ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

INFORME DE

MATERIA DE GRADUACIÓN

“Normalización de imágenes de placas vehiculares a través de corrección geométrica.”

Previa a la obtención de los títulos de:

Ingeniero en CIENCIAS COMPUTACIONALES ESPECIALIZACIÓN SISTEMAS MULTIMEDIA

PRESENTADO POR:

Carlos Leonardo Choez Álvarez
Steve Fernando Salas Guerrero

GUAYAQUIL – ECUADOR
2011

AGRADECIMIENTOS

A Dios
por inspirarme cada
día a cumplir mis sueños.
A mis profesores
por su ejemplo de profesionalismo
que nunca olvidaré.
A mis Padres
por enseñarme el amor
por los estudios.

Carlos Choez A.

Agradezco Dios,
a mis padres por brindarme
siempre su apoyo y
confianza incondicional.
A mis amigos y
a todas aquellas personas que de
una u otra manera me han
ayudado a forjarme
profesionalmente.

Steve Salas

DEDICATORIAS

A quienes fueron mi inspiración,
mis héroes, mis mejores amigos,
mis ejemplos de lucha
y perseverancia, mis padres
Sr. Juan Marcillo y Sra. Mirian Álvarez.

Carlos Choez A.

A mis padres Fernando Salas
y Gloria Guerrero, por su guía
permanente en mi diario vivir.
A mis hermanas María Fernanda,
Lissette y Mishell que siempre me
apoyan en todo lo que pueden
y que siempre me están brindando
su total confianza y afecto.

Steve Salas

TRIBUNAL DE SUSTENTACIÓN

_____

Phd. Boris Vintimilla
PROFESOR DE LA MATERIA DE GRADUACIÓN

_____

Phd. Daniel Ochoa
PROFESOR DELEGADO

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Trabajo de Grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

(Reglamento de Graduación de la ESPOL).

_____	_____
Carlos Choez A.	Steve Salas G.
Mat. 200211068	Mat. 200211696

RESUMEN

El proyecto en general consiste en implementar un sistema de control de acceso vehicular mediante el reconocimiento del número de placa de manera automática usando una cámara y algoritmos de procesamiento de imágenes incluyendo Reconocimiento Óptico de Caracteres (OCR). Para el reconocimiento de las placas se tomara una imagen al vehículo al momento de entrar a un parqueadero específico.

El proyecto general consta de 5 partes fundamentales que son:

1.- Detección y extracción de placas.- El objetivo inicial de esta parte del proyecto es la instalación y configuración de un sistema de cámara para realizar detección del número de placa en vehículos, de preferencia a la entrada de algún parqueadero. Adicionalmente el sistema será complementado con el desarrollo de una aplicación para detección y extracción de la placa de vehículos,

2.- Normalización de imágenes de placas.- Una vez que la placa ha sido detectada, la imagen de la placa debe ser normalizada. En este sentido el objetivo principal de esta parte es implementar una aplicación para alinear la imagen de la placa, así como ajustar la imagen resultante para los posteriores

pasos de reconocimiento del número de placa vehicular

3.- Segmentación de placas.- Este módulo tiene como propósito desarrollar una aplicación para segmentar la imagen de la placa previamente normalizada.

El resultado obtenido de este proceso de segmentación es etiquetar cada uno de los caracteres y números que contiene la placa, para posteriormente proceder al reconocimiento de estos elementos.

4.- Reconocimiento de número de placas.- Este módulo tiene como objetivo aplicar técnicas de reconocimiento de objetos para identificar los caracteres y números de la imagen de la placa.

Desarrollar una aplicación para reconocimiento de número de placas de vehículos usando modelos estadísticos de clasificación.

5.- Interfaz gráfica.- El objetivo principal de este módulo es desarrollar una interfaz gráfica que permita realizar análisis estadístico usando la información/parámetros extraídos desde el sistema de control de acceso vehicular. Este análisis permitirá generar gráficas dinámicas de interés del usuario con la información de los vehículos que entran y salen del parqueadero.

El proyecto propuesto en este trabajo final de carrera se enfoca, se enfoca en la implementación del módulo de normalización de imágenes de placas

vehiculares.

El objetivo principal del proyecto es implementar una aplicación para alinear la imagen de la placa, así como ajustar la imagen resultante para los posteriores pasos de reconocimiento del número de placa vehicular.

INDICE GENERAL

INDICE IMÁGENES

INTRODUCCIÓN

El procesamiento de imágenes se ha convertido en un tema de actualidad y de gran aplicación en beneficio de la sociedad con el pasar de los años y con los diversos avances tecnológicos.

El Tratamiento Digital de Imágenes contempla el procesamiento y análisis de figuras. El procesamiento está referido a la realización de transformaciones, restauración y mejoramiento de imágenes. El análisis consiste en la extracción de propiedades y características de las imágenes, así como la clasificación, identificación y reconocimiento de patrones.

Los procesos para mejorar una imagen pueden agruparse en diferentes categorías, teniendo en cuenta el efecto que producen sobre la imagen por ejemplo la modificación del brillo y contraste, reducción de ruido, desenfoque o suavizado de bordes, mejora del enfoque o realce de contornos, delineación de contornos, detección de micro estructuras, iluminación de masas, entre otros procesos.

Uno de los métodos más utilizados para la manipulación de imágenes es la utilización de filtros, los cuales modifican las imágenes ya sea para detectar los bordes o patrones de una escena o para cambiar su aspecto, otra de sus características es la eliminación de ruido en la imagen. Los filtros para la detección de bordes suministran una valiosa información sobre las fronteras de

los objetos y puede ser utilizada para segmentar la imagen, reconocer objetos. Existen una gran cantidad de filtros para la detección de bordes, los más utilizados son Canny, Hought, Harris, etc.

El Reconocimiento Óptico de Caracteres (OCR) es un método para reconocer la parte textual de una imagen digitalizada. El OCR recibirá como entrada la imagen digitalizada y el resultado de esto es un archivo de texto que puede ser editado y usado como tal por cualquier programa o aplicación que lo necesite.

PROBLEMA

El proyecto general se genera de la necesidad de controlar el acceso a los parqueaderos, tanto en el ingreso de los vehículos como tiempo de permanencia en los mismos.

Para esto, el problema ha sido dividido en varios módulos:

1. Modulo de adquisición
2. Modulo de normalización de placas
3. Modulo de segmentación de placas
4. Modulo de reconocimiento de placas

PROBLEMA ESPECÍFICO

Nuestro proyecto específico corresponde a la etapa de “Normalización de

imágenes de placas". Para lo cual se recibe una imagen ROI que define la región de interés (placa del vehículo) sobre la imagen original, útil cuando queremos realizar un análisis más cercano de un área específica.

Usualmente la imagen ROI puede contener información indeseable como por ejemplo ruido o borrosa, así como la orientación de la imagen puede tener una inclinación vertical, o una vista en perspectiva, nuestro objetivo específico es corregir esta imagen inicial.

OBJETIVO GENERAL

Para evitar la presencia de información indeseable en la imagen ésta será procesada mediante algoritmos de restauración y realzado de imágenes de tal forma de eliminar la presencia de ruido y de emborronamiento, mientras que se usarán transformaciones geométricas para corregir los problemas de orientación de la placa. El resultado de nuestro tema debe ser una placa alineada tanto horizontal como verticalmente, de esta manera quedaría ajustada para los siguientes pasos de segmentación y reconocimiento de la placa vehicular.

JUSTIFICACIONES

La aplicación de Normalización de imágenes de placas junto con otras técnicas ayudará al reconocimiento automático de matrículas que podrá ser utilizado de distintas maneras como por ejemplo en parqueaderos o centro comerciales el sistema se puede integrar junto con un control de acceso, esto ayuda a

sistematizar el ingreso y salida de vehículos, de igual forma genera mayor seguridad ya que a la salida la cámara y el control de acceso operan juntos, y el ticket que se presenta a la salida, debe coincidir con la placa del vehículo.

Utilizar todos los parámetros de control que se obtienen para generar modelos de comportamientos al ingreso o salida de un parqueadero. Esto permite detectar o predecir problemas o signos de interés para el control.

Se puede usar en otros ámbitos (no solo en parqueaderos) por ejemplo:

- Como método de recaudación electrónica de peaje en las autopistas de pago.
- Para vigilar la actividad del tránsito en una luz roja en una intersección.
- Para vigilar el exceso de velocidad en las carreteras.

METODOLOGÍA

Las correcciones geométricas tienen como finalidad orientar los píxeles de una imagen en un sistema de coordenadas de referencia.

El procedimiento utilizado se basa en la detección de las esquinas de la placa en la imagen dada, con esta información se obtienen las coordenadas (x,y) de las esquinas de la placa (4 esquinas), para luego a partir de estos puntos utilizar un algoritmo que realizará la corrección de la imagen original.

Los pasos seguidos para esta implementación son:

- **Preprocesamiento.-** Dada una imagen de entrada que contiene la placa vehicular se le aplica una binarización. La imagen resultante de este proceso tendría menos información, lo cual ayudaría a procesar la imagen con menos datos y de una forma más eficiente.

  Luego se aplican dilatación y erosión (operadores morfológicos) para eliminar ruidos innecesarios que se pueden generar en la imagen previamente binarizada.

- **Detección de esquinas y bordes de la placa.-** En este proceso inicialmente se detectan los bordes de la imagen binarizada para poder identificar si la placa se encuentra en una correcta posición. Para esto usaremos Canny para detección de bordes y aplicaremos el método de Laplace para el realzado de bordes.

  Luego con el propósito de obtener objetos de forma rectangular que se asemejen a una placa usaremos aproximación poligonal a partir de los bordes detectados. Esta aproximación poligonal generara polígonos cerrados cuya principal característica es que consisten de 4 vértices.

  Finalmente las coordenadas (x, y) de cada uno de los 4 vértices son

obtenidas. Esta información servirá para poder normalizar el polígono que representa a la placa.

- **Corrección geométrica.-** En este proceso se aplican los algoritmos de corrección geométrica encontrados para obtener la imagen en una posición correcta.

ORGANIZACIÓN DE LA TESIS

Esta tesis ha sido organizada tal como se detalla a continuación:

En el primer capítulo daremos una revisión bibliográfica al reconocimiento automático de placas de vehículos (ANPR), también información acerca de las transformaciones geométricas y su clasificación.

En el segundo capítulo se describen las técnicas usadas en el desarrollo del proyecto, como también se describen varias de las funciones usadas.

En el tercer capítulo tendremos la descripción de las etapas de desarrollo de nuestro proyecto, tanto en implementación como en corrección de errores.

El cuarto capítulo describe las pruebas realizadas para la corrección geométrica de placas vehiculares. Como resultado de las pruebas se incluye el alcance de la detección en una región determinada.

Finalmente, presentaremos las conclusiones obtenidas, y se plantean

recomendaciones de futuros trabajos relacionados.

CAPITULO I
REVISIÓN BIBLIOGRÁFICA

Resumen.-

En este capítulo detallaremos información bibliográfica acerca de los sistemas de reconocimiento automático de número de placas o también llamados ANPR (Automatic number plate recognotion en ingles), basándonos esencialmente en las transformaciones geométricas para la normalización de imágenes punto especifico de nuestra tesis.

1.1 Introducción

Resumen ANPR

El reconocimiento automático de matrículas (Automatic number plate recognotion o ANPR en ingles) es un método de vigilancia que utiliza reconocimiento óptico de caracteres en imágenes para leer las placas de los vehículos.

El ANPR se puede utilizar para almacenar las imágenes capturadas por las cámaras, así como el texto de la matrícula. Estos sistemas a menudo utilizan iluminación infrarroja para hacer posible que la cámara pueda tomar imágenes en cualquier momento del día. En al menos una versión

de cámara para la supervisión de intersecciones se incluye un flash de gran alcance, que sirve para iluminar la escena y hacer que el infractor se dé cuenta de su error. La tecnología ANPR tiende a ser específica para una región, debido a la variación entre matrículas de un lugar a otro.

Fig 1.1 – Imagen vehicular obteniendo varios sectores de interés

El reconocimiento automático de placa vehicular no significa más ni menos que la automatización de datos. El reconocimiento automático de la placa vehicular substituye o elimina la tarea de digitar manualmente el número de la placa del vehículo en el sistema informático.

Cuando se habla del sistema de reconocimiento del número de placa se entiende generalmente un sistema informático que efectúa el reconocimiento automático de la placa para automatizar la entrada de datos. El reconocimiento de la placa, abreviado como LPR, y el reconocimiento automático de la matrícula, abreviado como ANPR son los términos más comúnmente usados.

El reconocimiento automático de la es una forma especial del reconocimiento de caracteres ópticos (OCR). Este sistema es un tipo de tecnología, principalmente software, que permite leer automáticamente el número de registro o placa de los vehículos a partir de imágenes digitales. La lectura automática del número de registro significa transformar los píxeles de la imagen digital de la matrícula en texto en código ASCII.

Usualmente en la literatura podemos encontrar cinco algoritmos principales que un software de ANPR necesita para identificar una matrícula:

1. Localización de la matrícula - responsable de encontrar y aislar la región de la imagen de entrada que contiene únicamente información de la matrícula.

2. Orientación y tamaño de la matrícula - compensa los ángulos que hacen que la matrícula parezca "torcida" y ajusta las dimensiones al

tamaño requerido.

3. Normalización - ajusta el brillo y el contraste de la imagen.
4. Segmentación de los caracteres - encuentra los distintos caracteres presentes en la matrícula.
5. Reconocimiento óptico de caracteres.

Análisis sintáctico y geométrico - comprueba los caracteres encontrados y sus posiciones con las reglas específicas del país al que pertenece la matrícula.

La complejidad de cada una de estas subdivisiones del programa determina la exactitud del sistema. Durante la tercera fase (normalización) algunos sistemas utilizan técnicas de detección de borde para aumentar la diferencia en la imagen entre las letras y el fondo de la placa. También se puede utilizar un filtro digital de punto medio para reducir el "ruido" visual de la imagen.

El software de un sistema de ANPR debe ser capaz de afrontar diferentes dificultades posibles, que incluyen:

- Resolución pobre de la imagen de entrada, a menudo porque la matrícula está demasiado lejos, aunque este problema a menudo es

resultado del uso de una cámara de baja calidad.

- Imágenes desenfocadas, en particular desenfoque de movimiento y muy a menudo en unidades móviles.
- Iluminación pobre y bajo contraste debido a sobreexposición, reflexión o sombras.
- Un objeto que oscurece parte de la matrícula, a menudo una barra del remolque, o suciedad en la matrícula.
- Técnicas de evasión

Fig 1.2 – Imagen obtenida de una placa vehicular internacional

Los primeros sistemas ANPR eran incapaces de leer letras blancas o plateadas sobre un fondo negro, como se permitía en los vehículos del Reino Unido fabricados antes de 1973.

Aunque algunos de estos problemas se pueden corregir en el software, se dejan sobre todo en el lado del hardware del sistema para ofrecer soluciones a estos problemas. El aumento de la altura de la cámara puede

evitar problemas con los objetos (tales como otros vehículos) que oscurecen la placa, pero introduce y aumenta otros problemas como el ajuste según la oblicuidad creciente de la placa.

Muchos países utilizan matrículas retroreflectivas. Esto devuelve la luz hacia la fuente y mejora así el contraste de la imagen. En algunos países los caracteres de la matrícula no son reflectantes, dando un alto nivel del contraste con el fondo reflectante bajo cualquier condición de iluminación. Una cámara que utiliza imagen infrarroja (con un filtro normal de color sobre la lente y una fuente luminosa infrarroja al lado de ella) beneficia en gran medida, reflejándose las ondas infrarrojas desde la matrícula. Sin embargo, esto sólo es posible en cámaras ANPR dedicadas, por lo que las cámaras usadas para otros propósitos deben confiar en mayor medida en las capacidades del software. Además, cuando se necesita una imagen a todo color y la captación de detalles es necesario tener una cámara con infrarrojos y una cámara normal (en color) funcionando conjuntamente.

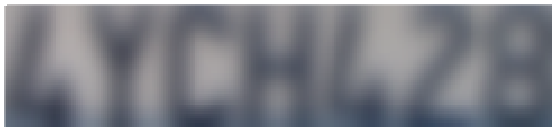


Fig 1.3 – Imagen borrosa y desenfocada

Imágenes borrosas dificultan el OCR – los sistemas ANPR deberían tener

altas velocidades de disparo para evitar el desenfoque de movimiento

Para evitar el desenfoque es ideal tener la velocidad del obturador de la cámara fijada a 1/1000 segundos. Debido a que el coche está en movimiento, el uso de velocidades más reducidas podría dar lugar a una imagen demasiado borrosa para ser leída con el software OCR, especialmente si la cámara está en una posición mucho más alta que el vehículo. Cuando el tránsito es lento o cuando la cámara está a una altura inferior y el vehículo está en un ángulo de aproximación a la cámara, no es necesario que la velocidad del obturador sea tan alta. Velocidades del obturador de 1/500 pueden funcionar correctamente con vehículos con una velocidad de hasta 64 kilómetros por hora y 1/250 hasta 8 kilómetros por hora.

Algunos sistemas a escala reducida permiten algunos errores en la matrícula. Cuando se utiliza para ofrecer acceso específico de los vehículos a una zona con barrera, la decisión puede ser tomada con un índice de error aceptable de un carácter. Esto es así porque la probabilidad de que un coche desautorizado con una matrícula tan similar se considera que es absolutamente pequeña. Sin embargo, este nivel de imprecisión no sería aceptable en la mayoría de las aplicaciones de un sistema ANPR.

Nuestro objetivo del proyecto es la normalización de imágenes para lo cual usaremos transformaciones geométricas las cuales nos ayudarán a realizar nuestro trabajo.

1.2 Transformaciones Geométricas

Se define una transformación geométrica como la operación que posibilita obtener una figura nueva a partir de otra dada. Por medio de esta transformación se establece una serie de correspondencias entre elementos (puntos, rectas) o figuras.

Con el nombre de movimientos se denominan las transformaciones geométricas que conservan la forma y el tamaño de la figura original.

Las transformaciones geométricas permiten eliminar las distorsiones geométricas que ocurren cuando se captura una imagen, por ejemplo, cuando se comparan imágenes de la misma zona tomadas con un cierto intervalo de tiempo de diferencia (probablemente la posición no sea exactamente la misma) o cuando el plano de captación no es el plano de los objetos de interés.

Para inspeccionar los cambios es necesario efectuar un proceso previo de “alineamiento”

1.3 Clasificación de las Transformaciones Geométricas

Existen muchos tipos de transformaciones geométricas, desde las más simples a las más complejas:

Afines, bilineales, perspectivas, basadas en superficies deformantes, morphing, mapeo arbitrario, etc.

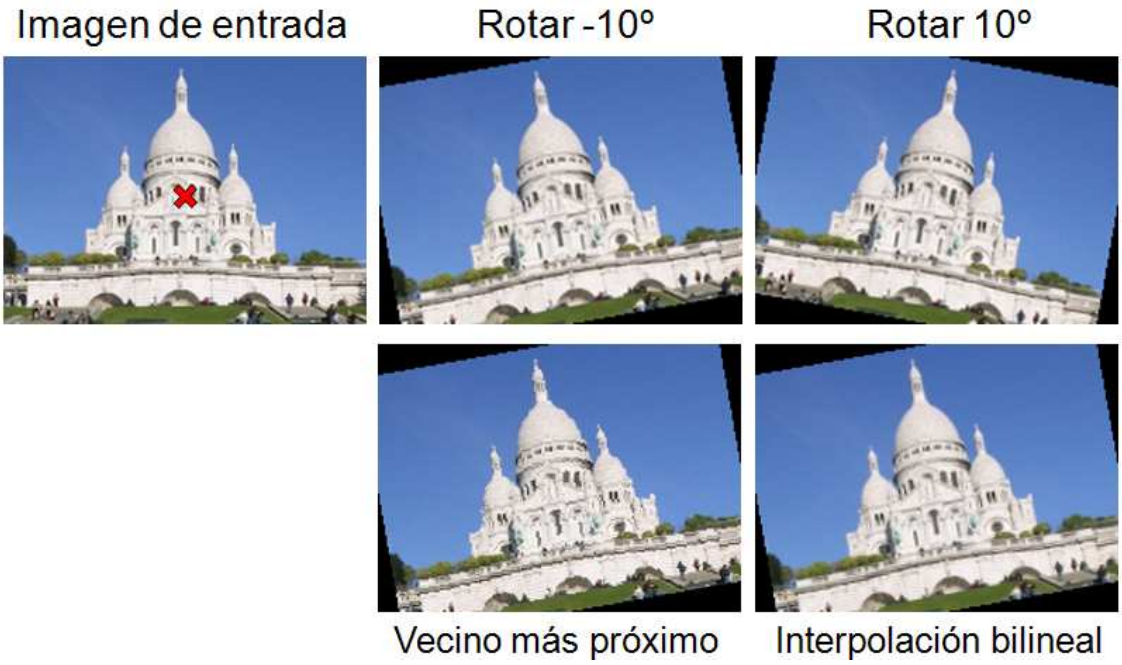


Fig 1.4 - Imagen aplicando transformaciones geométricas

1.3.1 Interpolación Bicúbica

Igual que la bilineal, se basa en dos interpolaciones cúbicas:

1. Interpolación cúbica horizontal, en las filas existentes (usando 4 puntos).

2. Interpolación cúbica vertical en todo el espacio usando 4 puntos.

La interpolación bicúbica siempre suele producir el mejor resultado, aunque es algo más costosa.

Imagen original 25x26

Aumento de 10x con interpolación bicúbica

Fig 1.5 – Interpolación bicúbica

1.3.2 Transformaciones Afines

Se entiende por transformación afín cualquier tipo de rotación escalado o traslación que se produzca en las dos direcciones espaciales del plano. Como transformaciones afines principales podemos citar: el escalado, la rotación y la traslación.

El elemento básico de toda transformación afín es la matriz de transformación, que será la encargada de definir la clase y los parámetros de la transformación a realizar. Así, tendremos siempre una matriz asociada tanto para el escalado, como para la rotación, como para la traslación.

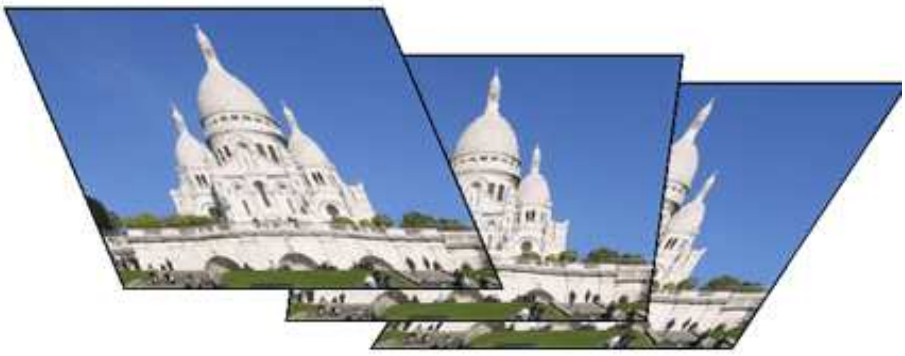


Fig 1.6 – Transformaciones afines

Traslación.- La traslación también es conocida como desplazamiento, y se basa en el cambio de posición de un objeto de acuerdo con una fórmula.

Por lo tanto, la imagen de salida obtenida se relaciona con la entrada de la siguiente manera:

Img_resultado (x,y) = Img_origen (x + dx, y + dy).

Siendo, dx = desplazamiento en el eje horizontal, dy = desplazamiento en el eje vertical.

En Opencv toda transformación afín se puede implementar con la función cvWarpAffine que genera una imagen de salida a la que se le aplica la transformación correspondiente.

La aplicación más inmediata y típica de las transformaciones afines es extraer y redimensionar un área de interés, dándole una forma predefinida de antemano. Esto es lo que se llama normalización.

Fig 1.7 – Imagen trasladada

Escalado.- El escalado es una transformación afín que puede aumentar o disminuir un objeto por un determinado factor.

La imagen que se obtiene a la salida se relaciona con la entrada siguiendo la siguiente regla:

Imag_destino (x , y) = Imag_origen (ex*x , ey*y) siendo:

ex: escala en el eje x

ey: escala en el eje y

Con la librería Opencv existen dos modos básicos de realizar estas operaciones: podemos utilizar la función genérica de transformaciones afines cvWarpAffine, o una función específica de escalado cvResize que nos aporta un grado mayor de versatilidad.

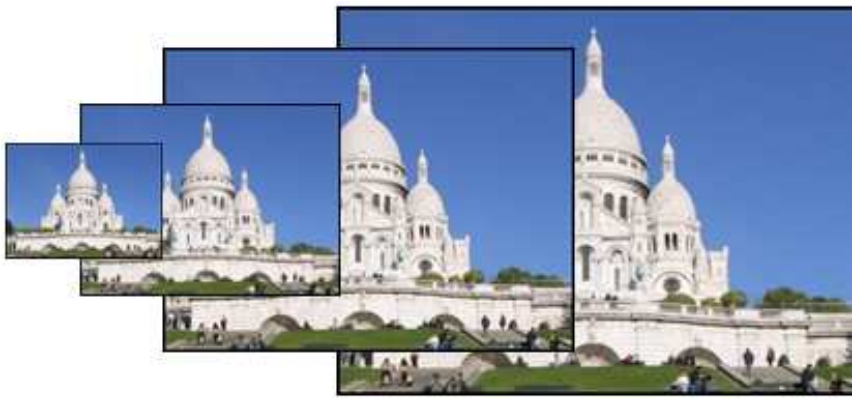


Fig 1.8 – Imagen escalada

Rotación.- Es una transformación en la que los ejes de coordenadas son rotados un determinado ángulo respecto al origen.

Las operaciones en las coordenadas que se llevan a cabo en la rotación se pueden sintetizar en la siguiente relación:

Imagen_resultado(x,y)= Imagen_inicial(x·cos α + y·sen α, x·sen α + y·cos α)

siendo α = ángulo de rotación

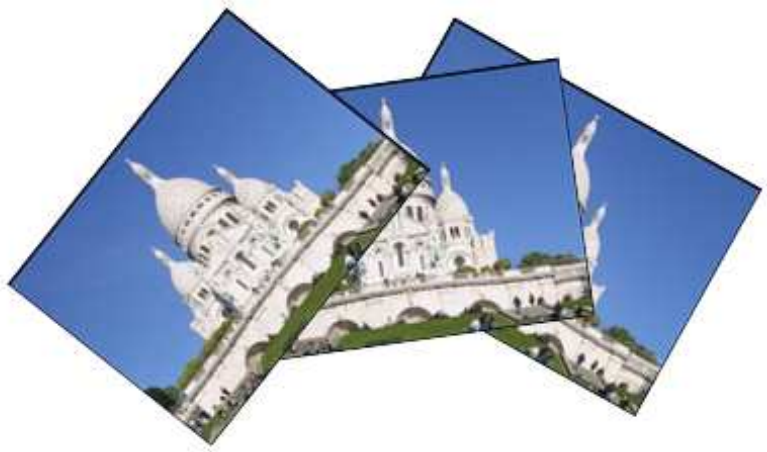


Fig 1.9 – Imagen rotada

1.3.3 Transformaciones Bilineal y Perspectivas

Las transformaciones bilineal y perspectiva se pueden ver como generalizaciones de las afines:

Transformación afín: cualquier rombo se mapea en un rombo.

Transf. bilineal y perspectiva: cualquier cuadrilátero se transforma en otro cuadrilátero (ambos convexos).

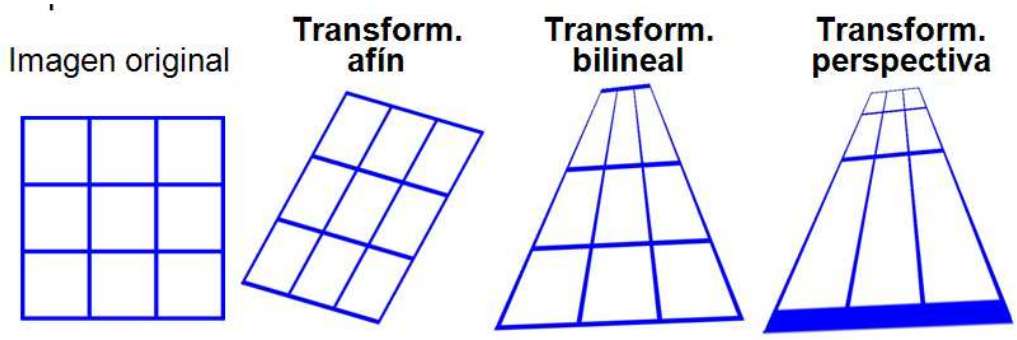


Fig 1.10 – Transformación bilineal y perspectiva

1.4 Técnica usada – Transformaciones Perspectivas

Para las transformaciones perspectivas, las operaciones disponibles en OpenCV son similares a las afines.

Aplicar una transformación: cvWarpAffine → cvWarpPerspective

Calcular los coeficientes: cvGetAffineTransform → cvGetPerspectiveTransform

En este caso la matriz que define la transformación es un CvMat de 3x3 de tipo CV_64FC1 o CV_32FC1.

OpenCV no tiene las transformaciones bilineales.

Hemos seleccionado este tipo de transformación geométrica para la realización de nuestro proyecto debido al costo computacional de las funciones y algoritmos usados en perspectivas, también son algoritmos de fácil comprensión y utilización.

Así mismo al usar esta técnica nos ahorramos realizar varias combinaciones de técnicas, como por ejemplo al usar las transformaciones afines nos tocaba realizar varias secuencias (rotación, traslación, escalonamiento) para obtener la imagen de la placa visualizada correctamente.

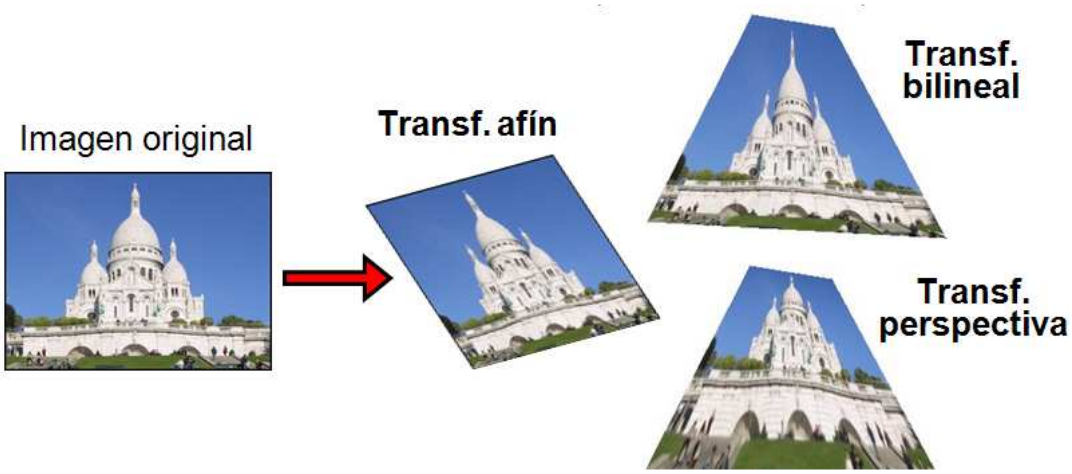


Fig 1.11 – Transformación perspectiva

CAPITULO II

Descripción de los operadores y técnicas usadas.

2.1 Resumen.-

En este capítulo se da una breve introducción a los operadores y técnicas usados en el transcurso de nuestro proyecto, los mismos que han sido utilizados para la implementación de las soluciones planteadas para cada uno de los problemas e inconvenientes generados en las transformaciones geométricas de la placa.

2.2 Binarización.-

La binarización de imágenes es una tarea básica en muchas aplicaciones de procesamiento digital de imágenes. El objetivo es obtener una imagen que solo sea representada por dos tonos de color, por general: blanco y negro. La idea para realizar este trabajo es sencilla, solo debemos decidir que tono de color dar a cada píxel que sea mayor que un determinado umbral (valor límite), el resto de píxeles tendrán por defecto el otro tono de color.

2.2.1 Descripción de los 24 puntos a analizar

Para nuestro proyecto se nos proporcionaron imágenes tomadas en 24 puntos distintos con diferentes distancias y ángulos de visualización.

Las imágenes adquiridas en cada punto son analizadas a manera de encontrar el rango de binarización que utilizaremos en nuestro proyecto.

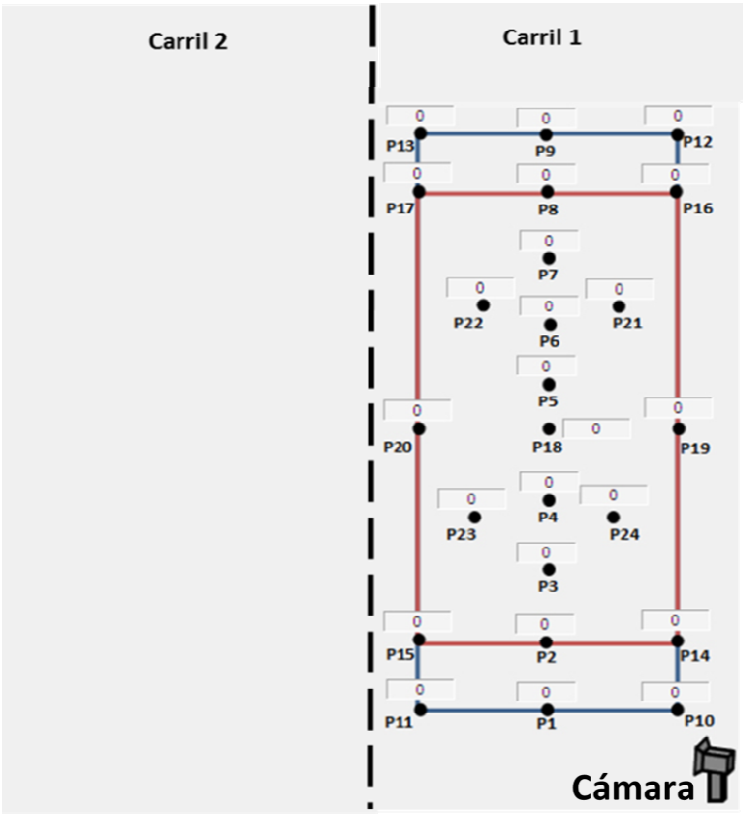


Fig 2.1 – Detalles de los 24 puntos en el carril

2.2.2 Análisis y resultados de los puntos

En el proceso y análisis de imagen, la binarización se emplea para separar las regiones u objetos de interés en una imagen del resto. Las imágenes binarias se usan en operaciones booleanas o lógicas para identificar individualmente objetos de interés o para crear mascaras sobre regiones.

En muchos casos, una imagen binaria es el resultado de una segmentación por niveles de gris o de una segmentación por selección de un rango de color determinado. En otros casos, una imagen binaria es simplemente el resultado de una selección interactiva de regiones de interés, las cuales se utilizaron como máscaras de comparación o referencia.

A continuación presentamos las imágenes en todos los puntos donde se realizo el análisis de binarización en los 24 puntos dados.

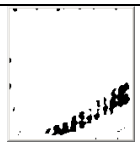
IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización – 17		Binarización – 23

Fig 2.2 – Binarización de una placa vehicular en el punto 1

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.3 – Binarización de una placa vehicular en el punto 2

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.4 – Binarización de una placa vehicular en el punto 3

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.5 – Binarización de una placa vehicular en el punto 4

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.6 – Binarización de una placa vehicular en el punto 5

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.7 – Binarización de una placa vehicular en el punto 6

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.8 – Binarización de una placa vehicular en el punto 7

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.9 – Binarización de una placa vehicular en el punto 8

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.10 – Binarización de una placa vehicular en el punto 9

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.11 – Binarización de una placa vehicular en el punto 10

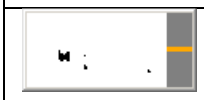
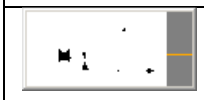
IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.12 – Binarización de una placa vehicular en el punto 12

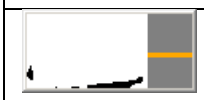
IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.13 – Binarización de una placa vehicular en el punto 13

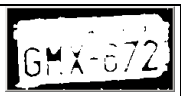
IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.14 – Binarización de una placa vehicular en el punto 14

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.15 – Binarización de una placa vehicular en el punto 15

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.16 – Binarización de una placa vehicular en el punto 16

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.17 – Binarización de una placa vehicular en el punto 17

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.18 – Binarización de una placa vehicular en el punto 18

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.19 – Binarización de una placa vehicular en el punto 19

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.20 – Binarización de una placa vehicular en el punto 20

IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.21 – Binarización de una placa vehicular en el punto 21

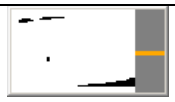
IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.22 – Binarización de una placa vehicular en el punto 22

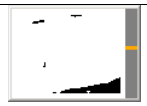
IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.23 – Binarización de una placa vehicular en el punto 23

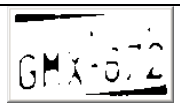
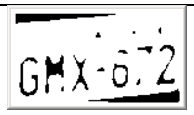
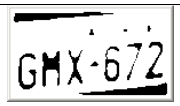
IMAGEN ORIGINAL	IMAGEN BINARIZADA		IMAGEN BINARIZADA	
		Binarización – 13		Binarización – 19
		Binarización – 15		Binarización – 21
		Binarización– 17		Binarización – 23

Fig 2.24 – Binarización de una placa vehicular en el punto 24

3.2.3 Selección del mejor rango de visualización para los 24 puntos

Después de realizar las pruebas necesarias con los distintos valores usados para la binarización {10 - 24}, los 24 puntos de enfoque de las imágenes de la placa nos presentan distintas imágenes binarizadas resultantes. De este modo se analizó cada imagen binarizada para encontrar el rango de umbralización mas apropiado para nuestro proyecto.

Como resultado se obtuvo que el rango global de binarización apropiado para los 24 puntos es de {10-24}, en los cuales la imagen resultante

elimina la mayor cantidad de ruido y presenta la placa sin perder la información necesaria para la continuación de nuestro proyecto.

2.2.4 Resultados Finales

A continuación presentamos una tabla donde mostramos la binarización para cada punto y los valores de binarización donde la imagen resultante es la óptima para realizar nuestro siguiente paso.

Puntos	Imagen tomada para análisis	Valores de Binarización
Punto 1	imagen 1	11,12,18,19,20
Punto 2	imagen 1	17,18
Punto 3	imagen 1	14,15
Punto 4	imagen 1	11,12
Punto 5	imagen 1	13,14,15
Punto 6	imagen 1	10,17,18
Punto 7	imagen 1	11,13,15,16,17,18
Punto 8	imagen 1	11,12,16,17
Punto 9	imagen 1	16
Punto 10	imagen 1	11,12
Punto 11	imagen 1	

Punto 12	imagen 1	10,13,15,16,19,22,24
Punto 13	imagen 1	11,12,13,19,20,21
Punto 14	imagen 1	10,11,13,15,24
Punto 15	imagen 1	11,12,16
Punto 16	imagen 1	10,14,17
Punto 17	imagen 1	11,12,13,16,18,19
Punto 18	imagen 1	11,14,15,17,21,22,23,24
Punto 19	imagen 1	11,13,14,15,16
Punto 20	imagen 1	10,11,12,13,15,16
Punto 21	imagen 1	12,13,15,18,21,22,23
Punto 22	imagen 1	11,12,13,18,20,21,22
Punto 23	imagen 1	10,11,14,15,16
Punto 24	imagen 1	11,13,16

Tabla 1 – Rango de binarización optimo para los distintos puntos

2.3 Canny.-

En el área de procesamiento de imágenes, la detección de los bordes de una imagen es de suma importancia y utilidad, pues facilita muchas tareas, entre ellas, el reconocimiento de objetos, la segmentación de regiones, entre otras.

Se han desarrollado variedad de algoritmos que ayudan a solucionar este inconveniente. El algoritmo de Canny es usado para detectar todos los bordes existentes en una imagen. Este algoritmo está considerado como uno de los mejores métodos de detección de contornos mediante el empleo de máscaras de convolución y basado en la primera derivada. Los puntos de contorno son como zonas de píxels en las que existe un cambio brusco de nivel de gris. En el tratamiento de imágenes, se trabaja con píxels, y en un ambiente discreto, es así que en el algoritmo de Canny se utiliza máscaras, las cuales representan aproximaciones en diferencias finitas.

El algoritmo de Canny consiste en tres grandes pasos:

- Obtención del gradiente: en este paso se calcula la magnitud y orientación del vector gradiente en cada píxel.

- Supresión no máxima: en este paso se logra el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho.

- Histéresis de umbral: en este paso se aplica una función de histéresis basada en dos umbrales; con este proceso se pretende reducir la posibilidad de aparición de contornos falsos.

2.3.1 Uso de Canny.-

En nuestro proyecto el algoritmo de Canny es usado para poder detectar el borde de la placa. Después de realizar la binarización de la misma, usaremos el algoritmo de Canny con las imágenes resultantes del proceso anterior es decir las binarizadas en el rango {10-24}, con esto podemos obtener los cuadros de la placa con gradiente sombreado.

Funcion usada: cvCanny(image,edges, threshold1, threshold2, aperture_size);

image.- Imagen de entrada

***edges*.-** Imagen de salida

threshold1.- El menor nivel de brillo que tendrá el borde, para nuestro algoritmo se determino un valor constante de 50.

threshold2.- El máximo nivel de brillo que tendrá el borde, para nuestro algoritmo se determino un valor constante de 200.

aperture_size.- Parámetro de la apertura del operador Sobel, para nuestro algoritmo se determino un valor constante de 200.

2.3.2 Resultados.-

Después de obtener las imágenes binarizadas al aplicar Canny resulta mucho más sencillo encontrar los bordes de la placa sin ruido e información innecesaria como podemos ver en las siguientes imágenes tomadas de distintos puntos.

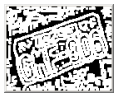
IMAGEN ORIGINAL	BORDES DETECTADOS - CANNY	Puntos analizados
		Punto 1
		Punto 4
		Punto 10
		Punto 14

Fig 2.25 – Bordes detectados con Canny

2.4 Operadores Morfológicos.-

Los operadores de morfología son un conjunto de filtros locales sencillos, que se pueden combinar para obtener resultados más complejos.

Originalmente, están definidos sobre imágenes binarias. La idea es muy parecida a una convolución, pero utilizando las operaciones booleanas

AND y OR. El elemento estructurante define los píxeles que se usan en la operación y los que no. Dado un elemento estructurante, E, de cierta forma y tamaño, y una imagen binaria B, se definen dos operaciones: Dilatación y Erosión.

2.4.1 Dilatación.-

Dilatación B⊕E. Combinar con OR los valores correspondientes a los píxeles 1 del elemento estructurante. El efecto de la dilatación es extender o ampliar las regiones de la imagen con valor 1 (color blanco). La cantidad depende del tamaño y forma del elemento estructurante y del número de veces que se aplican.

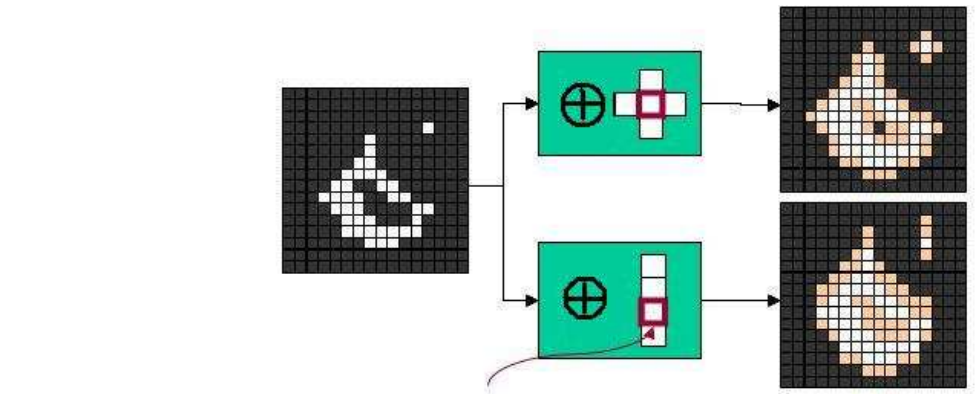


Fig 2.26 – Dilatación

Propiedades:

- Añade todos los puntos del fondo que tocan el borde de un objeto →la dilatación es extensiva.
- Rellena entrantes en los que no quepa el EE (pequeños agujeros y bahías)

2.4.2 Erosión.-

Erosión B⊗E. Combinar con AND los valores correspondientes a los píxeles 1 del elemento estructurante. El efecto de la erosión es reducir las regiones de la imagen.

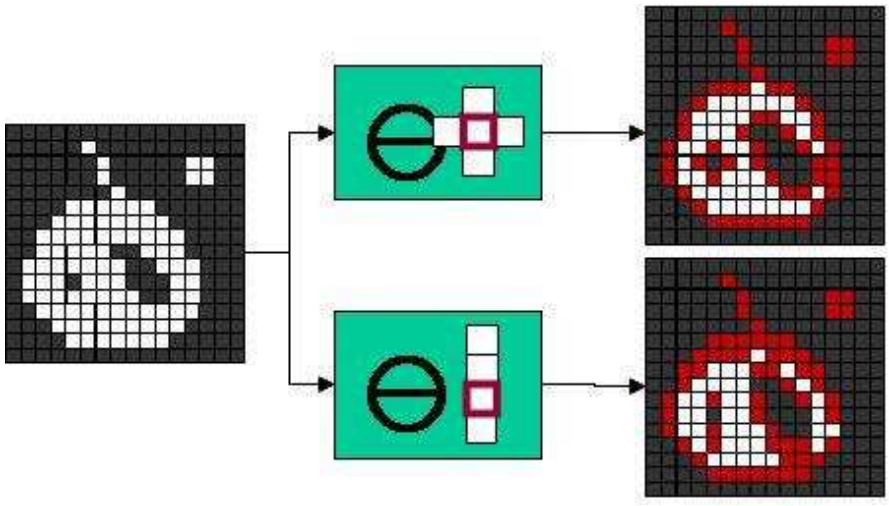


Fig 2.27 – Erosión

Propiedades:

- La erosión es antiextensiva → reduce el tamaño del objeto
- Elimina elementos en los que no quepa el EE (pequeñas islas y protuberancias)
- ¿Es la erosión inversa de la dilatación?
  - No, pero están relacionadas:

2.4.3 Operaciones basadas en dilatación y erosión.-

Existen otras dos operaciones frecuentes basadas en erosión y dilatación:

Abrir. Aplicar erosión y después dilatación: $(B \otimes E) \oplus E$

Cerrar. Aplicar dilatación y después erosión: $(B \oplus E) \otimes E$

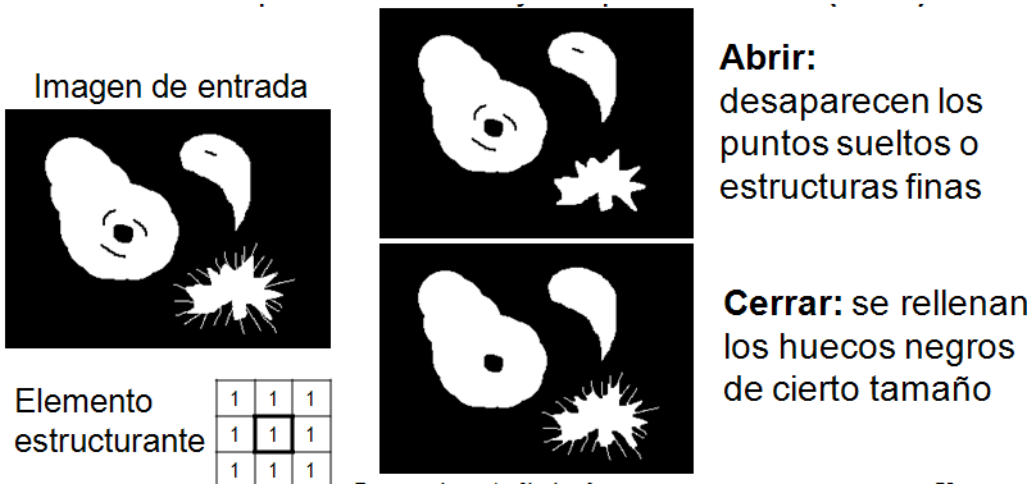


Fig 2.28 – Abrir y cerrar, operaciones basadas en erosión y dilatación

2.4.4 Resultados.-

En nuestro proyecto utilizamos después de realizar la binarización la siguiente operación:

Cerrar = Erosion -> Dilatacion

Abrir = Dilatación -> Erosión

Operaciones que nos dan como resultado imágenes sin ruido e información innecesaria.

IMAGEN ORIGINAL	IMAGEN RESULTANTE	PUNTO ANALIZADO
		Punto 15
		Punto 10
		Punto 14

Fig 2.29 – Imágenes resultantes de varios puntos después de aplicar operadores morfológicos

En los resultados como podemos observar en las imágenes anteriores tenemos la forma de la placa lista para proseguir con los siguientes procesos.

2.5 Detección de Contornos.-

La detección de contorno es parte de un proceso de aislamiento (segmentation), que consiste en la identificación de objetos dentro de una imagen.

Como es usual, hay varias posibles definiciones de un contorno, siendo cada una aplicable en distintas circunstancias. Una de las más comunes y generales definiciones es el contorno de paso ideal, los contornos pueden ser de una dimensión, donde el contorno es simplemente un cambio en el nivel de gris que ocurre en una ubicación especifica. Cuanto mayor es el cambio de nivel, más fácil resulta detectar el contorno, pero en el caso ideal cualquier cambio de nivel puede ser visto fácilmente.

La primera complicación ocurre debido a la digitalización. Es poco probable que una imagen sea muestreada de manera tal que todos los contornos correspondan exactamente con un pixel del borde. Por lo general, el cambio de nivel puede extenderse sobre varios pixeles.

El ruido es un efecto aleatorio, y es caracterizado sólo estadísticamente. El resultado del ruido en una imagen es que produce variaciones aleatorias en el nivel de un pixel a otro, y entonces las líneas suaves y rampas de los contornos ideales nunca son encontradas en las imágenes reales.

La **función** de Open CV utilizada para este fin es:

int cvFindContours(CvArr* img, CvMemStorage* storage, CvSeq** firstContour, int headerSize=sizeof(CvContour), CvContourRetrievalMode

mode=CV_RETR_LIST, CvChainApproxMethod method=CV_CHAIN_APPROX_SIMPLE);

- img: Representa a la imagen de entrada de un solo canal. Para obtener una imagen binaria desde una imagen de escala de grises se puede utilizar cvThreshold , cvAdaptiveThreshold o cvCanny . La función modifica el contenido de la imagen fuente.
- Storage: Contenedor de los contornos obtenidos.
- firstContour: Parámetro de salida, contiene el puntero a la primera contorno exterior.
- headerSize: Tamaño de la cabecera de la secuencia
- Mode: Recuperación de modo.
- Method: Representa a la aproximación del método.

La función cvFindContours recupera los contornos de la imagen binaria y devuelve el número de contornos recuperados. El puntero firstContour es llenado por la función. Contendrá puntero al primer contorno más externo o NULL si no se detecta contornos (si la imagen es completamente negro).

A la información de los otros contornos se puede llegar desde firstContour con h_next y v_next enlaces. La muestra en cvDrawContours muestra cómo utilizar contornos para la detección de componentes conectados. Los

contornos también pueden ser utilizados para el análisis de la forma y el reconocimiento de objetos.

Los contornos cuadrados deben tener 4 vértices después de la aproximación y ser convexos.

2.5.1 Resultados.-

Las imágenes presentadas a continuación muestran la función findContours en el

primer nivel de binarización escogido en el rango previamente descrita (rango 10)

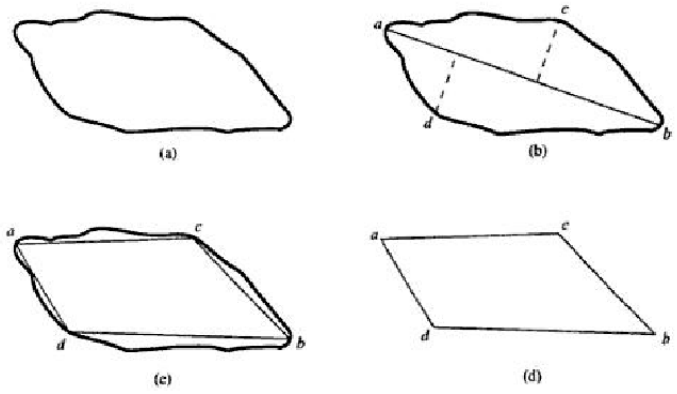
IMAGEN ORIGINAL	IMAGEN RESULTANTE	PUNTO ANALIZADO
		Punto 14
		Punto 20
		Punto 10

Fig 2.30 – Detección de contornos en varios puntos

2.6 Aproximación de Polígonos.-

Un contorno digital puede ser aproximado con una precisión arbitraria mediante un polígono. Para una curva cerrada, la aproximación es exacta cuando el número de lados del polígono es igual al número de puntos del contorno, de forma que cada par de puntos adyacentes define un lado o segmento del polígono. En la práctica, el objetivo de una aproximación poligonal es captar la esencia de la forma del contorno con un polígono del menor número posible de lados.

Para mostrar las técnicas de aproximación poligonal se empezará con un método para encontrar polígonos de perímetro mínimo. El procedimiento se explica mejor mediante un ejemplo. Suponiendo que se encierra el contorno en un conjunto de segmentos concatenados. Ayudaría a visualizar esta inclusión si se la observa como dos paredes correspondientes a los bordes exterior e interior de la sucesión de segmentos y pensando que el contorno del objeto es una tira de goma contenida entre las paredes.



(a)Contorno original; (b) Contorno dividido en dos lados basándose en el cálculo de distancias; (c) Unión de vértices; (d) Polígono aproximamente resultante.

Fig 2.31– Aproximación de polígonos

2.6.1 Funcion cvApproxPoly.-

Con el objetivo de aproximar los contornos detectados a los polígonos respectivos se usa la función de Open CV cvApproxPoly. Esta función viene definida como se detalla a continuación:

CvSeq * cvApproxPoly (const void * src_seq, int header_size, almacenamiento * CvMemStorage, método int, doble parámetro, int parámetro2 = 0);

Donde:

src_seq: representa la secuencia de la matriz de puntos que corresponden a los entornos detectados.

header_size: tamaño de la cabecera de la curva de aproximación.

Método de aproximación: El cual puede ser: CV_POLY_APPROX_DP sólo es compatible, que corresponde al algoritmo de Douglas-Peucker.

Parámetro: Método parámetro específico, en caso de CV_POLY_APPROX_DP es una aproximación de precisión deseado.

Parámetro2: Si el caso es src_seq secuencia que significa si la secuencia de un solo debería aproximarse o todas las secuencias en el mismo nivel o por debajo de src_seq.

La función cvApproxPoly se aproxima a una o más curvas y devuelve el resultado de la aproximación.

2.7 Áreas de Contornos.-

La función cvContourArea calcula el área de todo el contorno o la sección del contorno. En este último caso la superficie total delimitada por el contorno del arco y la cuerda que conecta los 2 puntos seleccionados se calcula como se muestra en la imagen siguiente:

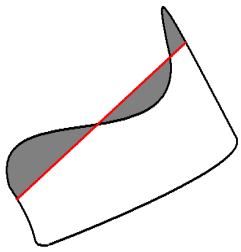


Fig 2.32–Superficie total delimitada por el contorno del arco

La orientación del contorno afecta a la señal de la zona, por tanto, la función puede devolver un resultado negativo.

Los contornos cuadrados deben tener 4 vértices después de la aproximación.

Calcula el área de todo el contorno o contorno de la sección:

doble cvContourArea (const * CvArr contorno, CvSlice rebanada = CV_WHOLE_SEQ);

Siendo:

contorno: Contorno (secuencia o serie de vértices).

Rebanada: Puntos inicial y final de la sección de curvas de nivel de interés, según el área por defecto de todo el contorno se calculan.

2.8 Warp Perspective.-

cvWarpPerspective void (const CvArr * src, dst CvArr *, const CvMat * map_matrix, int flags = CV_INTER_LINEAR + CV_WARP_FILL_OUTLIERS, CvScalar fillval = cvScalarAll (0));

- Src: Imagen de entrada.
- Dst: Imagen resultante.
- map_matrix: Matriz de transformación de tamaño 3x3.

- Flags: Una combinación de método de interpolación y las banderas opcionales siguientes:
  - o 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 .
  - o CV_WARP_INVERSE_MAP - indica que matrix 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 .

    fillval -> Un valor que se utiliza para rellenar los valores extremos

    La función cvWarpPerspective transforma la imagen de origen utilizando la matriz se especifica:

    dst (x ', y') : src (x, y)

(T • x ', t • y, t) = T map_matrix • (x, y, 1) T + b si CV_WARP_INVERSE_MAP no está establecido, (T • x, t • y, t) = T map_matrix • (x ', y', 1) T + b en caso contrario

Interpolación	**En OpenCV**
Vecino más próximo	CV_INTER_NN

Bilineal	CV_INTER_LINEAR
Bicúbica	CV_INTER_CUBIC
Supermuestreo	CV_INTER_AREA

2.9 Funciones Auxiliares.-

A continuación detallaremos varias de las funciones auxiliares utilizadas en el proyecto:

2.9.1 cvLoadImage

Para implementar la acción de abrir imágenes utilizando Opencv, se hace uso de la función cvLoadImage.

Pasamos a describir los parámetros necesarios para poder trabajar con esta función:

```
img=cvLoadImage(fileName,flag);
```

Siendo:

- fileName: Nombre del fichero que se quiere cargar
- flag: Características de carga en el fichero:

- flag: >0 : se obliga que la imagen cargada sea una imagen de color de 3 canales
- flag =0 : se obliga que la imagen cargada sea una imagen intensidad de 1 canal
- flag <0 : la imagen se carga tal cual es, con el número de canales que posea su fichero

  Cabe destacar que esta función puede recibir las imágenes en cualquier tipo de formato: BMP, DIB, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF, TIF.

  Siempre y cuando los parámetros de la misma se adecúen a la imagen en cuestión.

  Cuando dejemos de utilizar una imagen IplImage debemos liberar la memoria con:

  void cvReleaseImage(IplImage** image); de forma similar a free().

2.9.2 cvSaveImage

Se utiliza la función cvSaveImage para salvar las imágenes. Los parámetros requeridos para su uso son muy parecidos a los empleados en la función anterior.

Pasamos a describir los campos de la función:

cvSaveImage(outFileName,img)

Siendo:

- outFileName: Nombre del fichero de salida que deberá contener a la imagen a guardar.
- img: Imagen que se va a guardar

Como particularidad cabe destacar que la imagen que se salva es creada por el propio programa y almacenada en el directorio donde se encuentre el programa. Por lo tanto, no es necesario definir ningún tipo de fichero de almacenamiento con antelación.

CAPITULO III

Implementación de la normalización y corrección geométrica.

3.1 Resumen

En el presente capítulo se describe la implementación de los diferentes algoritmos que se desarrollaron, para encontrar la mejor solución para la normalización y corrección geométrica. En cada etapa se muestra el algoritmo, su implementación, análisis de resultados, corrección a los problemas encontrados y las respectivas conclusiones de las pruebas.

3.2 Algoritmo 1: Detección de silueta de la placa mediante esquinas de Harris.

En esta primera etapa se buscó corregir geométricamente la silueta de la placa buscando las líneas de los bordes superiores e inferiores de la placa mediante la transformada de Hough, para luego detectar las esquinas de estas líneas, mediante el detector de esquinas de Harris y asi poder proyectar la silueta detectada a una imagen base.

La solución planteada en la presente etapa consta de los siguientes pasos:

- Adquisición de la imagen

- Preprocesamiento
- Ecualización de histograma
- Binarización de imagen
- Detección de bordes
- Detección de líneas mediante las transformadas de Hough
- Detección de esquinas mediante CornerHarris
- Proyectar a una imagen base mediante WarpPerspective

3.2.1 Implementación

Adquisición de la imagen

Este proyecto toma como punto de partida el trabajo realizado por el primer grupo "Extracción de placas de Vehículos desde una señal de video" el cual da como resultado una imagen que contiene únicamente la placa.

Por lo tanto en este proyecto cargaremos las imágenes desde archivo mediante cvLoadImage.

cvLoadImage recibe dos parámetros:

- En el primer parámetro enviamos la dirección de nuestro disco duro local, en la cual se encuentra la carpeta con las imágenes de las placas.
- En el segundo parámetro enviamos cero para indicarle que será de un solo canal.

La función cvLoadImage carga la imagen de la ruta especificada y nos devuelve un puntero a la imagen.

El número de bits utilizados para la digitalización es de ocho, por lo tanto se utiliza 8U, con un solo canal, con lo cual obtenemos una imagen binaria. Se utilizan estos valores para tener un volumen mínimo de información y así resolver de forma óptima los siguientes procesamientos.

Preprocesamiento

La imagen de entrada en el proceso de adquisición necesita de procesamientos previos para poder realzar las características de los bordes y así poder detectarlos de una mejor manera.

Ecualización de histograma

Se realizó la ecualización del histograma para mostrar un mayor realce en los contrastes y el brillo de la imagen lo cual nos ayudará a detectar de una mejor manera los bordes de la placa. Para ello se utilizó

cvEqualizeHist el cual recibe como parámetros una imagen de entrada y una imagen de salida, que debe ser del mismo tipo es decir tener el mismo número de bits y canales.

La imagen de la Figura. 3.1. Muestra un ejemplo del proceso de ecualización del histograma, los histogramas correspondientes son dados en la Fig. 3.2.

Imagen de Entrada Imagen de Ecualizada

Fig. 3.1 Ecualización del histograma

Histograma Original Histograma Ecualizado

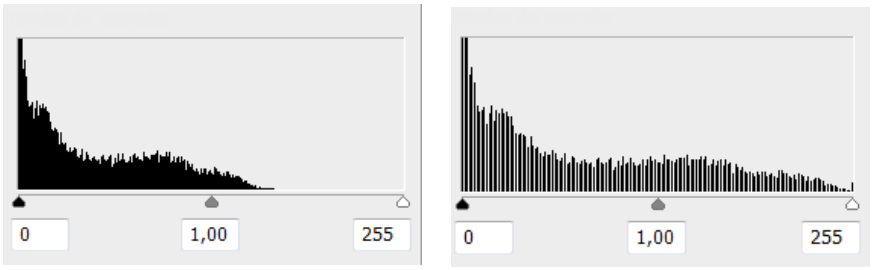


Fig. 3.2 Histograma ecualizado resultante

Binarización de Imagen

Se realiza una serie de binarizaciones con un umbral de binarización de cero a cien sobre la imagen ecualizada con el fin de encontrar la silueta de la placa e ir probando en cada uno de los niveles los métodos siguientes para la detección de los bordes de la placa. En la figura 3.3 se muestra la imagen de entrada y la resultante del proceso de binarización.

Imagen de Entrada (Ecualizada) Imagen Resultante (Binarizada)

Fig. 3.3 Imagen Binarizada

Detección de bordes

Se utiliza Canny para detectar los bordes de la imagen, en nuestro caso para detectar la los bordes de la placa, aunque también incluye los bordes de las letras.

Para ello se envió los siguientes parámetros:

```
cvCanny(image,edges, threshold1, threshold2,
aperture_size );
```

image.- Imagen de entrada

***edges*.-** Imagen de salida, Imagen en la cual se almacenarán los bordes detectados.

threshold1.- El menor nivel de brillo que tendrá el borde, para nuestro algoritmo se determino un valor constante de 50.

threshold2.- El máximo nivel de brillo que tendrá el borde, para nuestro algoritmo se determino un valor constante de 200.

aperture_size.- Parámetro de la apertura del operador Sobel, para nuestro algoritmo se determino un valor constante de 200.

Imagen de Entrada Imagen con Filtro Canny

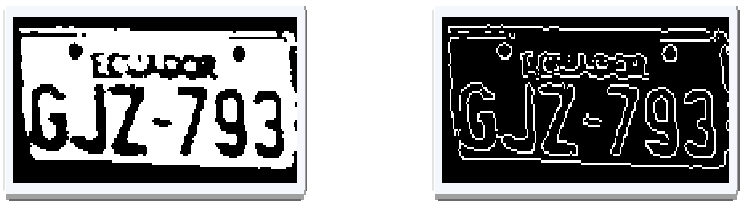


Fig. 3.4 Detección de bordes usando Canny

Detección de líneas mediante las transformadas de Hough

Se utilizó la transformada de Hough con la finalidad de encontrar las líneas presentes en la imagen, y se ingresó parámetros de tal manera que estas líneas sean paralelas a las líneas del borde superior e inferior de la placa.

Para ello se envió los siguientes parámetros:

```
cvHoughLines2(image,lineStorage,method,rho,theta,threshold,param1
, param2);
```

image.- Imagen de entrada, para nuestro algoritmo ingresamos la imagen con bordes detectados que se obtuvo en el proceso anterior.

lineStorage.- Es una variable de tipo cvMenStorage en la cual se almacenaran las líneas obtenidas.

method.- Se utilizó el método CV_HOUGH_PROBABILISTIC el cual es más eficiente en el caso de que la imagen contenga pocos segmentos lineales y devuelve los segmentos lineales.

rho.- Distancia de resolucion entre pixels, para nuestro algoritmo se determino un valor constante de 1.

theta.- Ánglo máximo de inclinación en Radianes, para nuestro algoritmo se determino un valor constante de 40*CV_PI/180.

threshold .- Umbral de los parámetros. Una línea es devuelta por la función correspondiente, si el valor del acumulador es mayor que el umbral, para nuestro algoritmo se determino un valor constante de 40.

param1.- En el caso del método probabilístico de Hough, es el mínimo largo de la línea, para nuestro algoritmo se determino un valor constante de 100.

param2.- En el caso del método probabilístico de Hough, es la distancia máxima que deben existir entre dos líneas que están en una misma dirección se unan en un solo segmento, para nuestro algoritmo se determino un valor constante de 200.

En la figura 3.5, se muestra la imagen de entrada y resultante del proceso de aplicar la detección de líneas mediante las transformadas de Hough.

Imagen de Entrada

Imagen con Lineas de hough

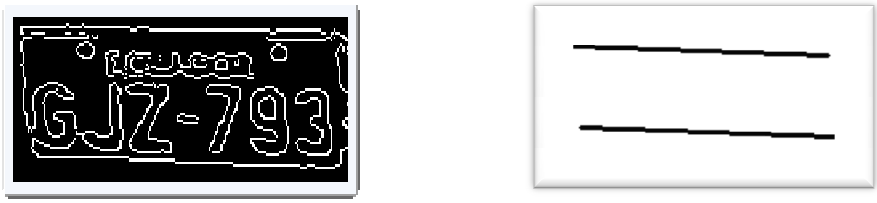


Fig. 3.5 Detección de líneas mediante las transformadas de Hough

Detección de esquinas mediante CornerHarris

Se utilizó el detector de esquinas de Harris para detector los vértices de la placa, para ello se lo aplica sobre la imagen que contiene las líneas de hough con el fin de identificar sus esquinas.

```
cvCornerHarris(image, harrisResponce, blockSize, apertureSize,
k);
```

image.- Imagen de Entrada, para nuestro algoritmo ingresamos la imagen con líneas de Hough detectadas.

harrisResponce.- Imagen resultante, que almacena las esquinas detectadas.

***blockSize*.-** Tamaño de sus vecinos, para nuestro algoritmo se determino un valor constante de 5.

***apertureSize*.-** Parámetro de la apertura del operador Sobel, para nuestro algoritmo se determino un valor constante de 3.

k.- Parametro de detector Harris, para nuestro algoritmo se determino un valor constante de 0.04.

A continuación se muestran las imágenes resultantes del proceso.

En la figura 4.6, se muestra la imagen de entrada y resultante del proceso de aplicar la detección de esquinas mediante CornerHarris.

Imagen de Entrada Imagen con esquinas de Harris detectadas

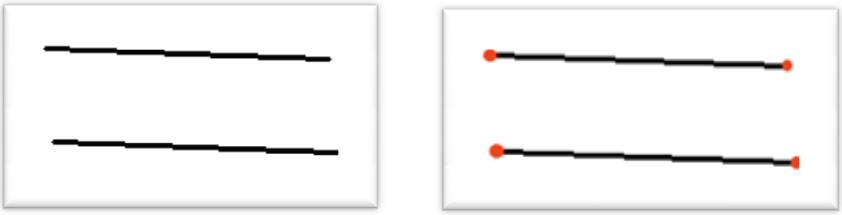


Fig. 3.6 Detección de esquinas mediante CornerHarris

Proyectar a una imagen base mediante WarpPerspective

Una vez seleccionada las coordenadas de las esquina de Harris se toma los puntos detectados en la imagen de original y se la proyecta en el tamaño total de la ventana. La función WarpPerspective acepta una imagen, un cuadrilátero, una matriz de transformación, y retorna una nueva imagen del área cubierta por el cuadrilátero sobre la que se ha aplicado la matriz de transformación para eliminar la deformación debida a la perspectiva.

Para implmentar el WarpPerspective en OpenCV utilizamos las siguientes funciones:

```
mmat = cvGetPerspectiveTransform(input, output, mmat);
```

Input.- Coordenadas de los 4 vertices del cuadrilátero correspondiente a la imagen de origen.

Output.- Coordenadas de los 4 vertices del cuadrilátero correspondiente a la imagen de salida.

Mmat.- Matriz de transformación.

```
cvWarpPerspective(src, dst, cvMat,flags,fillval);
```

src.- Imagen de entrada, se envía la imagen original que fue usada para la detección de bordes y esquinas en los procesos anteriores.

dst.- Imagen de destino, la cual contendrá la imagen con la proyección es decir con la corrección geométrica.

CvMat.- Matriz de transformación, para nuestro algoritmo se utilizó `mmat` encontrada anteriormente.

Flags.- Es una combinación de métodos de interpolación, para nuestro algoritmo se utilizó la combinación de los siguientes métodos.

CV_INTER_LINEAR.- Interpolacion bilinear usado por default

CV_WARP_FILL_OUTLIERS.- Llena todos los píxeles de la imagen de destino. Si alguno de ellos corresponden a los valores extremos de la imagen original se pone en cero.

fillVal.- Con 0 indica que el tamaño de salida es igual al tamaño de entrada y con 1 indica que el tamaño de salida puede ser diferente, para nuestro algoritmo utilizams 0 como parametro.

En la figura 3.7, se muestra la imagen de entrada y resultante del proceso de proyección de una imagen base, mediante WarpPerspective. En la primera imagen se muestra los puntos obtenidos con el detector de esquinas de Harris sobre las líneas de Hough, los cuales serán proyectados sobre los puntos de la segunda imagen.

Imagen de entrada Imagen Resultante con proyección

Fig. 3.7 Proyección de una imagen base, mediante WarpPerspective

En la figura 3.8 se muestra la misma proyección anterior pero con una imagen de placa real, y en la segunda imagen muestra el resultado de la perspectiva.

Imagen con los puntos a proyectar Placa corregida

Fig. 3.8 Proyección de una imagen base, mediante WarpPerspective con placa real.

3.2.2 Resultados

De los resultados mostrados se pudo determinar que las líneas de hough no siempre era posible encontrarlas, lo cual dificultaba la búsqueda de las esquinas de Harris.

3.2.3 Conclusiones de las pruebas

Este algoritmo es muy inestable y es muy susceptible a sombras lo cual provoca que las líneas hough no siempre cubran el borde superior e inferior por completo. Sin embargo nos damos cuenta que la debilidad del algoritmo está en las líneas de Hough que es la parte clave, por lo tanto se tendría que buscar una mejora en esa área.

Sin Embargo hay funciones como Canny, cvWarpPerspective y cvCornerHarris que podrían ser muy útiles.

3.3 Algoritmo Inicial 2: Detección de silueta de la placa mediante detección de Blobs.

En esta segunda etapa se busca corregir geométricamente la silueta de la placa, mejorando el algoritmo de la etapa uno para ello se utilizó la detección de Blobs en lugar de las líneas de Hough, lo cual le da una mejor aproximación a la detección de la silueta de la placa.

Para esto se propuso el siguiente procedimiento para la implementación de este segundo algoritmo.

- Adquisición de la imagen
- Preprocesamiento
  - Ecualización de histograma
  - Binarización de imagen
- Detección de bordes
- Detección Blobs
- Detección de esquinas mediante CornerHarris
- Proyectar a una imagen base mediante WarpPerspective

3.3.1 Implementación

Adquisición de la imagen

Al igual que en la primera etapa se utiliza la función cvLoadImage que carga la imagen de la ruta especificada y nos devuelve un puntero a la imagen.

El número de bits utilizados para la digitalización es de ocho, por lo tanto se utiliza 8U, con un solo canal, con lo cual obtenemos una imagen binaria. Se utilizan estos valores para tener un volumen mínimo de información y así resolver de forma óptima los siguientes procesamientos.

Preprocesamiento

Al igual que en la primera etapa la imagen tomada en el proceso de adquisición se necesita de procesamientos previos para poder realzar la características de los bordes y así poder detectarlos de una mejor manera.

Ecualización de histograma

Se utilizó la ecualización al igual que en la primera etapa para mostrar un mayor realce en los contrastes en la imagen lo cual nos ayudara a detectar de una mejor manera los bordes de la placa. Dando como resultado las siguiente imágenes que se muestran en la figura 3.9.

Imagen de entrada Imagen ecualizada

Fig. 3.9 Ecualización de histograma

Binarización de Imagen

Se realiza una serie de binarizaciones con un umbral de binarización de cinco a cien sobre la imagen ecualizada con el fin de encontrar la silueta

de la placa e ir probando en cada uno de los niveles los métodos siguientes para la detección de los bordes de la placa.

Imagen de Entrada Imagen Binarizada

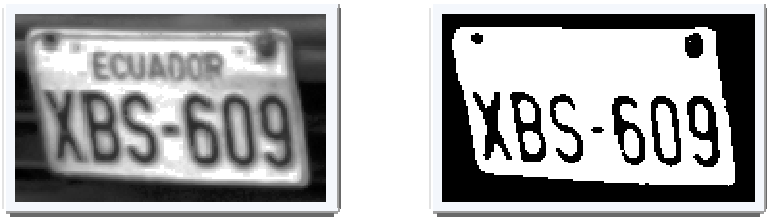


Fig. 3.10 Binarización de imagen

Detección de bordes

Al igual que en la primera etapa se utiliza Canny para detectar los bordes de la imagen, en nuestro caso para detectar la los bordes de la placa, aunque también incluye los bordes de las letras.

Imagen de Entrada Imagen Binarizada

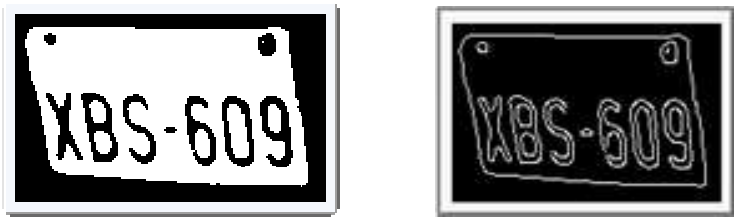


Fig. 3.10.1 Detección de bordes

Detección Blobs

Luego de haber encontrado los bordes con el filtro de Canny, se procede a detectar bloques o cajas en la imagen a esto se le denomina blobs. Para encontrar los blobs se utilizaron las siguientes funciones:

La función `CBlobResult,` encuentra todos los blobs en la imagen de entrada.

```
blobs = CBlobResult(source, mask, backgroundColor );
```

Source.- Imagen de entrada, se envía la imagen con bordes detectados en el proceso de detección de bordes.

Mask.- Mascara a aplicar. Para nuestro algoritmo no aplicamos mascara por lo tanto en este parámetro toma el valor de null.

backgroundColor.- Define el color de fondo del blob detectado, para nuestro algoritmo ingresamos 255.

La funcion filter excluye los blobs mas pequeños que el valor enviado por parametro.

```
blobs.Filter( blobs, B_EXCLUDE, CBlobGetArea(), B_LESS, param2 );
```

Luego se usan las siguientes funciones para mostrar los blobs filtrados.

```
cvMerge(originalThr, originalThr, originalThr, NULL,
displayedImage );


      for (i = 0; i < blobs.GetNumBlobs(); i++ )

      {

            currentBlob = blobs.GetBlob(i);

            currentBlob->FillBlob( displayedImage,
CV_RGB(255,0,0));

      }
```

En la figura 3.11 se muestra las imágenes de entrada con los bordes detectados previamente en el proceso de detección de bordes y la imagen resultante del proceso de detección de blobs.

Imagen de entrada Imagen con blob detectado

Fig. 3.11 Detección de Blobs sobre imagen con bordes detectados

Detección de esquinas mediante CornerHarris

Una vez detectados los blobs se procede a encontrar las esquinas mediante la función `cvCornerHarris` para ello se usa la función con los siguiente parámetros.

```
cvCornerHarris(image, imgharris, blockSize, apertureSize, k);
```

***image*.-** Imagen de entrada, imagen con blob detectado en proceso anterior.

***harrisResponce*.-** Imagen de salida, donde se almacenara el resultado es decir las esquinas detectadas.

***blockSize*.-** Valor entero que representa el tamaño del vecindario, para nuestro algoritmo se determino un valor constante de 5..

***apertureSize*.-** Apertura del operador Sobel, para nuestro algoritmo se determino un valor constante de 3..

***k* .-** Parametro de detección de Harris, para nuestro algoritmo se determino un valor constante de 0.04.

En la figura 3.12 contniación se muestra las imágenes de entrada y resultante del proceso de detección de esquinas mediante CornerHarris.

Imagen de entrada Imagen con esquinas detectadas

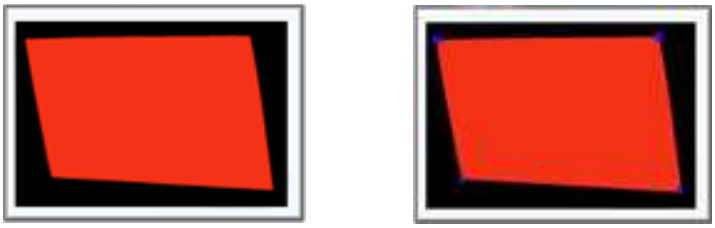


Fig. 3.12. Detección de esquinas

Proyectar a una imagen base mediante WarpPerspective

Al igual que en el primer algoritmo, en este presente algoritmo también se utiliza WarpPerspective para proyectar los puntos encontrados a unos nuevos puntos sobre una nueva imagen con el fin de corregir geométricamente la posición de la placa.

La figura 3.13 muestra la imagen de entrada del algoritmo y su imagen resultante que viene a ser la imagen de la placa corregida geométricamente.

Imagen de entrada Imagen proyectada

Fig. 3.13. Detección de esquinas

3.3.2 Resultados

De las pruebas realizadas se determino que el detector de esquinas de Harris no funcionaba correctamente en todas las imágenes lo cual le dio una debilidad notoria al algoritmo. Puesto que los blobs detectados no se ajustaban a la silueta de la placa, ya que este algoritmo es muy susceptible a los ruidos o sombras en las placas, aunque en imágenes con una buena definición tiene un aceptable resultado.

3.3.3 Conclusiones de las pruebas

El haber utilizado blobs le dio una mayor posibilidad de detectar la silueta de la placa, pero al momento de buscar las esquinas con Harris el algoritmo muestra problemas, tal como no detectar correctamente las esquinas, por lo tanto se tendría que buscar una mejor manera de buscar las esquinas, o buscar otra manera de detectar la silueta de la placa.

3.4 Algoritmo Inicial 3: Detección de bordes mediante aproximación de polígonos.

En esta tercera etapa se busca corregir geométricamente la silueta de la placa, reutilizando funciones implementadas en etapas anteriores tales como Canny, cvWarpPerspective.

En esta etapa se presenta una solución que consiste en aplicar un filtro para el ruido gaussiano, para luego aplicar Canny, niveles de binarización, y con esto encontrar contornos para luego por medio de varias heurísticas seleccionar un polígono que se aproxime de mejor manera a la silueta de la placa, una vez encontrado este polígono se procede a encontrar los puntos de las esquinas para proyectarlo a una imagen con una posición correcta, de esta manera se realiza una corrección geométrica de la placa.

La solución planteada en la presente etapa consta de los siguientes pasos:

- Adquisición de la imagen
- Filtro de ruido Gaussiano
- Detección de bordes
- Binarización
- Encontrar contornos
- Aproximar de polígonos
- Calcular área de polígonos
- Aplicar heurísticas para seleccionar polígono
- Seleccionar contorno

- Proyectar imágenes

3.4.1 Implementación

Adquisición de la imagen

Al igual que en las etapas anteriores se utiliza la función cvLoadImage que carga la imagen de la ruta especificada y nos devuelve un puntero a la imagen.

El número de bits utilizados para la digitalización es de ocho, por lo tanto se utiliza 8U, con un solo canal, con lo cual obtenemos una imagen binaria. Se utilizan estos valores para tener un volumen mínimo de información y así resolver de forma óptima los siguientes procesamientos.

Filtro de ruido Gaussiano

El ruido Gaussiano son valores de nivel de gris que se encuentran distribuidos a lo largo de una campana gaussiana, para ello se utilizó las siguientes funciones `cvPyrDown` y `cvPyrUp`.

Estas funciones convolucionan la imagen original con el filtro especificado y luego aumenta o disminuye la resolución de la imagen.

```
cvPyrDown(src, dst, filter);
cvPyrUp(src, dst, filter);
```

Los parámetros para `cvPyrDown` son los siguientes:

src .- Imagen de entrada, para nuestro algoritmo es la imagen de entrada.

dst .- Imagen de salida la cual es la mitad del ancho y alto de la imagen de entrada.

filter.- Tipo de filtro utilizado para la convolución, únicamente el CV_GAUSSIAN_5x5 es actualmente soportado.

Los parámetros para `cvPyrUp` son los siguientes:

src .- Imagen de entrada, para nuestro algoritmo se utiliza la imagen resultante de aplicar cvPyrDown.

dst .- Imagen de salida la cual es el doble del ancho y alto de la imagen de entrada.

filter.- Tipo de filtro utilizado para la convolución, únicamente el CV_GAUSSIAN_5x5 es actualmente soportado.

La figura 3.14 muestra la imagen de entrada del algoritmo y su imagen resultante que viene a ser la imagen de la placa aplicando el filtro de ruido Gaussiano.

Imagen de Entrada Imagen Filtro Gaussiano

Fig. 3.14 Filtro de ruido Gaussiano

Detección de bordes

Al igual que en los métodos anteriores se utiliza Canny para detectar los bordes de la imagen, para luego poder detectar los contornos. La imagen resultante de este proceso se muestra en la figura 4.15 siguiente.

Imagen de Entrada Imagen Filtro de Canny

Fig. 3.15 Filtro de Canny

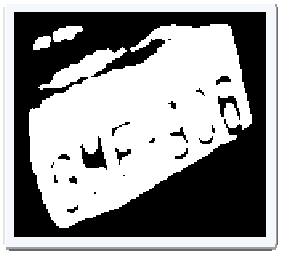
Se aplica canny para luego encontrar los contornos y los polígonos aplicando las ciertas heurísticas, y se almacena los contornos obtenidos, luego se busca mas contornos binarizando la imagen.

Binarización de Imagen

Se realiza binarizaciones con valores de umbral variante de 0 a 50 sobre la imagen filtrada con de ruido Gaussiano con el fin de encontrar la silueta de la placa e ir probando en cada uno de los niveles los métodos siguientes para la detección de los bordes de la placa.

En la figura 3.16 se muestra una placa a varios niveles de binarización.

Imagen de entrada Binarizada a nivel 10 Binarizada a nivel 16



Binarizada a nivel 18 Binarizada a nivel 20

Fig. 3.16 Imagen de placa con varios niveles de binarización

Encontrar Contornos

La función cvFindContours se utiliza para recuperar los contornos de una imagen. Mediante esta función se detecta los contornos en la imagen los cuales en lo posterior se recorren uno a uno para detectar polígonos que se aproximen a la silueta de la placa.

Los parámetros enviados a la función son los siguientes:

```
cvFindContours( image, storage, firstContour, headerSize, mode,
method, offset );
```

image.- Imagen de entrada, para nuestro algoritmo seria la imagen binarizada obtenida en el proceso anterior.

storage.- Variable de tipo cvMenStorage el cual almacena los contornos encontrados

firstContour.- Parametro de salida el cual contiene el puntero al primer contorno de salida.

headerSize.- El tamaño de la secuencia, para nuestro algoritmo se utilizó sizeof(CvContour).

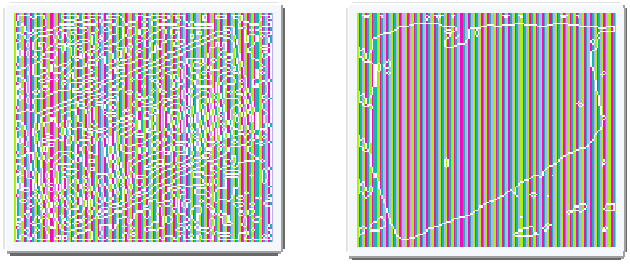
mode.- Modo de Recuperación, para nuestro algoritmo se utilizó CV_RETR_LIST.

method.- Método de aproximación, para nuestro algoritmo se utilizó CV_CHAIN_APPROX_SIMPLE.

offset.- Compensación, para nuestro algoritmo se utilizó cvPoint(0,0).

Las imágenes resultantes de este proceso son las siguientes, las imágenes de entrada son las imágenes de Canny y binarizadas previamente en los pasos anteriores.

Contornos imagen Canny Contornos Binarizada a nivel 10



Contornos Binarizada a nivel 16 Contornos Binarizada a nivel 20

Fig. 3.17 Imagen de placa en la cual se detectaron contornos

Aproximación de polígonos

Cada contorno encontrado anteriormente es aproximado usando polígonos aproximantes, para esto usamos el algoritmo de Ramer

Douglas que consiste en reducir el número de punto en una curva aproximada por una serie de puntos. Para poder realizar esto su utilizó la siguiente función:

```
cvApproxPoly( srcSeq, headerSize, storage,  method, parameter, parameter2)
```

Recibe los siguientes parámetros:

srcSeq.- Secuencia de puntos que representan el contorno.

headerSize.- Define el tamaño de almacenamiento utilizado por el contorno para nuestro algoritmo se uso `sizeof(CvContour)`.

Storage.- Contenedor para los contornos aproximados

Method.- Metodo de aproximación, solo `CV_POLY_APPROX_DP` es soportado.

Approximation method; only *CV_POLY_APPROX_DP* is supported.

Parameter.- Es la precisión de la aproximación deseada, para nuestro algoritmo se utilizó el siguiente valor `cvContourPerimeter(contours)*0.02`.

Parameter2.- Se envía el parámetro cero para identificar que se trata de una curva cerrada, para nuestro algoritmo se utilizó el valor constante de 0.

En la figura 3.18 se muestra las imágenes de las aproximaciones de polígonos.

Imagen de entrada Imágenes con aproximación de poligonos

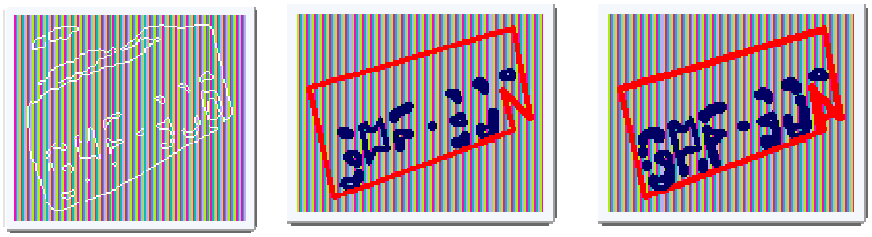


Fig. 3.18 Imagen de placa en la cual se detectaron contornos

Calcular Área de polígonos

A continuación se calcula el área de cada polígono resultante que aproximan a los contornos previamente encontrados, esta área nos servirá para en un paso posterior poder comparar y determinar si cumple con el tamaño mínimo para ser considerado como silueta de la placa.

Para encontrar esta área se usa la siguiente función:

```
cvContourArea(result,CV_WHOLE_SEQ)
```

La cual recibe una secuencia de vértices que definen al polígono cerrado, en nuestro caso los contornos encontrados anteriormente, y el segundo parámetro que indica que queremos el área de todo el contorno.

Aplicar Heurísticas para seleccionar polígono

Una vez detectado los contornos y luego de aproximarlos a sus respectivos poligonos aproximantes con la menor cantidad de puntos posibles procedemos a aplicar las siguientes heurísticas para seleccionar los mejores candidatos a contener la silueta de la placa.

Se aplican las siguientes heurísticas:

- Que el porcentaje de relación entre el área de la silueta de placa y de la caja englobante sea mayor que el porcentaje de área mínima que fue definido anteriormente en 35.
- Que el polígono aproximado tenga cuatro vertices
- Que sea un poligono convexo.
- El porcentaje de rotación tenga un máximo de 70 grados de rotación.

Si existen más de un polígono que hayan cumplido con estas heurísticas serán almacenados en una lista para un posterior análisis, en el cual se

seleccionará uno el cual será el que más se aproxime a la silueta de la placa.

Seleccionar contorno

Una vez aplicadas las heurísticas y haber seleccionados los candidatos, se procede a seleccionar uno de ellos, cabe recalcar que en promedio son seleccionados tres candidatos por cada imagen y que estos polígonos tienen buenas características, ya que se ajustan a la silueta de la placa.

En la figura 3.19 se muestran imágenes con tres polígonos encontrados para esta placa, cada polígono ha sido dibujado por separado para poder analizar sus características.

Fig. 3.19 Polígonos detectados, graficados por separado

En la figura 3.20 se muestra la siguiente imagen con todos los polígonos dibujados a la vez.

Fig. 3.20 Polígonos detectados, graficados sobre una imagen

De las múltiples pruebas realizadas se detecto que el polígono que contiene menor área es el que más se ajusta a la silueta de la placa por lo tanto de este ejemplo la imagen seleccionada sería la que se muestra en la figura 3.21.

Fig. 3.21 Polígonos seleccionado, el que contiene menor área.

Proyectar Imágenes

Al igual que en la primera etapa, realiza la corrección geométrica usando la función cvWarpPerspective, la cual me muestra el resultado como se muestran en las siguientes imágenes de la figura 3.22.

Imagen con polígono seleccionado Imagen con corrección

Fig. 3.22 proyección de imagen.

3.4.2 Análisis de Resultados

De lo mostrado en las pruebas se detectó varios problemas, los cuales se detallan a continuación:

Existen proyecciones que al momento de mostrarlas salen con una mala orientación es decir las placas se muestran giradas verticalmente.

Por otro lado en la mayoría de las pruebas las imágenes salen cortadas en los bordes, lo cual provoca perdida de información de la placa.

Al proyectar la imagen el tamaño se distorsiona ya que se la proyecta sobre el tamaño de la imagen de entrada.

3.4.3 Conclusiones de las pruebas

Como conclusión podemos decir que el algoritmo es bastante estable aunque tiene ciertas fallas que fueron determinadas en análisis de

resultados, por lo tanto se debe buscar soluciones a esas fallas para tener un algoritmo con mejores resultados.

3.5 Algoritmo Final: Detección de bordes mediante aproximación de polígonos.

En esta etapa se muestran las correcciones implementadas a los algoritmos anteriores.

Para esto, se utilizó el siguiente procedimiento.

- Encontrar valores de parámetros constantes
  - Mayor ángulo de rotación, en los planos XY, XZ, YZ
  - Rango de binarización
  - Porcentaje de relación entre el área de la placa y la caja englobante.
- Resolver problemas detectados:
  - Problema de orientación.
  - Problemas de bordes cortados.
  - Problema de proyección de los puntos detectados.

3.5.1 Implementación

A continuación se describen cada uno de las implementaciones realizadas para la detección del algoritmo final.

Encontrar valores de parámetros constantes

Los parámetros constantes que necesitamos para la implementación del proyecto son hallados tal como sigue.

Mayor ángulo de rotación en los planos XY, XZ y YZ

En esta etapa el objetivo es determinar cuál es el máximo ángulo aceptado para la rotación en cada uno de los planos, esto es para los planos XY, XZ y YZ.

Rotación en el plano XY

Para las rotaciones en el plano XY cuyas rotaciones se muestran a continuación, se ha determinado un máximo ángulo de rotación aceptado de 70^0. De esta forma, se asume que las placas a ser analizadas no deben tener un ángulo de rotación mayor de 70^0.

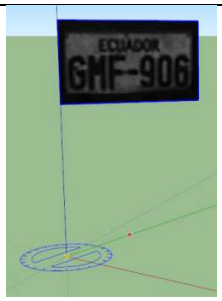
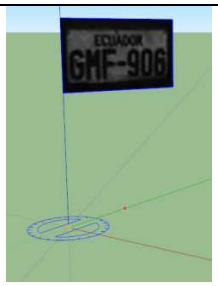
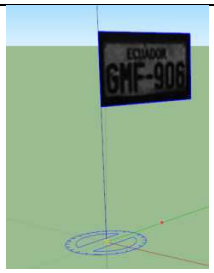
Rotación 0^0	Rotación 15^0	Rotación 30^0
		

Rotación 45^0	Rotación 60^0	Rotación 70^0
		

Fig. 3.23 rotaciones de la imagen original

Rotación en el plano XZ

Para las rotaciones en el plano XZ, cuyas rotaciones se muestran a continuación, se ha determinado un ángulo máximo de rotación de 60^0. Al igual que el proceso de rotación XY, se asume que si el ángulo de rotación de la placa sobrepasa este ángulo se hace muy complicado el corregir geométricamente la orientación de la placa.

	**Original** 
	**Rotada 15 Grados** 
	**Rotada 30 Grados** 
	**Rotada 45 Grados** 

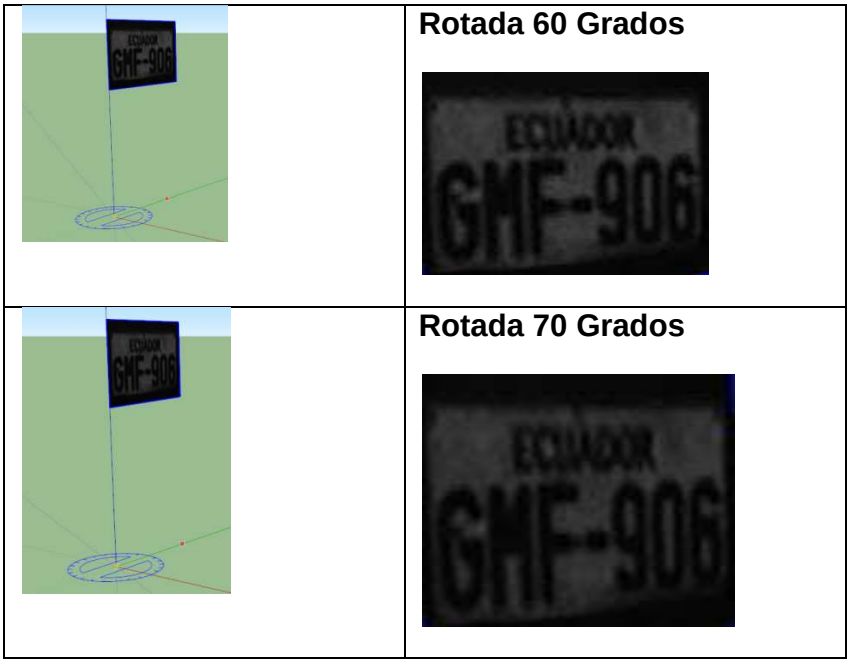
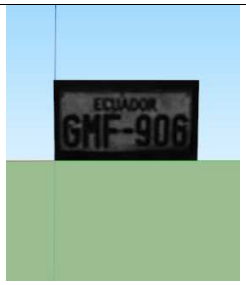
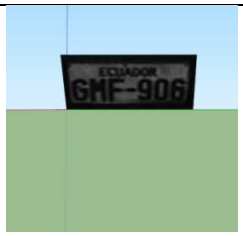
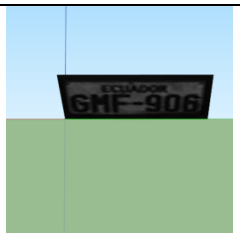


Fig. 3.24 Rotaciones en el plano XZ de placa a diferentes ángulos.

Rotación en el plano YZ

Para las rotaciones en el plano YZ cuyas rotaciones se muestran a continuación, se ha determinado un ángulo máximo de rotación de 60^0. Para ángulos mayores a este ángulo se hace muy complicado el corregir geométricamente la orientación de las placas.

	**Original** 
	**Rotada 15 Grados** 
	**Rotada 30 Grados** 
	**Rotada 45 Grados** 
	**Rotada 60 Grados** 

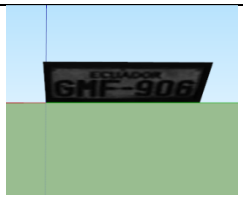
	**Rotada 70 Grados** 

Fig. 3.25 Rotaciones en el plano YZ de placa a diferentes ángulos.

Rango de binarización

En las etapas anteriores se trabajaba con un rango de binarización de 5 a 60, sin embargo esto provocaba que sea procesada una mayor cantidad de veces, Este inconveniente fue posteriormente analizado en el tercer capítulo y se determinó que el rango de binarización más adecuado sea de 10 a 24.

Porcentaje de relación entre el área de la placa y la caja englobante.

Este porcentaje nos ayuda a encontrar un porcentaje mínimo de relación que tendría un polígono dentro de una imagen para ser considerado como silueta de la placa. Para ello se rotó la imagen de la placa a varios

ángulos y de esta forma observó como varía el porcentaje de relación entre el área de la silueta de la placa y el área de la caja englobante.

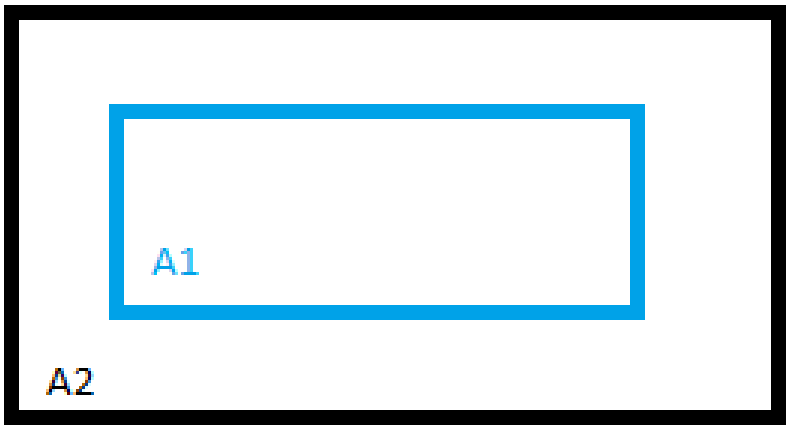


Fig. 3.25. Representación de la silueta de la placa y de la caja englobante.

Donde A1 represente el área de la silueta de la placa y A2 el área de la caja englobante.

La siguiente tabla muestra como varían, los porcentajes de relación entre la caja que encierra la silueta de la placa y la caja englobante de toda la imgen en función del ángulo de rotación.

Angulo (β)	Ancho(Pixel)	Alto(Pixel)	Porcentaje %
0	181	80	
5	195	102	72,80
10	200	117	61,88
15	204	132	53,77
20	206	145	48,48
25	207	157	44,56
30	206	169	41,59
35	203	178	40,07
40	199	187	38,91
45	194	194	38,47
50	187	199	38,91
55	178	203	40,07
60	169	206	41,59
65	157	207	44,56
70	145	206	48,48

Tabla 3.1 Muestra el porcentaje de variación en función del ángulo

En la figura 3.25.1 se muestra la variación tanto del ancho como del alto en función del ángulo de rotación, el cual fue descrito en la tabla 3.1

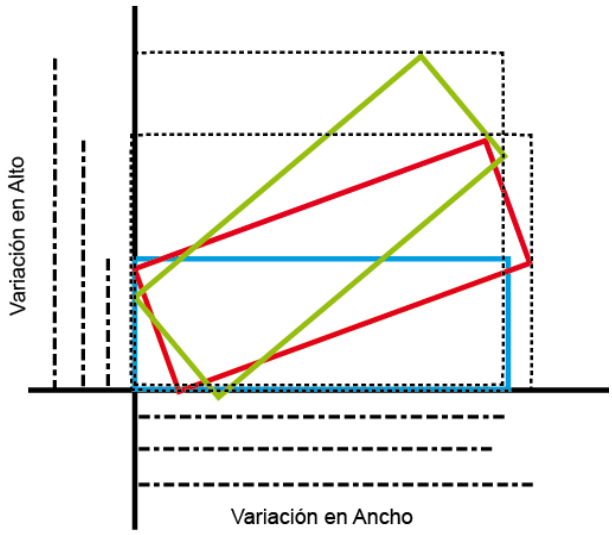


Fig. 3.25.1 Gráfica de la variación del ancho en función del ángulo

De esta tabla podemos sacar como conclusión que el menor porcentaje de relación se da a los 45 grados, ya que existe una mayor cantidad de área en la caja englobante y este valor es de 38.47%, pero para el sistema se lo redondeó a 35%.

Esto significa que al momento de recorrer los polígonos encontrados en la imagen se preguntará si tiene un porcentaje de relación de área mayor a 35% para ser aceptado caso contrario no será aceptado como posible silueta de la placa.

Resolver problemas detectados

A continuación se busca corregir los problemas detectados en la etapa tres.

Problema de orientación

Uno de los problemas detectados es el de la orientación, tal como nos podemos dar cuenta en la siguiente imagen, este problema se está dando al momento de proyectar los puntos encontrados en la imagen resultante.

Placa con orientación correcta Placa con orientación incorrecta

Fig. 3.26 Placa con orientación errónea.

Para buscar una solución al problema se graficó los puntos para tener claro la manera como se estaba graficando.

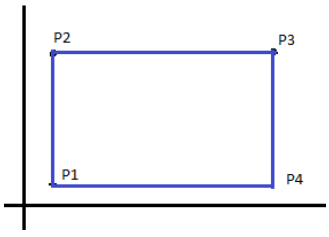


Fig. 3.27 Imagen con orientación incorrecta

En esta imagen vemos que muestra una orientación diferente a las otras imágenes que muestran la siguiente orientación.

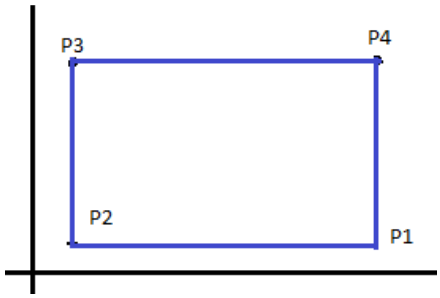


Fig. 3.28 Imagen con orientación correcta

Se intercambiaron los puntos P1 con P4, P2 con P1, P3 con P2, P4 con P3.

Al cambiarle la orientación, se corrigió el problema dando como resultado una imagen correctamente orientada como se muestra a continuación.

Fig. 3.29 Imagen con orientación correcta

Problemas de bordes cortados

El problema consiste en que al detectar el borde que más se ajusta a la silueta de la placa, es muy probable que se produzca un recorte en el borde de la placa, como se muestra en la figura 3.30.

Para resolver este problema se aumento un espacio constante tanto en el alto como el ancho con lo cual se obtuvo un mejor resultado.

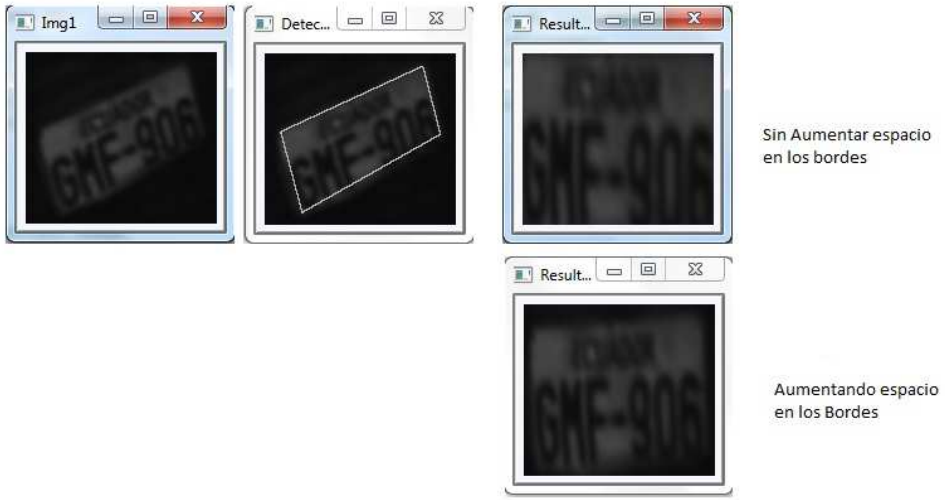


Fig. 3.30 Imagen con problema de bordes recortados

Problema de proyección de los puntos detectados

El problema consiste en que al momento de proyectar los cuatro puntos encontrados de los vértices de la silueta de la placa se los proyecta al

tamaño de la imagen de entrada, por lo tanto esto causa una distorsión en su ancho y alto original.

Imagen de entrada Silueta detectada Imagen proyectada

Fig. 3.31 Imagen que muestra el problema de proyección

Previamente se debe tener en cuenta que hay dos tipos de placas las de seis y las de 7 dígitos.

Las placas tienen los siguientes tamaños.

Placa de seis dígitos		Placa de siete dígitos	
Ancho(cm)	Alto(cm)	Ancho(cm)	Alto(cm)
30	15.5	40	15.5

Tabla 4.2 Ancho y alto de las placas

Pero no hay forma de identificar la cantidad de dígitos que hay en la imagen de placa que se va a corregir, por lo tanto se definió un tamaño promedio de la placa cuyo tamaño se muestra a continuación.

Placa Promedio	
Ancho(cm)	Alto(cm)
35	15.5

Tabla 4.3Placa Promedio

Nos damos cuenta que solo varia el ancho ya que el alto es el mismo para ambas placas, en las imágenes de seis dígitos no existe ningún problema ya que los caracteres se separan un poco lo cual ayuda al reconocimiento de caracteres OCR, y para las placas de siete dígitos tampoco existe problemas ya que tienen suficiente espacio entre los caracteres de placa por lo cual al reducir el ancho no causa ningún problema para los siguiente procesos.

Solución

Para proyectar los puntos detectados de la silueta de la placa se debe tener en cuenta que la imagen original posiblemente tuvo rotaciones en tres planos el xy, yz y/o xz.

Por lo tanto determinaremos el tamaño al cual será proyectada corrigiendo la orientación en cada uno de estos planos.

Rotación en el plano xy

Dada una placa correctamente orientada, la cual tiene 140 x 80 de ancho x alto en pixeles, vamos a rotarla a 15, 30, 45, 60 grados para analizar como varia el tamaño de la caja englobante.

Imagen de entrada

Imagen rotada 15°

Imagen rotada 30°

Imagen rotada 45°

Imagen rotada 60°

Fig. 3.32 Imagen que muestra rotaciones en el plano XY

De las imágenes mostradas podemos observar que a medida que varía el ángulo varía el tanto el ancho como el alto de la caja englobante, por lo tanto en nuestro proyecto podemos determinar cuál fue el tamaño original de la placa, conociendo cual fue el ángulo de rotación en el plano XY.

Por lo cual se procedió a encontrar una relación de variación en alto y ancho en función del ángulo de rotación, para ello se tomó una placa de muestra y se definió la tabla siguiente.

La primera columna de la tabla muestra el ángulo al cual fue rotada, la segunda fila muestra como fue variando el ancho en pixeles y la tercera el porcentaje de relación entre la variación del ancho y el tamaño original, la cuarta fila muestra la variación del alto en pixeles y la quinta muestra el porcentaje de relación entre la variación del alto y el tamaño original.

ANGULO	**ANCHO(pixel)**	**% Variacion ancho**	**ALTO(pixel)**	**% Variacion alto**
PLACA	181		80	
0	188	0	86	0
1	190	1,06	89	3,49
2	191	1,60	93	8,14
3	192	2,13	96	11,63
4	194	3,19	99	15,12
5	195	3,72	102	18,60
6	196	4,26	105	22,09
7	197	4,79	108	25,58
8	198	5,32	111	29,07

9	199	5,85	114	32,56
10	200	6,38	117	36,05
11	201	6,91	120	39,53
12	202	7,45	123	43,02
13	203	7,98	126	46,51
14	203	7,98	129	50,00
15	204	8,51	132	53,49
16	205	9,04	135	56,98
17	205	9,04	137	59,30
18	205	9,04	140	62,79
19	206	9,57	143	66,28
20	206	9,57	145	68,60
21	206	9,57	148	72,09
22	207	10,11	150	74,42
23	207	10,11	153	77,91
24	207	10,11	155	80,23
25	207	10,11	157	82,56
26	207	10,11	160	86,05
27	207	10,11	162	88,37
28	206	9,57	164	90,70
29	206	9,57	166	93,02
30	206	9,57	169	96,51
31	206	9,57	171	98,84
32	205	9,04	173	101,16
33	205	9,04	175	103,49
34	204	8,51	177	105,81
35	203	7,98	178	106,98
36	203	7,98	180	109,30
37	202	7,45	182	111,63
38	201	6,91	184	113,95
39	200	6,38	185	115,12
40	199	5,85	187	117,44
41	198	5,32	188	118,60
42	197	4,79	190	120,93
43	196	4,26	191	122,09
44	195	3,72	193	124,42

45	194	3,19	194	125,58
46	193	2,66	195	126,74
47	191	1,60	196	127,91
48	190	1,06	197	129,07
49	188	0,00	198	130,23
50	187	-0,53	199	131,40
51	185	-1,60	200	132,56
52	184	-2,13	201	133,72
53	182	-3,19	202	134,88
54	180	-4,26	203	136,05
55	178	-5,32	203	136,05
56	177	-5,85	204	137,21
57	175	-6,91	205	138,37
58	173	-7,98	205	138,37
59	171	-9,04	206	139,53
60	169	-10,11	206	139,53
61	166	-11,70	206	139,53
62	164	-12,77	206	139,53
63	162	-13,83	207	140,70
64	160	-14,89	207	140,70
65	157	-16,49	207	140,70
66	155	-17,55	207	140,70
67	153	-18,62	207	140,70
68	150	-20,21	207	140,70
69	148	-21,28	206	139,53
70	145	-22,87	206	139,53
71	143	-23,94	206	139,53
72	140	-25,53	205	138,37
73	137	-27,13	205	138,37
74	135	-28,19	205	138,37
75	132	-29,79	204	137,21
76	129	-31,38	203	136,05
77	126	-32,98	203	136,05
78	123	-34,57	202	134,88
79	120	-36,17	201	133,72
80	117	-37,77	200	132,56

81	114	-39,36	199	131,40
82	111	-40,96	198	130,23
83	108	-42,55	197	129,07
84	105	-44,15	196	127,91
85	102	-45,74	195	126,74
86	99	-47,34	194	125,58
87	96	-48,94	192	123,26
88	93	-50,53	191	122,09
89	89	-52,66	190	120,93
90	86	-54,26	188	118,60

Tabla 3.4 Tabla de variación de alto y ancho en función del ángulo

Para poder realizar un mejor análisis se procedió a graficar la tabla tanto para variación de ancho y alto en pixeles como para la variación en porcentajes.

En la figura 3.33 se muestra la variación del ancho en pixeles de la placa en función de la variación del ángulo de rotación de la placa, como se puede observar en la gráfica el ancho mantiene un periodo de variación de cero a ciento ochenta grados. Y que la curva de cero a noventa grados tiene una similitud con la curva de noventa a ciento ochenta por lo tanto, para nuestro algoritmo trabajaremos con rotaciones de de cero a noventa grados.

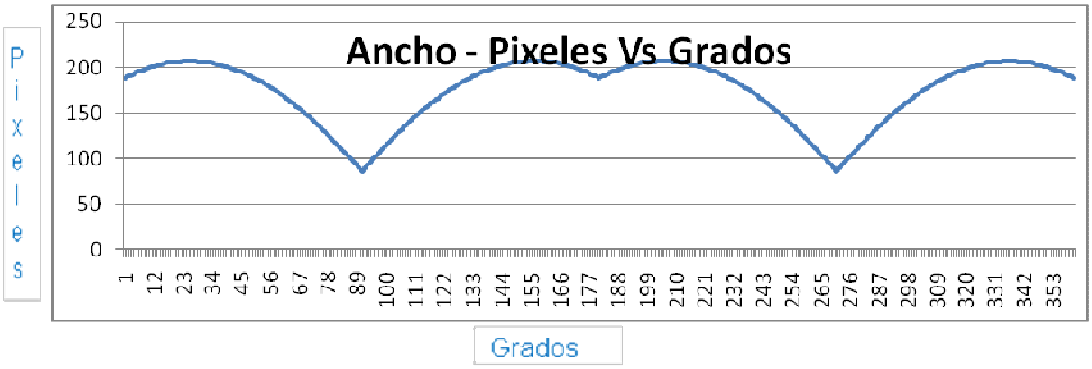


Fig. 3.33 Grafica de variación del ancho de la placa (Pixeles Vs Grados)

En la figura 3.34 se muestra la variación del ancho de la placa en función de la variación del ángulo de rotación de la placa en porcentajes de relación entre el ancho original y el nuevo ancho, como se puede observar en la gráfica el ancho mantiene un periodo de variación de cero a ciento ochenta grados. Y que la curva de cero a noventa grados tiene una similitud con la curva de noventa a ciento ochenta por lo tanto, para nuestro algoritmo trabajaremos con rotaciones de de cero a noventa grados.

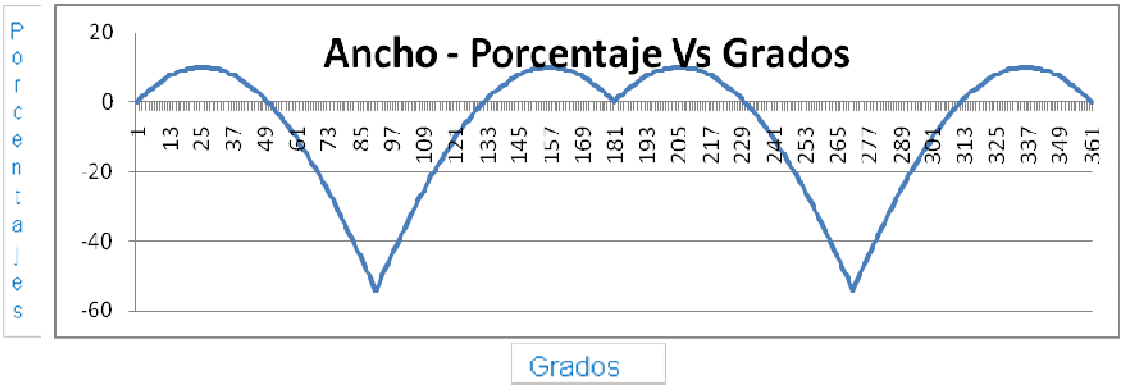


Fig. 3.34 Grafica de variación del ancho de la placa (Porcentaje Vs Grados)

En la figura 3.35 se muestra la variación del alto en pixeles de la placa en función de la variación del ángulo de rotación de la placa, como se puede observar en la gráfica el alto mantiene un periodo de variación de cero a ciento ochenta grados. Y que la curva de cero a noventa grados tiene una similitud con la curva de noventa a ciento ochenta por lo tanto, para nuestro algoritmo trabajaremos con rotaciones de de cero a noventa grados.

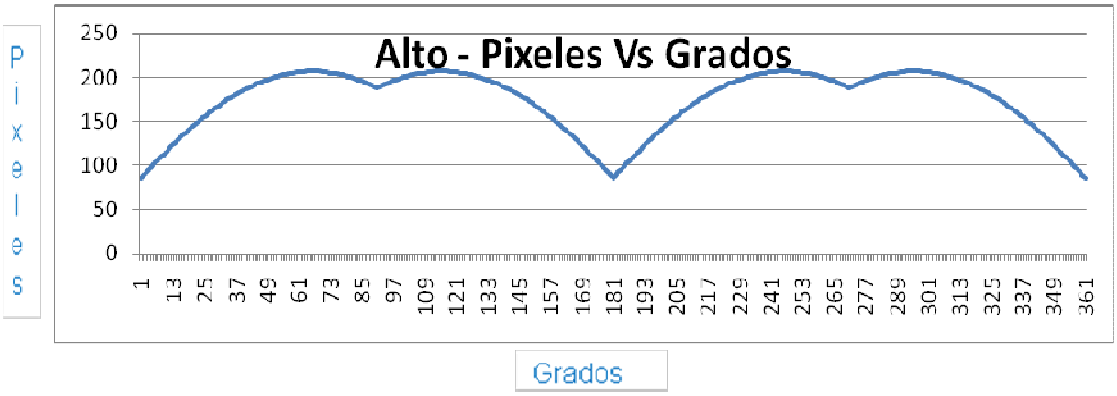


Fig. 3.35 Grafica de variación del ancho de la placa

En la figura 3.36 se muestra la variación del alto de la placa en función de la variación del ángulo de rotación de la placa en porcentajes de relación entre el ancho original y el nuevo ancho, como se puede observar en la gráfica el alto mantiene un periodo de variación de cero a ciento ochenta grados. Y que la curva de cero a noventa grados tiene una similitud con la curva de noventa a ciento ochenta por lo tanto, para nuestro algoritmo trabajaremos con rotaciones de de cero a noventa grados.

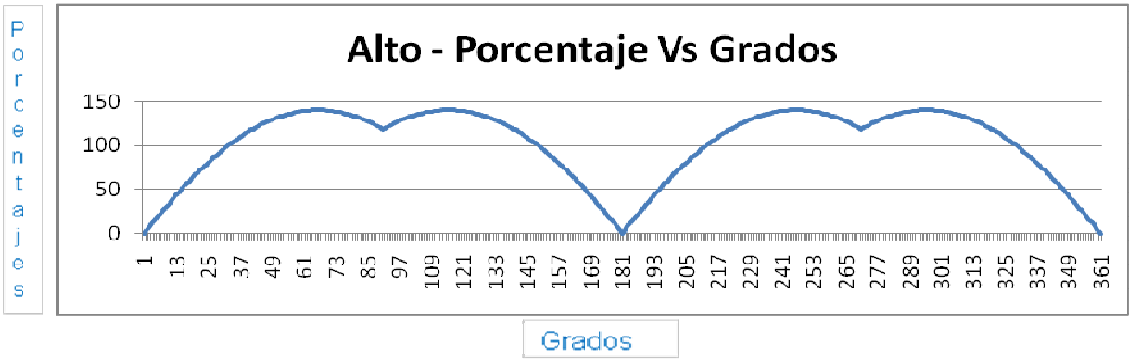


Fig. 3.36 Grafica de variación del ancho de la placa

Del análisis realizado con las gráficas anteriores y con los valores determinados en la tabla se puede decir que se tiene una imagen resultante mayormente aproximada a la imagen original de la placa.

Para ello se creó una tabla de búsqueda(Lut, look up table), con los valores de la tabla en el código, esto permitió encontrar el porcentaje de variación tanto en alto como en ancho y poder representar mejor la imagen de salida.

Rotación en el plano xz y el plano yz

Una vez corregido el problema de rotación en el plano XY, procedemos a corregir el problema de rotación en los planos XZ y YZ que podrían provocar tener distorsiones en las imágenes como se muestra a continuación.

Rotación en plano XZ Rotación en plano YZ

Fig. 3.37 Imagen que muestra rotaciones en el plano XY

Como podemos observar las imágenes cuando se rotan en el plano XZ se tiene una disminución en el ancho de la placa, y cuando se tiene una rotación en el plano YZ se tiene una disminución en el alto de la placa.

Para solucionar este problema se encontró un porcentaje de relación entre en el alto y ancho de la placa.

$$\%\, relación = \frac{Alto}{Ancho}$$

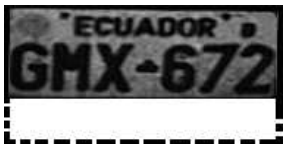
Este porcentaje de relación la calcularemos con los valores de ancho y alto obtenidos de la corrección en el plano XY que se realizó en el paso anterior, y la compararemos con el porcentaje de relación que tenemos de la placa base que es de 44.29%.

$$\%\, relación\ Base = \frac{15.5}{35} = 44.29\%$$

Al compararlos nos ayudará a detectar si existen problemas de rotación en los planos YZ y XZ.

Para solucionar la rotación en el plano YZ se realiza lo siguiente:

Si $\%\,relación$ es menor que $\%\,relación\ Base\ (44.29\%)$ entonces se aumenta el alto hasta que el $\%\,relación$ sea igual a $\%\,relación\ Base\ (44.29\%)$.



Para solucionar la rotación en el plano XZ se realiza lo siguiente:

Si $\%\,relación$ es mayor que $\%\,relación\ Base\ (44.29\%)$ entonces se aumenta el ancho hasta que el $\%\,relación$ sea igual a $\%\,relación\ Base\ (44.29\%)$.

3.5.2 Resultados

Las siguientes imágenes muestran el proceso completo desde el inicio hasta el resultado final, con las correcciones y parámetros aplicados en esta etapa, la cual mejoró notablemente el algoritmo planteado en la etapa tres.

Imagen de entrada Contorno seleccionado Contorno corregido

Imagen Resultante

Fig. 3.38 Resultados

A continuación se muestra los resultados para cada uno de los 24 puntos que se analizan en este proyecto.

Fig. 3.39 Resultados para el punto 1

Fig. 3.40 Resultados para el punto 2

Fig. 3.41 Resultados para el punto 3

Fig. 3.42 Resultados para el punto 4

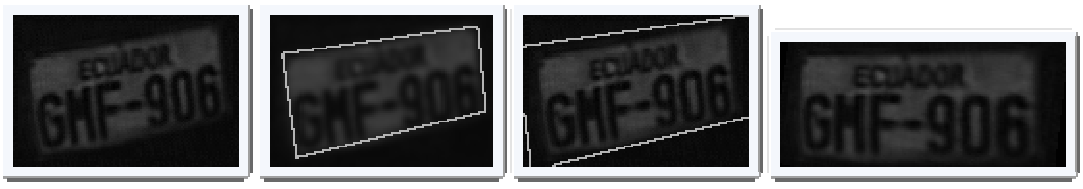


Fig. 3.43 Resultados para el punto 5

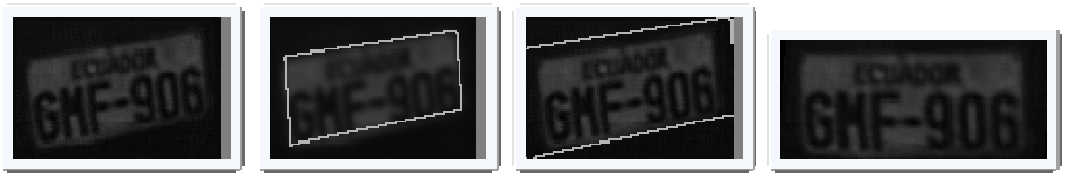


Fig. 3.44 Resultados para el punto 6

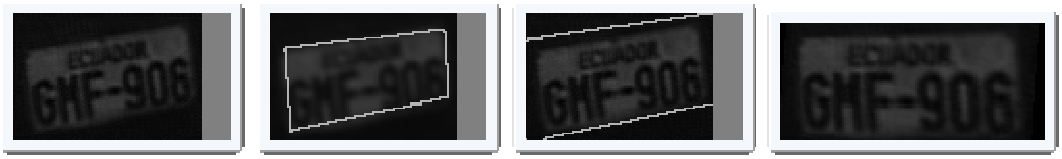


Fig. 3.45 Resultados para el punto 7

Fig. 3.46 Resultados para el punto 8

Fig. 3.47 Resultados para el punto 9

Fig. 3.48 Resultados para el punto 10

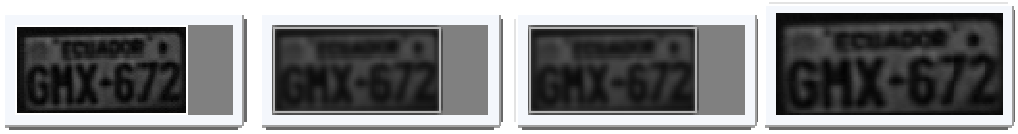


Fig. 3.49 Resultados para el punto 12

Fig. 3.50 Resultados para el punto 13

Fig. 3.51 Resultados para el punto 14

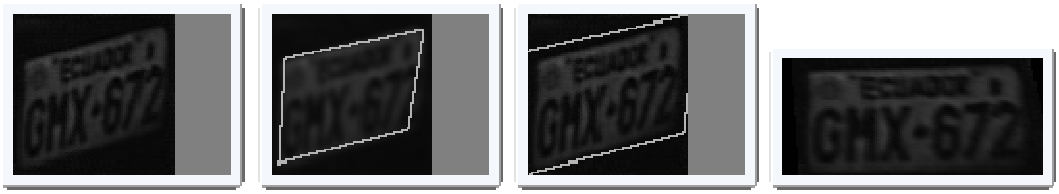


Fig. 3.52 Resultados para el punto 15

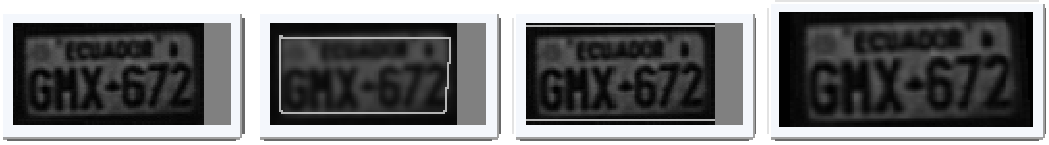


Fig. 3.53 Resultados para el punto 16

Fig. 3.54 Resultados para el punto 17

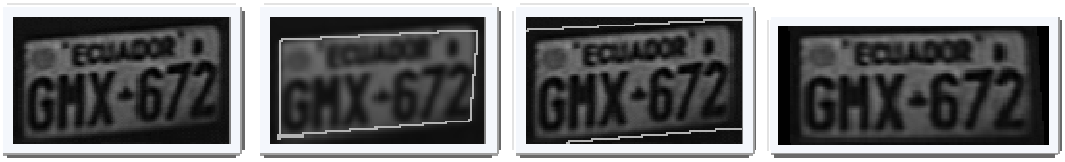


Fig. 3.55 Resultados para el punto 18

Fig. 3.56 Resultados para el punto 19

Fig. 3.57 Resultados para el punto 20

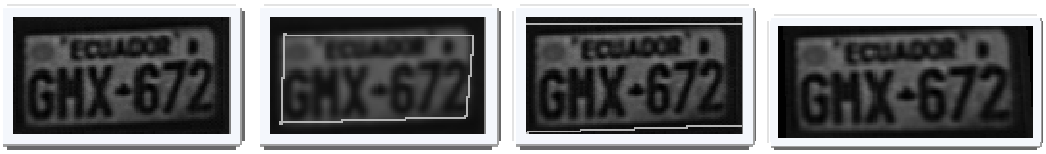


Fig. 3.58 Resultados para el punto 21

Fig. 3.59 Resultados para el punto 22

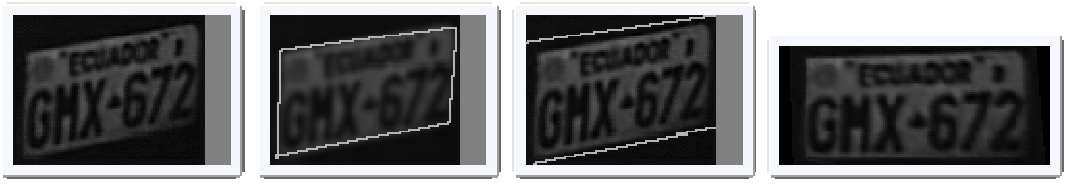


Fig. 3.60 Resultados para el punto 23

Fig. 3.61 Resultados para el punto 24

3.5.3 Conclusiones de las pruebas

En esta cuarta etapa se corrigió varios problemas que se detectaron en las tres etapas anteriores, y centrándose específicamente en el algoritmo

de la tercera etapa se corrieron los problemas que este presentaba, y al corregir los problemas de orientación, de bordes cortados y de proyección de los puntos detectados podemos concluir que ver que mejoro notablemente la corrección geométrica dándonos como resultado un algoritmo estable y con resultados óptimos.

El algoritmo final detección de bordes mediante aproximación de polígonos, pudo corregir los problemas de rotaciones en los diferentes planos XY, XZ y YZ, de una manera muy aceptable y soporta rotaciones que van de un ángulo de cero a sesenta grados en cada uno de los planos.

El algoritmo final logró obtener un mejor tamaño de la placa resultante, ya que este algoritmo analizó las características de una placa es decir su ancho y alto, con lo cual se generó un porcentaje de relación que debe ser cumplido al momento de mostrar la placa resultante.

CAPITULO V

ANÁLISIS Y RESULTADOS

4.1 Resumen

En este capítulo se muestran los resultados obtenidos de las pruebas de campo realizadas con las imágenes proporcionadas por el grupo de detección y extracción de placas, el cual obtuvo 10 imágenes para cada uno de los 24 puntos, dando un total de 240 imágenes analizadas.

4.2 Pruebas de campo

Sobre los 24 puntos señalados por el grupo de adquisición en base a parámetros establecidos por el mencionado grupo, se procedió a realizar el reconocimiento sobre las placas tomadas en cada uno de los puntos.

Los puntos mostrados de color verde en la figura encierran los puntos en los cuales se obtuvo un reconocimiento óptimo. Los puntos amarillos son aquellos cuyo porcentaje de reconocimiento es aceptable, es decir la placa fue corregida sin presentar mayores complicaciones; mientras que en los puntos rojos las correcciones geométricas no fueron buenas esto es debido a la calidad de las imágenes obtenidas en este punto.

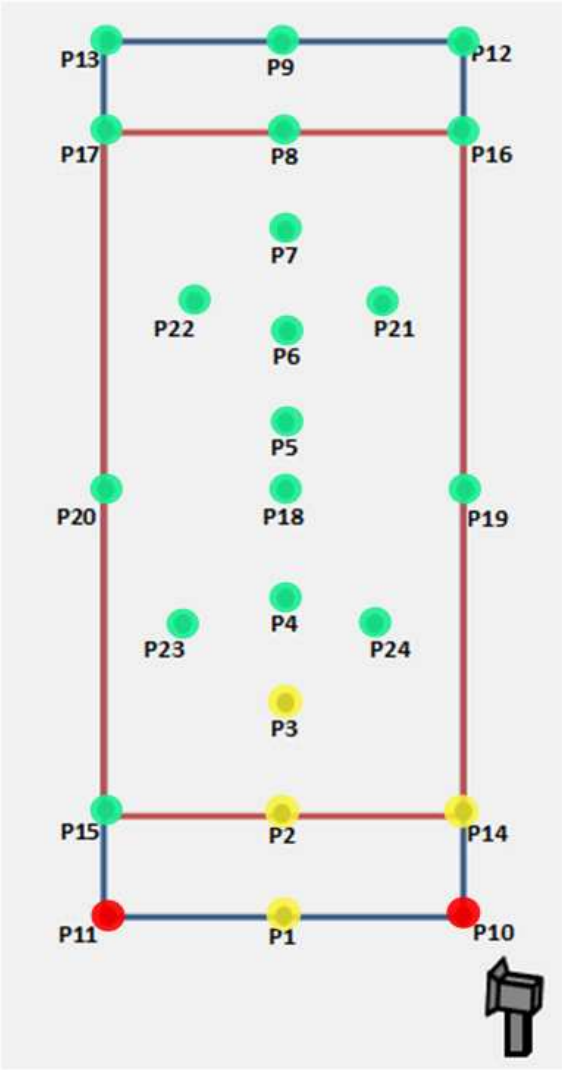


Fig. 4.1 Análisis de resultados

4.3 Resultados

Es importante mencionar que la información presentada en la siguiente tabla muestra los resultados obtenidos en las pruebas del proyecto. En la

primera columna se muestra el punto analizado, en la segunda columna se muestra el promedio de los ángulos de rotaciones de las imágenes de las placas de entrada, en la tercera columna se muestra el promedio de los ángulos de rotación de las placas sobre las imágenes resultantes y en la última columna se muestra las observaciones dadas para cada punto.

Punto	Ángulo (Entrada)	Ángulo (Salida)	Observaciones
P1	45.04	2	El problema de esta imagen se debe al nivel de rotación en los diferentes planos.
P2	36.25	1	Resultado aceptable
P3	30.65	1	Resultado aceptable
P4	26.68	1	Resultado aceptable
P5	23.68	2	Resultado aceptable
P6	21.33	1	Resultado aceptable
P7	19.42	1	Resultado aceptable
P8	17.84	1	Resultado aceptable
P9	16.50	1	Resultado aceptable
P10	16.30	6	La imagen que se presenta en este punto no tiene buena resolución y por el bajo contraste no se pueden detectar los contornos de una buena manera.
P12	3	1	Resultado aceptable
P13	5	1	Resultado aceptable

P14	10	3	Resultado aceptable
P15	35,04	2	Resultado aceptable
P16	4	1	Resultado aceptable
P17	6	1	Placa en mal estado
P18	15	2	Resultado aceptable
P19	14	2	Resultado aceptable
P20	20	1	Resultado aceptable
P21	10	1	Resultado aceptable
P22	14	1	Resultado aceptable
P23	24	1	Resultado aceptable
P24	16	3	Resultado aceptable

Tabla 4.1 Tabla de análisis de resultados en función del ángulo

En la tabla 4.2 se muestra el análisis de los resultados por medio de una escala de medición del algoritmo para cada uno de los 24 puntos.

La escala será la siguiente:

- Excelente
- Buena
- Aceptable

- Mala

Punto	Observaciones
P1	Aceptable
P2	Aceptable
P3	Buena
P4	Excelente
P5	Excelente
P6	Excelente
P7	Excelente
P8	Buena
P9	Buena
P10	Mala
P11	Mala
P12	Buena
P13	Buena
P14	Aceptable
P15	Buena
P16	Buena
P17	Aceptable
P18	Excelente
P19	Excelente

P20	Excelente
P21	Excelente
P22	Excelente
P23	Excelente
P24	Excelente

Tabla 4.2 Tabla de escala de medición del algoritmo

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. El algoritmo usado en la primera etapa, es muy inestable y es muy susceptible a sombras lo cual provoca que las líneas hough no siempre cubran el borde superior e inferior por completo. La debilidad del algoritmo está en las líneas de Hough que es la parte clave, por lo tanto se tornaba muy inestable.
2. El haber utilizado blobs le dio una mayor posibilidad de detectar la silueta de la placa, pero al momento de buscar las esquinas con Harris el algoritmo mostró problemas, tal como no detectar correctamente las esquinas.
3. La corrección geométrica aplicada en este proyecto muestra una imagen corregida con la mayoría de las características de la placa bien definida de tal manera que las posteriores etapas del proyecto global tengan la facilidad de trabajar con estos resultados.

En conclusión después de analizar estos algoritmos y aplicar los respectivos ajustes y correcciones, y en base a las pruebas realizadas se determinó que el algoritmo implementado corrigió el problema de la orientación de las placas las

cuales tenían problemas geométricos tales como la rotación en los distintos planos.

Recomendaciones Y Futuros Trabajos

Como posibles recomendaciones para futuras mejoras del proyecto se podría mencionar lo siguiente:

- En el presente proyecto se trabajó con un tamaño promedio de placa que se obtuvo de los tamaños de los dos tipos de placas existentes que son de seis y siete caracteres, pero en un futuro esto se regularizará, ya que todos los vehículos tendrán placas de siete caracteres, por lo tanto ya no se debe trabajar con el tamaño promedio sino con el tamaño de placas regularizadas.

- El presente proyecto de corrección geométrica podría aplicarse con otro tipo de placas, por ejemplo de otros países, para lo cual se tendría que pasar por parámetro el tamaño de las placas en los diferentes países o crear una tabla con los tamaños establecidos y si en un país tienen varios tipos de placas entonces deberían ingresar el promedio del tamaño.

BIBLIOGRAFIA

[1] Wikipedia. “Automatic number plate recognition”.

[Online] http://en.wikipedia.org/wiki/Automatic_number_plate_recognition [2011]

[2] Structural Analysis Reference.

[Online] http://www710.univ-lyon1.fr/~bouakaz/OpenCV-0.9.5/docs/ref/OpenCVRef_StructAnalysis.htm [2010]

[3] CV Reference Manual

[Online]

http://www.seas.upenn.edu/~bensapp/opencvdocs/ref/opencvref_cv.htm [2002]

[4] Structural Analysis Reference

[Online] http://www.netaro.info/techinfo/OpenCV/opencv094-man/ref/OpenCVRef_StructAnalysis.htm [2010]

[5] cvCanny Method

[Online] http://www.emgu.com/wiki/files/1.3.0.0/html/08eaf5db-eeb2-62c9-578e-ab1de01709d0.htm [2008]

[6] OPERACIONES MORFOLÓGICAS

[Online]

http://www.tsc.uc3m.es/imagine/Curso_ProcesadoMorfologico/Contenido/Operaciones/OperacionesMorfologicas.html [2000]

[7] Elementos del procesado morfológico

[Online]

http://www.tsc.uc3m.es/imagine/Curso_ProcesadoMorfologico/Contenido/Elementos/Elementos.html [2000]

[8] Transformaciones Geométricas

[Online] http:// dis.um.es/~ginesgm/files/doc/pav/tema4.ppt [1998]

[9] Filtros y Transformaciones Locales

[Online] http:// dis.um.es/~ginesgm/files/doc/pav/tema3.ppt [1998]

[10] Transformaciones Geométricas

[Online] http:// alereimondo.no-ip.org/OpenCV/uploads/41/tema4.pdf [1998]

[11] Aproximación de Poligonos

[Online] www3.espe.edu.ec:8700/bitstream/21000/248/.../T-ESPE-027516.pdf [2009]

[12] CvBlob. “Blob library for OpenCV”
[Online] http://code.google.com/p/cvblob/ [2000]

[12] Geometric Image Transformations
[Online]
http://opencv.willowgarage.com/documentation/cpp/geometric_image_transformations.html [2002]