplean el formato RGB. RGB también es usado para computadoras graficas.

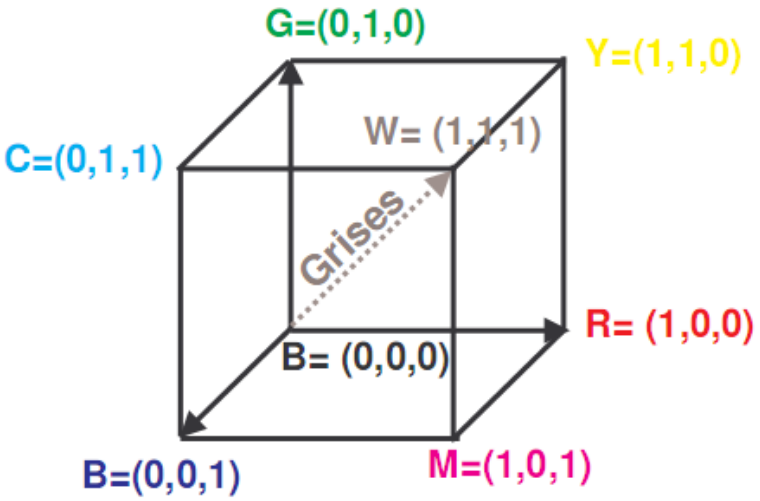


FIGURA 3. Diagrama Espacio RGB.
Fuente [10].

1.1.2.2. CMYK

Este espacio se define por los tres colores primarios de pigmentación: C (Cian), M (Magenta), Y (Amarillo) Y K (negro). En este caso un color primario de pigmentación se define según la absorción de luz del objeto.

Es decir, un color primario de pigmentación se define como uno que absorbe un color primario de luz y transmite los otros dos. El espacio CMYK es usado frecuentemente en las impresoras.

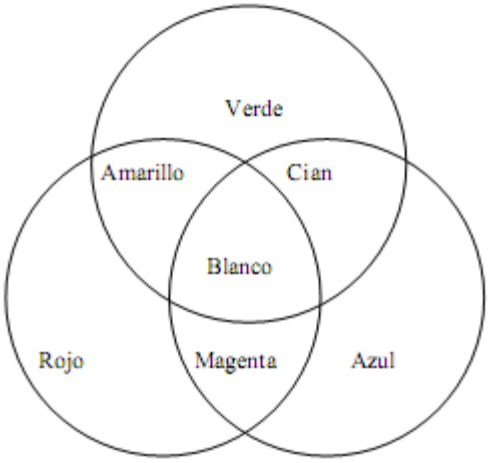


FIGURA 4. Espacio CMYK.
Fuente [5]

1.1.2.3. YUV y YIQ

YUV, formato de color usado por los estándares de televisión NTSC, PAL y SECAM. La Y representa la información en blanco y negro y la UV representan la información de color.

El formato YIQ, también se usa en las emisiones de televisión, en este formato se busca darle importancia a la televisión monocromo, por tal razón se usa mas ancho de banda en al transmisión de la luminancia (componente Y).

1.1.2.4. HSV

Consta de los componentes H (matiz), S (saturación) y V (valor de intensidad); este espacio de color suele representarse como en la Figura 5. H: ángulo dentro de la rueda cromática, S: diferencia de color respecto a un gris con la misma intensidad y V: cualidad de ser mas claro o mas oscuro. El modelo HSV esta pensado para ser mejor interpretable y legible por el ojo humano.

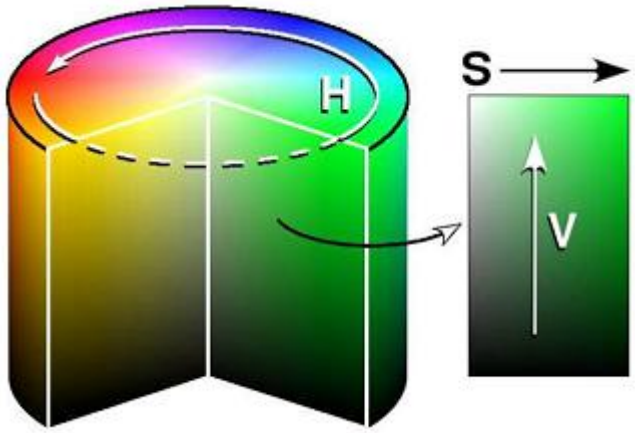


FIGURA 5. Espacio HSV.
Fuente [12].

1.1.2.5. Espacio de los Colores Oponentes de Hering.

Ewald Hering, se interesó en saber porqué hay ciertos pares de colores que nunca se ven mezclados, es decir, no se observa un verde rojizo, un azul amarillento (Figura 6), a partir de este pensamiento concluyó que dichas transiciones de color podían hacerse siempre y cuando se pasara por algún color intermedio; para pasar de rojo al verde se puede pasar por el color intermedio como puede ser el amarillo, para pasar de azul a amarillo se puede pasar por el color intermedio como el verde.

Las teorías de Colores Oponentes de Hering y Tricromaticidad (RGB) de Young-Helmholtz, se complementan al dar una explicación de la percepción de color por el ojo humano [6].

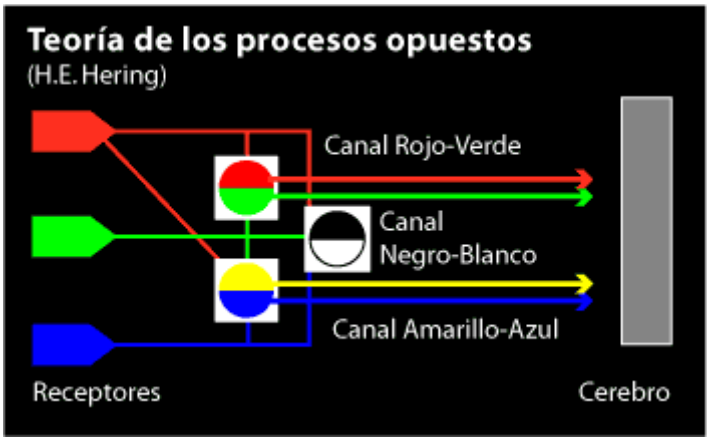


FIGURA 6. Espacios Oponentes Hering.
Fuente [13]

La transformación del espacio de color RGB al espacio de color de Hering, se logra mediante un sistema de ecuaciones lineales:

$$\begin{bmatrix} RG \\ YB \\ B_k W \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ 1/2 & 1/2 & -1 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2)$$

El rojo puro en RGB tiene valor en las tres componentes en Hering, por lo que desarrolló una nueva transformación no lineal, en la que el rojo puro no contenga valores en la componente YB de Hering, el nuevo espacio Hering [6]:

$$RG = R - G$$

$$WB_K = \frac{1}{3}(R + G + B) \quad (3)$$

$$YB = \frac{RG}{255^2}\left(\frac{R+G}{2} - B\right) - \frac{B}{255}\left(B - \frac{R+G}{2}\right)$$

En [2], se encuentra información acerca los otros espacios de color, así como de las conversiones para transformar un espacio a otro.

1.2. UMBRALIZACIÓN

Es una de las técnicas mas usadas en la segmentación debido a su sencilla implementación, dicha técnica se basa simplemente en la binarización de la imagen.

Para hacer más sencilla la explicación de Umbralización, se debe tener clara la definición de lo que es un histograma.

- **Histograma**

El histograma de intensidades de una imagen nos da información de la cantidad de pixeles de la imagen presentes en cada nivel de intensidad disponibles en la misma. En el eje X se presentan de forma ordenada los posibles niveles de intensidad, y sobre cada nivel, eje Y, se presenta una barra vertical que representa la cantidad de pixeles que posee ese nivel de intensidad en particular [15].

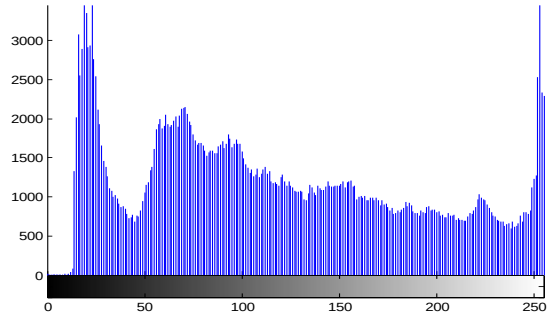


FIGURA 7. Histograma.
Fuente Autores

Con la definición de histograma, se da una definición más concreta de lo que es umbralización. En la Figura 7 se observa el histograma de una imagen, el cual presenta modas[5] dominantes.

La Umbralización global se define:

$$g(x,y) = \begin{cases} 1 & si\ f(x,y) \geq T \\ 0 & si\ f(x,\acute{y}) < T \end{cases} \qquad (4)$$

Donde $g(x,y)$ es la imagen $f(x,y)$ binarizada y T es el umbral empleado para la binarización; en este caso donde se empleó un umbral constante para toda la imagen hablamos de Umbralización global; cuando tenemos una imagen a la cual le aplicamos Umbralización por regiones, donde cada región tiene su propio valor de umbral, hablamos entonces de umbral adaptativo, el cual es especial en casos donde la imagen presenta iluminación no uniforme o sombras, en donde la aplicación de un único umbral a toda la imagen con seguridad no lleva a los mejores resultados.

1.2.1. Umbralización Multinivel

El uso de umbrales múltiples, es útil en casos en los cuales el interés es segmentar la imagen en varias regiones no binarias cada una de ellas con un conjunto muy limitado de niveles de gris, se habla entonces de Umbralización Multinivel:

[5] Máximos Locales

$$g(x,y) = \begin{cases} 1 & si\ f(x,y) \in D1 \\ 2 & si\ f(x,y) \in D2 \\ 3 & si\ f(x,y) \in D3 \\ & \vdots \\ n & si\ f(x,y) \in Dn \\ 0 & en\ el\ resto \end{cases} \quad (5)$$

Donde $g(x,y)$ es resultado e aplicarle a la imagen $f(x,y)$ Umbralización Multinivel, y $D1, D2, D3, \dots, Dn$, cada uno es un conjunto de niveles de gris presentes en la imagen, teniendo en cuenta que ninguno de ellos debe contener el mismo nivel de gris, es decir:

$$D1 \cap D2 \cap D3, \dots, Dn = \emptyset \quad (6)$$

1.2.2. Umbralización, P-cuantil.

Este método, se basa en el conocimiento a priori del área o tamaño de la región del objeto que se desea extraer para segmentar la imagen. Se emplea cuando los objetos ocupan un porcentaje P del tamaño del área total de la imagen; su uso es muy limitado aunque en aplicaciones para el preprocesamiento en reconocimiento de caracteres es muy útil [5].

1.2.3. Método de OTSU

Es uno de los métodos clásicos para umbral izar en la literatura, es un método basado en la información estadística y espacial del histograma de una imagen. Su objetivo es principalmente encontrar el valor del umbral T para el cual la varianza $\sigma^2(T)$ entre las regiones sea máxima. Aunque el método de OTSU es básicamente un método de Umbralización global, también esta presente como un

método de Umbralización Multinivel [14]. Dentro de las variables estadísticas que emplea el método de OTSU se encuentran:

- Media Aritmética ($\bar{x}$): La media aritmética o promedio de una cantidad finita de números de una muestra o población, se calcula sumando cada un de los datos (x_i) que pertenecen a la muestra, y este resultado es dividido en la cantidad de números sumados (N).

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} \qquad (7)$$

- Probabilidad: partiendo de un espacio muestra A, en el cual todos sus los sucesos elementales que lo componen son equiprobables; la probabilidad de un suceso B se define como el cociente entre el numero de resultados favorables a que ocurra el proceso B en el experimento y el numero de veces que se realiza el experimento:

$$P(B) = \frac{número\ de\ casos\ favorables\ al\ suceso\ B}{número\ de\ veces\ que\ se\ realiza\ el\ experimento} \qquad (8)$$

- Varianza entre clases σ^2: se define como el cuadrado de la desviación estándar, y esta a su vez es la variable estadística que me indica que tan dispersa esta la distribución de cada clase respecto a la media de todas las clases. Esta definida para el tratamiento digital de imágenes por [14]:

$$\sigma^2 = \sum_{k=1}^{N} \omega_k \left(u_k - \mu_T\right)^2 \qquad (9)$$

Donde N es el número de clases o regiones, ω_k es la probabilidad de que los pixeles pertenezcan a una clase k y μ_T la media global de la imagen.

1.3. PROYECCIONES

Teniendo en cuenta que una imagen en niveles de gris es un arreglo matricial $f(i,j)$, podemos emplear como método de segmentación de la región de interés en una imagen, el método de las proyecciones [18]; básicamente este método es el resultado de un análisis estadístico sobre los ejes X e Y. Como resultado de la aplicación del método de las proyecciones se obtiene un vector para el eje X, ó un vector para el eje Y, o ambos según sea el caso; matemáticamente dichos vectores se obtienen de la siguiente manera:

$$p_x(x) = \sum_{j=o}^{h-1} f(x,j)\,; \qquad p_y(y) = \sum_{i=o}^{w-1} f(i,y) \qquad (10)$$

En (10), h representa las dimensiones en el eje Y de la imagen $f(i,j)$; y w representa las dimensiones en el eje X de la imagen $f(i,j)$.

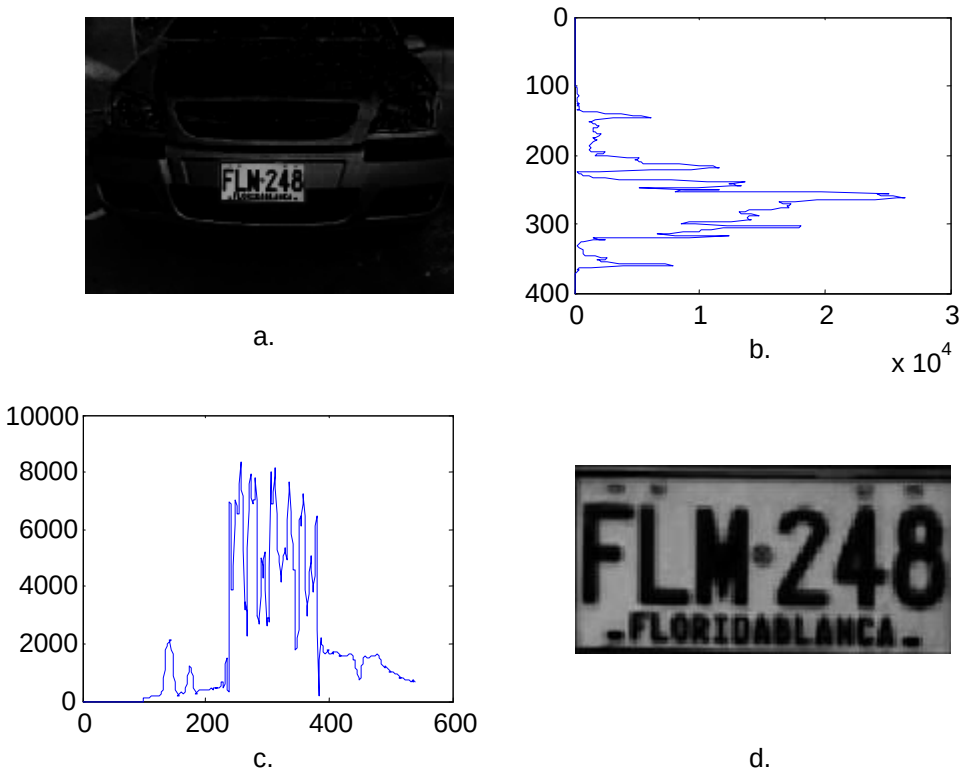


FIGURA 8. Proyecciones Eje Y y Eje X.

a. imagen Prueba. b. Proyección eje Y. c. Proyección Eje X. d. Resultado Segmentación método Proyecciones.

Fuente Autores.

1.4. OPENCV

OpenCV es un código abierto[6] de visión por computador, conformado por un conjunto de librerías[7] escrita en C y C++ optimizado. Corre bajo Linux, Windows y Mac OS X. Esta librería se puede ejecutar bajo interfaces como Python, Ruby, Matlab, y otros idiomas.

OpenCV fue diseñado para optimizar la computación y con un fuerte énfasis en aplicaciones en tiempo real.

[6] http://opensource.org
[7] http://SourceForge.net/projects/opencvlibrary

Uno de los objetivos de OpenCV es proporcionar un fácil uso de la infraestructura de cómputo visual que ayuda a las personas a construir sofisticadas aplicaciones de visión rápidamente. Contiene más de 500 funciones que abarcan muchas áreas de visión. OpenCV también contiene una plena Machine Learning Library (MLL), para fines generales,

Esta sublibrería estadística se centra en el reconocimiento de patrones y las agrupaciones. El MLL es muy útil para la visión de las tareas que están en el núcleo de la misión de la OpenCV.

Aunque Intel comenzó OpenCV, la intención de la librería, es y siempre fue de promover investigación. Por lo tanto, es abierta y libre, y el código en sí mismo puede ser incorporado o utilizado en otras aplicaciones, ya sea comercial o investigación. Esto no quiere decir que la aplicación que se desarrolle bajo este lenguaje, sea libre.

2. TARJETA ECB_AT91

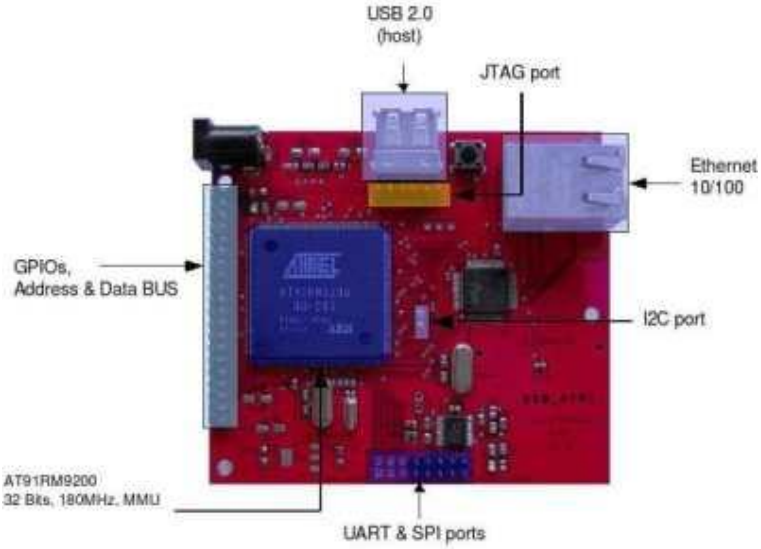


FIGURA 9. Tarjeta ECB_AT91 frente.
Fuente [20].

La tarjeta ECB_AT91[8] es una plataforma embebida moderna, realizada por la empresa emQbit, la cual es especializada en el diseño de Tarjetas de computadores, sistemas embebidos, expertos en diseño de Hardware y software, programas para FPGAs, DSP y microprocesadores, desarrollo de aplicaciones basadas en GNU/LINUX, ECOS para sistemas embebidos.

[8] http://wiki.emqbit.com/free-ecb-at91

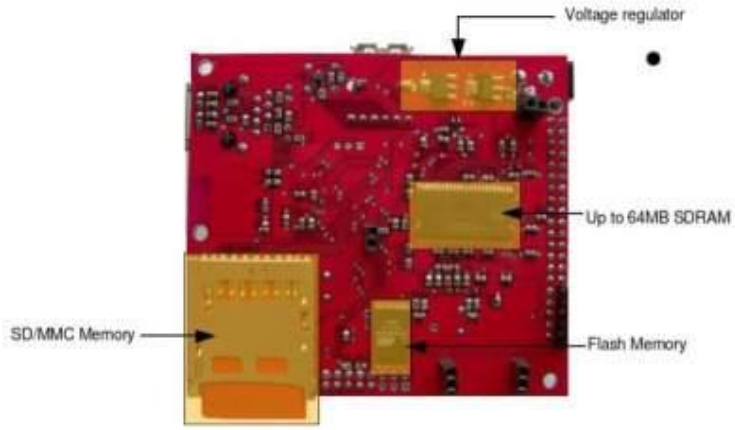


FIGURA 10. Tarjeta ECB_AT91 Respaldo.
Fuente [20].

La empresa emQbit emplea GNU/Linux en sus dispositivos ya que manejando el Kernel de Linux, se tiene Ethernet, Bluetooth y WIFI gratis, también IP, TCP, UDP, IPv4, IPv6, NFS, PPP, y muchos más protocolos que se pueden utilizar en las aplicaciones desarrolladas por los usuarios.

ECB_AT91 esta basada mas específicamente en el sistema operativo DEBIAN GNU/LINUX, ya que es un sistema operativo libre, gratis y completo, basado en el núcleo de Linux, pero la mayor parte de las herramientas básicas vienen del Proyecto GNU.

Por su parte, Linux, el núcleo del sistema operativo Debian, es el kernel de dicho sistema, el cual es un clon del sistema operativo Unix, realizado por Linus

Benedict Tolvars[9] a finales de 1991, con la ayuda de un equipo especializado en informática. Tiene todas las características modernas que puede ofrecer el sistema Unix, incluido el auténtico multitarea, memoria virtual, bibliotecas compartidas, compartidos copia por escritura ejecutables, correcta gestión de memoria, y la creación de redes multistack incluyendo IPv4 e IPv6.

Aunque originalmente desarrollado para PCs x86 de 32 bits (386 o superior), hoy también se ejecuta en el Alpha AXP, Sun SPARC, Motorola 68000, PowerPC, ARM, SuperH de Hitachi, IBM S/390, MIPS , HP PA-RISC, Intel IA-64, AMD x86-64, AXIS CRIS, Renesas M32R, Atmel AVR32, Renesas H8/300, NEC V850, Tensilica Xtensa, Analog Devices.

Las especificaciones de la tarjeta ECB_AT91[10] utilizada en el proyecto son las siguientes:

- Procesador ARM9 de 180 MHz (Atmel AT91RM9200)
- 2 MBytes de memoria flash serial.
- Hasta 64 MBytes de memoria SDRAM (Soporta 8M/16M/32M/64M)
- 1 slot SD/MMC
- 1 Puerto USB 2.0
- 1 Puerto I2C
- 1 interfaz Ethernet 10/100
- 4 interfaces SPI
- 2 interfaces seriales (RS232)
- Soporte JTAG
- Bus de Datos (16 bits), Dirección (7 bits) y Control Disponibles.
- 8 Pines de Entrada/Salida de Propósito General Disponibles.

[9] Ingeniero de software finlandés, nacido el 28 de diciembre de 1969, desarrollador de la primera versión del kernel GNU/Linux, el 5 de octubre de 1991

[10] Tomado de http://wiki.emqbit.com/free-ecb-at91

El procesador posee una arquitectura ARM9, el cual esta incluido en la familia de microprocesadores RISC, diseñados por la empresa Acorn Computers, y desarrollados por la empresa derivada de la anterior, Advanced RISC Machines Ltd., microprocesadores utilizados en aparatos electrónicos como son las consolas portátiles de videojuegos de la fabrica Nintendo[11], en productos de la fabrica Apple, como lo es el Iphone, entre otros dispositivos, y utilizados para una amplia gama de aplicaciones, videoteléfonos, celulares, PDAs, consolas de videojuegos, audio MP3, MP4, impresoras de escritorio, cámaras digitales, telemática.

ARM9TDMI e incorpora el set de instrucciones Thumb, el cual provee una mejora en la densidad de código de un 35%. La familia ARM9 contiene un amplio conjunto de funciones que permite a los desarrolladores aplicar sistemas modernos, y al tiempo ofrece ahorros en el área del chip, consumo de energía, ahorro en los costos de comercialización, de desarrollo.

La familia de procesadores ARM9 incluye los microchips de cache ARM920T y ARM922T, los cuales han sido desarrollados para los diferentes requerimientos de las aplicaciones: dual 16K y 8K caches para aplicaciones Symbian OS, Palm OS, Linux y Windows CE.

En si el procesador Atmel AT91RM9200[12] esta basado en ARM920T con la instrucción de 16K-bytes y 16K-bytes de memoria caché, 16 K-bytes SRAM, 128K bytes de ROM. SDRAM, Burst Flash y Static Memory Controllers, interface dispositivos USB y Host, Ethernet 10/100 BaseT MAC, controlador de administración de energía, Reloj en tiempo real, sistema de temporizador, controlador serie síncrona, Dos Wire-Interface, Interfaz de periféricos serie, Multimedia Card y el controlador de la interfaz paralelo I / O.

La tarjeta ECB_AT91 posee 2 MBytes de memoria flash, la cual permite que múltiples posiciones de memoria sean borradas o asignadas en una misma operación de programación mediante un pulso eléctrico, funciona a velocidades

[11] Empresa multinacional desarrolladora de software y hardware para videojuegos.
[12] Datasheet http://www.keil.com/dd/docs/datashts/atmel/at91rm9200_ds.pdf

muy superiores a sus antecesoras, las memorias EPROM, sólo podían ser leídas o borradas una celda a la vez. Las memorias flash son de tecnología no volátil, lo que significa, que no se borra la información cuando se desconecta la energía, por lo cual son muy utilizadas en dispositivos que necesitan esta característica, como son los teléfonos móviles, PDAs, cámaras de fotos digitales, MP3, MP4, etc. Este tipo de memorias, están fabricadas con compuertas lógicas NOR o NAND, para así almacenar ceros o unos. Actualmente en el mercado se pueden encontrar memorias flash desde los 128 MB hasta 32 GB con una velocidad de transmisión de 30 MBytes/segundo. Sin embargo, las memorias flash, cada celda de ella, tienen una vida útil de aproximadamente 10.000 y un millón de lecturas y borrados, dependiendo de las celdas, el voltaje necesario para su borrado.

Posee una ranura para memorias SDRAM (sus siglas en ingles, Synchronous Dynamic Random Access Memory, es decir, memoria RAM dinámica de acceso síncrono (misma velocidad que el sistema)), también conocidas como SDR SDRAM, la cual permite conectar de 8MB a 64MB de capacidad. Este tipo de memorias RAM, poseen tiempos de acceso de 10 y 25 ns, y utilizan tecnología DIMM (Dual In-line Memory Module, Módulo de Memoria lineal doble) de 128 contactos. Según la capacidad de trabajo, las memorias pueden ser: PC66 (66 MHz de velocidad de bus), PC100 (100MHz) y PC133 (133 MHz).

También posee una ranura para memorias SD/MMC, SD (Secure Digital) es un formato de tarjeta de memoria flash, utilizadas en dispositivos portátiles, por su versatilidad y tamaño, poseen dimensiones de 32X24X2,1 mm. Existen de 128,256 y 512 MB, y 1, 2, 4, 8, 16, 32 GBytes, con velocidades de 400Kbps, 25Mbps y 100Mbps. MMC (MultimediaCard) hace referencia a otro tipo de tarjeta de memoria flash, muy parecida a las tarjetas SD, aunque mas delgadas, con una capacidad de hasta 4 GB.

Además, la tarjeta posee diferente tipos de interfaces y puertos, que son útiles a la hora de implementar las diferentes tipos de aplicaciones, como son, puerto I2C, interfaces como la de Ethernet, USB 2.0, 4 SPI, 2 Serial RS232.

Referente al puerto I2C (Inter-Integrated Circuit), es un bus de comunicación implementado desde el año 1992, por la empresa Philips[13], con velocidades de 100 Kbits por segundo en su modo estándar, aunque soporta velocidades de 3.4 Mbits/seg. Es muy utilizado a la hora de comunicar, micro controladores y los periféricos en sistemas embebidos. Utiliza dos líneas de comunicación, una línea es empleada para la transmisión de los datos, entre tanto, la otra línea para la señal de reloj, y posee una tercera línea que es la línea de referencia (GND).

Concerniente a las interfaces que posee la Tarjeta, la 10/100 Ethernet, es para poder manejar una red Ethernet, la cual es la mas representativa desde los años 70s.

SPI (Serial Peripheral Interface), es un bus de comunicación, aunque mas rápido y eficiente, en relación del consumo de energía, que el bus I2C, aunque con la desventaja de que posee mas pines, posee comunicación full-dúplex, posee cuatro líneas, una para la señal de reloj, una línea utilizada para los datos salientes, otra para los datos saliente, y por ultima una línea denominada chip select, la cual conecta o desconecta el receptor.

[13] Empresa Multinacional de electrónica, con sede en Ámsterdam, fundada por Gerard Philips y familia en el año de 1981.

3. ADQUISICIÓN IMAGEN

Una particularidad de los trabajos basados en el tratamiento digital de imágenes, es el proceso de adquisición de la imagen a ser tratada, el cual juega un papel importante, pues dependiendo de esto se obtiene un buen desempeño del sistema desarrollado. Hay que tener en cuenta, en el momento del registro entran muchos factores en juego, como lo son la iluminación del lugar, la distancia de la toma, las condiciones del clima, la forma y la nitidez de la placa del automóvil, hacen que los resultados sean óptimos o fatales.

3.1. CONDICIONES GENERALES

Debido a la importancia de la captura de la imagen en este proyecto, se tuvieron en cuenta dos formas de capturas de ellas, debido a las condiciones reales de la captura, y tomando en cuenta las limitaciones de la tarjeta, no se podía generalizar demasiado, por eso se decidió realizar unos ciertos criterios referentes a este tema, para así estandarizar un poco el procedimiento.

Referente a la iluminación se realizaron al aire libre, con la iluminación natural del día, en la mañana y tarde antes de las 4 pm, donde la iluminación es óptima para este tipo de tomas.

En cuanto a la distancia de la captura, se trató de mantener una distancia considerable de un metro, desde la placa a la cámara, para mantener una relación entre el tamaño de imagen de la placa y el tamaño de la imagen original, cercana al 4%. Otra característica implícita, es la altura de la ubicación de la cámara, se

ubicó a la misma altura de las placas, teniendo en cuenta, que el parqueadero debe tener un control al acceso de los autos, para poder ubicar la cámara a esta altura, como se muestra en la siguiente figura.

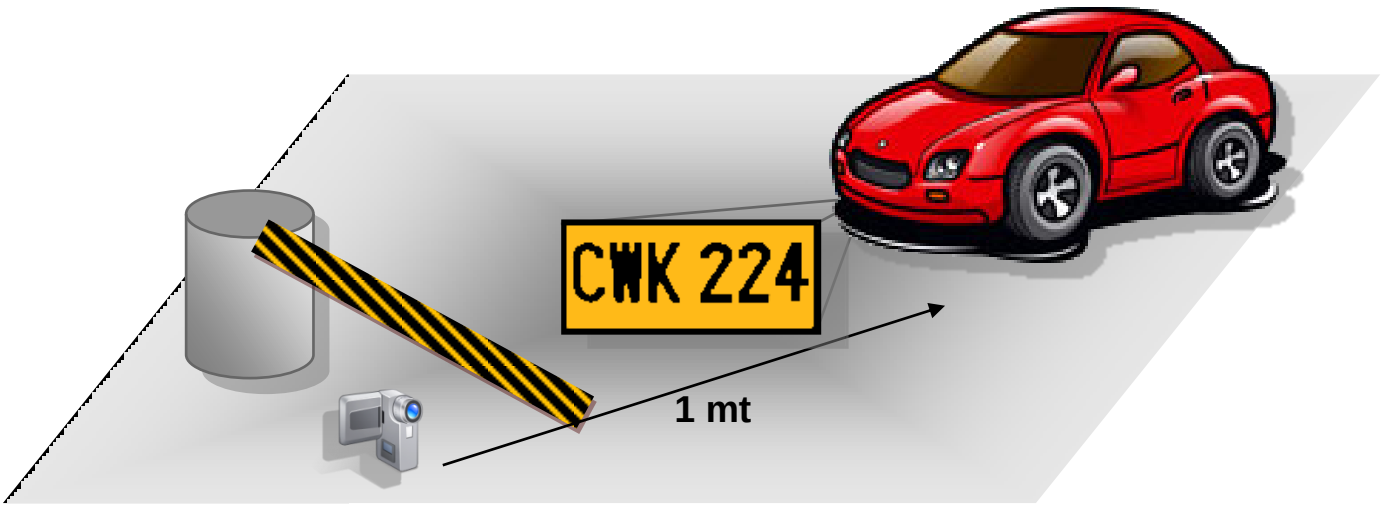


FIGURA 11. Diagrama de distancias de Captura.
Fuente Autores.

3.2. PROCESO DE CAPTURA.

Las cámaras digitales se han convertido cada vez más populares en los últimos años. La tecnología de la imagen digital ha crecido a nuevos mercados incluyendo teléfonos celulares y dispositivos PDA. Con los diversos mercados, una variedad de tecnologías de la imagen están disponibles. La tecnología de imágenes se ha ampliado para incluir tanto dispositivo acoplado por carga (CCD) y sensores de imagen CMOS.

Un sensor CCD o CMOS de imagen proporciona la tecnología digital para capturar una imagen. Sensores de imagen CCD se utilizan en muchas aplicaciones de gama alta, como cámaras digitales de alta resolución, ofrecen una mejor imagen en entornos de luz baja más que los sensores CMOS, pero son más costosos que estos últimos.

Sensores de imagen CMOS consumen mucho menos energía que los CCD. La cámara digital es capaz de funcionar con pilas, una gran ventaja en productos portátiles. Dado que los sensores CMOS poseen la misma plataforma de fabricación que la mayoría de microprocesadores y chips de memoria, son más fáciles de producir y más rentable que los CCD. Los sensores de imagen CMOS requieren un único poder suministro para la operación y sólo de 20-50 milivatios por píxel de salida.

El registro de la imagen del auto con su respectiva placa se trató de realizar con una cámara de un teléfono móvil, proporcionada por la Universidad, con tecnología CMOS.

El sensor de imagen de la cámara digital utilizada, es de diseño de la fábrica Spark Fun. Posee un sensor de imagen CMOS. Las características correspondientes de la cámara no se encontraron en si, pero la guía de cómo manejarla y de cómo funciona, fueron tomadas de otras dos cámaras muy parecidas, la vs6624 y la Spark Fun SENSE-CCAM, esta última con mayor similitud.

Este tipo de cámara puede emitir datos digitalmente procesados RGB o YCrCb. Este sensor CMOS es de tamaño 640 x 480 VGA y tiene más de 100 registros programables para la personalización de la misma, posee mas de 20 pines de señales, aunque los mas utilizados son: Y0-Y7 [salida], 8 pines para las salidas de las señales de los datos de los pixeles; VCC, 2 pines para energizar la cámara, 2.8-3 Volts recomendados, 3.2 Volts máximo, una análoga y la otra digital; GND, 2 pines de punto de referencia; MCLK [entrada], pin para el reloj al cual va a trabajar la cámara, no puede ser mayor de 25 Mhertz; Pclock [salida], es el reloj de sincronización de la salida de los datos de los pixeles; LVL y FVL [salidas],

señales de control de cambio de línea y de frame (imagen), respectivamente; SDA y SCL [entradas], señales de comunicación I2C, datos y reloj, respectivamente; que es como se accesa a los registros de la cámara, para así poder realizar cambios a la hora de la toma de las imágenes; ENB [entrada], pin para activar la cámara, entrar en función Standby o estar activa.

Referente a las señales de control, las formas de ondas de estas son:

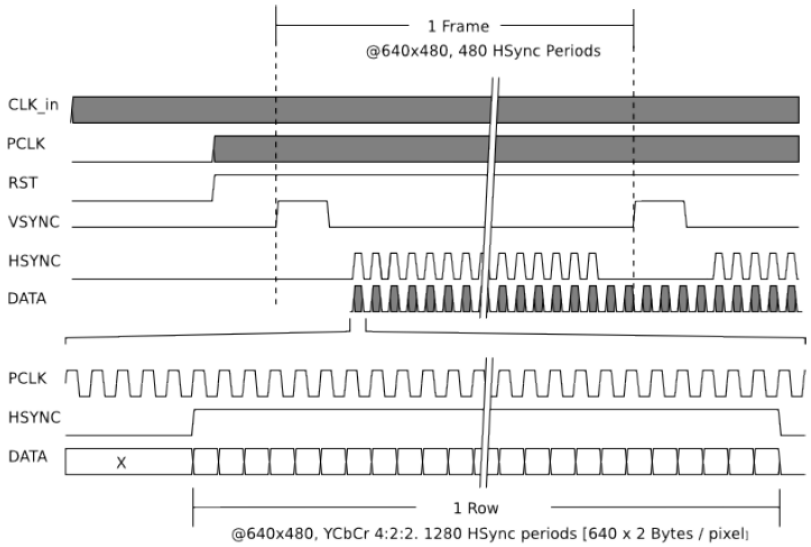


FIGURA 12. Diagrama Señales de Entrada/Salida cámara.
Fuente [19].

PCLK Y CLK_in, no necesariamente están a la misma frecuencia y se encuentran en fase, esto depende de cómo se configure los registros de la cámara por medio del SDA. Para entrar en funcionamiento, ENB debe encontrarse en un uno lógico (VCC), VSYNC, indica cuando se inicia y termina la adquisición de la imagen como tal, cuando se ha realizado el barrido de los 307200 puntos de la imagen, como se muestra en la figura 12. HSYNC, indica cuando se está realizando el barrido de una línea horizontal, o sea el barrido de los 480 puntos de cada línea, entonces por ende deben haber 480 pulsos de reloj de HSYNC, para así completar las 480 líneas horizontales. Mientras HSYNC se encuentra en un uno, se están enviando los 8 bits, referentes a los datos de los píxeles, estos se

deben leer en cada pulso de bajada del PCLK. En resumen, una imagen esta comprendida de todos los datos enviados por la cámara, mientras HSYNC se encuentra en alto, entre un pulso y otro de VSYNC.

La cámara es compatible con múltiples formatos de datos, el valor por defecto de las cuales es 4:2:2 YCbCr. YCbCr es un formato que codifica el color como la luminancia del color [cómo es brillante], Y, Cb Cr y que en conjunto indican el tono de color. La notación 4:2:2 indica que por cada 4 luminancia valores, se obtiene 2 Cb, y 2 Cr. La razón de esto es que el ojo humano es mucho más sensible a la luminancia que es a las variaciones de color, por lo que no se notará mucho la pérdida de calidad si se utiliza el mismo Cb / Cr de dos píxeles.

En el modo por defecto, los datos se transmiten en el siguiente orden:

Y0, Cb, Cr, Y1, Y2, Cb, Cr, Y3, etc.

El mismo patrón se repite para el resto de la imagen, ofreciendo 4 bytes de datos por cada 2 píxeles.

La programación para el funcionamiento de la cámara, se realizó con la ayuda de una FPGA[14] modelo Xilinx Spartan 2E XC2S200E, la cual esta integrada en la tarjeta Digilent 2E diseñada por Xilinx[15], con una velocidad de funcionamiento de 50 MHz. Se programó la cámara teniendo en cuenta sus características. La captura se realizó en una resolución de 640x480 pixeles, y en formato 4:2:2 YCbCr, espacio de color que se explica en el siguiente capitulo.

Una vez iniciada la Cámara, conectado todos sus pines, las fuentes de alimentación, la captura ya puede ser inicializada y ser enviada a otro dispositivo en forma serial. La interfaz utilizada entre la cámara y la tarjeta Digilent se muestra en la siguiente figura:

[14] Field Programmable Gate Array, dispositivo contenido por bloques de lógica programable. Ross Freeman 1984.
[15] Empresa desarrolladora de FPGAS, creada en 1984.

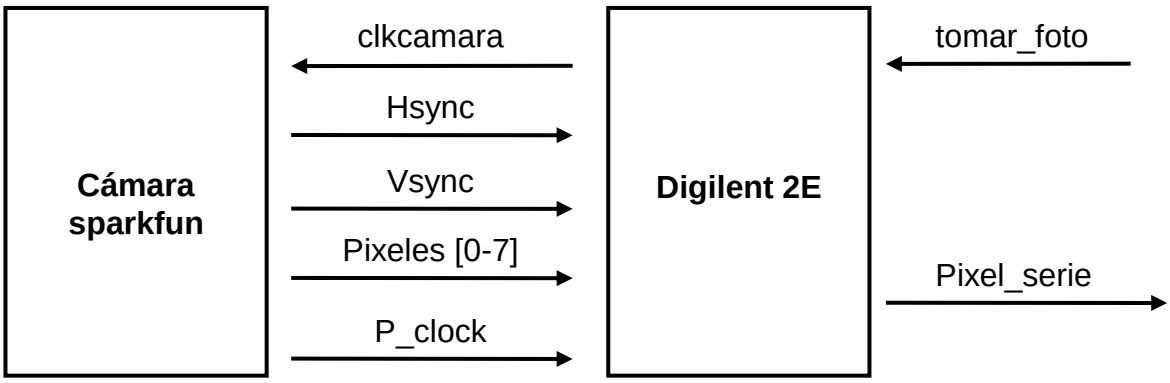


FIGURA 13. Diagrama de Bloques. Fuente Autores

La maquina de estados utilizada para la captura que cada imagen, se muestra en la figura 14.

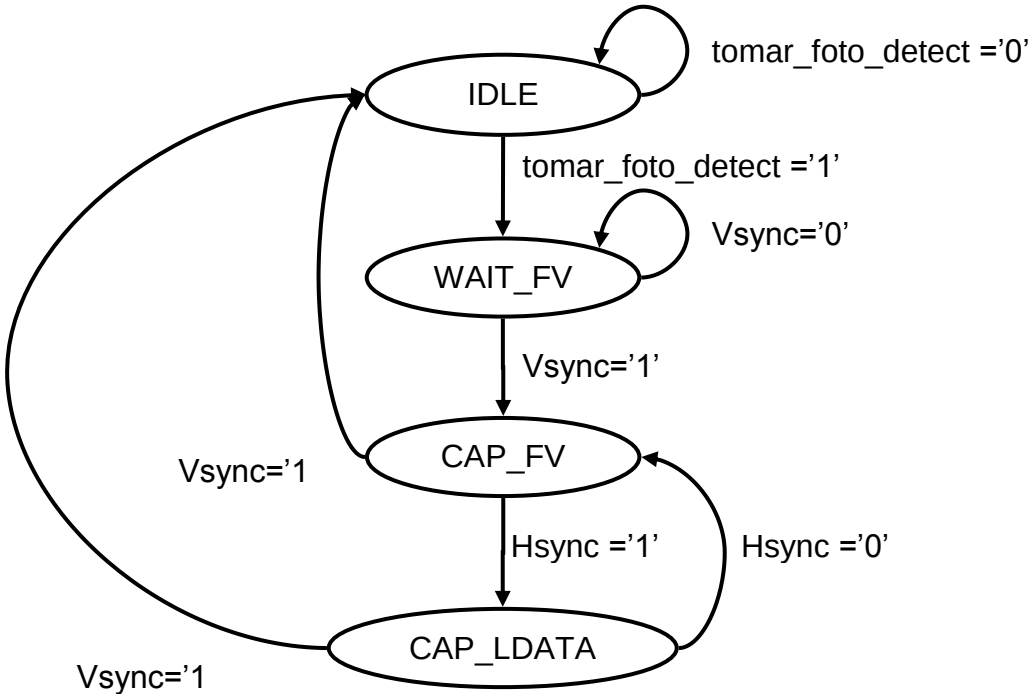


FIGURA 14. Diagrama Maquina de Estados. Fuente Autores.

La descripción de cada uno de los estados:

Nombre Estado	Descripción
IDLE	Espera por la señal de control tomar_foto. La cual es la que activa el automóvil en cierto punto.
WAIT_FV	Espera el aumento de borde en la señal Vsync
CAP_FV	Esperar que la señal de control de las líneas Horizontales sea valida, Hsync.
CAP_LDATA	Recibir los datos, enviados por la cámara, mientras Hsync sea valido, y con el flanco de bajada de P_clock.

El diseño para el funcionamiento de la cámara, fue probado al enviar los datos adquiridos en forma serial, a un PC con la herramienta de Matlab[16], para así poder armar la matriz correspondiente a cada imagen capturada.

El código en VHDL[17] del diseño se encuentra en el Anexo C.

El único inconveniente que se tuvo con la captura de esta forma, es en la forma como funciona la FPGA, ya que este dispositivo es volátil, es decir, se elimina lo programado a la hora en que desconecta la corriente, por lo que no era muy conveniente a la hora de transportar el dispositivo para realizar las tomas. Aunque el dispositivo si puede ser instalado directamente en el sitio de un parqueadero, ya que presenta la posibilidad de ser programada de una forma especial mediante otro dispositivo para que su programación no sea eliminada. Por lo que para las pruebas a realizar, se decidió tomar las imágenes con la cámara de un teléfono móvil instalada en uno de ellos. El teléfono móvil utilizado, es el modelo L6i de la fabrica Motorola[18], el cual posee una cámara con especificaciones parecidas a la mencionada durante el capitulo.

[16] MATrix LABoratory, software matemático con un entorno de desarrollo integrado (IDE)
[17] Lenguaje programacion usado para diseñar circuitos digitales.
[18] Empresa estadounidense, especializada en electrónica y telecomunicaciones, 1928.

4. SEGMENTACIÓN

Posterior a la etapa de adquisición de la imagen, encontramos la etapa de segmentación. El objetivo al segmentar una imagen es encontrar y extraer las zonas y objetos de interés, la idea es encontrar los pixeles que se encuentran juntos e identifican al objeto o al área que se desea trabajar, dichos objetos entonces se pueden clasificar según su color, textura, forma, etc.

4.1. SEGMENTACIÓN DE LA PLACA

El proceso de segmentar la placa, radica en que se debe centrar la atención en donde se encuentra los caracteres, y así evitar reconocer cualquier otro tipo de elemento de la imagen que pudiese diferir el resultado que se busca, como pueden ser las letras de la marca del carro, o algún sticker adherido al automóvil.

Para tal fin se utiliza un método, el cual es eficiente y minimiza el procesado necesario para encontrar la ROI[19] en la imagen, frente a otros métodos de segmentación conocidos. Este método se describirá a lo largo del capítulo.

Para reducir el procesado de la imagen y así mejorar el tiempo total de procesado, al evaluar las tomas de las imágenes, se observó que la placa del automóvil nunca aparece en los extremos de la imagen, es decir, la ROI tiende a concentrarse en la región media de la imagen. Aprovechando este hecho, se eliminó alrededor de ¼ de la imagen total por cada lado de esta, para así reducir el número de elemento que se tienen que procesar.

[19] Región de Interés.

4.1.1. Segmentación cromática.

En un sistema de ubicación de placas, la utilización del Nuevo Espacio de Hering presenta mejor desempeño en el realce de la ROI, ya que realiza un filtro de la mayor parte de los colores que son innecesarios en la imagen, esto teniendo en cuenta que solo se utiliza la componente YB, ya que el amarillo es el color que se desea en la imagen, por la característica del fondo de las matrículas de los carros de servicio particular en Colombia.

Para poder elegir este espacio de color, se probaron los espacios mencionados en el primer capitulo, para así seleccionar el que permitiese segmentar los pixeles que contienen la información del color amarillo, siendo el de la teoría de los Colores de Hering el que produjo mejores resultados.

FIGURA 15. Imagen Original.
Fuente Autores.

FIGURA 16. Representación componente YB Espacio Hering modificado. Fuente Autores.

4.1.1.1. Binarización Imagen.

Después de obtener la imagen correspondiente a aplicarle la transformación al espacio de Hering, se procedió a realizar un método para la ubicación de la matricula en sí.

Primero, se procede a umbralizar la imagen, con el método de P-cuantil, explicado en el primer capitulo. Tomando como criterio de selección para el umbral, con base el histograma de la imagen obtenida, que la sumatoria de los pixeles con los valores máximos de gris, no sobrepasara el valor de 7000, valor seleccionado teniendo en cuenta el promedio del número de pixeles que pertenecen a la placa del vehículo. Una vez calculado el umbral, se procedió a obtener la imagen formada por el cambio de intensidad de los píxeles mayores a dicho umbral, cambiando su valor por 255, esto conservando la imagen que se encuentra en el espacio de Hering. En la Figura 17 se presenta el resultado de la umbralización de la imagen que se presenta en Figura 16.

FIGURA 17. Imagen Umbralizada.
Fuente Autores.

Una vez umbralizada la imagen, se detectó la presencia de ruido o de áreas que no son de interés en el proceso, áreas generadas por partes dentro de la placa como tornillos y remaches los cuales deforman en algunos casos los caracteres presentes en la palca.

La mayor parte del ruido de la imagen se eliminó al realizar sobre la imagen binarizada algunas operaciones morfológicas. Un proceso morfológico sobre una imagen binarizada, se trata de operaciones mediante elementos estructurales, que varían la forma de las áreas de los objetos que se encuentra en ella.

Cuando se habla de procesado morfológico se habla fundamentalmente de dos operaciones básicas: erosión y dilatación.

Con la erosión se trata de eliminar la mayor parte de los puntos que aparecen en la imagen, y separar regiones unidas por una pequeña cantidad de píxeles. En esta altura del proceso, no importa si al erosionar la imagen se alcanza a afectar las letras de las placas, porque el propósito, por ahora, es la de detectar el área de la imagen. El elemento estructural implementado, es una máscara de tamaño de 3x3 píxeles con todos sus elementos en uno.

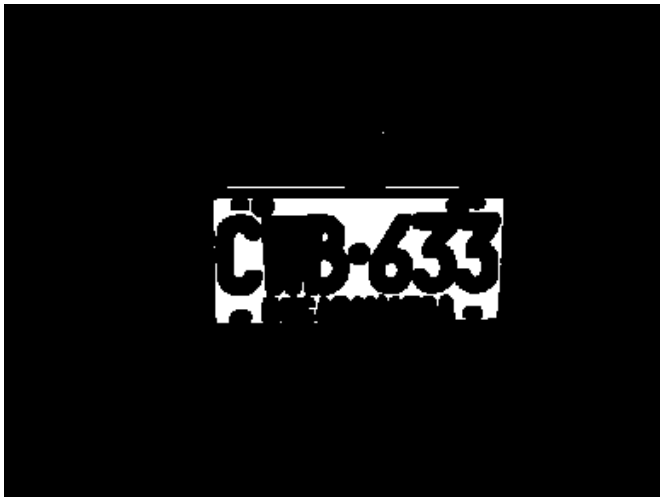


FIGURA 18. Procedimiento Morfológico. Erosión.
Fuente Autores.

4.1.1.2. Ubicación Placa.

La ubicación del área de la placa, se realizó por medio de la técnica de las proyecciones, expuesta en el capitulo 1, donde se trata de encontrar la mayor concentración de pixeles en eje Y y X.

Primero, para reducir aun mas el tamaño de la imagen, a la altura de la placa se realizó la proyección en el eje Y, y así ubicar solo la placa en estos ejes. Se trata de encontrar un punto referencia, para esto se hace suma de las proyecciones de la fila 480 hacia arriba, esto para evitar los elementos no deseados que introducen algunos automóviles, como las persianas y el logo de la marca del auto, hasta encontrar que dicha suma, sea alrededor de 1000, donde luego se escoge, unas coordenadas de 80 filas por arriba y por debajo de este punto encontrado. Ya teniendo las coordenadas, se realiza una comparación de cada una de las proyecciones de las filas, hasta encontrar en sí la altura de la matrícula.

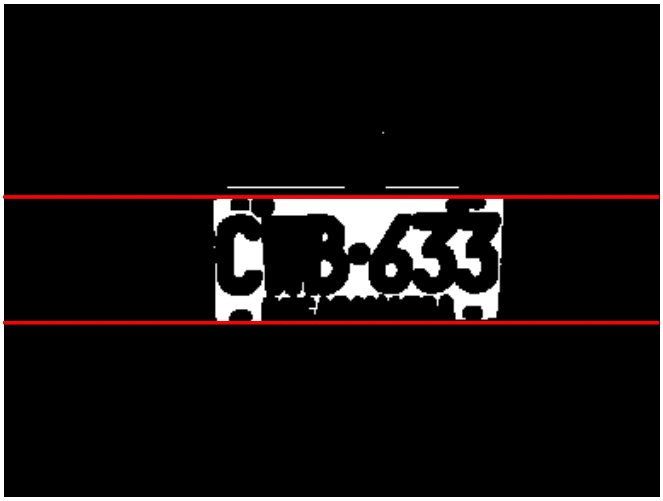


FIGURA 19. Placa con sus coordenadas Ymin y Ymax.
Fuente Autores.

Ahora con los puntos encontrados, Ymin y Ymax, mostrados en la figura 19, se realiza la respectiva proyección en el eje X pero de la imagen en YB, Figura 16, y con esta proyección encontrar los puntos Xmax y Xmin, para esto se toma como punto de referencia la mayor fila de la proyección, es decir, la que tenga el mayor valor, donde luego a partir de este valor, los puntos son tales los que sean menor que el 20% de éste. No se realizó del mismo modo como se hizo con la proyección en Y, ya que la iluminación de algunas matriculas deformaban las primera y ultima letra, por lo que al hacerlo así se recortaba gran parte de la matrícula.

FIGURA 20. Área de Matrícula.
Fuente Autores.

4.1.2. Detección Umbral y Binarización Placa.

Ya teniendo la placa segmentada, se mantiene la imagen con solo la matrícula dentro de las coordenadas encontradas en los pasos anteriores, con el fin de no gastar mas tiempo de procesado con recortar la imagen, y así poder binarizar con un método mas óptimo al empleado arriba.

El método elegido para la binarización de solo la matricula, es el método OTSU, eficiente en cuanto a la segmentación de caracteres, ya que permite diferenciar entre un carácter y otro dentro de la placa. EL método de OTSU aplicado consiste dividir la imagen en 8 partes en el eje Y, y hallar un umbral diferente a cada sección. La imagen ya binarizada con este método, se muestra en siguiente figura:

FIGURA 21. Placa Binarizada.
Fuente Autores.

4.2. SEGMENTACIÓN LETRAS.

Ya después de tener la matrícula binarizada, se procede a eliminar la mayor cantidad de ruido que se encuentra en ellas, como lo es el área de los tornillos, las letras referente a la ciudad de donde son las placas el automóvil. Para esto se realizó un código, con un proceso que hace arreglo de proyecciones y sumatorias de éstas, manejando los pixeles de la imagen obtenida. Para separar cada letra, primero se efectúa una operación morfológica, la erosión, con un máscara de 3x3para separar caracteres y eliminar puntos que no pertenecen a la misma letra. Al tener la imagen erosionada, se procede a hacer otra proyección en el eje X, para poder encontrar los mínimos a cada lado de la letra, teniendo como punto de referencia, filas que cortan, en por lo menos una parte, a la letra a encontrar. Dividida la imagen en 7 partes, teniendo en cuenta el ancho y la posible ubicación de la letra, se buscan los puntos Xmin y Xmax de cada letra, partiendo de las

sumatorias de las filas, dando como resultado la figura 22, las líneas verdes correspondiente a los Xmin y las rojas las Xmax de cada letra.

FIGURA 22. Letras recortadas, letras erosión.
Fuente Autores.

Para la parte de arriba y de abajo de cada letra, se hace un barrido por cada fila comprendida entre los puntos Xmin y Xmax, buscando la primera que su suma de por lo menos 1, para así poder hallar los punto Ymin y Ymax.

Encontrados todos los puntos Xmin, Xmax, Ymin, Ymax de cada letra, se procede a realizar otra operación de morfología, la dilatación, para así rellenar ciertos puntos de las letras, que se perdieron con la erosión.

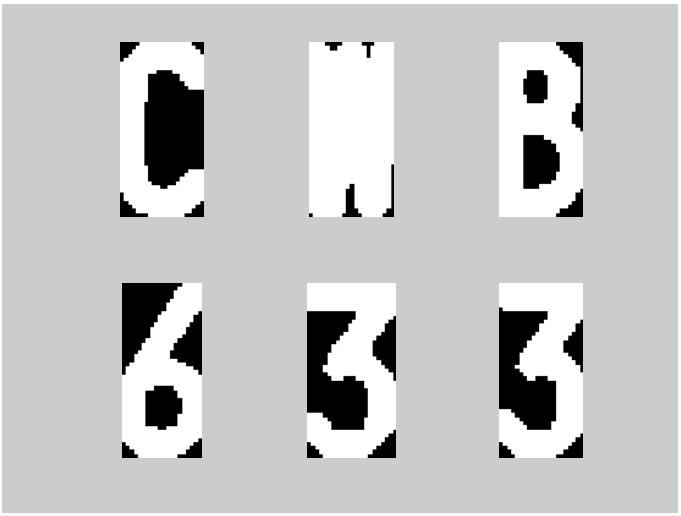


FIGURA 23. Letras Segmentadas.
Fuente Autores.

5. RECONOCIMIENTO DE CARACTERES

Aunque no se tiene claridad en la literatura sobre donde termina el procesamiento de imágenes para darle inicio al análisis de imágenes y a la visión por computadora, en este trabajo se tiene en cuenta la definición de procesamiento digital de imágenes que se presenta en [1], según la cual el procesamiento digital de imágenes abarca desde el proceso de captura de la imagen, el procesamiento de la imagen (extracción de atributos) hasta el reconocimiento de las regiones u objetos presentes en la imagen; según la definición de procesamiento digital de imágenes antes presentada encontramos al reconocimiento de objetos u regiones como un tema común entre la visión por computador y el procesamiento digital de imágenes, lo anterior teniendo en cuenta que varias aplicaciones de visión por computador tratan de emular la visión humana llegando a tener que tomar decisiones basadas en la información adquirida, el anterior es un caso particular de la inteligencia artificial la cual busca emular la inteligencia humana.

El tratamiento digital de imágenes presenta en la literatura técnicas de reconocimiento de las regiones u objetos, en [1] se presentan técnicas agrupadas en dos grupos: teoría de decisión y estructural, basadas en descriptores cuantitativos y cualitativos respectivamente.

En este trabajo, teniendo en cuenta la necesidad de implementación del proceso con un método eficiente y sencillo sin mayores requerimientos de cómputo, se empleó una versión modificada del método de correlación [1] usado para reconocimiento de regiones u objetos, el método de la correlación entre dos imágenes representadas por la funciones f y w respectivamente se define::

$$c(x,y) = \sum_s \sum_t f(s,t)w(x+s, y+t) \qquad (11)$$

Como resultado de aplicar el método de la correlación entre las imágenes f y w obtenemos una tercera imagen c que representa la correlación entre estas dos imágenes. La forma de saber dentro de la imagen de interés f en que lugar se localiza el patrón u objeto definido por w, es el lugar de la imagen c donde se presenta el mayor valor de correlación $c(x,y)$.

Cuando aplicamos el método de correlación a imágenes binarias, la correlación nos arroja el valor de aciertos o coincidencias de pixeles entre la imagen patrón w y la imagen sobre análisis f.

Como se mencionó anteriormente, en este trabajo se presenta un método que de aquí en adelante en el libro se denominara el método de correlación modificado que es el resultado de la conceptualización y análisis del método de correlación presentado brevemente en este capitulo y empleado en la literatura para el reconocimiento de regiones u objetos de una imagen.

A diferencia del método de correlación, el método empleado necesita que la imagen a la cual se le aplique el método contenga únicamente la región que se desea reconocer, ejemplos de este tipo de imágenes se presentan en cada una de las seis regiones de la figura 24.

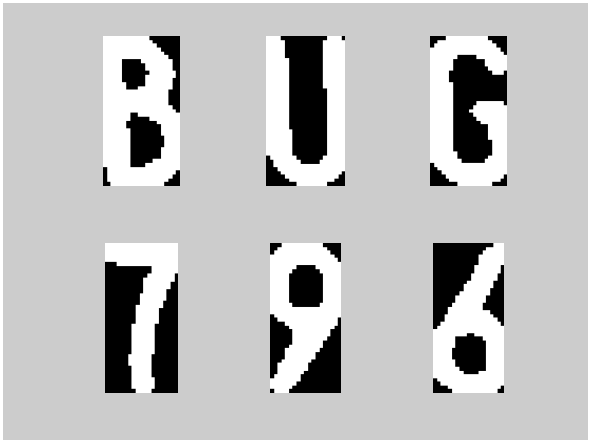


FIGURA 24. Segmentación de las seis regiones a reconocer.
Fuente Autores.

Como siguiente paso después de la segmentación de la región de interés, se encuentra la creación de las máscaras, un grupo de máscaras para cada región, donde cada grupo contiene una máscara para cada posible carácter presente en la placa del vehículo; las dimensiones cada máscara presente en el grupo son iguales a las de la imagen que contiene la región a clasificar; dimensiones que varían de una región a otra.

El reconocimiento de los caracteres se logra comparando el valor de los pixeles de la mascara con en valor de los píxeles de la región de interés.

Las máscaras de caracteres mencionadas anteriormente las conforman una serie de píxeles cuyo valor de intensidad y ubicación dependen del caracter que se desee reconocer, en la figura 25 un caso particular cuya región de interés a identificar hace referencia a la letra B, y en donde los píxeles en color rojo[20] conforman la máscara para esta región.

[20] color empleado solo para diferencia visual del lector

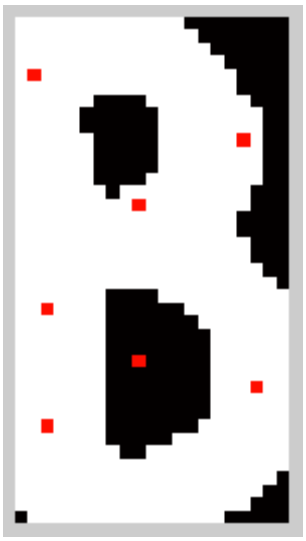


FIGURA 25. Región a reconocer y mascara empleada.
Fuente Autores.

Las coordenadas en las cuales se localizan cada uno de los píxeles de la máscara se calculan como una ponderación de las dimensiones de la imagen, dichos valores de ponderación son el resultado de un promedio realizado a imágenes cuya región a identificar representaban el mismo carácter.

6. PRESENTACIÓN Y ANÁLISIS DE RESULTADOS

Al efectuar cualquier tipo de trabajo de investigación o de desarrollo tecnológico, un aspecto que se debe tener en cuenta, es la presentación de los resultados, ya que permiten la evaluación, cuantitativa, de la calidad, eficiencia y robustez del trabajo y la metodología que se implemento. El sistema fué evaluado, con un conjunto de 150 imágenes de vehículo placas vehículo.

Para poder encontrar las falencias del sistema y qué fases presentan mayor vulnerabilidad, se realizaron las pruebas correspondientes una vez finalizada cada una de las fases principales, segmentación matrícula, segmentación letras, e identificación de estas. Por lo que las pruebas se presentan en fases por orden, y luego los resultados globales.

Para poder presentar los resultados, vamos a realizar un tipo de clasificación de las imágenes adquiridas. Las subdivisiones de las imágenes son:

- **Imágenes Defectuosas:** son imágenes que presentan dificultad para segmentar las letras a simple vista, puede ser por deterioro de la placa, o por falta de color, por la aparición de objetos no deseados, como los tornillos, marcos, etc.
- **Imágenes Oscuras:** imágenes con problemas de iluminación, dificultando el proceso.
- **Imágenes normales:** en este grupo se clasifican las imágenes que no presentan ningún tipo de problemas, como las anteriores.

(A)

(B)

(C)

FIGURA 26. Tipos de imágenes adquiridas.
(A). Imagen defectuosa. (B). Imagen Oscura. (C). Imagen normal.
Fuente Autores.

6.1. SEGMENTACIÓN DE LA PLACA.

Una vez desarrollado el algoritmo para esta fase del proyecto, se prosiguió a realizar las pruebas concernientes para así poder medir, de alguna manera, la eficiencia del método hasta ese momento.

Se presentaron muchos problemas, referente a efectos no deseados que son introducidos por algunos elementos, como lo son las hojas en el suelo,

calcomanías. Otro problema que se presentó, referente al tema, es la segmentación de las placas en los carros rojos o amarillos, ya que a la hora de realizar el cambio de espacios, el rojo alcanza a notarse en la nueva imagen, como se observa:

FIGURA 27. Transformación Espacio Carro Rojo.
Fuente Autores.

En general, los resultados de esta fase, son óptimos, las imágenes tratadas, así fuesen imágenes defectuosas, imagen oscura o imagen normal, el 100 % de ellas segmentaron la matricula, así hubiera algún tipo de ruido, o si el automóvil es de color rojo. Este porcentaje se calcula sobre una muestra de 150 imágenes, entre normales, oscuras y defectuosas.

6.2. SEGMENTACIÓN DE CARACTERES.

6.2.1. Separar caracteres de la Placa

La separación de los caracteres de la placa, una vez haya sido segmentada la placa, es otra fase a tener en consideración y decisiva en los resultados a obtenerse.

Para esta fase las pruebas se realizaron a las imágenes en las que la placa fue bien segmentada, o sea, todas de ellas.

Dependiendo de cómo sea la segmentación, los resultados se pueden clasificar en:

- **Buena:** cuando las seis letras logran ser separados sin alterar la estructura de su forma.
- **Regular:** es cuando es segmentada con algún tipo de impureza o de ruido que se encuentra presente en la imagen, como son tornillos, los marcos que algunas placas tienen en el carro. Las consecuencias de este problema, se ven reflejados a la hora de segmentar las letras definitivamente.
- **Mala:** entran este grupo, las placas en donde los caracteres están muy mal definidos, debido a la malformación de la placa, por la pintura de esta misma.

(A) (B)

(C)

FIGURA 28. Casos de separación de los caracteres.
(A). Segmentación Buena. (B).Segmentación Regular. (C). Segmentación mala.
Fuente Autores.

Los errores obtenidos, fueron minimizados en la forma como se binarizó la imagen, por secciones, o sea, dividida la imagen y binarizada. Con este método se redujo considerablemente la unión de letras, la aparición de los tornillos, aunque en algunos casos no fue muy óptimo el resultado.

El uso de la morfología puede mejorar los resultados, aunque en determinados casos la segmentación resulta imposible, como puede ser en la figura 28C. Teniendo en cuenta que si se aplica algún tipo de operación, como una dilatación, para unir los espacios separados, esta dilatación puede destruir las demás letras, llagándolas a unir, lo cual es una limitante bastante considerable.

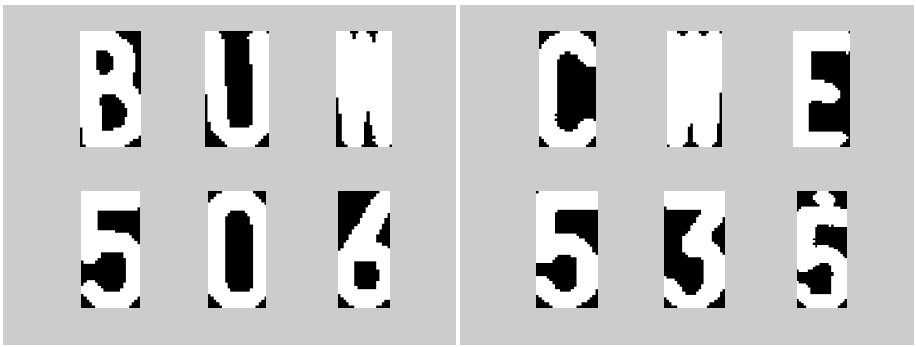
Es esta etapa en donde se presentan la mayor parte de errores. Por que de esta fase depende de cómo sean los resultados de la siguiente fase, la de segmentar letra por letra y luego identificarlas. De las 150 imágenes tratadas, se obtuvo como resultado de la segmentacion del grupo de caracteres como la mostrada en la figura 28, buenas, el 92%, regulares, 4 %, y malas, el 4% del total de las imágenes cuya placa quedo bien segmentadas en la fase anterior. Las tablas completas se muestran en el Anexo A.

6.2.2. Segmentación Caracteres.

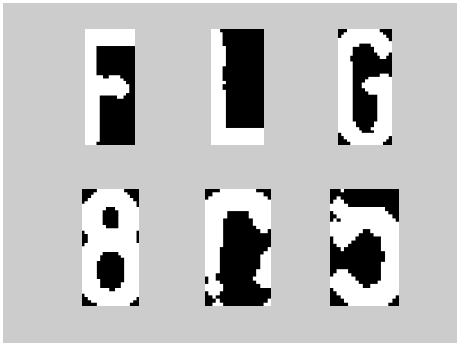
La segmentación de los caracteres, uno por uno, se puede clasificar de la forma en como sea separado cada uno de ellos.

- **Buenas:** en la que las seis letras son segmentadas perfectas.
- **Regular:** donde alguna de las seis letras no es bien segmentada.
- **Mala:** más de un carácter esta mal segmentado.

En esta fase, la mayoría de los errores son consecuencia debida de una segmentación defectuosa en la etapa anterior, como se ve en la figura 29C, donde es la segmentación de las letras de la Figura 28C.



(A) (B)



(C)

FIGURA 29. Casos segmentación Caracteres.
(A) Buena. (B) Regular. (C) Mala.
Fuente Autores.

Del total de las imágenes, el 82% de ellas el resultado es buena la segmentación decada uno de los caracteres, el 9.33 % es regular la segmentación, es decir, 5 de los seis caracteres fueron bien segmentados, y el 8.67% restante es mala la segmentación.

6.3. RESULTADOS TARJETA

Para la implementación de nuestro código propuesto como solución a la hora de identificar la placa, se implementó un código en leguaje C++ (Anexo C, Programación Identificación de las Placas (C++)), compatible con las librerías preinstaladas en Linux, ya que se presentaron inconvenientes en instalar librerías ajenas a las que contiene por defecto este sistema operativo, razón por la cual dificultó utilizar el paquete de librerías de OpenCv. Ya teniendo un código con los comandos ofrecidos por esta librería, se vio la necesidad de realizar el cambio del código ya realizado en OpenCv, a un código que hace uso de los comandos del lenguaje básico de C++ presentado en el Anexo C (Programación Identificación de las Placas (C++)).

Ya ejecutándose a cabalidad el código en un PC de escritorio, con el sistema operativo Linux/GNU instalado, se prosiguió a realizar el montaje respectivo en la tarjeta:

Primero se hace el traslado del código (archivo .cpp), y las imágenes adquiridas en formato de texto a un Pendrive, para así poder ejecutar dicho código desde la tarjeta ECB_AT91, que es la finalidad de este trabajo.

Los resultados al ejecutar el archivo se presenta en el Anexo B.

CONCLUSIONES Y RECOMENDACIONES.

- La implementación de un sistema LPR en un sistema embebido, posee ventajas como desventajas a la hora de ponerlo en práctica, dependiendo de las especificaciones técnicas de los instrumentos que se utilicen para su implementación. Como ventajas se tiene: su precio, su portabilidad, la instalación de este en cualquier ambiente; dentro de las desventajas se tiene la capacidad de procesamiento de los datos, ya que las especificaciones técnicas de los sistemas embebidos son frecuentemente inferiores a las que posee un PC de escritorio, exigiendo al programador soluciones mas eficientes.

- Las operaciones básicas en el tratamiento de Imágenes digitales, como lo son la erosión, la dilatación y la creación del histograma, entre otras, son herramientas basadas en la forma o estructura de los componentes de una imagen y, aparte de presentar fácil implementación, son muy útiles en el estudio, modificación, identificación y eliminación de objetos no deseados en una imagen.

- En una aplicación de tratamiento digital de imágenes un punto de partida importante es la correcta selección del espacio de colores que se piensa emplear; en la aplicación descrita en este trabajo, se presenta un claro ejemplo de cómo el espacio de colores empleado brinda ventajas en el proceso de ubicación de la ROI, facilitando de esta manera la

implementación de los siguientes pasos propuestos en la metodología desarrollada en este trabajo.

- El proceso de segmentación, se puede implementar desde diferentes perspectivas, en este trabajo se emplea el método de las proyecciones para realizar la segmentación tanto de la placa como de los caracteres de la misma, tomando como criterio de selección la geometría de las regiones a segmentar y el resultado de aplicar métodos estadísticos a las mismas, obteniendo un proceso segmentación automática.

- La plataforma ECB_AT91, sobre la cual se implementó el código propuesto en el presente trabajo, al permitir la ejecución de Linux en forma nativa, facilita su uso en cierta forma, pues Linux es un sistema operativo que cada día presenta más adeptos. En cuanto al desempeño de la plataforma en esta aplicación en particular, podemos decir que su desempeño fue aceptable teniendo como baso los tiempos de ejecución del código, tiempo en promedio cercano a los 2.8 segundos, el cual es aceptable para aplicaciones de LPR.

Recomendaciones de los autores para un trabajo futuro.

- Para el desarrollo total del sistema, se recomienda realizar el montaje completo en una versión mas reciente de la tarjeta ECB_AT91 V1, como puede ser la ECB_AT91 V2 o la ECBOT, productos de la misma empresa, las cuales son compatibles con la tarjeta utilizada en este trabajo, presentando complementos de hardware para usos mas avanzados: El mas representativo de estos componentes es una FPGA Spartan3 (XC3S400),

la cual facilita la captura de la imagen y añade la posibilidad del uso de un LCD para visualización de resultados del proceso.

- Mejorar la metodología planteada en este trabajo, en lo referente al proceso de ubicación de la placa del vehículo, reduciendo o eliminando la dependencia de conocer una aproximación del área de la placa del vehículo en la imagen de análisis; logrando con esto disminuir los requerimientos de control en la captura de la imagen, como lo son la distancia, el ángulo y la altura entre el vehículo y el dispositivo de captura. Así mismo, se recomienda la implementación de un sistema más robusto y eficiente para el reconocimiento de los caracteres presentes en la placa del vehículo.

- Ensamblar la plataforma y los demás componentes del proceso de captura, en un producto con una expresión creativa basada en lo estético, cumpliendo así con los requerimientos para la comercialización del sistema, y poseer con esto una apariencia agradable.

BIBLIOGRAFÍA

[1] SEGMENTACIÓN DE IMÁGENES TERMOGRÁFICAS DE GLÁNDULAS MAMARIAS UTILIZANDO TÉCNICAS DE DETECCIÓN DE DISCONTINUIDADES. **REINALDO JAIMES QUINTERO, JORGE ENRIQUE OSORIO ABAUNZA**. 2008. UNIVERSIDAD INDUSTRIAL DE SANTANDER.

[2] DIGITAL IMAGE PROCESSING SECOND EDITION, **RAFAEL C. GONZALEZ** UNIVERSITY OF TENNESSEE **RICHARD E. WOODS**, PRENTICE HALL.

[3] [Online]http://www.dfmf.uned.es/actividades/no_reglada/laboratorio/segmentacion1.pdf

[4] [Online] http://www.cimat.mx/eventos/pi08/p3.pdf

[5] INTRODUCCIÓN AL PROCESAMIENTO Y ANÁLISIS DE IMÁGENES DIGITALES, **R MOLINA**.

[6] DESARROLLO EN MATLAB DE UN ALGORITMO PARA SOLUCIONAR ESPACIOS DE COLOR NO LINEALES. **JESÚS ANTONIO VEGA URIBE**

[7] MEJORAS Y ALTERNATIVAS AL MÉTODO DE ESTANDARIZACIÓN DE COLOR EN IMÁGENES DIGITALES BAJO PATRONES DE ILUMINACIÓN CAMBIANTE. **ZULMA MILENE GARZÓN BOLÍVAR**

[8] DETECCIÓN DE BORDES EN IMÁGENES A COLOR EN REPRESENTACIÓN DE HERING CON APLICACIONES EN COMPRESIÓN. **ING. EDGAR J. PEÑUELA N**.

[9] [Online] http://en.wikipedia.org/wiki/color

[10] [Online] http://dis.um.es/~ginesgm/files/doc/pav/tema5.pdf

[11] [Online] http://hyperphysics.phy-astr.gsu.edu/hbase/vision/cie.html

[12] [Online] http://en.wikipedia.org/wiki/hsv_color_space

[13] [Online]http://ocwus.us.es/pintura/usos-plasticos-del-color/temario/temas2_imswct/page_11.htm

[14] DETECCIÓN DE OBJETOS POR SEGMENTACIÓN MULTINIVEL COMBINADA DE ESPACIOS DE COLOR. **P.GIL, F.TORRES,F.G.ORTIZ**. UNIVERSIDAD DE ALICANTE

[15] DETERMINACIÓN DE DESCRIPTORES PARA LA CARACTERIZACIÓN DE IMÁGENES DE MÚLTIPLES GALAXIAS UTILIZANDO WAVELETS. **JUAN CARLOS BASTO PINEDA.** UNIVERSIDAD INDUSTRIAL DE SANTANDER.

[16] A RECURSIVE UMBRALIZACIÓN TECHNIQUE FOR IMAGE SEGMENTATION. **CHERIET**

[17] PROBABILIDAD Y ESTADÍSTICA APLICADAS A LA INGENIERÍA. . **DOUGLAS C. MONTGOMERY, GEORGE C. RUNGER.** MÉXICO. MC GRAW HILL. 1996

[18] ALGORITHMIC AND MATHEMATICAL PRINCIPLES OF AUTOMATIC NUMBER PLATE RECOGNITION SYSTEMS.

[19] SPARK FUN CÁMARA MANUAL.

[20] [Online] http://wiki.emqbit.com/free-ecb-at91

ANEXO A

TABLAS DE RESULTADOS

Resultados Segmentación de la Placa.

IMÁGENES DE PRUEBA							
Grupo	**Imágenes**	**Buenas**	**% Buenas**	**Regular**	**% Regular**	**Malas**	**% Malas**
Normales	128	128	100	0	0	0	0
Defectuosas	10	10	100	0	0	0	0
Oscuras	12	12	100	0	0	0	0

Resultados Separación de los Caracteres de la Placa.

IMÁGENES DE PRUEBA							
Grupo	**Imágenes**	**Buenas**	**% Buenas**	**Regular**	**% Regular**	**Malas**	**% Malas**
Normales	128	128	100	0	0	0	0
Defectuosas	10	7	70	2	20	1	10
Oscuras	12	3	25	4	33.33	5	41.66
Totales	150	138	92	6	4	6	4

Resultados Segmentación de los 6 Caracteres.

IMÁGENES DE PRUEBA									
Grupo	**Imágenes**	**Separación caracteres**		**Buenas**	**% Buenas**	**Regular**	**% Regular**	**Malas**	**% Malas**
Normales	128	Buenas	128	115	89.84	8	6.25	5	3.9
		Regulares	0	0	0	0	0	0	0
		Malas	0	0	0	0	0	0	0
Defectuosas	10	Buenas	7	3	42.85	3	42.85	1	14.28
		Regulares	2	0	0	0	0	2	100
		Malas	1	0	0	0	0	1	100
Oscuras	12	Buenas	3	2	66.66	0	0	1	33.33
		Regulares	4	3	75	1	25	0	0
		Malas	5	0	0	2	40	3	60
Totales	150		150	123	82	14	9.33	13	8.67

Resultados Reconocimiento de los Caracteres.

Tomando los resultados de las anteriores tablas, el total de las letras bien segmentadas, con base en solo las buenas y regulares, es de 808 caracteres bien segmentados.

IMÁGENES DE PRUEBA										
Grupo	**Imágenes**	**Separación caracteres**		**Buenas**	**% Buenas**	**Regular**	**% Regular**	**Total letras**	**Letras reconocidas**	**%**
Normales	128	Buenas	128	115	89.84	8	6.25	730	648	88.76
		Regulares	0	0	0	0	0	0		
		Malas	0	0	0	0	0	0		
Defectuosas	10	Buenas	7	3	42.85	3	42.85	33	28	84.84
		Regulares	2	0	0	0	0	0		
		Malas	1	0	0	0	0	0		
Oscuras	12	Buenas	3	2	66.66	0	0	12	8	66.66
		Regulares	4	3	75	1	25	23	15	65.21
		Malas	5	0	0	2	40	10	9	90
Totales	150		150	123		14		808	708	87.62

ANEXO B.

RESULTADOS TARJETA.

Imagen a procesar.

Figura C1. Imagen de Prueba1. Fuente Autores.

Resultado Obtenido

```
                    UNIVERSIDAD INDUSTRIAL DE SANTANDER

                              TRABAJO DE GRADO:
        RECONOCIMIENTO DE PLACAS DE AUTOMOVIL USANDO LA PLATAFORMA ECB_AT91.

                                 AUTORES:
              SERGIO IVAN MERCHAN MEJIA - LUIS MIGUEL ARIZA FLOREZ

                              BUCARAMANGA 2009


La fecha y hora de ingreso del vehiculo son: Mon Jan 19 16:28:28 2009


La Placa del Vehiculo:  BSS 211
Tiempo de ejecucion: 2.69 seconds
-
```

Figura C2. Imagen resultado Prueba2. Fuente Autores.

Imagen a procesar

Figura C3. Imagen de Prueba2. Fuente Autores.

Resultado Obtenido.

```
                    UNIVERSIDAD INDUSTRIAL DE SANTANDER

                              TRABAJO DE GRADO:
RECONOCIMIENTO DE PLACAS DE AUTOMOVIL USANDO LA PLATAFORMA ECB_AT91.

                                  AUTORES:
            SERGIO IVAN MERCHAN MEJIA - LUIS MIGUEL ARIZA FLOREZ

                              BUCARAMANGA 2009


La fecha y hora de ingreso del vehiculo son: Mon Jan 19 16:36:20 2009


La Placa del Vehiculo:  BUY 277
Tiempo de ejecucion: 2.57 seconds
-
```

Figura C4. Imagen resultado Prueba2. Fuente Autores.

ANEXO C

CÓDIGOS DE PROGRAMACIÓN

Programación captura de la Imagen (VHDL).

Proyecto _Imagen_Serial

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Proyecto_Imagen_seial is
      Port (      clk               : in  STD_LOGIC;
                  clkcamara         : out  STD_LOGIC:='0';
                  p_clk             : in  STD_LOGIC;
                  pixeles           : in  STD_LOGIC_VECTOR (7 downto 0);
                  Vsync             : in  STD_LOGIC;
                  Hsync             : in  STD_LOGIC;
                  tomar_foto        : in  STD_LOGIC;
                  Pixel_serie       : out  STD_LOGIC:='1');
end Proyecto_Imagen_seial;

architecture Behavioral of Proyecto_Imagen_seial is

COMPONENT Captura_imagen

      Port (      clk               : in  STD_LOGIC;
                  clk_camara        : in  STD_LOGIC;
                  Pixeles           : in  STD_LOGIC_VECTOR (7 downto 0);
                  Vsync             : in  STD_LOGIC;
                  Hsync             : in  STD_LOGIC;
                  tomar_foto        : in  STD_LOGIC;
                  pixel_out         : out STD_LOGIC_VECTOR (7 downto 0);
                  cargar_dato       : out Std_logic);
end component;

COMPONENT ClK_Camara
      Port (      clk               : in  STD_LOGIC;
```

```
                clock_camara      : out  STD_LOGIC);
end component;

Component UART is
      Port (      clk         : in  STD_LOGIC;
                  carga       : in  STD_LOGIC;
                  DATO        : in  STD_LOGIC_VECTOR (7 downto 0);
                  tx          : out STD_LOGIC  );
end component;


signal clock_camara_1 : std_logic;
signal DATO_1: STD_LOGIC_VECTOR (7 downto 0);
signal Carga_dato_1 :Std_logic;

begin
      clkcamara<=clock_camara_1;

      Inst_Captura_imagen: Captura_imagen PORT MAP(
            clk_Camara  => p_clk,
            clk         => clk,
            Pixeles     => Pixeles,
            Vsync       => Vsync,
            Hsync       => HSync,
            tomar_foto  => Tomar_foto,
            pixel_out   => DATO_1,
            cargar_dato => carga_dato_1);

      Inst_ClK_Camara: ClK_Camara PORT MAP(
            clk => clk,
            clock_camara => clock_camara_1);

      Inst_UART: UART PORT MAP(
            clk         =>    clk,
            carga       =>    carga_dato_1,
            DATO        =>    DATO_1,
            tx          =>    Pixel_serie  );
end Behavioral;
```

Captura_imagen (VHDL)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Captura_imagen is
      Port (      clk              : in  STD_LOGIC;
```

```
                clk_Camara  : in  STD_LOGIC;
                Pixeles     : in  STD_LOGIC_VECTOR (7 downto 0);
                Vsync       : in  STD_LOGIC;
                Hsync       : in  STD_LOGIC;
                tomar_foto  : in  STD_LOGIC;
                pixel_out   : out STD_LOGIC_VECTOR (7 downto 0):=(others
=>'0');
                      cargar_dato : out Std_logic:='0');
end Captura_imagen;

architecture Behavioral of Captura_imagen is
      type IG_STATE_TYPE is ( IDLE, WAIT_FV, CAP_FV, CAP_LV,CAP_LDATA);
      signal ig_prs_state, ig_nxt_state : IG_STATE_TYPE;
      signal pixel_out_int : STD_LOGIC_VECTOR (7 downto 0);
      signal Hv_delay, Hv_delay2, Hv_edge_detect,fv_delay, fv_delay2,
fv_edge_detect,tomar_foto_delay, tomar_foto_delay2, tomar_foto_detect:
STD_LOGIC;


begin

fv_edge_detect <= '1' when (fv_delay = '1' and fv_delay2 = '0') else '0';
Hv_edge_detect <= '1' when (Hv_delay = '0' and Hv_delay2 = '0') else '0';
tomar_foto_detect <= '1' when (tomar_foto_delay = '1' and
tomar_foto_delay2 = '0') else '0';
pixel_out<=pixel_out_int;

      process (clk_Camara,clk)
      begin
            if clk_Camara'event and clk_Camara = '0' then
                  if (ig_prs_state = CAP_LDATA) then
                        pixel_out_int <= Pixeles;
                  elsif (ig_prs_state = CAP_FV or ig_prs_state =
idle)then
                        pixel_out_int <=(others => '0');
                  end if;
            end if;
      end process;

      FV_EDGE_DET: process (clk_Camara)
      begin
            if clk_Camara'event and clk_Camara = '1' then
                  fv_delay <= Vsync;
                  fv_delay2 <= fv_delay;
            end if;
      end process FV_EDGE_DET;

      HV_EDGE_DET: process (clk_Camara)
      begin
            if clk_Camara'event and clk_Camara = '1' then
                  Hv_delay <= Hsync;
                  Hv_delay2 <= Hv_delay;
            end if;
      end process HV_EDGE_DET;
```

```
      TOMAR_FOTO_DET: process (clk_Camara)
      begin
            if clk_Camara'event and clk_Camara = '1' then
                  tomar_foto_delay <= tomar_foto;
                  tomar_foto_delay2 <= tomar_foto_delay;
            end if;
      end process TOMAR_FOTO_DET;

      IG_COM: process (clk_Camara,clk)
      begin

            case ig_nxt_state is

                  ---------------------- IDLE State ---------------------
                  when IDLE =>

                        if (tomar_foto_detect = '1') then
                              ig_nxt_state <= WAIT_FV;
                        end if;
                  ---------------------- WAIT_FV State ------------------
--
                  when WAIT_FV =>
                        if (fv_edge_detect = '1') then
                              ig_nxt_state <= CAP_FV;
                        end if;
                  ---------------------- CAP_FV State -------------------
-
                  when CAP_FV =>

                        if (fv_edge_detect = '1') then
                              ig_nxt_state <= idle;
                        end if;
                        if (Hsync = '1') then
                              ig_nxt_state <= CAP_LDATA;
                        end if;
                  ------------------- CAP_LDATA State ------------------
--
                  when CAP_LDATA =>

                        if (fv_edge_detect = '1') then
                              ig_nxt_state <= idle;
                        end if;

                        if (Hv_edge_detect ='1') then
                              ig_nxt_state <= CAP_FV;
                              cargar_dato <= '0';
                        else cargar_dato <= clk_camara;
                        end if;

                  when others =>
                        ig_nxt_state <= IDLE;

            end case;
```

```
      end process IG_COM;
end Behavioral;
```

CLK_CAMARA (VHDL)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ClK_Camara is
    Port ( clk : in  STD_LOGIC;
           clock_camara : out  STD_LOGIC:='0');
end ClK_Camara;

architecture Behavioral of ClK_Camara is
SIGNAL A:STD_LOGIC_VECTOR (15 downto 0):=(others => '0');

begin
Sinc: PROCESS(clk)
      BEGIN
      IF (clk'EVENT) AND (clk='1') THEN
            A<=A+1;
            IF A<=2500 THEN
                  clock_camara<='1';
                  ELSIF A>=5000 THEN
                  A<=(others =>'0');
                  else clock_camara<='0';
            END IF;
      END IF;
 END PROCESS sinc;

end Behavioral;
```

UART (VHDL)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity UART is
    Port ( clk          : in  STD_LOGIC;
           carga        : in  STD_LOGIC;
           DATO         : in  STD_LOGIC_VECTOR (7 downto 0);
           tx                 : out STD_LOGIC:='0');
end UART;

architecture Behavioral of UART is
      COMPONENT BAUDGEN
```

```
      PORT(
            CLK : IN std_logic;
            TxCLK : OUT std_logic         );
      END COMPONENT;

      COMPONENT transmite
      PORT(
            CLK         : IN std_logic;
            TxCLK       : IN std_logic;
            CARGA       : IN std_logic;
            DATO        : IN std_logic_vector(7 downto 0);
            Tx_OUT      : OUT std_logic);
      END COMPONENT;

    signal reloj : std_logic;

begin

      Inst_BAUDGEN: BAUDGEN PORT MAP(
            CLK     => clk,
            TxCLK   => reloj  );


      Inst_transmite: transmite PORT MAP(
            CLK     => clk,
            TxCLK   => reloj,
            CARGA   => carga,
            DATO    => dato,
            Tx_OUT  => tx);

end Behavioral;
```

BAUDGEN (VHDL)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity BAUDGEN is
    port (  CLK   : in STD_LOGIC;
                        TxCLK : out STD_LOGIC
                  );
end BAUDGEN;

architecture BAUDGEN_arch of BAUDGEN is
      SIGNAL A:STD_LOGIC_VECTOR (19 downto 0):=(others => '0');

begin
Sinc: PROCESS(clk)
                  BEGIN
                        IF (clk'EVENT) AND (clk='1') THEN
                              A<=A+1;
                                    IF A<=217 THEN
                                          txclk<='1';
```

```
                                    ELSIF A>=434 THEN
                                          A<=(others =>'0');
                                    else txclk<='0';
                                    END IF;
                        end if;
            END PROCESS sinc;
end BAUDGEN_arch;
```

Transmite (VHDL)

```
library IEEE;
use IEEE.std_logic_1164.all;

entity transmite is
      port (
            clk                     : in  STD_LOGIC;
            TxCLK,CARGA       : in  STD_LOGIC;
            DATO                    : in  STD_LOGIC_VECTOR (7 downto 0);
            Tx_OUT                  : out STD_LOGIC
            );
end transmite;

architecture tx_arch of transmite is
      type estados is (IDLE,ESPERA,ENVIA);
      signal estado     : estados;
      signal Tx_Buf     : STD_LOGIC_VECTOR(9 downto 0);
      signal txclk_delay, txclk_delay2, txclk_edge_detect: STD_LOGIC;
      signal carga_delay, carga_delay2, carga_edge_detect: STD_LOGIC;


begin


      txclk_edge_detect <= '1' when (txclk_delay = '1' and txclk_delay2 =
'0') else '0';
      carga_edge_detect <= '1' when (carga_delay = '0' and carga_delay2 =
'1') else '0';

      txclk_EDGE_DET: process (clk)
      begin
            if clk'event and clk = '1' then
                  txclk_delay <= txclk;
                  txclk_delay2 <= txclk_delay;
            end if;
      end process txclk_EDGE_DET;

      carga_EDGE_DET: process (clk)
      begin
            if clk'event and clk = '1' then
                  carga_delay <= carga;
                  carga_delay2 <= carga_delay;
            end if;
      end process carga_EDGE_DET;
```

```
	process (txclk,CLK)
		variable    bitTx : integer range 10 downto 0:=0;
	begin
		if (CLK'EVENT and CLK='1') then

			case estado is

				when IDLE=>
					Tx_OUT            <= '1';
					Tx_Buf(9)   <= '1';
					Tx_Buf(0)   <= '0';
					If carga_edge_detect='1' then
						Tx_Buf(8 downto 1)<= DATO;
						estado      <=ENVIA;
					else
						Tx_Buf(8 downto 1)<= "00000000";
						estado      <= IDLE;
					end if;

				when ESPERA=>
					if txclk_edge_detect='1' then
						estado      <= ENVIA;
					else
						estado      <= ESPERA;
					end if;

				when ENVIA=>
					if txclk_edge_detect='1' then
						Tx_OUT      <=Tx_Buf(bitTx);
						if bitTx=9 then
							estado      <=IDLE;
							bitTx :=0;
						else
							bitTx :=bitTx+1;
						end if;
					end if;
			end case;
		end if;
	end process;
end tx_arch;
```

Programación Identificación de las Placas (C++).

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>
#include <sys/time.h>
```

```
using namespace std;


double timeval_diff(struct timeval *a, struct timeval *b)
{
      return
            (double)(a->tv_sec + (double)a->tv_usec/1000000) -
            (double)(b->tv_sec + (double)b->tv_usec/1000000);

}

int main(int argc, char *argv[])
{
system("clear");
////////////////////////////////////////////////////////////////////

printf ("\n\t\t\t UNIVERSIDAD INDUSTRIAL DE SANTANDER \n\n");
printf ("\t\t\t          TRABAJO DE GRADO: \n");
printf  ("\t RECONOCIMIENTO DE PLACAS DE AUTOMOVIL USANDO LA PLATAFORMA
ECB_AT91. \n\n");
printf ("\t\t\t               AUTORES: \n");
printf ("\t         SERGIO IVAN MERCHAN MEJIA - LUIS MIGUEL ARIZA FLOREZ
\n\n");
printf ("\t\t                    BUCARAMANGA 2009 \n\n\n");


  time_t rawtime;
  time ( &rawtime );
  printf ( "La fecha y hora de ingreso del vehiculo son: %s \n", ctime
(&rawtime) );


struct timeval t_ini, t_fin;
double secs;
gettimeofday(&t_ini,NULL);
int ini, fin, t;




int IMX=640;
int IMY=480;
int YMIN,YMAX,XMIN,XMAX;

YMIN=(IMY/10)*2-1;
YMAX=IMY-(IMY/10)*2-1;
XMIN=(IMX/10)*2-1;
XMAX=IMX-(IMX/10)*2-1;

int m,n,mm,sumhis,pos,umb;
int suma = 0;
int x,i,jj;
short int ii,j,xmax,ymax;
int IM1[289][385];
```

```
int b[111265];
ifstream inFile;

////////////////////////////////////////////////////////////////////////
/

inFile.open("carro_(1).txt");
if (!inFile)
{
 cout << "Unable to open file\n"; //tribunal << "No se puede abrir el
archivo";
 exit(1); // terminate with error salida (1); / / terminar con el error
}
i=0;
jj=0;

while (inFile >> x)
{
      b[i]=x;
      i++;

}
inFile.close();
/*
jj=1;
for (m=YMIN; m<=YMAX; m++)
{
      for (n=XMIN; n<=XMAX; n++)
      {
            IM1[jj][ii]=(R[m][n]*G[m][n]/(255^2))*((R[m][n]+G[m][n])/2-
B[m][n])-(B[m][n]/255)*(B[m][n]-(R[m][n]+G[m][n])/2);
              if (IM1[jj][ii]<0)
            {
                        IM1[jj][ii]=0;
            }
              if (IM1[jj][ii]>255)
            {
                  IM1[jj][ii]=255;
            }
            xmax=ii;
            ii++;
      }
      ii=2;
      jj++;
}
*/


for (i=0;i<289;i++)
{
      for (j=0;j<385;j++)
      {
            IM1[i][j]=b[jj];
            jj++;
```

```
      }
}
ymax=289;
xmax=385;

int his[256]={0};

for (m=0;m<ymax;m++)
{
      for (n=0;n<xmax;n++)
      {
            pos=(int)IM1[m][n];
            his[pos]=his[pos]+1;
    }
}

i=0;
sumhis=0;
mm=255;
while (i==0)
{
      sumhis=sumhis+his[mm];
            if (sumhis>7000)
            {
            i=1;
                  umb=mm;

            }
      mm--;
}

bool IM[ymax][xmax];
for (m=0;m<ymax;m++)
{
      for (n=0;n<xmax;n++)
      {
            if (IM1[m][n]>umb) IM[m][n]=1;
            else IM[m][n]=0;
      }
}


////////////////////////////////////////////////////////////////////////////
//////////////////////////
ymax=ymax;
xmax=xmax;

bool Cyy[ymax][xmax];
bool Cyy1;
for (m=ymax-10;m<ymax;m++)
{
      for (n=0;n<xmax;n++)
        {
            Cyy[m][n]=0;
```

```
            }
}

for (m=1;m<ymax;m++)
{
      for (n=1;n<xmax;n++)
        {
            Cyy[m][n]=IM[m-1][n-1]*IM[m-1][n]*IM[m-1][n+1]*IM[m][n-
1]*IM[m][n]*IM[m][n+1]*IM[m+1][n-1]*IM[m+1][n]*IM[m+1][n+1];
            }
}


////////////////////////////////////////////////////////////////////////////
////////////////////////////

int sumy,sumYT1[ymax],sumy1;
short int ymin,ybm,Ymax,Ymin;

m=ymax-5;
sumy=0;
sumy1=0;
mm=1;
ymin=0;
while (m>ymin)
{
      for (n=5;n<xmax-4;n++)
        {
            sumy=sumy+(int)Cyy[m][n];
      }
      sumy1=sumy+sumy1;
      sumYT1[m]=sumy;
      if (mm==1 && sumy1>1000)
      {
            ybm=m;
            mm=0;
            if (ybm-80>0)
            ymin=ybm-80;
            else ymin=1;
      }
      sumy=0;
      m=m-1;
}

if (ybm+80<ymax)
    ymax=ybm+80;
else ymax=ymax;


n=ybm;
mm=0;
while (n<ymax && mm==0)
{
```

```
      if (sumYT1[n]<3)
      {
            Ymax=n;
            mm=1;
      }
      n=n+1;
}
mm=0;
n=ybm;
while (n>ymin && mm==0)
{
      if (sumYT1[n]<3)
      {
            Ymin=n;
            mm=1;
      }
    n--;
}

//////////////////////////////////.............PROYECCIONES
X.................. ////////////////////////////////


int sumx,sumXT[Ymax-Ymin],xcontrol,xbm,Xmin,Xmax;
float xs,xbmax;

sumx=0;
xbm=1;
for (n=0;n<xmax;n++)
{
      for (m=Ymin;m<=Ymax;m++)
      {
            sumx=sumx+IM1[m][n];
      }
      sumXT[n]=sumx;
      if (sumXT[n]>sumXT[xbm])
                  xbm=n;
        sumx=0;
}

xbmax=0.20*sumXT[xbm];
j=0;
jj=xmax-1;
i=0;
ii=0;
while (i==0  ||  ii==0)
{
      if (i==0 && sumXT[j]>xbmax)
      {
            Xmin=j;
            i=1;
      }
      if (ii==0 && sumXT[jj]>xbmax)
      {
```

```
            Xmax=jj;
            ii=1;
      }
      j=j+1;
      jj=jj-1;
}

xcontrol=Xmax-Xmin;
xs=0.22;

while (xcontrol > 180)
{
      xbmax=xs*sumXT[xbm];
      j=0;
      jj=xmax;
      i=0;
      ii=0;
      while (i==0  ||  ii==0)
      {
            if (i==0 && sumXT[j]>xbmax)
            {
                  Xmin=j;
                  i=1;
            }
            if (ii==0 & sumXT[jj]>xbmax)
            {
                  Xmax=jj;
                  ii=1;
            }
            j=j+1;
            jj=jj-1;
      }
      xcontrol=Xmax-Xmin;
      xs=xs+0.02;
}


////////////////////////////////////////////////////////////////////////////
//////////////////////

int ymin1,xmin1,pasox,pasoy,sumpixel,sumi,sumj,umbral,umbralmayor;
float UT,Ui,Uj,Wj,Wi,S2,S;

pasox=(int)round((Xmax-Xmin)/4);
pasoy=(int)round((Ymax-Ymin)/2);
ymin1=Ymin;
xmin1=Xmin;
jj=0;


while (jj<8)
{
      sumpixel=0;
      for (m=ymin1;m<=ymin1+pasoy;m++)
```

```
{
      for (n=xmin1;n<=xmin1+pasox;n++)
      {
            sumpixel=IM1[m][n]+sumpixel;

      }
}

S=0;
for (umbral=1;umbral<=200;umbral=umbral+10)
{
      j=0;
      i=0;
      sumi=0;
      sumj=0;

      for (m=ymin1;m<=ymin1+pasoy;m++)
      {
            for (n=xmin1;n<=xmin1+pasox;n++)
            {
                  if (IM1[m][n]<umbral)
                  {
                        sumi=IM1[m][n]+sumi;
                        i=i+1;
                  }
                  else
                  {
                        sumj=IM1[m][n]+sumj;
                        j=j+1;
                  }

                  }
      }
      UT=(float)sumpixel/(i+j);
      Ui=(float)sumi/(i+1);
      Uj=(float)sumj/(j+1);
      Wj=(float)j/(i+j+1);
      Wi=(float)i/(i+j+1);
      S2=(float)Wj*(Uj-UT)*(Uj-UT)+Wi*(Ui-UT)*(Ui-UT);

      if (S2>S)
      {
            S=S2;
            umbralmayor=umbral;

      }
}

for (m=ymin1;m<=ymin1+pasoy;m++)
{
      for (n=xmin1;n<=xmin1+pasox;n++)
      {
            if (IM1[m][n]<umbralmayor)
                  IM1[m][n]=1;
```

```
                else  IM1[m][n]=0;

            }
      }
      xmin1=xmin1+pasox+1;
      if (jj==3 || jj==7)
      {
            ymin1=ymin1+pasoy+1;
            xmin1=Xmin;
      }

      jj=jj+1;

}


////////////////////////////////////////////////////////////////////////


int ybminf,ybmaxf,ybmin,ybmax;

sumy=0;
ybm=(int)(Ymin+round((Ymax-Ymin)/2));
ybminf=ybm;
ybmaxf=ybm;
ybmin=ybm-(int)round(3*(Ymax-ybm)/4);
ybmax=ybm+(int)round(3*(Ymax-ybm)/4);
//clear sumYT

int sumYT[ybmax-ybmin];


for (m=ybmin;m<=ybmax;m++)
{
      for (n=Xmin;n<=Xmax;n++)
      {
            sumy=sumy+IM1[m][n];
      }
      sumYT[m]=sumy;
      sumy=0;
}

for (m=ybm;m<=ybmax;m++)
{
      if (sumYT[m] < sumYT[ybmaxf])
            ybmaxf=m;
}

for (m=ybmin;m<=ybm;m++)
{
      if (sumYT[m]<sumYT[ybminf])
            ybminf=m;
}
```

```
for (m=ybmaxf;m<=ybmaxf+3;m++)
{
      for (n=Xmin;n<=Xmax;n++)
            IM1[m][n]=0;
}


for (m=ybminf-3;m<=ybminf;m++)
{
      for (n=Xmin;n<=Xmax;n++)
            IM1[m][n]=0;
}

////////////////////////////////////////////////////////////////////////


bool X5[ybmaxf-ybminf+1][Xmax-Xmin+1];


ii=0;
jj=0;
for(m=ybminf;m<=ybmaxf;m++)
{
      for (n=Xmin;n<=Xmax;n++)
      {
            X5[ii][jj]=  IM1[m-1][n-1] && IM1[m-1][n] && IM1[m-1][n+1] &&
IM1[m][n-1]  && IM1[m][n]  && IM1[m][n+1]  && IM1[m+1][n-1]  && IM1[m+1][n]
&& IM1[m+1][n+1];
            jj=jj+1;
            xmax=jj;

      }
      jj=0;
      ii=ii+1;
}
ymax=ii;

////////////////////////////////////////////////////////////////////////////
//////////////////


int ancho, paso1[8];

suma=0;
sumx=0;
for (n=0;n<xmax;n++)
{
      for(m=0;m<ymax;m++)
            sumx=sumx+X5[m][n];
      suma=suma+sumx;
    sumXT[n]=sumx;
      sumx=0;
}
```

```
Xmin=0;
for (n=9;n>=0;n--)
{
      if (sumXT[n]<3)
            Xmin=n;
}
Xmax=xmax;
for (n=xmax-11;n<xmax;n++)
{
      if (sumXT[n]<3)
            Xmax=n;
}

ancho=Xmax-Xmin+1;
paso1[0]=Xmin;
paso1[1]=Xmin+(int)round(ancho*0.10);
paso1[2]=Xmin+(int)round(ancho*0.25);
paso1[3]=Xmin+(int)round(ancho*0.4);
paso1[4]=Xmin+(int)round(ancho*0.5);
paso1[5]=Xmin+(int)round(ancho*0.6);
paso1[6]=Xmin+(int)round(ancho*0.73);
paso1[7]=Xmin+(int)round(ancho*0.9);//0.9
paso1[8]=Xmax;

int M[7]={0},O[7]={0},ymaxL[7]={0},yminL[7]={0},mayor,iii;

mm=1;
m=paso1[mm];

while (mm<8)
{
    for (n=paso1[mm];n<=paso1[mm+1];n++)
      {
            if (sumXT[n]<sumXT[m])
                  m=n-1;
        }
      M[mm-1]=m;
      mm=mm+1;
      m=paso1[mm];
}


mm=7;
m=paso1[mm];
ii=6;
while (mm>0)
{
      for (n=paso1[mm];n>=paso1[mm-1];n--)
      {
               if (sumXT[n]<sumXT[m])
                        m=n;
        }
    O[ii]=m+1;
```

```
    ii=ii-1;
    mm=mm-1;
    m=paso1[mm];
}

mayor=1;
for (j=0;j<=6;j++)
{
      ii=M[j]-O[j];
      if (ii>mayor)
            mayor=ii;
}

if (M[0]-O[0]<15)
{
      O[0]=M[0]-mayor;
      if (O[0]<1)
            O[0]=1;

}

ii=0;
jj=0;
iii=0;
while (jj < 7)
{
      while (iii==0)
      {
            for (n=O[jj];n<=M[jj];n++)
            {
                  if (X5[ii][n]==1)
            {
                   yminL[jj]=ii;
                   iii=1;
                  }

        }
        ii=ii+1;
      }
    jj=jj+1;
    if (jj==3)
      jj=4;
    ii=1;
    iii=0;
}



ii=ymax-1;
jj=0;
iii=0;
while (jj < 7)
{
      while (iii==0)
```

```
      {
            for (n=O[jj];n<=M[jj];n++)
            {
                  if (X5[ii][n]==1)
                  {
                        ymaxL[jj]=ii;
                        iii=1;
                  }
            }
              ii=ii-1;
      }
      jj=jj+1;
      if (jj==3)
            jj=4;
    ii=ymax-1;
    iii=0;
}

short int X6[ymax+5][xmax+5];
for (m=ymax-10;m<ymax+5;m++)
 {
     for (n=0;n<xmax+5;n++)
             X6[m][n]=0;

}
ii=2;
jj=2;

for (m=0;m<ymax;m++)
{
      for (n=0;n<xmax;n++)
      {
        X6[jj][ii]=X5[m][n];
        ii=ii+1;
    }
    ii=2;
    jj=jj+1;
}

bool X7[ymax+5][xmax+5];
short int xmax1,ymax1;

ii=0;
jj=0;
for (m=2;m<ymax+2;m++)
{
      for (n=2;n<xmax+2;n++)
      {
            X7[ii][jj]= X6[m-2][n] || X6[m-1][n-1] || X6[m-1][n] || X6[m-
1][n+1]  ||  X6[m][n-2]  ||  X6[m][n-1]  ||  X6[m][n]  ||  X6[m][n+1]  ||
X6[m][n+2] || X6[m+1][n-1] || X6[m+1][n] || X6[m+1][n+1] || X6[m+2][n];
            jj=jj+1;

      }
```

```
      xmax1=jj;
      jj=0;
      ii=ii+1;
}
ymax1=ii;

int xini,yini,alto; char placa[6];
for (n=0;n<3;n++)
    {     if (n==0)
                {  xini=O[0]-1;
                   yini=yminL[0]-2;
                   ancho=M[0]-O[0]+1;
                   alto=ymaxL[0]-yminL[0]+2;
                }

          else if (n==1)
                {  xini=O[1]-1;
                   yini=yminL[1]-2;
                   ancho=M[1]-O[1]+1;
                   alto=ymaxL[1]-yminL[1]+2;
                }
          else
                {  xini=O[2]-1;
                   yini=yminL[2]-2;
                   ancho=M[2]-O[2]+1;
                   alto=ymaxL[2]-yminL[2]+2;
                }

//Preceso de identificar las letras


if(X7[yini+(int)(0.131*alto)][xini+4]==1 &&//2
 X7[yini+(int)(0.815*alto)][xini+(int)(0.15*ancho)]==1 &&
  X7[yini+(int)(0.42*alto)][xini+(int)(.5*ancho)]==1 &&
   X7[yini+(int)(0.266*alto)][xini+(int)(.842*ancho)]==1 &&
    X7[yini+(int)(0.735*alto)][xini+(int)(.894*ancho)]==1 &&
       X7[yini+(int)(0.69*alto)][xini+(int)(.53*ancho)]==0 &&
         X7[yini+(int)(0.6*alto)][xini+(int)(.15*ancho)]==1)
{placa[n]='B';}//cout <<"B  "<< endl;}

else if(X7[yini+(int)(0.073*alto)][xini+(int)(0.116*ancho)]==1 &&
 X7[yini+(int)(0.8*alto)][xini+(int)(.25*ancho)]==1 &&
  X7[yini+(int)(0.073*alto)][xini+(int)(.625*ancho)]==1 &&
   X7[yini+(int)(0.52*alto)][xini+(int)(.6*ancho)]==1 &&
    X7[yini+(int)(0.268*alto)][xini+(int)(.791*ancho)]==0 &&
     X7[yini+(int)(0.731*alto)][xini+(int)(.875*ancho)]==0 &&
      X7[yini+(int)(0.92*alto)][xini+(int)(.708*ancho)]==0)
{placa[n]='F';}//cout <<"F  "<< endl;}

else if(X7[yini+(int)round(0.131*alto)][xini+(2)]==1 &&
 X7[yini+(int)round(0.815*alto)][xini+(int)round(.15*ancho)]==1 &&
  X7[yini+(int)round(0.5*alto)][xini+(int)round(.5*ancho)]==1 &&
   X7[yini+(int)round(0.266*alto)][xini+(int)round(.842*ancho)]==1 &&
    X7[yini+(int)round(0.733*alto)][xini+(int)round(.894*ancho)]==1 &&
```

```
     X7[yini+(int)round(0.263*alto)][xini+(int)round(.55*ancho)]==0 &&
      X7[yini+(int)round(0.694*alto)][xini+(int)round(.5*ancho)]==0 &&
       X7[yini+(int)round(0.5*alto)][xini+(int)round(.15*ancho)]==0)
{placa[n]='S';}

else    if(X7[yini+(int)round(0.071*alto)][xini+(int)round(.153*ancho)]==1
&&
 X7[yini+(int)round(0.809*alto)][xini+(int)round(.23*ancho)]==1 &&
  X7[yini+(int)round(0.071*alto)][xini+(int)round(.576*ancho)]==1 &&
   X7[yini+(int)round(0.452*alto)][xini+(int)round(.576*ancho)]==1 &&
    X7[yini+(int)round(0.261*alto)][xini+(int)round(.73*ancho)]==0 &&
     X7[yini+(int)round(0.714*alto)][xini+(int)round(.807*ancho)]==0 &&
      X7[yini+(int)round(0.5*alto)][xini+(int)round(.1*ancho)]==1)
{placa[n]='E';}

else    if(X7[yini+(int)round(0.054*alto)][xini+(int)round(.166*ancho)]==1
&&
 X7[yini+(int)round(0.081*alto)][xini+(int)round(.944*ancho)]==1 &&
  X7[yini+(int)round(0.688*alto)][xini+(int)round(.5*ancho)]==1 &&
   X7[yini+(int)round(0.81*alto)][xini+(int)round(.166*ancho)]==0 &&
    X7[yini+(int)round(0.81*alto)][xini+(int)round(.944*ancho)]==0 &&
     X7[yini+(int)round(0.135*alto)][xini+(int)round(.55*ancho)]==0 &&
      X7[yini+(int)round(0.244*alto)][xini+(int)round(.5*ancho)]==0)
{placa[n]='V';}

else    if(X7[yini+(int)round(0.054*alto)][xini+(int)round(.166*ancho)]==1
&&
 X7[yini+(int)round(0.081*alto)][xini+(int)round(.944*ancho)]==1 &&
  X7[yini+(int)round(0.81*alto)][xini+(int)round(.611*ancho)]==1 &&
   X7[yini+(int)round(0.81*alto)][xini+(int)round(.166*ancho)]==0 &&
    X7[yini+(int)round(0.81*alto)][xini+(int)round(.944*ancho)]==0 &&
     X7[yini+(int)round(0.135*alto)][xini+(int)round(.55*ancho)]==0 &&
      X7[yini+(int)round(0.387*alto)][xini+(int)round(.5*ancho)]==1)
{placa[n]='Y';}

else if(X7[yini+(int)round(0.444*alto)][xini+(int)round(.22*ancho)]==1 &&
 X7[yini+(int)round(0.2*alto)][xini+(int)round(.22*ancho)]==1 &&
  X7[yini+(int)round(0.794*alto)][xini+(int)round(.166*ancho)]==1 &&
   X7[yini+(int)round(0.147*alto)][xini+(int)round(.833*ancho)]==1 &&
    X7[yini+(int)round(0.882*alto)][xini+(int)round(.833*ancho)]==1 &&
     X7[yini+(int)round(0.47*alto)][xini+(int)round(.888*ancho)]==0 &&
      X7[yini+(int)round(0.47*alto)][xini+(int)round(.5*ancho)]==0)
{placa[n]='C';}

else if(X7[yini+(int)round(0.166*alto)][xini+(int)round(.25*ancho)]==1 &&
 X7[yini+(int)round(0.833*alto)][xini+(int)round(.22*ancho)]==1 &&
  X7[yini+(int)round(0.194*alto)][xini+(int)round(.833*ancho)]==1 &&
   X7[yini+(int)round(0.888*alto)][xini+(int)round(.833*ancho)]==1 &&
    X7[yini+(int)round(0.437*alto)][xini+(int)round(.933*ancho)]==0 &&
     X7[yini+(int)round(0.437*alto)][xini+(int)round(.22*ancho)]==0 &&
      X7[yini+(int)round(0.88*alto)][xini+(int)round(.5*ancho)]==1)
{placa[n]='Z';}
```

```
else if(X7[yini+(int)round(0.166*alto)][xini+(int)round(.25*ancho)]==1 &&
 X7[yini+(int)round(0.833*alto)][xini+(int)round(.22*ancho)]==1 &&
  X7[yini+(int)round(0.194*alto)][xini+(int)round(.833*ancho)]==1 &&
   X7[yini+(int)round(0.888*alto)][xini+(int)round(.833*ancho)]==1 &&
    X7[yini+(int)round(0.437*alto)][xini+(int)round(.933*ancho)]==0 &&
     X7[yini+(int)round(0.437*alto)][xini+(int)round(.12*ancho)]==0 &&
      X7[yini+(int)round(alto)][xini+(int)round(.5*ancho)]==0)
{placa[n]='X';}


else if(X7[yini+(int)round(0.166*alto)][xini+(int)round(.22*ancho)]==1 &&
 X7[yini+(int)round(0.833*alto)][xini+(int)round(.22*ancho)]==1 &&
  X7[yini+(int)round(0.194*alto)][xini+(int)round(.833*ancho)]==1 &&
   X7[yini+(int)round(0.12*alto)][xini+(int)round(.45*ancho)]==1 &&
    X7[yini+(int)round(0.888*alto)][xini+(int)round(.833*ancho)]==1 &&
     X7[yini+(int)round(0.92*alto)][xini+(int)round(.4*ancho)]==0&&//0.92
      X7[yini+(int)round(0.84*alto)][xini+(int)round(.4*ancho)]==0)
{placa[n]='R';}

else if(X7[yini+(int)round(0.166*alto)][xini+(int)round(.22*ancho)]==1 &&
 X7[yini+(int)round(0.833*alto)][xini+(int)round(.22*ancho)]==1 &&
  X7[yini+(int)round(0.194*alto)][xini+(int)round(.833*ancho)]==1 &&
   X7[yini+(int)round(0.888*alto)][xini+(int)round(.833*ancho)]==1 &&
    X7[yini+(int)round(0.555*alto)][xini+(int)round(.888*ancho)]==0 &&
     X7[yini+(int)round(0.5*alto)][xini+(int)round(.1*ancho)]==1 )

{placa[n]='K';}


else    if(X7[yini+(int)round(0.189*alto)][xini+(int)round(.235*ancho)]==1
&&
 X7[yini+(int)round(0.864*alto)][xini+(int)round(.235*ancho)]==1 &&
  X7[yini+(int)round(0.189*alto)][xini+(int)round(.882*ancho)]==1 &&
   X7[yini+(int)round(0.864*alto)][xini+(int)round(.882*ancho)]==1 &&
    X7[yini+(int)round(0.5*alto)][xini+(int)round(.5*ancho)]==1 &&
     X7[yini+(int)round(0.863*alto)][xini+(int)round(.45*ancho)]==0 &&
      X7[yini+(int)round(0.3*alto)][xini+(int)round(.4*ancho)]==1 &&
        X7[yini+(int)round(0.1*alto)][xini+(int)round(.45*ancho)]==0)
{placa[n]='N';}


else    if(X7[yini+(int)round(0.189*alto)][xini+(int)round(.235*ancho)]==1
&&
 X7[yini+(int)round(0.864*alto)][xini+(int)round(.235*ancho)]==1 &&
  X7[yini+(int)round(0.189*alto)][xini+(int)round(.882*ancho)]==1 &&
   X7[yini+(int)round(0.864*alto)][xini+(int)round(.882*ancho)]==1 &&
    X7[yini+(int)round(0.702*alto)][xini+(int)round(.5*ancho)]==1 &&
     X7[yini+(int)round(1*alto)][xini+(int)round(.437*ancho)]==0 &&
      X7[yini+(int)round(0.1*alto)][xini+(int)round(.5*ancho)]==0)
{placa[n]='M';}


else if(X7[yini+(int)round(0.11*alto)][xini+(int)round(.2*ancho)]==1 &&
 X7[yini+(int)round(0.9*alto)][xini+(int)round(.266*ancho)]==1 &&
```

```
   X7[yini+(int)round(0.92*alto)][xini+(int)round(.866*ancho)]==1 &&
    X7[yini+(int)round(0.5*alto)][xini+(int)round(.5*ancho)]==1 &&
     X7[yini+(int)round(0.5*alto)][xini+(int)round(.1*ancho)]==0 )

{placa[n]='I';}

else if(X7[yini+(int)round(0.151*alto)][xini+(int)round(.2*ancho)]==1 &&
 X7[yini+(int)round(0.9*alto)][xini+(int)round(.126*ancho)]==0  &&  //.266
para arreglar la 64
  X7[yini+(int)round(0.5*alto)][xini+(int)round(.5*ancho)]==1 &&
   X7[yini+(int)round(0.5*alto)][xini+(int)round(.126*ancho)]==0    //.266
para arreglar la 64
   )
{placa[n]='T';}

else if(X7[yini+(int)round(0.151*alto)][xini+(int)round(.2*ancho)]==1 &&
 X7[yini+(int)round(0.878*alto)][xini+(int)round(.266*ancho)]==1 &&
  X7[yini+(int)round(0.92*alto)][xini+(int)round(.866*ancho)]==1 &&
   X7[yini+(int)round(0.5*alto)][xini+(int)round(.833*ancho)]==0 &&
    X7[yini+(int)round(0.3*alto)][xini+(int)round(.5*ancho)]==0 )
{placa[n]='L';}

else    if(X7[yini+(int)round(0.135*alto)][xini+(int)round(.187*ancho)]==1
&&
 X7[yini+(int)round(0.135*alto)][xini+(int)round(.875*ancho)]==1 &&
  X7[yini+(int)round(0.5*alto)][xini+(int)round(.15*ancho)]==1 &&
   X7[yini+(int)round(0.5*alto)][xini+(int)round(.875*ancho)]==1 &&
     X7[yini+(int)round(0.5*alto)][xini+(int)round(.5*ancho)]==0 &&
    X7[yini+(int)round(0.1*alto)][xini+(int)round(.5*ancho)]==0)
{placa[n]='U';}



else if(X7[yini+(int)round(0.111*alto)][xini+(int)round(.5*ancho)]==1 &&
 X7[yini+(int)round(0.5*alto)][xini+(int)round(.941*ancho)]==1 &&
  X7[yini+(int)round(0.916*alto)][xini+(int)round(.5*ancho)]==1 &&
   X7[yini+(int)round(0.5*alto)][xini+(int)round(.187*ancho)]==1 &&
    X7[yini+(int)round(0.525*alto)][xini+(int)round(.583*ancho)]==0 &&
     X7[yini+(int)round(0.111*alto)][xini+(int)round(.95*ancho)]==0 &&
      X7[yini+(int)round(2)][xini+(int)round(2)]==1)
{placa[n]='D';}


else if(X7[yini+(int)round(0.111*alto)][xini+(int)round(.5*ancho)]==1 &&
 X7[yini+(int)round(0.5*alto)][xini+(int)round(.941*ancho)]==1 &&
  X7[yini+(int)round(0.916*alto)][xini+(int)round(.5*ancho)]==1 &&
   X7[yini+(int)round(0.5*alto)][xini+(int)round(.187*ancho)]==1 &&
    X7[yini+(int)round(0.5*alto)][xini+(int)round(.5*ancho)]==0 &&
      X7[yini+(int)round(1)][xini+(int)round(1)]==0)
{placa[n]='O';}



else if(X7[yini+(int)round(0.111*alto)][xini+(int)round(.5*ancho)]==1 &&
```

```
 X7[yini+(int)round(0.5*alto)][xini+(int)round(.941*ancho)]==1 &&
  X7[yini+(int)round(0.916*alto)][xini+(int)round(.5*ancho)]==1 &&
   X7[yini+(int)round(0.5*alto)][xini+(int)round(.187*ancho)]==1 &&
    X7[yini+(int)round(0.525*alto)][xini+(int)round(.583*ancho)]==1 &&
     X7[yini+(int)round(2)][xini+(int)round(2)]==0)
{placa[n]='G';}


else if(X7[yini+(int)round(0.111*alto)][xini+(int)round(.3*ancho)]==0 &&
 X7[yini+(int)round(0.5*alto)][xini+(int)round(.928*ancho)]==1 &&
  X7[yini+(int)round(0.916*alto)][xini+(int)round(.5*ancho)]==1 &&
   X7[yini+(int)round(0.5*alto)][xini+(int)round(.187*ancho)]==0)
  {    if( X7[yini+(int)round(0.111*alto)][xini+(int)round(.83*ancho)]==1
||    X7[yini+(int)round(0.111*alto)][xini+(int)round(.75*ancho)]==1)//.75
no sirve para carro4
    {placa[n]='J';}
  }

else if(X7[yini+(int)round(0.111*alto)][xini+(int)round(.22*ancho)]==1 &&
 X7[yini+(int)round(0.833*alto)][xini+(int)round(.22*ancho)]==1 &&
  X7[yini+(int)round(0.194*alto)][xini+(int)round(.833*ancho)]==1 &&
   X7[yini+(int)round(0.888*alto)][xini+(int)round(.833*ancho)]==1 &&
    X7[yini+(int)round(0.555*alto)][xini+(int)round(.888*ancho)]==1 &&
     X7[yini+(int)round(0.5*alto)][xini+(int)round(.5*ancho)]==1 &&
      X7[yini+(int)round(0.15*alto)][xini+(int)round(.5*ancho)]==0)
{placa[n]='H';}


else    if(X7[yini+(int)round(0.137*alto)][xini+(int)round(.222*ancho)]==1
&&
 X7[yini+(int)round(0.896*alto)][xini+(int)round(.222*ancho)]==1 &&
   X7[yini+(int)round(0.31*alto)][xini+(int)round(.5*ancho)]==0)
{placa[n]='P';}


else if(X7[yini+(int)round(0.166*alto)][xini+(int)round(.5*ancho)]==1 &&
 X7[yini+(int)round(0.857*alto)][xini+(int)round(.095*ancho)]==1 &&
  X7[yini+(int)round(0.857*alto)][xini+(int)round(.869*ancho)]==1 &&
   X7[yini+(int)round(0.785*alto)][xini+(int)round(.5*ancho)]==1 &&
    X7[yini+(int)round(0.071*alto)][xini+(int)round(.13*ancho)]==0 &&
     X7[yini+(int)round(0.071*alto)][xini+(int)round(.913*ancho)]==0)
{placa[n]='A';}

else if(X7[yini+(int)round(0.2*alto)][xini+(int)round(.2*ancho)]==1 &&
 X7[yini+(int)round(0.6*alto)][xini+(int)round(.2*ancho)]==1 &&
  X7[yini+(int)round(0.6*alto)][xini+(int)round(.8*ancho)]==1 &&
   X7[yini+(int)round(0.2*alto)][xini+(int)round(.8*ancho)]==1)
{placa[n]='W';}


else placa[n]='-';

if (n==10 )
```

```
{                                                                    cout      <<"
"<<X7[yini+(int)round(0.151*alto)][xini+(int)round(.2*ancho)] << endl;
 cout   <<"   "<<X7[yini+(int)round(0.9*alto)][xini+(int)round(.126*ancho)]
<< endl;
  cout <<" "<<X7[yini+(int)round(0.5*alto)][xini+(int)round(.5*ancho)] <<
endl;
   cout <<" "<<X7[yini+(int)round(0.5*alto)][xini+(int)round(.126*ancho)]
<< endl;
         }
}

for (n=3;n<6;n++)
    {     if (n==3)
                {  xini=O[4]-1;
                   yini=yminL[4]-2;
                   ancho=M[4]-O[4]+1;
                   alto=ymaxL[4]-yminL[4]+2;
                }

          else if (n==4)
                {  xini=O[5]-1;
                   yini=yminL[5]-2;
                   ancho=M[5]-O[5]+1;
                   alto=ymaxL[5]-yminL[5]+2;
                }
          else
                {  xini=O[6]-1;
                   yini=yminL[6]-2;
                   ancho=M[6]-O[6]+1;
                   alto=ymaxL[6]-yminL[6]+2;
                }


//Preceso de identificar los numeros


if(X7[yini+(int)(0.105*alto)][xini+(int)(0.5*ancho)]==1 &&//2
 X7[yini+(int)(0.921*alto)][xini+(int)(0.5*ancho)]==1 &&
  X7[yini+(int)(0.289*alto)][xini+(int)(.28*ancho)]==1 &&
   X7[yini+(int)(0.289*alto)][xini+(int)(.84*ancho)]==1 &&
    X7[yini+(int)(0.816*alto)][xini+(int)(.28*ancho)]==1 &&
       X7[yini+(int)(0.71*alto)][xini+(int)(.5*ancho)]==0 &&
         X7[yini+(int)(0.5*alto)][xini+(int)(.5*ancho)]==1)
{placa[n]='8';}

else if(X7[yini+(int)round(0.111*alto)][xini+(int)round(.5*ancho)]==1 &&
 X7[yini+(int)round(0.5*alto)][xini+(int)round(.941*ancho)]==1 &&
  X7[yini+(int)round(0.916*alto)][xini+(int)round(.5*ancho)]==1 &&
   X7[yini+(int)round(0.5*alto)][xini+(int)round(.187*ancho)]==1 &&
    X7[yini+(int)round(0.39*alto)][xini+(int)round(.5*ancho)]==0 &&
     X7[yini+(int)round(0.5*alto)][xini+(int)round(.5*ancho)]==0)
{placa[n]='0';}
```

```
else    if(X7[yini+(int)round(0.146*alto)][xini+(int)round(.192*ancho)]==1
&&
 X7[yini+(int)round(0.5*alto)][xini+(int)round(.192*ancho)]==1 &&
  X7[yini+(int)round(0.3*alto)][xini+(int)round(.192*ancho)]==1 &&
   X7[yini+(int)round(0.272*alto)][xini+(int)round(.888*ancho)]==0)
{placa[n]='5';}


else if(X7[yini+(int)round(0.096*alto)][xini+(int)round(.5*ancho)]==1 &&
 X7[yini+(int)round(0.419*alto)][xini+(int)round(.5*ancho)]==1 &&
  X7[yini+(int)round(0.935*alto)][xini+(int)round(.5*ancho)]==1 &&
   X7[yini+(int)round(0.709*alto)][xini+(int)round(.5*ancho)]==0)
{placa[n]='3';}

else if(X7[yini+(int)round(0.096*alto)][xini+(int)round(.5*ancho)]==0 &&
 X7[yini+(int)round(0.323*alto)][xini+(int)round(.545*ancho)]==1 &&
  X7[yini+(int)round(0.935*alto)][xini+(int)round(.5*ancho)]==1 &&
   X7[yini+(int)round(0.709*alto)][xini+(int)round(.5*ancho)]==0)
{placa[n]='6';}


else if(X7[yini+(int)round(0.96*alto)][xini+(int)round(.5*ancho)]==0 &&
   X7[yini+(int)round(0.29*alto)][xini+(int)round(.21*ancho)]==1)
{placa[n]='9';}


else if(X7[yini+(int)round(0.931*alto)][xini+(int)round(.25*ancho)]==1 &&
 X7[yini+(int)round(0.931*alto)][xini+(int)round(.75*ancho)]==1 &&
  X7[yini+(int)round(0.322*alto)][xini+(int)round(.5*ancho)]==1)
{placa[n]='1';}

else    if(X7[yini+(int)round(0.114*alto)][xini+(int)round(.238*ancho)]==1
&&
   X7[yini+(int)round(0.914*alto)][xini+(int)round(.1*ancho)]==0)
{placa[n]='7';}

else if(X7[yini+(int)round(0.909*alto)][xini+(int)round(.2*ancho)]==1 &&
   X7[yini+(int)round(0.909*alto)][xini+(int)round(.8*ancho)]==1)
{placa[n]='2';}

else if(X7[yini+(int)round(0.903*alto)][xini+(int)round(.235*ancho)]==0 )
{placa[n]='4';}

else placa[n]='-';
}
cout <<" "<< endl;
cout  <<"La  Placa  del  Vehiculo:    "<<placa[0]<<placa[1]<<placa[2]<<"
"<<placa[3]<<placa[4]<<placa[5]<< endl;

fin=clock();
t=(fin-ini);
gettimeofday(&t_fin,NULL);
secs = timeval_diff(&t_fin,&t_ini);
printf("Tiempo de ejecucion: %.3g seconds\n",secs);
```

```
getchar();
 return 0;}
```