ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES.

"ESTUDIO COMPARATIVO ENTRE ALGORITMOS DE RECONOCIMIENTO DE BORDE PARA IDENTIFICACIÓN DE PLACAS DE AUTOS"

TESIS DE GRADO

Previa la obtención del título de

Ingeniero en Electrónica Telecomunicaciones y Redes

Presentado por:

Juan Pablo Paredes Muñoz
Leidy Tatiana Guerrero García

Riobamba – Ecuador
2012

Un sincero agradecimiento a nuestros padres quienes han sido nuestro apoyo incondicional inculcándonos el afán de superación, guiándonos con sus consejos y amor, lo cual nos ha permitido con perseverancia alcanzar este logro.

Al Departamento de Sistemas y Telemática por el apoyo brindado a este trabajo investigativo, en especial al Ing. Carlos Buenaño y al Ing. Gonzalo Allauca.

Al Ing. Jorge Menéndez por su colaboración y asesoramiento en la dirección de la presente tesis. Al Ing. Patricio Romero por su permanente disposición y acertados consejos en el desarrollo de este trabajo.

.

Cada una de las líneas escritas en la presente tesis involucran el apoyo de muchas personas que tal vez sin saberlo me permitieron llegar al final de una meta. Nunca podría dejar de agradecer el apoyo constante de mi madre Mariana Muñoz, gracias por ser una mujer luchadora que siempre ha estado dispuesta a compartir mis alegrías y pesares; a mi padre Juan Paredes quien con su ejemplo de vida siempre me impulsó a ser perseverante y lograr mis metas, a mi hermana Carolina Paredes por ser mi compañía por muchos años en mi vida, pero en especial a mi sobrino Carlos Abdir por simplemente cambiar nuestras vidas.

Juan Pablo

Este trabajo está dedicado a Dios, mi Creador, que bendice cada uno de mis pasos. A mis padres, Esperanza y Miguel, que inspiran mi vida con su ejemplo y amor. A mis hermanos, a mis abuelos y a toda mi familia por brindarme su apoyo incondicional para seguir adelante y alcanzar mis sueños. Y a todas las personas que creyeron en este proyecto y que colaboraron de cualquier manera para su culminación.

Leidy

FIRMAS RESPONSABLES Y NOTA

NOMBRE	**FIRMA**	**FECHA**
Ing. Iván Menes **DECANO DE LA FACULTAD** **DE INFORMÁTICA Y ELECTRONICA**	----------	----------
Ing. Pedro Infante **DIRECTOR DE LA ESCUELA** **DE INGENIERÍA ELECTRÓNICA** **EN TELECOMUNICACIONES Y REDES**	----------	----------
Ing. Jorge Menéndez **DIRECTOR DE TESIS**	----------	----------
Ing. Patricio Romero **MIEMBRO DEL TRIBUNAL** **DE TESIS**	----------	----------
Lcdo. Carlos Rodríguez **DIRECTOR CENTRO** **DE DOCUMENTACIÓN**	----------	----------

NOTA DE TESIS ----------

RESPONSABILIDAD DEL AUTOR

"Nosotros, Juan Pablo Paredes Muñoz y Leidy Tatiana Guerrero García, somos los responsables de las ideas, doctrinas y resultados expuestos en esta Tesis, y el patrimonio intelectual de la misma pertenecen a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO".

---------- ----------

Juan Pablo Paredes Muñoz Leidy Tatiana Guerrero García

ÍNDICE DE ABREVIATURAS

ANPR Automatic Number Plate Recognition, Reconocimiento automático de matrículas.

ANSI American National Standards Institute, Instituto Nacional Estadounidense de Estándares

BD Base de Datos.

BSD Berkeley Software Distribution, Distribución de software Berkeley.

BTN Buttons.

DBMS Data Base Manager System, Sistema Administrador de Base de Datos.

ESPOCH Escuela Superior Politécnica de Chimborazo.

FRM Forms.

GRB GroupBox.

HW Hardware.

IEEE Institute of Electrical and Electronics Engineers, Instituto de Ingenieros Eléctricos y Electrónicos

LBL Labels.

MySQL My Structured Query Language, Mi Lenguaje de Consulta Estructurada.

OCR Optical Charcater Recognition, Reconocimiento Óptico de Caracteres.

OpenCv Open Source Computer Vision, Visión por Ordenador de Código Abierto.

REQ Requerimientos

RGB Red Green Blue, Rojo, Verde y Azul.

RUP Rational Unified Process, Proceso Unificado de Desarrollo de Software.

SO Sistema Operativo.

SRS Especificación de Requerimientos Software.

SW Software.

TXT TextBox.

ÍNDICE GENERAL

CAPÍTULO III
ESTUDIO COMPARATIVO ENTRE LOS ALGORITMOS SOBEL Y CANNY

CAPÍTULO IV
DESARROLLO DE LA APLICACIÓN

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

INTRODUCCIÓN

Hoy en día es muy común ver cámaras de seguridad en los principales accesos a un establecimiento monitoreadas por personal durante largas jornadas de trabajo, con el fin de llevar un control de los vehículos que ingresan a la institución.

El reconocimiento automático de placas de automóviles es conocido también como ANPR, Automatic Number Plate Recognition, nombre que se le atribuye a la compañía inglesa Police Development Branch en 1976. Este sistema consiste en la adquisición de imágenes del tráfico por medio de cámaras especializadas, a las cuales se les hace un pre-procesamiento generalmente compuesto por tres pasos principales: localización de la placa dentro de la imagen, normalización de contraste y brillo, y realzar la imagen de la matrícula, por último la segmentación de caracteres; una vez terminado, la imagen queda lista para pasar por un sistema de reconocimiento de caracteres OCR, Optical Character Recognition [16], para extraer los datos alfanuméricos de la matrícula. Esta información, convertida ahora en pequeños paquetes de datos, se puede transmitir fácilmente (de ser necesario) a otros dispositivos remoto para su almacenamiento o procesamiento posterior [16].

El reconocimiento de placas de autos basado en visión artificial permite obtener un sistema que pueda capturar imágenes para posterior toma de decisiones, el ANPR puede almacenar las imágenes capturadas por las cámaras, así como el texto de la matrícula [21].

El software del sistema de Visión Artificial se ejecutará sobre un PC convencional y estará enlazado con una base de datos. El sistema realiza una adquisición de imágenes que son procesadas, de esta manera mediante algoritmos se detecta el contorno de interés, en este caso la placa del vehículo, esta característica especialmente interesante de la librería OpenCv de lograr el reconocimiento de patrones la hace atractiva en aplicaciones exclusivamente orientadas al proceso de imágenes [10].

La Escuela Politécnica de Chimborazo no cuenta actualmente con un tipo de sistema para el control de placas de automóviles de esta manera, este trabajo consiste en desarrollar e

implementar un prototipo que reconozca y realice la lectura de placas de vehículos en movimiento a partir de un cuadro de video. Para tal fin se deberá realizar el procesamiento adecuado de la imagen, para posteriormente pasar al reconocimiento de caracteres.

CAPÍTULO I

MARCO METODOLÓGICO

En este primer capítulo se describen los objetivos de la presente tesis, además de un breve estudio acerca del reconocimiento de placas de automóviles así como las razones por las cuales se realiza la actual investigación.

1.1. ANTECEDENTES

Usar la informática como apoyo a procesos de análisis y muestreo de información ha sido uno de los pilares más importantes en la automatización de tareas; su asimilación dentro de empresas e instituciones ha aumentado en los últimos años y la demanda de software inteligente se incrementa día a día.

Actualmente la Escuela Superior Politécnica de Chimborazo no posee un sistema que permita controlar la identificación de vehículos que ingresan a la institución debido al gran tráfico existente siendo este irregular durante el transcurso del día; razón por la cual es indispensable contar con algún sistema automatizado que facilite la mencionada tarea.

Mediante soluciones apoyadas en software actualmente podemos hacer tratamiento avanzado de imágenes; el uso de este tipo de herramientas permite el desarrollo de aplicaciones un ejemplo claro de ellos es el reconocimiento de bordes dentro de una imagen para determinar las placas de un vehículo.

Mediante el "Estudio comparativo entre Algoritmos de Reconocimiento de Borde para Identificación de placas de autos" se logra mejorar la identificación de automóviles en la ESPOCH.

1.2. JUSTIFICACIÓN

En la actualidad existen muchos algoritmos que permiten identificar bordes dentro de una imagen; CANNY y SOBEL destacan entre ellos por su amplia utilización y ventajas mostradas frente a otros algoritmos [3]. OpenCV es una implementación que contiene un conjunto de librerías enfocadas en visión artificial desarrolladas por Intel. La implementación de software a menudo es una tarea planificada que requiere tiempo de desarrollo por lo cual se hace indispensable el uso de herramientas eficientes como C++.

Mediante el estudio comparativo de los métodos CANNY y SOBEL se logra reconocer los bordes que existen en una imagen; teniendo esto diversas aplicaciones, en nuestro caso centraremos el estudio en el reconocimiento de placas de los vehículos en tránsito.

Cuando se logre reconocer los caracteres existentes en una placa de auto; se hace indispensable almacenar esta información para su posterior procesamiento. MySQL es un motor de base de datos bastante eficiente [10] el cual nos ayudará en el almacenamiento y gestión de datos.

El presente trabajo desarrollará un prototipo; el cual se la realiza por el interés del Departamento de Mantenimiento y Desarrollo Físico y Jefe de Seguridad; con el apoyo y coordinación técnica por parte del Departamento de Sistemas y Telemática en la ESPOCH.

La implementación del prototipo se llevará a cabo sobre el sistema operativo "Ubuntu" por su robustez y soporte a todo tipo de tecnologías.

1.3. OBJETIVOS

1.3.1. Objetivo General

Realizar un estudio comparativo entre algoritmos de reconocimiento de borde SOBEL y CANNY para la determinación del algoritmo más eficiente para la identificación de placas de autos en la ESPOCH.

1.3.2. Objetivos Específicos:

- Estudiar los algoritmos SOBEL y CANNY dentro del conjunto de librerías OpenCv.
- Realizar un estudio comparativo entre los algoritmos SOBEL y CANNY para determinar el más eficiente.
- Desarrollar un prototipo de reconocimiento de placas con el algoritmo escogido para buscar figuras geométricas, filtrar y reconocer caracteres, para su posterior implementación.

1.4. HIPÓTESIS

El estudio comparativo de los algoritmos SOBEL y CANNY de reconocimiento de bordes conlleva a la selección del más eficiente para la implementación de un prototipo que mejorará la identificación de automóviles en la ESPOCH.

CAPÍTULO II

MARCO TEÓRICO

El reconocimiento de imágenes es un proceso complejo que requiere una serie de pasos que sucesivamente transforman los datos a información que la computadora puede reconocer. Este capítulo expone los conceptos y la teoría que fundamenta los procesos por los que pasa la imagen desde su ingreso al sistema hasta su reconocimiento.

2.1. INVESTIGACIONES PRELIMINARES

Para el desarrollo del proyecto era necesaria la exploración de las herramientas a ser utilizadas y el estado de investigaciones similares, que pudieran servir de punto de inicio al proyecto, definiendo los algoritmos, bibliotecas y las técnicas de resolución de problemas que se usan en el desarrollo del sistema.

La visión artificial, también conocida como visión por computador (del inglés computer vision) o visión técnica, es un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen [13].

2.1.1. OpenCV

OpenCV (Open Source Computer Vision) es una biblioteca de funciones de programación para la visión de computadora en tiempo real. En sus principios fue desarrollada por Intel, disponibles en *http://SourceForge.net/projects/opencvlibrary*. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicativos de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas [12].

OpenCv es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. En Windows se puede probar con C#, Visual y C++ en Visual Studio. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica.

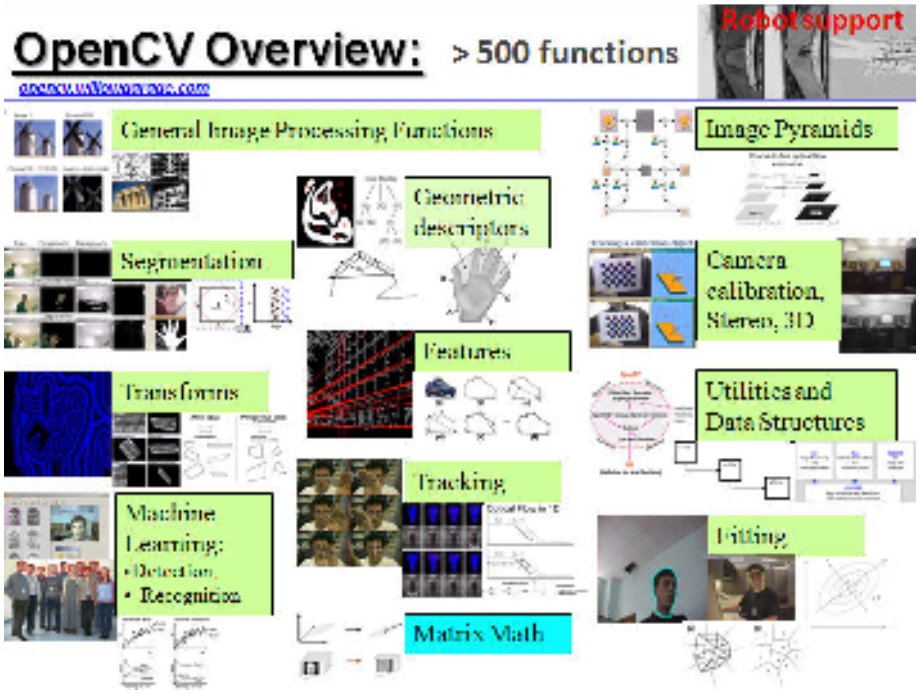


Figura II- 1. Resumen de las funciones de OpenCv

El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados [15], aprovechando además las capacidades que proveen los procesadores multi núcleo.

OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel.

La Librería de visión por computador open source, trata de un grupo de funciones orientadas a la resolución de problemas típicos de la visión por computador y con una comunidad detrás muy importante de profesionales de la visión que se encarga de mantenerla actualizada [10].

2.1.1.1. Estructura de OpenCV

Estas bibliotecas se dividen en cinco grandes grupos, en donde se encuentran divididas las funciones dependiendo de su utilidad (*Figura II-2*): CXCORE, donde se encuentran las estructuras y algoritmos básicos que usan las demás funciones; CV donde están implementadas las funciones principales de procesamiento de imágenes y algoritmos de visión; HighGUI, con todo lo relacionado a la sencilla interfaz grafica de OpenCV y las funciones que permitan importar imágenes y video ; ML, que cuenta con algoritmos de aprendizaje, clasificadores y demás; y CvAux, con funciones experimentales entre ellas BG/FG, esta última biblioteca cuenta con menos documentación que las demás siendo en algunos casos inexistente una documentación oficial del uso de la función ó incluyendo solo el artículo del algoritmo implementado en otros casos.

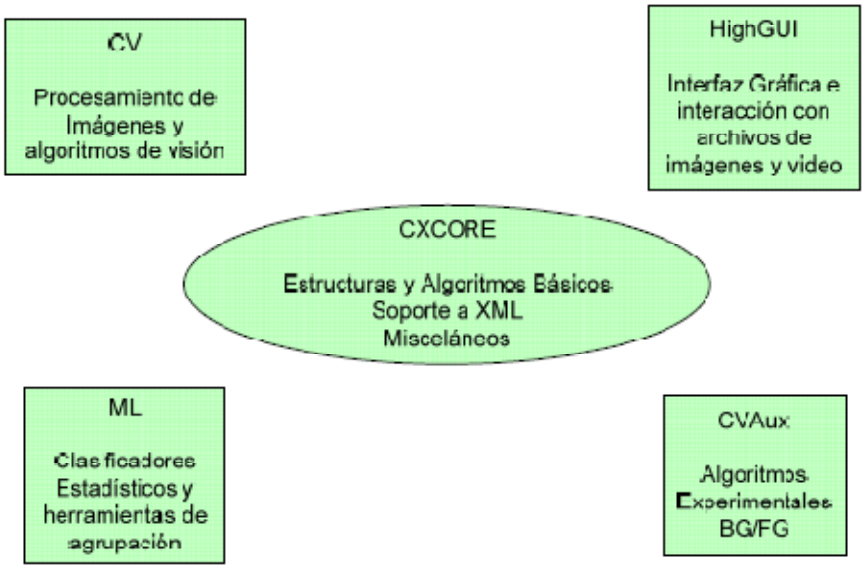


Figura II- 2. Distribución de bibliotecas OpenCv

2.1.1.2. Aplicaciones conocidas

Entre los usos que se le ha dado a esta herramienta se puede citar:

- OpenCV ha sido usada en el sistema de visión del vehículo no tripulado Stanley de la Universidad de Stanford, el ganador en el año 2005 del Gran desafío DARPA.
- OpenCV se usa en sistemas de vigilancia de vídeo
- OpenCV es la clave en el programa Swistrack, una herramienta de seguimiento distribuida

2.2. ALGORITMOS INVESTIGADOS

La localización acertada de la placa es fundamental para el correcto procesamiento de la imagen, al aplicar operadores de detección de líneas se puede encontrar los bordes que conforman la placa.

Existen varios algoritmos de borde que pueden ser utilizados en el sistema, entre los cuales se ha investigado a Sobel y a Canny por sus destacadas características, el método se basó en probar cada uno de los algoritmos antes descritos y mediante pruebas compararlos hasta obtener como resultado el más eficiente para la implementación del sistema.

2.2.1. Operador de Sobel

El operador Sobel es una técnica basada en el gradiente, es decir, se logra la localización de los puntos en los que se produce la variación de intensidad se aplica la primera derivada y se busca los picos más grandes.

Los operadores basados en el gradiente asumen que los bordes de una imagen son píxeles con un alto gradiente. Un rápido índice de cambio de intensidad en alguna dirección dada por el ángulo del vector gradiente puede observarse en los píxeles de los bordes. En la *Figura II-3*. se muestra un píxel de borde ideal con su correspondiente vector de gradiente. En el píxel, la intensidad cambia de 0 a 255 en dirección del gradiente. La magnitud del gradiente indica qué tan marcado está el borde. Si calculamos el gradiente en regiones uniformes obtendremos un vector de valor 0, lo que significa que no hay píxeles de borde [15].

Un píxel de borde se describe mediante dos características importantes:

1. Intensidad del borde, que es igual a la magnitud del gradiente.

2. Dirección del borde, que es igual al ángulo del gradiente.

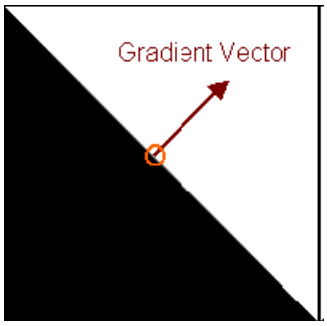


Figura II- 3. Gradiente de un píxel de borde.

Es un operador discreto, es decir en un conjunto finito de puntos se puede tomar por separados cada punto, en este caso actúa sobre los valores de cada pixel y los relaciona con los de su entorno inmediato analizando la velocidad de cambio en cada dirección. Calcula lo que se conoce como gradiente de la función de intensidad de una imagen [14].

Figura II- 4. Reconocimiento de contornos mediante el operador Sobel.

2.2.1.1. Funcionamiento

Este operador toma en cuenta todos los píxeles vecinos al píxel de interés, funciona aplicando una ventana de tamaño 3x3 sobre todos los píxeles de la imagen [15].

A0	A1	A2
A7	(i,j)	A3
A6	A5	A4

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Tabla II- I. Píxeles del entorno de un punto y dos ejemplos de máscaras

El operador de Sobel es la magnitud del gradiente calculado mediante:

$$M = \sqrt{s_{x+}^2 s_y^2}$$

Donde las derivadas parciales se calculan mediante:

$$S_x = (A_0 + cA_7 + A_6) - (A_4 + cA_3 + A_2)$$

$$S_y = (A_0 + cA_1 + A_2) - (A_4 + cA_5 + A_6)$$

Con la constante *c*=2.

Tanto Sx como Sy, reciben el nombre de operadores Sobel. El requisito básico de un operador de derivación es que la suma de los coeficientes de la máscara sea nula, para que la derivada de una zona uniforme de la imagen sea cero.

Estos operadores de gradiente pueden implementarse utilizando máscaras de convolución.

Es importante notar que este operador le da más peso a los píxeles más cercanos al centro de la máscara.

2.2.2. Algoritmo de Canny

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 píxeles en las que existe un cambio brusco de nivel de gris. En el tratamiento de imágenes, se trabaja con píxeles y en un ambiente discreto, es así que en el algoritmo de Canny utiliza máscaras, las cuales representan aproximaciones en diferencias finitas.

2.2.2.1. Criterios del Algoritmo de Canny

En 1986, Canny propuso un método para la detección de bordes, el cual se basaba en tres criterios, estos son:

- Un criterio de detección expresa el hecho de evitar la eliminación de bordes importantes y no suministrar falsos bordes.
- El criterio de localización establece que la distancia entre la posición real y la localizada del borde se debe minimizar.
- El criterio de una respuesta que integre las respuestas múltiples correspondientes a un único borde.

2.2.2.2. Funcionamiento

Uno de los métodos relacionados con la detección de bordes es el uso de la primera derivada, la que es usada por que toma el valor de cero en todas las regiones donde no varía la intensidad y tiene un valor constante en toda la transición de intensidad. Por tanto un cambio de intensidad se manifiesta como un cambio brusco en la primera derivada [4], característica que es usada para detectar un borde, y en la que se basa el algoritmo de Canny.

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.

2.2.2.3. Obtención del gradiente

Para la obtención del gradiente, lo primero que se realiza es la aplicación de un filtro gaussiano a la imagen original con el objetivo de suavizar la imagen y tratar de eliminar el posible ruido existente. Sin embargo, se debe de tener cuidado de no realizar un suavizado excesivo, pues se podrían perder detalles de la imagen y provocar un pésimo resultado final. Este suavizado se obtiene promediando los valores de intensidad de los píxeles en el entorno de vecindad con una máscara de convolución de media cero y desviación estándar σ. En la *Figura II-5.* se muestran dos ejemplos de máscaras que se pueden usar para realizar el filtrado gaussiano.
Una vez que se suaviza la imagen, para cada píxel se obtiene la magnitud y módulo (orientación) del gradiente, obteniendo así dos imágenes.

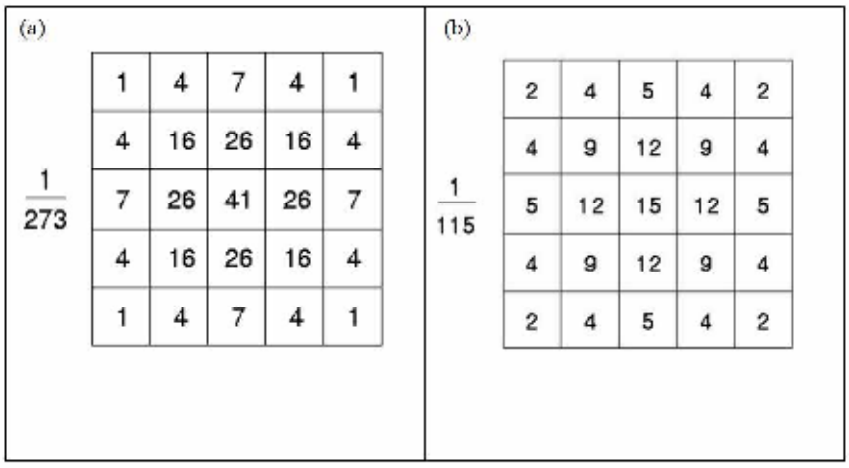


Figura II- 5. Máscaras de convolución recomendadas para el obtener el filtro gaussiano. La máscara (a) fue obtenida de [2], mientras que la máscara (b) fue obtenida de [19].

2.2.2.4. Supresión no máxima al resultado del gradiente

Las dos imágenes generadas en el paso anterior sirven de entrada para generar una imagen con los bordes adelgazados. El procedimiento es el siguiente: se consideran cuatro direcciones identificadas por las orientaciones de 0º, 45º, 90º y 135º con respecto al eje horizontal. Para cada píxel se encuentra la dirección que mejor se aproxime a la dirección del ángulo de gradiente.

2.2.2.5. Histéresis de umbral a la supresión no máxima

La imagen obtenida en el paso anterior suele contener máximos locales creados por el ruido. Una solución para eliminar dicho ruido es la histéresis del umbral. El proceso consiste en tomar la imagen obtenida del paso anterior, tomar la orientación de los puntos de borde de la imagen y tomar dos umbrales, el primero más pequeño que el segundo.

Para cada punto de la imagen se debe localizar el siguiente punto de borde no explorado que sea mayor al segundo umbral. A partir de dicho punto seguir las cadenas de máximos locales conectados en ambas direcciones perpendiculares a la normal del borde siempre que sean mayo res al primer umbral. Así se marcan todos los puntos explorados y se almacena la lista de todos los puntos en el contorno conectado. Es así como en este paso se logra eliminar las uniones en forma de Y de los segmentos que confluyan en un punto.

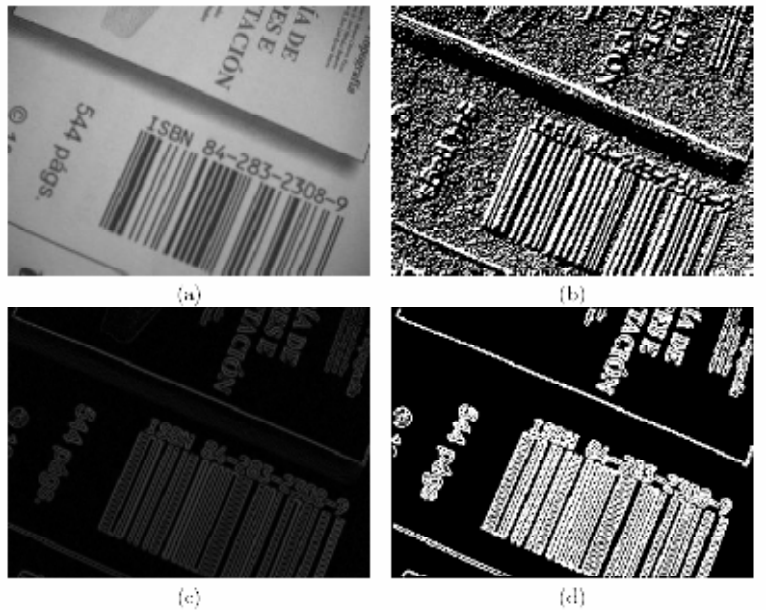


Figura II- 6. Resultado de aplicar el detector de bordes de Canny: (a) imagen original; (b) orientación; (c) supresión no máxima; (d) histéresis de umbral.

2.2.2.6. Un cuarto paso

Frecuentemente, es común que un cuarto y último paso se realice en el algoritmo de Canny, este paso consiste en cerrar los contornos que pudiesen haber quedado abiertos por problemas de ruido.
Un método muy utilizado es el algoritmo de Deriche y Cocquerez. Este algoritmo utiliza como entrada una imagen binarizada de contornos de un píxel de ancho. El algoritmo busca los extremos de los contornos abiertos y sigue la dirección del máximo gradiente hasta cerrarlos con otro extremo abierto.

El procedimiento consiste en buscar para cada píxel uno de los ocho patrones posibles que delimitan la continuación del contorno en tres direcciones posibles. Esto se logra con la convolución de cada píxel con una máscara específica. Cuando alguno de los tres puntos es ya un píxel de borde se entiende que el borde se ha cerrado, de lo contrario se elige el píxel con el valor máximo de gradiente y se marca como nuevo píxel de borde y se aplica nuevamente la convolución. Estos pasos se repiten para todo extremo abierto hasta encontrar su cierre o hasta llegar a cierto número de iteraciones determinado.

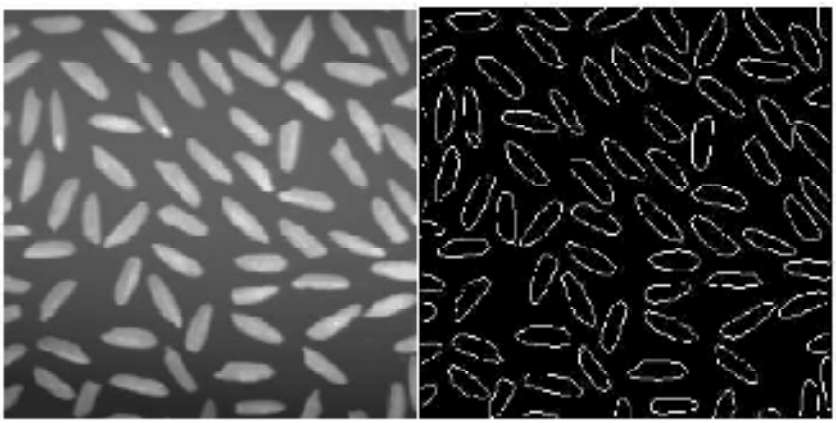


Figura II- 7. Resultado de la detección de bordes mediante el algoritmo de Canny.

CAPÍTULO III

ESTUDIO COMPARATIVO ENTRE LOS ALGORITMOS SOBEL Y CANNY

En este capítulo se muestra las etapas para efectuar el reconocimiento de placas utilizando las herramientas estudiadas en el apartado anterior, para obtener la mejor solución se decidió realizar dos aplicaciones con los algoritmos de borde antes señalados. Finalmente se entregan los resultados de las comparaciones de las dos metodologías evaluando los factores que inciden en la confianza del sistema.

3.1. DESCRIPCIÓN GENERAL Y ESPECIFICACIONES

Desde hace poco tiempo se ha incrementado en gran medida el interés por lo Artificial; las razones son muchas pero todas convergen hacia el extraordinario avance de la tecnología y las diversas herramientas con las que nos podemos apoyar para la consecución de nuestros objetivos.

Las aplicaciones en las que pueden ser utilizadas un reconocimiento de placas vehiculares son numerosas; sin embargo el estudio de la presente tesis centra su objetivo en ser un apoyo y ayuda en el reconocimiento vehicular en la "Escuela Superior Politécnica de Chimborazo".

Un problema suele ser más fácil de resolver dividiéndolo en fragmentos de tal manera que se ha decidido implementar cuatro etapas claramente definidas como: El pre procesamiento de la imagen, localización de bordes, segmentación y reconocimiento de caracteres. Cada una de las etapas mencionadas posee múltiples opciones de algoritmos los cuales son genéricos en el procesamiento digital de imágenes; siendo uno más eficiente que otro en

determinadas circunstancias; sin embargo la eficiencia puede ser un concepto relativo; ya que podría recaer en el hecho de que eficiencia sea el tiempo en que tarda el sistema en responder o tal vez la detección correcta de placas del sistema obviando el tiempo de respuesta; pues es una postura relativa al punto de vista que sea sometido.

3.1.1. Características Fotográficas

Para un adecuado reconocimiento de la placa vehicular se deberá tomar en cuenta varias características y estas son:

- La cámara de adquisición de la imagen y el vehículo no estén a distancias muy largas, ni muy cortas, por las que se deben cumplir las siguientes condiciones:
    - Máxima distancia del foco de la cámara al vehículo es de 2 mts.
    - Mínima distancia del foco de la cámara al vehículo es de 0.5 mts

La posición de la cámara con respecto al vehículo puede colocarse frente a este o a un lado. En el caso de ubicar la cámara a un costado, el ángulo entre la posición de la cámara con respecto al vehículo debe estar entre de 20 º y 60 º.

- La captura de las imágenes no debe realizarse con el vehículo en movimiento.
- Iluminación de la escena en el día.
- Caracteres de la placa no pueden estar despintados (más del 10%).
- La placa puede estar girada con un ángulo de inclinación máximo de 10º [17].

3.1.2. Placas Automotrices

Cada automóvil posee una o dos placas para su identificación, estas placas metálicas contienen grabados los caracteres de la matricula de forma inalterable.

3.1.2.1. Características

Dependiendo de la legislación de cada país; las características de las placas pueden diferir. En Ecuador los artículos los artículos 104 a 108 del Reglamento a la Ley de Tránsito y Transporte Terrestre definen las características para las placas de identificación que deben poseer aquellos vehículos que circulan por el país.

3.1.2.2. Dimensiones

Todas las placas ecuatorianas tienen forma rectangular, sus dimensiones son: 30 cm de ancho y 15 cm de alto. El ancho para cada carácter es de 4cm y el grosor del trazo es de 1cm [18].

Figura III- 1. Placa de automóvil

Tiene 6 caracteres escritos de color negro dentro de un marco negro, como se muestra en la figura III-1, y están divididos en 3 letras y 3 números, también existen placas que cuentan con 7 caracteres de los cuales 4 son números.

TIPO DE VEHÍCULO	COLOR
Alquiler	Naranja
Entidades del Estado	Oro o Verde fondo bandera del Ecuador

Particulares	Blanco
Servicio Diplomático, Consular, Asistencia Técnica, Organismos Internacionales	Azul
Vehículos de manera Temporal	Roja

Tabla III- I.Tipos de Placas

Los colores establecidos son de trascendental importancia ya que definen el tipo de vehículo. En nuestro país se ha establecido básicamente cuatro colores: blanco, rojo amarillo y azul. El color blanco está destinado a los vehículos particulares, los de placa amarilla para vehículos de servicio público y los de placa azul para vehículos oficiales [17].

3.1.2.3. Material

Las placas son hechas en lámina metálica de aluminio, y recubiertas con papel reflectivo, de color acorde al tipo de placa.
Las letras están en relieve de 2 mm y son cubiertos de negro mate.

3.2. PRE PROCESAMIENTO DE LA IMAGEN

Antes de proceder a la extracción de segmentos en una imagen, se convierte la imagen de color a niveles de gris ponderando adecuadamente los canales de rojo (R), verde (G) y azul (B), luego se obtienen los bordes existentes en la misma. Para cumplir la mencionada tarea se optó por utilizar dos algoritmos “Sobel” y “Canny”, los cuales se emplean para estimar únicamente la intensidad de borde aproximada en cada punto de la imagen.

En ocasiones puede resultar adecuado realizar un procesado previo de la imagen original con algún supresor de ruido debido a que es común al analizar la imagen capturada encontrar problemas de ruido, el método utilizado para este fin es el Filtro Gaussiano (CV_GAUSSIAN) con lo cual se obtiene un suavizado que puede provocar perdida del detalle de la imagen.

3.2.1. Modelo de color RGB

Una imagen RGB hace referencia a la combinación de colores que existe en la misma para poder obtener el color deseado; tomando en cuenta que una imagen no es más que un arreglo de píxeles MxN x 3; donde cada píxel de color es un componente de la tripleta correspondiente a rojo, verde y azul en una localización espacial específica en una imagen RGB, figura III-2 [18].

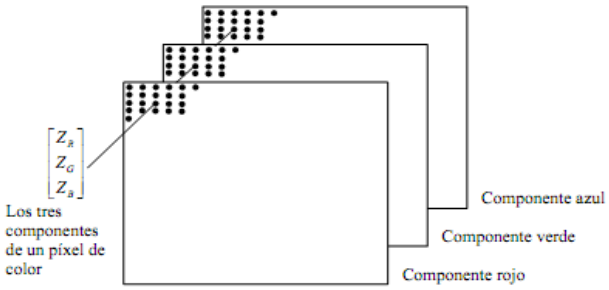


Figura III- 2. Esquema de una imagen a color, con 3 planos R, G y B

Se puede mostrar gráficamente el concepto de RGB como un cubo de color, pues como se aprecia en la figura anterior tiene ancho, alto y también fondo.

Los vértices del cubo son los colores primarios (rojo, verde y azul) y los secundarios (cian, magenta y amarillo).

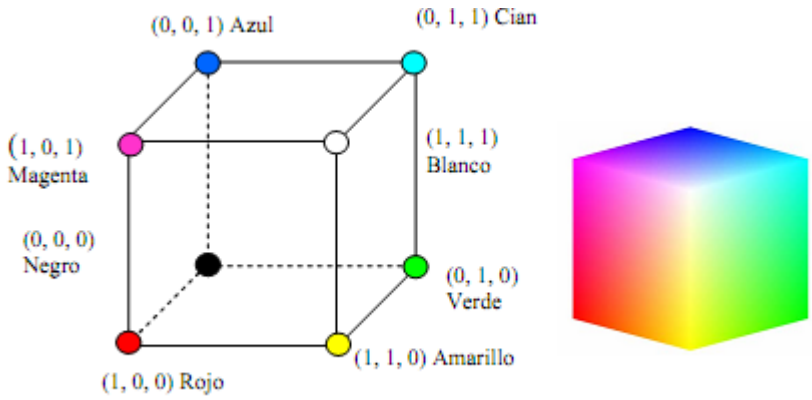


Figura III- 3. Cubo RGB

3.2.1.1. Tipos de modelo

Existen otros modelos utilizados para el procesamiento de imágenes en color que se detallan a continuación.

a) Modelo YIQ

Una de las principales ventajas de este formato es que la información en escala de grises es separada de los datos de color. Los datos de la imagen consisten en tres componentes: Los datos de la imagen consisten en tres componentes:

- Luminancia (Y),
- Tono (I),
- Saturación (Q).

El componente luminancia representa la información en escala de grises, y los otros dos componentes representan la información del color. Los componentes YIQ son obtenidos desde los componentes de una imagen RGB usando la transformación:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Figura III- 4. Componentes YIQ

Note que los elementos de la primera fila suman 1 y los de las siguientes 2 filas suman 0, esto es porque para una imagen en escala de grises todos los componentes RGB son iguales y los componentes I y Q son cero.

b) Modelo HSI

En el modelo HSI (Hue, Saturation, Intensity), la componente de la intensidad (I), está separada de la información cromática contenida en la imagen y las otras dos componentes están muy relacionadas con la forma en que el ser humano percibe el color. Estas características hacen que el modelo HSI sea una herramienta ideal para desarrollar algoritmos de procesamiento de imágenes basados en algunas propiedades de la percepción del color del sistema visual humano.

Las componentes de color tono y saturación del modelo HSI, están definidas con respecto al triángulo mostrado en la figura III-5, se puede observar que el tono, H, es el ángulo del vector con respecto al eje rojo. Así cuando H = 00, el color es rojo, cuando H= 600, el color es amarillo y así sucesivamente. La saturación, S, es el grado en que el color no está diluido con blanco y es proporcional a la distancia desde el centro hasta el triángulo [3,5], la intensidad se mide con respecto a una línea perpendicular al plano del triángulo y que pasa por su centro.

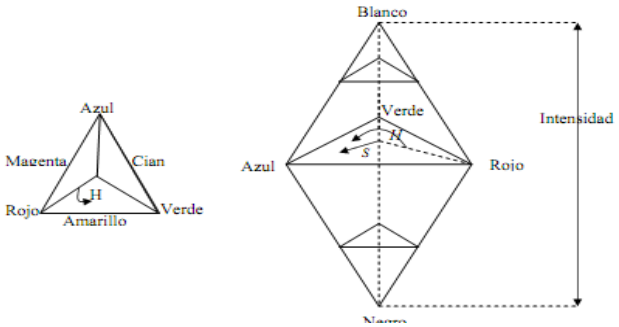


Figura III- 5. Triángulo de color HSI

Dada una imagen RGB, el componente H de cada píxel RGB es obtenido usando la ecuación.

$$H = \begin{cases} \theta & if\ B \le G \\ 360 - \theta & if\ B > G \end{cases}$$

Donde

$$\theta = COS^{-1}\left\{ \frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)^{1/2}\right]} \right\}$$

El componente está dado por

$$s = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)]$$

Finalmente, el componente de la intensidad esta dado por

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

Esto es asumiendo que los valores RGB han sido normalizados al rango [0 ,1], y que el ángulo θ es medido con respecto a el eje rojo del espacio RGB. El tono, H, puede ser normalizado al rango [0, 1] dividiendo los valores resultantes por 360 en la ecuación de H.

c) Modelo CIE

En el Sistema CIE (Commission Internationale de l'Eclairage), un color se define por las componentes de X, Y y Z, figura III-6. La proyección en el eje X Y produce el diagrama de cromaticidad de la CIE, figura III-7.

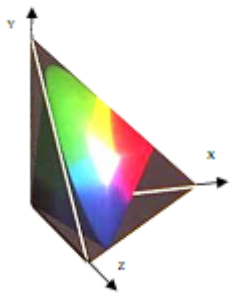


Figura III- 6. Componentes de CIE XYZ

Estas componentes no son reales sino imaginarias, pero cualquier color se puede definir como una combinación de estas. Asimismo, toda la información de intensidad está comprendida únicamente en la componente Y.

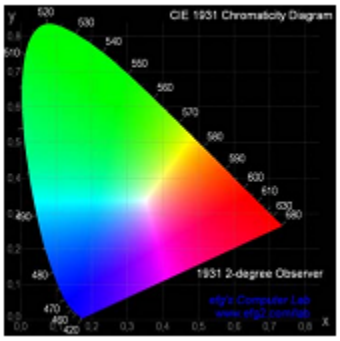


Figura III- 7. Diagrama CIE

El diagrama de cromaticidad está basado en los valores triestimulos normalizados x, y y z donde, x = X/(X+Y+Z), y = Y/(X+Y+Z), y z = Z/(X+Y+Z). Esta normalización remueve el brillo del diagrama, así que solamente dos coordenadas x y y son necesarias para describir la cromaticidad [20].

Los componentes se pueden extraer a partir de una imagen RGB como:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0.000 & 0.066 & 0.116 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

3.2.2. Filtro Gaussiano

Las imágenes tienen cierta cantidad de ruido, ya sea por el medio de trasmisión de la señal o por el sensor de la cámara. La manifestación del ruido ocurre generalmente en píxeles aislados que toman un valor de gris diferente al de sus vecinos.
Existen varios tipos de ruido, pero el tipo que más se asemeja a las imágenes es el Gaussiano ya que produce pequeñas variaciones en la imagen.
Si se considera una imagen como una función real de dos variables I(x, y), el *Filtro Gaussiano* es una convolución entre la señal de entrada (la imagen I(x, y)) y otra función real g(x, y) (una función gaussiana bidimensional por ejemplo). En términos genéricos la aplicación de una convolución a una imagen se puede escribir discretamente como:

$$g(i,j) * I(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} g(k,l) I(i-k, j-l)$$

La manera más simple de ver el efecto de usar el filtro gaussiano anterior es considerar su versión bidimensional. Cada nuevo pixel queda definido por el promedio de los valores de sus píxeles vecinos, dándoles más peso mientras estén más cerca al pixel que se está procesando.

3.2.3 Conversión a escala de gris

Las imágenes se convierten a escala de grises para poder manipularlas, donde los datos de la imagen consisten en un solo canal que representa la intensidad, brillo o densidad de la imagen. En la mayoría de los casos, los valores positivos son los que tienen sentido (e.g la intensidad de la luz). Típicamente una imagen en escala de gris usa 8 bits (1 byte) por pixel y el rango de los valores de intensidad se encuentra entre [0-255], donde el valor mínimo del rango representa el brillo mínimo (Negro) y el valor máximo representa el brillo máximo. La función predefinida por OpenCv es cvCvtColor(origen,destino, CV_RGB2GRAY).

3.3. ADQUISICIÓN DE LA IMAGEN DIGITAL

Para capturar la imagen utilizando Opencv, se hace uso de la función cvLoadImage. A continuación se describen 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.

Si la variable “flag” es mayor que cero obligamos a que la imagen cargada sea una imagen de color de 3 canales (RGB), al ser la mencionada variable igual que cero obligamos a ser una imagen de intensidad de un canal. Finalmente al ser menos que cero; cargamos la imagen 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.

3.4. LOCALIZACIÓN DE BORDES

Las técnicas de localización se refieren a los algoritmos utilizados para detectar la zona en la cual se encuentra la placa.

Se presentan cuatro diferentes métodos en esta etapa:

- Transformada de Hough.
- Cálculo de los gradientes de la imagen (método del gradiente).
- Morfología matemática, (método morfológico)
- Basada en el color.

Tanto el Operador Sobel como el Detector Canny usan la técnica basada en la primera derivada (gradiente), de forma que el valor de la variable es mayor cuanto más rápidas son las variaciones, además Canny tiene la características que emplea mascaras de convulsión.

3.4.1. Detección de bordes

El borde de una imagen digital se podría definir como la transición entre dos regiones de niveles de gris significativamente distintos. Esta suministra una importante información sobre las fronteras de los objetos y puede ser utilizada para segmentar la imagen, reconocer objetos, etc. Es así que una buena detección de bordes es fundamental para la visión

artificial orientada al reconocimiento de objetos. Si la detección es pobre o da bordes donde no los hay, el programa puede llegar a reconocer una placa donde no la hay.
La mayoría de las técnicas para detectar bordes emplean operadores locales basados en distintas aproximaciones discretas de la primera y segunda derivada de los niveles de grises de la imagen.

Para la etapa se utilizaron dos algoritmos que permiten la detección de bordes, el Operador Sobel y el Detector de bordes Canny, los cuales se detallan en el capítulo 2.

3.4.1.1. Operador Sobel

Técnicamente es un operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad de una imagen.

void cvSobel(src, dst, xorder, yorder, aperture_size=3);

Donde

src: Imagen de origen
dst: Imagen de destino
xorder: Orden de derivada en X, toma un valor igual o mayor que 1.
yorder: Orden de derivada en Y, toma un valor igual o mayor que 0.
aperture_size: Tamaño de la extensión del kernel de Sobel, este puede tomar los valores de 1,3,5 o 7. El kernel depende de la imagen de origen.
Muy amenudo la funcion suele utilizar los siguientes valores:

cvSobel(image,Edge_image1,1,0,3);

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

Figura III- 8. Kernel de convolucion del detector de Sobel para el primer caso.

O, cvSobel(image,Edge_image1,0,1,3);

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

Figura III- 9. Kernel de convolucion del detector de Sobel para el segundo caso.

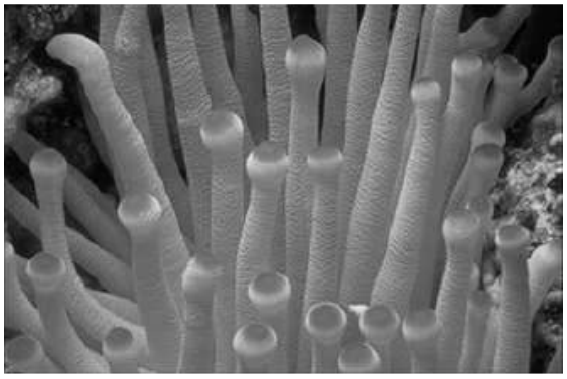
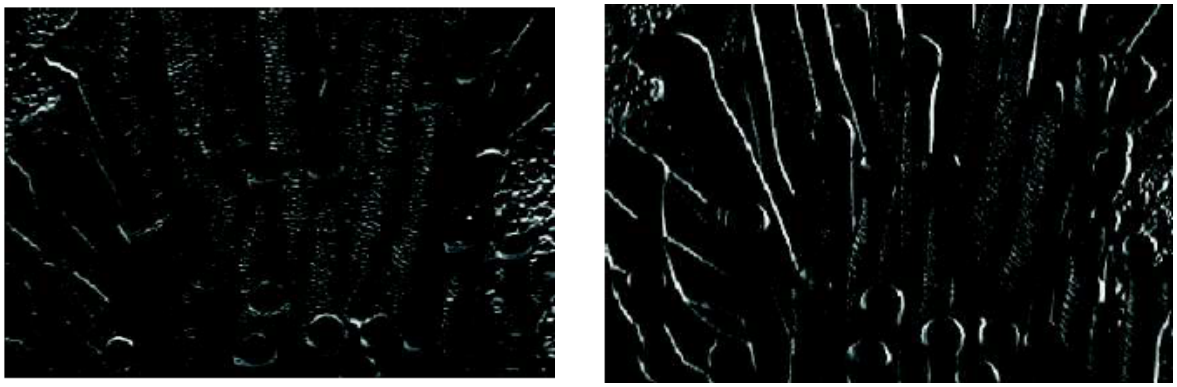


Figura III- 10. Imagen original.



(a) (b)

Figura III- 11. Gradiente en x (a) y en y (b).

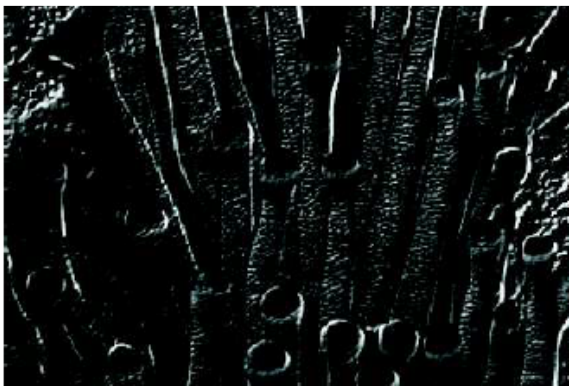


Figura III- 12. Gradiente de Sobel en ambas direcciones.

Se calcula solamente en 2 direcciones ortogonales y luego se calcula el modulo. Se tiene la ventaja de que con una máscara de 3 x 3 pixeles, se obtiene un alisado adicional (diferenciación + suavizado). Ya que la derivación acentúa el ruido, el efecto de suavizado es particularmente interesante, puesto que elimina parte del ruido, de esta manera el procesamiento es más inmune al ruido [11].

3.4.1.2. Detector de bordes Canny

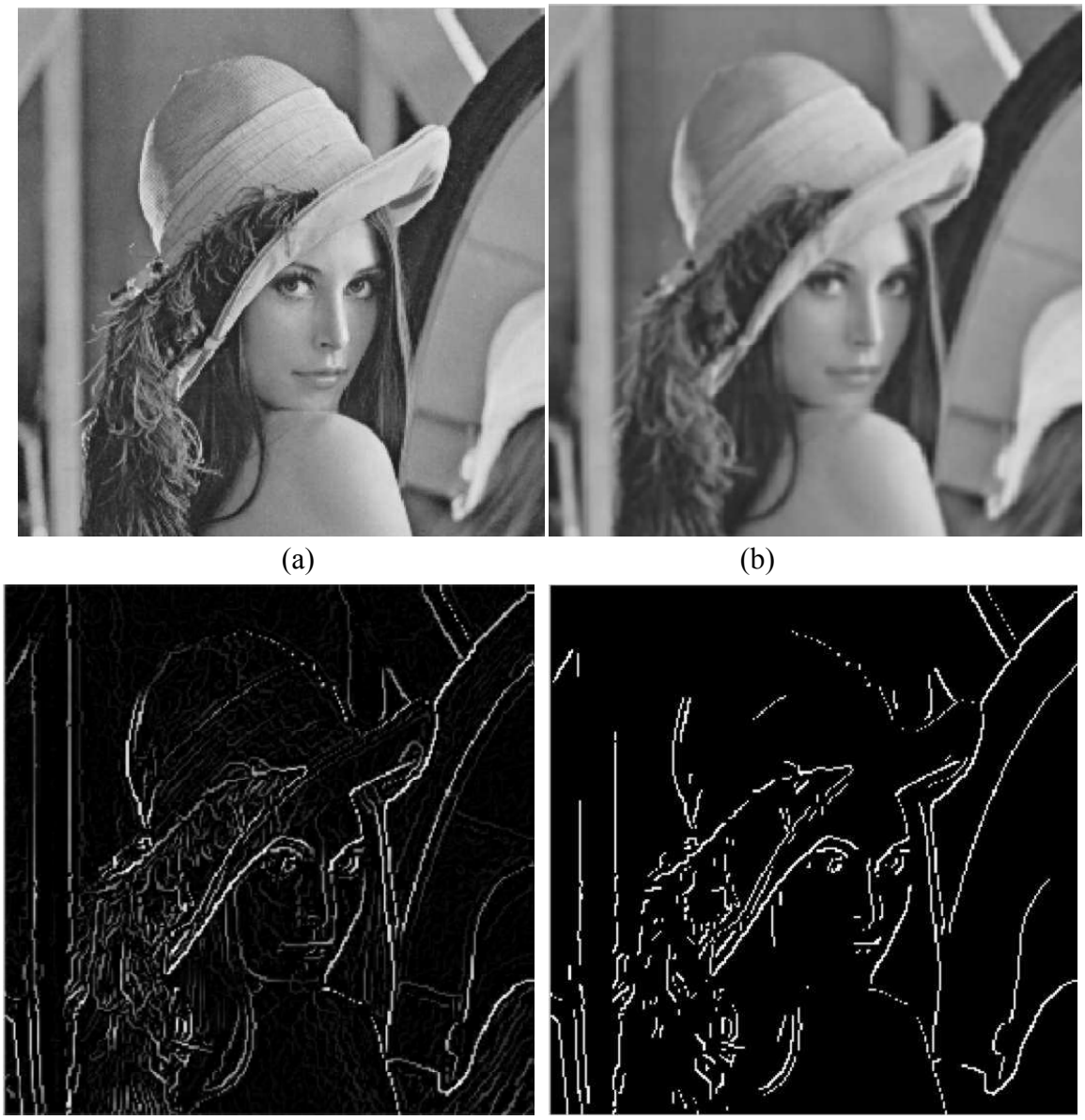
Para la ejecución del algoritmo de Canny, se usa la siguiente función:

void cvCanny(image, edges, umbral1, umbral2, aperture_size=3);
Donde
image: Imagen de origen
edges: Imagen de destino
umbral1: Primer Umbral, toma un valor igual o mayor que 50.
umbral2: Segundo Umbral, toma un valor igual o mayor que 100.
aperture_size: Tamaño de extensión del kernel, este puede tomar un valor igual o mayor que 3.

La funcion cvCanny busca los bordes de la imagen de entrada y marca los bordes en la imagen de salida. El proceso de Histéresis usa un umbral alto (Th) y un umbral bajo (Tl), cualquier pixel que tenga un valor mayor a Th se presume que es un pixel de contorno, y es marcado como tal inmediatamente. Luego, cualquier pixel que esté conectado con este pixel de contorno y que tenga un valor mayor a Tl también se vincula al borde como un

pixel de contorno, y también marcado. El marcado de los vecinos se puede realizar recursivamente, como se hace en la función histéresis.
Muy amenudo la funcion suele utilizar los siguientes valores:

cvCanny(image, edges,50,100,3);



(a) (b)

(c) (d)

Figura III- 13. Resultado de aplicar el detector de bordes de Canny: (a) imagen original; (b) imagen después de haber pasado un filtro gaussiano; (c) imagen después de haber pasado por la supresión de no máximos; (d) imagen después del proceso de histeresis, y en los que se usado umbrales de 100 y 130.

(a) (b)

Figura III- 14. Se muestran otras dos figuras en las que se aplicó el algoritmo de detección de bordes de Canny con umbrales diferentes al aplicar la histéresis. En la figura (a) se usó umbrales con los valores 110 y 150. En la figura (b) se usó umbrales con los valores 120 y 180.

Canny favorece la detección de verdaderos positivos al maximizar el ratio señal ruido, realiza una buena localización con el fin de marcar bien los bordes además de favorecer la identificación de falsos verdaderos es decir, zonas que no son bordes no serán marcadas [9]. Este método es uno de los más robustos contra el ruido (filtro óptimo) en comparación con el método Sobel.

3.5. SEGMENTACIÓN

Las salidas del bloque anterior se convierten en las entradas de este, de esta manera se verifica la elección de imágenes que cumplan con las características propias de una placa, luego se extraen dichos objetos mostrando a la salida imágenes aisladas de las placas encontradas.

Según la morfología de una placa que corresponde a un rectángulo horizontal, las imágenes candidatas pasan por un filtro que evalúa el ancho y alto que fue especificado en las dimensiones de la sección 3.2, esta característica depende del ángulo de captura y ubicación

del carro dentro de la imagen, por lo tanto los limites corresponden a las tomas de 20 º y 60º.

Figura III- 15. Placa de automóvil de Ecuador

La segmentación permite cambiar la representación de la imagen en otra más significativa que permite mayor simplicidad en el análisis de la siguiente etapa, esta es usada típicamente para localizar objetos en imágenes.

El resultado de la segmentación de imágenes es un conjunto de regiones que colectivamente cubren la imagen original entera, o un conjunto de contornos extraídos de la imagen. Cada píxel en cada región comparte características similares como color, intensidad o textura respecto de los demás píxel en la misma región. Los píxeles en regiones adyacentes tienen grandes diferencias respecto de las mismas características [5].

3.5.1. Coincidencias geométricas

Uno de los procesos más importantes en el reconocimiento de Placas vehiculares es encontrar todas las posibles figuras geométricas que se encuentren en la imagen. Para poder encontrar figuras geométricas en una imagen OpenCv aplica internamente la "Transformada de Hough".

La "Transformada de Hough" procesa toda la imagen, haciéndola robusta ante la presencia de ruido o las discontinuidades de las etapas previas, ya que para hacer esta transformada primero hemos de binarizar la imagen con los bordes seleccionados, o sea tenemos que aplicar un detector de bordes. A partir de aquí el algoritmo intentará buscar figuras geométricas, como pueden ser líneas, círculos elipses.

3.5.2. Filtrado De Rectángulos

Al pasar el proceso de detección de figuras geométricas se ha procedido a filtrar exclusivamente los rectángulos presentes en la imagen para ello nos basamos en el principio de encontrar exclusivamente cuatro vértices para determinar que estamos tratando con una placa vehicular.

Para encontrar sólo cuatro vértices podemos apoyarnos en la función CV_NEXT_SEQ_ELEM.

Para tener un mejor acierto en la localización de un rectángulo que contenga la placa vehicular se ha tomado ciertas consideraciones para el mencionado objetivo; pues las medidas de las placas vehiculares en Ecuador siguen ciertos estándares de fabricación.

En este proceso la característica que resultó de especial interés es el hecho de que la medida de la placa vehicular por estándar su alto tiene que ser un tercio en relación a su ancho; esta característica establece un filtro bastante bueno de clasificación ya que podemos obviar a los rectángulos que no cumplan con esta condición.

3.5.3. Coordenadas De Imagen

Una vez encontrada la región en donde se encuentra la placa vehicular se ha procedido a extraer las coordenadas de la misma. Para esto se debe considerar que ya tenemos la posición de los vértices y los podemos alojar en una posición de memoria con la función “memcpy”.

Considerando los cuatro vértices encontrados la tarea se centra en encontrar el vértice superior izquierdo y de esta manera con la información obtenida tener un alto y alto en la imagen.

3.5.4. Recortar Placas

Se ha procedido en esta etapa a recortar la porción de imagen que contiene la placa gracias a las coordenadas obtenidas en el proceso anterior. Cabe resaltar que para el recorte y una mayor precisión de detección siempre se recorta la placa menos la parte superior en la que se encuentran las letras de nuestro país “ECUADOR”” por lo tanto se procedió a recortar la

imagen menos el treinta por ciento de alto la cual corresponde a las letras antes mencionadas. El mismo proceso se lo realiza con el ancho ya que por lo regular la imagen consta de bordes no deseados que podrían ser interpretados como una letra.

La función que nos permite recortar una porción de imagen es cvSetImageROI; la cual toma por parámetros una coordenada (x,y) seguido de un ancho y alto de recorte tomando en cuenta las consideraciones mencionadas.

Con el objetivo de tratar con una medida estándar se cambió de tamaño a la placa encontrada para ello se puede fácilmente llamar a la función cvResize , la misma que puede ampliar o reducir la porción de imagen a nuestra conveniencia. Lo interesante de esta función es el hecho de que nunca pierde calidad en la fotografía y al proceder a reducir de tamaño de hecho aumentamos la cantidad de puntos en la misma mejorando su calidad.

3.6. RECONOCIMIENTO DE CARACTERES OCR

El Reconocimiento Óptico de Caracteres (OCR) nos permite identificar automáticamente símbolos o caracteres que pueden pertenecer a un determinado alfabeto, a partir de la porción de imagen recortada en etapas anteriores se procede a extraer los caracteres contenidos. De tal forma que se puede interactuar mediante filtrado posterior.

Existen diversos algoritmos que permiten hacer OCR apoyándonos en OpenCv como base; sin embargo este proceso puede involucrar esfuerzo innecesario ya que para conseguir este fin se debe recopilar una "Biblioteca de imágenes" con el objetivo de que OpenCv encuentre equivalencias entre ellas. Al hablar de "Biblioteca de imágenes" hablamos de recopilar una gran cantidad de posibles imágenes que pueden aparecer en las placas vehiculares. Para tener una muestra que arroje resultados bastante efectivos se debería a recopilar por lo menos cien modelos de posibles formas de cada letra. Por lo tanto requeriremos de por lo menos tres mil setecientos archivos de imagen que contengan las posibles formas de los caracteres.

Se ha podido simplificar este proceso gracias a "GOCR". El cual no es más que un "OCR" de libre distribución que facilita en gran medida nuestro trabajo ya que puede ser llamado desde nuestra aplicación pasándole como parámetro la ubicación de la imagen a reconocer.

```
gocr -C [A-Z][0-9] resultado.jpg
```

Lo interesante de "GOCR" es que podemos pasar varios argumentos antes de indicarle la ruta de la imagen; en nuestro caso hemos añadido una pequeña expresión regular con el objetivo que sólo reconozca caracteres comprendidos entre el alfabeto castellano y que estén en mayúscula; así también sólo números.

3.6.1. Filtrado OCR y envío a BD

Al ya tener un resultado por parte del proceso de OCR, obtenemos la información como datos y no como imagen lo cual resuelve muchos problemas que pueden surgir al desear almacenar esta información. Si pensamos en que se enviará a la base de datos texto y no imágenes alivianamos en gran medida la presión que puede tener el servidor por almacenar demasiadas imágenes.

Una ventaja de mayor importancia es que en esta etapa se ha procedido a desechar información inconsistente o falsa que pueda arrojar el OCR.

3.7. ESTUDIO COMPARATIVO

En el estudio comparativo se analizan el número de aciertos y tiempo de cada algoritmo, obtenidos en cada imagen, y se compara los resultados que varían según el reconocedor que se esté usando, Sobel o Canny. La cantidad de aciertos es uno de los factores más importantes que incide en la eficiencia de los algoritmos con lo cual nos permite saber la confianza que proporciona al sistema de identificación de placas de automóviles.

Luego de observar los resultados se procede a considerar el algoritmo de borde más adecuado para un exitoso sistema de reconocimiento.

Para la comprobación de la Hipótesis "*El estudio comparativo de los algoritmos SOBEL y CANNY de reconocimiento de bordes conlleva a la selección del más eficiente para la implementación de un prototipo que mejorará la identificación de automóviles en la ESPOCH.*", se procedió a evaluar los algoritmos antes mencionados en la detección de bordes de una imagen, se realiza el análisis con las variables examinadas para de esta forma encontrar el detector más adecuado para la implementación del sistema de identificación de automóviles.

3.7.1. Determinación de las Variables a ser Analizadas

Las variables que se han identificado para el análisis de comprobación de la hipótesis son:

- **Variable Independiente:**

Análisis comparativo de los algoritmos SOBEL y CANNY para la implementación del sistema de reconocimiento de placas de vehículos.

Definición: Análisis de funcionamiento y ventajas de los algoritmos SOBEL y CANNY de la biblioteca OpenCv en la identificación de bordes de una imagen.

- **Variable Dependiente:**

Eficiencia

Definición: Es la selección del algoritmo más adecuado para la implementación de un prototipo de reconocimiento de placas de vehículos.

3.7.2. Métodos y Técnicas Utilizadas

En el siguiente apartado se detallan los métodos y técnicas utilizados en el desarrollo de este proyecto.

3.7.2.1. Métodos

En el presente trabajo se han utilizado los siguientes métodos:

a) **Método Deductivo**

En el método deductivo, se suele decir que se pasa de lo general a lo particular, de forma que partiendo de unos enunciados de carácter universal y utilizando instrumentos científicos, se infieren enunciados particulares [7].

A partir de la tecnología OpenCv, que es una biblioteca que permite el reconocimiento de imagen se utiliza las librerías de Sobel y Canny para la identificación de bordes, mediante el algoritmo seleccionado en el estudio comparativo se procede a realizar el prototipo de identificación de placas en automóviles.

b) Método Comparativo

El método comparativo es el procedimiento de comparación sistemática de objetos de estudio que, por lo general, es aplicado para llegar a generalizaciones empíricas y a la comprobación de hipótesis [7].

La presente investigación no se limita al estudio de una sola herramienta sino al estudio comparativo entre dos métodos de identificación de borde en imágenes para poder determinar el más eficiente.

3.7.2.2. Técnicas Documental

Permite la recopilación de información para enunciar las teorías que sustentan el estudio de los fenómenos y procesos. Incluye el uso de instrumentos definidos según la fuente documental a que hacen referencia [7].

Entre las fuentes utilizadas están:

Primaria: Se utilizará a la observación como técnica.

Secundarias: Se tomará información de medios como Internet, revistas y libros.

3.7.3. Operacionalización de las variables

En las siguientes tablas se presentan la operacionalización conceptual y metodológica de las variables de acuerdo a la hipótesis:

VARIABLE	TIPO	INDICADORES	TÉCNICA	FUENTE DE VERIFICACIÓN
V1. Análisis comparativo de los algoritmos SOBEL y CANNY para la implementación del sistema de reconocimiento de placas de vehículos.	Independiente	I1. Funcionamiento de los algoritmos SOBEL y CANNY. I2. Ventajas de los algoritmos SOBEL y CANNY.	Recopilación de información Observación	Información bibliográfica(libros, internet)

Tabla III- II. Operacionalización Metodológica de la Variable Independiente

VARIABLE	TIPO	INDICADORES	INDICE	INSTRUMENTO
V2. Eficiencia	Dependiente	I3. Tiempo de procesamiento. I4. Aciertos.	Tiempo del proceso medido en segundos. Cantidad de aciertos.	Pruebas Pruebas

Tabla III- III. Operacionalización Metodológica de las Variables Dependientes

3.7.3.1. Descripción de variables con sus respectivos indicadores

Cada indicador posee su respectivo índice, para la medición del indicador de aciertos se ha elegido una escala de 0 a 100 %, mientras que el indicador de tiempo de procesamiento se medirá en segundos.

V1. Variable Independiente: Análisis comparativo de los algoritmos SOBEL y CANNY para la implementación del sistema de reconocimiento de placas de vehículos.

Indicadores

I1. Funcionamiento de los algoritmos SOBEL y CANNY.

El funcionamiento consiste en conocer los criterios usados en cada algoritmo para la detección de contornos.

I2. Ventajas de los algoritmos SOBEL y CANNY.

Son las consideraciones que se toman en cuenta al momento de elegir el algoritmo adecuado para el sistema de identificación de placas de automóviles.

V2. Variable Dependiente: Eficiencia

Indicadores

I3. Tiempo de Procesamiento. Es el tiempo que requiere el algoritmo para realizar los diferentes procesos.

Indicadores

I4. Aciertos. Se refiere al porcentaje de éxito que tiene el algoritmo en la entrega de la placa, si la placa es encontrada y recortada correctamente se considera como éxito, lo contrario se denomina como fracaso. Para llevar una identificación de placas idónea es necesario probar el éxito del algoritmo considerando que el 100% corresponde a los 400 vehículos que fueron tomados como muestra y que hayan pasado la prueba favorablemente.

$$\%\ \text{de Aciertos} = \frac{\text{Número de fotografías detectadas correctamente}}{\text{Total de fotografías evaluadas}} \text{x } 100\ \%$$

(a)

(b)

Figura III- 16. Ejemplo de resultados de localización. (a) Detección exitosa y (b) Distorsión de placa.

El resultado de este indicador se lo puede clasificar como:

Éxito: Son aquellas fotografías en la que la placa fue detectada y recortada correctamente.

Fracaso: Son aquellas fotografías en la que la placa no fue detectada, o se obtiene una placa con caracteres distorsionados.

3.7.3.2. Descripción de los datos de entrada

Las imágenes de entrada fueron tomadas según la distancia y ángulo de inclinación, especificados en el apartado 3.1.1, en horarios de 9:00 am hasta 17:30 pm en donde se obtiene condiciones de iluminación adecuadas para el sistema.

Otro aspecto de consideración es el deterioro encontrado en algunas placas de autos que dificultan la localización de la misma y el posterior reconocimiento de caracteres.

Se utilizaron 400 fotografías para la etapa de pruebas, con las características anteriormente señaladas.

Para el análisis del tamaño de la muestra se ha estimado dentro de un error de 5%, a un nivel de confianza del 95%.

$$n = \frac{(z^2 x 0.25)}{\text{error muestral}^2} = \frac{(2^2 x 0.25)}{0.05^2} = 400 \text{ fotografias}$$

Donde

n es el tamaño de la muestra.
z es el nivel de confianza, z = 2 correspondiente a un nivel de confianza del 95%.

3.7.3.3. Procesamiento de información e interpretación

Para conocer cual algoritmo de borde se presenta como la solución más adecuada para la variable independiente se calificara de forma cualitativa y cuantitativa.

V2. Variable dependiente: Eficiencia

	I3. Tiempo de Procesamiento
Con algoritmo Sobel	37 ms
Con algoritmo Canny	40 ms

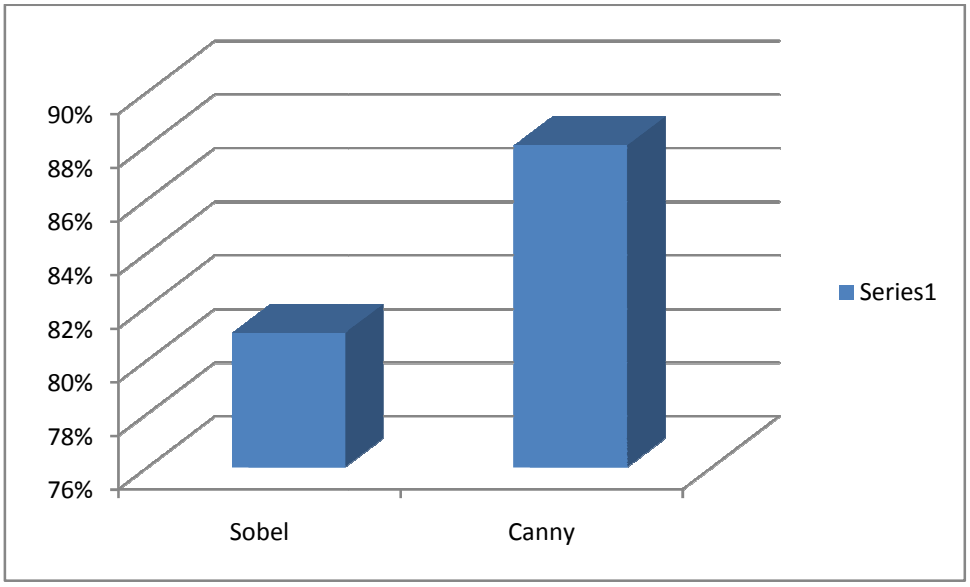


Figura III- 17. Resultado de tiempo de procesamiento.

Interpretación

Según los resultados el algoritmo de Sobel presenta un menor tiempo en el procesamiento de respuesta debido a su sencillez, ya que este aplica métodos de diferencias finitas basadas en la primera derivada, sin embargo se considera que el tiempo obtenido en ambos algoritmos es satisfactorio para el sistema a implementar.

V2. Variable dependiente: Eficiencia

	I4. Aciertos
Con algoritmo Sobel	81%
Con algoritmo Canny	85%

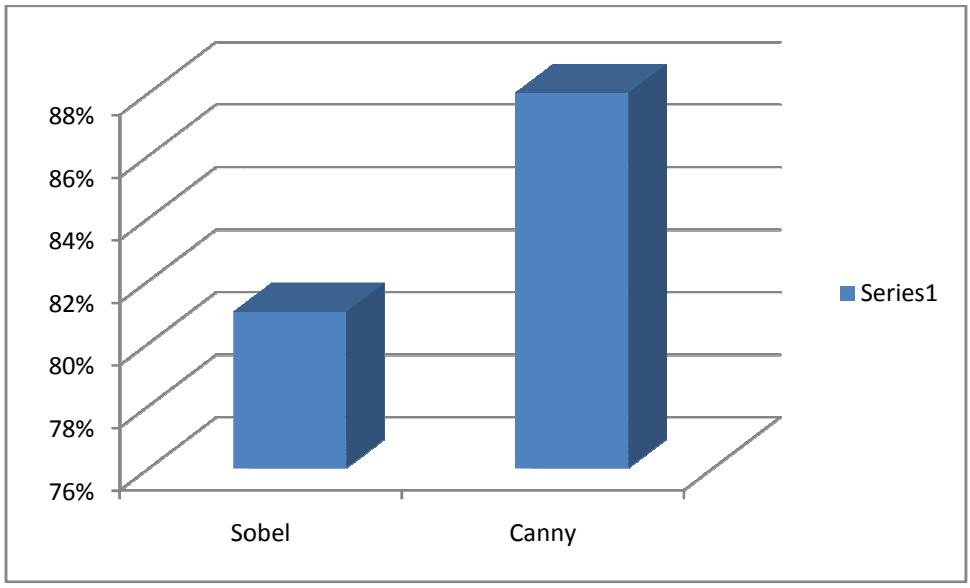


Figura III- 18. Resultado de Aciertos.

Interpretación

Experimentalmente se ha encontrado que los dos operadores tienen medidas similares, y su respuesta se va deteriorando de acuerdo a la cantidad de ruido en la imagen. En la figura anterior se muestra el resultado de aciertos.

3.7.3.4. Análisis de resultados

El análisis de los resultados es fundamental para establecer la mejor alternativa para la siguiente etapa del proyecto que es la implementación. Los métodos de localización tienen diferentes formas de buscar la placa es así que si la imagen de entrada cumple con las especificaciones que exige cada algoritmo este tendrá su mayor rendimiento, pero en situaciones reales estas condiciones difícilmente podrían cumplirse. Como se explicó anteriormente las imágenes de entrada fueron tomadas de forma aleatoria y las mismas fueron utilizadas para los dos detectores.

Con el algoritmo de Canny se logró un porcentaje de éxito de 85% que es un buen resultado. De las pruebas realizadas se detectaron 340 placas correctamente y 60 fallidas, los errores detectados se dieron en imágenes con pronunciados ángulos de inclinación y las placas de automóviles en estado deteriorado, ya que este método depende mucho de la calidad de la imagen. Este algoritmo toma un tiempo considerable en la entrega resultados de 40ms debido al procesamiento que garantiza un mayor éxito en el reconocimiento.

El operador Sobel presenta la ventaja que es fácil y rápido de computar por lo que obtuvimos tiempos de respuestas cortos sin embargo, la rapidez de respuesta puede afectar el nivel de acierto en la obtención de la localización de la placa. La detección de bordes de Canny es ligeramente superior en cuanto a detecciones exitosas, ya que atenúa ciertos sectores en donde no hay bordes sino más bien ruido, pero de todos modos es mínima la diferencia.

El algoritmo Canny se presenta como la solución más adecuada luego de realizar un estudio comparativo entre algoritmos de reconocimiento de borde, de esta manera se demuestra que es el más eficiente para la implementación del prototipo de identificación de placas de autos en la ESPOCH.

CAPÍTULO IV

DESARROLLO DE LA APLICACIÓN

Para el desarrollo de este proyecto fue de vital importancia la definición de una metodología que proporcione una guía acerca de cómo realizar este proceso para reducir la complejidad inherente de este tipo de desarrollos.

El objetivo de este documento es presentar la metodología usada para el modelado del negocio, análisis de requerimientos, análisis y diseño, implementación y pruebas en el desarrollo del software LJPlate.

4.1. Análisis Preliminar

En este apartado se ha realizado un análisis previo para la obtención de las características que debe cumplir el software así como la factibilidad para el desarrollo del mismo.

4.1.1. Visión General

Se ha realizado un análisis conjuntamente con los posibles usuarios del sistema para el establecimiento de los movimientos (Ingresos y consultas) que intervienen en el proceso de entrada de una placa de automóvil, y se ha logrado identificar la meta fundamental que busca implementar una solución la cual se propone desde la perspectiva del equipo de desarrollo frente a las diferentes características que se ha logrado abstraer de quienes participarán e interactuarán estrechamente con la aplicación.
El sistema funcionará dentro del entorno de Ubuntu, además se utilizaran herramientas que faciliten el desarrollo de una aplicación con las características antes mencionadas y que

permitan cubrir la mayoría de las necesidades expresadas por los usuarios finales del sistema.

4.1.2. Perspectiva del Producto

El sistema propuesto contará con todo lo necesario para el reconocimiento eficiente de las placas de automóviles que ingresan a la ESPOCH, los mismos que son necesarias por parte del Departamento de Mantenimiento y Desarrollo Físico y Jefe de Seguridad para poder realizar consultas y observar reportes con el fin de obtener un servicio que ayude al control de la seguridad en la institución y a futuras investigaciones, los mecanismos que serán implementados incluirán el más alto grado de facilidad en el trabajo, para que los usuarios puedan adaptarse rápidamente al mismo.

4.1.3. Funciones del Producto

El sistema LJPlate en términos generales, deberá proporcionar soporte en el control de reconocimiento de placas de automóviles. A continuación se detallarán estas tareas y como serán soportadas por el sistema:

Inicio de Sesión del Usuario

Permite que el usuario ingrese su nombre de usuario y contraseña que serán validados como medida de seguridad, de esta manera obtiene acceso a las tareas que les son permitidas.

Los Administradores pueden realizar:

- Creación de usuarios.
- Ingresar y modificar entradas de la tabla de placas.
- Consultas y lectura de reportes.

Gestión del Sistema.

El sistema se encargará de las siguientes funcionalidades:

- Permitir al usuario conectarse de manera segura a su estación de trabajo.
- Dar de alta automáticamente la matrícula, fecha y hora en la que el vehículo ingreso a la institución.

4.1.4. Beneficios

Entre los beneficios que brinda este producto se puede describir los siguientes:

- Beneficios tangibles.
    - Mayor productividad
    - Menor tiempo de respuesta
    - Eliminación de tareas redundante
- Beneficios intangibles.
    - Mayor control en seguridad.
    - Mejora en el ambiente de trabajo.
    - Mejor servicio a los usuarios.
    - Mejor imagen de la institución.

4.1.5. Análisis de Requerimientos

En esta etapa se especificó las condiciones que debe cumplir el programa, para lograr esto se usó la definición de requerimientos. Además se realiza una estimación del costo y limites del sistema LJPlate.

4.1.5.1. Especificación de requerimientos software (SRS)

La implementación LJPlate es un sistema de control de Reconocimiento de Placas de Automóviles para el Departamento de Mantenimiento y Desarrollo Físico y Jefe de Seguridad, en el cual se obtiene información acerca del Parque Automotor que ingresa en la ESPOCH. Este sistema permitirá registrar las placas de todos los automóviles que ingresan a la institución logrando una documentación de forma fiable y automática, además de garantizar el acceso y permanencia de la información con una robusta base de datos.

El desarrollo del sistema LJPlate tiene funcionalidades como ingreso, modificación, búsqueda y listado total de las placas obtenidas por fechas, además de visualización del contenido por investigación seleccionada.

4.1.5.2. Características de los Usuarios

LJPlate presentará una interfaz sencilla y de fácil manejo por lo que se requiere del usuario únicamente el conocimiento básico del manejo del computador.

Los siguientes usuarios que han sido identificados necesitan poseer las siguientes características:

Administrador.- Estará enfocado en la administración, debe tener amplios conocimientos en computación y redes para poder solucionar cualquier problema con la red corporativa y brindar el soporte adecuado a los clientes, además saber y estar capacitado en el manejo y funcionamiento del sistema LJPlate.

Usuario.- Debe tener conocimiento en el manejo y funcionamiento del sistema LJPlate.

4.1.5.3. Limitaciones Generales

LJPlate será implementado utilizando Qt Creator como herramienta de desarrollo y GOCR, que convierte el texto en imágenes a archivos de texto. Por lo tanto la aplicación será desplegada en ordenadores cuyo sistema operativo se base en Linux ya que GOCR es una herramienta propia de Linux, además se utilizará MySQL como DBMS.

El sistema será desarrollado dentro de una arquitectura que permita implementar las siguientes características:

- Sea escalable y confiable
- Proporcione amplios niveles de disponibilidad y robustez.
- Sea de fácil instalación en las estaciones de trabajo
- Permita un fácil mantenimiento.

4.1.5.4. Supuestos

Se asume que las políticas detalladas en este documento serán permanentes una vez que sean aprobados por las autoridades y personas directamente implicadas en el sistema.

Cualquier petición de cambio debe ser aprobada por todas las partes y gestionada por el grupo de desarrollo.

4.1.5.5. Requerimientos Específicos

Especifica las características y funcionalidades que debe cumplir el software para satisfacer la tarea de reconocimiento de placas.

a) Requerimientos Funcionales

Validar usuarios.

- REQ 1: El sistema validará el inicio de sesión y password ingresados por el usuario, esta verificación le permitirá interactuar con la aplicación y ejecutar las tareas que le son permitidas.

Administración del sistema LJPlate

- REQ 2: El sistema podrá ser controlado por el Administrador, el mismo que tendrá la facultad ingresar y modificar placas, además de consultar reportes y realizar búsquedas personalizadas.
- REQ 3: El sistema permitirá dar de alta y modificar la placa de un vehículo.
- REQ 4: El sistema permitirá al usuario realizar el reconocimiento de la placa desde una cámara web, o desde un archivo en formato .jpg, .avi, .mjpeg. los campos que se añaden son el número de matrícula, el tipo de categoría al que pertenece el vehículo, imagen de la placa capturada, fecha y hora.
- REQ 5: El sistema permitirá visualizar el total de las placas reconocidas, dada la fecha y tipo de categoría.
- REQ 6: El sistema permitirá visualizar los resultados de una búsqueda de placa (matricula y tipo de categoría) dada su matrícula.
- REQ 7: El sistema permitirá ingresar usuarios.

Desarrollo

- REQUERIMIENTO FUNCIONAL 1 Inicio de Sesión.

  INTRODUCCIÓN
  Inicio de Sesión de Usuarios, Iniciar sesión.

ENTRADAS

- Usuario
- Password

PROCESO

LJPlate permitirá el ingreso mediante la validación de nombre del usuario y password.

SALIDAS

Validación Correcta
Ventana de Ingreso al Sistema.

- REQUERIMIENTO FUNCIONAL 2 (Validar usuarios)

INTRODUCCIÓN

Validar usuarios.

ENTRADAS

Inicio de sesión y password ingresados por el usuario para el uso del sistema LJPlate.

PROCESO

El sistema validará el inicio de sesión y password ingresados, esta verificación permitirá al sistema otorgar las tareas que le son permitidas.

SALIDAS

Validación Correcta (Ingreso al sistema), caso contrario validación errónea (usuario no existe).

- REQUERIMIENTO FUNCIONAL 3

INTRODUCCIÓN

Dar de alta o modificar placas de automóviles.

ENTRADAS

- ID de la placa.

- Matricula de la placa.
- Categoría
- Imagen de la placa
- Fecha de ingreso del automóvil.

PROCESO
El sistema permitirá al usuario dar de alta o modificar placas de automóviles que ingresan a la ESPOCH.

SALIDAS
Ingreso o modificación correcta de la placa de automóvil.

- REQUERIMIENTO FUNCIONAL 4

INTRODUCCIÓN
Iniciar reconocimiento de placa.

ENTRADAS
Conectar cámara web o examinar el archivo a ser analizado.

PROCESO

El sistema permitirá al usuario elegir entre usar una cámara web o especificar la ruta donde se encuentra el archivo para el posterior reconocimiento de placas.

SALIDAS
Ingreso correcto de las placas.

- REQUERIMIENTO FUNCIONAL 5

INTRODUCCIÓN
Obtener un reporte con los datos de las placas de automóviles ingresadas en el sistema.

ENTRADAS

Categoría.
Fecha.

PROCESO
El sistema permitirá visualizar el total de automóviles.

SALIDAS
Datos de las placas (Matricula, fecha, imagen de placa).

- REQUERIMIENTO FUNCIONAL 6

INTRODUCCIÓN
Búsqueda de una placa automóvil.

ENTRADAS
Número de la matrícula.

PROCESO
El sistema permitirá la búsqueda de una placa de automóvil específica dada su matrícula.

SALIDAS
Datos de una placa de automóvil definida (matricula, categoría)

- REQUERIMIENTO FUNCIONAL 7

INTRODUCCIÓN
Dar de alta usuarios.

ENTRADAS
- Usuario
- Password

PROCESO
El sistema permitirá dar de alta a usuarios para el sistema LJPlate.

SALIDAS
Ingreso correcto de la cuenta de usuario.

b) Requisitos No Funcionales.

Son requisitos que constituyen las características de la aplicación software, se les conoce como la CALIDAD DEL SOFTWARE

Limitaciones software

Los requisitos de software necesarios para la aplicación son:

- Ubuntu 10.04
- G++
- MySql
- QT
- OpenCv

c) Requerimientos de interfaz

Son los elementos que provee el sistema para permitir la interacción entre el usuario y las funcionalidades que tiene el software.

i. Interfaces de Usuario.

Las interfaces de LJPlate serán implementadas en Qt Creater 4, que permite el desarrollo de aplicaciones en C++ de manera sencilla y rápida. La interfaz mostrará una fácil interacción de los usuarios con el sistema mediante el uso de elementos como:

- TextBox (txt)
- Buttons(btn)
- Labels(lbl)
- Forms(frm)
- GroupBox(grb)
- ListString

En este momento se manifestará la funcionalidad de cada uno de los elementos que se mostrarán en pantalla:

- BARRA DE HERRAMIENTAS. Muestra la agrupación de un conjunto de iconos gráficos, los cuales al ser desplegados y elegidos independientemente permiten realizar una tarea específica. Generalmente se encuentran en la parte superior de una ventana.

- TEXTBOX. Son componentes que permiten el ingreso de texto a la aplicación.

- BUTTONS. Se constituyen como componentes que al ser accionados mediante un clic sobre ellos, realizan una determinada acción previamente programada para ellos.

- LABELS. Permiten la salida de mensajes provenientes de la aplicación y que necesiten ser visualizados por los usuarios.

- PRESENTACIÓN DE MENSAJES DE ERROR. Notificarán al usuario o cliente que realizó una acción errónea.

- Forms(frm)
  Pantallas interactivas que permiten la presentación, introducción y respuesta de información, especialmente utilizada en entornos de ventanas.

- ListString
  Componente que permite la adición de Arreglos de Strings de forma dinámica.

ii. Interfaces Hardware.

El sistema LJPlate necesita de los siguientes componentes hardware para su funcionamiento:

- La Visualización gráfica, mostrada en el monitor.
- Conexión directa a la impresora, teclado, mouse haciendo uso de los diferentes puertos, dando la opción de crear un reporte, realizar consultas y la entrada de datos.
- Tarjetas de Red compatibles con fastethernet
- Cableado Estructurado de acuerdo a la norma IEEE 802.3
- Computadores clientes para los usuarios.

iii. Interfaces de Software.

El sistema propuesto se basará en los siguientes componentes software:

- MySQL
- Qt Creator 4.
- SO Ubuntu 10.04

4.1.6. Factibilidad del proyecto

La factibilidad del proyecto permite conocer la disponibilidad de recursos necesarios para el desarrollo de la presente tesis.

4.1.6.1. Factibilidad Técnica

En la Tabla IV- I y Tabla IV- II se indica el hardware y software requerido para el buen funcionamiento del sistema.

Cantidad	Descripción	Estado
1	**Computadora personal** -2 GB de RAM -250 GB de Disco Duro -Procesador AMD Athlon X2 2.8 Ghz. - Mainboard Intel	Apto para el proyecto
1	I**mpresora** -Epson Photo T50	Apto para el proyecto
1	**Cámara** Toshiba Ik-wb21a	Apto para el proyecto

Tabla IV- I. Hardware requerido

Nombre	Descripción	Estado
Ubuntu 10.04	SO	Legal
C++	Lenguaje de Programación	Legal
MySql	Base de Datos	Legal
QT	Entorno de Desarrollo	Legal
OpenCv	Librerías para tratamiento de fotografías.	Legal

Tabla IV- II. Software requerido

A continuación se describen las principales responsabilidades de cada uno de los puestos en el equipo de desarrollo de LJPlate:

Nombre	Función
Analista	Captura, especificación y validación de requisitos, interactuando con el cliente y los usuarios mediante entrevistas. Elaboración del Modelo de Análisis y Diseño. Colaboración en la elaboración de las pruebas funcionales y el modelo de datos.
Programador	Construcción de prototipos. Colaboración en la elaboración de las pruebas funcionales, modelo de datos y en las validaciones con el usuario

Tabla IV- III. Recursos humanos requeridos

4.1.6.2. Factibilidad Operativa

Es importante una apropiada interacción entre la aplicación y los usuarios de esta manera se asegura el uso adecuado de la herramienta, para lograr este fin el personal que se muestra en la *Figura IV- 1* se sugiere sea sometido a capacitación.

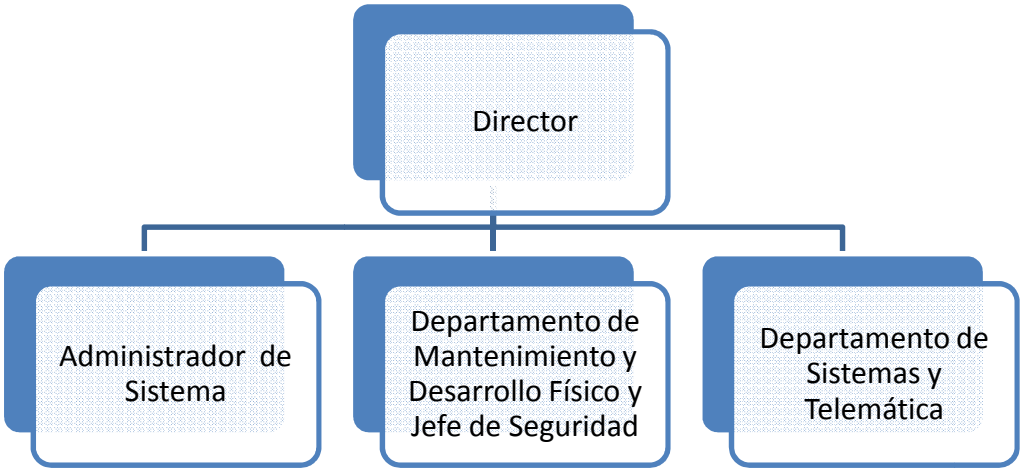


Figura IV- 1. Factibilidad Operativa

Capacitación

Capacitar al Director, Administrador del Sistema, Departamento de Mantenimiento y Desarrollo Físico y Jefe de Seguridad, Departamento de Sistemas y Telemática acerca del funcionamiento del nuevo sistema informático.

4.1.6.3. Factibilidad Económica.

En la Tabla IV- IV se indica los costos implicados en el proyecto.

COSTOS DE DESARROLLO	Costo Personal	1	Analista	1450.00
		1	Programador	1450.00
	Total costo de personal			2900.00
	Costos de Hardware y software para desarrollo	En lo referente a costos de adquisición de equipos será responsabilidad de la ESPOCH.		
	Total de Hardware y Software			
	Costos de suministros para el desarrollo			150.00
Total costo de Desarrollo				3050.00
COSTO DE INSTALACIÓN DEL SISTEMA	Costo de personal de instalación			150.00
	Costo de capacitación a los usuario directos			100.00
Total de costo de instalación del sistema				250.00
TOTAL				3300.00

Tabla IV- IV. Factibilidad Económica

La duración del proyecto es de 1 año según la aplicación del Modelo matemático COCOMO. El salario del personal es un costo referencial tomando en consideración un sueldo aproximado del analista y programador en nuestro país.

Personal	Salario mensual	Total en duración del proyecto
Analista	120.83	1450
Programador	120.83	1450

Tabla IV- V. Costos de personal

4.1.6.4. Cronograma de Actividades

Para la planeación del proyecto se utilizó la herramienta Microsoft Office Project que permite la organización de las actividades en un horizonte de tiempo. El inicio del proyecto fue el 1 de Marzo 2011 y se ha establecido la finalización el 30 de Marzo del año en curso cumpliendo el correspondiente cronograma de actividades.

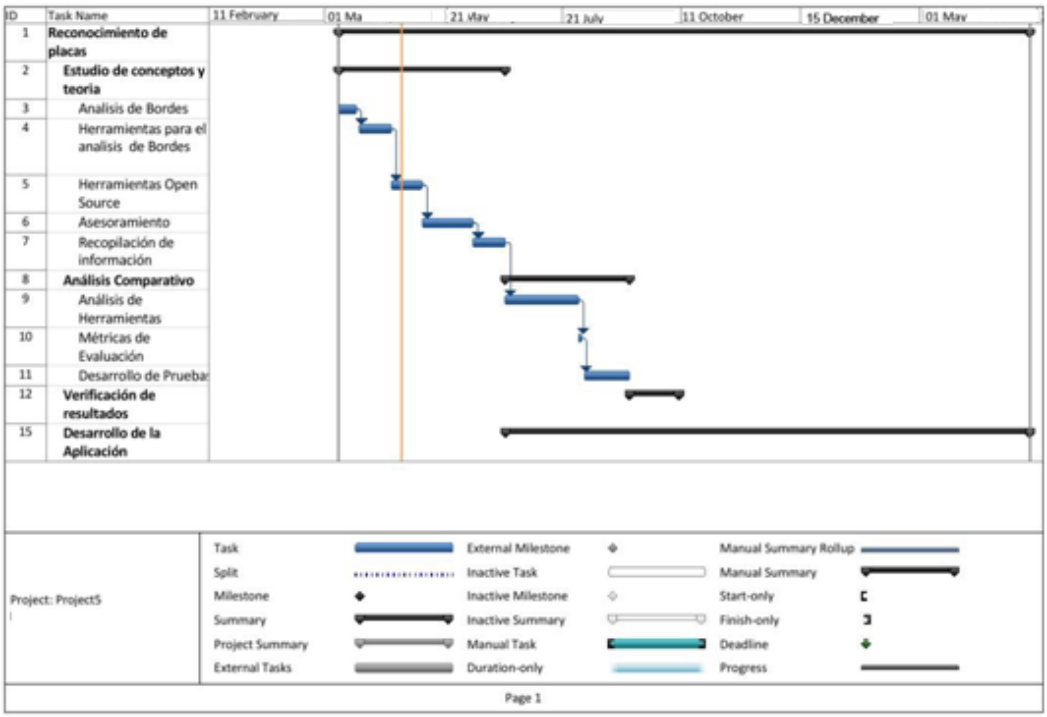


Figura IV- 2. Cronograma de Actividades

4.2. ANÁLISIS Y DISEÑO

4.2.1. Modelado del negocio

A continuación se presentan los modelos definidos en RUP como modelo del negocio: modelo de casos de uso del negocio y modelo de objetos del negocio.
Para realizar la tarea de reconocimiento de placas se interactúa con dos elementos externos, entre los que se identifica el vehículo (objeto que contiene la placa que se dará de alta) y el usuario (persona que solicita operaciones)

4.2.1.1. Modelo de Casos de Uso del Negocio

El modelo de Casos de Uso describe las funciones del sistema y los actores que hacen uso de ellas. Esta herramienta es un apoyo para la explicación de las funcionalidades que presenta el software.

Identificador caso de uso	I-SESION
Nombre del caso de uso	Iniciar Sesión
Actores	Usuario
Propósito	Realizar una operación de inicio de sesión de una cuenta del sistema LJPlate
Visión General	Un usuario del sistema introduce su nombre y contraseña, y solicita la operación de autenticación, el sistema valida los datos ingresados y permite iniciar sesión al usuario.
Tipo	Primario
Referencias	Funciones: R1
Curso típico de eventos	
Acciones del Actor (Usuario)	Respuesta del Sistema
Usuario introduce su nombre y contraseña	Validación de datos

Ingresa al sistema.	En espera
Cursos Alternativos	
El usuario o la clave son incorrectos. Se indica el error y se cancela la operación	

Tabla IV- VI. Plantilla de caso de uso Iniciar Sesión en formato extendido

El Diagrama de Casos de Uso permite una mejor interpretación de la información contenida en la plantilla anterior. El diagrama que se presenta a continuación pertenece al caso de uso Iniciar Sesión.

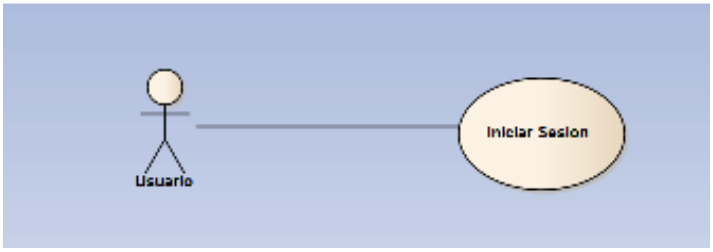


Figura IV- 3. Diagrama de Casos de Uso del Negocio 1

El resto de plantillas y diagramas de casos de uso se encuentran en el Anexo B.

4.2.1.2. Modelo del Dominio

Los modelos de objetos del dominio están asociados a cada uno de los casos de uso del negocio. El caso de uso para el cual se desarrolló el modelo de objetos, fue el del caso de uso del negocio “ingreso de placas” por ser la funcionalidad más importante.

Figura IV- 4. Ingreso de Placas

4.2.2. Modelo Relacional

En este modelo los datos se almacenan en relaciones, y cada relación es un conjunto de datos. A continuación se presenta el diagrama del modelo relacional (Figura IV-5) donde se muestran un conjunto de tablas que representan las relaciones definidas en el proyecto.

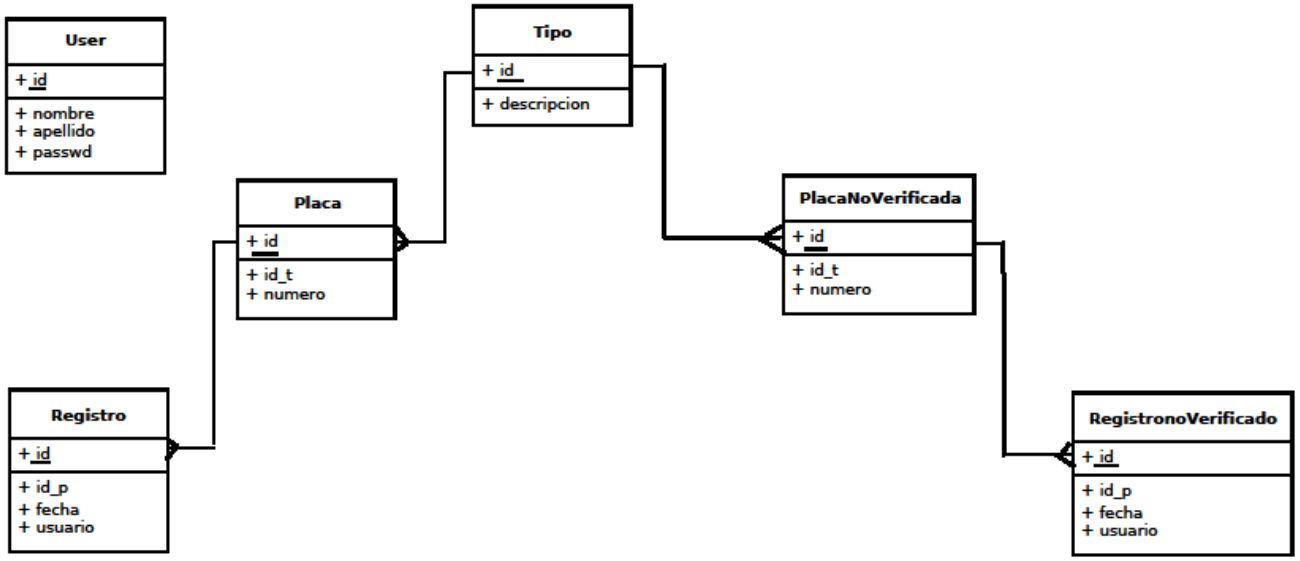


Figura IV- 5. Modelo Relacional

4.2.3. Diagramas de Clases

En la Figura IV-6 se muestra el diagrama de clases, este representa las clases que se utilizan dentro del sistema LJPlate y las relaciones que existen entre ellas, junto con sus métodos y atributos.

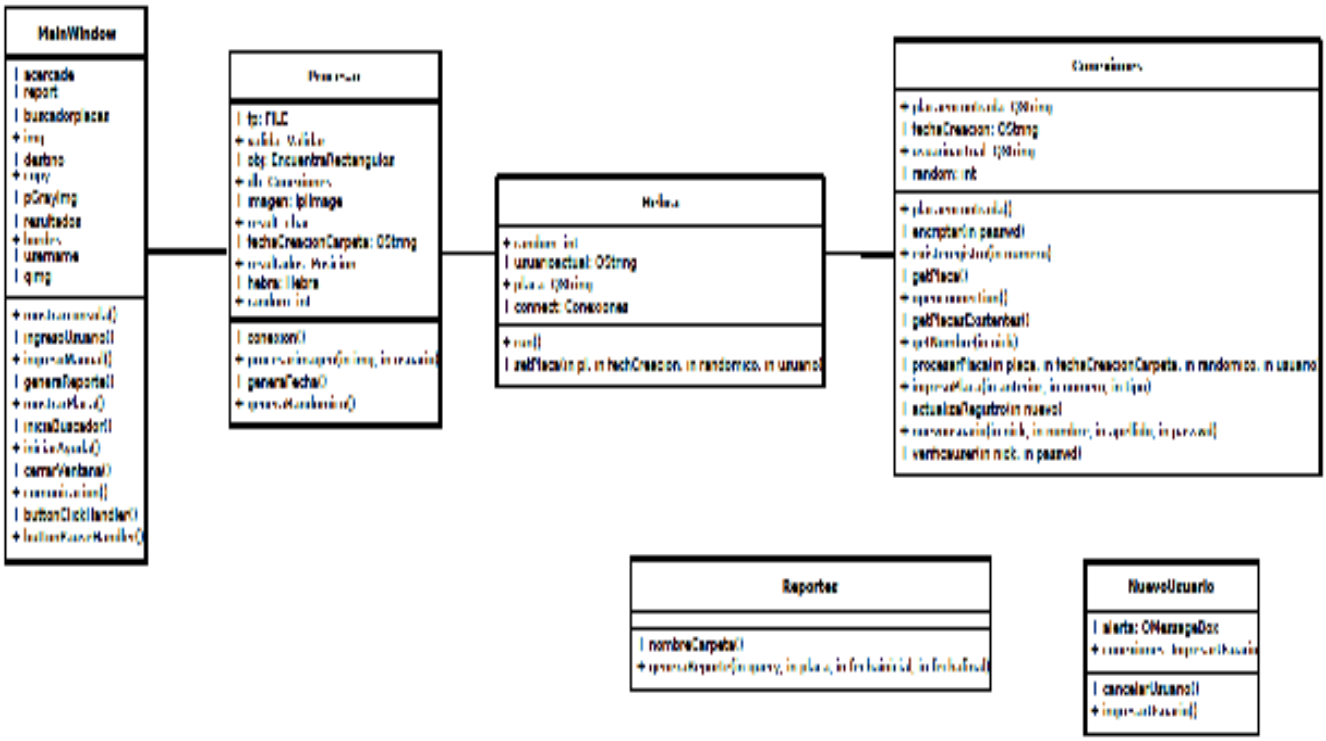


Figura IV- 6. *Diagramas de Clases*

4.2.4. Codificación

Realizado el diseño del sistema LJPlate se procedió a la generación de código. El lenguaje de programación utilizado es C++ y como compilador G++, además se realiza llamadas a librerías y métodos de OpenCV para el tratamiento de las imágenes.

Posteriormente se detalla los bloques de código más relevantes del programa.

En las siguientes líneas de código podemos apreciar la instanciación de la clase base para el reconocimiento de la placa vehicular, antes de procesar cada fotograma es necesario conocer el origen del mismo; por este hecho se creó una estructura de selección múltiple la misma que se encarga de verificar el origen de los fotogramas.

```
switch(tiporigen){
  case 1:{
  camera=cvCaptureFromFile( li.absoluteFilePath().toUtf8().data());
  IplImage *frame = cvQueryFrame(camera);

 while(activo&&(frame!=NULL)){
     QObjectCleanupHandler *cleaner=new QObjectCleanupHandler();
       img =cvCloneImage(frame);
       proimg=new Procesar();
       proimg->procesarimagen(frame,this->user);
       this->mostrarimagen(proimg->imagen);

       cvvWaitKey(10);
       if((proimg->conexion.size()>0)&&(proimg->conexion.size()<=6)){
           this->mostrarConsola(proimg->conexion);
           this->mostrarplaca(proimg->conexion);
       }
       frame = cvQueryFrame(camera);
       cvReleaseData(img);
       cleaner->add(proimg);
       cleaner->clear();
       cleaner->~QObjectCleanupHandler();
   }
 ui->Pausar->setDisabled(true);
 ui->Empezar->setDisabled(false);
  break;
}
```

Figura IV- 7. *Archivo mainwindow.cpp*

Es necesario resaltar la importancia que tiene manejar un procesamiento en hebras; por el hecho de mejorar sustancialmente el rendimiento de nuestra aplicación al realizar varias tareas en un mismo tiempo de ejecución. El uso de hilos en la presente tesis representa una herramienta de vital importancia pues se los empleó en la generación de carpetas dinámicas para el alojamiento de imágenes, registro de la placa detectada en la base de datos. El siguiente código es el encargado de realizar las tareas antes descritas:

```
1  void Procesar::procesarimagen(IplImage *img,QString usuario){
2      this->valida= new Validar();
3  /*Procedemos a detectar placas*/
4  obj= new EncuentraRectangulos();
5      obj->procesar(img,algoritmo);
6      int incremento=0;
7      resultados obj->getPosiciones();
8      incremento= (resultados.alto*30)/100;
9      copy= cvCloneImage(img);
10     if(((resultados.x+2)> 0)
11         &&((resultados.y+incremento)> 0)
12         &&((resultados.ancho-2)> 0)
13         &&((resultados.alto-incremento)> 0)
14         )
15     cvSetImageROI(copy, cvRect(resultados.x+2,resultados.y+
16     incremento, resultados.ancho-2, resultados.alto-incremento));
17     destination   cvCreateImage
18     ( cvSize(83,24),copy->depth, copy->nChannels );
19     pGrayImg    0;
20     pGrayImg    cvCreateImage(cvSize(83,24), 8, 1);
21     cvResize(copy, destination);
22     cvCvtColor(destination,pGrayImg, CV_RGB2GRAY);
23     cvSaveImage("resultado.jpg", pGrayImg);
```

Figura IV- 8. Archivo procesar.cpp

El código antes expuesto empieza instanciando la clase responsable de validar la placa extraída; lo cual involucra verificar si los caracteres detectados corresponden a una placa ecuatoriana; en el caso de no tener las características básicas como empezar con letras y terminar con tres o cuatro números.

En la línea cuatro se instancia la clase “EncuentraRectangulos” cuyo único objetivo en la aplicación es encontrar todo tipo de rectángulos existentes en el fotograma; posteriormente este resultado es filtrado como se aprecia desde las líneas diez a catorce.

Una vez que el algoritmo determina si es un rectángulo con las características de una placa ecuatoriana se procede a dibujar el mencionado rectángulo y guardar la porción de imagen como se aprecia en la línea veinte y tres.

```
24      /*Aplicacion de algoritmo de reconocimiento
25       de caracteres con imagen temporal*/
26       fp   popen("gocr -C [A-Z][0-9] resultado.jpg","r");
27       fread(result,1,sizeof(result),fp);
28       fcloso (fp);
29       conexion QString(result);
30       conexion conexion.mid(0,6);
31       if(this->valida->verificaPlaca(conexion,0))
32       this->param 0;
33           else if(this->valida->verificaPlaca(conexion,1))
34           this->param 1;
35           else this->param -1;
36       if ((this->param  0)||(this->param  1)){
37       this->generaCarpeta(this->param);
38       hebra new Hebra();
39       this->random this->generaRandomico(1,10000);
40       system(("cp resultado.jpg "+
41       QString::number(this->random)+".jpg").toUtf8().data());
42       hebra->setPlaca(conexion,this->fechaCreacionCarpeta,
43       this->random,usuario,this->param);
44       hebra->start();
45       }else conexion "";
46      /*fin de reconocimiento de caracteres*/
```

Figura IV- 9. Archivo procesar.cpp

Una vez obtenida la posible porción de imagen que contiene la placa; se procede a extraer los caracteres de la imagen por medio de la aplicación del algoritmo de "OCR"; nótese que en la línea veinte y seis se invoca al algoritmo con los dos parámetros el primero denota que buscamos mayúsculas, seguido de una expresión regular la misma que expresa la búsqueda de letras y números; finalmente se indica que se aplique este algoritmo sobre la imagen resultado.jpg la cual es temporal para cada análisis de fotograma.

En la línea treinta y siete se genera una carpeta en el caso de no existir la misma que corresponde a un formato de fecha actual; lo que resulta bastante útil para tener un registro de las placas detectadas tanto con motivos de auditoría como para poder visualizar en un futuro en los reportes gráficos.

A partir de la línea treinta y ocho se recurre a instanciar la clase “Hebra”; la misma que se encarga de realizar todas las tareas pertinentes para clasificar nuestro resultado; decidiéndose en este punto si la placa es correcta o posiblemente correcta permitiendo así almacenar los resultados en la base de datos.

```
47      CvPoint pt1, pt2;
48      pt1.x   resultados.x; //coordenada x
49      pt1.y   resultados.y; //coordenada y
50
51      pt2.x   resultados.x+resultados.ancho; //coordenada x
52      pt2.y   resultados.y+resultados.alto; //coordenada y
53      cvRectangle( img, pt1 , pt2 , CV_RGB(0,0,255), 3, 8, 0 );
54
55 /* Fin dtoccion de rectangulos*/
56
57    if (img->origin) {
58    cvFlip(img);
59    img->origin  0;
60    }
61    if (copy->origin) {
62    cvFlip(copy);
63    copy->origin  0;
64    }
65    if (pGrayImg->origin) {
66    cvFlip(pGrayImg);
67    pGrayImg->origin  0;
68    }
69    if (destination->origin) {
70    cvFlip(destination);
71    destination->origin  0;
```

Figura IV- 10. Archivo procesar.cpp

La porción de código que corresponde al intervalo de líneas cuarenta y siete, cincuenta y tres se encargan de dibujar un rectángulo en nuestro fotograma para ilustrar visualmente la detección de la placa.

El código restante corresponde al manejo de memoria y la liberación de la misma; las líneas comprendidas entre cincuenta y cinco, ochenta y seis destinan su funcionamiento ha borrado de memoria ocupada y destrucción de variables. La clase responsable de ser limpiar todo el uso restante es “QObjectCleanUpHundler” que es instanciada en la línea ochenta y cuatro, ejecutada en la línea ochenta y tres.

```
72      }
73      this->imagen=cvCloneImage(img);
74      cvReleaseData(copy);
75      cvReleaseData(pGrayImg);
76      cvReleaseData(destination);
77      cvReleaseData(copy);
78      cvReleaseImage(&copy);
79      cvReleaseImage(&pGrayImg);
80      cvReleaseImage(&destination);
81      QObjectCleanupHandler *cleaner=new QObjectCleanupHandler();
82      cleaner->add(obj);
83      cleaner->clear();
84      cleaner->~QObjectCleanupHandler();
85
86  }
```

Figura IV- 11. Archivo procesar.cpp

El uso de las distintas alternativas existentes dentro de los actuales lenguajes de programación provee grandes ventajas en el momento del desarrollo; entre ellas destaca el uso de semáforos. Para ilustrar mejor esta idea usted puede imaginar que tiene dos interfaces gráficas completamente diferentes; sin embargo se requiere que la primera interfaz de alguna manera comunique a la segunda que ha ocurrido un evento en especial como por ejemplo que el usuario presionó un botón en especial; al tener interfaces diferentes que responden a eventos diferentes esto representa un gran problema. Sin al tener una forma de comunicar las dos interfaces por medio de una señal nuestro problema se resuelve. El uso de una señal involucra especificar el evento que se espera y enlazar con una clase diferente por medio de la ejecución de un método de la clase que se espera reaccione.

El uso de Señales en nuestra aplicación nos permitió realizar de una forma sencilla el inicio de sesión de los usuarios pues la ventana de login maneja dos conceptos:

1. Autenticación: Permite conocer si un usuario es parte del sistema y comprobar que su contraseña sea correcta.
2. Autorización: Si se concluye que un usuario forma parte del grupo de administradores el siguiente paso es la "Autorización" que indica qué acciones puede ejecutar ese usuario.

Tomando en cuenta lo expuesto si autenticamos a un usuario el siguiente paso lógico es levantar la interfaz de administración y es en donde nuestro problema requiere del uso de "Señales" entre interfaces que terminan siendo dos "Clases" diferentes pero que se comunican por este método.

En las líneas venideras se aprecia el uso de Señales entre interfaces gráficas:

```
1  #include <QtGui/QApplication>
2  #include "mainwindow.h"
3  #include "loqueowindow.h"
4  #include "reportesfrm.h"
5  #include "buscadorplacas.h"
6  #include "opencv/cv.h"
7  #include "opencv/highgui.h"
8  int main(int argc, char *argv[])
9  {
10     QApplication a(argc, argv);
11     MainWindow w;
12     LoqueoWindow w2;
13     QObject::connect(&w2, SIGNAL(envio(const QString&,const QString&)),
14     &w, SLOT(comunicacion(const QString&,const QString&)));
15     QObject::connect(&w2, SIGNAL(cerrar()),&w, SLOT(cerrarventana()));
16     w2.show();
17     return a.exec();
18 }
19
```

Figura IV- 12. Archivo main.cpp

En el archivo main.cpp entre las líneas uno y siete se encuentra la declaración de librerías pertinentes para proceder a la ejecución de la aplicación. Las líneas trece y quince muestran la utilización y enlace de interfaces mediante señales.

La codificación completa se entrega en el CD que acompaña este trabajo profesional.

4.3. IMPLEMENTACIÓN

En esta sección se muestra los prototipos de interfaces gráficas de usuario diseñadas para la aplicación final, como ejemplo se presenta la pantalla de autenticación.

Figura IV- 13. Autenticación de usuario.

La aplicación dispone de una primera ventana de identificación del usuario, *Figura IV-13*. Sólo usuarios que contengan los datos de autenticación correcta pueden hacer uso de las funciones del sistema.

En el Anexo C se encuentra la información detallada de las pantallas que posee el sistema, así como su funcionalidad respectiva.

4.4. PRUEBAS

Las pruebas nos permiten identificar los posibles fallos existentes en la aplicación de tal forma que se pueda verificar la calidad del producto.

4.4.1. Plan de pruebas

Objetivo de la prueba Detectar errores de programación y funcionamiento en el sistema

Objetos a probar

- Detectar placa
- Reconocimiento de caracteres

Características a probar Funcionamiento de cada uno de los procesos para realizar las tareas definidas.

La prueba se realizará en entorno Ubuntu.

4.4.2. Documento de diseño de la prueba

Procedimiento de pruebas

Las pruebas serán llevadas a cabo de acuerdo a lo descrito en la Especificación del procedimiento de prueba registrándose las anomalías observadas.

Métodos de prueba a utilizar

Se utilizará el método de caja negra, adivinación de errores, para poder así determinar las posibles fallas del sistema.

Criterio para la aprobación de las pruebas
Los criterios para la aprobación de las pruebas se realizarán de acuerdo a la siguiente tabla:

Excelente: Cuando el resultado obtenido luego de realizada la prueba es idéntico al resultado citado en la - Especificación de los casos de prueba.

Muy bueno: Cuando el resultado obtenido luego de realizada la prueba es parecido al resultado citado en la - Especificación de los casos de prueba.
Bueno: Cuando el resultado obtenido luego de realizada la prueba no fue el resultado citado en la– Especificación de los casos de prueba, pero no ha provocado anomalías en el funcionamiento de la herramienta.

Regular: Cuando el resultado obtenido luego de realizada la prueba no fue el resultado citado en la sección - Especificación de los casos de prueba, pero ha provocado anomalías en el funcionamiento del programa.

Malo: Cuando el resultado obtenido luego de realizada la prueba no fue el resultado citado en la sección - Especificación de los casos de prueba, pero ha provocado anomalías en el funcionamiento del programa tales como la salida del sistema o "colgarse".

Ítem	**Objetivo**	**Acción**	**Entrada**	**Resultado Esperado**
1	Ingresar datos erróneos en autenticación	Ejecutar Autenticación	Usuario Contraseña	Se debe informar al usuario datos incorrectos
2	Indicar un origen de video al software con condiciones de luz de tarde	Ejecutar Algoritmo de detección	Fotograma de video.	El algoritmo genera un conjunto de caracteres y los ingresa en la base de datos.
3	Reconocer caracteres inválidos de una placa	Ejecutar GOCR	Fotograma de video.	Se filtra la cadena de caracteres y se guarda los datos.

Tabla IV- VII. Especificación de los casos de prueba

Ítem	Objetivo	Acción	Entrada	Resultado Esperado
1	Ingresar datos erróneos en autenticación	Ejecutar Autenticación	Usuario Contraseña	Se debe informar al usuario datos incorrectos
2	Indicar un origen de video al software con condiciones de luz de tarde	Ejecutar Algoritmo de detección	Fotograma de video.	El algoritmo genera un conjunto de caracteres y los ingresa en la base de datos.
3	Reconocer caracteres inválidos de una placa	Ejecutar GOCR	Fotograma de video.	Se filtra la cadena de caracteres y se guarda los datos.

Tabla IV- VIII. Especificaciones del procedimiento de prueba

Ítem	Objetivo	Acción	Entrada	Resultado Esperado	Resultado Obtenido
1	Ingresar datos erróneos en autenticación	Ejecutar Autenticación	Usuario Contraseña	Se debe informar al usuario datos incorrectos	Excelente
2	Indicar un origen de video al software con condiciones de luz de tarde	Ejecutar Algoritmo de detección	Fotograma de video.	El algoritmo genera un conjunto de caracteres y los ingresa en la base de datos.	Muy bueno
3	Reconocer caracteres inválidos de una placa	Ejecutar GOCR	Fotograma de video.	Se filtra la cadena de caracteres y se guarda los datos.	Excelente

Tabla IV- IX. Informe de los casos de prueba ejecutados

4.2.1.1 Referencias

- La IEEE es la encargada para la práctica de especificación de requerimientos de software según ANSI/IEEE estándar 830.
- Políticas y disposiciones del Departamento de Mantenimiento y Desarrollo Físico y Jefe de Seguridad; con el apoyo y coordinación técnica por parte del Departamento de Sistemas y Telemática en la ESPOCH.

CONCLUSIONES

1. OpenCV es un set de librerías de programación especializada en Visión Artificial, que al ser libre permite a cualquier programador de C/C++, indistintamente del entorno de desarrollo utilizado, realizar programas que resuelvan un problema específico. El algoritmo Sobel y Canny son parte de las más de 500 funciones que contiene OpenCV, que pueden ser utilizadas en infinidad de áreas, como el reconocimiento facial, de objetos, o la visión robótica, lo que la convierte en una herramienta poderosa y flexible para el análisis y tratamiento de imágenes.

2. El uso de QT Creator facilitó en gran medida el desarrollo de la aplicación tanto en el aspecto de interfaces como en el rendimiento de la misma.

3. El uso de threads contribuyó sustancialmente en mejorar el tiempo de respuesta que presenta la aplicación.

4. El operador "Sobel" presenta tanto una diferenciación como un efecto de suavizado. Debido a que las derivadas realzan el ruido, el suavizado permite eliminarlo en un gran porcentaje pero se pierden también ciertas regiones de interés importantes, disminuyendo la exactitud en la forma de la placa.

5. Al aplicar el algoritmo Canny se obtiene como resultado una imagen binaria en la que cada píxel se define como píxel de borde o como píxel no perteneciente al borde. Su principal ventaja es su gran adaptabilidad para poder ser aplicado a diversos tipos de imágenes, además de no disminuir su performance ante la presencia de ruido en la imagen original. Sin embargo, al implementar este algoritmo se identificaron desventajas en el suavizado puesto que al aumentar el σ de la mascarilla logramos reducir el ruido pero se difuminan los bordes y se pierde calidad al momento de calcular la orientación.

6. Considerando el análisis de resultados de la sección "Estudio Comparativo", el detector "Canny" se presenta como la mejor solución dado que se obtuvieron resultados muy satisfactorios al implementarse en imágenes reales, algunas de las cuales no eran de buena definición. De esta manera se concluye que el algoritmo de Canny es uno de los mejores métodos para la detección de bordes, el cual aplica métodos de diferencias finitas basadas en la primera derivada y cuya popularidad se debe, además de sus buenos resultados, a su sencillez, la cual permite una gran velocidad de procesamiento al ser implementado.

RECOMENDACIONES

1. Todos los algoritmos están sujetos a la desventaja de la complejidad computacional, siendo bastante importante el tiempo de cómputo en las maquinas convencionales. Como consecuencia, se debe conseguir o bien una mayor capacidad computacional o bien una mejora en la complejidad de los algoritmos.

2. Es necesario tener cuidado con el manejo de memoria dentro de la programación con C++ ya que no considerar este aspecto involucra un consumo excesivo de la misma.

3. El uso de cámaras IP involucra un conocimiento bastante amplio en el funcionamiento de la misma; por lo tanto se vuelve indispensable tener siempre al alcance el manual del desarrollador.

4. Se debe tener en cuenta que el algoritmo Sobel presenta un excelente tiempo en procesamiento pero en contraste, la rapidez de respuesta afecta el nivel de efectividad en la obtención de la localización de la placa. Mientras, que el detector Canny puede presentar excelentes resultados a pesar de trabajar con imágenes que no cuentan con una buena definición.

RESUMEN

Estudio comparativo entre Algoritmos de Reconocimiento de Borde para Identificación de placas de autos en el Departamento de Sistemas y Telemática en la ESPOCH.

La presente investigación está apoyada en el método "Deductivo", el mismo que ha permitido el análisis estadístico partiendo de un procedimiento que consiste en desarrollar una teoría y deduciendo luego sus consecuencias y el método "Comparativo" que ha permitido hacer una comparación sistemática entre los objetos de estudio determinando el algoritmo más eficiente para identificar los caracteres encontrados en una placa vehicular.

Se tomó fuentes de información de páginas web, observación, fotografías, entre los materiales utilizados están Hardware: Computador, Cámara Ip y en Software: DBMS Mysql, QT Creator, OpenCv para determinar los lineamientos que se deben cubrir.

Los resultados obtenidos en base a los parámetros comparados son tiempo de procesamiento: algoritmo Sobel presenta una respuesta de 37 milisegundos frente al algoritmo Canny con 40 ms; aciertos: algoritmo Sobel presenta un porcentaje de 81% y algoritmo Canny 85% .

Se puede concluir que la implementación del algoritmo Canny es más eficiente frente al algoritmo Sobel, su eficiencia no precisamente se ve reflejada en el tiempo de respuesta, sino en la calidad de detección de bordes.

Se recomienda la utilización del algoritmo Sobel por su eficiencia en tiempo de respuesta y en la calidad de detección de bordes, para el tratamiento de imágenes el conjunto de librerías "OpenCv" por ser de libre distribución reduce notablemente los costos de implementación, sin olvidar su excelente uso de recursos hardware.

SUMMARY

Comparative study of Edge Recognition Algorithms for identification license plates in the Department of Systems and Telematics in ESPOCH.

This research is supported in the method "Deductive" that allowed the same statistical analysis based on a procedure that is to develop a theory and then deducing its consequences and the method "Comparative" has permitted a systematic comparison between objects of study by determining the most efficient algorithm to identify the characters found on a license plate.

Sources of information was taken from web pages, comments, photographs, and materials used are Hardware: Computer, IP Camera and Software: DBMS MySQL, QT Creator, OpenCV to determine the guidelines that should be covered.

The results obtained based on the parameters are compared processing time: Sobel algorithm has a 37 millisecond response against Canny algorithm with 40 ms; successes: Sobel algorithm presents a percentage of 81% and 85% Canny algorithm.

It can be concluded that implementation of the Canny algorithm is more efficient compared with Sobel algorithm, its efficiency is not just reflected in the response time, but on the quality of edge detection.

We recommend the use of the Sobel algorithm for efficiency in response time and quality of edge detection for image processing libraries of "OpenCV" to be made freely available significantly reduces implementation costs, without forgetting excellent use of hardware resources.

GLOSARIO

Administrador	Persona autorizada para la administración del sistema propuesto, tiene la posibilidad de ejecutar todos los movimientos con respecto a las placas (actualizaciones, eliminación, consultar reportes, etc.)
Archivo	Colección de información (datos relacionados entre sí), localizada o almacenada como una unidad.
Atributos	Son características de una Clase que definen el grado de comunicación y visibilidad de ellos con el entorno.
Clase	Es la unidad básica que encapsula toda la información de un Objeto
Factibilidad	Indica que es posible de llevar a cabo o que es realizable en la realidad, y se espera que su resultado sea exitoso o satisfaga las necesidades.
Ingeniería de Software	Es una disciplina o área de las ciencias de la computación que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelve problemas de todo tipo.
Inteligencia artificial	Rama de la Informática que pretende desarrollar programas en los que el ordenador desarrolle conductas típicas de los seres inteligentes.
Interfaz	La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo, normalmente suelen ser fáciles de entender y fáciles de accionar.
Licencia BSD	Licencia que permite el uso del código fuente en software no libre.
LJPlate	Sistema de Reconocimiento de Placas de automóviles de la ESPOCH.

Máscara	Es una matriz de tamaño N*M que se aplica a un entorno o parte de una imagen centrada en el pixel (i,j). La aplicación de la máscara consiste en multiplicar cada elemento de la misma por el equivalente en el entorno o parte de la imagen en cuestión, sumar todos los productos, y asociarlo al pixel (i,j)
Máscaras de convolución	Consiste en recorrer la imagen global centrando la máscara en cada uno de los pixeles, aplicar la máscara sobre ellos y guardar el resultado en lo que será la nueva imagen.
Metodología	Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.
Métodos	Los métodos u operaciones de una clase son la forma en cómo ésta interactúa con su entorno.
Multi núcleo	Son chips independientes que contienen dos o más procesadores o núcleos de ejecución distintos en el mismo circuito integrado
Multiplataforma	Se refiere a que el software puede funcionar en diversas plataformas
Open source	Se conoce al software distribuido y desarrollado libremente.
Placa de automóvil	Es una placa metálica en donde se graba la matricula que identifica e individualiza el vehículo respecto a los demás.
Reportes	Documento caracterizado por contener información reflejando el resultado de una búsqueda adaptado al contexto de una situación dada.
Requerimiento	Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.
Usuario LJPlate	Estará orientado al monitoreo del sistema y consultar reportes de placas si así lo requiere dentro del LJPlate.
Visión artificial	Subcampo de la inteligencia artificial, cuyo propósito es programar un computador para que "entienda" una escena o las características de una imagen

ANEXOS

Anexo 1

Manual LJPlate

Anexo 2

Casos de Uso

Anexo 3

Pantallas del Sistema

BIBLIOGRAFÍA

1.- BRADSKY, G., Learning OpenCV Computer Vision with the OpenCV Library., Washington_United States of America., Oreilly., 2008., Pp. 31-120, 209-311.

2.- ESCOLANO, F.; COLOMINA, O. y otros., Inteligencia artificial: modelos, técnicas y áreas de aplicación., Madrid_España. Thomson., 2006., Pp. 4-7, 277-282.

3.- MIESENBERGER, K., Computers Helping People with Special Needs., Viena_Austria., Springer., 2010., Pp. 721-728.

4.- PAJARES, G. y DE LA CRUZ, J., Visión por Computador. Imágenes digitales y aplicaciones., México_México., Ra-Ma., 2001., Pp. 410-415.

5.- SHAPIRO, LINDA G. & STOCKMAN, y otros., Computer Vision., New Jersey__United States of America., Prentice Hall., 2001., Pp. 348-356

6.- VARGAS, J., Un nuevo algoritmo de detección de bordes en imágenes con ruido gaussiano., Sevilla_España., Universidad de Sevilla**.,** 2010., Pp. 4-8.

7.- VARGAS, R., Herramientas para realizar una investigación., Cochabamba_Bolivia., Universidad Mayor de San Simón., 2006. Pp. 23-31.

BIBLIOGRAFIA DE INTERNET

8.- ALGORITMO DE CANNY

http://es.wikipedia.org/wiki/Algoritmo_de_Canny

2011/06/21

9.- ANÁLISIS DE IMÁGENES

http://www.fic.udc.es/files/asignaturas/58IB/Analisis%20de%20imagenes%20biomedicas.pdf

2011/07/20

10.- LOS CIMIENTOS DEL OPENCV

http://www.starcostudios.com/blog/2011/03/los-cimientos-del-opencv-en-android/

2011/06/20

11.- MANUAL DE REFERENCIA DE CV

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

2011/07/20

12.- OPENCV.

http://es.wikipedia.org/wiki/OpenCV

2011/06/20

13.- OPENCV.

http://opencv.willowgarage.com/wiki/

2011/06/20

14.- OPERADOR SOBEL

http://piziadas.com/2011/02/editor-de-nodos-efectos-sobel-blender.html

2011/06/21

15.- PROCESAMIENTO DE IMÁGENES.

http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/ramos_r_m/capitulo3.pdf

2011/06/21

16.- RECONOCIMIENTO AUTOMÁTICO DE MATRÍCULAS

http://es.wikipedia.org/wiki/ANPR

2011/06/20

17.- RECONOCIMIENTO DE PLACAS

http://pvm1-placas.blogspot.com/

2011/06/21

18.- REGLAMENTO A LA LEY DE TRÁNSITO Y TRANSPORTE TERRESTRE

http://www.corpaire.org/siteCorpaire/upload_Files/varios/ReglamentoGeneralLeyTransito.pdf

2011/06/21

19.- SEGMENTATION (IMAGE PROCESSING)

http://en.wikipedia.org/wiki/Segmentation_(image_processing)

2011/07/20

20.- TEORÍA DEL COLOR

http://es.wikipedia.org/wiki/Teor%C3%ADa_del_color

2011/06/21

21.- VISIÓN ARTIFICIAL

http://es.wikipedia.org/wiki/Visi%C3%B3n_artificial

2011/06/20

Software de
Identificación de Placas
de Autos.

ESCRIBA EL TÍTULO DEL DOCUMENTO

[Escriba aquí una descripción breve del documento. Una descripción breve es un resumen corto del contenido del documento. Escriba aquí una descripción breve del documento. Una descripción breve es un resumen corto del contenido del documento.]

1. Generalidades

1.1. Introducción

El Software LJPlate surgió de la necesidad de automatizar y llevar un control de los vehículos que ingresan a la institución de la Escuela Superior Politécnica de Chimborazo (ESPOCH).

El presente manual pretende transmitir la estructura, conceptos e información básica de la operación del software, con la finalidad de que conociendo su funcionamiento los usuarios lo puedan utilizar de manera adecuada. Por lo mencionado, es de suma importancia, leer el manual que se detalla a continuación antes y durante el manejo del software.

El manual comienza explicando el ingreso al software, para luego ir explicando una a una todas las ventanas que conforman el Software y todas sus funciones.

1.2. Objetivos

El principal objetivo del presente manual, es ayudar al personal que vaya a utilizar este software a que conozca los conocimientos básicos de cómo opera y así mismo tenga una idea clara del alcance que tiene a través de la explicación ilustrada de cada una de las opciones que lo conforman.

1.3. A quién va dirigido el Manual

Este manual va dirigido a los usuarios finales involucrados en la operación del sistema, con conocimientos básicos del uso del computador.

2. Introducción al sistema

En este capítulo se detallará los requerimientos recomendados de hardware y software para el correcto funcionamiento del Sistema. Además, se detallan los elementos del entorno con el que el usuario debe relacionarse para el manejo del Sistema.

2.1. Requerimientos de Hardware y Software

Los requisitos recomendados de hardware y software son:

Hardware Procesador AMD Athlon X2 2.8 Ghz.
Memoria RAM de 1 GB o Superior
Cámara Toshiba Ik-wb21a

Software Sistema Operativo Ubuntu 10.04
MySql
QT Creator
OpenCv

2.2. Ingreso al Sistema

En este apartado, se detallará brevemente la forma en que el usuario debe acceder al Sistema:

1. En el escritorio se muestra el icono de acceso directo llamado LJPlate:

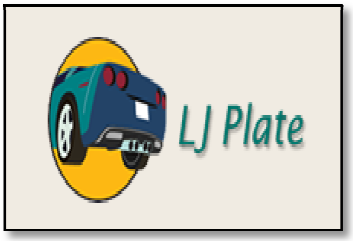


Ilustración 1. Acceso Directo a JLPlate

2. Dar doble clic sobre el icono mostrado en el paso anterior.

3. A continuación, aparecerá la pantalla de Autenticación del Sistema LJPlate que requiere el ingreso del Usuario y la Clave por parte del usuario:

Ilustración 2. Autenticación a JL Plate

4. Si el Usuario y la Clave ingresados son correctos, aparecerá la pantalla principal del software, donde se podrá visualizar las diferentes opciones del menú:

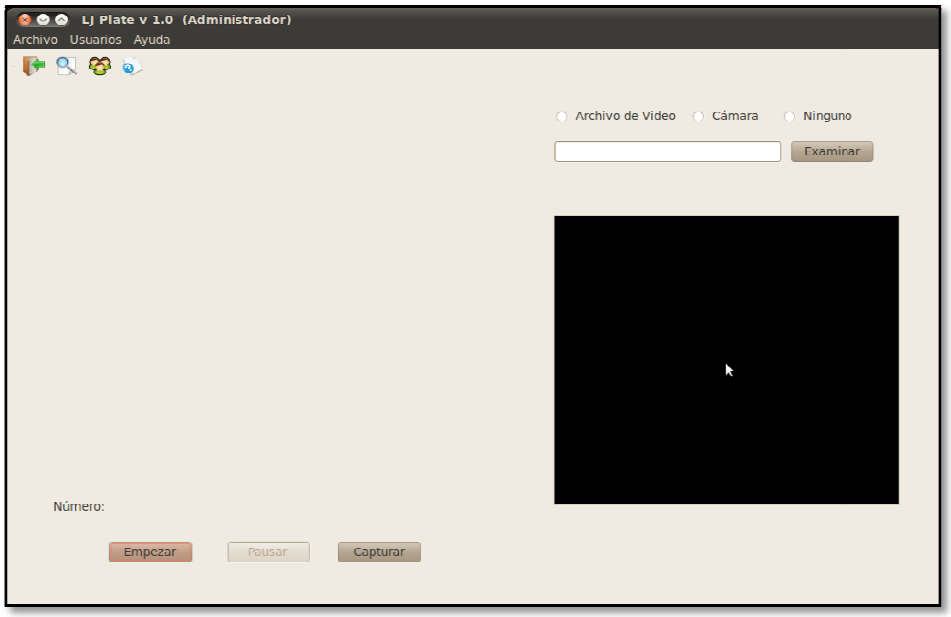


Ilustración 3. Pantalla Principal a LJPlate

2.3. Componentes de la Pantalla

En esta sección explicaremos cómo está compuesta la pantalla principal del software:

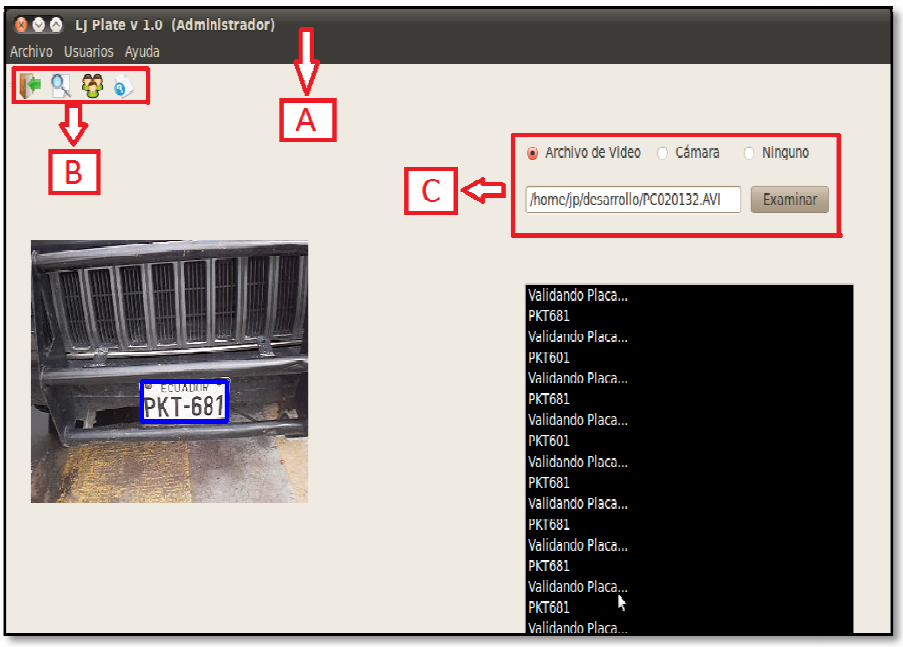


Ilustración 4. Componentes de la Pantalla LJPlate

De donde:

	Sección	Descripción
A	**Título de la Pantalla**	Indica el nombre de la pantalla, este título permite tener una idea referente a los datos que contienen dicha pantalla.
B	**Panel de Botones**	Estos botones realizan operaciones necesarias para manejar los datos que se ingresarán.
C	**Panel de Datos**	En esta parte el usuario efectúa el tratamiento de la imagen para reconocimiento de la placa.

Tabla 1. Descripción de Componentes.

2.4. Botones

En esta sección se describe la función principal de los botones que se presentan en el panel (Ver Ilustración 4). Cada botón tiene una etiqueta que indica la función del mismo.

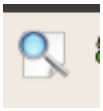
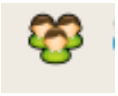
	Permite abrir y seleccionar un archivo.
	Busca una placa
	Permite ingresar un nuevo usuario
	Muestra información de los desarrolladores del software.

Tabla 2. Descripción de Botones.

Una vez detallado las generalidades básicas del Software, en los siguientes capítulos se describirá el manejo de los menús que ofrece el Software LJPlate.

3. Menú Sistema

Este menú presenta las siguientes opciones: Reportes, Buscar, Corregir Placas, Salir.

Ilustración 5. Opciones Menú Archivo

3.1. Reportes

Esta opción permite crear Reportes.

1. Se da clic en Menú Archivo, Reportes.

Ilustración 6. Opciones Menú Reportes

2. Aparece la siguiente pantalla donde podemos listar todas las placas o una placa determinada especificando su Tipo (Empleado, Estudiante, Docente, Particular). Y se establece el rango de Fecha y de Hora.

Ilustración 7. Tipificación de placas.

Ilustración 8. Rango de fechas para generar reporte.

3. Una vez que ya hemos configurado todos los parámetros para obtener nuestro Reporte damos clic en Generar y nos aparecerá la siguiente ventana en la cual podemos imprimir nuestro Reporte Generado.

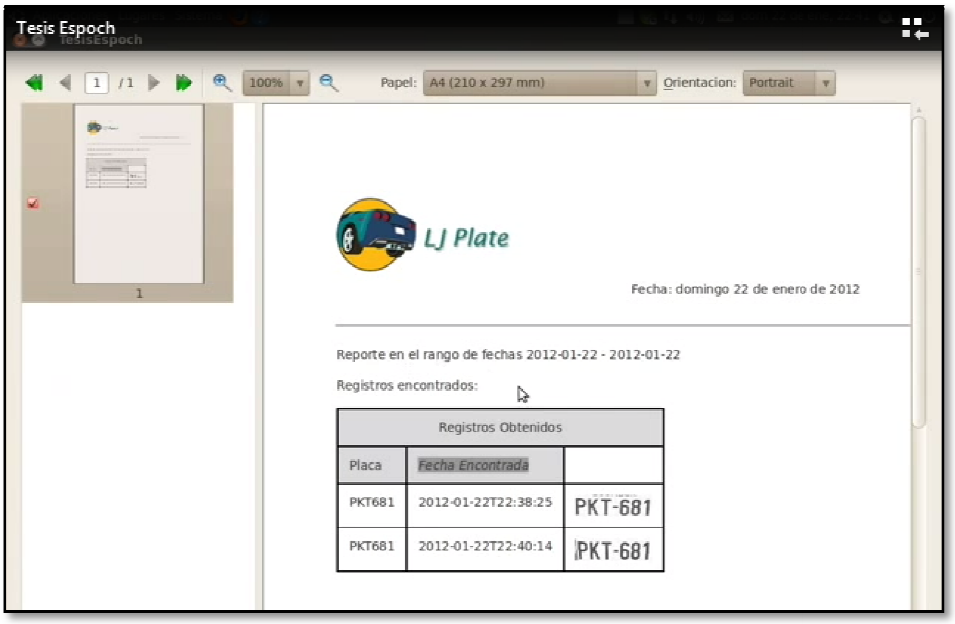


Ilustración 9.Reporte Generado.

3.2. Buscar

Esta opción permite buscar una placa.

1. Se da clic en Menú Archivo y Buscar.

Ilustración 10. . Opciones Menú Buscar

2. Nos aparece la siguiente pantalla en la cual se muestra la placa que queremos.

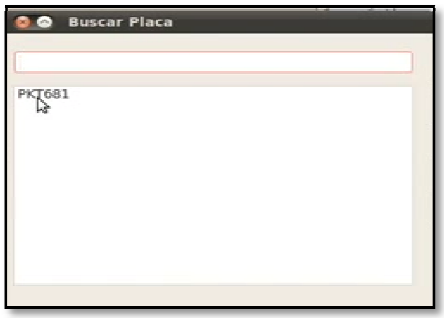


Ilustración 11. Placa encontrada

3. Ahora nos aparecerá el número de placa que seleccionamos y podemos cambiarle el tipo, una vez hecho esto procedemos a darle clic en Grabar.

Ilustración 12. Cambio de tipo de placa

3.3 Corregir

Esta opción permite corregir una placa.

1. Se da clic en Menú Archivo y Corregir Placas.

Ilustración XIII. Corregir placas.

2. Nos aparecerá la siguiente pantalla en la cual se muestra la placa que queremos corregir.

Ilustración XIV. Placa a corregir.

3. Ahora nos aparecerá el número de placa que seleccionamos y podemos cambiarle el número de placa y tipo, una vez hecho esto procedemos a darle clic en Grabar.

Ilustración XV. Modificación de Placa.

3.4. Usuarios

Esta opción permite ingresar usuarios al sistema.

1. Se da clic en Menú Archivo y Usuario.

Ilustración XVI. Opciones Menú Usuarios.

2. Nos aparecerá la siguiente pantalla en la cual se muestra los campos requeridos para dar de alta el nuevo usuario.

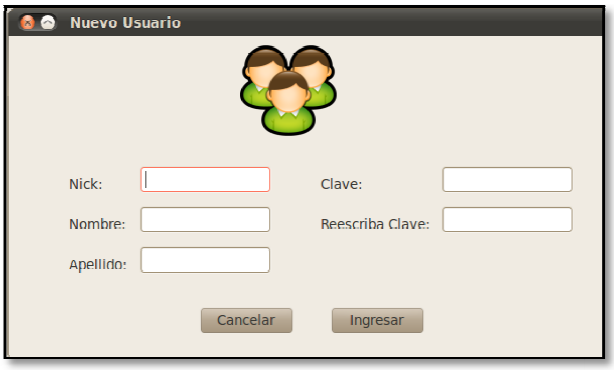


Ilustración XVII. Nuevo Usuario.

Requerimientos Funcionales

Validar usuarios.

- REQ 1: El sistema validará el inicio de sesión y password ingresados por el usuario, esta verificación le permitirá interactuar con la aplicación y ejecutar las tareas que le son permitidas.

Administración del sistema LJPlate

- REQ 2: El sistema podrá ser controlado por el Administrador, el mismo que tendrá la facultad ingresar y modificar placas, además de consultar reportes y realizar búsquedas personalizadas.
- REQ 3: El sistema permitirá dar de alta y modificar la placa de un vehículo.
- REQ 4: El sistema permitirá al usuario realizar el reconocimiento de la placa desde una cámara web, o desde un archivo en formato .jpg, .avi, .mjpeg. los campos que se añaden son el número de matrícula, el tipo de categoría al que pertenece el vehículo, imagen de la placa capturada, fecha y hora.
- REQ 5: El sistema permitirá visualizar el total de las placas reconocidas, dada la fecha y tipo de categoría.
- REQ 6: El sistema permitirá visualizar los resultados de una búsqueda de placa (matricula y tipo de categoría) dada su matrícula.
- REQ 7: El sistema permitirá ingresar usuarios.

Modelo de Casos de Uso del Negocio

El modelo de Casos de Uso describe las funciones del sistema y los actores que hacen uso de ellas. Esta herramienta es un apoyo para la explicación de las funcionalidades que presenta el software. Las plantillas que se muestran están en formato extendido y pertenecen a los requisitos funcionales del sistema LJPlate.

Identificador caso de uso	I-SESION
Nombre del caso de uso	Iniciar Sesión
Actores	Usuario
Propósito	Realizar una operación de inicio de sesión de una cuenta del sistema LJPlate
Visión General	Un usuario del sistema introduce su nombre y contraseña, y solicita la operación de autenticación, el sistema valida los datos ingresados y permite iniciar sesión al usuario.
Tipo	Primario
Referencias	Funciones: R1
Curso típico de eventos	
Acciones del Actor (Usuario)	Respuesta del Sistema
Usuario introduce su nombre y contraseña	Validación de datos
Ingresa al sistema.	En espera
Cursos Alternativos	
El usuario o la clave son incorrectos. Se indica el error y se cancela la operación	

Plantilla de caso de uso Iniciar Sesión en formato extendido

Identificador caso de uso	R-INGRESOV
Nombre del caso de uso	Realizar Ingreso desde Video
Actores	Usuario
Propósito	Realizar una operación de ingreso de placa de automóvil desde video.
Visión General	Un usuario del sistema pulsa Examinar e indica la ubicación del archivo de video, el sistema carga temporalmente el archivo. El usuario pulsa el botón de Empezar, el sistema realiza el análisis del fotograma actual y devuelve la placa del automóvil. Seguidamente el usuario pulsa Buscar y digita el número de placa a ingresar e indica la categoría que pertenece, finalmente pulsa el botón de Guardar, el sistema guarda la placa del automóvil y la hora en que se realizó la operación.
Tipo	Primario
Referencias	Funciones: R3,R4

Curso típico de eventos

Acciones del Actor (Usuario)	Respuesta del Sistema
Usuario pulsa el botón Examinar e indica la ubicación del archivo.	Carga el archivo de video.
Pulsa el botón Empezar	Procesa la petición y realiza el análisis del fotograma actual. Devuelve la placa del automóvil.
Pulsa el botón Buscar y digita el número de placa a ingresar e indica la categoría.	En espera
Pulsa el botón Guardar	Se guarda la placa.

Cursos Alternativos
La placa mostrada por el sistema es incorrecta. El usuario modifica la placa y pulsa el botón Guardar.

Plantilla de caso de uso Ingreso de placa desde video en formato extendido

Identificador caso de uso	R-INGRESOI
Nombre del caso de uso	Realizar Ingreso desde Imagen.
Actores	Usuario
Propósito	Realizar una operación de ingreso de placa de automóvil desde una imagen.
Visión General	Un usuario del sistema pulsa Examinar e indica la ubicación de la imagen, el sistema carga temporalmente el archivo. El usuario pulsa el botón de Empezar, el sistema realiza el análisis de la fotografía y devuelve la placa del automóvil. Seguidamente el usuario pulsa Buscar y digita el número de placa a ingresar e indica la categoría que pertenece, finalmente pulsa el botón de Guardar, el sistema guarda la placa del automóvil y la hora en que se realizó la operación.
Tipo	Primario
Referencias	Funciones: R3,R4

Curso típico de eventos	
Acciones del Actor (Usuario)	Respuesta del Sistema
Usuario pulsa el botón Examinar e indica la ubicación del archivo.	Carga el archivo de imagen.
Pulsa el botón Empezar	Procesa la petición y realiza el análisis. Devuelve la placa del automóvil.
Pulsa el botón Buscar y digita el número de placa a ingresar e indica la	En espera

categoría.	
Pulsa el botón Guardar	Se guarda la placa.
Cursos Alternativos	
La placa mostrada por el sistema es incorrecta. El usuario modifica la placa y pulsa el botón Guardar.	

Plantilla de caso de uso Ingreso de placa desde fotografía en formato extendido

Identificador caso de uso	E-REPORTE
Nombre del caso de uso	Emitir Reportes
Actores	Usuario
Propósito	Realizar una operación de emisión de reportes.
Visión General	Un usuario solicita realizar la operación de emitir, el sistema genera el reporte.
Tipo	Primario
Referencias	Funciones: R5,R6
Curso típico de eventos	
Acciones del Actor (Usuario)	Respuesta del Sistema
Usuario selecciona la operación de emitir reportes e ingresa la fecha y/o número de placa (opcional).	Procesa la petición y emite reportes.
Cursos Alternativos	

Plantilla de caso de uso Emitir reportes en formato extendido

Identificador caso de uso	R-BUSQUEDA
Nombre del caso de uso	Realizar Búsqueda
Actores	Usuario
Propósito	Realizar una operación de búsqueda de placa de automóvil
Visión General	Un usuario del sistema introduce la placa de automóvil y solicita realizar la operación de búsqueda, el sistema visualiza los datos de la placa tras comprobar que esta existe.
Tipo	Primario
Referencias	Funciones: R6
Curso típico de eventos	
Acciones del Actor (Usuario)	Respuesta del Sistema
Usuario introduce la placa de automóvil	Procesa la petición y muestra los datos de la placa especificada.

Cursos Alternativos
La placa no existe. Se indica el error y se cancela la operación.

Plantilla de caso de uso Búsqueda en formato extendido

Identificador caso de uso	C-USUARIO
Nombre del caso de uso	Crear Usuario
Actores	Usuario
Propósito	Realizar una operación de creación de usuarios
Visión General	Un usuario introduce el nombre y contraseña del nuevo usuario del sistema, el sistema da de alta al nuevo usuario.
Tipo	Primario
Referencias	Funciones: R7
Curso típico de eventos	
Acciones del Actor (Usuario)	Respuesta del Sistema

Usuario introduce nombre y contraseña	Procesa la petición y da de alta al nuevo usuario.
Cursos Alternativos	
El nombre de usuario ya existe. Se indica el error y se cancela la operación.	

Plantilla de caso de uso Creación de usuarios en formato extendido

Diagrama de Casos de Uso del Negocio

Los Diagramas de Casos de Uso permiten una mejor interpretación de la información contenida en las plantillas anteriores.

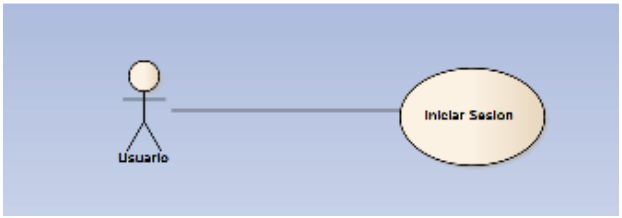


Diagrama de Casos de Uso del Negocio 1

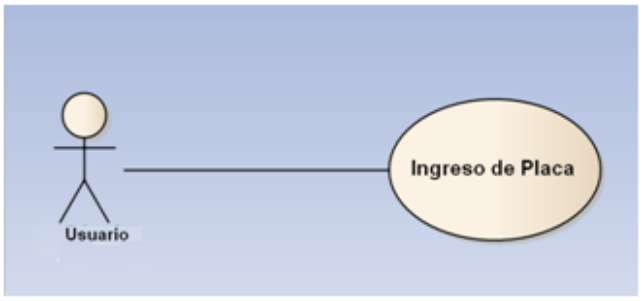


Diagrama de Casos de Uso del Negocio 2

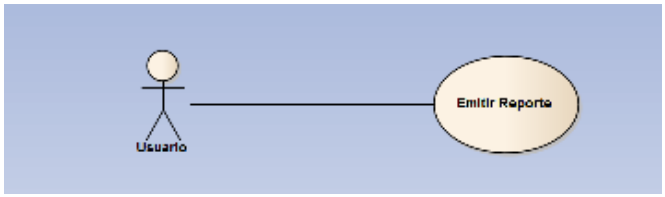


Diagrama de Casos de Uso del Negocio 3

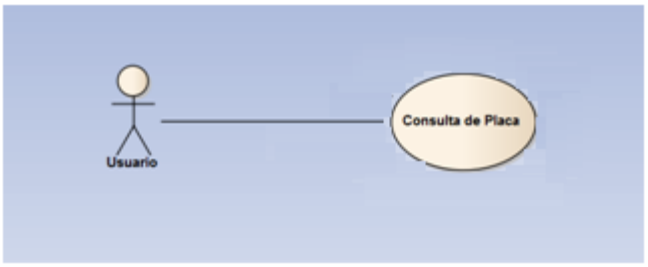


Diagrama de Casos de Uso del Negocio 4

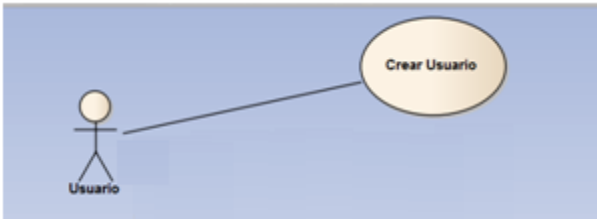


Diagrama de Casos de Uso del Negocio 5

Pantallas del Sistema

En esta sección se muestra los prototipos de interfaces gráficas de usuario diseñadas para la aplicación final.

Pantalla Autenticación

La aplicación dispone de una primera ventana de identificación del usuario. Sólo usuarios que contengan los datos de autenticación correcta pueden hacer uso de las funciones del sistema.

Figura 1. Autenticación de usuario.

Pantalla Principal

Pantalla principal que permite acceder a las distintas funcionalidades del programa.

Figura 2. Pantalla principal.

Pantalla de Reconocimiento de placas

Esta pantalla permite reconocer placas desde un video, imagen o una cámara. En la *Figura 3* se muestra cómo funciona el reconocimiento desde un video.

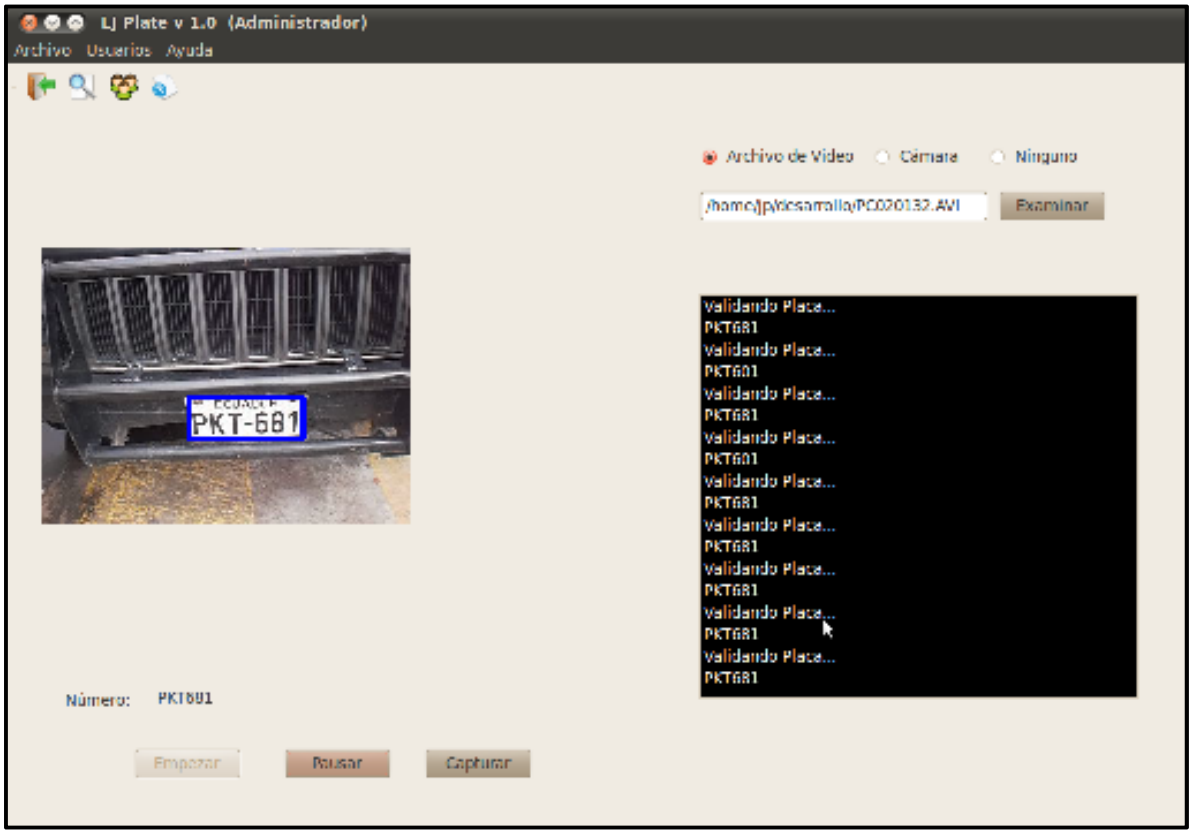


Figura 3. Reconocimiento de placas utilizando un archivo de video.

Pantalla de Búsqueda

Esta pantalla permite realizar búsqueda personalizadas por el usuario.

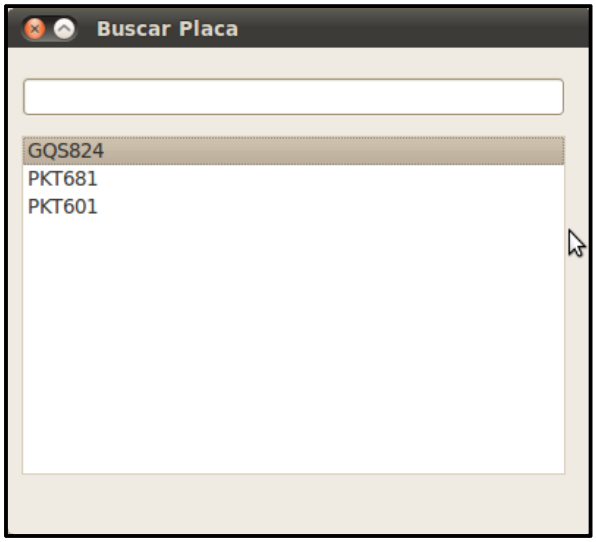


Figura 4. Búsqueda de placa.

Pantalla de Corregir Placa

Esta pantalla permite realizar una modificación de una placa que haya sido ingresada.

Figura 5. Modificación de placa.

Pantalla de Reporte

Esta pantalla permite mostrar reportes por fechas de las placas que han ingresado.

Figura 6. Reportes

Pantalla de Creación de Usuarios

Esta pantalla permite crear cuentas para el manejo del sistema.

Figura 7. Creación de usuarios.

Pantalla Acerca de

Muestra información de los desarrolladores del software.

Figura 8. Acerca de.