



**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**

**INFORME DE MATERIA DE GRADUACIÓN**

**“SEGMENTACIÓN DE IMÁGENES DE PLACAS VEHICULARES USANDO  
TÉCNICA DE CRECIMIENTO DE REGIONES”**

**Previo a la obtención del título de:**

**INGENIERO EN CIENCIAS COMPUTACIONALES**

**ESPECIALIZACIÓN SISTEMAS TECNOLÓGICOS**

**PRESENTADA POR:**

**JOSE MAXIMILIANO RIVERA DE LA CRUZ**

**ALEXANDRA DE LOURDES BURI HERAS**

**GUAYAQUIL – ECUADOR**

**2011**

## AGRADECIMIENTO

“A Dios por la culminación de esta etapa académica “.

“A todas las personas que de alguna manera

Contribuyeron desde el primer semestre

Hasta la culminación universitaria”.

Por sus enseñanzas durante

Nuestra vida estudiantil”.

## DEDICATORIA

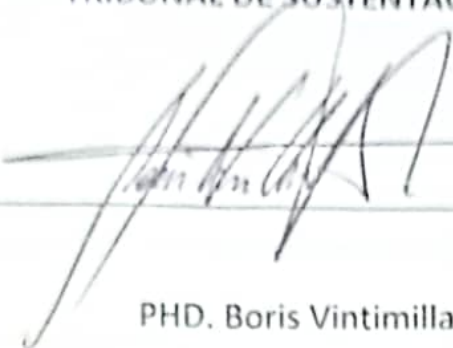
“A nuestras familias por todo su apoyo

Comprensión y la confianza depositada

Que nos han acompañado a lo largo del camino

Como fuente inspiradora y guía continua”.

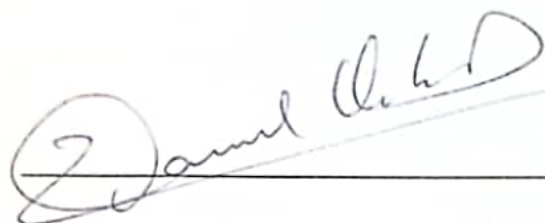
TRIBUNAL DE SUSTENTACIÓN



---

PHD. Boris Vintimilla

PROFESOR DE LA MATERIA DE GRADUACION



---

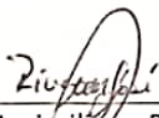
Ing. Daniel Ochoa

PROFESOR DELEGADO POR EL DECANO

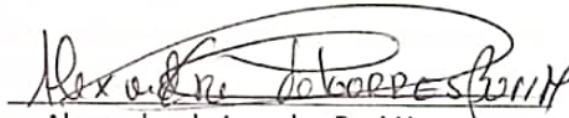
## DECLARACIÓN EXPRESA

“La responsabilidad por los hechos ideas y doctrinas expuestas en este trabajo nos corresponden exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral”.

(Reglamento de exámenes y títulos profesionales de la ESPOL)



José Maximiliano Rivera De La Cruz



Alexandra de Lourdes Buri Heras

## RESUMEN

El procesamiento digital de imágenes es el conjunto de técnicas que se aplican a las imágenes digitales con el objetivo de mejorar la calidad o facilitar la búsqueda de información. La correcta visualización y evaluación de detalles de una imagen que no son perceptibles en su toma original es uno de los propósitos del procesamiento de imágenes.

En este sentido una adecuada visualización de detalles como los números y caracteres correspondiente a una imagen de placa vehicular es fundamental a la hora de establecer las características de un vehículo proporcionando información vital sobre él y su propietario.

El presente trabajo sobre imágenes de placas de vehículos logra segmentar caracteres que servirán de base para el reconocimiento de las placas. Primeramente utilizaremos técnicas de pre procesamiento para la imagen ya que puede haber elementos no deseables como ruido que se presentan generalmente en imágenes.

Para esto se realiza un pre procesamiento de la imagen, posteriormente se detecta los contornos que definen nuestro área de interés dentro de toda la imagen consistente en la placa del carro. Una vez detectada la placa continuaremos con la técnica de crecimiento de regiones en la segmentación a partir de un conjunto de semillas que irán creciendo hasta detectar los bordes que definen un carácter además del uso de filtros para la eliminación de regiones no válidas dentro de la

imagen. Finalmente se tiene la presentación de la placa obteniendo nuestro objetivo que es la placa segmentada y que servirá para posteriores procesos como OCR.

Todos estos módulos incluidos dentro de un sistema de reconocimiento automático de una placa vehicular.

Para el desarrollo de este trabajo se utiliza la librería OpenCv herramienta que permite el análisis y tratamiento de imágenes con muchas funciones implementadas para el procesamiento de las mismas en conjunto con Visual Studio 2008 y Visual C++.

## Índice General

AGRADECIMIENTO .....	I
DEDICATORIA .....	II
TRIBUNAL DE SUSTENTACIÓN .....	III
DECLARACIÓN EXPRESA.....	IV
RESUMEN .....	V
INTRODUCCIÓN .....	1
CAPÍTULO 1.....	1
DESCRIPCIÓN DEL PROYECTO .....	3
1.1 Justificación.....	3
1.2 Objetivos del proyecto.....	4
1.2.1 Objetivo general .....	4
1.2.2 Objetivos específicos .....	4
CAPÍTULO 2.....	6
FUNDAMENTOS TEÓRICOS.....	6



2.1 Resumen .....	6
2.2 Clasificación de las técnicas de segmentación .....	7
2.3 Segmentación por discontinuidad .....	9
2.3.1 Detección de discontinuidades:.....	9
2.3.1 .1 Técnica 1: Detección de puntos.....	10
2.3.1.2 Técnica 2: Detección de líneas.....	10
2.3.1.3 Técnica 3: Detección de bordes.....	11
2.3.2 Unión de bordes y detección de discontinuidades .....	15
2.3.2.1 Técnica 4: Procesado local.....	15
2.3.2.2 Técnica 5: Procesado global vía transformada de Hough.....	15
2.3.2.3 Técnica 6: Seguimiento de contornos (teoría de grafos) .....	18
2.4 Segmentación por similitud .....	19
2.4 .1 Técnica 7: Umbralización .....	20
2.4.2 Técnica 8: Crecimiento de regiones.....	22
2.4.2.1 Técnica 8.1: División y fusión de regiones .....	22
2.4.2.2 Técnica 8.2: Crecimiento de regiones.....	24
2.5 Comparativa de los diferentes modelos.....	29
2.5.1 Elección del modelo de segmentación .....	31
2.6 Historia y estructura de la librería OpenCv.....	38

2.7 Estructura de OpenCV.....	40
2.8 Glosario.....	42
<b>CAPÍTULO 3.....</b>	<b>47</b>
<b>DETECCIÓN DE PLACA.....</b>	<b>47</b>
3.1 Características de la imagen.....	48
3.2 Diagrama de flujo para la detección de la placa.....	53
3.3 Diagrama de módulos para la detección de la placa.....	55
3.3 Implementación.....	55
3.4 Etapa de pre procesamiento de la imagen.....	56
3.4 .1 Conversión a nivel de gris.....	56
3.4.2 Ecuilización.....	57
3.4.3 Umbralización.....	58
3.5 Etapa de detección de la placa.....	63
3.5.1 Detección de contornos.....	64
3.5.2 Selección de rectángulos.....	66
3.5.3 Corrección geométrica.....	67
3.5.4 Selección de placa.....	69
<b>CAPÍTULO 4.....</b>	<b>72</b>
<b>SEGMENTACIÓN DE LA PLACA.....</b>	<b>72</b>

4.1 Diagrama de módulos para la segmentación de placa.....	72
4.2 Diagrama de flujo para segmentación de la placa.....	74
4.3 Descripción de los módulos para la segmentación de placa.....	75
4.3.1 Obtención de semillas.....	75
4.3.2 Crecimiento de regiones.....	77
4.3.3 Eliminación de regiones no válidas.....	79
4.3.4 Presentación de placa.....	82
CAPÍTULO 5.....	83
ANÁLISIS DE RESULTADOS Y CONCLUSIONES .....	83
5.1 Análisis de resultados .....	83
5.2 Conclusiones y recomendaciones.....	90
Anexo 1: .....	72
Bibliografía .....	84

## Índice de figuras

### Capítulo 2

Fig.2. 1 Esquema general de visión artificial .....	7
Fig.2. 2 Clasificación de las técnicas de segmentación .....	8

Fig.2. 3 Máscara y parte de la imagen .....	9
Fig.2. 4 Máscara utilizada por la detección de puntos.....	10
Fig.2. 5 Ejemplo de detección de líneas a 45 grados. ....	11
Fig.2. 6 Ejemplo de bordes a) borde digital ideal b) borde digital tipo rampa .....	12
Fig.2. 7 Uso de la derivada para la detección de los bordes.....	13
Fig.2. 8 Transformación del espacio(x,y)→espacio(ρ,θ). ....	16
Fig.2. 9 Grafo de elementos de bordes de una imagen .....	19
Fig.2. 10 Uso de umbrales múltiples en un imagen .....	21
Fig.2. 11 a) Imagen dividida en subregiones b) árbol cuaternario de división.....	23
Fig.2. 12 Ejemplo de división y fusión de imágenes.....	24
Fig.2. 13 Técnica de crecimiento de regiones.....	26
Fig.2. 14 Vecindad vertical y horizontal .....	27
Fig.2. 15 Vecindad diagonal .....	27
Fig.2. 16 Conectividad a ocho .....	28
Fig.2. 17 Conectividad mezclada.....	29
Fig.2. 18 Diagrama de bloque inicial .....	31
Fig.2. 19 Segundo diagrama de bloques .....	32
Fig.2. 20 Resultados de métodos de umbralización iniciales .....	35
Fig.2. 21 Porcentajes de efectividad de métodos de umbralización .....	35
Fig.2. 22 : Resultados de métodos de umbralización secundarios. ....	37
Fig.2. 23 Estructura de la librería OpenCv .....	41

Fig.2. 24 Imagen original y zoom de la imagen .....	42
Fig.2. 25 Imagen monocroma codificada en 256 niveles de intensidad (niveles de gris) y su correspondiente histograma de intensidades o niveles de gris. ....	43
Fig.2. 26 La línea amarilla en el histograma indica el valor medio de intensidad (valor medio de nivel de gris). ....	43
Fig.2. 27 Imagen en modo escala de grises.....	44
Fig.2. 28 a) Función de transformación b) grafica del operador umbral. ....	45
Fig.2. 29 a) Imagen original b) segmentación con umbral de 30 c) segmentación con umbral de 52 .....	46
Fig.2. 30 a) Función de transformación b) operador intervalo de umbral binario ....	46

### Capitulo3

Fig.3. 1 Distancias de enfoques utilizados para la adquisición de las imágenes ,24 distancias de enfoque. ....	49
Fig.3. 2 Esquema general del proyecto global. ....	50
Fig.3. 3 Alto de la placa en la imagen para cada punto, puntos analizados 1-12.....	51
Fig.3. 4 Alto de la placa en la imagen para cada punto, puntos analizados 14-24....	52
Fig.3. 5 Diagrama de flujo para la detección de la placa. ....	54
Fig.3. 6 Diagrama de módulos para la detección de la placa.....	55
Fig.3. 7 Diagrama de módulos para la etapa de pre procesamiento.....	56
Fig.3. 8 Conversión de una imagen a color a su correspondiente imagen de nivel de gris.....	57

Fig.3. 9 Ecuación de histogramas.....	58
Fig.3. 10 Imagen umbralizada .....	59
Fig.3. 11 Ejemplo de umbralización usando Otsu.....	60
Fig.3. 12 Descripción de histograma .....	61
Fig.3. 13 Umbral óptimo .....	61
Fig.3. 14 Imagen 6 * 6 .....	62
Fig.3. 15 Cálculo de parámetros para el fondo ( <i>background</i> ) .....	62
Fig.3. 16 Cálculo de parámetros para el primer plano ( <i>foreground</i> ) .....	63
Fig.3. 17 Diagrama de módulos para la detección de placa. ....	64
Fig.3. 18 Detección de contornos en la placa. ....	65
Fig.3. 19 Selección de rectángulo (área de interés).....	67
Fig.3. 20 Estado de la imagen después de la corrección.....	69
Fig.3. 21 Placa detectada .....	70

#### Capítulo 4

Fig.4. 1 Diagrama de módulos para la segmentación de placa. ....	73
Fig.4. 2 Diagrama de flujo para la segmentación.....	74
Fig.4. 3 Estadísticas para el umbral.....	76
Fig.4. 4 Imagen con puntos semillas de color blanco .....	76
Fig.4. 5 Imagen previa al crecimiento de regiones .....	77

Fig.4. 6 Inicio de crecimiento de regiones. ....	78
Fig.4. 7 Valores experimentales de área (números). ....	80
Fig.4. 8 Valores experimentales de área (letras) ....	80
Fig.4. 9 Estadísticas de relación alto carácter respecto al alto de placa (valores experimentales). ....	81
Fig.4. 10 Eliminación de regiones no válidas.....	82
Fig.4. 11 Imagen segmentada. ....	82

### Capítulo 5

Fig.5. 1 Imagen de puntos de muestra 1, 3, 5,16.....	86
Fig.5. 2 Imagen segmentada resultante para el punto 1 .....	87
Fig.5. 3 Imagen segmentada resultante par el punto 3 .....	87
Fig.5. 4 Imagen segmentada resultante pare el punto 5 .....	87
Fig.5. 5 Imagen segmentada resultante para el punto 16 .....	88
Fig.5. 6 Imagen de puntos que el algoritmo segmenta correctamente .....	89

### Índice de tablas

Tabla 1 Comparación de los modelos de segmentación. ....	30
Tabla 2 Tamaños de la imagen según algoritmo .....	36
Tabla 3 Porcentaje de imágenes segmentadas según cada punto.....	84

## INTRODUCCIÓN

El separar la imagen en unidades significativas es un paso importante en visión computacional para llegar al reconocimiento de objetos. Este proceso se conoce como segmentación.

La detección de dígitos en una placa vehicular proporciona información vital como paso previo para el reconocimiento de caracteres que pueden ser utilizados en diferentes aplicaciones que sirve como base de información para instituciones sobre el tipo de vehículo además de información del conductor.

El sistema propuesto en este trabajo se compone de 2 módulos principales estos son: la detección de la placa y la segmentación de los caracteres. El módulo de detección se encarga de encontrar la placa del vehículo en la imagen capturada y recortarla. Esta imagen recortada es pasada al módulo de segmentación que selecciona y separa los caracteres.

Estos dos módulos trabajan en conjunto, tanto la detección de la placa [1] como la segmentación, ya que el resultado del primer módulo afecta directamente el resultado de la segmentación de la imagen.

En este documento se describe un método para la detección de placa que utiliza la detección de contornos el análisis de la proyección vertical para encontrar y recortar



la placa vehicular de una imagen. Esto es debido a que la placa tiene una forma rectangular como característica.

En el capítulo 1 se describe el problema que se pretende resolver también se establece condiciones generales y requerimientos de diseño.

En el capítulo 2 se describe los fundamentos teóricos sobre diferentes métodos de segmentación.

En el capítulo 3 se describe la forma en la que se realizó la implementación de la detección de placa.

En el capítulo 4 se describe la forma en la que se realizó la implementación de la segmentación una vez obtenida la placa.

En el capítulo 5 se realiza una evaluación de los resultados obtenidos de implementar nuestra aplicación así como conclusiones y recomendaciones para futuros trabajos.

## **CAPÍTULO 1**

### **DESCRIPCIÓN DEL PROYECTO**

En la actualidad las técnicas de procesamiento y análisis de imágenes han adquirido gran importancia en los sistemas de seguridad monitoreo y reconocimiento de patrones usándolo esencialmente en el control de acceso vehicular a entidades y establecimientos en general. Una de las características dentro de un vehículo es su placa que permite conocer al propietario características del automóvil, récord de manejo y otra información de mucho interés.

Para el presente proyecto se plantea el tema **“SEGMENTACIÓN DE IMÁGENES DE PLACAS VEHICULARES USANDO TÉCNICA DE CRECIMIENTO DE REGIONES”**.

Este proyecto desea resolver el siguiente problema: obtenida la imagen de una placa normalizada, cada dígito de la imagen de la placa es segmentado y descompuesto de manera individual aplicando para ello una técnica de crecimiento de regiones. Los dígitos que han sido segmentados pueden pasar posteriormente a un proceso de reconocimiento de caracteres (OCR).

#### **1.1 Justificación**

El reconocimiento de placas vehiculares es utilizado en diversas aplicaciones tales como:

- Control de acceso a parqueaderos.
- Control de peaje.
- Tráfico vehicular.
- Sistemas de seguridad aeroportuaria.

Por mencionar algunas de sus aplicaciones de ahí la relevancia del presente trabajo como un aporte a la sociedad y la ciudadanía en general.

## **1.2 Objetivos del proyecto**

El objetivo general y los objetivos específicos de este proyecto son descritos a continuación:

### **1.2.1 Objetivo general**

Desarrollar un sistema que permita detectar y segmentar la placa de un vehículo partir de una imagen dada.

Dentro de los diferentes métodos para segmentar una imagen elegimos el método crecimiento de regiones con el cual se logra una mejor segmentación con un menor costo computacional.

### **1.2.2 Objetivos específicos**

Los objetivos específicos del proyecto son los siguientes:

Leer la imagen previamente obtenida por medio de una cámara.

Seleccionar y extraer la placa del vehículo.

Realizar la corrección geométrica de la placa del vehículo.

Segmentar la imagen en cada uno de sus dígitos mediante el método de crecimiento de regiones.

Realizar filtrado a la imagen segmentada tanto por área y tamaño.

Realizar evaluación y comparación de los resultados obtenidos.

## CAPÍTULO 2

### FUNDAMENTOS TEÓRICOS

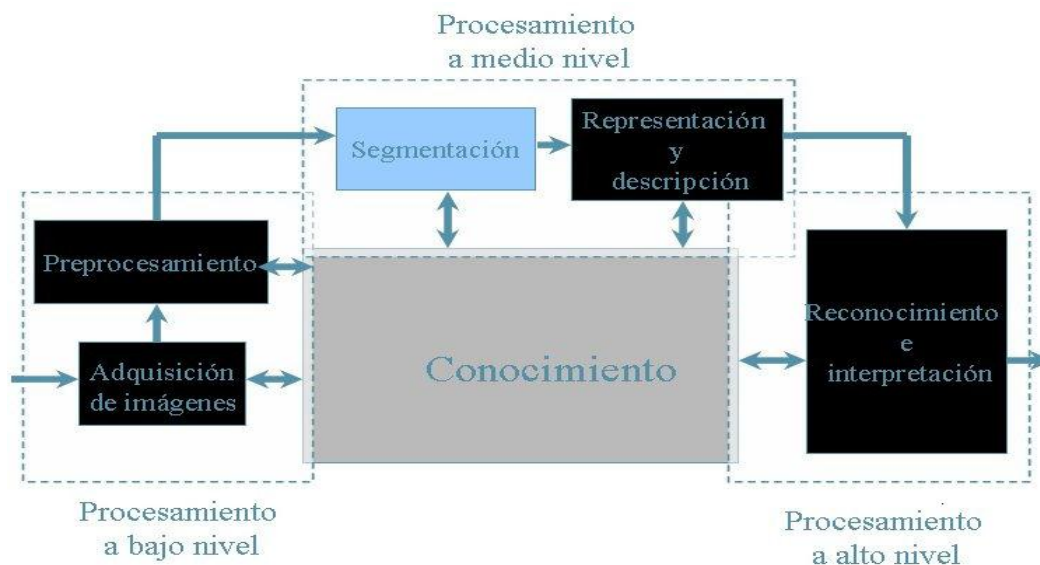
#### 2.1 Resumen

En este capítulo se describe el marco teórico para el presente trabajo con una pequeña introducción al procesamiento de imágenes. Posteriormente tenemos la clasificación de las diferentes técnicas de segmentación con su respectiva descripción realizando un comparativo entre las ventajas y desventajas de cada una de ellas. Finalmente realizamos un análisis para la selección de la técnica de segmentación a usaren este proyecto.

En el procesamiento de imágenes, la segmentación es un proceso fundamental, ya que permite diferenciar los diferentes objetos que la componen y etiquetar cada una de ellos para su posterior aplicación de un determinado tratamiento. Una buena segmentación permite el desarrollo de numerosas aplicaciones tecnológicas en muchos ámbitos industriales [\[2\]](#).

Se trata de agrupar los píxeles por algún criterio de homogeneidad para fraccionar la escena en regiones de interés. Estas áreas deben tener algún significado físico. Por tanto la segmentación de una imagen es un proceso de extracción de los objetos de interés insertados en la escena capturada. La imagen estará definida por un conjunto de objetos habiendo pasado de un

nivel bajo a otro más elaborado o nivel medio visual. La información estará preparada para el reconocimiento e interpretación de la imagen. El esquema resultante es mostrado en la Fig.2.1.



**Fig.2. 1 Esquema general de visión artificial**

Fuente: Apuntes visión artificial 2007[2]

## 2.2 Clasificación de las técnicas de segmentación

Existen varios tipos de segmentación las cuales se clasifican de acuerdo a las características entre píxeles de una imagen [3].

La Fig. 2.2 muestra la clasificación de las técnicas de segmentación y el criterio sobre el cual éstas se clasifican y que son descritas a continuación:

La segmentación se clasifica basada en:

Discontinuidad: Se propone la división de la imagen según cambios abruptos del nivel de gris.

Similitud: Comparar grupos de píxeles considerando una región a aquel grupo de píxeles que tienen propiedades similares.

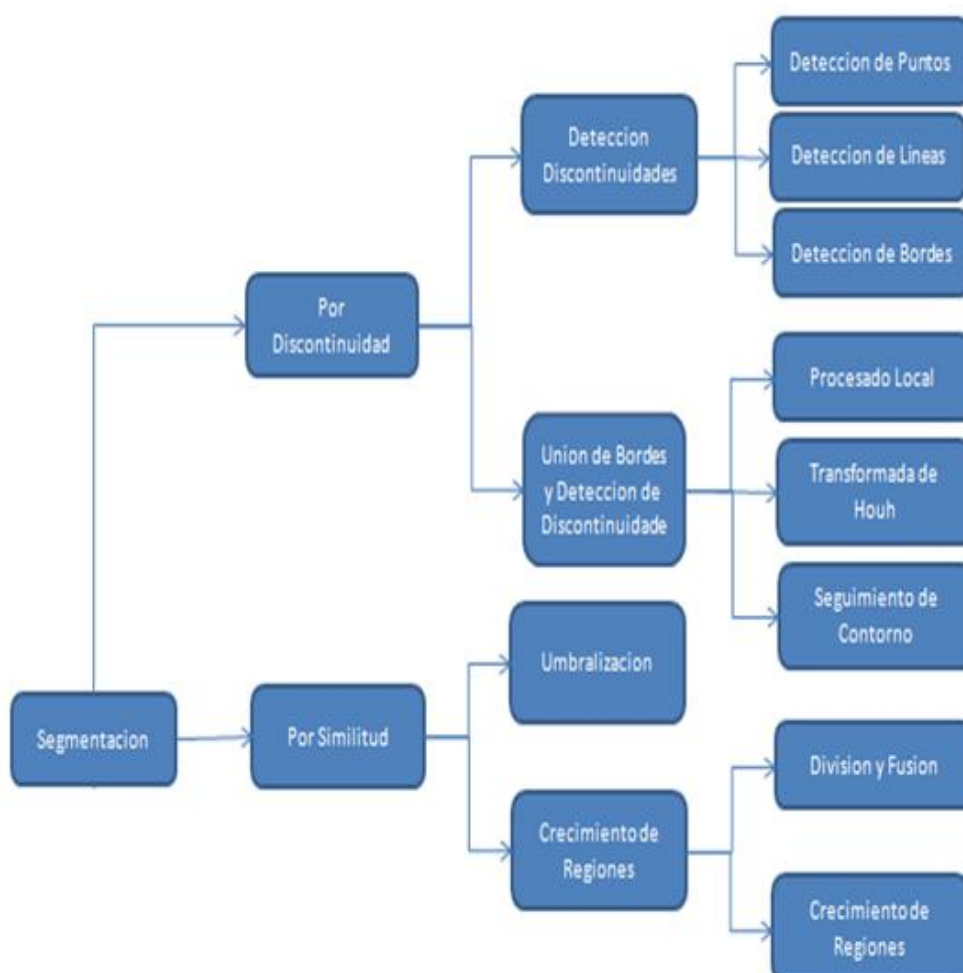


Fig.2. 2 Clasificación de las técnicas de segmentación

Fuente: Segmentación por intersección de histograma de color y textura 2009 [3]

### 2.3 Segmentación por discontinuidad

La segmentación por discontinuidad se divide a su vez en dos grupos:

- 1) Por detección de discontinuidades
- 2) Unión de bordes y detección de discontinuidades.

#### 2.3.1 Detección de discontinuidades:

Para hallar discontinuidades dentro de una imagen ya sean puntos líneas o bordes se aplican máscaras a la zona de la imagen que se desea analizar. Esto se realiza mediante la multiplicación de cada uno de los valores de los píxeles de la zona a examinar resultados que se suman para obtener un valor R. Si R cumple cierto umbral se habrá detectado cierta forma Fig. 2.3.

$$R = \sum_{i=1}^9 W_i Z_i = W_1 Z_1 + W_2 Z_2 + W_3 Z_3 + \dots + W_i Z_i \quad (1.1)$$

$W_1$	$W_2$	$W_3$
$W_4$	$W_5$	$W_6$
$W_7$	$W_8$	$W_9$

Máscara

$Z_1$	$Z_2$	$Z_3$
$Z_4$	$Z_5$	$Z_6$
$Z_7$	$Z_8$	$Z_9$

Parte de la imagen

Fig.2. 3 Máscara y parte de la imagen

Fuente: Segmentación de color por intersección de histograma 2009



Dentro de la detección de discontinuidades tenemos:

### 2.3.1 .1 Técnica 1: Detección de puntos

La detección de puntos es un método muy simple basado en la aplicación de una máscara centrada en el píxel a analizar. Si cumple con cierto umbral T el punto es detectado:  $|R| \geq T$  (1.2)

La máscara habitualmente usada para la detección de puntos es de la Fig. 2.4

$$\begin{pmatrix} 1 & -1 & -1 \\ -1 & -8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

**Fig.2. 4 Máscara utilizada por la detección de puntos**

**Fuente: Segmentación de color por intersección de histograma 2009**

Esta máscara mide la relación de píxeles entre un píxel y sus adyacentes detectando aquel píxel que marque mucho la diferencia de intensidad sobre sus adyacentes.

### 2.3.1.2 Técnica 2: Detección de líneas

Se usan diferentes máscaras según la dirección que se pretenda identificar. Cada máscara puede ser aplicada por separado o en conjunto dependiendo de lo que si desea es encontrar cualquier línea

dentro de la imagen o líneas con dirección determinada. Se muestra diferentes tipos de máscaras (Fig. 2.5):

$$\begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix} \begin{pmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{pmatrix} \begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix} \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

Horizontal

b) + 45 °

c) Vertical

d) -45 °



Fig.2. 5 Ejemplo de detección de líneas a 45 grados.

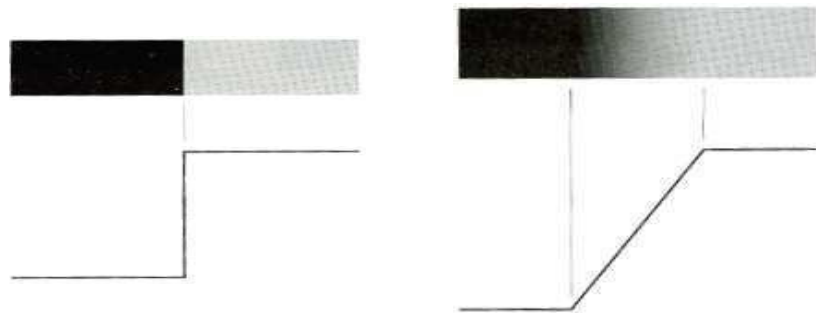
Fuente: Segmentación de color por intersección de histograma 2009

Cada una de estas máscaras responde a unos umbrales  $R_1 R_2 R_3 R_4$  . Cada máscara puede ser aplicada por separada o en conjunto dependiendo si lo que se pretende es cualquier línea que encuentra en la imagen o en una determinada dirección Fig. 2.5.

### 2.3.1.3 Técnica 3: Detección de bordes

Es la principal técnica de segmentación de imágenes en escala de grises. Un borde dentro de una imagen es considerado un tipo de discontinuidad, por lo que este puede ser detectado usando derivadas de primer y segundo orden.

Básicamente se trata del cálculo de un operador derivada local de primer o de segundo orden (gradiente o laplaciana). Se puede decir que la primera derivada proporciona la dirección del borde mientras que la segunda derivada indica si los píxeles implicados pertenecen a la zona clara u oscura del borde. Los bordes son modelados en forma de rampa y no en escalón como debería ser y provoca que la segmentación de la imagen no sea exacta (Fig. 2.6).



**Fig.2. 6 Ejemplo de bordes a) borde digital ideal b) borde digital tipo rampa**

**Fuente: Segmentación de color por intersección de histograma 2009**

El uso de las derivadas primera y segunda ayuda a modelar este tipo de bordes ya que una detecta el borde completo y la otra detecta el comienzo y fin del mismo (Fig. 2.7).

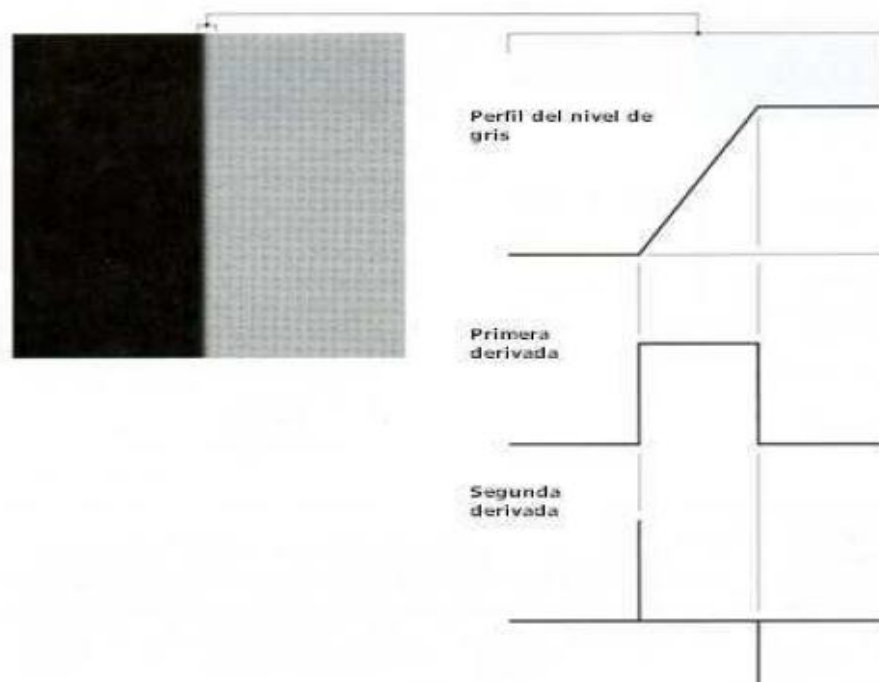


Fig.2. 7 Uso de la derivada para la detección de los bordes

Fuente: Segmentación de color por intersección de histograma 2009

### Derivada de primer orden (gradiente):

La derivada de primer orden en imágenes digitales se aproxima con la función gradiente de dos dimensiones así pues el gradiente de la función  $f(x,y)$  en el punto  $(x,y)$  se denota por  $\nabla f(x,y)$  y se define como:

$$\nabla f(x,y) = \begin{bmatrix} \frac{df(x,y)}{dx} \\ \frac{df(x,y)}{dy} \end{bmatrix} = \begin{bmatrix} f_x(x,y) \\ f_y(x,y) \end{bmatrix} = \begin{bmatrix} Gf \\ Gc \end{bmatrix} \quad (1.2)$$

$Gf$ = Gradiente de filas.

$Gc$ = Gradiente de columnas.

El módulo del vector gradiente es muy importante dentro del proceso de detección de bordes ya que proporciona el máximo incremento de  $f(x, y)$  por unidad de distancia y se determina mediante:

$$|\nabla f(x, y)| = \sqrt{G_f^2 + G_c^2} \quad (1.3)$$

La dirección del vector gradiente representa el ángulo perpendicular al vector de dirección del borde en el punto  $(x, y)$ :

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_c}{G_f}\right) \quad (1.4)$$

El cálculo del gradiente de una imagen se realiza calculando las derivadas parciales en cada posición de cada píxel  $\frac{df}{dx}$ ;  $\frac{df}{dy}$ . Todo este cálculo se realiza usando máscaras de píxeles que suelen ser de  $2 \times 2$  o  $3 \times 3$  dependiendo del método a usar a partir de ello se hallan los valores de  $G_f G_c$ :

#### **Derivada de segundo orden (Laplaciana):**

$$\nabla^2 f(x, y) = \frac{d^2}{dx^2} f(x, y) + \frac{d^2}{dy^2} f(x, y) \quad (1.5)$$

La Laplaciana vale cero  $f(x, y)$  es constante o cambia linealmente su amplitud. El cambio de signo de la función resultante indica que en este lugar existe un cruce por cero y por tanto indica la presencia de un borde.

### 2.3.2 Unión de bordes y detección de discontinuidades

Los anteriores métodos detectan los bordes existentes en la imagen pero no toman en cuenta la continuidad de dichos píxeles para formar la frontera para ello se proponen los siguientes métodos.

#### 2.3.2.1 Técnica 4: Procesado local

Analiza zonas de píxeles con propiedades parecidas y une dicha regiones formando un borde continuo. Es una deducción en el cómputo del gradiente bajo un umbral determinado.

$$|\nabla f(x,y) - \nabla f(x_0,y_0)| \leq E \quad (1.6)$$

Donde:  $\nabla f(x,y)$  = Valor de la función gradiente en el punto (xy)

$\nabla f(x_0,y_0)$  = Valor de la función gradiente en el punto (x0y0)

E = umbral

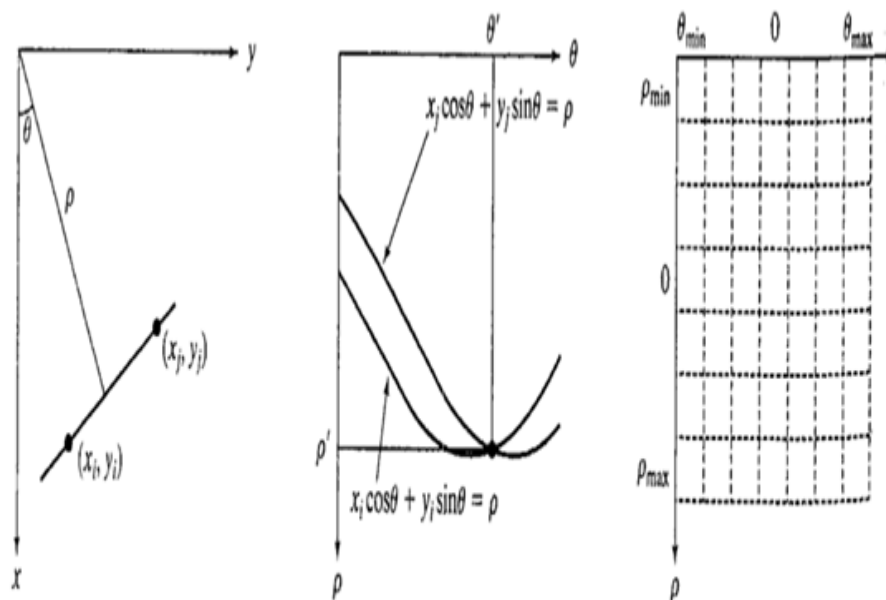
#### 2.3.2.2 Técnica 5: Procesado global vía transformada de Hough

Es una técnica que permite detectar curvas en una imagen trabajando con una imagen binaria (blanco y negro) de los píxeles que forman parte de las fronteras

Para calcular la transformada de Hough se usan coordenadas polares:

$$X \cos(\theta) + y \sin(\theta) = \rho \quad (1.7)$$

La Fig. 2.8 nos muestra la transformación del espacio  $(x,y) \rightarrow$  espacio  $(\rho,\theta)$ :



**Fig.2. 8** Transformación del espacio  $(x,y) \rightarrow$  espacio  $(\rho,\theta)$ .

Fuente: <http://www.640kbits.es>

Esta es la llamada ecuación en la forma normal de la recta donde  $p$  es la distancia de la recta al origen y  $\theta$  es el ángulo entre la perpendicular y el eje  $x$  de esta forma son menos los puntos que hay que recorrer y por lo tanto más rápido es el algoritmo.

Los límites de estos están dados por las siguientes condiciones  $\theta$  varía entre 0 y 180 ya que a partir de 180 hasta 360 grados se vuelven a cruzar las curvas y  $p$  varía en la diagonal de la imagen es decir la

hipotenusa con respecto a los lados de la imagen original. Se suele usar un algoritmo que solucione el problema del enlazado de bordes de manera que:

Se calcula el gradiente de la imagen umbralizando una imagen binaria.

Se especifica las subdivisiones en el plano  $(\rho, \theta)$ :

Estos  $(\rho, \theta)$  valores se almacenan en celdas. La "celda acumuladora"  $A(i,j)$  corresponde al punto  $(x(i), y(j))$  del espacio paraférrico  $(\rho, \theta)$ . Inicialmente las "celdas acumuladoras" tienen un valor 0.

Una recta queda determinada mediante un punto con coordenadas  $((x_i)_{(recta)}, y_{(j)}_{(recta)})$  mientras que un punto se representa como una función senoidal. Si por ejemplo tenemos dos puntos tendremos dos senoides desfasadas  $\theta$  grados dependiendo de las coordenadas de los puntos. Dichas senoides se irán cruzando cada  $180^\circ$ . La interpretación geométrica de este hecho es que la función seno de cada punto representa las infinitas rectas que pasan por cada punto cuando dos puntos comparten la misma recta sus representaciones senoidales se cruzan se obtiene un punto. Cada vez que se da media vuelta ( $\rho=180^\circ$ ) se vuelve a repetir la misma recta por lo que volvemos a obtener otro punto que de hecho es la misma recta. Se



examina las celdas acumulativas de una alta concentración de píxeles.

Se examina las relaciones entre los píxeles de una misma celda.

### 2.3.2.3 Técnica 6: Seguimiento de contornos (teoría de grafos)

Se busca la unión de bordes a través del camino más óptimo entre elementos del grafo. Esta representación proporciona una aproximación fuerte en condiciones de ruido adversas. El procedimiento es complicado y requiere mayor tiempo de computacional que otros métodos.

Se define un borde como la frontera entre dos píxeles  $p$ ,  $q$  denotados en este ejemplo como:  $p = (1,2)$  y  $q = (2,2)$  que son vecinos dentro de una vecindad de cuatro elementos. Dichos elementos se identifican por las coordenadas  $(x, y)$  para cada píxel. Si se observa la Fig. 2. 9 los números de fuera son las coordenadas de los píxeles y los números entre corchetes son los niveles de intensidad de dicho píxeles.

Cada elemento tiene un costo o peso asociado definido de la forma:

$$C_{(x,y)} = H - [f(p) - f(q)] \quad (1.8)$$

H: Es el máximo nivel de Intensidad de la imagen.

$f(p)$  y  $f(q)$ : Son los niveles de intensidad en los puntos  $p$  y  $q$ .

Por convenio se representan los desplazamientos por parejas de izquierda a derecha es decir desde el píxel  $p(1,2)$  al  $q(2,2)$ . Así pues se calculan todos los costos y luego se elige el camino con menor costo o peso.

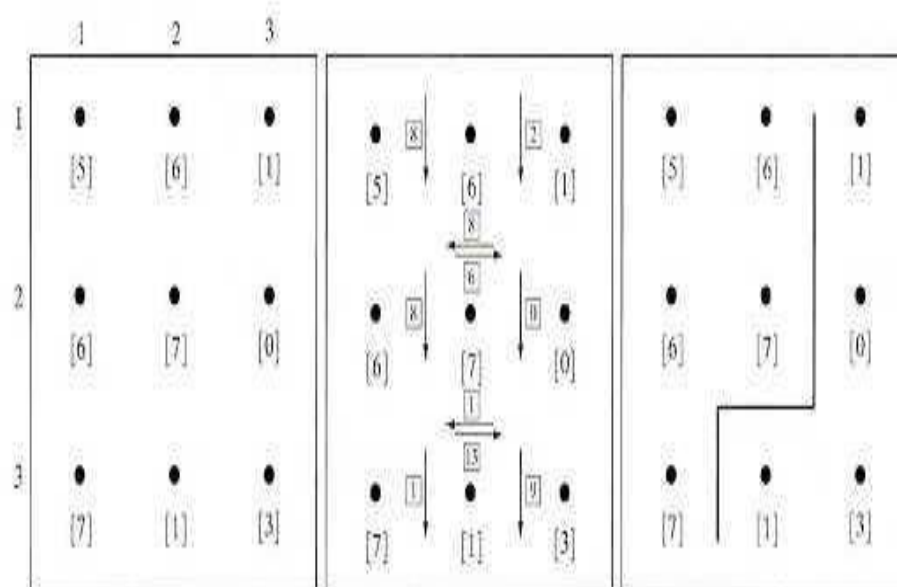


Fig.2. 9 Grafo de elementos de bordes de una imagen

Fuente: Segmentación de color por intersección de histograma 2009

## 2.4 Segmentación por similitud

La segmentación por similitud agrupa los métodos que dividen una imagen comparando las propiedades de diferentes partes de una imagen que forman regiones [1]. Esta técnica se puede clasificar en:

- 1) Umbralización.

2) Crecimiento de regiones.

### **2.4 .1 Técnica 7: Umbralización**

Es una técnica de segmentación rápida ya que tiene un costo computacional bajo. Esta técnica toma como punto de partida un histograma de la imagen. Se trata de convertir una imagen de niveles de grises o color en una imagen binaria de tal forma que los objetos de interés se etiqueten con un valor distinto de los píxeles del fondo. Dentro de la umbralización tenemos los siguientes tipos de umbralización:

#### **Técnica 7.1: Umbralización global**

Consiste en segmentar una imagen donde los objetos (regiones de píxeles) contienen niveles de gris dentro del rango de valores y el fondo tiene píxeles con valores en otro rango disjunto.

#### **Técnica 7.2: Multiumbralización**

Esta técnica consiste en la elección de múltiples valores de umbral dentro del proceso permitiendo separar los distintos objetos dentro de una escena cuyos niveles de gris difieran el resultado no será ahora una imagen binaria sino que los diferentes objetos (regiones) tendrán etiquetas diferentes como lo muestra la imagen (Fig. 2.10):

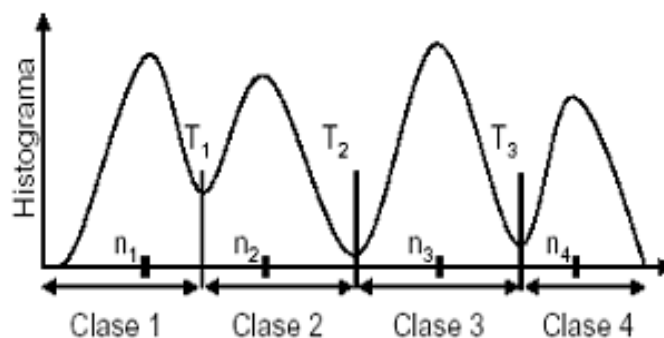


Fig.2. 10 Uso de umbrales múltiples en un imagen

Fuente: Segmentación de color por intersección de histograma 2009

### Técnica 7.3: Umbralización local

La imagen original se divide en sub imágenes y se encuentra un umbral para cada una de ellas por alguno de los métodos de umbralización global.

### Técnica 7.4 Umbralización adaptativa

En las otras técnicas el umbral o umbrales se consideran fijos independientemente de las características locales de la imagen considerada.

Sin embargo en muchas imágenes donde la iluminación no es uniforme o en aquellas donde los objetos sean muy pequeños con respecto al fondo puede ocurrir que píxeles de un mismo objeto a segmentar tengan niveles de gris diferentes. La umbralización adaptativa consigue que el valor del umbral varíe como una función

de las características locales de una imagen. La imagen se divide en sub imágenes y para cada una de ellas se calcula un umbral. Cada sub imagen se procesara según su propio umbral. El histograma de una imagen no tiene en cuenta la información espacial sino solamente la distribución de grises en la imagen por ello dos imágenes diferentes pueden tener el mismo histograma ello hace que los métodos basados en la búsqueda de umbrales mediante análisis de histograma resulten limitados en algunos problemas reales.

#### **2.4.2 Técnica 8: Crecimiento de regiones**

La agregación de píxeles o subregiones en regiones más grandes de acuerdo a unas propiedades comunes y no solo por un nivel de gris determinado es otra de las técnicas de segmentación. Dentro de los métodos para la agregación de píxeles tenemos dos métodos:

##### **2.4.2.1 Técnica 8.1: División y fusión de regiones**

La técnica de segmentación de imágenes conocida como “Split and Merge” (división y fusión) trata de dividir la imagen en regiones uniformes de manera que una región con propiedades no uniformes se divide sucesivamente hasta que sus partes sean uniformes. A la vez que se divide se unen aquellas regiones que sean adyacentes y tengan propiedades similares.

El pilar de este método es la creación de un árbol cuaternario de división de la imagen es decirse comprueba la uniformidad de una imagen dividiéndola en cuatro regiones y aquellas regiones que no sean uniformes se dividen en cuatro subregiones y se comprueban cada una de ellas. Se sigue dividiendo sucesivamente en regiones hasta alcanzar un tamaño mínimo de región. De esta manera se consigue una división total de la imagen en regiones cuadradas (Fig. 2.11).

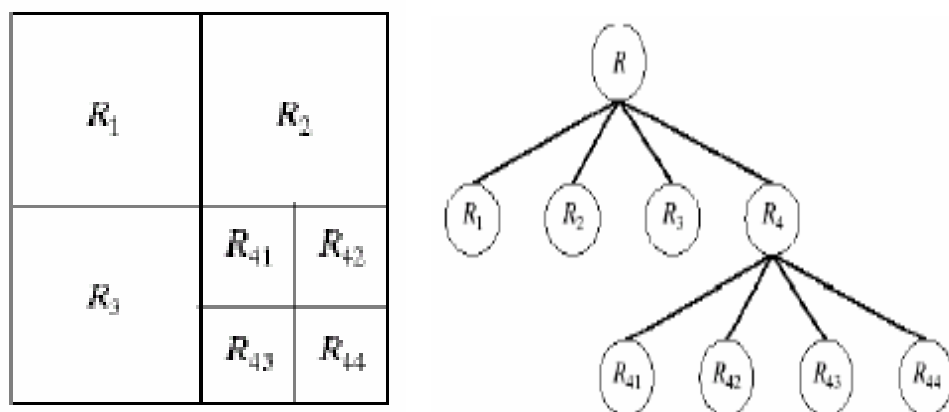


Fig.2. 11a) Imagen dividida en subregiones    b) árbol cuaternario de división

Fuente: Segmentación de color por intersección de histograma 2009

A la vez que se construye el árbol cuaternario, en cada nivel, se unen aquellas regiones que sean adyacentes y tengan propiedades similares (Fig. 2.12).

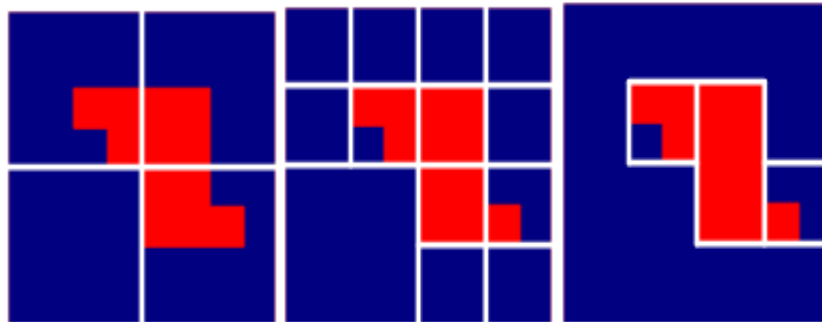


Fig.2. 12 Ejemplo de división y fusión de imágenes

Fuente: Segmentación de color por intersección de histograma 2009

#### 2.4.2.2 Técnica 8.2: Crecimiento de regiones

Se parte de un conjunto de puntos semillas a los que se les van añadiendo píxeles vecinos. Si se usan  $n$  semillas al final se podrá obtener una segmentación con un máximo de  $n$  regiones además del fondo.

Este método estaría formado por tres fases:

1. Elección de los píxeles semillas.

La determinación del punto de partida para la segmentación suele ser un aspecto crítico del proceso ya que va a influir totalmente en la evolución de la segmentación.

Existen métodos completamente automáticos pero suelen resultar imprácticos debido a que la complejidad y variabilidad de las imágenes tiende a limitar el alcance de su aplicabilidad.

2.- Formulación de unas propiedades para la inclusión de píxeles a la región.

La definición de una regla para el crecimiento depende en gran medida del tipo de imagen que se pretenda procesar por ello existen muchos tipos de descriptores que marcan dichas propiedades. Una vez se tengan dichos descriptores se evalúan el conjunto de píxeles asociados a la zona que se pretende agregar y si cumple dicha regla se añade los grupos de píxeles de dicha semilla. Así se va formando la región.

3.- Finalización del crecimiento a través de una regla de parada.

A veces puede resultar difícil dejar de ampliar la región debido a una gran similitud entre todos los píxeles de la imagen por ello la regla de parada es también importante. Se ha de tener en cuenta la historia de ampliaciones que se va haciendo como el número de píxeles de cada región su histograma etc. (Fig.2.13):



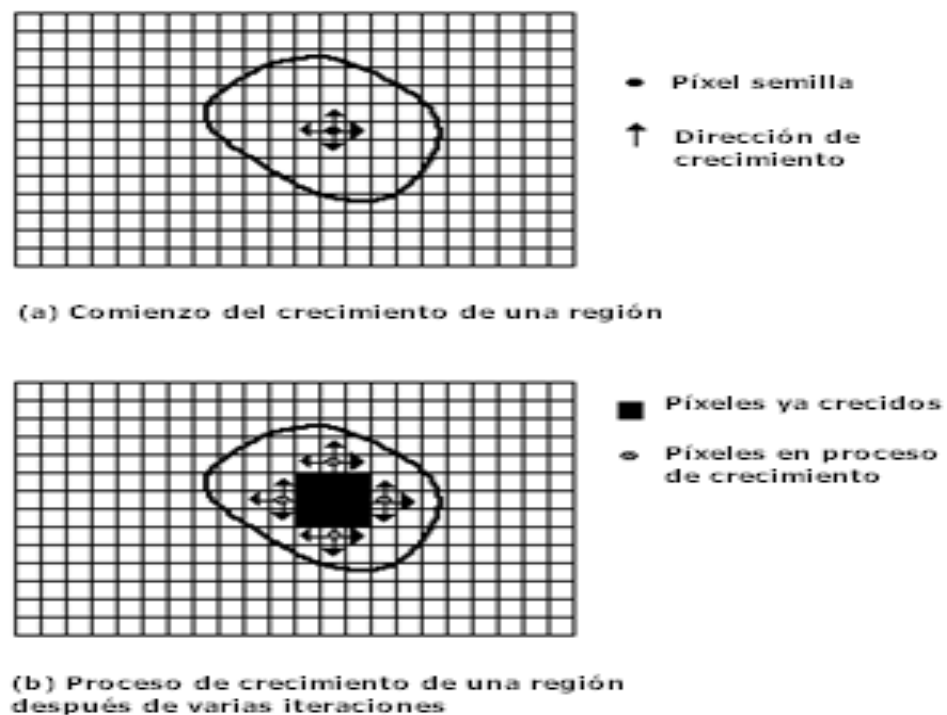


Fig.2. 13 Técnica de crecimiento de regiones

Fuente: Segmentación de color por intersección de histograma 2009

### Vecindad y conectividad

Para el crecimiento de regiones debemos considerar la vecindad y conectividad entre píxeles que se definen a continuación [4]:

La vecindad define la relación entre un píxel y su entorno. El píxel  $p(x,y)$  tiene 2 vecinos verticales y 2 vecinos horizontales conocidos como vecindad de cuatro

(Fig. 2.14).

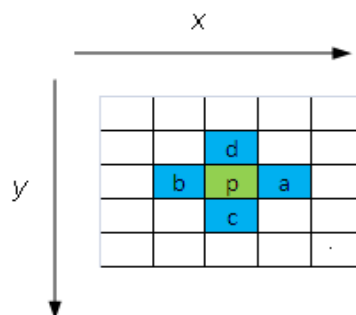


Fig.2. 14 Vecindad vertical y horizontal

Fuente: Segmentación de imágenes médicas para detección de detalles 2009

A este tipo de vecindad también se lo conoce como vecindad directa denotada por  $N_{(4)}(P)$ . [4].

Otro tipo de vecindad es la llamada vecindad diagonal que esta denotada por  $N_D(P)$  y se muestra en la (Fig 2.15):

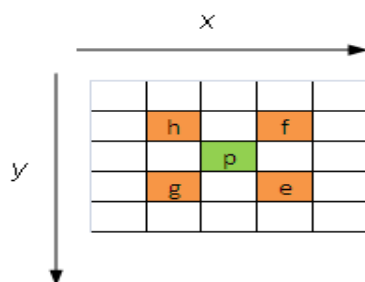


Fig.2. 15 Vecindad diagonal

Fuente: Segmentación de Imágenes médicas para detección de detalles 2009

Si dos píxeles se topan en solo una de sus esquinas se le llaman vecinos indirectos y finalmente tenemos la vecindad de ocho denotado por  $N_{(s)}(P)$  que es la unión de dos vecindades anteriores y se define como:  $N_{(s)}(p) = N_{(4)}(P) \cup N_D(P)$  (Fig 2.16)

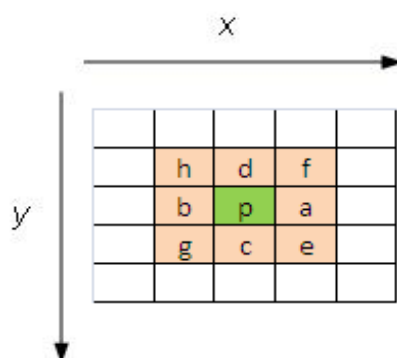


Fig.2. 16 Conectividad a ocho

Fuente: Segmentación de imágenes médicas para detección de detalles 2009

La conectividad entre dos píxeles p y q está dada por los siguientes criterios:

Vecindad: p y q son vecinos directos o indirectos.

Similitud: Si p y q comparten alguna propiedad como por ejemplo el nivel de gris.

Se pueden definir los siguientes tipos de conectividad (Fig. 2.17):

Conectividad 4: Si q pertenece a  $N_{(s)}(P)$

Conectividad a 8: Si  $q$  pertenece a  $N_{(8)}(p)$

Conectividad mezclada: Si  $q$  pertenece a  $N_{(4)}(P)$  o pertenece a  $N_{(D)}(p)$

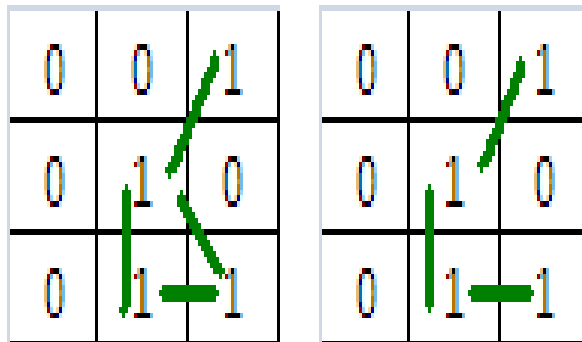


Fig.2. 17Conectividad mezclada

Fuente: Segmentación de imágenes médicas para detección de detalles 2009

## 2.5 Comparativa de los diferentes modelos

A continuación se presenta un pequeño resumen de los diferentes métodos que se usan en la segmentación de imágenes. Esto nos permite decidirse por un modelo de segmentación en concreto [2]:

Técnica	Método	Ventajas	Inconvenientes	
Discontinuidad	Detección de puntos	Es bastante simple.	Solo válido para puntos.	
	Detección de líneas	Es bastante simple	Solo válido para líneas Usa diferentes máscaras en función de la dirección.	
	Detección de bordes	Es bastante simple. Existen varios operadores.	Es sensible al ruido. Presenta ruido en determinadas direcciones según el operador.	
Unión de Bordes	Procesado local	Es simple	Limitado a determinadas direcciones	
	Transformada de Hough	Encuentra rectas de forma precisa	Limitado a rectas y curvas.	
	Teoría de Grafos	Funciona bien ante el ruido	Es complicado. Requiere mucho cálculo computacional.	
Similitud	Umbralización	Banda	Es simple	No funciona bien con imágenes homogéneas
		Multiumbralización	Es simple	Depende de más factores aparte del umbral. Sensible ante la iluminación.
		Adaptativa	Fuerte frente a iluminación variable	Costo computacional elevado.
	Crecimiento de regiones	División y fusión	Buena detección Autónomo	Las imágenes deben ser potencias de 2 Contornos no reales.
		Crecimiento de regiones	Ofrece un resultado muy completo	Necesidad de semilla Necesidad de punto de parada.

**Tabla 1: Comparación de los modelos de segmentación.**

### 2.5.1 Elección del modelo de segmentación

El objetivo es obtener regiones homogéneas en cuanto a sus características.

Para escoger la mejor técnica se pasó por varias etapas tratando de hallar la mejor solución a nuestro problema de segmentación.

Nuestro primer diagrama de bloques para este proceso fue el siguiente

Fig.2.18:

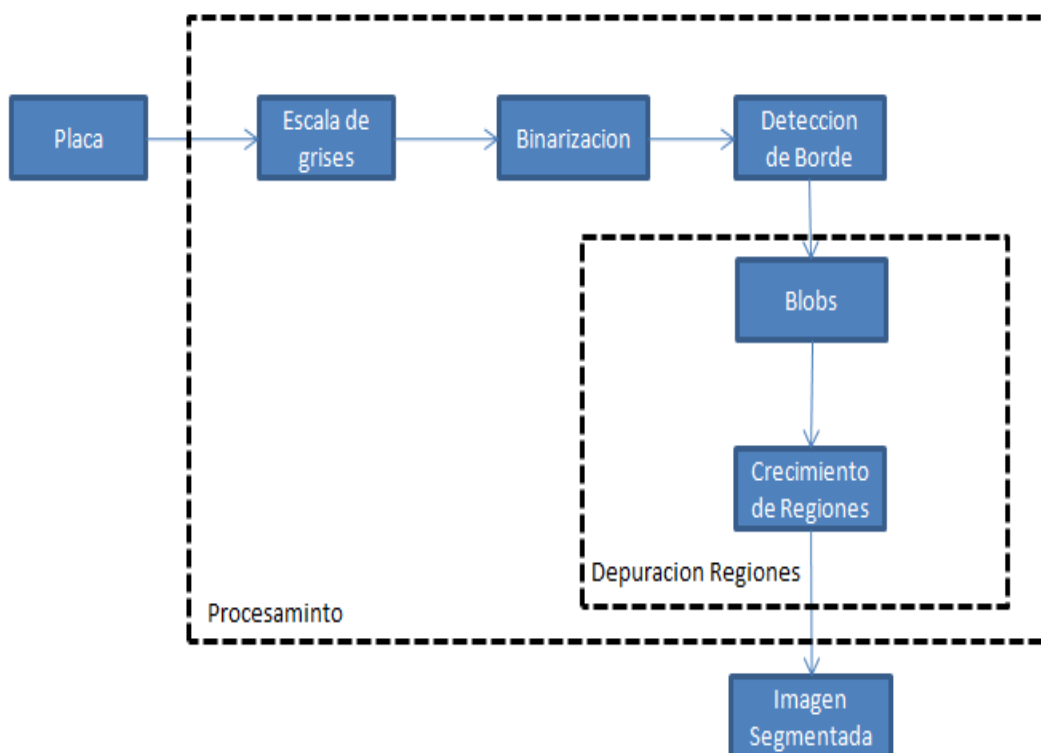
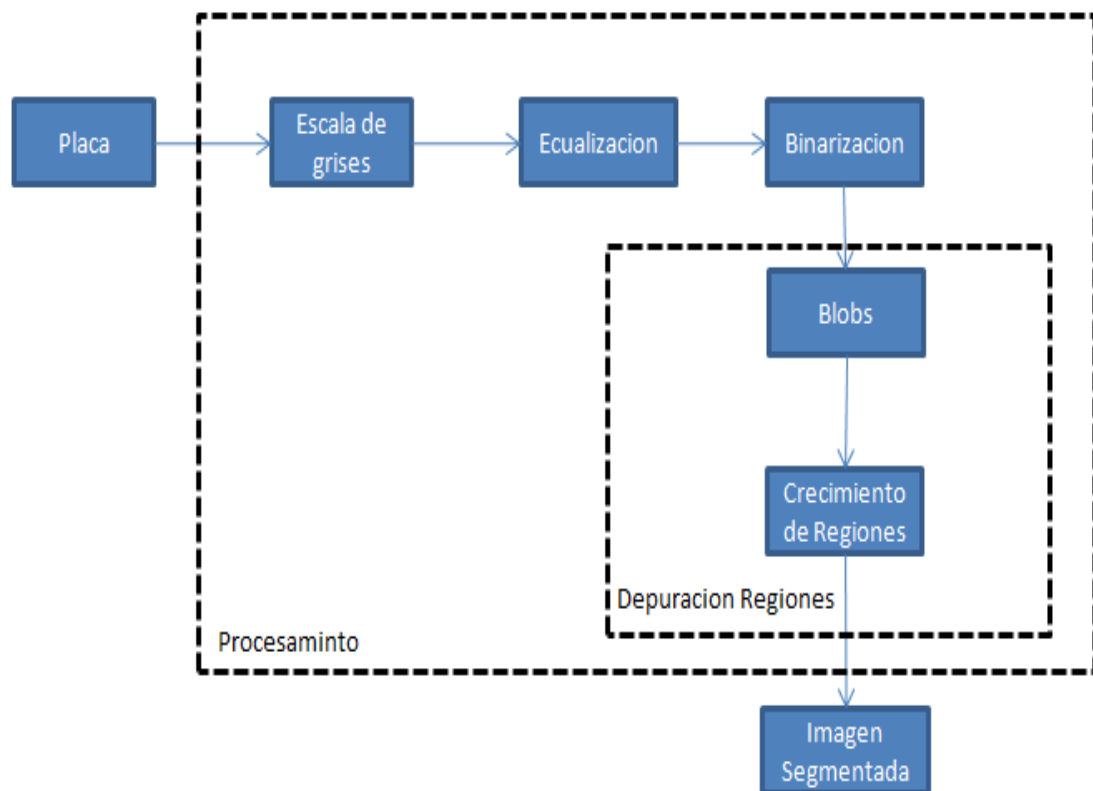


Fig.2. 18 Diagrama de bloque inicial

Pero a medida que fuimos evolucionándonos dimos cuenta que no todos los bloques eran necesarios existía redundancia y a su vez tuvimos que aumentar un bloque de realzado como la ecualización de histogramas para quedarnos de la siguiente manera Fig. 2.19:



**Fig.2. 19 Segundo diagrama de bloques**

Con el diagrama anterior escogemos la mejor técnica de umbralización global para nuestro proyecto entre los cuales podemos mencionar:

**Huang**[\[14\]](#)

Implementa el método de umbral difuso Huang. Esta función utiliza la entropía de Shannon (también se puede utilizar la función de entropía Yager).

#### **Intermodes**[\[14\]](#)

Esto supone un histograma bimodal. El histograma es suavizado iterativamente con un promedio móvil de tamaño 3 hasta que sólo hay dos máximos locales: J y K. El umbral  $t$  se calcula como  $(j + k) / 2$ . Las imágenes con histogramas que tienen picos muy desiguales o un valle amplio y plano no son adecuadas para este método.

#### **Li**[\[14\]](#)

Implementa método mínimos de Cross Li umbral de entropía sobre la base de la versión iterativa del algoritmo.

#### **Otsu**[\[14\]](#)

Algoritmo del umbral de Otsu busca el umbral que minimiza la varianza entre clases que se define como una suma ponderada de las varianzas de las dos clases.

#### **Promedio**[\[14\]](#)



Utiliza la media de niveles de gris como el umbral. Es utilizado por otros métodos como el umbral de primera aproximación.

### **Máxima Entropía**[\[14\]](#)

Implementa el método de umbral de Kapur-Sahoo-Wong (máxima entropía)

### **Momentos**[\[14\]](#)

Método de Tsai preserva los momentos de una imagen original en el resultado de un umbral.

### **Mínimo Error**[\[14\]](#)

Un proceso iterativo de la aplicación de Kittler y un umbral mínimo error (Illingworth). Esta aplicación parece converger con más frecuencia que el original. Sin embargo a veces el algoritmo no converge a una solución. En este caso una advertencia se informa a la ventana de registro y los valores por defecto resultado de la estimación inicial del umbral de la cual se calcula utilizando el método de la media. Ignorar el color negro o blanco ignorar las opciones podría ayudar a evitar este problema.

Una vez probado los métodos nos daba los siguientes resultados como muestra la Fig.2.20:

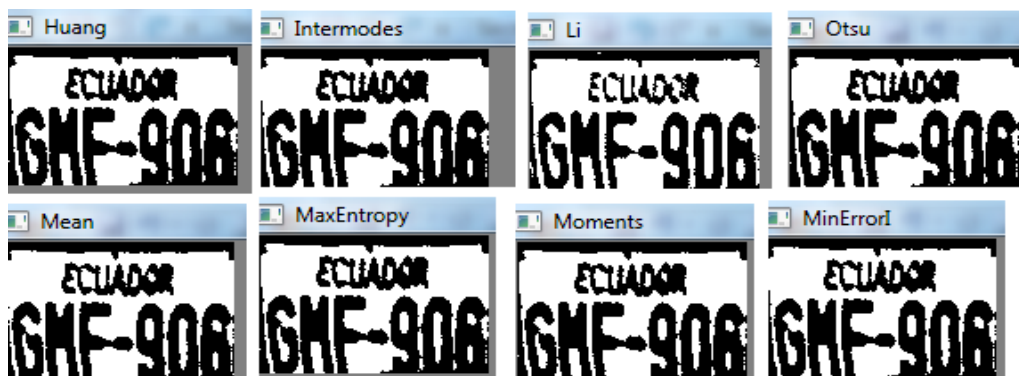


Fig.2. 20 Resultados de métodos de umbralización iniciales

A continuación mostramos los siguientes resultados de todas nuestras imágenes

Fig. .2.21:

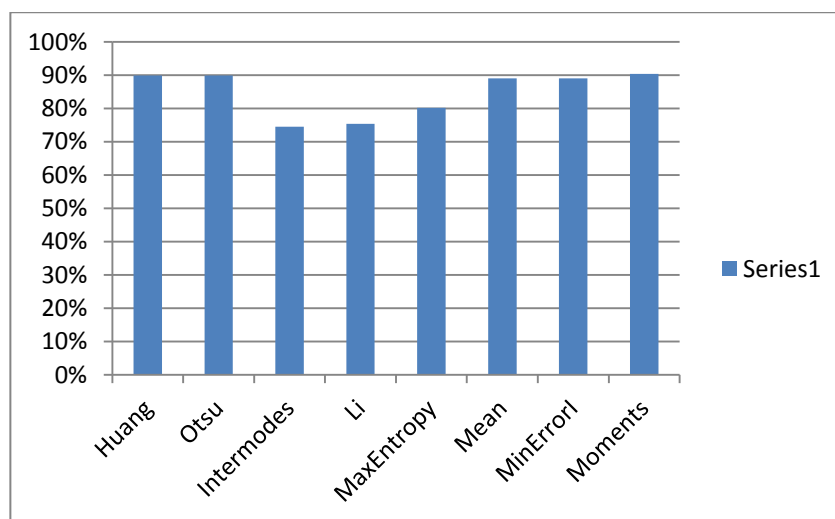


Fig.2. 21 Porcentajes de efectividad de métodos de umbralización

Según nuestras imágenes resultantes teníamos otro problema el nivel de sombra o también llamado *shadow* para resolver aquello recurrimos a otra técnica de umbralización llamada local pero debemos escoger en cuántas

regiones deberíamos dividir la imagen lo cual lo hicimos de manera experimental desde una división de 5x3 5x4 5x5 6x3 6x4 6x5 7x3 7x4 7x5 8x3 8x4 8x5 con los siguientes resultados:

Técnicas de umbralizacion	Tamaño de regiones											
	5x3	5x4	5x5	6x3	6x4	6x5	7x3	7x4	7x5	8x3	8x4	8x5
Huang	17	67	87	30	20	27	53	73	77	30	40	50
Otsu	93	93	97	97	90	90	97	97	97	93	90	90
Intermodes	27	33	37	33	23	13	20	23	6.7	6.7	10	6.7
Li	97	97	97	97	97	97	97	97	97	97	97	97
Max Entropy	0	0	0	0	0	0	0	0	0	0	0	0
Mean	53	73	63	63	83	57	77	80	60	47	67	50
Min Error	0	0	3.3	0	0	3.3	0	3.3	3.3	3.3	3.3	3.3

**Tabla 2: Tamaños de la imagen según algoritmo**

Mostrándonos imágenes como las siguientes (Fig. 2.22):

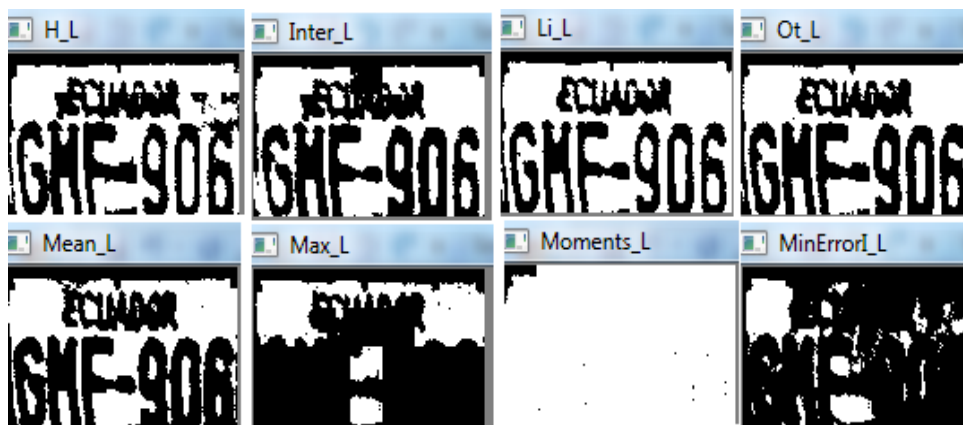


Fig.2. 22: Resultados de métodos de umbralización secundarios.

Como vemos los mejores métodos son el de Li y Otsu y su mejor división fue de 7x4 regiones que era donde mayor porcentaje se obtuvo en relación a todos los métodos.

Para la umbralización de la imagen utilizamos el método de Otsu que en base de los resultados obtenidos ya que nos permitió una mejor calidad en la imagen de la placa.

Una vez obtenido los resultados el siguiente paso fue trabajar con blobs pero el problema ahora es que en la mayoría de resultados, teníamos caracteres pegados o problemas de sombras propias del método de umbralización y que en un siguiente nivel sería mucho más complicado resolverlo por esto se estudió mejor las técnicas y para obtener un resultado más inteligente se aplicó la técnica de crecimiento de regiones la cual nos da la posibilidad de

crecer hasta cumplir cierta regla , y a su vez trae embebido la parte de blobs que se detallará en el capítulo 4.

## 2.6 Historia y estructura de la librería OpenCv

A continuación una breve descripción y diferentes aplicaciones de OpenCv la cual hemos utilizado como librería para el procesamiento y análisis de las imágenes en el presente trabajo [[17](#)].

La librería OpenCV (Open Computer Vision Library ) fue creada en el año 2000 por Intel® Corporation. OpenCV es una librería Open Source para la visión artificial o visión por computadora escrita en C y C++. Corre bajo Linux, Windows y Mac OS X. También se puede utilizar con lenguajes como Python,Ruby,Matlab entre otros.

Con el objetivo de que OpenCV sea utilizada para promover el uso comercial y la investigación es abierta y libre. Esto quiere decir que puede ser embebida completa o parcialmente en otras aplicaciones sin la obligación de que esas aplicaciones sean abiertas o libres. Esta librería puede ser encontrada en SourceForge.net donde podrán descargarla o ir a la documentación en línea, la cual es muy completa.

### **Aplicaciones de OpenCv:**

- **Detección reconocimiento y rastreo de objetos**

En este campo de aplicación algunos usos puntuales son por ejemplo: la detección de objetos "olvidados" en un aeropuerto como valijas, bolsas, paquetes, etc. Esto ayuda al personal de seguridad a tomar acciones preventivas contra atentados terroristas.

- **Detección de movimiento:**

En sistemas de seguridad en oficinas, bancos, comercios etc.

- **Grabación de frames para sistemas de seguridad:**

Se dispone una webcam conectada a una PC y el software con OpenCV es capaz de grabar aquellos frames donde se ha detectado movimiento para que luego el personal de seguridad analice las imágenes para saber quién o qué "anduvo" por el lugar.

- **Reconocimiento de rostros:**

El reconocimiento de rostros trae aparejada una gran cantidad de tareas que combinadas logran el objetivo.

El software de reconocimiento debe "saber" qué características buscar dentro de una imagen y luego comparar esas características con la imagen para lograr coincidencias y si las hubiere determinar si es o no un rostro. Un software sencillo solo debería considerar que un rostro posee dos ojos nariz y boca todo circunscripto en una figura

geométrica que bien podría ser un círculo o elipse.

Ahora bien ,el truco está en que dependiendo del ángulo de la cámara el rostro y todo su "contenido" se deforma ,es decir adquiere perspectiva con lo cual el reconocimiento puede complicarse.

También debe considerar el tamaño tiene que poder saber que un rostro pequeño y uno grande siguen siendo rostros no solo porque puede ser el de un niño o el de un adulto sino que puede ser un rostro alejado de la cámara o uno muy cercano .Las técnicas para el reconocimiento de rostros son diversas y pueden utilizarla personas que no tiene acceso a la tecnología por ejemplo a una PC debido a que por algún motivo le es imposible utilizar sus manos ya sea porque le fueron amputadas o por algún tipo de parálisis. Este tipo de software podría ayudar a estas personas.

## **2.7 Estructura de OpenCV**

La librería OpenCv está dirigida fundamentalmente a la visión por computador en tiempo real [\[5\]](#).En la Fig.2.23 se muestra la estructura de la librería OpenCv.

La librería OpenCv proporciona varios paquetes de alto nivel para el desarrollo de aplicaciones de visión. Todos ellos se pueden agrupar en librerías de C/C++ dirigidas a usuarios avanzados a usuarios de nivel medio





Además utilizaremos Visual Studio 2008 y específicamente Visual C++ para la realización de este proyecto.

## 2.8 Glosario

### Píxel [\[10\]](#)

Elemento más pequeño en que puede dividirse una imagen digital la superficie real que representa cada uno de ellos define los objetos o detalles más pequeños que pueden observarse en una imagen. El cual es utilizado por las cámaras por la limitación en la visión humana que lo percibe como un conjunto y no como unidades independientes Fig. 2.24.

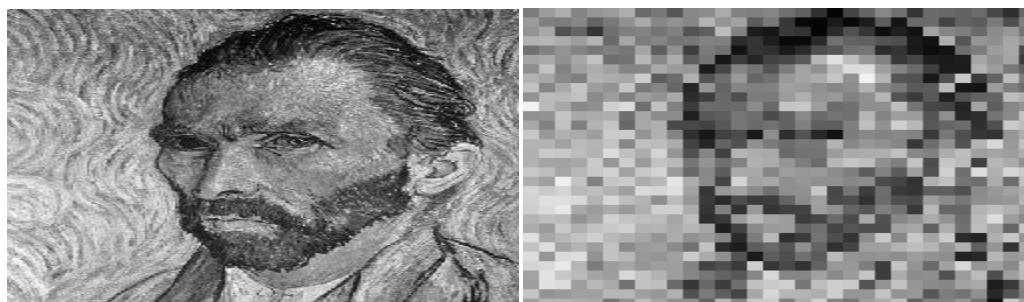


Fig.2. 24 Imagen original y zoom de la imagen

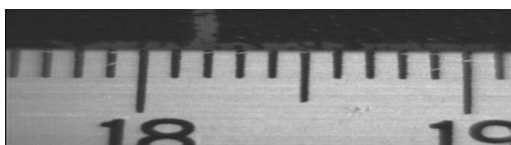
Fuente: <http://www.dimages.es/Tutorial%20A./introduccion/resolucion.htm#top>

### Histograma [\[11\]](#)

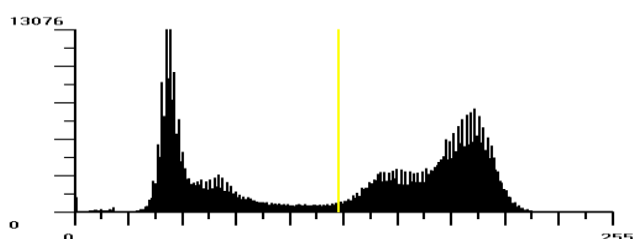
Un histograma es un gráfico que muestra la distribución de los colores o tonos de un color en una imagen según su luminosidad. En el histograma, el eje horizontal indica la luminosidad (más a la izquierda, más oscuro y más a la

derecha, más luminoso). El eje vertical indica la cantidad de píxeles con esa luminosidad. Un pico en nuestro histograma en el lado izquierdo indica un gran número de píxeles que están oscuros o negros (posiblemente una foto subexpuesta) mientras que un pico en la parte derecha indica un gran número de luminosos o blancos (posiblemente una foto sobreexpuesta). Por este razonamiento un histograma uniforme (sin picos) en todos los tonos es probable que indique que la imagen está debidamente expuesta.

La Fig. 2.25 y la Fig.2.26 nos muestra una imagen monocromática y su histograma [10].



**Fig.2. 25** Imagen monocroma codificada en 256 niveles de intensidad (niveles de gris) y su correspondiente histograma de intensidades o niveles de gris.



**Fig.2. 26** La línea amarilla en el histograma indica el valor medio de intensidad (valor medio de nivel de gris).

Fuente: [http://www.dimages.es/Tutorial%20A.I/enhancement/marcos\\_enhan.htm](http://www.dimages.es/Tutorial%20A.I/enhancement/marcos_enhan.htm)

El histograma muestra que la distribución de intensidades no cubre todo el rango posible de los 256 niveles. También muestra una distribución multimodal que corresponde a los distintos tipos de objetos que aparecen en la imagen: fondo de la regla números grabados en la misma y fondo de la imagen.

**Nivel de gris:** Luminosidad o intensidad del píxel que va de 0 (negro) a 255 (blanco). El tono de gris de cada píxel se puede obtener bien asignándole un valor de brillo que va de 0 (negro) a 255 (blanco) bien como porcentajes de tinta negra (0% es igual a blanco y 100% es igual a negro). Las imágenes producidas con escáneres en blanco y negro o en escala de grises se visualizan normalmente en el modo escala de grises. Este modo maneja un solo canal (el negro) para trabajar con imágenes monocromáticas de 256 tonos de gris entre el blanco y el negro Fig. 2.27 [12].

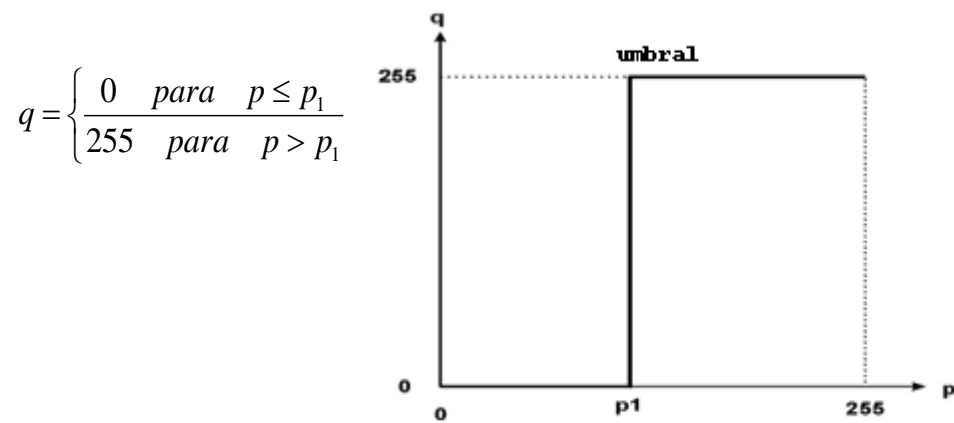


Fig.2. 27Imagen en modo escala de grises

Fuente: <http://www.desarrolloweb.com/articulos/1778.php>

### Umbralización [9].

También conocida como *thresholding* es una técnica de segmentación de imágenes en la que cada píxel pertenece obligatoriamente a un segmento y sólo uno. Si el nivel de gris del píxel es menor o igual al umbral se pone a cero si es menor se pone a 255. En función del valor umbral que se escoja el tamaño de los objetos irá oscilando Fig. 2.28 [9].



a)

b)

Fig.2. 28 a) Función de transformación b) grafica del operador umbral.

Fuente: Procesamiento Digital de imágenes-Oliver Rojas -Mayo 2009

La Fig. 2.29 muestra la oscilación de la calidad de la imagen con respecto a un valor de umbral [13].



Fig.2. 29a) Imagen original b) segmentación con umbral de 30 c) segmentación con umbral de 52

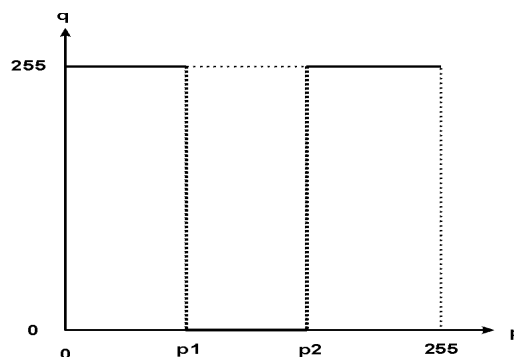
Fuente: [http://es.wikipedia.org/wiki/Metodo\\_del\\_valor\\_umbral](http://es.wikipedia.org/wiki/Metodo_del_valor_umbral)

### Binarización [9]

Es una variante de la umbralización crea una imagen de salida binaria a partir de una imagen de grises donde todos los valores de gris cuyo nivel está en el intervalo definido por  $p_1$  y  $p_2$  son transformados a 255 y todos los valores fuera de ese intervalo a 0 Fig. 2.30.

$$q = \begin{cases} 255 & \text{para } p \leq p_1 \text{ ó } p \geq p_2 \\ 0 & \text{para } p_1 < p < p_2 \end{cases}$$

a)



b)

Fig.2. 30 a) Función de transformación b) operador intervalo de umbral binario

Fuente: Procesamiento Digital de imágenes-Oliver Rojas -Mayo 2009

## CAPÍTULO 3

### DETECCIÓN DE PLACA

En este capítulo se describe el procedimiento seguido para la detección de la placa la cual posteriormente será utilizada en el proceso de segmentación.

El reconocimiento automático de matrículas (*Automatic number plate recognition* o *ANPR* en inglés) es un método de vigilancia en masa que utiliza reconocimiento óptico de caracteres en imágenes para leer las matrículas de los vehículos

Iniciamos mencionando las características de la imagen que hemos utilizado en este proyecto, posteriormente para detectar la placa se describe el algoritmo propuesto el cual ha sido dividido en dos etapas. Tenemos primeramente la etapa de pre procesamiento el cual se ha dividido en tres módulos los cuales son: conversión a nivel de gris, ecualización y umbralización que son descritos en la sección 3.4 obteniendo como resultado de esta etapa una imagen umbralizada. Dicha imagen será utilizada para la etapa de detección de placa, el que se ha dividido en cuatro módulos los cuales son: detección de contornos, selección de rectángulos, corrección geométrica y selección de placa para finalmente obtener nuestra área de interés en este caso la placa.

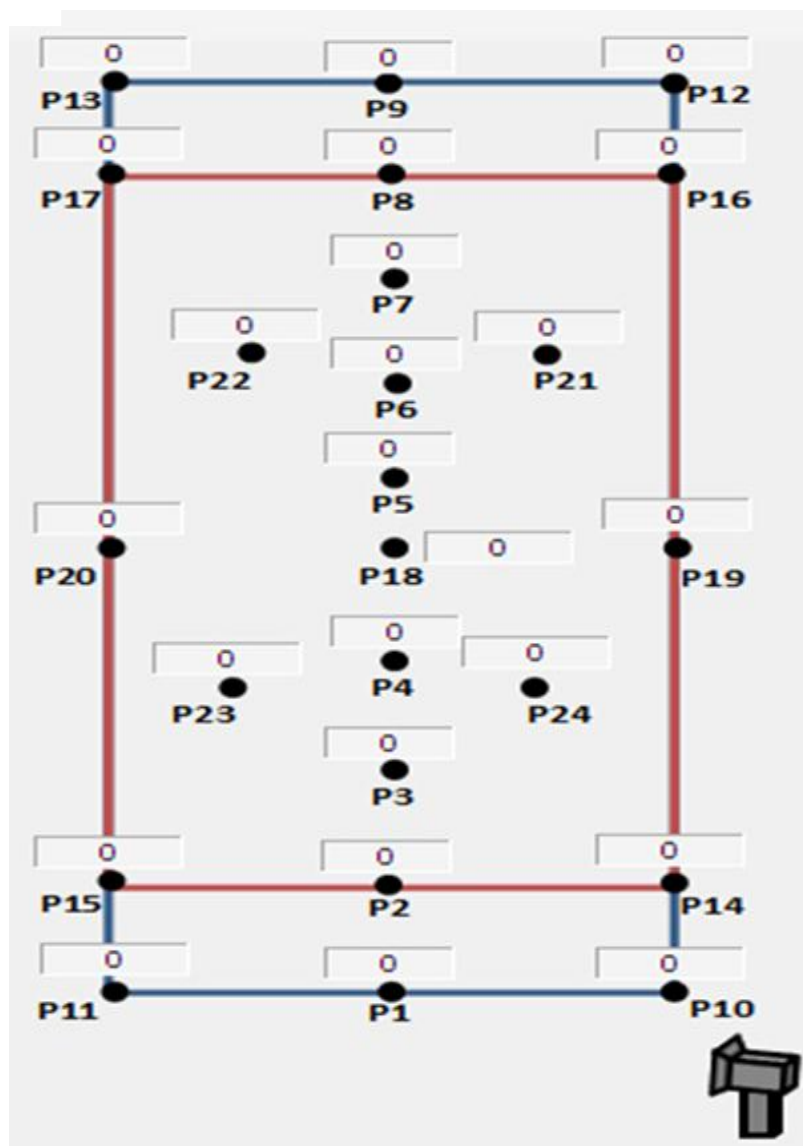
### 3.1 Características de la imagen

Cabe destacar que las características de la imagen con la que se trabaja son las siguientes:

- Las dimensiones de la placa con la que trabajaremos es de 30 cm\*15 cm.
- Las imágenes originales han sido tomadas mediante una cámara previamente calibrada y tienen formato ppm. La cámara utilizada en el reconocimiento de placas vehiculares es de marca BOSCH con modelo REG-L1 –S8XC16 y tiene una resolución de 600 líneas de TV.
- Estas imágenes fueron tomadas a diferente distancia de enfoque. Para nuestro trabajo se usaran 24 puntos y se adquirieron 10 imágenes para cada punto tal se muestra en la Fig. 3.1:

Carril # 2

Carril # 1



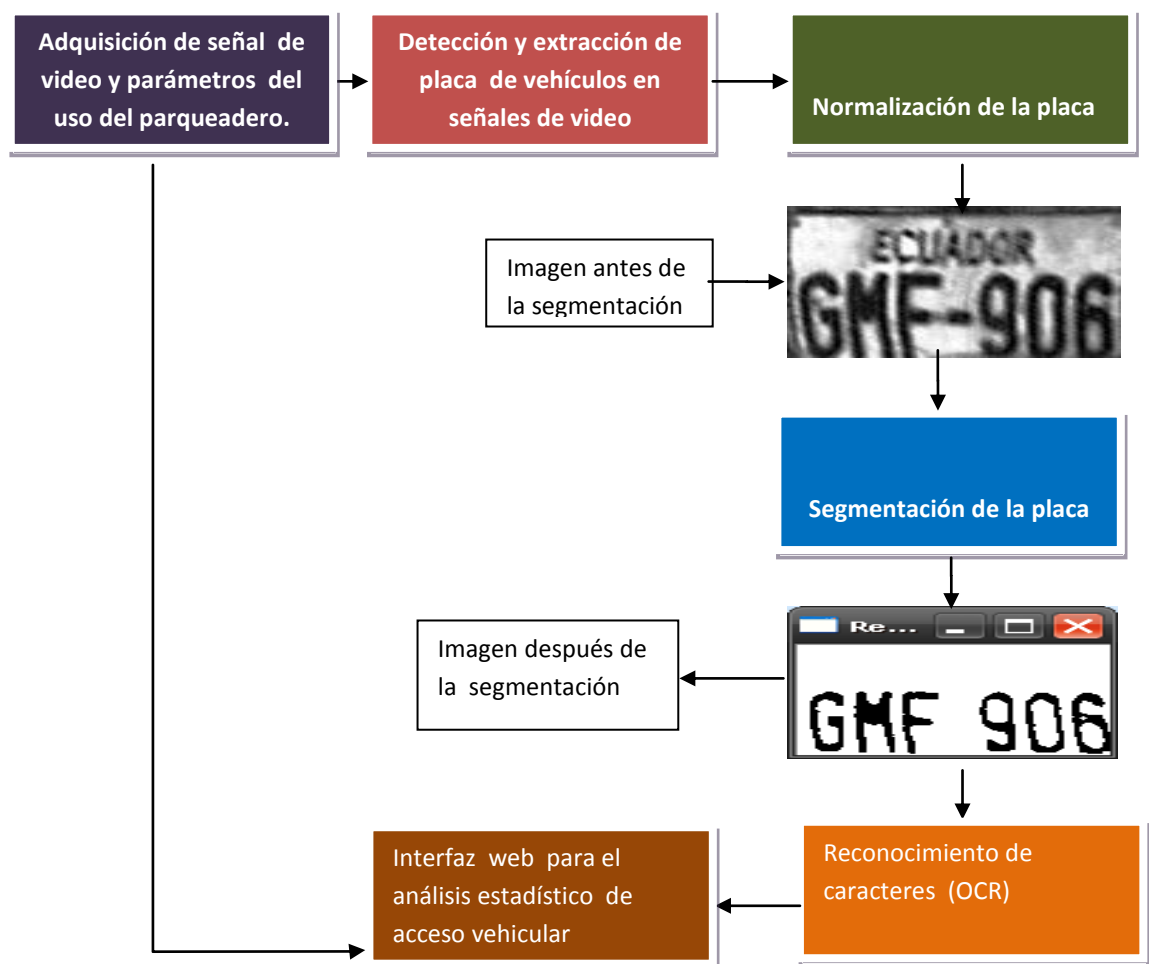
Cámara

Fig.3. 1 Distancias de enfoques utilizados para la adquisición de las imágenes ,24 distancias de enfoque.



Cabe mencionar que la segmentación es parte de un proyecto global que va desde la obtención de la imagen mediante una cámara para después la imagen ser normalizada y posteriormente segmentar la placa la cual será enviada a un proceso de reconocimiento de caracteres (OCR) y finalmente una interface web que permite el análisis estadístico de los datos generados.

La Fig. 3.2 muestra un esquema de los módulos de todo el proyecto global.



**Fig.3. 2 Esquema general del proyecto global.**

En la Fig. 3.6 se describe un diagrama de cada uno de los módulos de la detección de la placa dada una imagen de entrada. Para una mejor visualización de los resultados de la ejecución del programa los distribuimos en diferentes carpetas (ver Anexo 1 [literal I](#)).

La Fig. 3.3 muestra el alto de la placa en la imagen para los puntos 1 al 12 teniendo en el eje x denota el punto y el número de la toma en ese punto (Ej. en el eje x (210): Punto 2 de los 24 de la imagen prueba número 10 en el punto 2). El ancho de la imagen no es mostrado ya que lo que nos interesa es la relación alto placa con respecto al alto de la imagen.

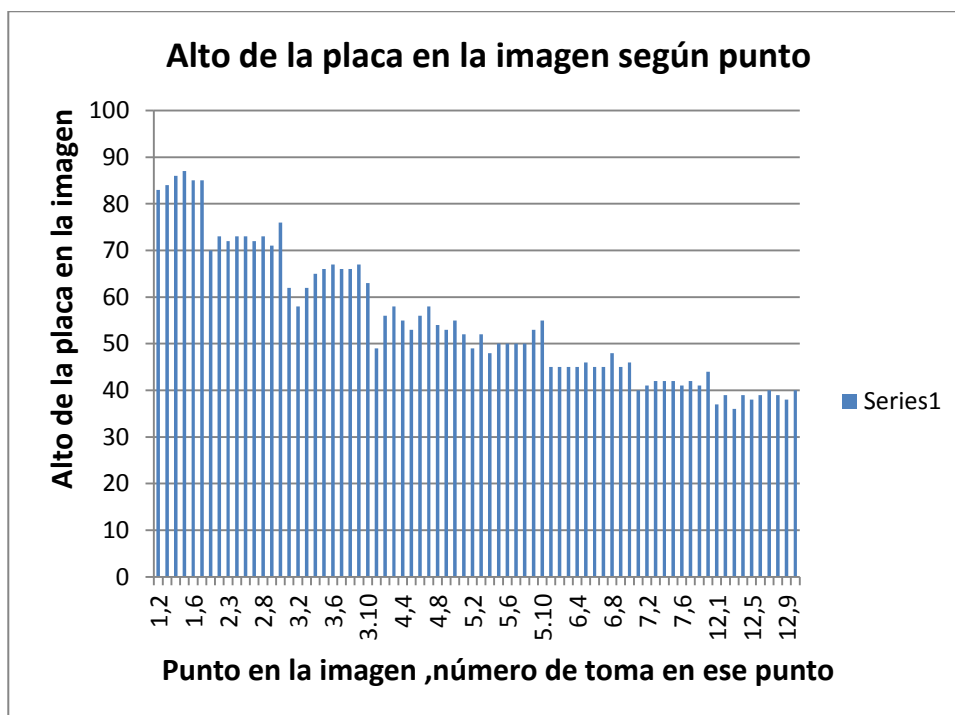
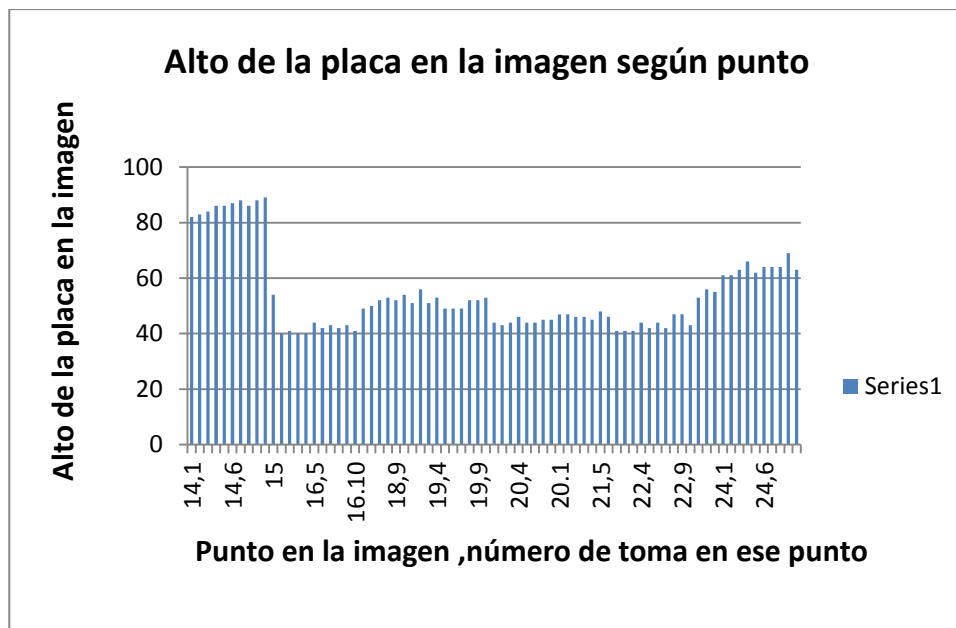


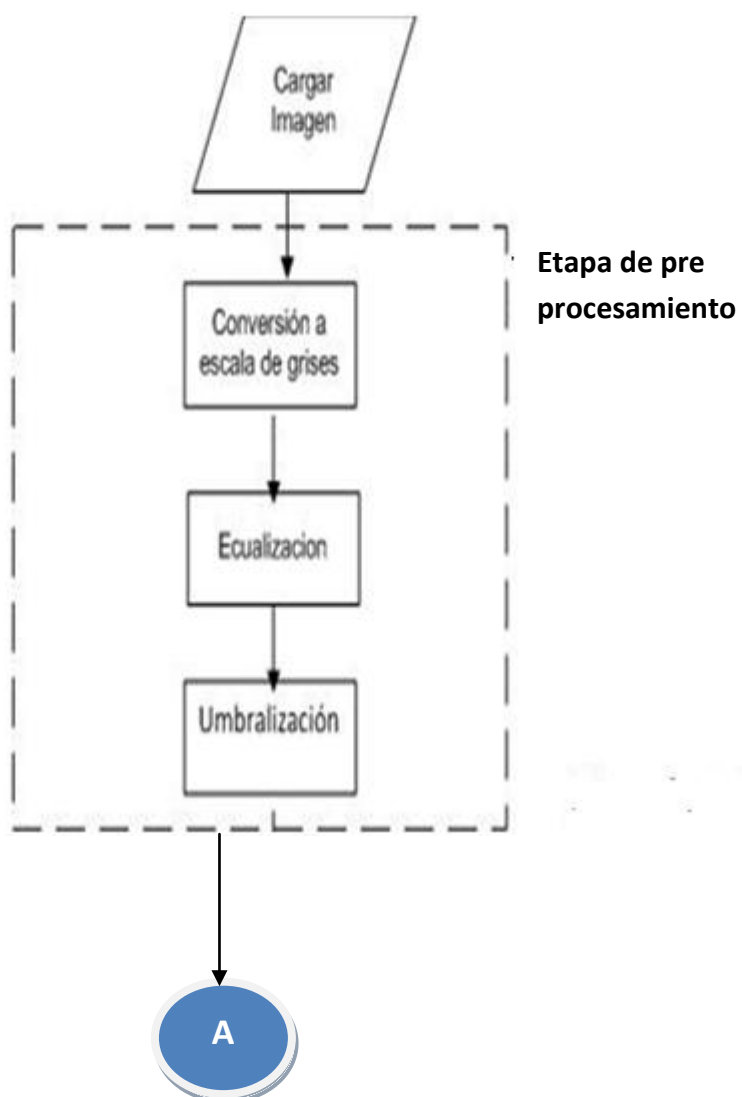
Fig.3. 3Alto de la placa en la imagen para cada punto, puntos analizados 1-12.

La Fig. 3.4 muestra el alto de la placa en la imagen para los puntos 12 al 24.



### 3.2 Diagrama de flujo para la detección de la placa

Se muestra el diagrama de flujo utilizado para la detección de la placa:



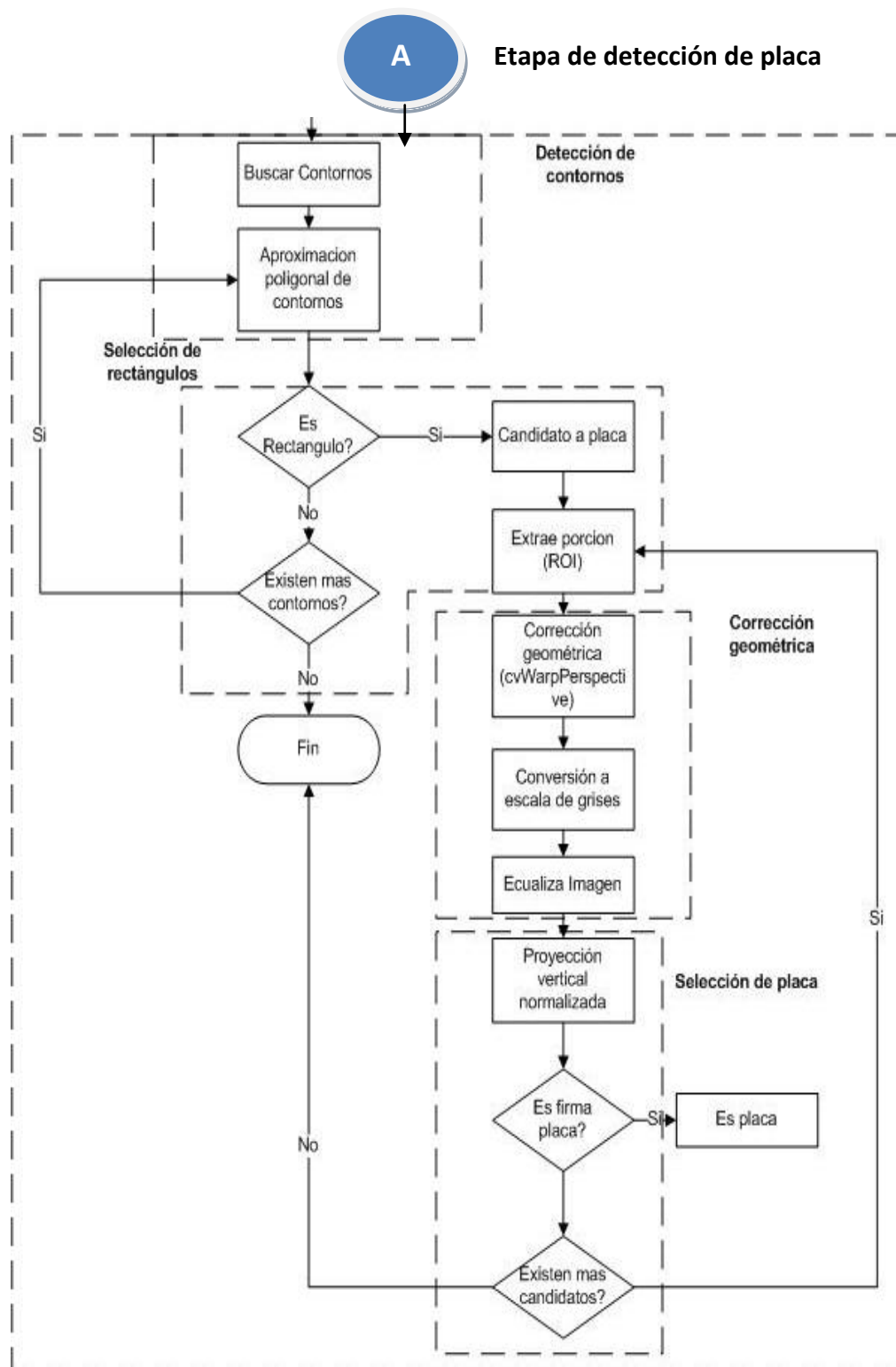


Fig.3. 5Diagrama de flujo para la detección de la placa.

### 3.3 Diagrama de módulos para la detección de la placa

Una vez recibida la imagen ésta será procesada hasta detectar dentro de la misma el área de interés en este caso la placa. En la Fig. 3.6 se muestran los diferentes módulos que conllevan esta etapa.

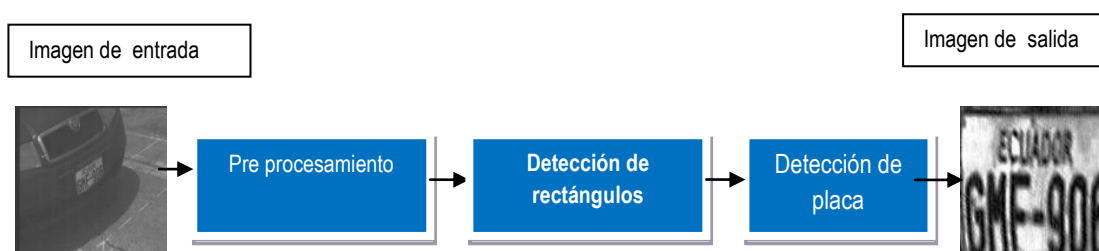


Fig.3. 6 Diagrama de módulos para la detección de la placa.

### 3.3 Implementación

El objetivo de esta implementación es presentar cómo se desarrolló el algoritmo de detección y corrección de la placa de una imagen tomada como referencia de un vehículo.

Para esta implementación se necesitó un conjunto de imágenes de entrada con la finalidad de hacer pruebas en cada etapa y de esta forma ir obteniendo los resultados. Se trabajó con un total de 240 imágenes poniendo mayor énfasis en que los contornos de la placa sean detectados para así obtener un buen resultado.

En la Fig. 3.6 se muestra de una manera general las dos etapas que comprenden el algoritmo de detección de placa. Esta etapa son la etapa de pre procesamiento y la etapa de detección de placa en sí.

### 3.4 Etapa de pre procesamiento de la imagen.

El pre-procesamiento de una imagen comprende tres módulos estos son:(1) conversión a nivel de gris (2) ecualización y (3) umbralización que según nuestro diagrama de la Fig. 3.5 son los primeros pasos. La Fig.3.7 muestra el diagrama para esta etapa.

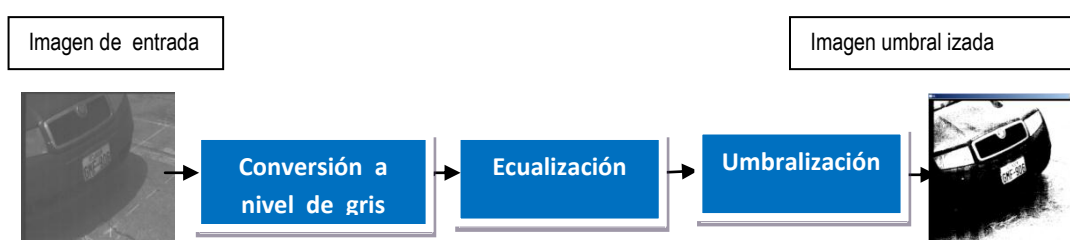


Fig.3. 7Diagrama de módulos para la etapa de pre procesamiento

#### 3.4 .1 Conversión a nivel de gris

Funciones definidas en OpenCv se encargan de leer la imagen de entrada y almacenarla en memoria. Una vez ahí, la imagen en color es representada usando tres bytes por píxel siendo un byte para el componente rojo uno para el verde y uno para el azul.

La conversión a nivel de gris se realiza simplemente copiando los bytes correspondientes a un componente de color a un nuevo espacio de memoria.

Este espacio representa una imagen en escala de gris donde cada píxel es representado usando un byte lo que permite 256 niveles de grises.

La función de OpenCv que nos permite realizar esta conversión es:

**void CvCvtColor ( constCvArr\* src, CvArr\* dst,intcode)** (Esta función esta descrita en el Anexo 1 [literal a](#))

La Fig. 3.8 nos muestra la conversión a nivel de gris de una imagen a color y que será posteriormente ecualizada aplicando funciones de OpenCv.



a) Imagen original

b) imagen convertida a nivel de gris

Fig.3. 8 Conversión de una imagen a color a su correspondiente imagen de nivel de gris

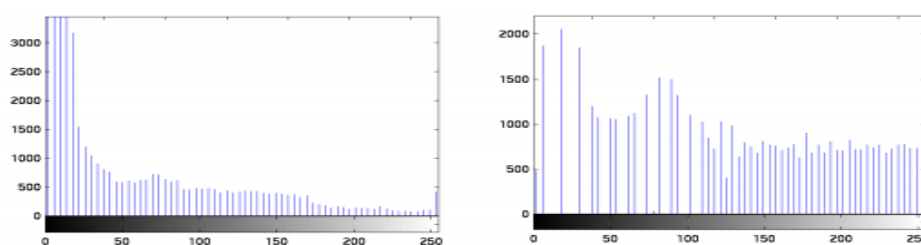
### 3.4.2 Ecuación

Es una forma de manipulación del histograma de la imagen que reduce automáticamente el contraste en las áreas muy claras o muy oscuras de una imagen. También expande los niveles de gris a lo largo de todo el intervalo.



Consiste en una transformación no lineal que considera la distribución acumulada de la imagen original para generar una imagen resultante cuyo histograma será aproximadamente uniforme tal como se visualiza en el ejemplo de la Fig. 3. 9. La función que nos permite realizar esto es:

**CvEqualizeHist( CvArr\* src, CvArr\* dst )** (Esta función esta descrita en el Anexo 1 [literal b](#))



a) Histograma de imagen entrada

b) histograma resultante

Fig.3. 9Ecuación de histogramas

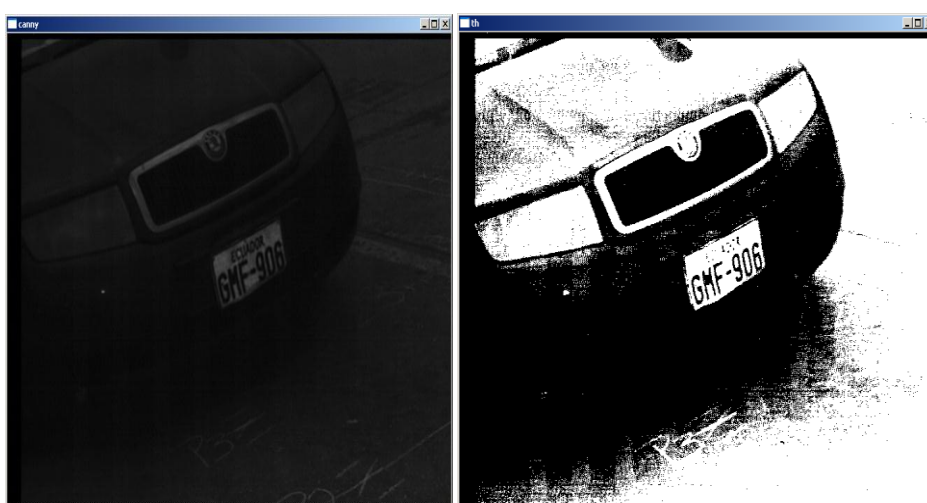
### 3.4.3 Umbralización

Para el proceso de umbralización o binarización se utilizó el algoritmo de Otsu[8] ya que fue el que mejores resultados presentó en relación a varios algoritmos propuestos. Dada esta imagen umbralizada a solo dos niveles de gris (0 y 1) esto nos permite localizar de una manera más fácil el objeto de interés (la placa) y separarla del fondo.

Para ello usamos la siguiente función de OpenCv:

`CvThreshold(constCvArr *src, CvArr *dst, double threshold , double max_value, intthreshold type)`(Esta función esta descrita en el Anexo [1literal c](#))

La Fig. 3.10 nos muestra la imagen después de la aplicación de la umbralización:



a)Imagen de entrada( convertida a nivel de gris) b)Imagen de salida(umbralizada)

**Fig.3. 10 Imagen umbralizada**

El método de Otsu usado en la umbralización de la imagen puede describirse de la siguiente manera:

Puede modelarse un umbral simple de una imagen de un solo canal a partir de la expresión [7]:

$$g(x, y) = \begin{cases} 1 & \text{si } f(x, y) \geq T \\ 0 & \text{por otro lado} \end{cases}$$

Donde T sería el valor dado para la umbralización y  $f(x, y)$  la imagen origen y  $g(x, y)$  la imagen umbralizada,  $(x, y)$  representan columnas y filas respectivamente.

La Fig. 3.11 muestra un ejemplo de la aplicación de la técnica de Otsu :



a) Imagen original de placa b) imagen de placa umbralizada.

**Fig.3. 11 Ejemplo de umbralizacion usando Otsu.**

Un histograma puede usarse para diferenciar los niveles de grises de la imagen (Fig. 3.12).

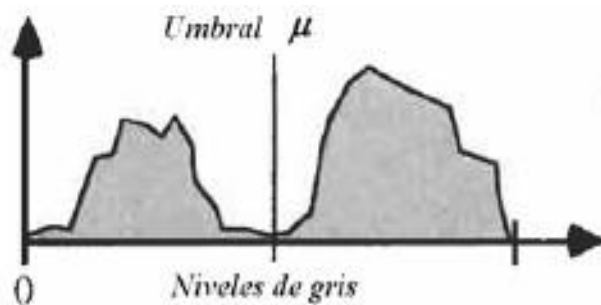


Fig.3. 12 Descripción de histograma

Fuente: Visión artificial – umbralización 2011]

El método de Otsu calcula en umbral que minimiza la varianza de los píxeles de las dos clases (Fig.3.13).

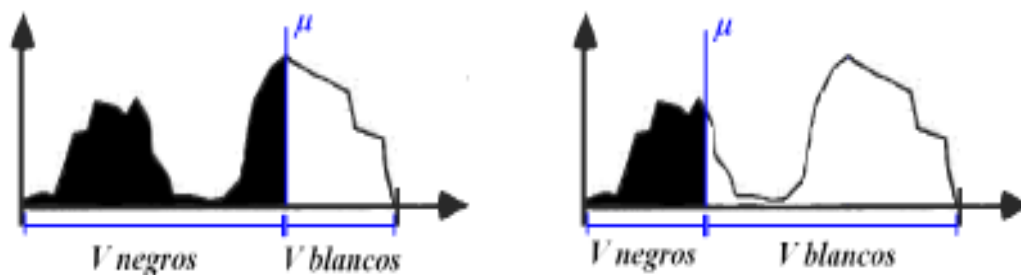


Fig.3. 13 Umbral óptimo

Fuente: Visión artificial – umbralización 2011

Para explicar el método de Otsu se usará un ejemplo sobre la imagen de 6X6 mostrada Fig.3.14:

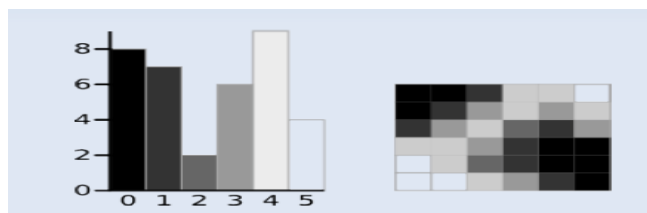


Fig.3. 14 Imagen 6 \* 6

Fuente: Visión artificial – umbralización 2011

Por ejemplo para el caso de usar un valor de 3 para el fondo y primer plano tenemos: para el fondo "0" (*background*) (Fig. 3.15)

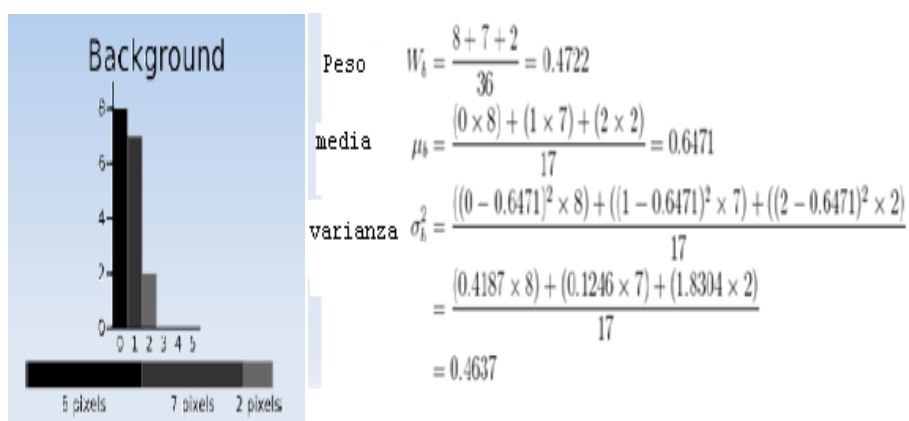


Fig.3. 15 Cálculo de parámetros para el fondo (*background*)

Fuente: Visión artificial – umbralización 2011]

Para el primer plano "1" (*foreground*)(Fig. 3.16)

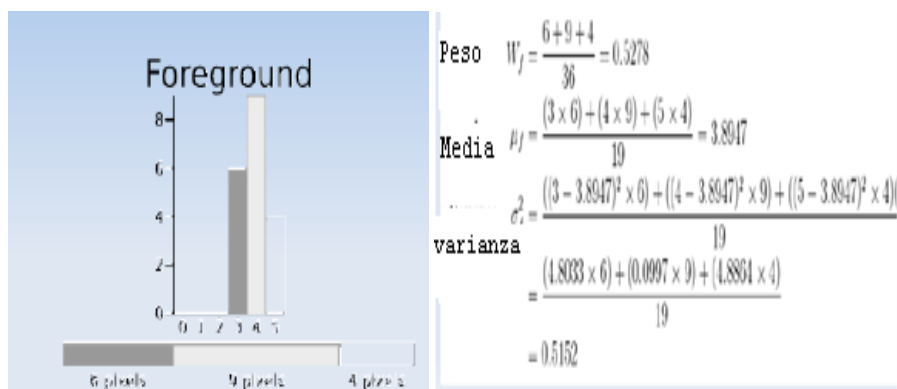


Fig.3. 16 Cálculo de parámetros para el primer plano (*foreground*)

Fuente: Visión Artificial – Umbralización2011

Finalizando con la obtención del umbral óptimo.

varianza entre clases  $\sigma_w^2 = W_b \sigma_b^2 + W_f \sigma_f^2 = 0.4722 * 0.4637 + 0.5278 * 0.5152$

$$= 0.4909$$

### 3.5 Etapa de detección de la placa.

La detección de la placa comprende cuatro módulos: (1) detección de contornos (2) selección de rectángulos (3) corrección geométrica y (4) selección de la placa los cuales se muestra en el diagrama de la Fig. 3.5.

La Fig. 3.17 nos muestra los módulos del proceso de detección de placa en sí:

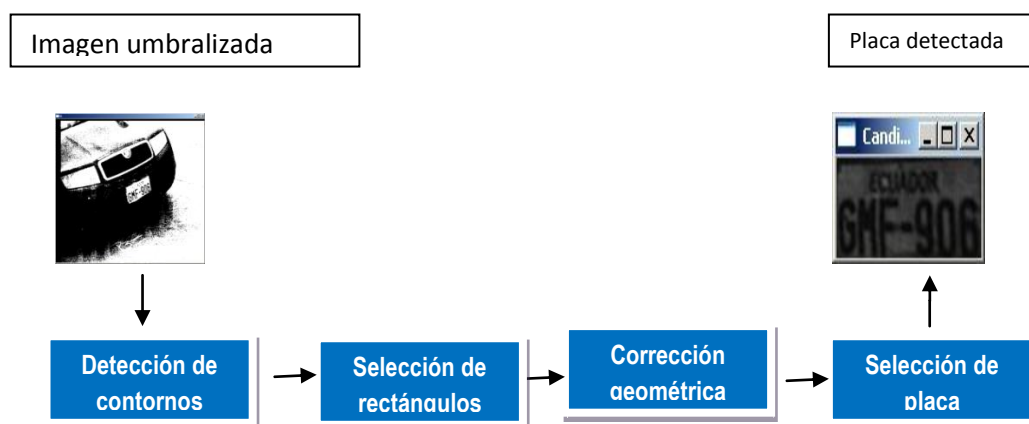


Fig.3. 17 Diagrama de módulos para la detección de placa.

### 3.5.1 Detección de contornos.

La detección de contornos tiene como objetivo recibir como entrada una imagen previamente binarizada y devolver como resultado un conjunto de contornos cerrados, los cuales están representados por polígonos. Para este propósito un operador de detección de contornos es aplicado a la imagen de entrada. Luego los contornos resultantes inicialmente analizados mediante una aproximación poligonal para encontrar únicamente los contornos cerrados.

Funciones definidas en OpenCv ayudan a los procesos de detección de contornos y aproximación poligonal. Para la detección de contornos las siguientes funciones fueron utilizadas:

```

int CvFindContours (CvArr *image, CvMemStorage*storage, CvSeq*
first_contour,      inthead_size sizeof (CvContour), int mode

```

`CV_RETR_LIST, int method CV_CHAIN_APPROX_SIMPLE , CvPoint offset cvPoint (0, 0))` (Esta función está descrita en el Anexo 1 [literal d](#)).

`void CvDrawContours(CvArr *img, CvSeq* contour , CvScalar external_color , CvScalar hole_color, int max_level int thickness=1 int lineType=8)` (Esta función está descrita en el Anexo 1 [literal f](#)).

La fig. 3.18 muestra el estado de la placa después de la detección de contornos.



Fig.3. 18 Detección de contornos en la placa.



Para la aproximación poligonal es necesario determinar un valor de tolerancia a la aproximación la librería de OpenCv sugiere un valor de 0.025 pero experimentalmente obtuvimos mejores resultados con 0.07 y es con el valor que trabajamos. Cada polígono obtenido es representado como un vector de puntos que contiene las coordenadas de dicho polígono. La función de OpenCv para este propósito fue:

**CvSeq \* cvApproxPoly (const void \*src\_seq, int header\_size, CvMemStorage \*storage , int method ,doble parameter ,int parameter2 = 0)**(Esta función esta descrita en el Anexo 1 [literal g](#)) .

Finalmente para la selección de los polígonos resultantes se aplico el siguiente análisis. Cada polígono obtenido fue revisado individualmente en busca de aquellos que cumplan con las siguientes restricciones: numero de lados, área, relación entre sus lados adyacentes y ángulo. El resultado de este análisis es un conjunto de polígonos candidatos que posteriormente serán procesados por el módulo de selección de rectángulos

### 3.5.2 Selección de rectángulos

En esta etapa el conjunto de polígonos candidatos obtenidos del módulo anterior son aproximados a rectángulos con el propósito de buscar la placa. Para esto se uso la ley del coseno para calcular el ángulo entre los lados adyacentes del rectángulo.

Teóricamente el ángulo entre cada lado del rectángulo es de  $90^\circ$  grados, pero en nuestro caso el polígono que aproxima al rectángulo de la placa posee imperfecciones en el proceso de la detección del rectángulo por lo que debíamos escoger un rango para este ángulo. De esta forma, experimentalmente se obtuvo que el rango del ángulo entre 2 lados del polígono aproximadamente esté entre  $75^\circ$  y  $105^\circ$ grados.

En la Fig. 3.19 se selecciona el rectángulo en la placa que es nuestra área de interés

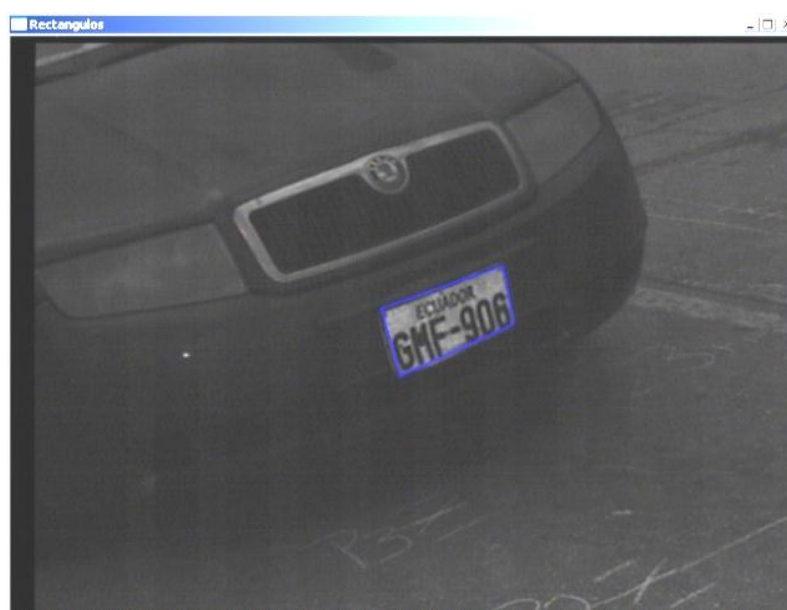


Fig.3. 19 Selección de rectángulo (área de interés).

### 3.5.3 Corrección geométrica

Una vez encontrado el conjunto de rectángulos aproximantes se procede a realizarla corrección geométrica delos polígonos usando una función de

perspectiva. Esto debido a que en ciertas ocasiones la imagen de la placa tiene cierta inclinación o perspectiva.

La función de OpenCv que nos ayudó para la corrección:

```
void cvWarpPerspective ( constCvArr* src, CvArr* dst, constCvMat*  
map_matrix,intflags=CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS,CvScalar  
fillval=cvScalarAll(0) ) (Esta función esta descrita en el Anexo 1literal h).
```

Cabe recalcar que la corrección geométrica la realizamos sobre la imagen original mas no sobre los anteriores procesos. Esto es debido a que los procesos para la detección del área de interés se puede perder información como por ejemplo el color ya que la imagen anterior a este proceso es una imagen umbralizada y para el siguiente proceso de segmentación se necesita la imagen original para realizar la misma.

En la Fig. 3.20 se muestra la corrección a la placa ya que la misma podría presentar desviaciones que perjudicarían la detección correcta de la placa.

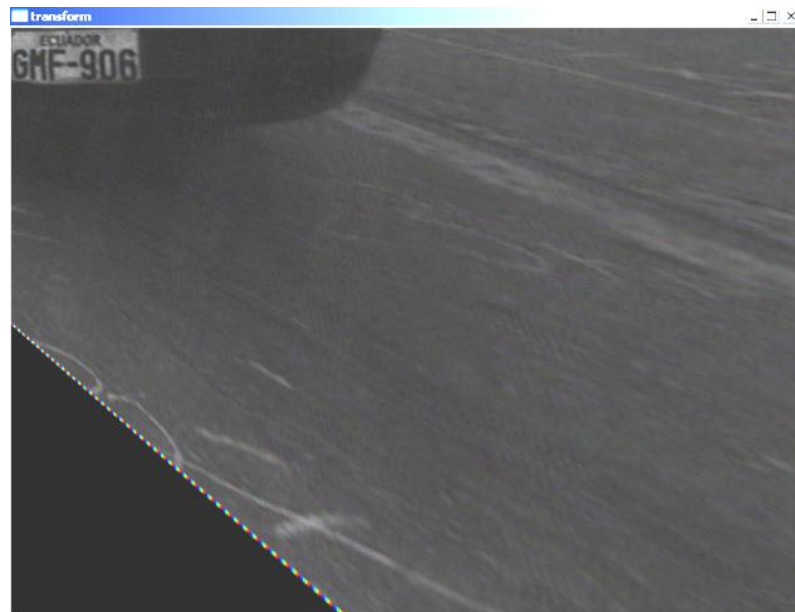


Fig.3. 20 Estado de la imagen después de la corrección.

#### 3.5.4 Selección de placa

En esta etapa se realiza la proyección vertical de la porción de imagen extraída en la sección anterior y se realiza un análisis en busca de una firma características esto es una secuencia de picos de la proyección vertical. Esto puede interpretarse como un proceso de filtrado

Si el proceso anterior es verdadero tenemos una placa detectada y almacenada para ser utilizada en el siguiente proceso que es la segmentación de la placa. Finalmente tenemos la placa detectada la cual se muestra en la Fig. 3.21 la que es enviada al proceso de segmentación.

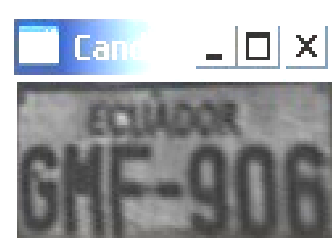


Fig.3. 21 Placa detectada

Para la realización de este módulo utilizamos la siguiente función:

**double cvGetReal1D(const CvArr\* arr ,int idx0)**(Esta función esta descrita en el Anexo 1 [literal i](#)).

#### **Descripción de la firma.**

Para discernir si el rectángulo identificado es efectivamente una placa o no nos basamos en una firma característica [6] de una placa para la cual se toma en cuenta los siguientes parámetros:

**Desviación Estándar:** En base del resultado de la proyección vertical se considera que los valores de la desviación estándar son grandes.

**Valor promedio:** El balance entre zonas oscuras y claras permite suponer que los valores promedios otorgados por la proyección serán similares para todas las placas.

**Número de picos:** En la proyección vertical denota el espacio entre los caracteres los cuales tiene una amplitud que depende del ancho mínimo y máximo de los espacios entre los caracteres.

**Número de valles:** Dentro de la proyección vertical está relacionado con los caracteres dentro de la placa con el ancho mínimo y máximo de los caracteres. Los valores de los picos que estén encima de la proyección vertical son eliminados.

## **CAPÍTULO 4**

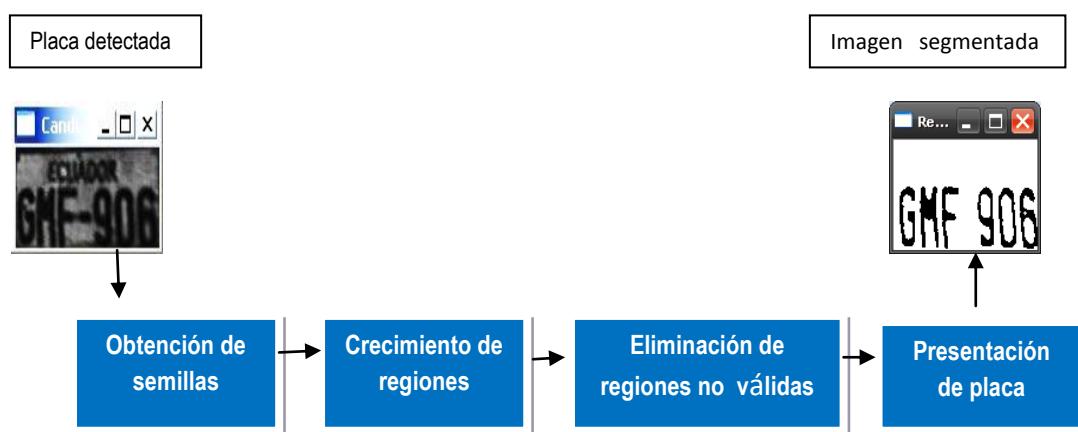
### **SEGMENTACIÓN DE LA PLACA**

En este capítulo describiremos cuáles han sido los pasos que hemos seguido para realizar la implementación de la segmentación de la placa detectada en el capítulo anterior. Partimos de un diagrama de módulos para esta etapa con su consiguiente diagrama de flujo para finalmente describir en detalle cada uno de los cuatro módulos que son: 1) obtención de semillas, es en este módulo donde se encuentran los puntos iniciales del crecimiento de regiones como lo describe en 4.3.1; 2) crecimiento de regiones, es en este módulo donde una vez obtenida la semilla se procede con el crecimiento de la región de acuerdo a una regla definida detallada más adelante en 4.3.2; 3) eliminación de regiones no validas, es en este módulo donde eliminamos regiones que de acuerdo a sus características son designada como no validas como se lo detalla en 4.3.3 ; y 4) presentación de la placa, es en este módulo donde hacemos la presentación nítida de la placa donde solo están caracteres y dígitos de la placa.

#### **4.1 Diagrama de módulos para la segmentación de placa.**

Para la implementación de esta etapa hemos considerado los siguientes módulos: obtención de los puntos semillas, crecimiento de regiones, eliminación de regiones no válidas y presentación de regiones los cuales

serán descritos posteriormente la Fig. 4.1 muestra el diagrama de módulos usado:



**Fig.4. 1 Diagrama de módulos para la segmentación de placa.**



## 4.2 Diagrama de flujo para segmentación de la placa.

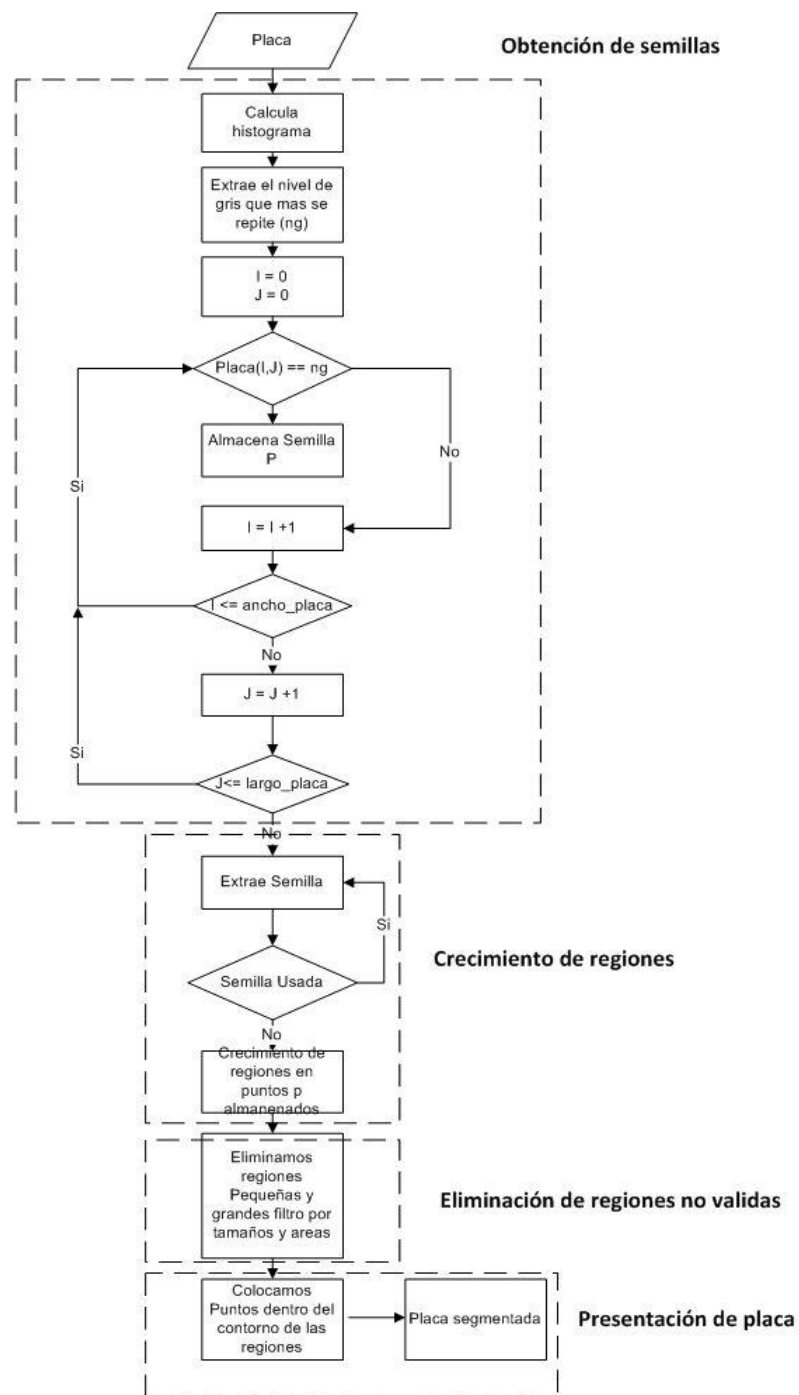


Fig.4. 2 Diagrama de flujo para la segmentación

### 4.3 Descripción de los módulos para la segmentación de placa

Se describe a continuación los módulos que se desarrollaron para segmentar la placa una vez detectada:

#### 4.3.1 Obtención de semillas.

Dentro de este módulo se obtienen los puntos semillas para nuestras regiones, el cual se lo desarrolló en base a histogramas de la siguiente manera:

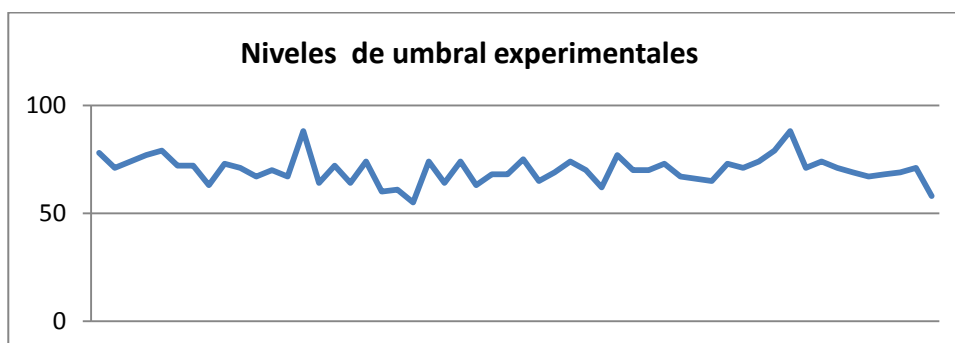
Dentro de lo que es el módulo de crecimiento de regiones se tuvo un gran dilema el cual era cómo encontrar las semillas para realizar el crecimiento, entonces decidimos obtener el histograma de la imagen y obtener el nivel de gris que más veces se repetía (histograma de frecuencias), y todos los puntos que tuvieran dicho nivel de gris serían los puntos semillas a seleccionar experimentalmente más del 90% de las imágenes usadas obtuvieron buenas semillas. Siguiendo los siguientes pasos:

1.- Obtención del histograma

2.- Escoger el nivel de gris en el que el histograma tiene el mayor valor teniendo en cuenta nuestro umbral de hasta qué punto puede ser muy buena semilla ya que nos interesa segmentar solo puntos cercanos al 0 y no tan mayores a 70 que según nuestros datos estadísticos tomados

experimentalmente, definían un nivel de sombra luego de este valor.

La Fig.4.3 nos muestra que el promedio del nivel sombra es 70:



**Fig.4. 3 Estadísticas para el umbral.**

3.- Dentro de la imagen de la placa buscar todos los puntos que tengan ese nivel de gris y almacenarlos en un vector de puntos semillas.

La Fig. 4.4 muestra los puntos semillas dentro de la placa, marcados de color blanco.



**Fig.4. 4 Imagen con puntos semillas de color blanco**

La Fig. 4.5 indica la imagen utilizada para el crecimiento de regiones.



Fig.4. 5Imagen previa al crecimiento de regiones

#### 4.3.2 Crecimiento de regiones

Una vez obtenidas las semillas la decisión a tomar fue hasta dónde puede crecer y cuál será la regla con la que termine el crecimiento y experimentalmente se decidió crecer sólo hasta el umbral definido por el nivel de sombra ya que si crecía más juntaba muchos píxeles lo cual haría difícil luego su separación.

Esto viene definido por:

$$0 < p(y) < \text{Umbral sombra}$$

Donde  $p(x, y)$  es el valor del nivel de gris en el punto  $(x, y)$ , el valor del umbral sombra ya fue definido y calculado en 4.3.1. El valor de 0 fue definido directamente por que nuestro caso solo necesitamos una imagen binarizada.

En nuestra función de OpenCv definimos el tipo de vecindad que queremos usar en el crecimiento este puede ser en vecindad de 4 ó vecindad de 8.

La función de OpenCv nos devuelve las regiones de una manera no tan inteligente así que deberemos implementar una función para buscar el contorno de cada una de las regiones que conforman la placa vehicular.

Al resultado de las operaciones anteriores lo guardamos en una variable de componentes conectados que ya viene definido en OpenCv la cual está compuesta por:

- El área de la región.
- Promedio del nivel de gris de la región.
- Las coordenadas de la región.

Con estos datos los almacenamos y a su vez revisamos cada uno de los puntos de la placa de tal forma que un mismo punto no se pueda tomar en cuenta en otra región y evitar costos computacionales en el procesamiento.

La Fig. 4.6 señala el inicio del crecimiento de la imagen con la que comenzamos este módulo:



**Fig.4. 6 Inicio de crecimiento de regiones.**

En este modulo usamos la siguiente función:

```
void cvFloodFill ( CvArr* image CvPointseed_point, CvScalarnew_val,
CvScalarlo_diff=cvScalarAll(0) ,CvScalarup_diff=cvScalarAll(0)
CvConnectedComp* comp=NULL, int flags=4 ,CvArr* mask=NULL) (Esta
función esta descrita en el Anexo 1literal j).
```

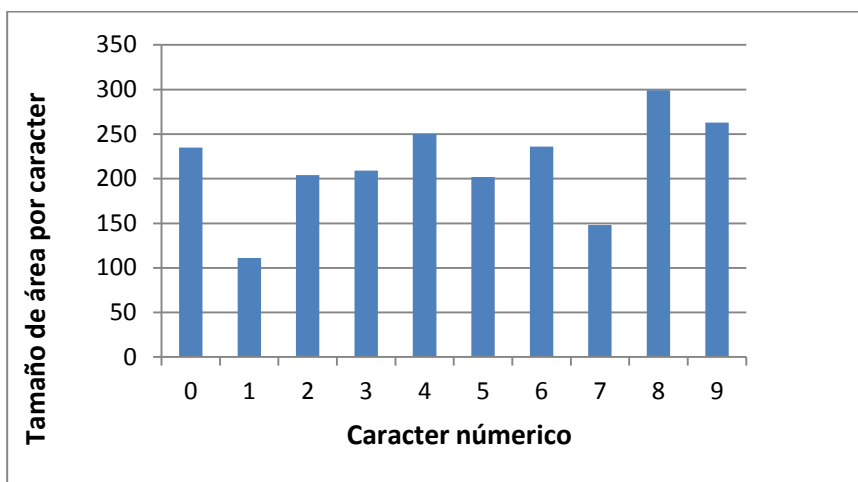
#### 4.3.3 Eliminación de regiones no válidas.

Dentro de este módulo lo que haremos será eliminar regiones que han sido seleccionadas del paso previo y que no correspondan a caracteres de una placa. Para esto tomamos datos estadísticos experimentalmente de tamaños de área máximos y mínimos la altura de caracteres con respecto a la altura de la placa que casi es un valor constante pero tiene cierto nivel de incertidumbre.

Para este modulo utilizaremos la siguiente función de OpenCv:

```
double CvPointPolygonTest (constCvArr*contour,CvPoint2D32fpt,int
measure_dist ) (Esta función esta descrita en el Anexo 1literal k).
```

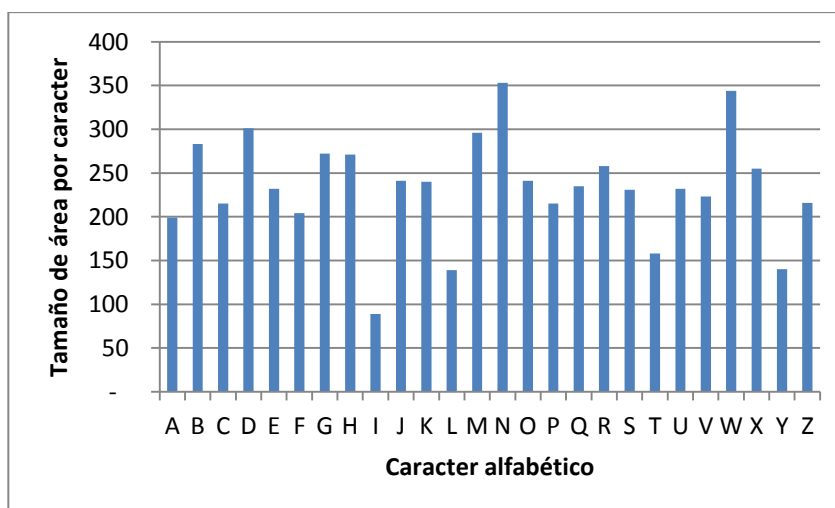
Experimentalmente para el tamaño del área mínimo y máximo se escogió el tamaño estándar para la imagen que es 125x70píxeles y se calculó obteniendo el siguiente resultado mostrado en la Fig.4.7:



**Fig.4. 7 Valores experimentales de área (números).**

Los resultados obtenidos muestran que el área mínima en número lo tiene el número 1, con un valor de 111, y el máximo el número 8 con un valor de 299

En cuanto a letras tenemos el siguiente resultado mostrado en la Fig.4.8:

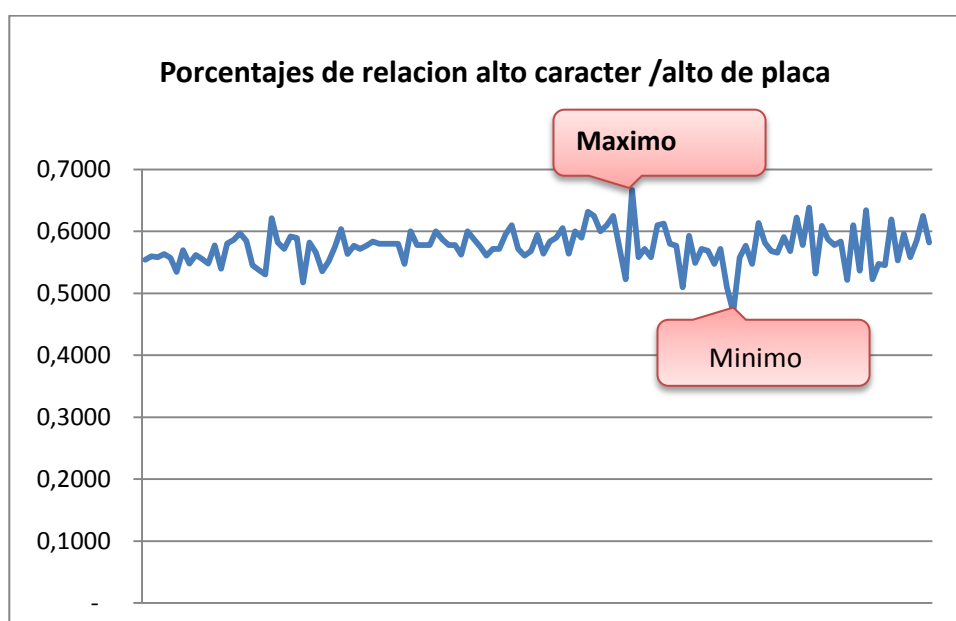


**Fig.4. 8 Valores experimentales de área (letras)**

Dándonos como resultado la letra con menor área la I con un valor de 89 y la máxima la N con un valor de 353

Formando el mínimo valor entre caracteres y números el área mínima será de 89 para nuestro proyecto.

En cuanto al filtro por tamaños sabemos que existe una relación entre el alto de la placa y el alto del carácter el cual también lo elegimos estadísticamente como lo muestra la Fig.4.9:



**Fig.4. 9** Estadísticas de relación alto carácter respecto al alto de placa (valores experimentales).

Analizando la gráfica la mínima relación entre el alto de la placa y el carácter debe ser del 47%, y la máxima relación debe ser 67%.

La Fig. 4.10 muestra la imagen con todas las regiones de la placa.





Fig.4. 10 Eliminación de regiones no válidas

#### 4.3.4 Presentación de placa.

Finalmente, en este módulo se presenta la placa una vez eliminadas las regiones que no son de interés ya sea por el área, tamaño de caracteres, tamaño de regiones (como han sido descritos en el anterior módulo) se obtiene como resultado final la placa segmentada como nos muestra la Fig.4.11.



Fig.4. 11 Imagen segmentada.

## **CAPÍTULO 5**

### **ANÁLISIS DE RESULTADOS**

En este capítulo mostramos los resultados que se obtuvieron de las diferentes pruebas realizadas en este proyecto durante la fase de implementación del mismo hasta llegar a una segmentación con un margen aceptable de efectividad. Posteriormente se describen las conclusiones y recomendaciones que consideramos pertinentes mencionar como parte final de este informe.

#### **5.1 Análisis de resultados**

Mediante la ejecución de nuestro proyecto se obtuvieron los resultados para los 24 puntos en los que se tomaron las imágenes originales del sistema ANPR.

La tabla N<sup>o</sup> 3 nos muestra los resultados obtenidos de nuestra implementación.

Puntos	# de Imágenes tomadas	# de placas detectadas	# de placas segmentadas	% Segmentación
Punto 1	1100	700	700	100%
Punto 2	1000	900	900	100%
Punto 3	1000	1000	1000	100%
Punto 4	1000	1000	1000	100%
Punto 5	1000	1000	1000	100%
Punto 6	1000	1000	1000	100%
Punto 7	1000	900	800	89%
Punto 12	1000	1000	1000	100%
Punto 15	1000	1000	-	0%
Punto 16	1000	1000	900	90%
Punto 18	1000	600	500	83%
Punto 19	1000	1000	1000	100%
Punto 20	1000	900	300	33%
Punto 21	1000	600	600	100%
Punto 22	1000	1000	900	90%
Punto 23	1000	300	300	100%
Punto 24	1000	1000	1000	100%

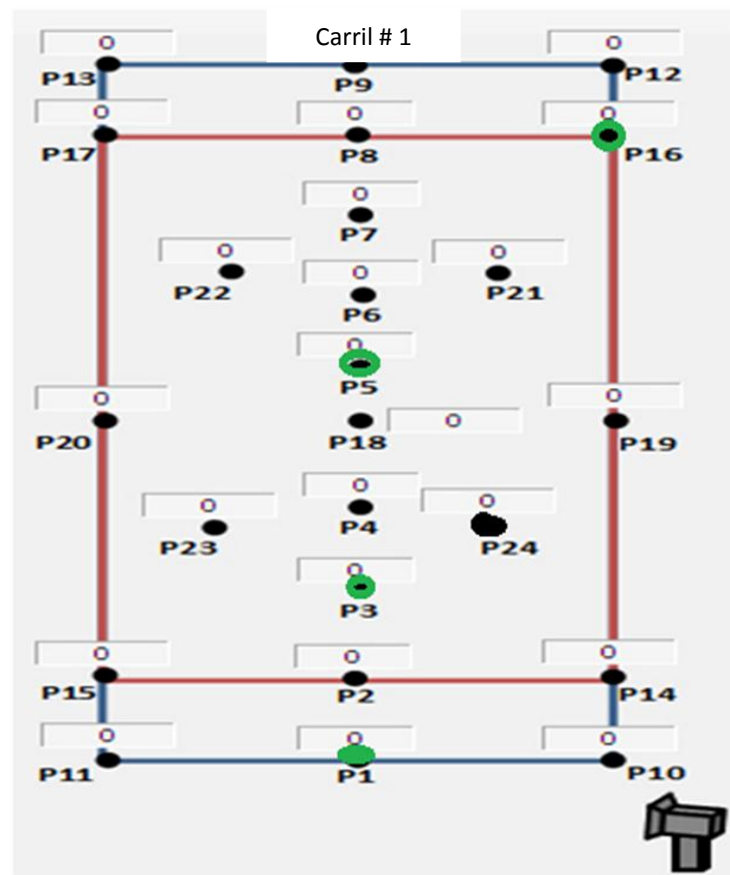
**Tabla 3 Porcentaje de imágenes segmentadas según cada punto.**

Dentro de los 24 puntos analizados nuestro algoritmo de detección de placas pudo localizar en 16 de ellos placas válidas y las cuales fueron segmentadas de forma aceptable.

La Fig.5.1 muestra los 24 puntos que indican las diferentes distancias de enfoque usadas para nuestro proyecto. Para cada punto o distancia se capturan 10 imágenes de prueba.

Nuestro algoritmo de segmentación fue ejecutado sobre este conjunto de imágenes de prueba. En la Fig.5.1 se marcan de color verde los puntos 1, 3, 5 y 16, los cuales son usados para mostrar los resultados obtenidos aquí en esta tesis.

Carril #2



Cámara

Fig.5. 1 Imagen de puntos de muestra 1, 3, 5,16.

A continuación mostraremos varias imágenes segmentadas resultantes sobre los puntos 1, 3, 5, y 16 después de ejecutar nuestro algoritmo propuesto.

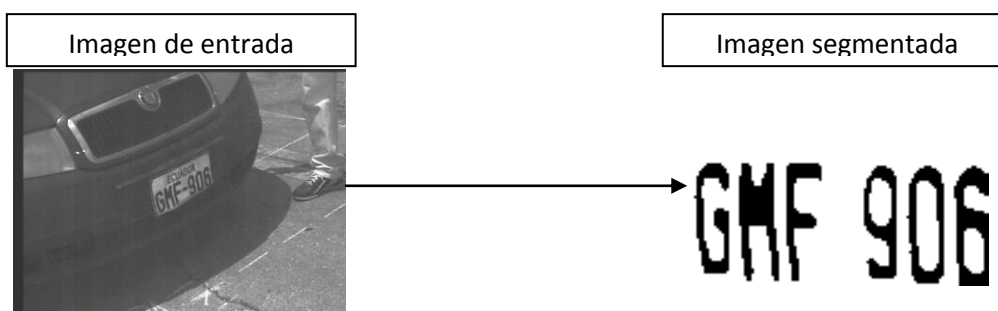
**Punto1**

Fig.5. 2Imagen segmentada resultante para el punto 1

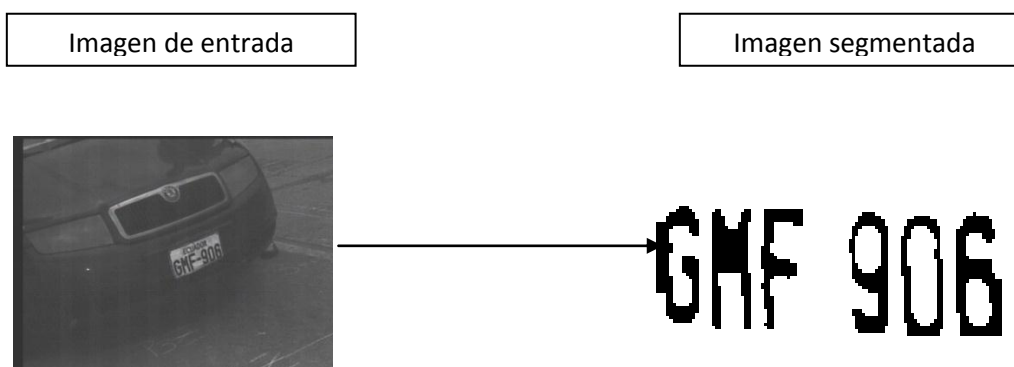
**Punto 3**

Fig.5. 3Imagen segmentada resultante par el punto 3

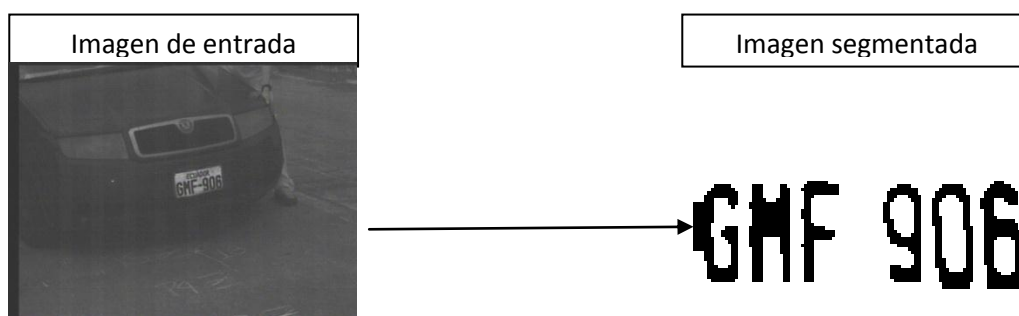
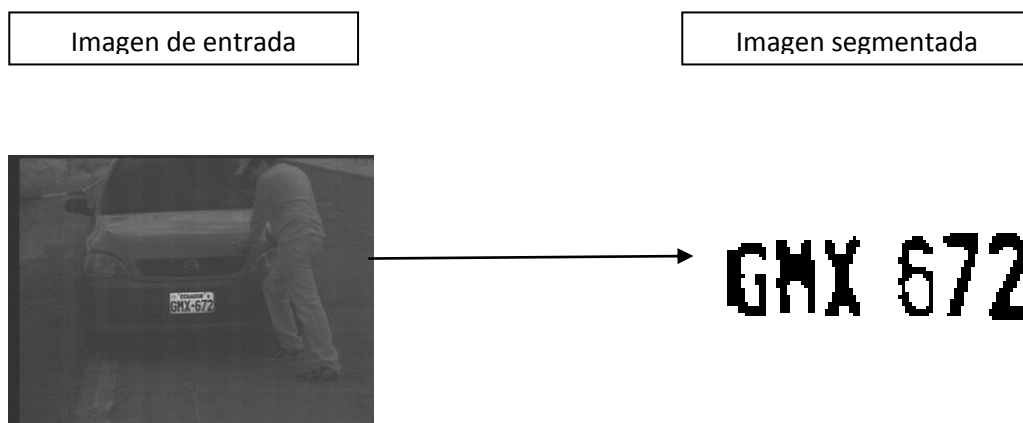
**Punto 5**

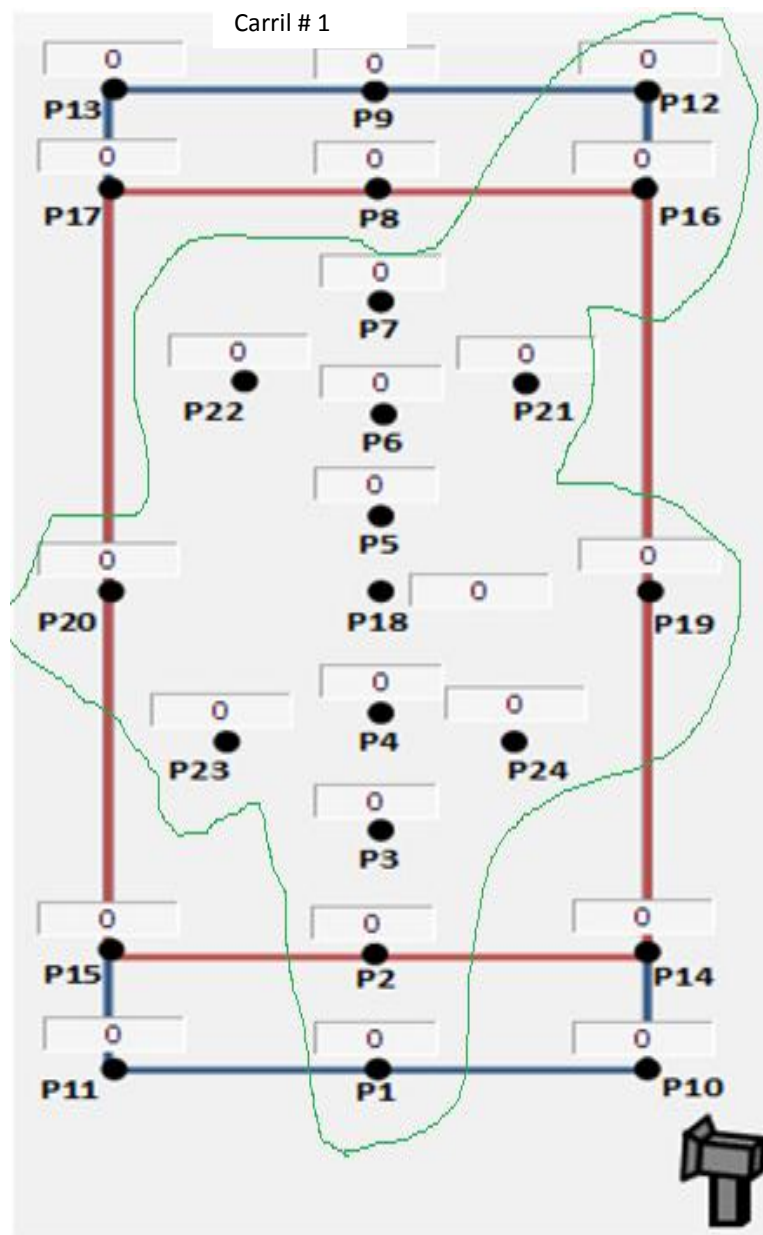
Fig.5. 4Imagen segmentada resultante pare el punto 5

**Punto 16**

**Fig.5. 5Imagen segmentada resultante para el punto 16**

A continuación en la Fig.5.6 se muestra todos los puntos en los que nuestro algoritmo funciona correctamente. Estos puntos han sido encerrados por una línea de color verde.

Carril #2



Cámara

Fig.5. 6 Imagen de puntos que el algoritmo segmenta correctamente



Como muestran la Fig.5.6 nuestro algoritmo presenta mejor resultado en los puntos {1, 3,5, 16}, lo cual representa el 66.7 % de los 24 puntos usados para nuestras pruebas.

Adicionalmente podemos notar que la distancia de enfoque para que nuestro algoritmo sea eficiente se concentra en la mayoría de los puntos centrales de nuestra plantilla usada.

## **Conclusiones y recomendaciones**

### **Conclusiones**

- 1.- En nuestro análisis los mejores puntos para que nuestro algoritmo funcione mejor sería dentro de los primeros 6 puntos de toma de imagen, según se muestra en la Fig.5.6.
- 2.- A nuestra consideración el tiempo de respuesta de la técnica propuesta es rápido para imágenes pequeñas y que solo tenga como objetivo imágenes en las que solo se desee segmentar el color negro.
- 3.- A pesar de que el siguiente proceso que tendrá nuestra imagen resultante será el de OCR y lo recomendable es tener como dimensiones mínimas de imagen 120x50píxeles nuestra propuesta funciona hasta con imágenes de 85x45píxeles.
- 4.-La iluminación de la imagen influye en el procesamiento de la imagen ya que dependiendo de esto aumenta o disminuye el nivel de sombra.
- 5.-Pueden existir caracteres unidos entre si que dificultan el proceso de segmentación de la placa.

## **Recomendaciones**

1.-Sería interesante probar otras técnicas que ayuden a corregir lo que es el contorno del objeto se trató de corregir con operaciones morfológicas pero el resultado no fue el esperado.

2.-Si se trabaja con una implementación de crecimiento de regiones propia tener en cuenta que se debe tomar como argumento el área y los contornos de la región.

3.- Se podría mejorar las imágenes de entrada para este proyecto.

## Anexo 1

Algunas de las funciones de OpenCv se describen a continuación:

**a) void CvCvtColor(constCvArr\*src ,CvArr\*dst,int code)**

Convierte una imagen desde el espacio de color a otro.

**Parámetros:**

src - La fuente de 8 bits (8) 16-bit (16U) o de un solo punto flotante de precisión (32f) imagen

dst - La imagen de destino del mismo tipo de datos como fuente. El número de canales puede ser diferente

Código - Color operación de conversión que se pueden utilizar specified CV\_  
src\_color\_space \* \* 2 \* \* dst\_color\_space constantes (ver más abajo)

La función convierte la imagen de entrada desde el espacio de color a otro. La función pasa por alto el modelo de color y los campos de la cabecera channelSeqIplImage por lo que la imagen de origen del espacio de color debe ser especificado correctamente (incluyendo el orden de los canales en el caso del espacio RGB. Por ejemplo BGR medio formato de 24 bits con \$ B\_0 G\_0 R\_0 B\_1 g\_1 R\_1 ... \$ layout mientras que los medios RGB de 24 con el formato \$ R\_0 G\_0 B\_0 R\_1 g\_1 B\_1 ... \$ layout).

La gama convencional de R G B valores de los canales es la siguiente:

0 a 255 para imágenes de 8 bits

0 a 65535 para imágenes de 16 bits y

0 a 1 para punto flotante de las imágenes.

Por supuesto en el caso de las transformaciones lineales de la gama puede ser específico pero con el fin de obtener resultados correctos en el caso de transformaciones no lineales la imagen de entrada debe ser a escala.

**b) void CvEqualizeHist( constCvArr\* src, CvArr\* dst )**

Ecuáliza el histograma de la imagen de entrada mediante el siguiente algoritmo:

1. Calcular H histograma de src.
2. Normalizar el histograma de modo que la suma de los cubos de histograma es de 255.
3. Calcular integral del histograma:

$$(I) H' = \sum_{0 \leq j \leq i} H(j)$$

4. Transformar la imagen utilizando H' como una tabla de búsqueda:

$$DST(x, y) = H'(src(x, y))$$

5. El algoritmo normaliza el brillo y aumenta el contraste de la imagen.

**c) void CvThreshold( constCvArr\* src , CvArr\* dst , double threshold ,double max\_value, intthreshold\_type )**

Aplica un umbral fijo a nivel de elementos de la matriz. La función se aplica a nivel de umbral fijo a una matriz de un solo canal. La función se suele utilizar para obtener una de dos niveles (binaria) de la imagen en escala de grises (CMPS puede ser utilizado para este propósito) o para la eliminación de un ruido es decir el filtrado de píxeles con un valor demasiado pequeño o demasiado grande. Hay varios tipos de umbral que ayuda a la función que están determinadas por thresholdType.

Además el especial valor CV\_THRESH\_OTSU se puede combinar con uno de los valores anteriores. En este caso la función determina el valor del umbral óptimo utilizando el algoritmo de Otsu y lo usa en lugar del umbral especificado. La función devuelve el valor del umbral calculado. En la actualidad el método de Otsu se implementa sólo para imágenes de 8 bits.

**d)int CvFindContours(CvArr \*image, CvMemStorage\*storage, CvSeq\* first\_contour, intheader\_size sizeof (CvContour), int mode CV\_RETR\_LIST, int method CV\_CHAIN\_APPROX\_SIMPLE , CvPoint offset cvPoint (0, 0))**

Encuentra los contornos de una imagen binaria.

La función recupera los contornos de la imagen binaria mediante el algoritmo de Suzuki85. Los contornos son una herramienta útil para el análisis de la forma y la detección de objetos y reconocimiento.

La función recupera los contornos de la imagen binaria y devuelve el número de contornos recuperados. El puntero `first_contour` es llenado por la función que contendrá un puntero al primer contorno exterior o NULL si no se detectan los contornos (si la imagen es completamente negra). Otros contornos se pueden obtener desde `first_contour` con el `h_next` y enlaces `v_next`. Puede ser utilizado para el análisis de la forma y el reconocimiento de objetos.

**e) `void CvDrawContours(CvArr *img, CvSeq* contour , CvScalarexternal_color , CvScalarhole_color, intmax_levelint thickness=1 intlineType=8)`**

Dibuja contornos externos e interiores en una imagen.

Si `thickness >= 0` dibuja contornos externos en la imagen.

Si `thickness < 0` llena el área rodeada por el contorno.

**f) `CvSeq * cvApproxPoly (const void *src_seq, intheader_size, CvMemStorage *storage , int method ,doble parameter int parameter2 = 0)`**

Aproxima a la curva poligonal con la precisión especificada por el método Aproximación CV\_POLY\_APPROX\_DP sólo es compatible que se corresponde con el algoritmo de Douglas-Peucker.

parameter.- parámetro específico en el caso de CV\_POLY\_APPROX\_DP se trata de una precisión de aproximación deseada.

parameter2.- Si src\_seq es una secuencia el parámetro determina si la secuencia de un solo debería aproximarse o todas las secuencias en el mismo nivel o por debajo de src\_seq (ver FindContours para la descripción de las estructuras jerárquicas de contorno).

Si es un src\_seq \* CvMat matriz de puntos el parámetro especifica si la curva es cerrada (parámetro2! = 0) o no (parámetro 2 = 0).

La función se aproxima a una o más curvas y devuelve el resultado de aproximación. En el caso de múltiples curvas el árbol resultante tendrá la misma estructura que la entrada de uno (1:1 correspondencia).

```
g)void cvWarpPerspective( constCvArr* src, CvArr* dst, constCvMat*  
map_matrix,intflags=CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS,CvScalarfillval=  
cvScalarAll(0) )
```



CV\_WARP\_FILL\_OUTLIERS - rellenar todos los píxeles de la imagen de destino. Si algunos de ellos corresponden a los valores extremos de la imagen original que se establecen en fillval.

CV\_WARP\_INVERSE\_MAP - indica que la matriz es transformada inversa de la imagen de destino a la fuente y por tanto se puede utilizar directamente para la interpolación de píxeles. De lo contrario la función encuentra la transformada inversa de map\_matrix.

La función cvWarpPerspective transforma la imagen de origen utilizando la matriz específica.

**h) double cvGetReal1D(constCvArr\* arrint idx0)**

Devuelve un elemento concreto de una matriz de un solo canal. Si la matriz tiene canales múltiples un error de ejecución es elevado. Tenga en cuenta que función Get se puede utilizar de forma segura tanto para un solo canal y matrices de múltiples canales aunque es un poco más lento.

**i) void cvFloodFill ( CvArr\* image CvPointseed\_point, CvScalarnew\_val, CvScalarlo\_diff=cvScalarAll(0) ,CvScalarp\_diff=cvScalarAll(0) CvConnectedComp\* comp=NULL, int flags=4 ,CvArr\* mask=NULL)**

Llena un componente relacionado con el color dado.

Parámetros:

image - Entrada 1 - o 3 canales la imagen de 8 bits o de punto flotante. Es modificado por la función a menos que la bandera CV\_FLOODFILL\_MASK\_ONLY.

seed\_point - El punto de partida.

new\_val - Nuevo valor de los píxeles de dominio repintado.

lo\_diff - Diferencia de color entre los píxeles se observa actualmente y uno de sus vecinos pertenecientes a la componente o un píxel semilla que se añade al componente.

up\_diff - Diferencia de color entre los píxeles se observa actualmente y uno de sus vecinos pertenecientes a la componente o un píxel semilla que se añade al componente.

Comp - Puntero a la estructura que la función se llena con la información sobre el dominio repintado.

Flags.- Las banderas de la operación. Inferior bits contienen el valor de conectividad 4 (por defecto) u 8 que se utiliza en la función. La conectividad determina que los vecinos de un píxel se consideran. Bits superiores puede ser 0 o una combinación de las siguientes opciones:

- `CV_FLOODFILL_FIXED_RANGE` si está establecido la diferencia entre el píxel actual y píxel semilla es considerada.

- `CV_FLOODFILL_MASK_ONLY` si está establecido la función no se llena la imagen (`new_val` se tiene en cuenta).

`mask` - máscara de la operación debe ser un solo canal de 8 bits de la imagen.

La función llena un componente conectado desde el punto de semillas con el color especificado. La conectividad está determinada por la cercanía de los valores de píxel.

Un ejemplo del uso de esta función es mostrado a continuación:

### **Uso de `CvFillFlood`**

El siguiente código muestra la aplicación sencilla de la función `CvFillFlood`:

```
#include "stdafx.h"
```

```
#include "cv.h"
```

```
#include "cxcore.h"
```

```
#include "highgui.h"
```

```
int _tmain(int argc, _TCHAR* argv[])
```

```
{
```

```
IpImage* newImg = NULL;

IpImage* fImg = NULL;

//flood and fill parameters

int lo_diff, up_diff; //the low and up flood range which can be adjusted

CvConnectedComp comp;

CvPoint floodSeed; //the original pixel where the flood begins

CvScalar floodColor;

lo_diff=8;

up_diff=8;

floodColor = CV_RGB( 255 0 0 ); //set the flood color to red

cvNamedWindow("src" 1);

cvNamedWindow("flood&fill" 1);

//load original image

newImg = cvLoadImage("apple.jpeg" 1);

cvShowImage( "src" newImg );

//make a copy of the original image
```

```
ffImg=cvCloneImage( newImg );

floodSeed=cvPoint(60,60); //flooding start from pixel(60 60)

//Flood and Fill from pixel(60 60) with color red and the flood range of (-8 +8)

cvFloodFill( ffImg,floodSeed,floodColor CV_RGB( lo_diff,lo_diff,lo_diff )

CV_RGB( up_diff,up_diff,up_diff ) &comp 8 NULL);

cvShowImage( "flood&fill" ffImg );

cvWaitKey(0);

cvDestroyWindow( "src" ); cvDestroyWindow( "flood&fill" );

cvReleaseImage( &newImg ); cvReleaseImage( &ffImg );

return 0;}
```

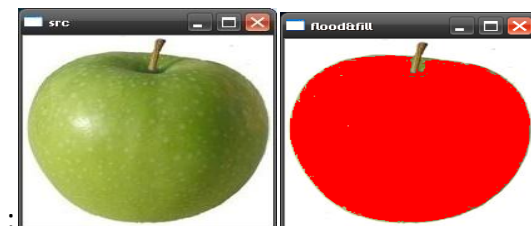


Fig. a) Imagen Original      b) después de CvFloodFill

**j) double CvPointPolygonTest(const CvArr\* contour, CvPoint2D32f pt,  
int measure\_dist)**

contour - contorno de entrada

pt - El punto de prueba en contra del contorno

measure\_dist - Si no es cero la función calcula la distancia desde el punto hasta el borde más cercano del contorno.

La función determina si el punto está dentro de un contorno fuera o se encuentra en un borde (o si coincide con un vértice). Devuelve un valor positivo negativo o cero según corresponda. Cuando measure\_dist = 0 el valor devuelto es +1 -1 y 0 respectivamente. Cuando measure\_dist ≠ 0 una distancia es asignado entre el punto y el borde más cercano del contorno

#### **k) Distribución de archivos del proyecto**

Para una mejor visualización de los resultados generados durante la ejecución del programa se ha distribuido de la siguiente manera:

La carpeta rectángulos contiene las imágenes área que denota la placa dentro de la imagen.

La carpeta placas contiene las imágenes con la placa ya detectada y que será utilizada para la etapa de segmentación.

Las imágenes que generan cada una de los módulos de esta etapa se almacenan en estas carpetas para agilizar la visualización de los resultados.

## Bibliografía

[1]Jun Wei Hsieh Shih Hao Yu and Yung-Sheng Chen,"Morphology-based-License Plate Detection from Complex Scenes Proceedings of the Complex Scenes",Proceedings of the 16 th International Conference on Pattern Recognition (ICPR02)", 2002.

[2] Platero Dueñas Carlos, "Apuntes de Visión Artificial", Dpto. Electrónica Automática Industrial, 2005, Pág. 149.

[3]Javier Cuadri Vázquez", Segmentación de Imágenes en color por intersección de histogramas de color y textura",Escuela Técnica de Ingenieros Universidad de Sevilla 2009,Cap.4 Pág. 30-46.

[4] Becilla Jonathan Matías Joseph," Segmentación de imágenes medicas para detección de detalles", Materia de Graduación Facultad en Electricidad y Computación, 2009, Pág. 14 -16.

[5] Calasanz Sapunar Lucy, "Diseño e implementación de un sistema de monitorización del nivel de alerta humano en tiempo real basado en el comportamiento ocular aplicado a seguridad vial ", Materia de Graduación Facultad en Electricidad y Computación, Oct. 14 2007, Cap. 3 Pág. 12-14.



[6] Azansa Maldonado, "Diseño de módulo de Pre procesamiento y Extracción de características físicas de una imagen", Escuela Politécnica Nacional, 2006, Cap. 3 Pág. 54-55.

[7] Calderón Carlos Francisco, "Visión Artificial–Umbralizacion", Universidad Pontificia Javeriana, 2011, Pág. 1-11.

[8] Quing Chen David, "A Basic Introduction to OpenCv for Image Processing", University of Ottawa, 2007, Pág.12.

[9] Oliver Rojas Juan Carlos, "Procesamiento Digital de imágenes", Universidad Vasco de Quiroga Mayo, 2009, Pág. 55-57-58.

[10] "Resolución y escala de imágenes digitales", <http://www.dimages.es/Tutorial/introduccion/resolucion.htm>, última consulta: 26/octubre/2011.

[11] "Entender el histograma", <http://www.webdefotografia.es/entender-el-histograma>, última consulta: 26/octubre/2011.

[12] "Modos de color", <http://www.desarrolloweb.com/articulos/1778.php>, última consulta: 26/octubre/2011.

[13] "Método del valor del umbral", [http://es.wikipedia.org/wiki/Método\\_del\\_valor\\_umbral](http://es.wikipedia.org/wiki/M%C3%A9todo_del_valor_umbral), última consulta: 26/octubre/2011.

[14] "Auto Treshold", [http://pacific.mpi-cbg.de/wiki/index.php/Auto\\_Threshold](http://pacific.mpi-cbg.de/wiki/index.php/Auto_Threshold)",  
última consulta: 26/octubre/2011.

[15]"OpenCv 2.0 CReference", <http://opencv.willowgarage.com/documentation/index.html>,  
última consulta: 26/octubre/2011.

[16]"CVReferenceManual",[http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref\\_cv.htm](http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref_cv.htm), última consulta: 26/octubre/2011.

[17]"OpenCV Computer Vision Library",[http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref\\_cv.htm](http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref_cv.htm), última consulta: 26/octubre/2011.