Universidad Internacional de La Rioja
Escuela Superior de Ingeniería y Tecnología

Máster en Ingeniería Matemática y Computación

Algoritmo de identificación de la matrícula, color y modelo de un vehículo

Trabajo fin de estudio presentado por:	Eduardo Fernández Ortiz Jose Luis Pañi Pizarro José Luis Tinajero León
Tipo de trabajo:	Tipo 1. Desarrollo práctico
Modalidad:	Grupal
Director/a:	Dr. Leonardo Flores Cano
Fecha:	22/07/2021

Resumen

La toma de decisiones asociado a accidentes de tránsito en áreas como Sistema Judicial, Planificación vial, Salud Publica, etc., dependen exclusivamente de la manera en cómo se registra, almacena y se da seguimiento a la información. Un ejemplo claro se puede notar en el ámbito Judicial en donde, este ente muchas veces se ve condicionada a impartir justicia, debido a que, en un accidente de tránsito con un porcentaje muy alto, el infractor se da a la fuga y la carencia de equipos técnico-científicos limitan la obtención de la información. Esto sucede sobre todo en países con escaso desarrollo tecnológico. Para abordar este problema, se propone el desarrollo de un algoritmo que permita identificar de forma automática y a través de imágenes, ciertas características únicas de un vehículo como: la placa, el color y su modelo. Utilizando como entorno para su desarrollo el software de programación MATLAB®. El mismo que además de permitir la implementación del algoritmo, sirve para obtener datos estadísticos del funcionamiento a partir de aciertos y errores del sistema, mediante una comparación entre la imagen ingresada y la imagen de la base de datos. Para la evaluación y validación de este desarrollo, a los datos estadísticos generados por el software, se le aplica diversas técnicas estadísticas entre las que se destaca la matriz de confusión, que permite validar el modelo a partir de un clasificador previamente entrenado en la aplicación de MATLAB Classification Learner.

Palabras clave: ALPR, ANPR, VSM, procesamiento de imágenes, reconocimiento color, reconocimiento marca vehículo.

Abstract

Decision-making associated with traffic accidents in areas such as: Judicial System, Road Planning, Public Health, etc., depends exclusively on how the information is recorded, stored, and tracked. A clear example can be seen in the judicial field where this entity is often conditioned to impart justice, because, in a traffic accident with a very high percentage, the offender runs away, and the lack of technical-scientific teams limit the obtaining of information. This is especially the case in countries with little technological development. To address this problem, it is proposed to develop an algorithm that allows to identify automatically and through images, certain unique characteristics of a vehicle such as: the license plate, the color, and its model. Using MATLAB® programming software as an environment for its development. The same that in addition to allowing the implementation of the algorithm, serves to obtain statistical data of the operation from successes and errors of the system, by means of a comparison between the entered image and the image from the database. For the evaluation and validation of this development, various statistical techniques are applied to the statistical data generated by the software, among which the confusion matrix stands out, which allows validating the model from a classifier previously trained in the application of MATLAB Classification Learner.

Keywords: ALPR, ANPR, VSM, Computer vision, Color Recognition, Brand Recognition

Índice de contenidos

Índice de figuras

Índice de tablas

Índice de ecuaciones

1. Organización del trabajo en grupo

A continuación, se describe las diferentes áreas del trabajo donde se ha asignado a un responsable para cada una de ellas para así alcanzar los objetivos propuestos.

Tabla 1. Organización del trabajo en grupo.

Epígrafe	Alumno responsable	Objetivos perseguidos
2. Introducción	Todos, responsable Pañi	Descripción del proyecto para el desarrollo del TFM y planteamiento de los objetivos generales. Elaboración del primer borrador de la introducción.
3. Contexto y estado del arte	Todos, responsable Pañi	Análisis de estado del arte actual.
4. Identificación de requisitos	Todos, responsable Tinajero	Análisis de los datos a utilizar y modelado propuesto.
5. Descripción del sistema de software desarrollado	Todos, responsable Tinajero	Detallar las tareas a realizar por cada grupo implicado en el proyecto.
6. Metodología del trabajo	Todos, responsable Eduardo	Detallar los detalles técnicos del sistema creado.
7. Evaluación	Todos, responsable Tinajero	Medir la bondad del modelado realizado y mostrar una prueba.
8. Conclusiones, limitaciones y plan de contingencia	Todos	Objetivos perseguidos: exponer conclusiones y valoración del proyecto entendiendo las limitaciones.
Referencias	Todos	Detallar las fuentes de donde hemos obtenido información y basado nuestro trabajo.

En cuanto a los mecanismos de coordinación empleados a lo largo del trabajo, se detallan los siguientes:

- Se realizan reuniones virtuales por parte del equipo con una frecuencia mínima de una cada semana para poder un correcto seguimiento y alineamiento.
- Reuniones virtuales y menajes con el tutor para tratar el avance y seguimiento del trabajo realizado.

Herramientas de coordinación utilizadas:

- Microsoft Teams y Google Meet, utilizada para compartir escritorio y consensuar el progreso del proyecto.
- Plataforma Adobe Connect, para el envío de mensajes y reuniones con el tutor.
- Google Drive: para como repositorio de toda la documentación compartida, para evitar duplicados y trabajar con la misma versión por parte de todos los componentes del equipo.
- Grupo de WhatsApp y correo electrónico, para reuniones cortas y no programas.

2. Introducción

Según los datos de la Organización Mundial de la Salud, las muertes por accidentes de tráfico han aumentado a 1,35 millones al año, equivalente a casi 3.700 personas que mueren en las carreteras del mundo todos los días (World Health Organization, 2018). Esto debido al crecimiento de la población y con ello también el incremento del parque automotor. Afectando directamente a la movilidad humana y al tránsito vehicular, siendo este último un aspecto urgente por resolver, debido sobre todo a los altos índices de accidentabilidad que afecta la vida de millones de personas.

Un aliado para contrarrestar esta problemática sin duda son los avances tecnológicos. Por cuanto ha permitido encontrar nuevas formas de desarrollo en seguridad del transporte. Los Sistemas Inteligentes de Transporte (SIT) son un ejemplo, cuyo éxito se basa en el análisis combinado de la información que envían los equipos instalados en el lugar mismo de un posible siniestro de tránsito. Estos ofrecen soluciones muy útiles en cuanto al aumento de capacidad de unas infraestructuras insuficientes, pueden mejorar la situación de las redes en congestiones, tasas de siniestralidad, protección ambiental, etc.

Tomando como referencia lo antes mencionado este trabajo pretende aportar un algoritmo que permita a partir de una aplicación, identificar de forma semiautomática ciertas características únicas de un vehículo, mediante imágenes tomadas por una cámara en color, en blanco y negro o en infrarrojo.(Du et al., 2013). A diferencia de otras tecnologías, el uso de las imágenes proporciona comprobación visual y con la ayuda de técnicas de procesamiento de imágenes, cuyo objetivo es extraer información concreta de una imagen.(Vera et al., 2010) se logrará el cometido.

2.1.Justificación

En la actualidad, el alto índice de siniestros en el tránsito terrestre se posiciona como uno de los problemas más graves que afectan a la población mundial, debido a su impacto negativo que afecta considerablemente el bienestar de las personas quienes lo sufren.

Específicamente en Ecuador, según las estadísticas de la Agencia Nacional de Tránsito (ANT), en el año 2020 se produjeron 16.972 siniestros de tránsito, de los cuales 5.358 tuvieron que ver con transporte tipo automóvil. Además, en 3.236 casos no se pudo identificar el tipo de transporte que ocasionó el accidente (Agencia Nacional de Transito, 2020).

Según el criterio judicial la falta de equipos técnico-científicos al producirse un siniestro de tránsito, limitan el accionar de la autoridad encargada para realizar las respectivas pericias accidentológicas, impidiendo un juzgamiento efectivo al no contar con la identificación del vehículo que se dio a la fuga.

Para contrarrestar estas deficiencias y a partir del desarrollo tecnológico, se viene desarrollando diversas aplicaciones, cuyas técnicas permiten el procesamiento digital de imágenes. Una de las pioneras son los Sistemas Inteligentes de Transporte (SIT) que permiten incrementar la seguridad, eficiencia y confort del transporte mejorando la funcionalidad de los coches y las carreteras, usando tecnologías de la información (Collado & Hilario, 2003).

Haciendo referencia a la problemática descrita, este trabajo de investigación pretende aportar mediante conocimientos de procesamiento de imágenes, adquiridos en el Máster de estudio, un algoritmo de identificación de placa, color y modelo de vehículo con la finalidad de identificar al móvil causante del accidente, en caso de que este fugara del lugar del siniestro.

El algoritmo propuesto trabajará con imágenes previamente ingresadas en una base de datos y otras que se ingresaran al sistema. En una primera fase de experimentación estas imágenes estarán delimitadas a distintos modelos y colores de vehículos de la marca Chevrolet, que según las estadísticas de la Asociación de empresas Automotrices del Ecuador (AEADE) es la marca con mayor demanda en el mercado automotriz del país (AEADE, 2021).

Cabe destacar que este trabajo de desarrollo práctico, integra la temática de procesamiento de señales sonido e imágenes digitales. De manera específica, aborda la digitalización de imágenes y vídeo que están relacionadas al máster en estudio.

2.2. Planteamiento del trabajo

El presente Trabajo de Fin de Máster, pretende implementar un algoritmo capaz de identificar ciertas características de un automóvil como la placa, el color y su modelo, el cual permitirá identificar al vehículo que abandonase el lugar del siniestro. Para su identificación hará uso de imágenes que reposaran en una base de datos y otras que se deben ingresar al sistema. A estas imágenes se le aplicará una etapa de preprocesado, que permitirá mejorar las condiciones de las imágenes.

Para lograr este cometido se trabajará con tres editores de código fuente distintos: El uno será destinado para la programación del reconocimiento de la placa del vehículo, otro para el reconocimiento de color del vehículo y el ultimo destinado para la programación del reconocimiento del modelo del vehículo. Estos tres editores distintos serán compilados en un uno solo a través de un GUIDE, que será la interfaz para la interacción con el usuario.

En el algoritmo de detección de placa se realizará varias etapas de procesamiento digital de imágenes, entre las cuales están: el corte del área de interés, es decir, se selecciona las letras y números que conforman la identificación única del vehículo. Transformación de una imagen RGB a una imagen en escala de grises, para evitar el coste computacional. Cálculo del umbral de la imagen anterior, este valor se calcula para binarizar la captura procesada, es decir, que la captura solo esté conformada por valores de 0 y 1. Aplicación de un elemento estructurante y la operación de cierre en la imagen, se lo realiza para tratar de eliminar cualquier píxel aislado que posee la captura.

En cuanto al algoritmo de detección de color, se utilizará dos técnicas para su reconocimiento: La primera identificar por comparación, realiza la identificación de color mediante una comparación de píxel de la imagen ingresada con la base de datos que posee el sistema. La segunda identificar por clasificación, realiza una identificación basada en un clasificador el mismo que será entrenado empleando un clasificador por aprendizaje o entrenamiento. Este clasificador utilizara una base de datos que cuenta con 5052 combinaciones que corresponden a 11 colores en los cuales se pueden identificar Negro, Azul, Café, Verde, Gris, Naranja, Rosa, Morado, Rojo, Blanco y Amarrillo. Estos colores son clasificados considerando una Máquina de Vector de soporte en una condición Cubic SVM.

En la detección del modelo se aplicará transformaciones geométricas que indiquen el nivel de parecido entre dos imágenes. Si el nivel de correlación es alto, se detectará y extraerá los puntos característicos de las imágenes, para realizar su respectiva estimación geométrica.

Para la implementación del algoritmo se utilizará el software de programación MATLAB®, validando y evaluando el algoritmo mediante el ingreso de un grupo de imágenes, las mismas que permitirán obtener datos estadísticos del funcionamiento a partir de aciertos y errores del sistema.

2.3. Estructura del trabajo

El presente trabajo está organizado en 8 capítulos, detallados de la siguiente manera:

- Capítulo 1 Organización del trabajo en grupo. Explica el proceso para desarrollar el trabajo a partir de la división de tareas y los compromisos que asume cada integrante en este trabajo grupal.
- Capítulo 2 Introducción. Presenta el contexto de los siniestros de tránsito y la importancia de la aplicación tecnológica, para resolver casos en donde luego de un accidente uno de los vehículos involucrados abandone el lugar de los hechos.
- Capítulo 3 Contexto y estado del arte. Fundamenta el trabajo a desarrollar, partiendo de la revisión de trabajos similares y las soluciones desarrolladas para el mismo problema. Además, hace hincapié a trabajos relevantes, los mismos que darán realce al trabajo propuesto.
- Capítulo 4 Identificación de requisitos. Determina los requisitos necesarios para el desarrollo y usabilidad del algoritmo por parte del usuario. Estos requisitos se enfocan al comportamiento que va a tener el software en la interacción directa con el usuario. Permitiéndole una navegación libre cuyo propósito es dar solución a las diversas etapas de identificación que posee el sistema.
- Capítulo 5 Descripción del sistema de software desarrollado. Aporta detalles del proceso para desarrollar el algoritmo que permita identificar los caracteres de la placa de un vehículo, su color y el modelo, a partir de las etapas de identificación y reconocimiento.

- Capítulo 6 Metodología del trabajo. Establece los detalles técnicos del algoritmo desarrollado, a partir de las tecnologías utilizadas. Determina también la forma de evaluar el modelo para resolver el problema planteado.
- Capítulo 7 Evaluación. Detalla el proceso y los resultados obtenidos, luego de la evaluación de usabilidad y tomando como referencia el problema propuesto a resolver.
- Capítulo 8 Conclusiones y Trabajos Futuros. Presenta las conclusiones del trabajo realizado en función de los resultados obtenidos. También puntualiza las líneas de trabajo futuras que se puedan derivar de este trabajo de investigación.

3. Contexto y estado del arte

El presente capítulo detalla los fundamentos y el contexto teórico, empleados en el desarrollo del algoritmo de detección, así como también el análisis del estado del arte, que se agrupa en tres líneas de desarrollo: reconocimiento automático de matrículas, color y modelo de un vehículo.

3.1. Reconocimiento automático de matrículas de vehículos

El reconocimiento automático de matrículas conocido también como ANPR ("Automatic Number Plate Recognition") o LPR ("License Plate Recognition"), es la extracción de la información de las matrículas de los vehículos a partir de una imagen o una secuencia de imágenes, la información extraída puede utilizarse con o sin una base de datos en muchas aplicaciones (Du et al., 2013). Esta información se obtiene de una placa de metal, que está fijada en el exterior de la carrocería del vehículo, la misma incluye palabras y caracteres cuya finalidad es identificar a los vehículos.

Debido a la diversa discriminación de placas de vehículos con respecto a la forma, tamaño, lenguaje, signos y colores de un país a otro. Se han propuesto muchos métodos de ANPR dependiendo de las características y regulaciones de cada país (Bedir Yousif et al., 2020). Sin embargo, la extracción exitosa de la información que proporcione la placa no es una tarea fácil, se deben considerar algunos factores importantes como el tamaño de la placa, la calidad de la imagen, el estilo de la placa, la condición de iluminación, la ubicación de la placa y las especificaciones del fondo (Corneto et al., 2017).

Es así como, el proceso de reconocimiento automático se ayuda, de una secuencia de tratamientos sobre la imagen, para posteriormente mediante algoritmos de reconocimiento de patrones extraer los caracteres de la matrícula.

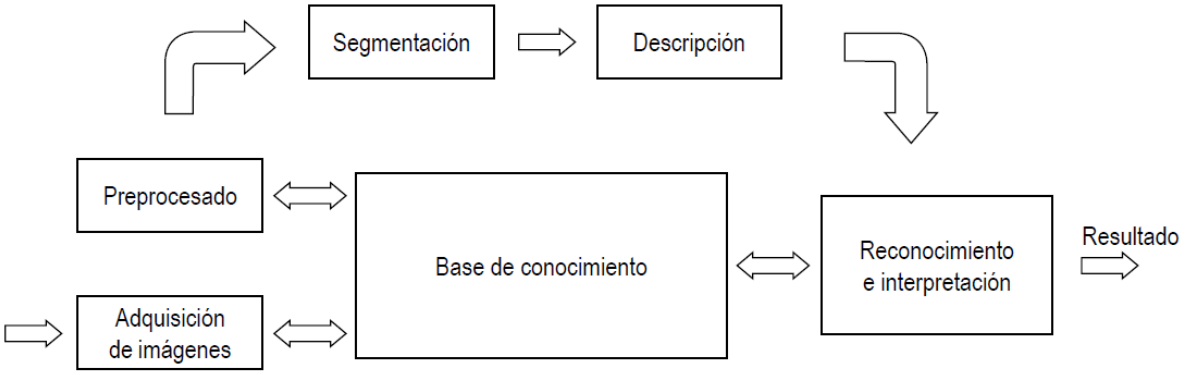
3.1.1. Procesamiento digital de imágenes.

Una imagen digital puede ser definida como una función bidimensional de intensidad de luz $f(x, y)$, donde x e y representan las coordenadas espaciales y el valor de f en un punto cualquiera (x, y) , es proporcional al brillo (o nivel de gris) de la imagen en ese punto considerado como una matriz, cuyos índices de fila y de columna identifica un punto de la imagen y el valor del correspondiente elemento de la matriz indica el nivel de gris en ese punto

(Palomino et al., 2009) denominados elementos de la imagen o más comúnmente pixels (picture elements).

En la Figura 1, se puede divisar las etapas para el procesamiento de imágenes. El proceso inicia con la obtención de las imágenes, seguido de un preprocesamiento para detectar y eliminar defectos, en la siguiente etapa se puede ver la segmentación cuyo propósito es separar sectores que no interesan, la siguiente etapa es la extracción de características, se viene la etapa de reconocimiento que asigna el protocolo de acuerdo con la información proporcionada y finalmente la base del conocimiento que guía la operación de los módulos de procesamiento.

Figura 1. *Etapas del Procesamiento Digital de Imágenes.*



Fuente. Tomada de (Palomino et al., 2009).

3.1.2. Extracción de la matrícula

Existen diversas metodologías tales como:

A. Extracción de la matrícula mediante detección de bordes en la imagen.
   Mediante la aplicación de filtros en la imagen, se detecta líneas horizontales y verticales que delimitan la porción de la imagen con la matricula.
B. Extracción de la matrícula mediante el uso de información global de la imagen.
   Donde mediante el análisis del etiquetado de componentes conectados de la imagen escanea una imagen y agrupa sus píxeles en componentes según la conectividad de píxeles, es decir, todos los píxeles de un componente conectado comparten valores de intensidad de píxeles similares y están conectados entre sí de alguna manera.
C. Extracción de la matrícula mediante el uso de texturas en la imagen.

Con este método se buscan los cambios significativos de escala de grises en los bordes de la matrícula.

D. Extracción de la matrícula mediante el uso del color en la imagen.
   Con este método se buscan los colores significativos de las matrículas según el país de origen, donde la combinación de los colores de la placa y caracteres es única. Los colores de las matrículas con blanco, negro, rojo y verde, de ahí el fundamento de este método.
E. Extracción de la matrícula mediante el uso de caracteres en la imagen.
   Donde la imagen se examina en búsqueda de caracteres para localizar la matrícula.

A continuación, en la tabla 2 se resumen los diferentes métodos destacando sus ventajas e inconvenientes.

Tabla 2. *Ventajas y Desventajas Métodos Extracción de Matricula.*

Método	Raciocinio	Ventajas	Desventajas
Contornos	Matricula rectangular	Simple, rápido y directo	Sensible a bordes inesperados
Imagen global	Objeto similar a matrícula	Directo.	Objetos rotos
Textura	Transición color	Incluso placas deformadas	Coste computacional alto
Color	Color específico	Incluso placas deformadas	Limitado a buena iluminación
Caracteres	Debe haber caracteres	Robusto en rotación	Consumo alto de tiempo
Usando características combinadas	Mas efectivo	Más fiable	Computacionalmente complejo

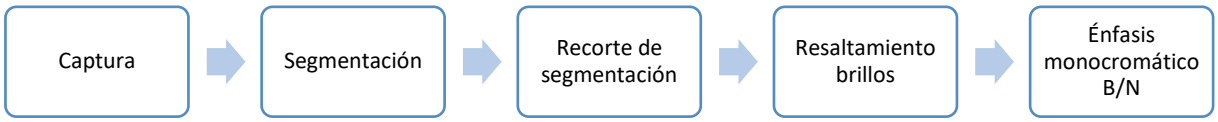
Fuente. Adaptada de (S. Du, 2013).

3.1.3. Preprocesado de imágenes

Es el proceso de tratamiento de la imagen, que permite eliminar ruido, minimizar la interferencia de elementos no requeridos, mejorar ciertas características que son de interés, previo a iniciar con la identificación de caracteres. Es decir, el proceso desde la captura a la

segmentación de la imagen, con la finalidad que, al momento de aplicar el algoritmo, funcione de manera adecuada cada una de sus etapas y se pueda obtener una imagen con mejor detalle para el análisis. Este flujo de preprocesamiento se detalla en la figura 2.

Figura 2. *Flujo de Preprocesamiento de Imágenes.*



Fuente. Elaboración Propia.

Esta secuencia de preprocesamiento en la imagen es realizada mediante algoritmos implementados en el Software MATLAB® incluidos en el paquete "Image Processing Tool", imread para la lectura de la imagen, rgb2gray para pasar la imagen a escala de grises, imcrop para recortar la imagen, sobel para aplicar el filtro Sobel en encontrar los bordes, imbothat para aplicar el filtro Bottom-Hat para resaltar la matrícula y ciertos bordes, strel para aplicar un filtro morfológico y resaltar elementos verticales y horizontales, imdilate para aplicar otro filtro morfológico y añadir pixeles a los contornos.

3.1.3.1. Captura de la imagen

Captura la imagen de la matrícula mediante el uso de una cámara. Los parámetros de la cámara tales como: tipo, resolución, velocidad de obturación, la orientación y la luz deben ser considerados.

3.1.3.2. Segmentación de la matrícula

La segmentación subdivide una imagen en sus partes constituyentes u objetos, con la finalidad de separar las partes de interés del resto de la imagen, por lo tanto el nivel al que se lleva a cabo esta subdivisión depende del problema a resolver (Palomino et al., 2009).

Existen diversas metodologías de segmentación tales como:

A. Uso de conectividad de píxeles.
   Mediante el etiquetado de los pixeles conexionados en la imagen de la matrícula.
B. Uso búsqueda líneas horizontales o verticales.
   Mediante el contraste del borde generado entre el carácter y el color de la matrícula.
C. Uso de conocimientos previos de caracteres.

Se procede a un escaneo horizontal para localizar el inicio y final de los caracteres mediante comparación con el color de la placa.

D. Usando contornos de caracteres.

Mediante dos pasos aplicando técnicas ordinarias y especiales de fast marching. Un método numérico creado por james Sethian para resolver problemas de contorno.

E. Usando características combinadas.

Combinando el uso de dos o más algoritmos de morfología adaptiva, tales como algoritmos de ensanchamiento (thickening) y de adelgazamiento (thinning).

A continuación, en la tabla 3 se resumen los diferentes métodos destacando sus ventajas e inconvenientes.

Tabla 3. *Ventajas y Desventajas Métodos Segmentación Matrícula.*

Método	Ventajas	Desventajas
Uso de conectividad de píxeles.	Simple y directo, robusto a la rotación de la matrícula.	No se pueden extraer todos los caracteres cuando hay personajes unidos o rotos.
Uso búsqueda líneas horizontales o verticales.	Independientemente de las posiciones de carácter, ser capaz de lidiar con alguna rotación.	El ruido afecta al valor de proyección, requiere conocimiento del número de caracteres de matrícula.
Uso de conocimientos previos de caracteres.	Sencillo.	Limitado por el conocimiento previo, cualquier cambio puede producir errores.
Usando contornos de caracteres.	Puede obtener límites exactos de caracteres.	Lento y puede generar un contorno incompleto o distorsionado.
Usando características combinadas.	Más fiable.	Computacionalmente complejo.

Fuente. Adaptada de (S. Du, 2013).

Para concluir, una vez generadas la segmentación de la matricula en caracteres individuales, se procede al reconocimiento de estos, mediante técnicas de correlación de imágenes. Para

ello será necesario crear un bucle donde se compara el carácter a reconocer con cada una de las 26 letras del alfabeto y los diez números naturales.

El punto de partida es un conjunto de imágenes con las letras del alfabeto y números de 0 a 9. Ordenados en una matriz de tres columnas con sus valores RGB por columna. El reconocimiento de cada carácter en RGB se realiza comparando con la matriz anterior y tomando el índice de mayor correlación a través del comando CORR2. Por último, una vez obtenidos los índices de la matriz para cada uno de los caracteres, se cruza con el carácter relativo al índice obteniendo así las letras y números de la matrícula.

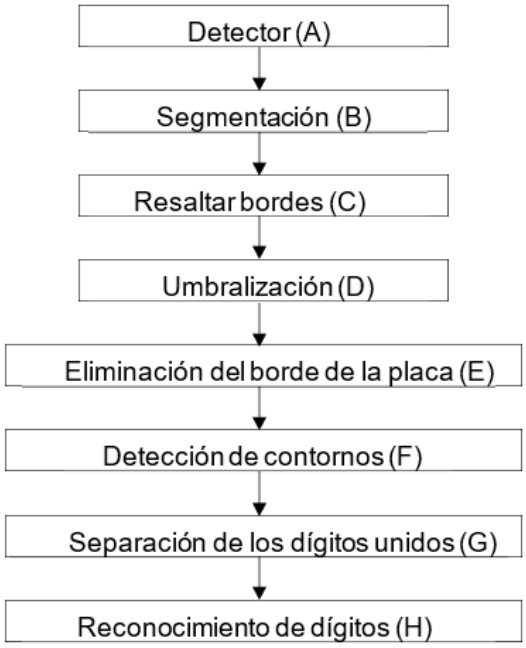
3.1.4. Trabajos relacionados

- Torres Cabrera (Torres Cabrera, 2020) emplea un sistema de reconocimiento automático de placas vehiculares, tomando en cuenta las características de una placa estándar del Ecuador. Para el procesamiento de imágenes, el sistema utiliza binarización Otsu, además del algoritmo de Componentes Conectados para la segmentación y dos redes neuronales para la clasificación (una para la clasificación de letras y otra para la clasificación de números).

  Para la configuración y entrenamiento de las redes neuronales, utiliza los respectivos paquetes de librería de MATLAB; el código en general es escrito en C#, que es un lenguaje de desarrollo de programación construido por Microsoft, basado en la sintaxis de C y C++.

  Las imágenes requeridas para este proyecto se manejan en condiciones relativamente controladas como: la distancia estándar que es de 2 a 3 metros a partir del vehículo, además se requiere de iluminación mediante luz natural y direccionar la cámara de manera que forme un ángulo perpendicular entre la línea visual y la superficie de la placa. Bajo estas condiciones, la tasa de efectividad del sistema es del 97%.
- Corneto, G. L., Silva, F. A., Pereira, D. R., Almeida, L. L., Artero, A. O., Papa, J. P., De Albuquerque, V. H. C., & Sapia, H. M. (2017) presentan un método para la detección y segmentación en tiempo real de matrículas en vehículos, basados en técnicas de análisis y procesamiento de imágenes. La figura 3, detalla el funcionamiento de este trabajo.

Figura 3. *Pasos del Método Propuesto.*



Fuente. Adaptada de:(Corneto et al., 2017).

Tras la fase de entrenamiento se aplica un detector, que permite identificar el objeto de interés. Este detector que esta implementado en OpenCV define un rectángulo y como parámetro su área, el mismo que se utiliza para clasificar la región de interés. Con la región de la placa definida se segmenta los dígitos a reconocer, a través de la mejora de bordes, umbralización y detección de contornos. Además, aplica la separación de los dígitos unidos, esta fase es considerada solo cuando existen dígitos unidos, esto debido a las distorsiones de la imagen, tomando como referencia que cada dígito de más de 60 pixeles de ancho se trata como dígitos unidos. Por último, el reconocimiento de dígitos.

- Gnanaprakash, V., Kanthimathi, N., & Saranya, N. (2021) El trabajo propuesto basado en un algoritmo de aprendizaje profundo (Deep Learning), consta de 4 pasos: El primero, obtiene imágenes de vehículos a partir de secuencias de video. El segundo, detecta el vehículo a partir fotografías, en el siguiente paso se detecta la matrícula y por último el reconocimiento de los caracteres de la placa del vehículo. Este trabajo utiliza dos modelos CNN, uno para detección de coches y otro para detección de matrículas, además utiliza la biblioteca ImageAI, la misma que facilita el proceso de entrenamiento. Este modelo consigue el 97% en la detección de vehículos, una

precisión del 98% para la localización de las matrículas y un 90% para el reconocimiento de caracteres.

Este sistema esta entrenado con NVIDIA Jetson Nano e implementado en Python, genera 5 modelos de detección en formato H5 y un archivo JSON diferentes. Construido para acceder a todas las cámaras utilizando su dirección IP. Su base de datos para detección de matrículas consta de 1000 imágenes, los datos de entrenamiento consisten en 800 imágenes y se anotan utilizando LabelImg. Luego del entrenamiento se realiza la prueba para finalmente realizar la detección.

- Farhat, A., Hommos, O., Al-Zawqari, A., Al-Qahtani, A., Bensaali, F., Amira, A., & Zhai, X. (2018). Es de las técnicas más actuales, proponen el reconocimiento óptico de caracteres en un SoC (sistema heterogéneo en chip) heterogéneo para un sistema de reconocimiento automático de matrículas en alta definición. Para lo cual plantean, cuatro algoritmos para la etapa de OCR (Reconocimiento Óptico de Caracteres) de un sistema de reconocimiento automático de matrículas de alta definición en tiempo real. Estos algoritmos usan técnicas de extracción de características (cruce de vectores, zonificación, zonificación combinada y cruce de vectores) y coincidencia de plantillas. El entorno utilizado para la implementación es MATLAB, a través de una plataforma heterogénea de sistema en chip (SoC) denominada Xilinx Zynq-7000 All Programable SoC, que consta de un procesador ARM y lógica programable. Reconoce un carácter en 0.63 ms, con una precisión del 99.5% y utilizando alrededor del 6% de los recursos de la lógica programable. Es fundamental optimizar la implementación del hardware para lograr una tasa de reconocimiento optima y cumplir con el procesamiento en tiempo real.
- Bedir Yousif, B., Maher Ata, M., Fawzy, N., & Obaya, M. (2020) proponen una nueva metodología para el reconocimiento de matrículas de vehículos egipcios que son uno de los más complejos. El punto de inicio para este trabajo es aplicar algunas técnicas de procesamiento de imágenes, como la detección de bordes y operaciones morfológicas para localizar la matrícula. Se extrae las características más destacadas mediante la implementación de una nueva metodología utilizando algoritmo genético (GA) para optimizar las operaciones del conjunto neutrosófico optimizado (NS). El uso de NS disminuye la indeterminación en las imágenes de la matrícula.

Para segmentar los caracteres de la matricula se ha aplicado el algoritmo de agrupación de k-media. Además, con la aplicación del algoritmo de análisis de etiquetado de componentes conectados (CCLA), identifica las regiones de píxeles conectados y de esta manera agrupa los píxeles apropiados en componentes para extraer cada carácter de manera efectiva.

Los resultados experimentales muestran que la metodología propuesta tiene la capacidad de ser adecuada tanto para matrículas (árabe-egipcio) como para inglés. El sistema propuesto logra un alto grado de precisión de reconocimiento para todo el sistema de acuerdo con los siguientes estudios de caso;

(i) Para un egipcio de alta resolución (LP), el sistema propuesto logra aproximadamente un 96,67% de precisión de reconocimiento correcto.

(ii) Para una matrícula inglés de baja resolución, el sistema propuesto alcanza una precisión del 94,27%. Además, al sistema propuesto intencionalmente se le aplica algún tipo de alteración de la imagen, es decir (flash en la imagen, ruido externo y variación de la iluminación), logrando aproximadamente un 92,5% de precisión en la identificación correcta.

3.2. Reconocimiento automático del color de vehículos

En la actualidad, ante el aumento del parque automotor en las principales urbes de un país, el reconocimiento automático de vehículos juega un rol muy importante en la planificación de sistemas de transporte y configuración de tráfico inteligente (ITS). La principal ventaja para desarrollarlos, son los atributos con los que cuenta un vehículo tales como: la matrícula del vehículo, el color, el modelo y su marca, entre las características principales.

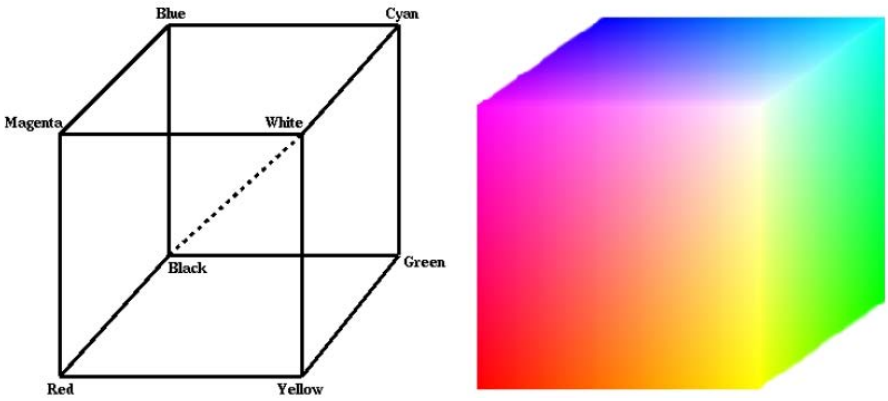
El color es uno de los atributos más estables de los vehículos y, a menudo, se utiliza como una pista valiosa en algunas aplicaciones importantes (Zhang et al., 2017). Pero a pesar de ello, el proceso de reconocimiento del color de los vehículos a través de una imagen no siempre será una tarea fácil, esto debido a factores ambientales complejos, como la iluminación, el clima, el ruido, etc. Añadiendo a esto, la constitución propia de un vehículo cuyas partes como las ventanas, las ruedas y el fondo en el que se genere la imagen no dan información relevante de color, lo que impide tener una precisión optima en el reconocimiento.

3.2.1. Espacio o modelo de color

Se define como un modelo matemático abstracto, representados mediante tuplas de números, normalmente como tres o cuatro valores o componentes de color. Se originan de modelos de sistemas visuales (RGB, espacio de colores oponentes, IHS), también de adaptaciones del dominio técnico (colorimetría: XYZ, televisión: YUV, etc.) o desarrollados especialmente para el procesamiento de imágenes (Caldas et al., 2005). A continuación, se detalla el espacio RGB utilizado en el presente trabajo.

Espacio de color RGB, llamado así por sus siglas en ingles Red, Green, Blue, es un modelo sensorial caracterizado por representar cada color por sus tres componentes espectrales primarias de rojo, verde y azul(Lillo, 2010). El subespacio de color asociado a este modelo se detalla en la figura 4, donde se observa que tres de las esquinas corresponden con los colores primarios rojo, verde y azul; tres esquinas más con los colores secundarios magenta, cian y amarillo; y el origen y la esquina más alejada a este representan los colores acromáticos negro y blanco, respectivamente. En este modelo, la escala de grises se extiende a lo largo de la línea que une blanco y negro atravesando el volumen del cubo, correspondiéndose con aquellos colores que tienen valores idénticos de las componentes roja, verde y azul.

Figura 4. *Representación Gráfica del Espacio de Color RGB.*

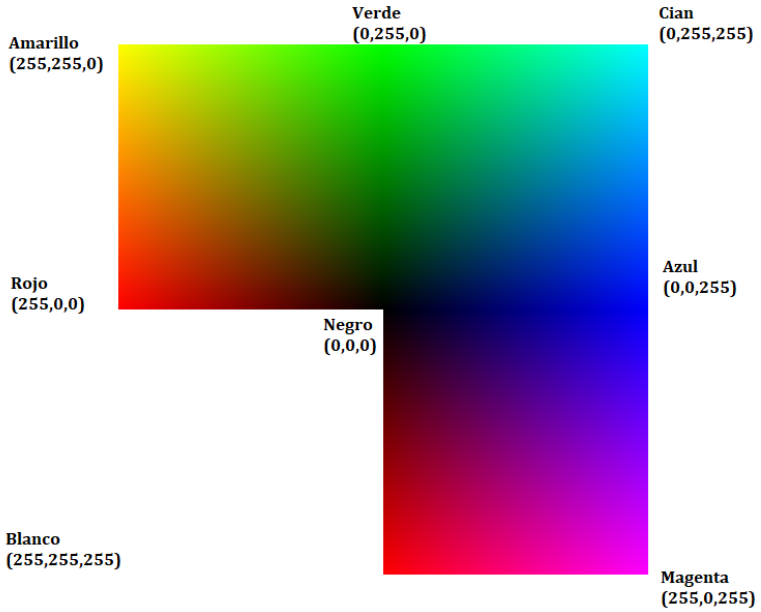


Fuente. Tomada de (Lillo, 2010).

La variedad de colores existentes influye en la facilidad de interpretación y análisis de elementos, además que permite dar sentido de atracción hacia un elemento con un determinado color. Para conseguir esta variedad de colores, se puede crear una mezcla de cada color, asignando una numeración a cada uno de los colores primarios. El valor 0 significará que no actúa en la mezcla y su valor en aumento indicará la intensidad del color.

Para especificar la numeración a cada color primario, se codifica con un byte (8 bits), de esta forma el rango en la intensidad de cada una de las componentes se mide según una escala que va del 0 al 255, representados mediante paréntesis (correspondientes a valores "R", "G" y "B") y separados por comas. En la figura 5, se puede observar la configuración con distintas numeraciones para cada color primario a través de diferentes mezclas.

Figura 5. *Modelo de Color RGB.*



Fuente. Adaptada de: (Marc Mongenet,2005).

Tomando como referencia lo antes detallado, se hace necesario categorizar los colores para que a partir de una clasificación se pueda extraer información de forma más precisa. La categorización del color se ha estudiado ampliamente como una ventana al lenguaje y la cognición humana, y se ha utilizado pragmáticamente por separado en sistemas de recuperación de bases de datos de imágenes. Esto sugiere la hipótesis de que el mejor sistema de categorías para propósitos pragmáticos coincide con las categorías humanas (es decir, los colores básicos) (Griffin, 2006). Estos colores básicos se asientan sobre el estudio de Berlín & Kay en donde afirman que los colores básicos de cualquier idioma siempre se extraen de un inventario universal de once: negro, gris, blanco, rojo, naranja, amarillo, verde, azul, violeta, rosa y marrón. Estos once colores se denominan a menudo colores básicos.

3.2.2. Técnicas de clasificación multiclase

En la actualidad el auge tecnológico y la revolución digital han generado un incremento en el número de datos que se debe clasificar y transmitir. Hacerle frente a esta realidad con métodos tradicionales, resultaría una tarea imposible. Para ello en los últimos años, se ha venido adaptando y desarrollando software que optimizan y facilitan estas tareas.

En el estudio de Mohamed (2017), se hace una comparativa acerca de cuatro técnicas de aprendizaje automático supervisado para clasificación, se resume cada técnica a continuación.

- KNN, K Vecinos más Cercanos o en inglés (K-nearest neighbors algorithm), se fundamenta en la densidad de probabilidades. Su tarea es clasificar los ejemplos no vistos basándose en la base de datos almacenada. Las observaciones se presentan en un espacio dimensional.

  Dado un nuevo punto, se clasifica según su similitud con el resto de los puntos de datos almacenados en el modelo. El algoritmo decide la clase del nuevo punto escogido a través de los K puntos más cercanos al nuevo ejemplo y toma la clase más común entre ellos por el voto mayoritario para hacer la clase del nuevo punto. Si=1, el nuevo punto se clasificará según el punto de datos más cercanos al nuevo ejemplo. Si=2, la clase del nuevo punto elegirá entre los dos vecinos más cercanos y la votación no ayudará. La votación ayuda a partir de 3, clasificando el nuevo punto según la clase más común entre estos vecinos más cercanos
- ANN, Redes Neuronales Artificiales o en inglés (Artificial neural network): Es un modelo matemático que intenta simular la funcionalidad del sistema nervioso biológico. Este modelo matemático tiene 3 reglas básicas: multiplicación suma y activación. Básicamente se trata de entradas ponderadas lo que significa que cada valor de entrada se multiplica por un peso específico. A continuación, todas las entradas ponderadas se suman con un término de sesgo. Al final la suma de todas las entradas ponderadas y el término de sesgo será transformada por una función de activación para calcular la salida. Los pesos asociados a cada entrada proporcionan la fuerza de la sinapsis. Cuanto más alto sea el peso asociado a una entrada específica más fuerte será la entrada, estos pesos pueden ser positivos o negativos. Cuando el peso es positivo indica una conexión excitatoria, mientras que un peso negativo inhibe la actividad de la neurona. El elemento básico de procesamiento se llama se perceptrón.

- SVM, Máquinas de Vectores de Soporte o en inglés (Support Vector Machine): esta técnica está basado en observaciones para calcular el hiperplano no lineal más discriminativo entre clases. Utiliza el concepto de margen, que es la esencia del método. Hay un compromiso entre maximizar el margen y minimizar el número de ejemplos mal clasificados. Existen algunos límites que rigen la relación entre el rendimiento del modelo y su capacidad. Esto puede utilizarse para equilibrar el compromiso entre el sesgo y la varianza del modelo.
- Árboles de decisión o en inglés (Random Forest): en el ámbito de la estadística y aprendizaje automático son los más utilizados. Se implementa bajo un diseño jerárquico cuyo enfoque es divide y vencerás. Es una técnica no paramétrica que se utiliza tanto para la clasificación como para la regresión. Se puede convertir directamente en un conjunto de reglas simples sí-entonces. Para este clasificador se determina 100 árboles de decisión a generar.

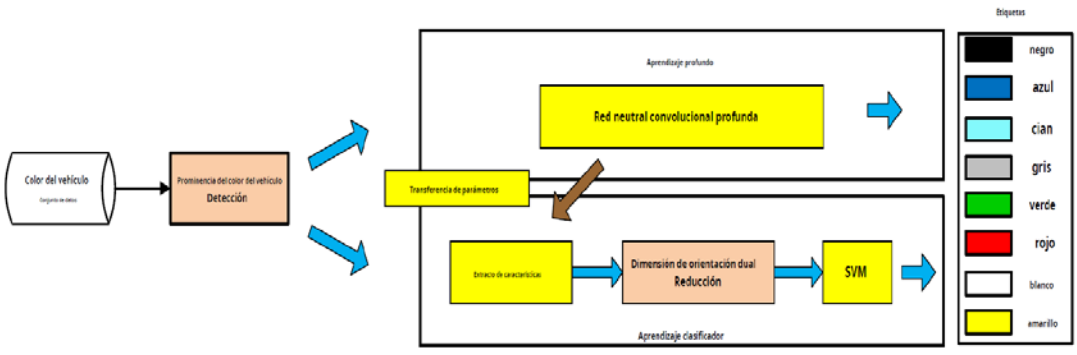
Además, de estas 4 técnicas podemos mencionar también el siguiente:

- PC, Clasificador Parzen o en inglés (Parzen´s Classifier): está relacionado con probabilidades y requiere un parámetro de suavizado para la distribución Gaussiana que debe ser optimizado.

3.2.3. Trabajos Relacionados

- Zhang et al (2017) El concepto de la propuesta se puede observar en la figura 6, la misma que se divide en cuatro partes: detección de prominencia del color del vehículo, extracción profunda de características, reducción de dimensionalidad de orientación dual y entrenamiento de clasificadores.

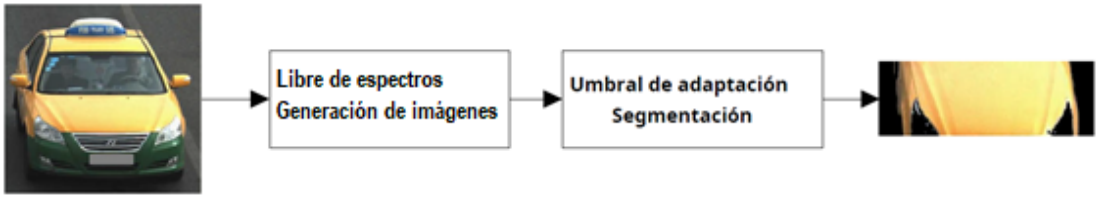
Figura 6. *Marco Metodológico para el Reconocimiento del Color de Vehículos.*



Fuente. Adaptada de (Zhang et al., 2017).

En primer lugar, la imagen sin especular se extrae de la imagen original, que puede resaltar eficazmente las características visuales del color del vehículo, y luego se detectan las regiones de prominencia del color del vehículo de acuerdo con la distribución de información de color de la imagen sin especular, detallado en la figura 7. Las subimágenes correspondientes a las regiones de prominencia del color del vehículo se ingresan en CNN para obtener las características profundas mediante el uso de preentrenamiento. Se puede generar un mapa de características profundas 2D de alta dimensión después de repetidas iteraciones. Finalmente, SVM es utilizado como clasificador para entrenar el modelo de reconocimiento en las características de dimensionalidad reducida.

Figura 7. *Diagrama de Detección de Prominencia del Vehículo.*



Fuente. Adaptada de (Zhang et al., 2017).

- Rotaru et al (2008) propone un método para clasificar puntos de color para aplicaciones automotrices en el espacio de intensidad de saturación de tono (HSI) basado en las distancias entre sus proyecciones en el plano SI. En primer lugar, se analiza en detalle el HSI Space. En segundo lugar, se muestra la proyección de puntos de imagen de una escena automotriz típica en el plano SI. Se derivan las clases mínimas relevantes para las aplicaciones de asistencia al conductor. Se obtienen los requisitos para la clasificación de los puntos en esas clases. Se proponen varias funciones de ponderación y se investiga en detalle una forma rápida de métrica euclidiana. Para mejorar la sensibilidad de la función de ponderación, se introducen coeficientes dinámicos. Se muestra cómo calcularlos automáticamente para obtener resultados óptimos para la clasificación.
- Korznikov et al.(2021) en su trabajo plantea como objetivo, el reconocimiento de árboles individuales sobre la base de imágenes de satélite RGB (rojo, verde, azul) de muy alta resolución, utilizando enfoques de aprendizaje profundo para los bosques mixtos templados del norte en la región de Primorsky del Lejano Oriente ruso.

Utilizan una imagen RGB satelital pansharpened de GeoEye-1 con una resolución espacial de 0.46 m / píxel. Los autores, parametrizan la red neuronal convolucional (CNN) U-Net estándar y la entrenan con imágenes satelitales delineadas manualmente para resolver el problema de la segmentación de imágenes de satélite.

Para fines de comparación aplican algoritmos de clasificación estándar basados en píxeles, como el k-clasificador de vecino más cercano, clasificador de Bayes ingenuo y discriminación cuadrática. La precisión del reconocimiento se estima mediante varias métricas y supera los valores obtenidos para los enfoques estándar de aprendizaje automático.

A diferencia de los algoritmos de reconocimiento basados en píxeles, la CNN similar a U-Net no conduce a un aumento en las decisiones positivas falsas cuando se enfrentan a objetos de color verde que son similares a los árboles. Por medio de CNN similar a U-Net, se obtiene una puntuación media de precisión de hasta 0,96 en experimentos computacionales. La CNN similar a U-Net reconoce las copas de los árboles no como un conjunto de píxeles con intensidades RGB conocidas, sino como objetos espaciales con una geometría y un patrón específicos.

- Gu & Lee (2012) Este estudio propone un algoritmo inteligente con arquitectura de tres estados para la extracción de carrocerías y clasificación de colores en tiempo real. El algoritmo es capaz de gestionar las dificultades como la reflexión de la luz.

  Debido a que la influencia del reflejo de la luz es significativamente diferente en autos brillantes, oscuros y de colores, se diseñan tres estrategias diferentes para varias categorías de color, para adquirir una carrocería más intacta. Se propone un algoritmo SARM (Separating and Re-Merging) para separar la carrocería del vehículo y el fondo, y recuperar toda la carrocería de forma más completa. También se realiza un algoritmo de selección robusto para determinar la categoría de color y la carrocería correctas.

  El tipo de color del vehículo se decide sólo por los píxeles en la carrocería extraída. Los resultados experimentales muestran que el método de tres estados puede extraer casi el 90% de los píxeles de la carrocería de una imagen de automóvil. Más del 98% de las imágenes de automóviles, se distinguen correctamente en sus categorías, y la precisión promedio de la clasificación de tipo de 10 colores es superior al 93%. Además, la carga de cálculo del método propuesto es ligera; por lo tanto, es aplicable para sistemas en tiempo real.

3.3. Reconocimiento automático del modelo de vehículo

La detección y coincidencia de características o elementos de interés (Feature detection and matching) es parte de muchas aplicaciones de computer vision, tales como seguimientos de objetos, reconocimiento de objetos, computado de disparidad, segmentado en base al movimiento, reconstrucción 3D, navegación de robots, entre otras.

El concepto de FEATURE se explica como una característica o elemento de interés a obtener mediante una tarea computacional relacionada con una tarea determinada. Se pueden clasificar en dos: en patrones de puntos o esquinas características y en bordes o contornos característicos.

Los principales componentes de la detección de patrones y macheo son:

- Detección. Identificar los puntos de interés.
- Elementos descriptivos. Suelen ser elementos invariantes a cambios en la iluminación, translación, escala y/o rotación en el plano.
- Macheo. Proceso por el cual se comparan los diferentes puntos descriptivos con objeto de buscar similitudes entre ambos.

3.3.1. Puntos de interés

Podemos definir punto de interés a un elemento con una posición bien definida en el espacio de la imagen, es estable frente a perturbaciones locales y externas, con lo que puede ser usado de manera eficiente para la detección. En la figura 8 se observa la detección de estos puntos de interés.

Figura 8. *Detección de Puntos de Interés.*

Fuente. Elaboración Propia.

El problema o tarea se puede afrontar de diversas maneras tales como métodos basados en el brillo de la imagen (generalmente por derivada de la imagen) y métodos basados en extracción de límites (generalmente por detección de bordes y análisis de curvatura).

Existen diversos algoritmos para la identificación, tales como:

- Harris Corner. (Harris & Stephens, 1998).
- Scale Invariant Feature Transform – SIFT. (Lowe, 2004).
- Features from Accelerated Segment Test – FASR. (Rosten & Drummond, 2206).
- Oriented FAST and Rotated BRIEF – ORB. (E. Rublee, 2011)
- Speeded Up Robust Feature – SURF. (Bay & Tuytelaars, 2006)

3.3.2. Elementos descriptivos

Los diferentes elementos descriptivos serán obtenidos por un algoritmo que toma una imagen como entrada. Estos elementos son la información relevante de cada imagen que hacen que se pueda diferenciar una característica de otra entre imágenes.

Estos elementos, deberán ser como se dijo anteriormente, invariantes frente a una transformación de la imagen. Pudiendo ser locales específicos a un elemento o globales donde describen la imagen.

Los algoritmos más utilizados son:

- Binary Robust Invariant Scalable Keypoints - BRISK
- Binary Robust Independent Elementary Features – BRIEF
- Oriented FAST and Rotated BRIEF – ORB. (E. Rublee, 2011)
- Scale Invariant Feature Transform -SIFT. (Lowe, 2004)
- Speeded Up Robust Feature – SURF. (Bay & Tuytelaars, 2006)

3.3.3. Secuencia macheo

La secuencia para la detección de patrones y macheo es:

1. Encontrar puntos de interés.
2. Definir una región alrededor del mismo.
3. Extraer y normalizar el contenido en la región anteriormente definida
4. Obtener un elemento descriptivo del contenido anteriormente normalizado.
5. Comparar los elementos descriptivos locales entre las imágenes a evaluar.

3.3.4. Speeded Up Robust Feature - SURF

El método SURF (Bay & Tuytelaars, 2006) es un algoritmo rápido y robusto para la representación y comparación de imágenes locales, invariantes en similitudes. De manera análoga a muchos otros enfoques basados en descriptores locales, los puntos de interés de una imagen dada se definen como características sobresalientes de una representación invariante de escala. Este análisis de múltiples escalas se obtiene mediante la convolución de la imagen inicial con núcleos discretos a varias escalas (filtros de caja). El segundo paso consiste en construir descriptores invariantes de orientación, utilizando estadísticas de gradiente local (intensidad y orientación). El principal interés del enfoque SURF radica en su rápido cálculo de los operadores mediante filtros de caja, lo que permite aplicaciones en tiempo real como el rastreo y el reconocimiento de objetos.

3.3.5. Trabajos previos

Antes de que esta aplicación viese la luz en 2006, varios autores han ido publicando diversas publicaciones que fueron los pioneros de SURF. El primero fue (Harris & Stephens, 1998) que introdujo el método para la detección de puntos de interés usando los valores Eigen de la matriz momento de segundo orden, aunque este método no era invariante según la escala.

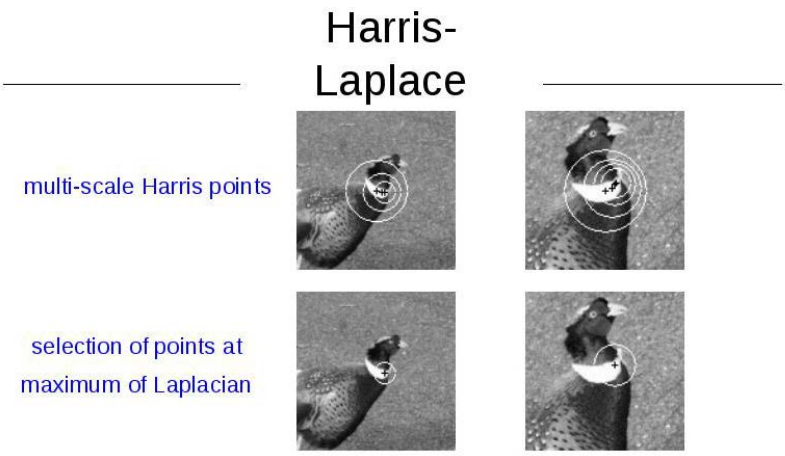
$$\boldsymbol{M} = \sum_{x,y} \boldsymbol{w}(\boldsymbol{x},\boldsymbol{y}) \begin{bmatrix} \boldsymbol{I_x^2} & \boldsymbol{I_x I_Y} \\ \boldsymbol{I_x I_y} & \boldsymbol{I_y^2} \end{bmatrix}$$ **, Ecuación 1.** Matriz Momento.

El siguiente fue (Lindeberg, 1198), que introdujo el concepto de detección automática de escala usando el laplaciano de Gauss y el determinante de la matriz Hessiana.

$$\boldsymbol{\nabla^2 G}(\boldsymbol{x},\boldsymbol{y}) = \frac{\boldsymbol{\partial^2}}{\boldsymbol{\partial_x^2}} \boldsymbol{G}(\boldsymbol{x},\boldsymbol{y}) + \frac{\boldsymbol{\partial^2}}{\boldsymbol{\partial_y^2}} \boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})$$ **, Ecuación 2.** Laplaciano de Gauss.

El siguiente fue el Harris – Laplace detector (Mikolajczyk, 2001), donde se introduce el detector de gradiente de escala, donde se mejoró el Harris detector (Harris & Stephens, 1998) usando la combinación de la detección de Harris Corner (Harris & Stephens, 1998), la detección de puntos de interés a través del determinante Hessiano y el laplaciano para la detección de la escala.

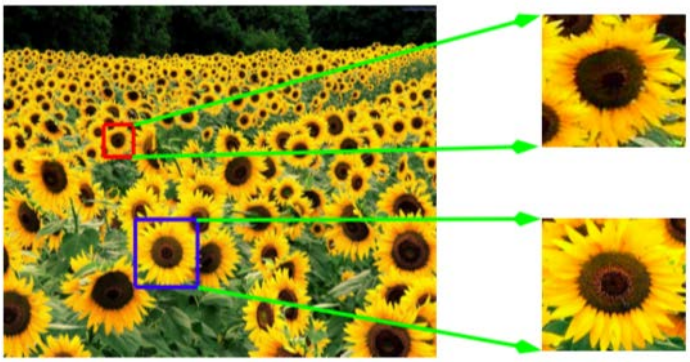
Figura 9. *Puntos Invariantes.*



Fuente. Adaptada de (Mikolajczyk, 2001).

Ya en 2004, Lowe, D. introdujo SIFT, una metodología para un cálculo más rápido del laplaciano de Gauss (Lowe, 2004), a través de aproximaciones de la diferencia gaussiana. Este algoritmo fue ampliamente utilizado, aunque presentaba problemas al tener 128 dimensiones y era demasiado grande.

Figura 10. *Scale Invariant Feature Transform – SIFT.*



Fuente. Tomada de Minghao Ning. (https://towardsdatascience.com/).

3.3.5.1. Metodología.

Una visión general del método se presenta en la tabla 4.

Tabla 4. *Visión General del Método SURF*

Detector (features)	Descriptor. (Puntos descriptivos).
Basado en matriz Hesiana de los puntos de interés	Describe la distribución de la respuesta de Haar wavelet con los puntos de interés cercanos.
Usa una aproximación de la diferencia Gaussiana (DoG)	Solo usa 64 dimensiones
Usa una imagen integral	Introduce un nuevo paso de indexado que es base de la señal laplaciana

Fuente. Elaboración Propia.

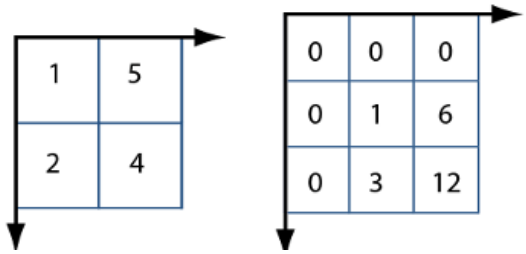
SURF (Bay & Tuytelaars, 2006) está compuesto de dos pasos:

A. Extracción de patrones (features).
B. Descripción de patrones (features).

3.3.5.2. Extracción de puntos de interés.

El concepto de imagen integral figura 11, se explica sustituyendo cada píxel de la imagen original con la suma de los pixeles que están arriba y a la izquierda de este, al hacer un barrido por la imagen. Píxel a píxel.

Figura 11. *Imagen Integral.*

Imágenes integrales	Formula
 Imagen de entrada Imagen integral	$ii(x, y = \sum_{x' \leq x, y' \leq y} i(x', y')$ La imagen integral en la localización (x, y) contiene la suma de los pixeles arriba y a la izquierda de (x, y), incluyendo estos últimos

Fuente. Elaboración Propia.

Objetos rectangulares pueden ser computados muy rápidamente usando la imagen integral. Como dijimos anteriormente, el detector rápido Hessiano se realiza mediante el valor del

determinante de la matriz Hessiana para cada píxel (Lowe, 2004). Que es computado usando la imagen integral, para la detección de los puntos clave y la detección de la escala.

$$\mathcal{H}(\boldsymbol{X},\boldsymbol{\sigma}) = \begin{bmatrix} \boldsymbol{L_{xx}}(\boldsymbol{X},\boldsymbol{\sigma}) & \boldsymbol{L_{xy}}(\boldsymbol{X},\boldsymbol{\sigma}) \\ \boldsymbol{L_{xy}}(\boldsymbol{X},\boldsymbol{\sigma}) & \boldsymbol{L_{yy}}(\boldsymbol{X},\boldsymbol{\sigma}) \end{bmatrix}$$, **Ecuación 3.** Matriz Hessiana.

Donde X es la localización del píxel, σ la escala, L_{xx},L_{yy} y L_{xy} son las convoluciones de la derivada de segundo orden de la matriz Gaussiana con la imagen integral.

Los Gaussianos son considerados ideales para el análisis de la escala (Lindeberg T., 2003), pero en la práctica, las funciones continuas gaussianas son cortadas y discretizadas, aunque incluso con un filtrado cuidadoso se presenta fenómenos de aliasing. De ahí que se introduce un segundo nivel de aproximación mediante filtrado de cajas a través de derivadas gaussianas de segundo orden.

Para calcular el determinante de la matriz Hessiana, primero debemos aplicar la convolución con el núcleo de Gauss y luego la derivada de segundo orden. Después del éxito de Lowe's con las aproximaciones de LoG (SIFT), SURF impulsa la aproximación (tanto la convolución como la derivada de segundo orden) aún más con los filtros de caja. Estas derivadas gaussianas de segundo orden aproximadas pueden evaluarse a un costo computacional muy bajo, utilizando imágenes integrales e independientemente del tamaño, y esto es parte de la razón por la que SURF es rápido.

3.3.5.3. Descripción de patrones.

La descripción de patrones se lleva a cabo en dos pasos: El primero, fijando una orientación alrededor de una zona circular del punto de interés, para crear una región cuadrada alineada con la anterior orientación de donde se extrae el patrón. La segunda, mediante la asignación de la orientación con la finalidad de ser invariante a la rotación. En este caso, SURF intenta identificar una orientación reproducible para los puntos de interés. Para lograr esto:

- Surf primero calcula las respuestas de las ondas de Haar-wavelet en las direcciones x e y, en una zona circular de radio 6s alrededor del punto de interés, con s la escala en la que se detectó el punto clave.
- Luego calculamos la suma de las respuestas de ondas verticales y horizontales en un área de escaneo, se cambia la orientación de escaneo (agregando π/3) y se vuelve a

calcular, hasta que encontremos la orientación con el valor de suma más grande, esta orientación es la orientación principal de descriptor de patrones.

En cuanto a la extracción de los patrones, el primer paso es construir como dijimos anteriormente una región cuadrada centrada en el punto de interés de tamaño 20s x 20s. Esta región se divide en un segundo paso en regiones más pequeñas de 4s x 4s para incrementar la robustez del método.

3.4. Análisis final del estado del arte

Luego de analizar, el estado del arte con respecto al ámbito de estudio se evidencia que el tema propuesto presenta bastante interés y es así como se observa diversos trabajos previos o similares, todos enmarcados a obtener eficiencia en el reconocimiento de las características de un vehículo y especializándose en un solo tipo de reconocimiento. Haciendo mención de lo manifestado, se puede decir que, si bien esta propuesta no presenta técnicas de reconocimiento de lo más eficientes, se destaca con respecto a los demás trabajos en la usabilidad, debido a que los tres tipos de reconocimiento, placa, color y modelo del vehículo están compilados en uno solo, permitiendo al usuario obtener una información más completa del automotor que ocasionase el accidente. Además, que el algoritmo propuesto está pensado para que sea una herramienta intuitiva y de fácil manipulación.

4. Identificación de requisitos

La aplicación por desarrollarse debe considerar un entorno que posee diversas ventanas en las cuales le permitirá al usuario una navegación libre, para dar solución a las diversas etapas de identificación que posee el sistema.

Una primera ventana contará con datos informativos tales como nombre de la Universidad, el estudio desarrollado, una imagen alusiva a la temática a desarrollarse, los nombres de los integrantes del trabajo de fin de máster y botones de navegación que permitan cerrar la aplicación o mostrar una nueva ventana del desarrollo. Una nueva ventana permitirá cargar la imagen que se desea analizar en el sistema, el usuario deberá escoger la captura que permitirá identificar los parámetros establecidos, tales como, los caracteres de la placa vehicular, el color del vehículo y el modelo de este.

Si el usuario o individuo que haga uso de la aplicación selecciona el botón de Placa Vehicular se deberá mostrar un nuevo GUIDE en el cual se le mostrará un cuadro de dialogo en el cual se indicará las acciones que debe realizar para obtener el resultado deseado, se le pedirá que cargue la imagen anteriormente seleccionada para proceder al análisis de esta información. El usuario deberá seleccionar los caracteres que conforman la placa, es decir, las letras y números identificativos de cada móvil sin distractores de manera que se apliquen un recorte del área seleccionada, el mismo que será sometido a una etapa de preprocesamiento aplicando conversión del tipo de imágenes, aplicación de operaciones lógicas y morfológicas para eliminar los pixeles aislados y obtener una identificación plena de los caracteres.

Al seleccionar el botón de Color se mostrará una nueva ventana la cual que identificará el color del vehículo considerando dos formas de comparación; la primera será mediante una comparación entre el vehículo escogido por el operador del sistema y una base de datos establecida en este GUIDE en el cual para permitir que se identifique el color del vehículo de forma acertada se le asignará un rango de tolerancia a cada color considerado. El segundo método de identificación será aplicando un clasificador el cual dispone de una cantidad considerable de datos que serán entrenados para obtener el clasificador óptimo y que permita mostrar la información de forma acertada.

La última etapa que ha desarrollarse en el sistema corresponde al reconocimiento del Modelo en esta fase se empleará una detección, extracción y comparación de características del móvil

seleccionado en la ventana de Menú con una base de datos formada por vehículos de una determinada marca del mercado local que se considera que posee una gran aceptación. Se deberán visualizar la base de datos, así como el móvil que se va analizar, además de la representación de los puntos identificados en los vehículos en una nueva imagen.

En todos los GUIDES a desarrollarse deberán poseer botones que permitan realizar una navegación sencilla e intuitiva entre todas las ventanas, así como un botón que finalice la ejecución del programa en cualquier momento que considere necesario el operador o usuario del sistema.

5. Descripción del sistema de software desarrollado

Para el desarrollo propuesto se emplea el Software MATLAB®, pues permite un desarrollo intuitivo y que es capaz de satisfacer las necesidades establecidas en la temática planteada.

El diseño propuesto está basado en determinar características propias de los vehículos que pueden causar un accidente de tránsito y abandonar el lugar de la escena, por lo cual se ha desarrollado un GUIDE que posee tres etapas de identificación y reconocimiento, la primera de ellas consiste en detectar los elementos de la placa de un vehículo, una segunda etapa permite identificar el color de móvil y la tercera etapa realiza la identificación y reconocimiento del modelo del automotor.

Cada una de las etapas anteriormente descritas posee su propio GUIDE, en la cual se identifican y muestran datos e información correspondiente al objetivo que se desarrolla, los diversos entornos permiten realizar una navegación por las diversas ventanas desarrolladas generando un ambiente muy amigable al usuario del sistema.

5.1. GUIDE de Presentación

En la primera ventana desarrollada para el sistema se pueden identificar datos informativos del sistema como: el nombre de la Universidad, el nombre de Máster, un Axis que permitirá colocar una imagen que hace alusión a la temática propuesta y finalmente se muestran los nombres de las personas que forman parte del trabajo de fin de Máster.

En la figura 12 se muestra el desarrollo del GUIDE denominado como Portada. además de lo anteriormente expuesto se colocaron botones que permiten ingresar al sistema o finalizar la ejecución del programa.

Figura 12. *GUIDE Portada.*

Fuente. Elaboración Propia.

En base al diseño desarrollado se obtiene como resultado la Figura 13, en la cual se visualiza la estructura que el usuario encontrará al ejecutar el programa.

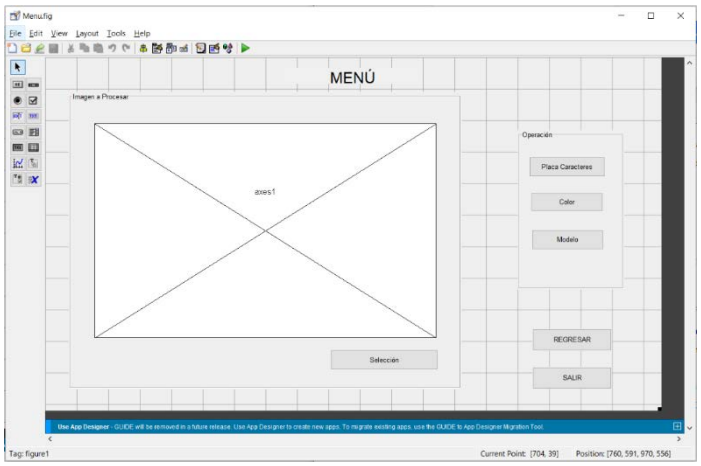
Figura 13. *GUIDE Portada Ejecución Programa.*

Fuente. Elaboración Propia.

5.2. GUIDE de Procesado

El siguiente GUIDE desarrollado fue Menú, en este se incluye un Axis el cual permitirá ingresar la imagen que se desea procesar para lo cual dispone de un botón denominado “Selección”, al dar clic sobre este abrirá la carpeta sobre la que se desea localizar la imagen. Existe un bloque de botones con el Nombre Operación que abrirán nuevas ventanas en las que se podrá realizar la identificación de los caracteres de la placa, reconocer el color o a su vez identificar el modelo del vehículo. Se adicionan botones que permiten la navegación en el sistema, es decir, facultan al usuario regresar a la portada o salir del sistema. En la Figura 14 se muestra el esquemático del GUIDE.

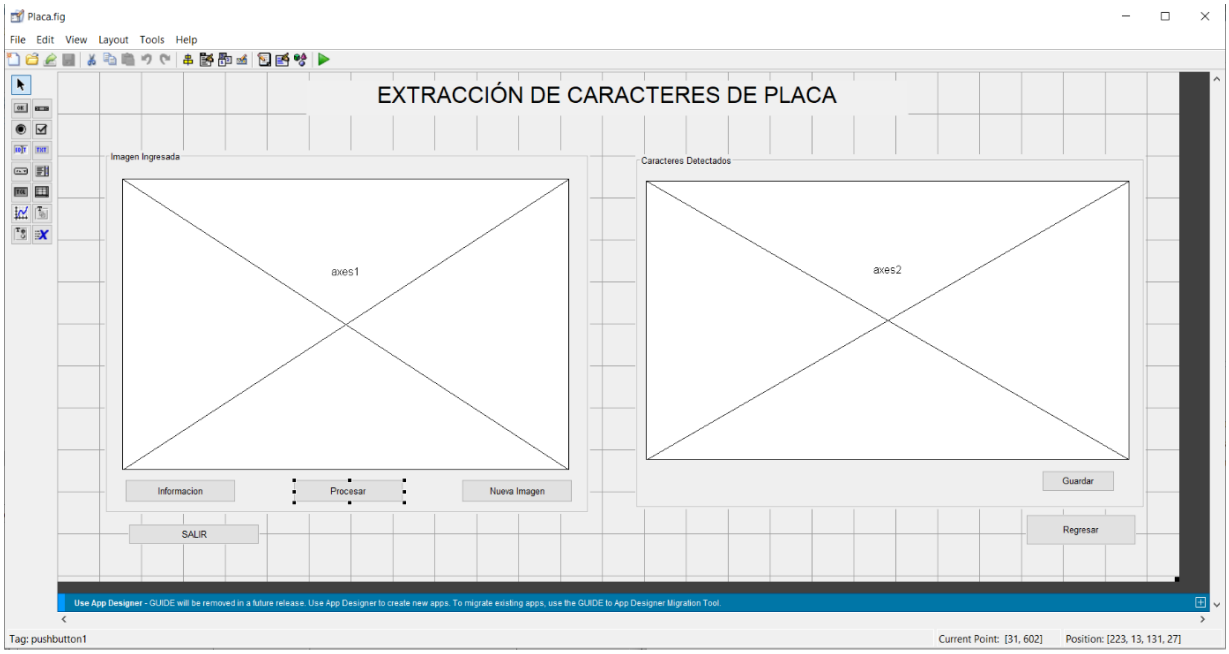
Figura 14. *GUIDE Menú.*



Fuente. Elaboración Propia.

En la ejecución del programa al seleccionar el botón “Placa Caracteres” se despliega un nuevo entorno de desarrollo Gráfico, el mismo que está configurado para que se pueda cargar la imagen ingresada en cuya parte inferior se colocan los botones de “Información”, que desplegará un mensaje indicando cuales son los pasos a seguir en el procesamiento de la imagen. El botón “Procesar”, que se encarga del recorte y análisis de elementos detectados en la placa del móvil. El botón “Nueva Imagen” que permite navegar en carpetas para seleccionar otra imagen para un nuevo procesamiento. Para visualizar el resultado del procesamiento realizado el procesamiento digital de la imagen se colocó un bloque de Caracteres detectados, aquí se podrá visualizar todos y cada uno de los elementos detectados en la placa, en este bloque se adiciona el botón “Guardar” que almacenará los caracteres detectados. Lo expuesto se muestra en la Figura 15.

Figura 15. *GUIDE Visualización Caracteres Placa_1.*



Fuente. Elaboración Propia.

En la Figura 16, se muestra la ejecución de este GUIDE ya con información real que se puede obtener de un dispositivo de captura, en esta figura se identifica el proceso de reconocimiento de los dígitos de la placa ingresada.

Figura 16. *GUIDE Visualización Caracteres Placa_2.*

Fuente. Elaboración Propia.

Para lograr este resultado se realizaron varias etapas de procesamiento digital de imágenes, entre las cuales se realiza el corte del área de interés, es decir, se selecciona las letras y números que conforman la identificación única del vehículo. El resultado de esta operación sobre la imagen ingresada se visualiza en la Figura 17.

Figura 17. *GUIDE. Procesamiento Imagen_1*

Fuente. Elaboración Propia.

Para que el procesamiento no tenga un costo computacional alto, se realiza la conversión del tipo de imagen. Se la transforma de una imagen RGB a una imagen en escala de grises, adicionalmente se ajusta la calidad de la imagen, el resultado del proceso mencionado se lo muestra en la Figura 18.

Figura 18. *GUIDE. Procesamiento Imagen_2.*

Fuente. Elaboración Propia.

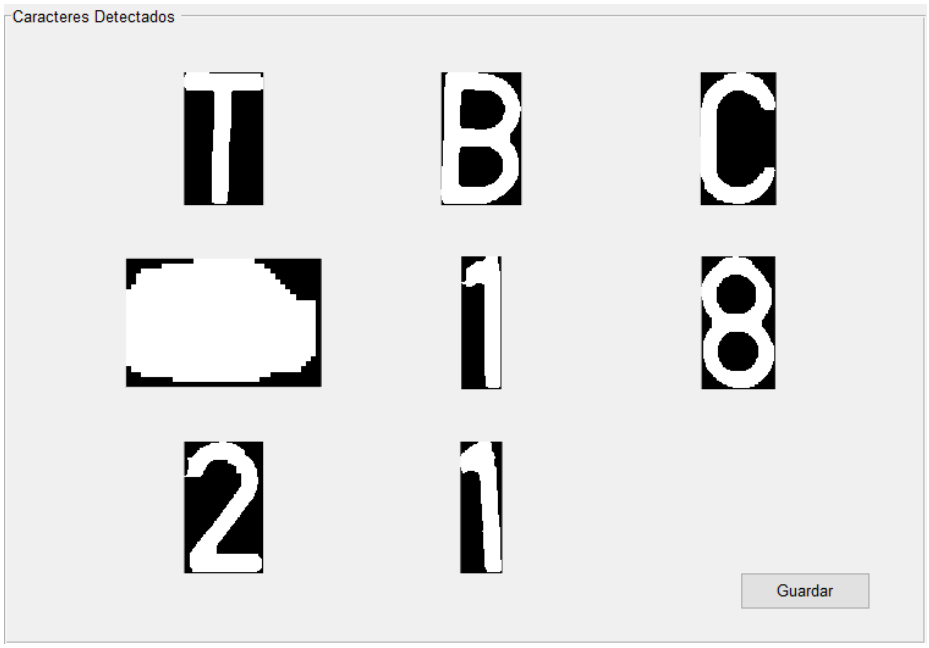
El siguiente paso fue el cálculo del umbral de la imagen anterior, este valor se calcula para binarizar la captura procesada, es decir, que la captura sólo esté conformada por valores de 0 y 1. Una vez que está binarizada se niega la imagen para obtener como resultado el fondo de la placa de color negro, las letras y dígitos identificadores de color blanco, lo que facilita la etapa de localización de los objetos. En la figura 19, se observa este resultado.

Figura 19. *GUIDE. Procesamiento imagen_3.*

Fuente. Elaboración Propia.

En este punto del desarrollo se aplican un elemento estructurante y la operación de cierre en la imagen, se les realiza para tratar de eliminar cualquier píxel aislado que posee la captura y que puede afectar a la localización de los elementos de la placa, cabe mencionar que esto permite identificar de forma clara y precisa las letras y dígitos. Después se recortan los elementos seleccionados y se muestran como resultado en el GUIDE denominado Placa, tal como se evidencia en la Figura 20.

Figura 20. *GUIDE Caracteres Detectados.*



Fuente. Elaboración Propia.

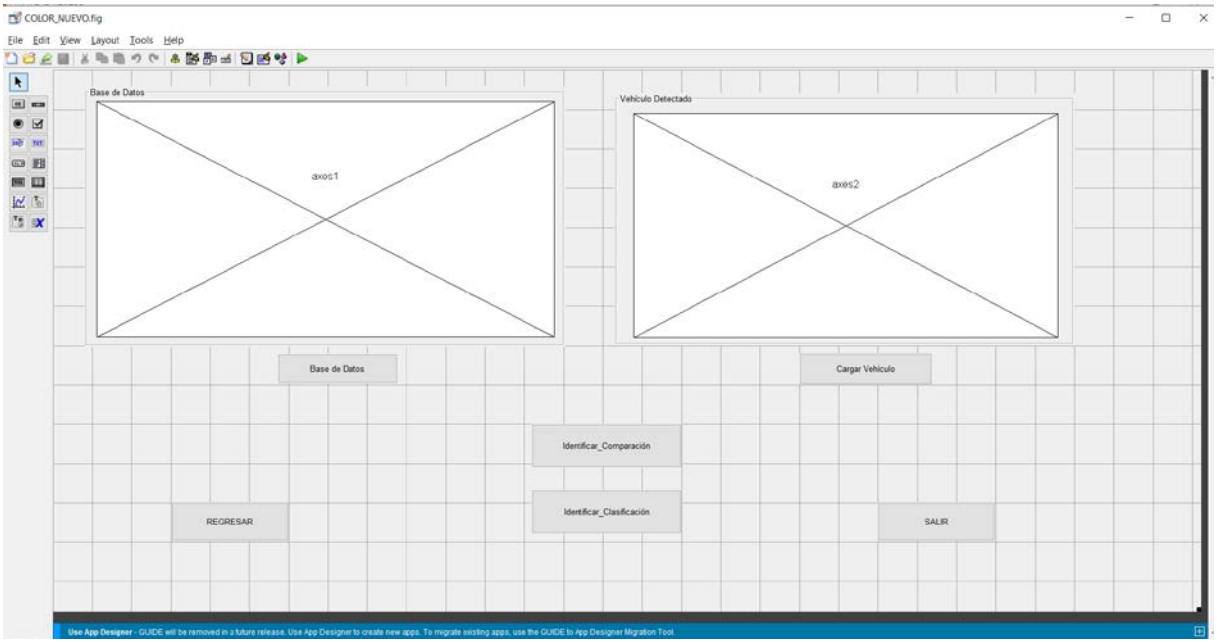
De forma adicional, se ha desarrolló el GUIDE denominado COLOR_NUEVO, con esta programación se pretende que el algoritmo logre la identificación del color del vehículo analizado. En esta pantalla el usuario dispone de 2 axis, los cuales son empleados para mostrar los colores de la base de datos y el móvil que se analiza, se dispone de 6 botones; el botón de Base de Datos muestra los colores almacenados, los mismos que corresponden a los colores típicos de la marca Chevrolet disponibles para todos los modelos de vehículos que la marca dispone en el país, estos permiten realizar la respectiva identificación del color. El botón Cargar Vehículo muestra el móvil que se comparará con los colores existentes de la base de datos. Regresar, es el botón que permite retornar al Menú principal de la aplicación, en tanto, el botón Salir cierra todas las ventanas abiertas del sistema y finaliza la ejecución de este.

Además de los anteriores, se puede observar dos botones más. El botón "Identificar_Comparación", que realiza la identificación de color mediante una comparación de pixel de la imagen ingresada con la base de datos que posee el sistema, para permitir que el sistema pueda identificar diversos colores que pueden modificar su característica propia debido a factores de iluminación, resolución del elemento de captura, entre otras. Se crea un determinado rango de tolerancia para cada color perteneciente a los colores considerados; el valor establecido del rango se lo encontró mediante prueba y error hasta evitar que exista solapamiento en los colores parecidos.

El botón "Identificar_clasificación", que realiza una identificación basado en un clasificador el mismo que fue entrenado empleando un clasificador por aprendizaje o entrenamiento, se utilizó una base de datos que cuenta con 5052 combinaciones que corresponden a 11 colores en los cuales se pueden identificar Negro, Azul, Café, Verde, Gris, Naranja, Rosa, Morado, Rojo, Blanco y Amarrillo. Estos colores son clasificados considerando una Máquina de Vector de soporte en una condición Cubic SVM. Se debe destacar que cada color posee diferentes valores en sus componentes RGB y esto se debe a las diversas tonalidades que puede presentar un color debido a diversos factores externos en los que se realiza la captura de la imagen.

En la Figura 21 se muestra el GUIDE desarrollado, en donde se evidencia lo anteriormente descrito.

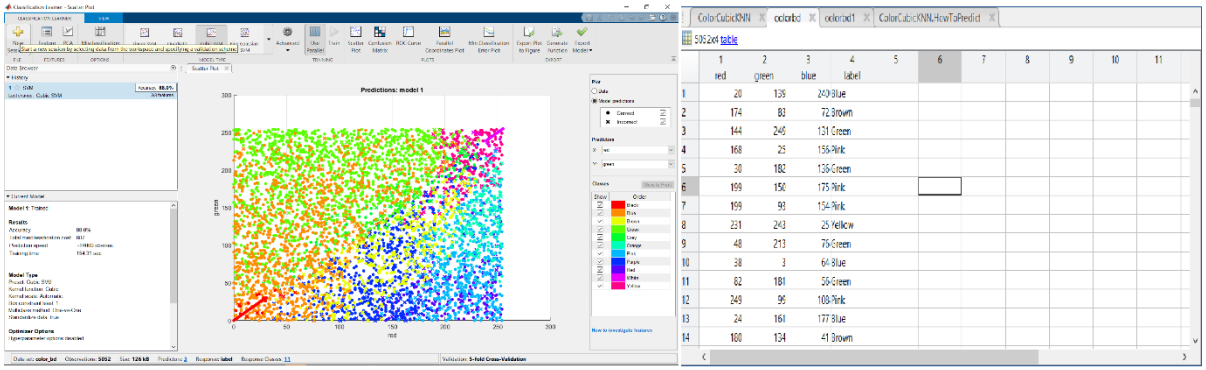
Figura 21. *GUIDE Desarrollado.*



Fuente. Elaboración Propia.

En la figura 22 se muestra la etapa del clasificador realizado, el cual posee un 88.0% de precisión para la deteción, además se muestra una pequeña porción de la base de datos empleada para el entrenamiento.

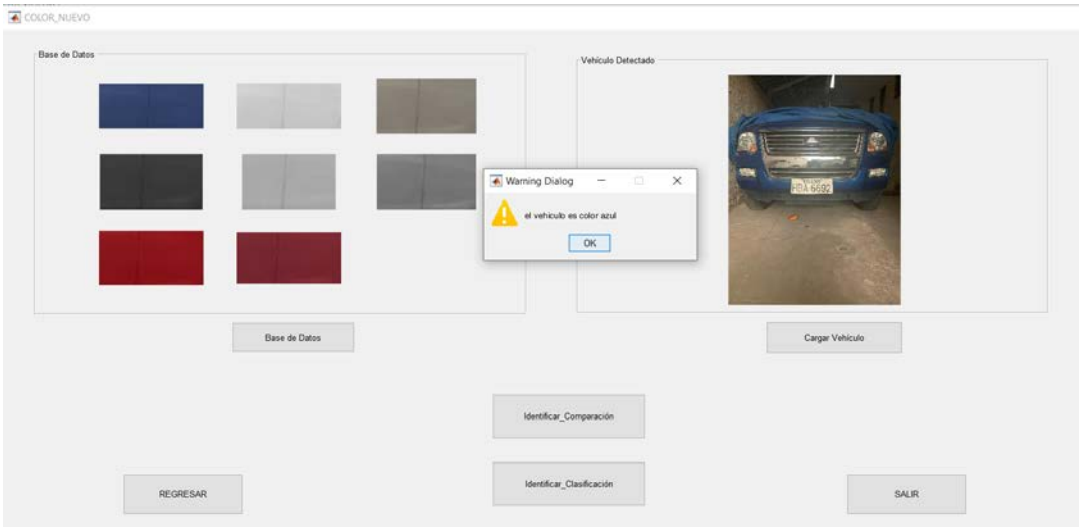
Figura 22. *Etapa del Clasificador y Base de Datos.*



Fuente. Elaboración Propia.

Los resultados obtenidos se despliegan mediante cuadros de dialogo que indican el resultado de la aplicación de los colores, el funcionamiento del GUIDE se lo evidencia en la Figura 23.

Figura 23. *Resultado de la Aplicación de Colores*.



Fuente. Elaboración Propia.

La identificación del color del vehículo mediante el clasificador permite al operador del sistema seleccionar el píxel del que desea obtener el color, es decir que le da la posibilidad de seleccionar el mejor píxel que considere para la identificación del color; el resultado de la identificación mediante el Clasificador se lo muestra en la figura 24.

Figura 24. *Resultado de la Identificación Mediante el Clasificador.*

Editor - COLOR_NUEVO.m | Variables - colorbd

ColorCubicKNN | colorbd | colorbd1 | ColorCubicKNN.HowToPredict

5052x4 table

	1 red	2 green	3 blue	4 label	5	6	7	8	9	10	11
1	20	139	240	Blue							
2	174	83	72	Brown							
3	144	249	131	Green							
4	168	25	156	Pink							
5	30	182	136	Green							
6	199	150	175	Pink							
7	199	93	154	Pink							
8	231	243	25	Yellow							
9	48	213	76	Green							
10	38	3	64	Blue							
11	82	181	56	Green							
12	249	99	108	Pink							
13	24	161	177	Blue							
14	180	134	41	Brown							

Command Window

```
1×4 table

  red    green    blue       label
  ___    _____    ____    ___________

  58      78      103     <undefined>

   Blue
```

Fuente. Elaboración Propia.

Para la identificación del modelo del vehículo se ha considerado al igual que los casos anteriores el desarrollo de un GUIDE, lo suficientemente interactivo de manera que le permite al operador o usuario del sistema realizar una identificación del modelo del vehículo de la marca Chevrolet de una forma sencilla e intuitiva.

En el archivo Modelo.fig se dispone de 3 axis que permiten la ubicación de las imágenes presentes en el procesamiento digital de las imágenes, el axis ubicado en la parte superior izquierda del GUIDE se lo emplea para visualizar la base de datos que esta conformada por 13 modelos de vehículos los mismos que pertenecen a una misma casa comercial. Los modelos que se considerarón para el desarrollo de la aplicación son: Aveo, Beat, Cavalier, Cruze, D-Max, Equinox, Sail, Scross, Spark, Tahoe, Tracker, Vans y el VitaraSZ. Todos los modelos enunciados se los puede visualizar en la figura 25.

Figura 25. *Modelos de Vehículos Utilizados en la Base de Datos.*

Fuente. Elaboración Propia.

En la parte derecha superior de la aplicación se encuentra localizado el segundo axis, este permite visualizar el vehículo que se desea comparar con los modelos establecidos en la base

de datos, se debe destacar que en este entorno el vehículo que se despliega en este etapa corresponde al móvil ingresado en el Menú de la aplicación.

En la parte inferior de este desarrollo se colocó un axis en el cual se cargan los resultados de la identificación del modelo, para llevar a cabo la identificación se emplea una detección y extracción de puntos característicos de la imagen perteneciente a la base de datos y también a la imagen a ser analizada; considerando la herramiento detectSURFFeatures del software empleado, la cual genera una burbuja sobre los puntos mas relevantes de las imágenes. Se considerarón obtener 100 puntos característicos de cada una de las fotos que ingresan al sistema, este valor se considera suficiente para una comparación efectiva de los puntos más representativos y característicos de las imágenes, además de reducir el costo computacional.

De forma adicional a esta etapa de extracción y análisis de puntos característicos se empleó un estimador geométrico, que permite posicionar de forma adecuada los puntos más relevantes de la imagen, es decir, se logra relacionar los puntos similares de acuerdo a la posición en la que estos se localizan.

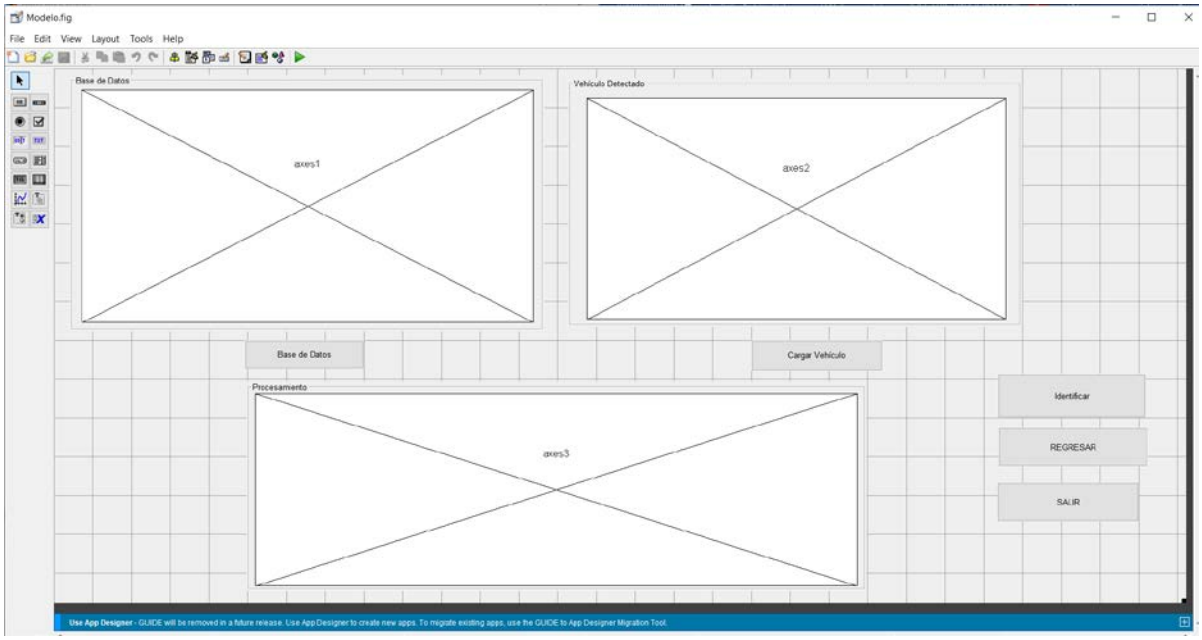
Para reducir el procesamiento en esta etapa se considera una correlación de imágenes, es así que se busca el mayor valor de parecido del modelo ingresado con los existentes en la base de datos. Para que la correlación se realice de forma exitosa se consideraron las etapas de redimensionamiento de las imágenes tanto de la base de datos como de la imagen a ser analizada, mediante el operador “bicubic”que considera una vecindad de 8x8 pixel alrededor del pixel analizado logrando preservar caracteríticas propias de los modelos inalterables a pesar de reducir el tamaño de la imagen. Después de esta operación se cambio de un formato RGB a uno en escala de grises, este se lo realizó en base a reducir el tiempo de espera o costo computacional que conlleva trabajar con imágenes. Al momento del cambio de formato de la imagen se logra reducir de la imagen RGB formada por 3 matrices a una sola matriz que posee valores en escala de grises que van desde 0 a 255.

En esta etapa se realiza una comparación de cada una de las imágenes de la base de datos con respecto de la imagen ingresada para el análisis buscando el mayor valor de la correlación, recordando que los resultados que se pueden encontrar son entre -1 y 1; si los valores encontrados se aproximan a 1 se dice que las imágenes son muy parecidas o iguales, si el resultado es cercano a -1 nos indica que las imágenes son parecidas pero opuestas, es decir,

que sus colores son opuestos en la misma posición, y finalmente si el valor de la correlación es igual o cercano a 0 nos indica que las imágenes comparadas son completamente diferentes.

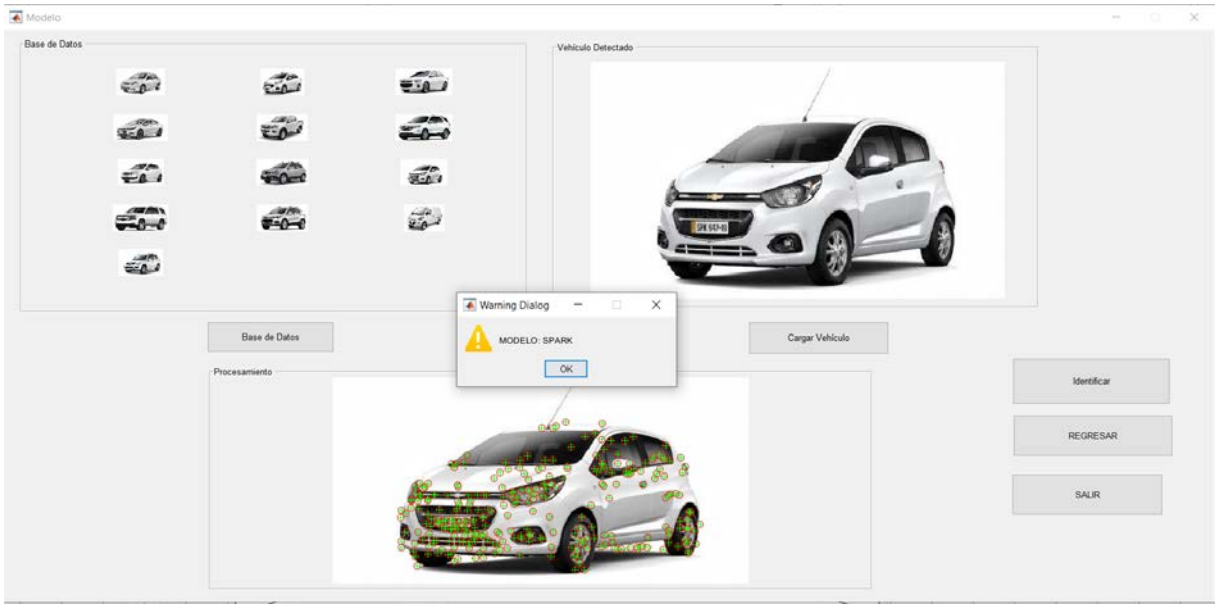
En la figura 26 se expone la estructura del GUIDE correspondiente al Modelo.

Figura 26. *GUIDE Correspondiente al Modelo de Vehículo.*



Fuente. Elaboración Propia.

Figura 27. *Reconocimiento del Modelo de Vehículo*



Fuente. Elaboración Propia.

El modelo del vehículo analizado se lo despliega mediante un cuadro de dialogo que se muestra en la figura 27. De forma oculta se realiza todo el procesamiento descrito en el cual se puede destacar los siguientes resultados alcanzados en el desarrollo de la aplicación.

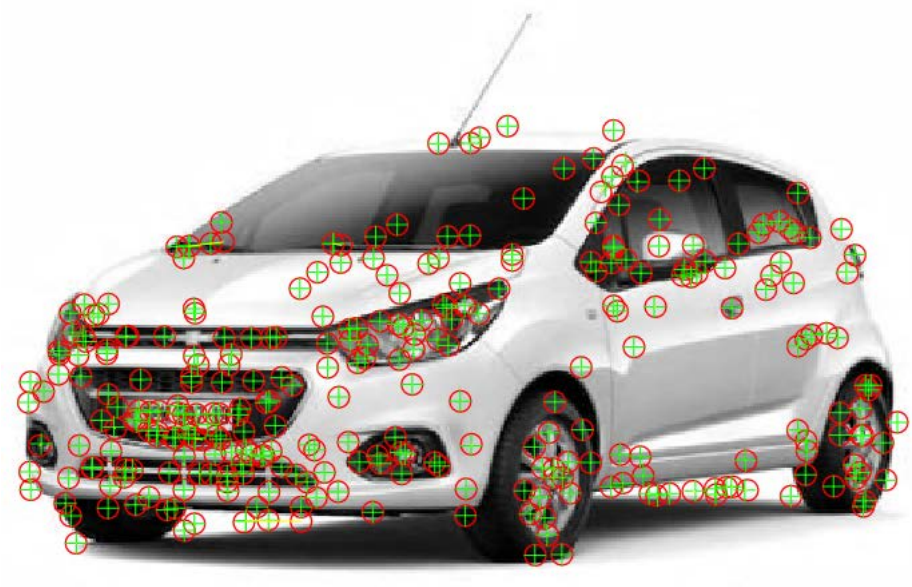
La aplicación de detectSURFFeatures genera imágenes como se muestra en la figura 28.

Figura 28. *Aplicación de DetectSURFFeatures*

Fuente. Elaboración Propia.

La extracción y emparejamiento de las figuras se muestra en la figura 29, en esta se debe destacar que se empleó la estimación geométrica que permite analizar los puntos que se encuentran ubicados en la misma posición.

Figura 29. *Extracción y Emparejamiento de las Figuras.*



Fuente. Elaboración Propia.

6. Metodología del trabajo

La temática propuesta en el presente trabajo desarrolla un algoritmo, que genera tres etapas de identificación y reconocimiento: placa de matrícula, color y modelo de un vehículo. Cuyo entorno visual es implementado a través de un GUIDE con la finalidad de mostrar datos e información al usuario, de acuerdo con su requerimiento.

Para cumplir con lo mencionado, la metodología de trabajo aplicada depende del requerimiento de cada una de las etapas de identificación y reconocimiento. Siendo comunes para todas las etapas, las metodologías derivadas en las siguientes fases:

1. Verificación del estado del arte
2. Recopilación y selección de imágenes, tanto para la base de datos como para evaluar el algoritmo.
3. Implementación del algoritmo en MATLAB.
4. Evaluación y análisis de resultados.

La implementación del algoritmo en MATLAB es la fase que hasta cierto punto difiere en cada una de las tres etapas de identificación y reconocimiento, debido a la utilización de funciones y comandos que son aplicados de cuerdo a la necesidad y al fundamento matemático que ofrecen. Estas funciones y comandos principales se describirán de acuerdo a cómo fueron aplicadas en cada una de las etapas.

Se adjunta en el Anexo A. Código Matlab, el cogido de los tres módulos.

6.1. Algoritmo de reconocimiento de la matrícula de un vehículo

El reconocimiento de la matrícula se realizó a través de la lectura de una imagen, la misma que debe reposar en el actual directorio de trabajo o estar incluido en una ruta relativa a un archivo. A esta imagen se le han aplicado distintos procesos de mejoramiento con la finalidad de obtener caracteres lo más claros y precisos posibles. Como primera acción para lograr el cometido, es leer la imagen, para esto se ha utilizado la sentencia, k= imread ('movil1.jpg'), donde “imread”, lee la imagen que previamente ha sido definido con el parámetro “movil1.jpg”, cuyo formato de imagen JPEG ha sido ubicado como predeterminado, existiendo también la opción que la aplicación permita reconocer distintos formatos de imagen. La imagen leída es devuelta a través de la variable “k”, para que luego con el comando “imshow”

se pueda mostrar la imagen. La sentencia utilizada para mostrar la imagen es, imshow (k). El comando imshow llama a imread para leer la imagen del archivo, pero no almacena los datos de la imagen en el espacio de trabajo de MATLAB.

Al mostrar la imagen "k", con la sentencia I = imcrop(k), se da la opción al usuario que recorte la imagen según la necesidad que crea conveniente. El comando "imcrop" permite crear una herramienta de recorte de imágenes interactiva. La herramienta es un rectángulo móvil y redimensionable que permite al usuario manipular directamente con el ratón del ordenador. Una vez que el usuario activa esta opción, recorta la imagen de destino haciendo doble clic en la herramienta. En este algoritmo se especificó el tipo de imagen como argumento de entrada, por lo tanto, la imagen de salida tiene la misma clase que la imagen de entrada, en este caso es la variable "I".

Una vez recortada la imagen, para evitar el coste computacional y a través de la caja de herramientas, Image Processing Toolbox de MATLAB, se utilizó la línea de código I = rgb2gray (I), cuya función llamada "rgb2gray" permitió convertir la imagen "I", que previamente ha sido recortada por el usuario (parámetro de entrada de esta función) a escala de grises, eliminando la información de tono y saturación, pero conservando la luminancia. Esta función (rgb2gray), matemáticamente se fundamenta en la conversión de valores RGB en valores de escala de grises mediante un mezclado de los tres canales RGB, para obtener uno solo en gris tomando porcentajes: Rojo 30%, Verde 59%, Azul 11%, mediante la siguiente formula:

$$\mathbf{0.2989} * \boldsymbol{R} + \mathbf{0.5870} * \boldsymbol{G} + \mathbf{0.1140} * \boldsymbol{B}$$ **, Ecuación 4.** Conversión de RGB a escala de grises

La ecuación 4, es aplicado a cada píxel de la imagen RGB, para realizar la conversión y reducir la información con la que de ahora en adelante se va a trabajar, obteniendo de esta forma una nueva matriz de un byte por píxel que sería la información de luminancia.

Para mejorar la calidad de imagen que es devuelta de la conversión a escala de grises, se aplica la sentencia J=imadjust(I), donde "imadjust" toma como parámetro de entrada la imagen "I", previamente convertida a escala de grises, con la finalidad de asignar valores de intensidad de la imagen convertida en escala de grises a nuevos valores para la salida en "J", de forma predeterminada, satura el 1% inferior y el 1% superior de todos los valores de píxel. Esta operación permitió aumentar el contraste de la imagen de salida "J".

Una vez que la imagen en escala de grises ha sido ajustada, se estableció el umbral a través de la función "graythresh(I)", que toma como parámetro de entrada la última imagen ajustada

"I", para calcular su histograma y luego encontrar el valor umbral. Graythresh, se fundamenta en el método Otsu, el cual elige un umbral que minimiza la varianza intraclase de los píxeles en blanco y negro. Este método Otsu según Huamaní Navarrete,(2016) matemáticamente se respalda a partir de un histograma, donde es necesario normalizar dicho histograma como una función de probabilidad del tipo discreta P_r. Donde N es el número total de pixeles en la imagen, n_q es el numero de pixeles que tiene un nivel de intensidad r_q y L los posibles niveles de intensidad en la imagen. Lo mencionado se detalla en la ecuación 5.

$$\boldsymbol{P_r(r_q) = \frac{n_q}{N},\ q = 0, 1, 2, \ldots, L-1}$$ **Ecuación 5.** Función de probabilidad

Una vez normalizado el histograma, se elige un umbral k en donde, C_0 represente a un conjunto de pixeles con niveles $0,1,2, \ldots, k-1$ y C_1 otro grupo de pixeles con niveles $k, k+1, k+2, \ldots, L-1$. El método de Otsu selecciona el valor del umbral maximizando la varianza σ_B^2 , la cual queda definida por la ecuación 6:

$$\boldsymbol{\sigma_B^2 = w_0 * (u_0 - u_T)^2 + w_1 * (u_1 - u_T)^2}$$ **Ecuación 6.** Cálculo de la Varianza

Donde:

$$\boldsymbol{w_0 = \sum_{q=0}^{k-1} P_q(r_q)}$$ **Ecuación 7.** Cálculo del peso Wo

$$\boldsymbol{w_1 = \sum_{q=k}^{L-1} P_q(r_q)}$$ **Ecuación 8.** Cálculo del peso W1

$$\boldsymbol{u_0 = \sum_{q=0}^{k-1} q * \frac{P_q(r_q)}{w_0}}$$ **Ecuación 9.**Calculo del parámetro $\boldsymbol{U_0}$

$$\boldsymbol{u_1 = \sum_{q=k}^{L-1} q * \frac{P_q(r_q)}{w_1}}$$ **Ecuación 10.** Cálculo del parámetro $\boldsymbol{U_1}$

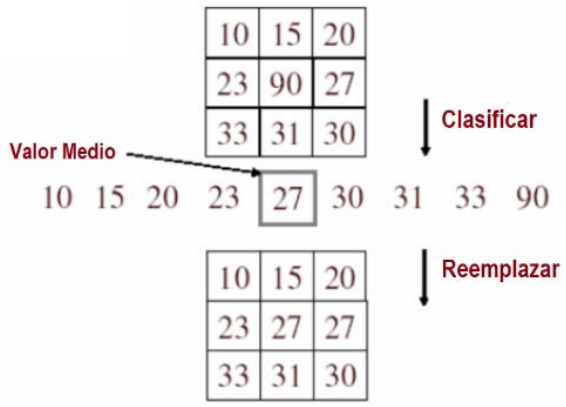
$$\boldsymbol{u_T = \sum_{q=0}^{L-1} q * P_q(r_q)}$$ **Ecuación 11.** Calculo Umbral total

El umbral devuelto es expresado a través de la variable "umbral", y se encuentra en el rango [0, 1] como valor normalizado. Esto determinó, la cantidad de píxel transformados a blanco y la cantidad de píxel a negro. Todo lo mencionado, se ejecuta a través de la línea de código umbral = graythresh(I).

El paso siguiente a la umbralización es la binarización de la imagen, en este algoritmo la sentencia utilizada para este fin es, I1 = im2bw (I, umbral), donde "im2bw", utiliza dos parámetros el "**I",** que viene siendo la imagen en escala de grises y el "umbral" que se calculó a partir de la función anterior. Esta función produce imágenes binarias a partir de imágenes indexadas, de intensidad o RGB, es decir convierte la imagen en escala de grises a blanco o negro de acuerdo al cálculo del umbral. La imagen binaria de salida "I1" tiene valores de 1 (blanco) para todos píxeles en la imagen de entrada con luminancia superior al nivel y 0 (negro) para todos los demás píxeles.

Aplicar la binarización a una imagen no es suficiente para obtener un buen detalle de los caracteres que posee la imagen. Por lo que se ha aplicado un filtro de mediana, mediante la siguiente sintaxis, i3=medfilt2(i2), donde "medfilt2(i2)" elimina los pixeles aislados. Para esto, realiza el filtrado medio de la imagen (i2) en dos dimensiones. Cada píxel de salida contiene el valor mediano en una vecindad de 3 por 3 alrededor del píxel correspondiente en la imagen de entrada. Si la imagen de entrada es de clase entera, toda la salida se devuelven valores como números enteros. Si el número de píxeles en la vecindad (es decir, M * N) es par, algunos de los valores medianos pueden no ser números enteros. En estos casos, las partes fraccionarias se descartan. Una forma de comprender este filtrado es a partir de su base matemática, para lo cual en la Figura 30 se puede observar que el filtrado consiste en reemplazar el valor central $u = I(x, y)$ por el valor de la mediana u_{med} previamente calculado. Este proceso se repite para cada píxel de la imagen, siendo escrito el resultado de la mediana en la imagen resultado "i3".

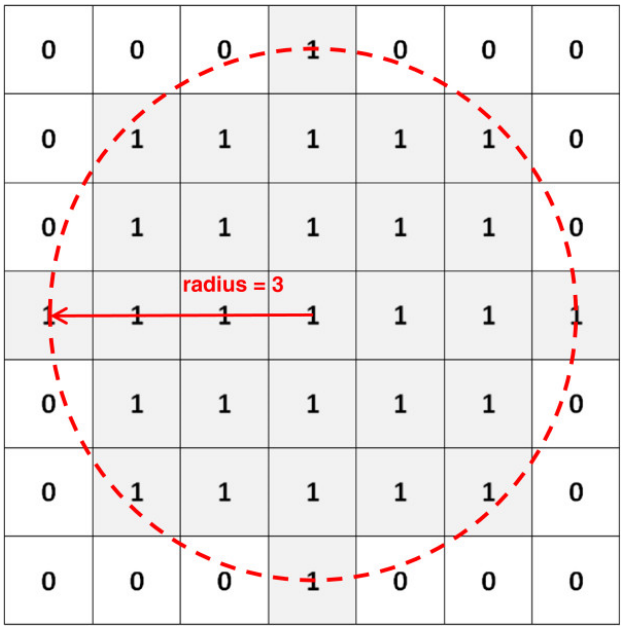
Figura 30. *Aplicación del filtro de mediana*



Fuente. Tomado de http://www.dccia.ua.es/dccia/inf/asignaturas/Vision/vision-tema2.pdf

A partir de un elemento estructurante aplicado en la sentencia se=strel('disk',7), se logró mejorar aún más el detalle de los caracteres de la imagen, mediante un barrido de pixeles aislados que todavía no se han eliminado, este proceso se logra a través de operaciones morfológicas mediante el elemento estructurante “strel('disk',7)”, que como referencia toma el parámetro “disk”, el cual crea un elemento de estructuración en forma de disco, donde el valor del radio debe ser un número entero no negativo, para este caso se especifica el valor del radio igual a 7. Las operaciones morfológicas que utilizan aproximaciones de disco tienen la ventaja de ejecutarse mucho más rápido cuando el elemento de estructuración utiliza aproximaciones. La imagen de salida obtenida se evidencia en la variable “se”. En la figura 31 se muestra un ejemplo de un elemento estructurante en forma de disco con un radio de 3. La ubicación del 1 en la matriz define la vecindad para la operación morfológica.

Figura 31. *Elemento estructurante en forma de disco*



Fuente. Tomado de (Haggerty et al., 2014).

Establecido el elemento estructurante, se realiza el cierre morfológico a través de la sentencia i4=imclose(i3, se), sobre la imagen que fue aplicada el filtro de mediana “i3” y el elemento estructurante “se”. Esta operación de cierre dilata una imagen y, a continuación, erosiona la imagen dilatada, utilizando el mismo elemento de estructuración para ambas operaciones.

Una vez realizo el cierre morfológico, la imagen esta lista para el etiquetado de los elementos. La sentencia que permite este proceso es i5 = bwlabel(i4), donde “bwlabel(i4)”, etiqueta los

elementos detectados en la última imagen que fue aplicada el cierre morfológico “i4”. Este proceso devuelve una matriz “i5”, que contiene etiquetas para los 8 objetos conectados. Para una mejor comprensión la Figura 32 indica un ejemplo del etiquetado de componentes conectados, cuyo proceso es identificar los componentes conectados en una imagen y asignar a cada uno una etiqueta única

Figura 32. *Etiquetado de componentes conectados*

Fuente. Adaptada de MathWorks (2021a).

Los píxeles etiquetados como 0 son el fondo, los píxeles etiquetados 1 forman un objeto, los píxeles etiquetados 2 forman un segundo objeto, y así sucesivamente.

Por último, a través de la línea de código stats = regionprops(i5, 'all'), se pretende medir un conjunto de propiedades para cada región etiquetada, en la imagen “i5”, el parámetro “all” es el que permite el acceso a estas propiedades, esto de acuerdo a cada componente conectado que fue etiquetado anteriormente. Por ejemplo, los elementos igual al número entero 1 corresponden a la región 1; los elementos igual al número entero 2 corresponden a la región 2, y así sucesivamente.

Complementando a esta función se hace uso de la sentencia E1=stats(1).BoundingBox, el cual permite generar el primer elemento detectado con sus dimensiones, para luego realizar un recorte con el valor o lugar en donde este fue localizado. Esta sentencia se utilizará para generar todos los elementos detectados y a partir de esto representar cada uno de los caracteres de la placa de matrícula de un vehículo.

6.2. Algoritmo de reconociendo del color de un vehículo

El reconocimiento del color del vehículo se realiza mediante dos técnicas,

6.2.1. **Identificar_Comparación**

Identifica el color del vehículo a través de la comparación de la imagen que ingresa el usuario con otra ingresada previamente en una base de datos. Para esto con la siguiente sentencia v=imread('Vino.jpg'), se lee la imagen ingresada en la base de datos, el comando que permite esto es, “imread('Vino.jpg')”, el archivo de nombre “vino.jpg” tiene el color vino que es leído por el comando y devuelto como salida a través de la variable “v”. En este proceso se ha considerado 8 colores en la base de datos, utilizando la misma sentencia para leer los demás colores.

Con la sintaxis, [c1,f1,v1] = impixel(v,20,36), y a partir de los parámetros ingresados, devuelve los valores de los píxeles especificados en la imagen “v”. Los valores 20 y 36 son vectores de igual longitud que especifican las coordenadas de los píxeles cuyos valores RGB se devuelven en una matriz “v1”.

Para leer la imagen que ingresa el usuario, se utiliza una sintaxis similar a la anterior, vim= imread('movil1.jpg'), en donde el comando “imread('movil1.jpg'), lee la imagen ingresada por el usuario con el parámetro “móvil1.jpg”, esta imagen luego de ser leída es devuelta a través de la variable “vim”. De la misma forma que la anterior, la sintaxis [c9,f9,v9]=impixel(vim,446,183), ahora devuelve los valores de los píxeles especificados en la imagen “vim”. Los valores 446 y 183 son vectores de igual longitud que especifican las coordenadas de los píxeles cuyos valores RGB será devueltas también en una matriz “v9”.

Por último, para identificar el color del vehículo ingresado se crea un determinado rango de tolerancia. Las siguientes sentencias se toman como ejemplos, vmax=v1+10; vmin=v1-10, “v1” es el vector anterior, establecido según la cantidad de pixeles a ser analizado. A este vector se le sumó un valor de 10 para valor máximo y se restó un valor de 10 para valor mínimo, estos valores de tolerancia generan un vector entre valores máximos y mínimos para determinar el color. Por ejemplo vino=[vmin, vmax]; la finalidad de este rango de tolerancia es evitar que exista solapamiento entre colores parecidos.

6.2.2. Identificar_ Clasificación

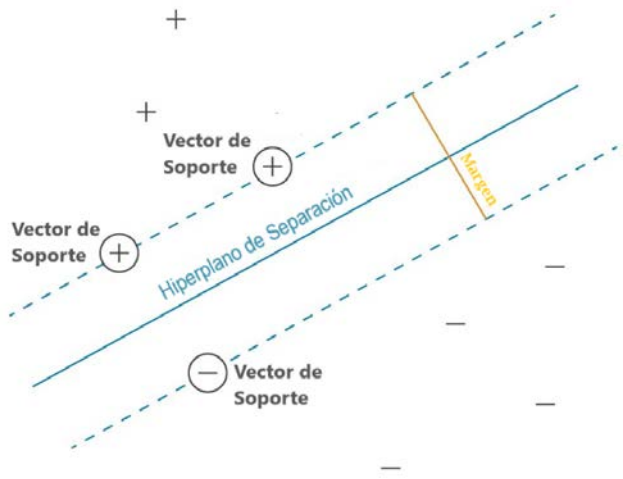
Identifica el color del vehículo a través de un clasificador previamente entrenado a través de una aplicación de Matlab (Classification Learner). Se describirá el clasificador debido a que es la parte central de la técnica.

6.2.2.1. Clasificador Cubico SVM

Este clasificador utiliza una SVM (Máquinas de Vectores de Soporte) que clasifica los datos al encontrar el mejor hiperplano, separando los puntos de datos de una clase respecto de la otra clase. El mejor hiperplano para una SVM significa el que tiene el mayor margen entre las dos clases. Es decir, el ancho máximo de la losa paralela al hiperplano que no tiene puntos de datos interiores (MathWorks, 2021b)

Los vectores de soporte son los puntos de datos más cercanos al hiperplano de separación, estos puntos están en el límite de la losa. En la Figura 33 se observa estas definiciones, con + indica puntos de datos de tipo 1 y - indica puntos de datos de tipo -1.

Figura 33. *Puntos de datos más cercanos al hiperplano de separación.*



Fuente. Adaptada de MathWorks, (2021c).

6.2.2.2. Modelo de clasificación de color RGB

Inicialmente se parte de una base de datos obtenida por Chavan (2020) donde se recopila más de 5,000 combinaciones de colores RGB etiquetados en base a los mencionados 11 colores principales, los 3 colores básicos rojo, verde y azul más amarillo, naranja, rosa, morado, marrón, gris, negro y blanco. (Griffin, 2005). La tabla 5 muestra esta base de datos.

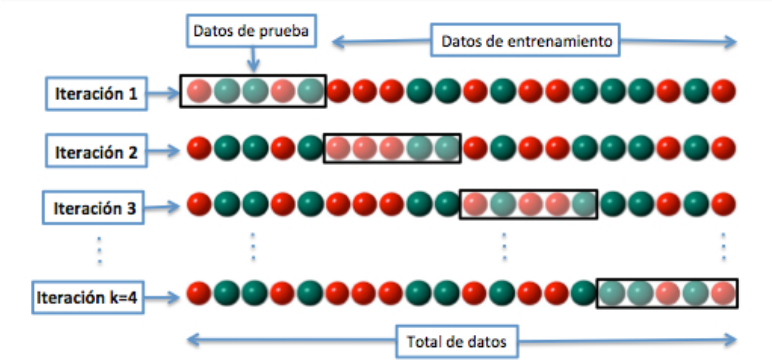
Tabla 5.*Base de datos, colores RGB etiquetados.*

ROJO	VERDE	AZUL	ETIQUETA
20	139	240	Azul
174	83	72	Marrón
144	249	131	Verde
168	25	156	Rosa
…	…	…	…

Fuente. Adaptada de (Griffin, 2005).

Para el entrenamiento del modelo de clasificación se usa la aplicación de MATLAB "Classification Learner App" (MATLAB, 2021). Se puede elegir entre los siguientes modelos: árboles de decisión, análisis discriminante, máquinas de vectores de soporte, regresión logística, vecinos más cercanos, Bayes ingenuos, conjuntos y redes neuronales. En la Figura 34 se puede apreciar, cómo se utiliza la validación cruzada con particiones de 5 elementos.

Figura 34. *Esquema k-f_cross validation, con k=4 y un solo clasificador.*



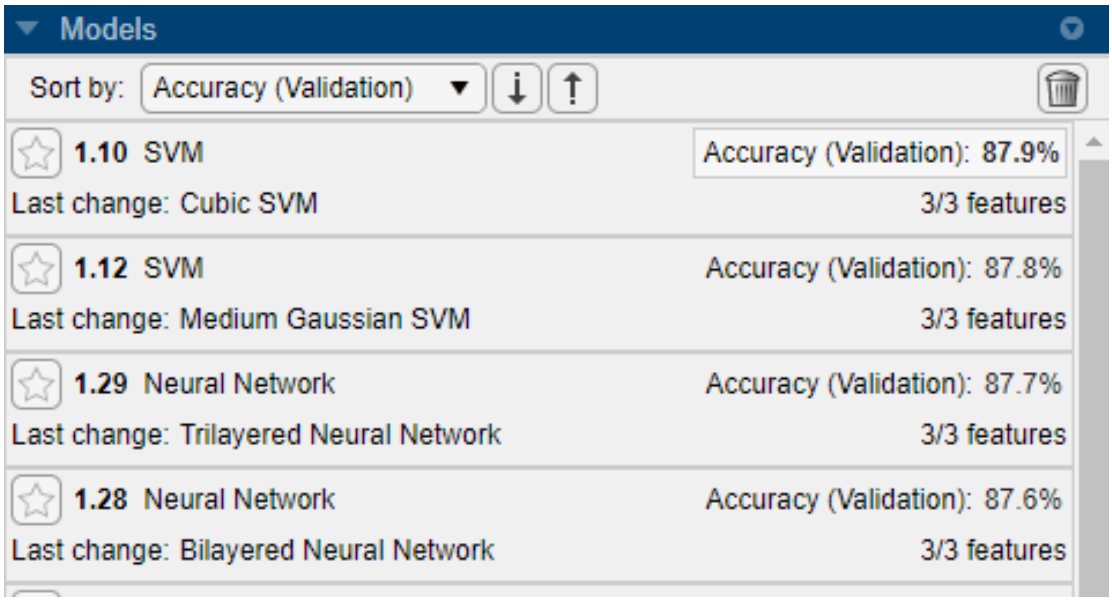
Fuente. Tomado de Wikipedia (2021).

Se ha elegido la opción de entrenamiento automatizado, donde MATLAB itera entre todos los modelos mencionados anteriormente disponibles. La aplicación muestra los resultados del modelo validado. Las medidas de diagnóstico, como la precisión del modelo, y los gráficos, como un gráfico de dispersión o el gráfico de matriz de confusión, reflejan los resultados del modelo validado. Se puede entrenar automáticamente una selección de todos los clasificadores, comparar los resultados de la validación y elegir el mejor modelo que funcione

para el problema de clasificación. Cuando se elige un modelo para exportar al espacio de trabajo, Classification Learner exporta el modelo completo.

Después de entrenar a los clasificadores en Classification Learner, puede comparar modelos basados en puntajes de precisión, visualizar resultados trazando predicciones de clase y verificar el rendimiento usando la matriz de confusión y la curva ROC. La figura 35 hace referencia a lo antes mencionado.

Figura 35. *Precisión de los modelos ordenados de mayor a menor.*



Fuente. Tomado de (MATLAB, 2021).

Después de entrenar un modelo en Classification Learner, se verifica el panel modelos, para ver qué modelo tiene la mejor precisión general en porcentaje. La tabla 5 indica para nuestro caso el Modelo VSM Cubico.

Tabla 6. *Resultados del Modelo VSM Cubico.*

RESUMEN DEL MODELO CENTRAL	
Modelo 1.10 entrenado	
Resultados de entrenamiento	
Precisión de resultados	87.9%
Costo total (validación)	613
Velocidad de Predicción	~ 18000 obs/sec
Tiempo de entrenamiento	200.4 sec
Tipo de modelo	
Preestablecido:	Cubic SVM
Función del Kernel:	Cubic
Escala Kernel:	Automático
Nivel de restricción de caja	1
Método multiclase	Uno a uno
Estandarizar datos	Verdadero
Opciones de optimizador	
Opciones de hiperparámetros deshabilitados	
Selección de características	
Todas las funciones utilizadas en el modelo, antes de PCA	
PCA	
PCA desactivado	
Costos de clasificación errónea	
Matriz de costos	Por defecto

Fuente. Elaboración Propia.

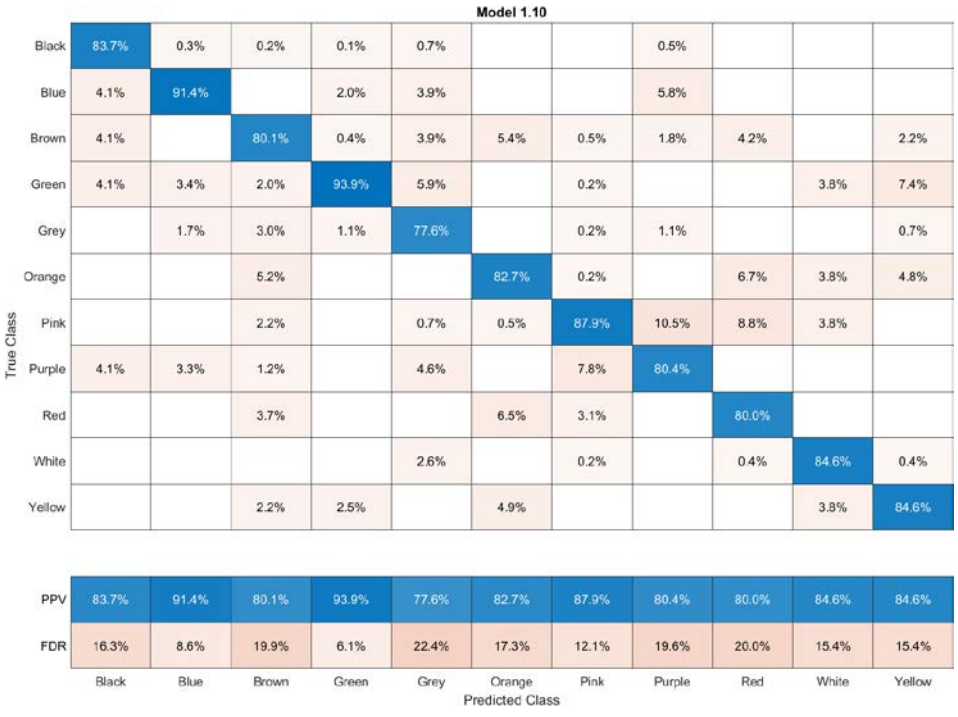
Se verifica el desempeño por clase, utilizando la matriz de confusión. En la Figuras 36 y 37 se puede observar la configuración de esta matriz en donde las filas muestran la clase verdadera y las columnas muestran la clase predicha. Las celdas diagonales muestran dónde coinciden la clase verdadera y la clase predicha. Si estas celdas diagonales son azules, el clasificador ha clasificado las observaciones de esta clase como correctas

Para ver el rendimiento del clasificador por clase, se selecciona la opción tasas positivas verdaderas (TPR), tasas negativas falsas (FNR). El TPR es la proporción de observaciones correctamente clasificadas por clase verdadera. El FNR es la proporción de observaciones clasificadas incorrectamente por clase verdadera. La figura 36 muestra resúmenes por clase real en las dos últimas columnas de la derecha.

Figura 36.. *Matriz de confusión (filas) - Modelo VSM Cubico.*

Fuente. Elaboración Propia.

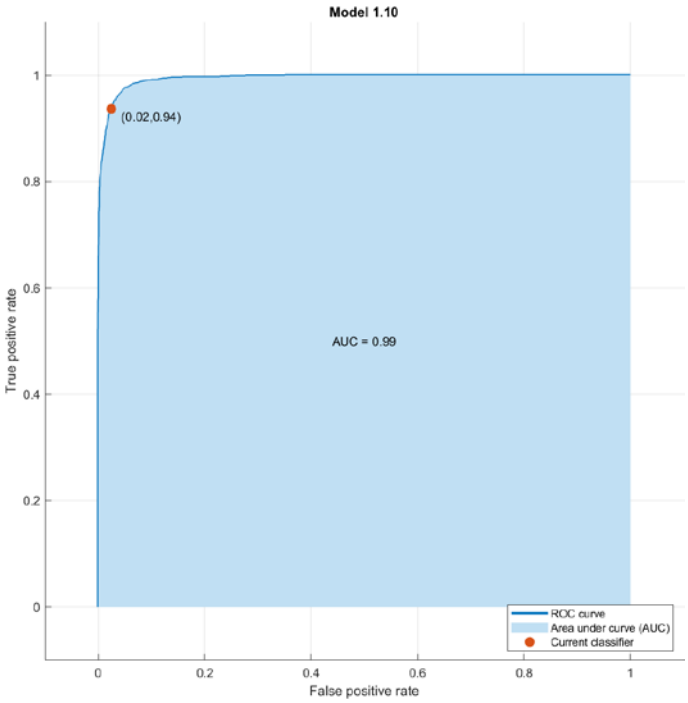
Figura 37.. *Matriz de confusión (columnas) - Modelo VSM Cubico.*



Fuente. Elaboración Propia.

Se verifique también la curva ROC. La curva ROC muestra la tasa de verdaderos positivos versus la tasa de falsos positivos para el clasificador entrenado actualmente seleccionado.

Figura 38. Curva ROC con el clasificador entrenado.



Fuente. Elaboración Propia.

El marcador en la Figura 38 muestra el desempeño del clasificador seleccionado actualmente. El marcador muestra los valores de la tasa de falsos positivos (FPR) y la tasa de verdaderos positivos (TPR) para el clasificador seleccionado actualmente. En nuestro caso, una tasa de falsos positivos (FPR) de 0,02 indica que el clasificador actual asigna el 2% de las observaciones incorrectamente a la clase positiva. Una tasa de verdaderos positivos de 0,94 indica que el clasificador actual asigna correctamente el 94% de las observaciones a la clase positiva.

6.3. Reconocimiento modelo vehículo

6.3.1. Imresize

En nuestro caso se usa la sentencia J = imresize (I, scale, method). (MATLAB, 2021), para escalar todas las imágenes a la misma escala de referencia elegida 280x566x3. Siendo:

- J, la imagen resultado del uso del comando imresize.
- I, la imagen a escalar. La imagen de entrada puede ser una imagen en escala de grises, RGB, binaria o categórica.
- Scale, factor de cambio de tamaño, especificado como un numero positivo.

- Si la escala es menor que 1, entonces la imagen de salida es más pequeña que la imagen de entrada.
- Si la escala es mayor que 1, entonces la imagen de salida es más grande que la imagen de entrada.

Imresize aplica el factor de escala a cada dimensión de la imagen.

- Method, es el método de interpolación elegido, en nuestro caso se ha elegido 'bicubic'. Donde se realiza una interpolación bicubica; el valor del píxel de salida es un promedio ponderado de los pixeles en la vecindad 4 x 4 más cercana. LA interpolación bicubica es el método predeterminado para las imágenes numéricas y lógicas, donde la ecuación fundamental de la convolución cubica es (Keys, 1981):

$$u(s) = \begin{cases} \frac{3}{2}|s|^3 - \frac{5}{2}|s|^2 + 1, para\ 0 < |s| < 1 \\ -\frac{1}{2}|s|^3 + \frac{5}{2}|s|^2 - 4|s| + 2, para\ 1 < |s| < 2 \\ 0\ para\ 2 < |s| \end{cases}$$ **Ecuación 12.** Convolución cubica.

En la figura 39 se muestra un ejemplo de imresize desde la imagen original con 315x528x3 pixeles a otra de 280x566x3

Figura 39. *Ejemplo de comando 'Imresize'*

Fuente. Elaboración Propia.

6.3.2. Corr2

La correlación de dos imágenes nos indica cuánto de parecidas son dichas imágenes, cuyo resultado tiende a 1 cuanto más se parezcan.

Se usa la sentencia R = corr2(A,B). (MATLAB, 2021) para hacer la correlación de dos imágenes y obtener la correlación. Cada imagen será una matriz de dimensiones X e Y, donde X es el número de píxeles horizontales e Y el número de píxeles verticales. X será el coche a catalogar e Y la imagen plantilla del modelo tipo a evaluar. El proceso será iterativo barriendo todas las imágenes plantilla disponibles.

El algoritmo utilizado es corr2 (MATLAB, 2021):

$$r = \frac{\sum\sum(A_{mn}-\bar{A})(B_{mn}-\bar{B})}{\sqrt{(\sum\sum(A_{mn}-\bar{A})^2)(\sum\sum(B_{mn}-\bar{B}))}}$$, **Ecuación 13**. Corr2.

Donde $\bar{A}$ es mean2(A), y $\bar{B}$ es mean2(B). Siendo mean2 la media de todos los valores de la matriz analizada.

Figura 40. *Ejemplo de comando 'Corr2'.*

Fuente. Elaboración Propia.

6.3.3. Modulo 'Computer Vision Toolbox'

Con objeto de detectar y extraer características o patrones para poder realizar la predicción del modelo de vehículo se usa la aplicación/paquete de MATLAB Computer VIsion Toolbox, donde de las categorías disponibles, se usa el paquete de comandos 'Feature Detection and Extraction".

A continuación, expondremos el funcionamiento de los comandos utilizados en nuestra aplicación: Detect Features, Extract Features y Match Features.

El comando points = detectSURFFeatures (I) (MATLAB, 2021) devuelve un objeto SURFPoints mediante un conjunto de puntos que contienen información sobre las características de SURF detectadas en la imagen 2D I de entrada en escala de grises.

La función detectSURFFeatures implementa el algoritmo de Speeded Up Robust Feature (SURF).

Figura 41. *Ejemplo de comando 'detectSURFFeatures'.*

Fuente. Tomada del módulo de ayuda de MATLAB®.

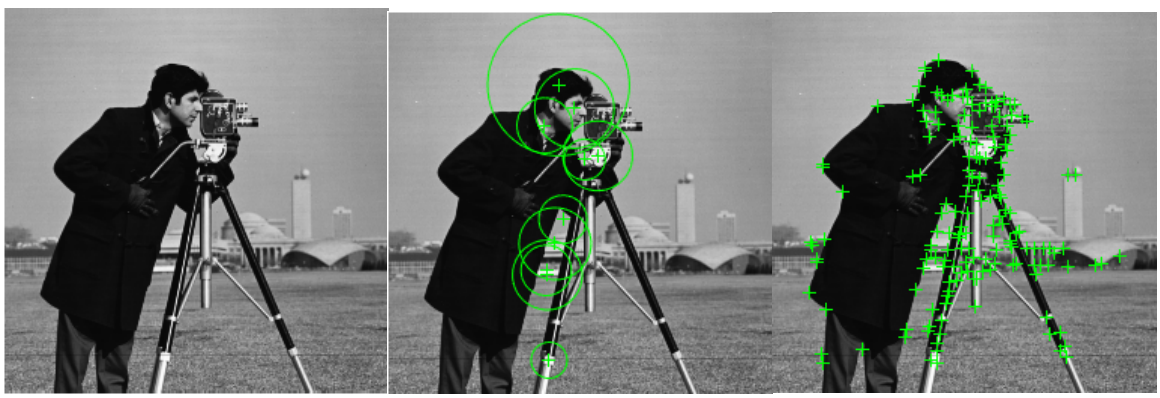
Un objerto SURFPoints proporciona la capacidad de pasar datos entre las funciones detectSURFFeatures y extractFeatures.

6.3.3.1. ExtractFeatures

Con este comando se extrae los descriptores y sus ubicaciones de los puntos de interés. La sentencia utilizada del comando es [features,validPoints] = extractFeatures(I,points) (MATLAB, 2021). Donde I es la imagen anterior desde la cual se han detectado los puntos de interés y points son los SURFPoints detectados anteriormente.

El comando deriva los descriptores de los píxeles que rodean un punto de interés. Los píxeles representan y coinciden con las características especificadas por una ubicación de un solo punto. Cada punto único especifica la ubicación central de un vecindario. El método que utiliza para la extracción de descriptores depende de la clase de los puntos de entrada.

Figura 42. *Ejemplo del proceso del comando 'extractSURFFeatures'.*



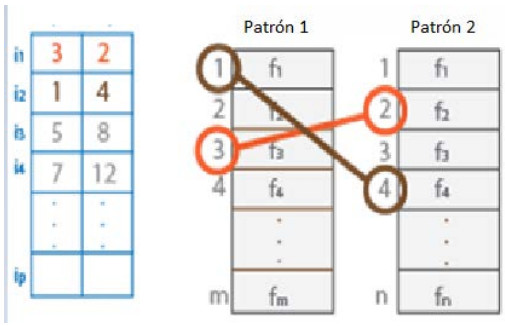
Fuente. Adaptada del módulo de ayuda de MATLAB®.

6.3.3.2. MatchFeatures

El comando matchFeatures encuentra características – patrones comunes entre dos imágenes. El comando utilizado es indexPairs = matchFeatures(features1, features2) (MATLAB, 2021), donde features1 y features2 son objetos binaryFeatures o una matriz M1 por N. La matriz contiene M1 características-patrones (features) y N corresponde a la longitud de cada vector de features. Podemos obtener el objeto binaryFeatures utilizando la función extractFeatures.

indexPairs corresponde a los índices comunes de los dos conjuntos de características de entrada, devueltos como una matriz P por2 de P índices. Cada par de índices corresponde a una característica coincidente entre las entradas features1 y features2. El primer elemento indexa la característica en features1. El segundo elemento indexa la característica coincidente en features2.

Figura 43. *Ejemplo del proceso del comando 'matchSURFFeatures'.*



Fuente. Adaptada del módulo de ayuda de MATLAB®.

Dando como resultado las features comunes.

Figura 44. *Resuldato del proceso del comando 'matchSURFFeatures'.*

Fuente. Tomada del módulo de ayuda de MATLAB®.

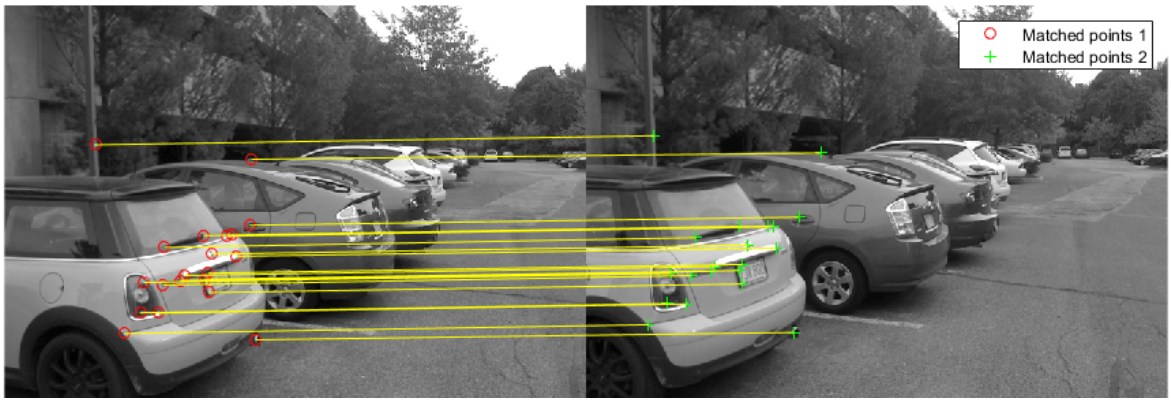
6.3.3.3. ShowMatchedFeatures

Con este comando lo que hacemos en visualizar los correspondientes puntos característicos.

El comando utilizado es showMatchedFeatures(I1, I2 ,matchedPoints1, matchedPoints2, method) (MATLAB, 2021).

I1 y I2 son las dos imágenes a comparar, matchedPoints1 y matchedPoints2 contienen las coordenadas de los puntos comunes (objetos SURFPoints, de matrizes Mx2) y method especifica el método utilizado, en nuestro caso se ha utilizado 'montaje', donde se representa una imagen junto a la otra.

Figura 45. *Ejemplo del proceso del comando 'showMatchedFeatures'.*



Fuente. Tomada del módulo de ayuda de MATLAB®.

7. Evaluación

Para el módulo de identificación de los caracteres de la placa se realizan varias pruebas que permiten identificar en qué características y entornos el prototipo funciona de forma adecuada, los datos alcanzados en cada desarrollo se evidencian en la Tabla 7. Se debe indicar que para mostrar los resultados obtenidos en las diversas pruebas se optó por la presentación en valores de 0 y 1; considerando que 1 indica que se realizó una identificación correcta según la programación desarrollada, y 0 que no se logró la identificación.

Para las pruebas de detección de los caracteres de la placa se realizaron varias capturas considerando una distancia menor o igual que 30 cm, de forma similar se realizaron varias capturas en un rango de 30 cm a 50 cm que se ingresaron al sistema para ver la eficiencia del módulo, se consideraron capturas del identificar vehicular considerando una distancia de 50 cm a 80 cm desde la placa a la cámara, y finalmente se realiza un set de capturas para el análisis considerando una distancia de 80 cm a 100 cm, la idea de aumentar la distancia de las capturas es validar el sistema funcionamiento del sistema.

Tabla 7. *Resultados identificación de placa.*

PLACA #	FRONTAL			
	HASTA 30CM	DE 30CM A 50CM	DE 50CM A 80CM	DE 80CM A 100CM
1	1	0	0	0
2	1	0	0	0
3	1	0	0	0
4	1	0	0	0
5	1	0	0	0
6	1	0	0	0
7	1	0	0	0
8	1	0	0	0
9	1	0	0	0
10	1	0	0	0

Fuente. Elaboración Propia.

La tabla de porcentajes hallada de los aciertos de cada columna de las diversas pruebas aplicadas se detalla en la Tabla 8.

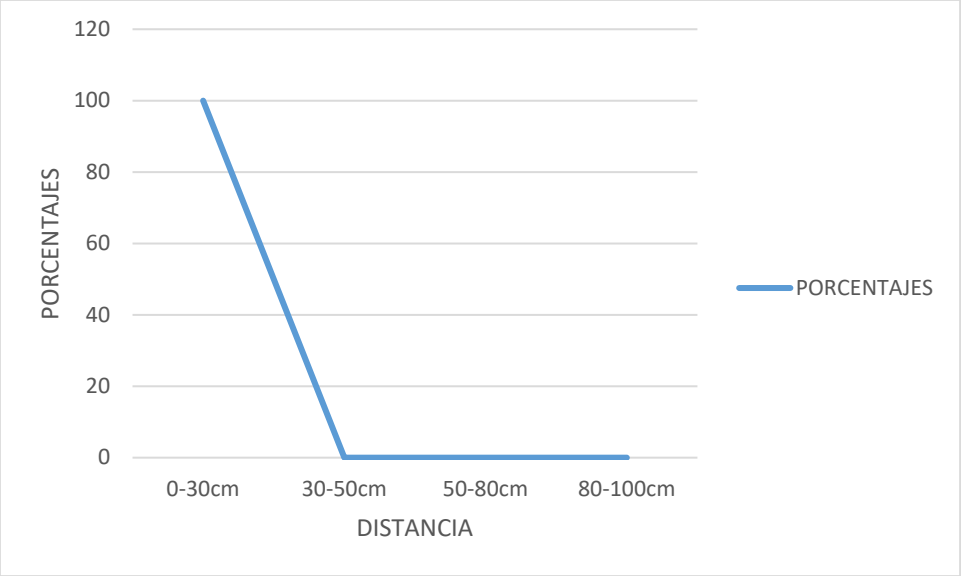
Tabla 8. *Porcentajes identificación de placa.*

PLACA	
DISTANCIA	PORCENTAJES
0-30cm	100
30-50cm	0
50-80cm	0
80-100cm	0

Fuente. Elaboración Propia.

Se ha realizado una gráfica con los valores porcentuales de cada uno de estos datos el mismo que se muestra en la Figura 46.

Figura 46. Porcentaje de identificación de los caracteres de la placa según distancia.



Fuente. Elaboración Propia.

Análisis: En la figura 46, se observa que existe un 100% de identificación de los caracteres de la placa cuando se analiza una imagen a una distancia de hasta 80 cm; en el caso distancias mayores a 50 el porcentaje es del 0%.

Además de la prueba de distancia se desarrolló la evaluación del módulo de detección de los caracteres de la placa considerando un ángulo de desplazamiento en las capturas realizadas, los resultados alcanzados en esta prueba se los adjunta en la Tabla 9.

Tabla 9. *Resultados identificación de placa.*

ANGULO\DISTANCIA	30 CM	50 CM	80 CM	100 CM
10	0	0	0	0
20	0	0	0	0
30	0	0	0	0
40	0	0	0	0

Fuente. Elaboración Propia.

Análisis: Se evidencia en las pruebas realizadas que se debe realizar la captura de la imagen de forma frontal a este elemento, debido que cualquier pequeña variación afecta en la identificación de las letras y números que conforman a la placa vehicular.

En el identificar de color se realizan las pruebas de funcionamiento considerando dos modelos de vehículos en todos sus posibles colores disponibles en el mercado.

Las tablas 10 y 11 muestran los resultados alcanzados; se debe indicar que en el caso de este módulo se han verificado el funcionamiento de cada uno de los procesos que se han programado, como es el del identificador de color en base a la comparación del vehículo ingresado con los de la base de datos. Y también se indican los valores del identificados mediante el clasificador previamente entrenado.

Tabla 10. *Resultados identificación del color - Auto 1.*

#	ROJO		AZUL		BLANCO		NEGRO		GRIS		GRIS RATÓN	
	A	B	A	B	A	B	A	B	A	B	A	B
1	0	1	0	1	0	1	1	1	0	1	1	1
2	1	1	0	1	0	1	1	1	0	1	1	1
3	1	1	0	1	0	1	1	1	1	1	1	1
4	0	1	0	1	0	1	1	1	0	1	0	1
5	0	1	1	1	0	1	1	1	1	1	0	1
6	0	1	1	1	1	1	1	0	1	1	0	1
7	1	1	1	1	0	1	0	1	1	1	0	1
8	1	1	0	1	0	1	1	1	0	1	1	1
9	0	1	1	1	0	1	1	1	0	1	1	1
10	0	1	1	1	0	1	1	1	1	1	0	1

Nota: Método usados A: mediante comparación. B: mediante el clasificador.

Fuente. Elaboración Propia.

Tabla 11. *Resultados identificación del color - Auto 2.*

#	ROJO		AZUL		BLANCO		NEGRO		GRIS		GRIS RATÓN	
	A	B	A	B	A	B	A	B	A	B	A	B
1	0	1	0	1	0	1	0	1	0	1	0	1
2	0	1	0	1	0	1	1	1	0	1	0	1
3	1	0	0	1	0	1	0	1	1	1	1	1
4	1	1	0	1	0	1	1	1	1	1	0	1
5	1	1	1	1	0	1	1	1	1	1	0	1
6	1	1	0	1	0	1	1	1	0	0	1	1
7	1	1	0	1	0	1	1	1	1	1	0	1
8	1	1	0	1	0	1	1	1	1	1	1	1
9	1	1	1	1	0	0	1	1	0	1	0	1
10	1	1	0	1	1	1	0	1	0	1	1	1

Nota: Método usados A: mediante comparación. B: mediante el clasificador.

Fuente. Elaboración Propia.

De forma similar al módulo anterior para realizar la validación del funcionamiento de éste, se analizan los porcentajes de aciertos que poseen cada una de las pruebas realizadas.

En la tabla 12 se analizan los porcentajes de acierto del método de comparación entre la imagen ingresada al sistema y la base de datos formada de la misma.

Tabla 12. *Porcentajes identificación del color mediante el método de comparación.*

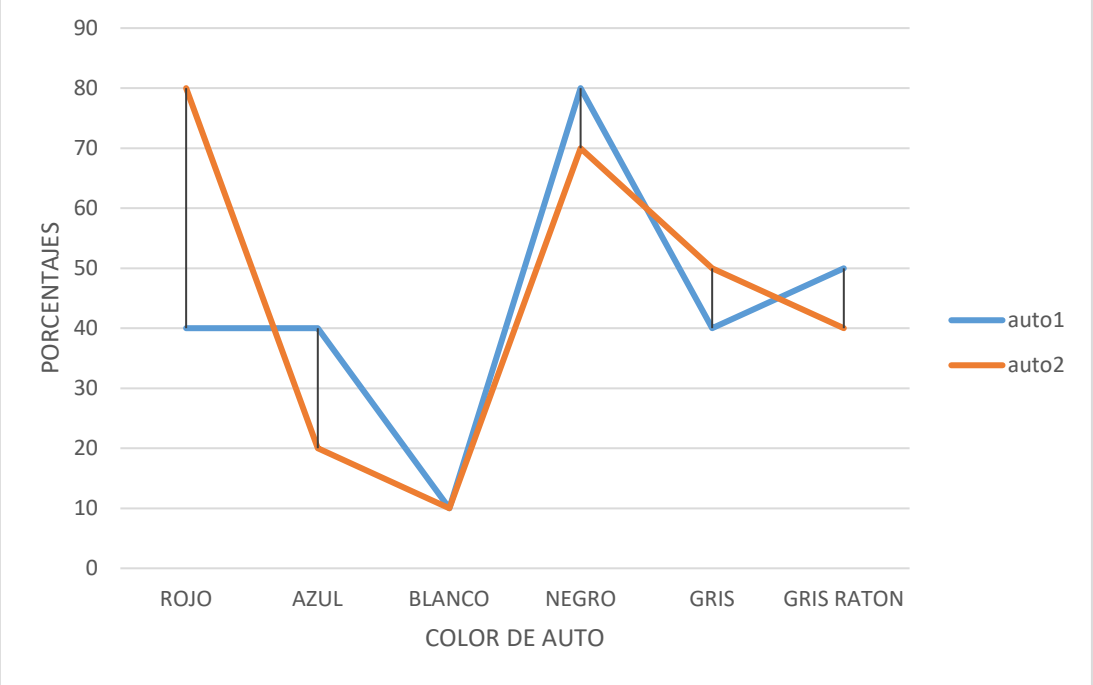
COLORES	AUTO1	AUTO2
Rojo	40	80
Azul	40	20
Blanco	10	10
Negro	80	70
Gris	40	50
Gris ratón	50	40

Nota: valores en porcentaje.

Fuente. Elaboración propia.

Los valores expuestos en la tabla 12 se los representa de forma gráfica mediante la Figura 47.

Figura 47. Porcentaje de identificación de colores de los autos por comparación.



Fuente. Elaboración Propia.

Análisis: En la figura 47 se observa los resultados del método de clasificación por comparación, en el caso del color blanco en los dos vehículos de prueba se logra únicamente una identificación del 10%; en el caso del color negro hay una 70% de identificación para el auto 2 y un 80% para el auto 1.

Para el caso de la prueba realizada sobre el clasificador de color, considerando los porcentajes de aciertos alcanzados en las diferentes pruebas de detección se detallan en la Tabla 13.

Tabla 13. *Porcentajes identificación del color mediante el método del clasificador.*

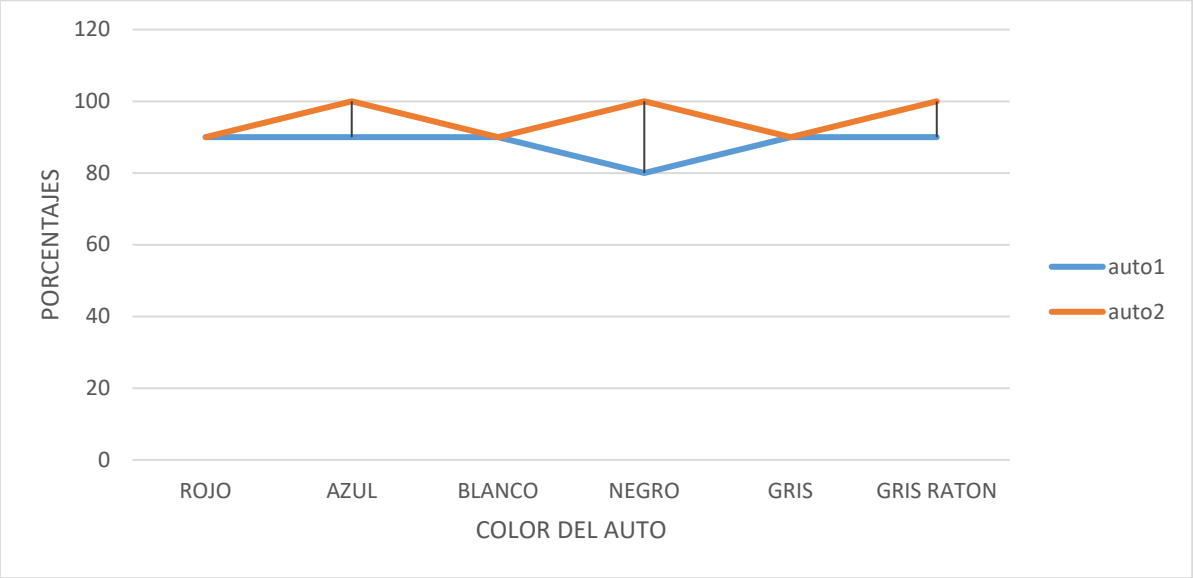
COLORES	AUTO1	AUTO2
Rojo	90	90
Azul	90	100
Blanco	90	90
Negro	80	100
Gris	90	90
Gris ratón	90	100

Nota: valores en porcentaje.

Fuente. Elaboración Propia.

Con los valores de los porcentajes expuestos en la Tabla 13 se realiza la tabla estadística que se muestra en la figura 48.

Figura 48. Porcentaje de identificación de colores de los autos por el clasificador.



Fuente. Elaboración Propia.

Análisis: en la figura 48 se observa los resultados del método de identificación del color aplicando un clasificador, en el caso de los autos de color rojo, blanco y gris el porcentaje de identificación es del 90% en los dos autos; para el color azul se alcanza un porcentaje de identificación del 90% para el auto 1 y 100% para el auto 2. Los resultados del módulo de Identificación del modelo del vehículo se detallan en la tabla 14, para las respectivas pruebas

se rotaron las imágenes más o menos 10 grados, tratando de generar una determinada confusión al sistema.

Tabla 14. *Resultados identificación del vehículo.*

MODELO	SPARK	BEAT	AVEO	CAVALIER	CRUZE	D_MAX	EQUINOX	SAIL	SCROSS	TAHOE	TRACKER	VANZ	VITARA_SZ
1	1	1	1	1	1	1	1	0	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	0	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	0	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	0	1
8	1	1	1	1	1	1	1	1	1	1	1	1	1
9	0	0	1	1	1	1	0	1	1	1	0	1	1
10	0	1	1	1	1	1	1	0	0	1	1	1	1

Fuente. Elaboración Propia.

Los valores porcentuales de la detección, es decir, los aciertos que genera el sistema para cada uno de los modelos tratados se adjuntan en la Tabla 15.

Tabla 15. *Porcentajes identificación del vehículo.*

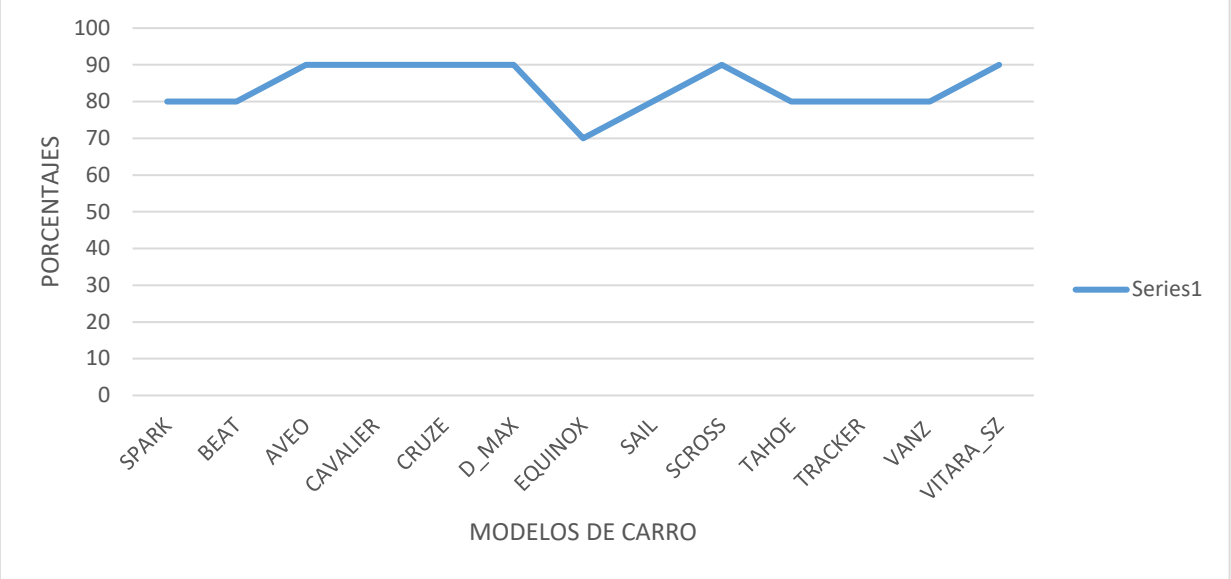
MODELO		MODELO	
SPARK	80	SAIL	80
BEAT	80	SCROSS	90
AVEO	90	TAHOE	80
CAVALIER	90	TRACKER	80
CRUZE	90	VANZ	80
D_MAX	90	VITARA_SZ	90
EQUINOX	70		

Nota: valores en porcentaje.

Fuente. Elaboración Propia.

La representación gráfica de los resultados porcentuales de la Tabla 15 se los muestra en la figura 49.

Figura 49. Porcentaje de identificación de colores de los autos por comparación.



Fuente. Elaboración Propia.

Análisis: en la figura 49 se observa los resultados de la detección acertada del modelo del vehículo analizado, se evidencia que el modelo de vehículo Equinox es el que posee un porcentaje de 70% de identificación, a diferencia de los vehículos de modelo Aveo, Cavalier, Cruze, D-Max, Scroos y Vitara SZ que poseen un 90% de identificación del modelo.

8. CONCLUSIONES Y TRABAJO FUTURO

8.1. Conclusiones

El trabajo es interesante porque permite identificar ciertos parámetros propios de los vehículos de marca Chevrolet que se comercializan en el Ecuador permitiendo identificar de acuerdo a la información que contenga la imagen, los caracteres de la placa, el color o el modelo, de esta forma se puede identificar al vehículo que pueda causar un accidente y fugar del lugar del siniestro; contribuye una solución nueva que permitirá reducir las estadísticas de carros que abandonan el lugar donde se produjo un accidente.

Para el desarrollo de los diversos módulos se aplicó un procesamiento de imágenes en los cuales se pueden resaltar, la aplicación de redimensionamiento de imágenes, operaciones de conversión de imágenes para modificar el espacio de color, operaciones lógicas y morfológicas, se aplicaron filtros, procesos de correlación, de extracción de características y emparejamiento de las mismas, además de despliegue de los resultados en pantalla.

Para el módulo de detección de los caracteres de la placa se debe considerar que la distancia con la que se realice la captura debe ser menor a los 30cm de distancia, de esa forma se logra obtener una detección adecuada de los elementos que forman a la placa vehicular; además la captura realizada para este módulo no debe poseer un ángulo de inclinación debido a que este provoca caos en el sistema, ya que la variación en el ángulo de captura induce un mal funcionamiento de los elementos estructurantes que se consideraron en el desarrollo del módulo.

El desarrollo del identificador del color mediante la comparación entre la base de datos y el vehículo ingresado al sistema no arrojó los resultados deseados, pues logra pocos aciertos al modificar los colores de los vehículos siendo el 10% el valor más bajo de aciertos en la detección del color y un 80% el valor más alto. A diferencia de la identificación llevada a cabo mediante el clasificador Cubic, este arroja resultados favorables para la mayoría de los casos analizados, pues comete pocos errores en clasificar los colores de los vehículos ingresados para su respectivo análisis, se debe destacar que el valor porcentual más bajo que posee es del 80% y el máximo es del 100%, es decir, que tiene un alto nivel de identificación del color.

Para el módulo de detección del modelo del vehículo se logró identificar que el módulo posee un funcionamiento bastante aceptable, ya que el valor porcentual mínimo de reconocimiento se tiene un 70% y ocurre en la detección del modelo Scross y para el resto de los modelos analizados se considera un valor mayor o igual al 80%.

De manera general, se evidencia en los datos estadísticos que se obtienen valores que satisfacen los diversos módulos de detección a criterio de los desarrolladores.

Para mejorar los resultados alcanzados en esta investigación se recomienda que se pueda realizar el desarrollo con el empleo de nuevas tecnologías, ya sea mediante redes neuronales o sistemas de aprendizaje, pues el sistema puede aportar soluciones a varios países de América Latina que poseen una alta estadística de accidentes de tránsito.

8.2. Líneas de trabajo futuro

Se deberá realizar una investigación procurando crear una red neuronal para el aprendizaje de cada módulo, lo cual permitirá ampliar las diversas bases de datos tanto del color como del modelo de los vehículos existentes en una determinada región, además puede permitir crear una identificación de los diversos tonos de un color que pueden poseer los vehículos, además de considerar un data set de imágenes con características reales y no imágenes ideales como se ha realizado en este trabajo.

Identificar el hardware óptimo que permita la adquisición adecuada de la información del vehículo causante del accidente y permita identificar las características propuestas de manera acertada y eficiente.

Se deberá desarrollar una plataforma que permita simular el funcionamiento real del sistema; dotando de sensores, tarjetas de desarrollo y cámaras que permitan el ingreso de información al sistema para su respectivo procesamiento.

Referencias bibliográficas

AEADE. (2021). *BOLETÍN DE PRENSA Ventas de vehículos-Resumen*. https://www.aeade.net/wp-content/uploads/2021/04/BOLETIN-DE-VENTAS-PARA-PRENSA-ABRIL-2021.pdf

Agencia Nacional de Transito. (2020). *Estadísticas de Siniestros de Tránsito*. https://www.ant.gob.ec/?page_id=2670

Bedir Yousif, B., Maher Ata, M., Fawzy, N., & Obaya, M. (2020). *Toward an Optimized Neutrosophic k-Means With Genetic Algorithm for Automatic Vehicle License Plate Recognition (ONKM-AVLPR)*. https://doi.org/10.1109/ACCESS.2020.2979185

Caldas, C. de I. de la U. D. F. J. de, Figueroa, M. A. R., & Palacios, Al. R. (2005). Aplicación de teorías de color en imágenes digitales. *Ingeniería*, *10*(2), 14–22. https://doi.org/10.14483/23448393.2712

Chavan, A. (2020, September 7). *Building RGB Color Classifier: Part 1 | by Ajinkya Chavan | Analytics Vidhya | Medium*. https://medium.com/analytics-vidhya/building-rgb-color-classifier-part-1-af58e3bcfef7

Collado, J., & Hilario, C. (2003). Visión por computador para vehículos inteligentes. *XXIV Jornadas de ….* http://www.ceautomatica.es/old/actividades/jornadas/XXIV/documentos/viar/61.pdf

Corneto, G. L., Silva, F. A., Pereira, D. R., Almeida, L. L., Artero, A. O., Papa, J. P., De Albuquerque, V. H. C., & Sapia, H. M. (2017). A New Method for Automatic Vehicle License Plate Detection. *IEEE Latin America Transactions*, *15*(1), 75–80. https://doi.org/10.1109/TLA.2017.7827890

Du, S., Ibrahim, M., Shehata, M., & Badawy, W. (2013). Automatic license plate recognition (ALPR): A state-of-the-art review. *IEEE Transactions on Circuits and Systems for Video Technology*, *23*(2), 311–325. https://doi.org/10.1109/TCSVT.2012.2203741

Farhat, A., Hommos, O., Al-Zawqari, A., Al-Qahtani, A., Bensaali, F., Amira, A., & Zhai, X. (2018). *Optical character recognition on heterogeneous SoC for HD automatic number plate recognition system*. https://doi.org/10.1186/s13640-018-0298-2

Gnanaprakash, V., Kanthimathi, N., & Saranya, N. (2021). Automatic number plate recognition using deep learning. *IOP Conference Series: Materials Science and Engineering*, *1084*(1), 012027. https://doi.org/10.1088/1757-899x/1084/1/012027

Griffin, L. D. (2006). Optimality of the basic colour categories for classification. *Journal of the Royal Society Interface*, *3*(6), 71–85. https://doi.org/10.1098/rsif.2005.0076

Gu, H.-Z., & Lee, S.-Y. (2012). *A view-invariant and anti-reflection algorithm for car body extraction and color classification*. https://doi.org/10.1007/s11042-012-0996-1

Haggerty, J. M., Wang, X. N., Dickinson, A., O'Malley, C. J., & Martin, E. B. (2014). Segmentation of epidermal tissue with histopathological damage in images of haematoxylin and eosin stained human skin. *BMC Medical Imaging*, *14*(1). https://doi.org/10.1186/1471-2342-14-7

Huamaní Navarrete, P. (2016). Umbralización múltiple utilizando el método de Otsu para reconocer la luz roja en semáforos. *Scientia*, *17*(17), 247–262. https://doi.org/10.31381/scientia.v17i17.393

Korznikov, K. A., Kislov, D. E., Altman, J., Doležal, J., Vozmishcheva, A. S., & Krestov, P. V. (2021). Using u-net-like deep convolutional neural networks for precise tree recognition in very high resolution rgb (Red, green, blue) satellite images. *Forests*, *12*(1), 1–17. https://doi.org/10.3390/f12010066

Lillo, J. M. (2010). *Detección Automática de Señales de Tráfico Mediante Procesamiento Digital de Imagen*. Universidad Rey Juan Carlos. https://eciencia.urjc.es/handle/10115/7851

MathWorks. (2021a). *Etiquetar y medir objetos en una imagen binaria - MATLAB & Simulink - MathWorks España*. https://es.mathworks.com/help/images/label-and-measure-objects-in-a-binary-image.html

MathWorks. (2021b). *Máquinas vectoriales de soporte para la clasificación binaria - MATLAB & Simulink - MathWorks España*. https://es.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html

MathWorks. (2021c). *Support Vector Machine (SVM) - MATLAB & Simulink*. https://es.mathworks.com/discovery/support-vector-machine.html

Mohamed, A. E. (2017). Comparative Study of Four Supervised Machine Learning Techniques

for Classification. In *International Journal of Applied Science and Technology* (Vol. 7, Issue 2). www.ijastnet.com

Palomino, N. L. S., Palomino, N. L. S., & Concha, U. N. R. (2009). Técnicas de Segmentación en Procesamiento Digital de Imágenes. *Revista de Investigación de Sistemas e Informática*, *6*(2), 9–16. https://revistasinvestigacion.unmsm.edu.pe/index.php/sistem/article/view/3299

Rotaru, C., Graf, T., & Zhang, J. (2008). Color image segmentation in HSI space for automotive applications. *Journal of Real-Time Image Processing*, *3*(4), 311–322. https://doi.org/10.1007/s11554-008-0078-9

Torres Cabrera, M. D. (2020). *Reconocimiento Automatico de la placa de un vehiculo de Ecuador*. http://dspace.ups.edu.ec/handle/123456789/18537

Vera, A., Ramos, W., & Rojas, J. (2010). *Diseño de un sistema de seguridad basado en procesamiento de imágenes para el acceso vehicular a un campus*. http://ingelectuq.net/gdsproc/wp-content/uploads/2017/09/2010-vera.pdf

World Health Organization, O. (2018). *Global status report on road safety 2018 (OMS)*. https://www.who.int/publications/i/item/9789241565684

Zhang, Q., Li, J., Zhuo, L., Zhang, H., & Li, X. (2017). Vehicle Color Recognition with Vehicle-Color Saliency Detection and Dual-Orientational Dimensionality Reduction of CNN Deep Features. *Sensing and Imaging*, *18*(1), 20. https://doi.org/10.1007/s11220-017-0173-8

Bay, H., & Tuytelaars, T. (2006). SURF: Speeded Up Robust Features. *Computer Vision – ECCV 2006*, (pp. 404-417). Berlin, Heidelberg.

D., G. L. (2006). Optimality of the basic colour categories for classification. *J. R. Soc. Interface, 3*, 71-85. doi:https://doi.org/10.1098/rsif.2005.0076

E. Rublee, V. R. (2011). ORB: An efficient alternative to SIFT or SURF. *2011 International Conference on Computer Vision* (pp. 2564-2571). IEEE.

Griffin, L. D. (2005, 9 6). Optimality of the basic colour categories for classification. *Journal of The Royal Society Interface, 3*(6), 71-85. doi:https://doi.org/10.1098/rsif.2005.0076

Harris, C., & Stephens, M. (1998). A combined corner and edge detector. *Proceedings*, 147-151.

Keys, R. (1981). Cubic Convolution Interpolation for Digital Image Processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1153-1160.

Lindeberg T., B. L. (2003). Real-Time Scale Selection in Hybrid Multi-scale Representations. *4th International Conference on Scale Space Methods in Computer Vision Location*, (pp. 148-163). Springer .

Lindeberg, T. (1198). Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision*, 79-116.

Lowe, D. (2004). Distinctive image features from scaleinvariant keypoints. *International Journal of Computer Vision*, 91-110.

MATLAB. (2021). *Version: 9.10.* Natick, Massachusetts: The MathWorks Inc.

Melo Morales, J. y. (2018). *Sistema Automático de Reconocimiento de Placas Vehiculares.* Bogotá: Universidad Cooperativa de Colombia.

Mikolajczyk, K. &. (2001). Indexing Based on Scale Invariant Interest Points. *International Conference on Computer Vision*, (pp. 525-531). Vancouver, Canada.

Pearson, K. (1896, 1 1). Mathematical contributions to the theory of evolution. III. Regression, heredity and panmixia. *Transactions of the Royal Society of London, 187*, pp. 253-318.

Rosten, E., & Drummond, T. (2206). Machine Learning for High-Speed Corner Detection. *Computer Vision – ECCV 2006*, (pp. 430-443). Berlin, Heidelberg.

S. Du, M. I. (2013, Feb). Automatic License Plate Recognition (ALPR): A State-of-the-Art Review. *IEEE Transactions on Circuits and Systems for Video Technology, 23*(2), 311-325. doi:10.1109/TCSVT.2012.2203741

Anexo A. Código Matlab

```
function varargout = Placa(varargin)
% PLACA MATLAB code for Placa.fig
%      PLACA, by itself, creates a new PLACA or raises the existing
%      singleton*.
%
%      H = PLACA returns the handle to a new PLACA or the handle to
%      the existing singleton*.
%
%      PLACA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PLACA.M with the given input arguments.
%
%      PLACA('Property','Value',...) creates a new PLACA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Placa_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Placa_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Placa

% Last Modified by GUIDE v2.5 14-Apr-2021 13:24:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Placa_OpeningFcn, ...
                   'gui_OutputFcn',  @Placa_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Placa is made visible.
function Placa_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Placa (see VARARGIN)

% Choose default command line output for Placa
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Placa wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Placa_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
close(Placa);
Menu;
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% close all; clear all; clc;
v=0;
%% elementos de la estructura
load estructuras
%% imagen
k= imread('movil1.jpg'); %se lee la imagen
disp('Seleccione la placa del vehiculo')
disp('Para que continue la ejecucion debe dar doble click')
disp('Procure solo recortar las letras y numeros de la placa')
disp('Sin Distractores')
axes(handles.axes1);
imshow(k);
I=imcrop(k);
figure(1); subplot(1,2,1), imshow(k),title('Imagen Original'), subplot(1,2,2),
imshow(I);title('Imagen Seccionada')
I=rgb2gray(I); I=imadjust(I); % se ajusta la imagen
umbral = graythresh(I); % método Otsu metodo automático
I1 = im2bw(I, umbral); %binariza la imagen
figure(2); subplot(1,2,1); imshow(I); subplot(1,2,2); imshow(I1);
i2=not(I1);
figure(3); subplot(1,2,1); imshow(I1); subplot(1,2,2); imshow(i2);
i3=medfilt2(i2);%pixel aislado
figure(4); subplot(1,2,1); imshow(i2); subplot(1,2,2); imshow(i3);
se=strel('disk',7); %elemento estructurante
i4=imclose(i3,se);
figure(5); subplot(1,2,1); imshow(i3); subplot(1,2,2); imshow(i4);
%% se etiqueta los elementos detectados
i5 = bwlabel(i4);
stats = regionprops(i5, 'all');
v=stats;
% 1er Elemento letra
E1=stats(1).BoundingBox;
l1=imcrop(i4,E1);
% 2do Elemento letra
E2=stats(2).BoundingBox;
l2=imcrop(i4,E2);
% 3er Elemento letra
E3=stats(3).BoundingBox;
l3=imcrop(i4,E3);
% gui[on elemento sin relevancia
E4=stats(4).BoundingBox;
l4=imcrop(i4,E4);
% 1er Elemento #
E5=stats(5).BoundingBox;
l5=imcrop(i4,E5);
% 2do Elemento #
E6=stats(6).BoundingBox;
```

```
l6=imcrop(i4,E6);
% 3er Elemento #
E7=stats(7).BoundingBox;
l7=imcrop(i4,E7);
%se compara los valores
co=isequal(s,v)
if (co==1)
    axes(handles.axes2);
    subplot(3,3,1), imshow(l1),subplot(3,3,2), imshow(l2),subplot(3,3,3), imshow(l3)
    subplot(3,3,4), imshow(l4),subplot(3,3,5), imshow(l5),subplot(3,3,6), imshow(l6)
    subplot(3,3,7), imshow(l7)

else
% 4to Elemento #
    E8=stats(8).BoundingBox;
    l8=imcrop(i4,E8);
    axes(handles.axes2);
    subplot(3,3,1), imshow(l1),subplot(3,3,2), imshow(l2),subplot(3,3,3), imshow(l3)
    subplot(3,3,4), imshow(l4),subplot(3,3,5), imshow(l5),subplot(3,3,6), imshow(l6)
    subplot(3,3,7), imshow(l7),subplot(3,3,8), imshow(l8)
end
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
close all;
clear all;
clc;
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton5.


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function pushbutton6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
warndlg('Se debe seleccionar el área de interes en este caso solo se deberá seleccionar
la placa, para realizar el recorte selccione el [area de interés y de doble clik')


% --- Executes during object creation, after setting all properties.
function pushbutton3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global movil1
global movil2
```

```
206   [filename path]=uigetfile(('*.jpeg'),'Cargar Imagen');
207   if isequal(filename,0)
208       return
209   else
210       folder='E:\TFM\MatLab\imagenes'
211       movil1=imread(strcat(path,filename));
212       axes(handles.axes1);
213       imshow(movil1);
214       imwrite(movil1,'movil1.jpg')
215   end
216
217
218   % --- Executes during object creation, after setting all properties.
219   function pushbutton7_CreateFcn(hObject, eventdata, handles)
220   % hObject    handle to pushbutton7 (see GCBO)
221   % eventdata  reserved - to be defined in a future version of MATLAB
222   % handles    empty - handles not created until after all CreateFcns called
223
```

```m
function varargout = Color(varargin)
% COLOR MATLAB code for Color.fig
%      COLOR, by itself, creates a new COLOR or raises the existing
%      singleton*.
%
%      H = COLOR returns the handle to a new COLOR or the handle to
%      the existing singleton*.
%
%      COLOR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in COLOR.M with the given input arguments.
%
%      COLOR('Property','Value',...) creates a new COLOR or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Color_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Color_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Color

% Last Modified by GUIDE v2.5 27-Jan-2021 15:40:17

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Color_OpeningFcn, ...
                   'gui_OutputFcn',  @Color_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Color is made visible.
function Color_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Color (see VARARGIN)

% Choose default command line output for Color
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Color wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Color_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
global movil1
global movil2
v=0;
%% elementos de la estructura
load estructuras
%% imagen
vim= imread('movil1.jpg'); %se lee la imagen
[c9,f9,v9]=impixel(vim);
%% BASE DE DATOS LECTURA
v=imread('Vino.jpg');[c1,f1,v1]=impixel(v,20,36);
r=imread('Rojo.jpg');[c2,f2,v2]=impixel(r,20,36);
p=imread('Gris-Plomo.jpg');[c3,f3,v3]=impixel(p,20,36);
ag=imread('Plata.jpg');[c4,f4,v4]=impixel(ag,20,36);
n=imread('Negro.jpg'); [c5,f5,v5]=impixel(n,20,36);
d=imread('Dorado.jpg');[c6,f6,v6]=impixel(d,20,36);
b=imread('Blanco.jpg');[c7,f7,v7]=impixel(b,20,36);
a=imread('Azul.jpg');[c8,f8,v8]=impixel(a,20,36);
m=imread('Amarillo.jpg');[c10,f10,v10]=impixel(m,20,36);

%% LECTURA DEL VEHICULO SINIESTRADO
%vim=imread('vehiculo4.JPG'); [c9,f9,v9]=impixel(vim,446,183);
%imshow(vim);
%v9min=v9-20;

%% RANGO DE VALORES DE TOLERANCIA-COLOR
vmax=v1+10; vmin=v1-10;vino=[vmin vmax];
rmax=v2+10; rmin=v2-10;rojo=[rmin rmax];
pmax=v3+10; pmin=v3-10;plomo=[pmin pmax];
agmax=v4+10; agmin=v4-10;plata=[agmin agmax];
nmax=v5+10; nmin=v5-10;negro=[nmin nmax];
dmax=v6+10; dmin=v6-10;dorado=[dmin dmax];
bmax=v7+10; bmin=v7-10;blanco=[bmin bmax];
amax=v8+30; amin=v8-30;azul=[amin amax];
mmax=v10+30; mmin=v10-30;amarillo=[mmin mmax];

%% COMPARACION DE COLORES
if ((v9>vmin)&(v9<vmax))
    disp('el vehiculo es color vino')
elseif ((v9>rmin)&(v9<rmax))
    disp('el vehiculo es color rojo')
elseif ((v9>pmin)&(v9<pmax))
    disp('el vehiculo es color plomo')
elseif ((v9>agmin)&(v9<agmax))
    disp('el vehiculo es color plata')
elseif ((v9>nmin)&(v9<nmax))
    disp('el vehiculo es color negro')
elseif ((v9>dmin)&(v9<dmax))
    disp('el vehiculo es color dorado')
elseif ((v9>bmin)&(v9<bmax))
    disp('el vehiculo es color blanco')
elseif ((v9>amin)&(v9<amax))
    disp('el vehiculo es color azul')
elseif ((v9>mmin)&(v9<mmax))
    disp('el vehiculo es color amarillo')
else
    disp('el color no existe en la base de datos')
end
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```
139
140 % --- Executes on button press in pushbutton2.
141 function pushbutton2_Callback(hObject, eventdata, handles)
142 close(Color);
143 Menu;
144 % hObject    handle to pushbutton2 (see GCBO)
145 % eventdata  reserved - to be defined in a future version of MATLAB
146 % handles    structure with handles and user data (see GUIDATA)
147
148
149 % --- Executes on button press in pushbutton3.
150 function pushbutton3_Callback(hObject, eventdata, handles)
151 close all;
152 clear all;
153 clc;
154 % hObject    handle to pushbutton3 (see GCBO)
155 % eventdata  reserved - to be defined in a future version of MATLAB
156 % handles    structure with handles and user data (see GUIDATA)
157
```

```m
function varargout = COLOR_NUEVO(varargin)
% COLOR_NUEVO MATLAB code for COLOR_NUEVO.fig
%      COLOR_NUEVO, by itself, creates a new COLOR_NUEVO or raises the existing
%      singleton*.
%
%      H = COLOR_NUEVO returns the handle to a new COLOR_NUEVO or the handle to
%      the existing singleton*.
%
%      COLOR_NUEVO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in COLOR_NUEVO.M with the given input arguments.
%
%      COLOR_NUEVO('Property','Value',...) creates a new COLOR_NUEVO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before COLOR_NUEVO_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to COLOR_NUEVO_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help COLOR_NUEVO

% Last Modified by GUIDE v2.5 18-May-2021 15:37:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @COLOR_NUEVO_OpeningFcn, ...
                   'gui_OutputFcn',  @COLOR_NUEVO_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before COLOR_NUEVO is made visible.
function COLOR_NUEVO_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to COLOR_NUEVO (see VARARGIN)

% Choose default command line output for COLOR_NUEVO
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes COLOR_NUEVO wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = COLOR_NUEVO_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate axes1


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
az=('azul.jpg');bl=('blanco.jpg'); do=('Dorado.jpg'); ne=('Negro.jpg');
pl=('Plata.jpg'); pb=('Plomo.jpg'); ro=('Rojo.jpg'); vi=('Vino.jpg');
axes(handles.axes1);
    subplot(3,3,1), imshow(az),subplot(3,3,2), imshow(bl),subplot(3,3,3), imshow(do)
    subplot(3,3,4), imshow(ne),subplot(3,3,5), imshow(pl),subplot(3,3,6), imshow(pb)
    subplot(3,3,7), imshow(ro), subplot(3,3,8), imshow(vi)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
k= imread('movil1.jpg'); %se lee la imagen
axes(handles.axes2);
imshow(k);
vim=k;
[c9,f9,v9]=impixel(vim,446,183);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
v=imread('Vino.jpg');[c1,f1,v1]=impixel(v,20,36);
r=imread('Rojo.jpg');[c2,f2,v2]=impixel(r,20,36);
p=imread('Plomo.jpg');[c3,f3,v3]=impixel(p,20,36);
ag=imread('Plata.jpg');[c4,f4,v4]=impixel(ag,20,36);
n=imread('Negro.jpg'); [c5,f5,v5]=impixel(n,20,36);
d=imread('Dorado.jpg');[c6,f6,v6]=impixel(d,20,36);
b=imread('Blanco.jpg');[c7,f7,v7]=impixel(b,20,36);
a=imread('Azul.jpg');[c8,f8,v8]=impixel(a,20,36);
m=imread('Amarillo.jpg');[c10,f10,v10]=impixel(m,20,36);
%% vehiculo siniestrado
vim= imread('movil1.jpg'); %se lee la imagen
[c9,f9,v9]=impixel(vim,446,183);
%% RANGO DE VALORES DE TOLERANCIA-COLOR
vmax=v1+10; vmin=v1-10;vino=[vmin vmax];
rmax=v2+10; rmin=v2-10;rojo=[rmin rmax];
pmax=v3+10; pmin=v3-10;plomo=[pmin pmax];
agmax=v4+10; agmin=v4-10;plata=[agmin agmax];
nmax=v5+10; nmin=v5-10;negro=[nmin nmax];
dmax=v6+10; dmin=v6-10;dorado=[dmin dmax];
bmax=v7+10; bmin=v7-10;blanco=[bmin bmax];
amax=v8+30; amin=v8-30;azul=[amin amax];
mmax=v10+30; mmin=v10-30;amarillo=[mmin mmax];

%% COMPARACION DE COLORES
if ((v9>vmin)&(v9<vmax))
```

```
    warndlg('el vehiculo es color vino')
    disp('el vehiculo es color vino')
elseif ((v9>rmin)&(v9<rmax))
    warndlg('el vehiculo es color rojo')
    disp('el vehiculo es color rojo')
elseif ((v9>pmin)&(v9<pmax))
    warndlg('el vehiculo es color plomo')
    disp('el vehiculo es color plomo')
elseif ((v9>agmin)&(v9<agmax))
    warndlg('el vehiculo es color plata')
    disp('el vehiculo es color plata')
elseif ((v9>nmin)&(v9<nmax))
    warndlg('el vehiculo es color vino')
    disp('el vehiculo es color negro')
elseif ((v9>dmin)&(v9<dmax))
    warndlg('el vehiculo es color dorado')
    disp('el vehiculo es color dorado')
elseif ((v9>bmin)&(v9<bmax))
    warndlg('el vehiculo es color blanco')
    disp('el vehiculo es color blanco')
elseif ((v9>amin)&(v9<amax))
    warndlg('el vehiculo es color azul')
    disp('el vehiculo es color azul')
elseif ((v9>mmin)&(v9<mmax))
    warndlg('el vehiculo es color amarillo')
    disp('el vehiculo es color amarillo')
else
    warndlg('el color no existe en la base de datos')
    disp('el color no existe en la base de datos')
end


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(COLOR_NUEVO);
Menu;


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all;
clear all;
clc;


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
load('clasificador.mat')
vim= imread('movil1.jpg'); %se lee la imagen
[c9,f9,v9]=impixel(vim)
global ColorCubicKNN;
global colorbd1;

v91=v9(1);
v92=v9(2);
v93=v9(3);
save('col.mat', 'v91', 'v92', 'v93')
load('col.mat')
colorbd1.label(1) = '<undefined>';
colorbd1.red(1) = v91
```

```
colorbd1.green(1) = v92
colorbd1.blue(1)=v93

yfit = ColorCubicKNN.predictFcn(colorbd1);
disp(yfit)
%warndlg(yfit)
%warndlg('el color es:'\yfit);

```

```m
function varargout = Modelo(varargin)
% Modelo MATLAB code for Modelo.fig
%      Modelo, by itself, creates a new Modelo or raises the existing
%      singleton*.
%
%      H = Modelo returns the handle to a new Modelo or the handle to
%      the existing singleton*.
%
%      Modelo('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in Modelo.M with the given input arguments.
%
%      Modelo('Property','Value',...) creates a new Modelo or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Modelo_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Modelo_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Modelo

% Last Modified by GUIDE v2.5 21-Apr-2021 13:08:19

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Modelo_OpeningFcn, ...
                   'gui_OutputFcn',  @Modelo_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Modelo is made visible.
function Modelo_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Modelo (see VARARGIN)

% Choose default command line output for Modelo
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Modelo wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Modelo_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate axes1


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
v1=('AVEO.jpg');v2=('BEAT.jpg'); v3=('CAVALIER.jpg'); v4=('CRUZE.jpg');
v5=('D_MAX.jpg'); v6=('EQUINOX.jpg'); v7=('SAIL.jpg'); v8=('SCROSS.jpg');
v9=('SPARK.jpg'); v10=('TAHOE.jpg'); v11=('TRACKER.jpg'); v12=('VANS.jpg');
v13=('VITARASZ.jpg');
axes(handles.axes1);
    subplot(5,3,1), imshow(v1),subplot(5,3,2), imshow(v2),subplot(5,3,3), imshow(v3)
    subplot(5,3,4), imshow(v4),subplot(5,3,5), imshow(v5),subplot(5,3,6), imshow(v6)
    subplot(5,3,7), imshow(v7), subplot(5,3,8), imshow(v8),subplot(5,3,9), imshow(v9)
    subplot(5,3,10), imshow(v10), subplot(5,3,11), imshow(v11),subplot(5,3,12),
    imshow(v12)
    subplot(5,3,13), imshow(v13)

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
k= imread('movil1.jpg'); %se lee la imagen
axes(handles.axes2);
imshow(k);
vim=k; vim=imresize(vim,[280
566],'bicubic');vim=rgb2gray(vim);%vim=im2bw(vim);[c9,f9,v9]=impixel(vim,446,183);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
c1=imread('VITARASZ.JPG');c1=imresize(c1,[280
566],'bicubic');c1=rgb2gray(c1);%c1=im2bw(c1);
c2=imread('VANS.JPG'); c2=imresize(c2,[280 566],'bicubic');c2=rgb2gray(c2);
%c2=im2bw(c2);
c3=imread('TRACKER.JPG'); c3=imresize(c3,[280
566],'bicubic');c3=rgb2gray(c3);%c3=im2bw(c3);
c4=imread('TAHOE.JPG');c4=imresize(c4,[280 566],'bicubic'); c4=rgb2gray(c4);
%c4=im2bw(c4);
c5=imread('SPARK.JPG');c5=imresize(c5,[280 566],'bicubic'); c5=rgb2gray(c5);
%c5=im2bw(c5);
c6=imread('SCROSS.JPG');c6=imresize(c6,[280 566],'bicubic');
c6=rgb2gray(c6);%c6=im2bw(c6);
c7=imread('SAIL.JPG');c7=imresize(c7,[280 566],'bicubic'); c7=rgb2gray(c7);%c7=im2bw(c7);
c8=imread('EQUINOX.JPG');c8=imresize(c8,[280 566],'bicubic');
c8=rgb2gray(c8);%c8=im2bw(c8);
c9=imread('D_MAX.JPG');c9=imresize(c9,[280 566],'bicubic');
c9=rgb2gray(c9);%c9=im2bw(c9);
c10=imread('CRUZE.JPG');c10=imresize(c10,[280 566],'bicubic');
c10=rgb2gray(c10);%c10=im2bw(c10);
c11=imread('CAVALIER.JPG');c11=imresize(c11,[280
566],'bicubic');c11=rgb2gray(c11);%c11=im2bw(c11);
```

```
c12=imread('BEAT.JPG');c12=imresize(c12,[280 566],'bicubic'); c12=rgb2gray(c12);
%c12=im2bw(c12);
c13=imread('AVEO.JPG');c13=imresize(c13,[280 566],'bicubic');
c13=rgb2gray(c13);%c13=im2bw(c13);
k= imread('movil1.jpg'); %se lee la imagen
vim=k; vim=imresize(vim,[280
566],'bicubic');vim=rgb2gray(vim);%vim=im2bw(vim);[c9,f9,v9]=impixel(vim,446,183);
%vim=imread('vehiculo1.JPG');vim=imresize(vim,[280
566],'bicubic');vim=rgb2gray(vim);%vim=im2bw(vim);
%k1=std2(c1); k2=std2(c2); k3=std2(c3);
%kc1=corrcoef(c1,vim)

k1=corr2(c1,vim);
k2=corr2(c2,vim);
k3=corr2(c3,vim);
k4=corr2(c4,vim);
k5=corr2(c5,vim);
k6=corr2(c6,vim);
k7=corr2(c7,vim);
k8=corr2(c8,vim);
k9=corr2(c9,vim);
k10=corr2(c10,vim);
k11=corr2(c11,vim);
k12=corr2(c12,vim);
k13=corr2(c13,vim);

k=[k1 k2 k3 k4 k5 k6 k7 k8 k9 k10 k11 k12 k13]
k1=max(k)
kk=find((k==k1))
%kk=find((k>0.8))%|(k<-.85))
%kk=find(max(k))
if (kk==1)
    warndlg('MODELO: VITARA SZ')
    disp('MODELO: VITARA SZ')
    pc1=detectSURFFeatures(c1);
    figure(1), subplot(1,2,1);imshow(c1);hold on; plot(pc1.selectStrongest(100))
    pvim=detectSURFFeatures(vim);
    figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
    [f1,vpc1]=extractFeatures(c1,pc1);
    [f2,vpim2]=extractFeatures(vim,pvim);
    ip=matchFeatures(f1,f2);
    mp1=vpc1(ip(:,1));
    mp2=vpim2(ip(:,2));
    ax=axes;
    figure(2);showMatchedFeatures(c1,vim,mp1,mp2,'montage', 'Parent',ax)%'falsecolor');

    [tform,inlierPtsDistorted,inlierPtsOriginal] =
    estimateGeometricTransform(mp1,mp2,'similarity');
    axes(handles.axes3);
    showMatchedFeatures(c1,vim,inlierPtsOriginal,inlierPtsDistorted);

elseif(kk==2)
    disp('MODELO: VANS 300 NS')
    warndlg('MODELO: VANS 300 NS')
    pc1=detectSURFFeatures(c2);
    figure(1), subplot(1,2,1);imshow(c2);hold on; plot(pc1.selectStrongest(100))
    pvim=detectSURFFeatures(vim);
    figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
    [f1,vpc1]=extractFeatures(c2,pc1);
    [f2,vpim2]=extractFeatures(vim,pvim);
    ip=matchFeatures(f1,f2);
    mp1=vpc1(ip(:,1));
    mp2=vpim2(ip(:,2));
    figure(2);showMatchedFeatures(c2,vim,mp1,mp2,'falsecolor');
    [tform,inlierPtsDistorted,inlierPtsOriginal] =
    estimateGeometricTransform(mp1,mp2,'similarity');
    axes(handles.axes3);
    showMatchedFeatures(c2,vim,inlierPtsOriginal,inlierPtsDistorted);

```

```
elseif(kk==3)
    warndlg('MODELO: TRACKER')
    disp('MODELO: TRACKER')
    pc1=detectSURFFeatures(c3);
    figure(1), subplot(1,2,1);imshow(c3);hold on; plot(pc1.selectStrongest(100))
    pvim=detectSURFFeatures(vim);
    figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
    [f1,vpc1]=extractFeatures(c3,pc1);
    [f2,vpim2]=extractFeatures(vim,pvim);
    ip=matchFeatures(f1,f2);
    mp1=vpc1(ip(:,1));
    mp2=vpim2(ip(:,2));
    figure(2);showMatchedFeatures(c3,vim,mp1,mp2,'falsecolor');
    [tform,inlierPtsDistorted,inlierPtsOriginal] =
    estimateGeometricTransform(mp1,mp2,'similarity');
    axes(handles.axes3);showMatchedFeatures(c3,vim,inlierPtsOriginal,inlierPtsDistorted);

elseif(kk==4)
    warndlg('MODELO: TAHOE')
    disp('MODELO: TAHOE')
    pc1=detectSURFFeatures(c4);
    figure(1), subplot(1,2,1);imshow(c4);hold on; plot(pc1.selectStrongest(100))
    pvim=detectSURFFeatures(vim);
    figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
    [f1,vpc1]=extractFeatures(c4,pc1);
    [f2,vpim2]=extractFeatures(vim,pvim);
    ip=matchFeatures(f1,f2);
    mp1=vpc1(ip(:,1));
    mp2=vpim2(ip(:,2));
    figure(2);showMatchedFeatures(c4,vim,mp1,mp2,'falsecolor');
    [tform,inlierPtsDistorted,inlierPtsOriginal] =
    estimateGeometricTransform(mp1,mp2,'similarity');
    axes(handles.axes3);showMatchedFeatures(c4,vim,inlierPtsOriginal,inlierPtsDistorted);

elseif(kk==5)
    warndlg('MODELO: SPARK')
    disp('MODELO: SPARK')
    pc1=detectSURFFeatures(c5);
    figure(1), subplot(1,2,1);imshow(c5);hold on; plot(pc1.selectStrongest(100))
    pvim=detectSURFFeatures(vim);
    figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
    [f1,vpc1]=extractFeatures(c5,pc1);
    [f2,vpim2]=extractFeatures(vim,pvim);
    ip=matchFeatures(f1,f2);
    mp1=vpc1(ip(:,1));
    mp2=vpim2(ip(:,2));

    %ax=axes;
    figure(2);showMatchedFeatures(c5,vim,mp1,mp2,'falsecolor');

    %figure(2);showMatchedFeatures(c5,vim,mp1,mp2,'falsecolor');
    [tform,inlierPtsDistorted,inlierPtsOriginal] =
    estimateGeometricTransform(mp1,mp2,'similarity');
    axes(handles.axes3);
    showMatchedFeatures(c5,vim,inlierPtsOriginal,inlierPtsDistorted);

elseif(kk==6)
    warndlg('MODELO: SCROSS')
    disp('MODELO: SCROSS');
    %pc6=detectSURFFeatures(c6);
    pc1=detectSURFFeatures(c6);
    figure(1), subplot(1,2,1);imshow(c6);hold on; plot(pc1.selectStrongest(100))
    pvim=detectSURFFeatures(vim);
    figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
    [f1,vpc1]=extractFeatures(c6,pc1);
    [f2,vpim2]=extractFeatures(vim,pvim);
    ip=matchFeatures(f1,f2);
    mp1=vpc1(ip(:,1));
    mp2=vpim2(ip(:,2));
```

```
253         figure(2);showMatchedFeatures(c6,vim,mp1,mp2,'falsecolor');
254         [tform,inlierPtsDistorted,inlierPtsOriginal] =
            estimateGeometricTransform(mp1,mp2,'similarity');
255         axes(handles.axes3);
            showMatchedFeatures(c6,vim,inlierPtsOriginal,inlierPtsDistorted);
256
257     elseif(kk==7)
258         warndlg('MODELO: SAIL')
259         disp('MODELO: SAIL')
260         %pc7=detectSURFFeatures(c7);
261         pc1=detectSURFFeatures(c7);
262         figure(1), subplot(1,2,1);imshow(c7);hold on; plot(pc1.selectStrongest(100))
263         pvim=detectSURFFeatures(vim);
264         figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
265         [f1,vpc1]=extractFeatures(c7,pc1);
266         [f2,vpim2]=extractFeatures(vim,pvim);
267         ip=matchFeatures(f1,f2);
268         mp1=vpc1(ip(:,1));
269         mp2=vpim2(ip(:,2));
270         figure(2);showMatchedFeatures(c7,vim,mp1,mp2,'falsecolor');
271         [tform,inlierPtsDistorted,inlierPtsOriginal] =
            estimateGeometricTransform(mp1,mp2,'similarity');
272         axes(handles.axes3);
            showMatchedFeatures(c7,vim,inlierPtsOriginal,inlierPtsDistorted);
273
274     elseif(kk==8)
275         %pc8=detectSURFFeatures(c8);
276         warndlg('MODELO: EQUINOX')
277         disp('MODELO: EQUINOX')
278         pc1=detectSURFFeatures(c8);
279         figure(1), subplot(1,2,1);imshow(c8);hold on; plot(pc1.selectStrongest(100))
280         pvim=detectSURFFeatures(vim);
281         figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
282         [f1,vpc1]=extractFeatures(c8,pc1);
283         [f2,vpim2]=extractFeatures(vim,pvim);
284         ip=matchFeatures(f1,f2);
285         mp1=vpc1(ip(:,1));
286         mp2=vpim2(ip(:,2));
287         figure(2);showMatchedFeatures(c8,vim,mp1,mp2,'falsecolor');
288         [tform,inlierPtsDistorted,inlierPtsOriginal] =
            estimateGeometricTransform(mp1,mp2,'similarity');
289         axes(handles.axes3);
            showMatchedFeatures(c8,vim,inlierPtsOriginal,inlierPtsDistorted);
290
291     elseif(kk==9)
292         %pc9=detectSURFFeatures(c9);
293         warndlg('MODELO: LUV D-MAX')
294         disp('MODELO: LUV D-MAX')
295         pc1=detectSURFFeatures(c9);
296         figure(1), subplot(1,2,1);imshow(c9);hold on; plot(pc1.selectStrongest(100))
297         pvim=detectSURFFeatures(vim);
298         figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
299         [f1,vpc1]=extractFeatures(c9,pc1);
300         [f2,vpim2]=extractFeatures(vim,pvim);
301         ip=matchFeatures(f1,f2);
302         mp1=vpc1(ip(:,1));
303         mp2=vpim2(ip(:,2));
304         figure(2);showMatchedFeatures(c9,vim,mp1,mp2,'falsecolor');
305         [tform,inlierPtsDistorted,inlierPtsOriginal] =
            estimateGeometricTransform(mp1,mp2,'similarity');
306         axes(handles.axes3);
            showMatchedFeatures(c9,vim,inlierPtsOriginal,inlierPtsDistorted);
307
308     elseif(kk==10)
309         warndlg('MODELO: CRUZE')
310         disp('MODELO: CRUZE')
311         %pc10=detectSURFFeatures(c10);
312         pc1=detectSURFFeatures(c10);
313         figure(1), subplot(1,2,1);imshow(c10);hold on; plot(pc1.selectStrongest(100))
```

```
314         pvim=detectSURFFeatures(vim);
315         figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
316         [f1,vpc1]=extractFeatures(c10,pc1);
317         [f2,vpim2]=extractFeatures(vim,pvim);
318         ip=matchFeatures(f1,f2);
319         mp1=vpc1(ip(:,1));
320         mp2=vpim2(ip(:,2));
321         figure(2);showMatchedFeatures(c10,vim,mp1,mp2,'falsecolor');
322         [tform,inlierPtsDistorted,inlierPtsOriginal] =
            estimateGeometricTransform(mp1,mp2,'similarity');
323         axes(handles.axes3);
            showMatchedFeatures(c10,vim,inlierPtsOriginal,inlierPtsDistorted);
324
325     elseif(kk==11)
326         warndlg('MODELO: CAVALIER')
327         disp('MODELO: CAVALIER')
328         %pc11=detectSURFFeatures(c11);
329         pc1=detectSURFFeatures(c11);
330         figure(1), subplot(1,2,1);imshow(c11);hold on; plot(pc1.selectStrongest(100))
331         pvim=detectSURFFeatures(vim);
332         figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
333         [f1,vpc1]=extractFeatures(c11,pc1);
334         [f2,vpim2]=extractFeatures(vim,pvim);
335         ip=matchFeatures(f1,f2);
336         mp1=vpc1(ip(:,1));
337         mp2=vpim2(ip(:,2));
338         figure(2);showMatchedFeatures(c11,vim,mp1,mp2,'falsecolor');
339         [tform,inlierPtsDistorted,inlierPtsOriginal] =
            estimateGeometricTransform(mp1,mp2,'similarity');
340         axes(handles.axes3);
            showMatchedFeatures(c11,vim,inlierPtsOriginal,inlierPtsDistorted);
341
342     elseif(kk==12)
343         warndlg('MODELO: BEAT')
344         disp('MODELO: BEAT')
345         %pc12=detectSURFFeatures(c12);
346         pc1=detectSURFFeatures(c12);
347         figure(1), subplot(1,2,1);imshow(c12);hold on; plot(pc1.selectStrongest(100))
348         pvim=detectSURFFeatures(vim);
349         figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
350         [f1,vpc1]=extractFeatures(c12,pc1);
351         [f2,vpim2]=extractFeatures(vim,pvim);
352         ip=matchFeatures(f1,f2);
353         mp1=vpc1(ip(:,1));
354         mp2=vpim2(ip(:,2));
355         figure(2);showMatchedFeatures(c12,vim,mp1,mp2,'falsecolor');
356         [tform,inlierPtsDistorted,inlierPtsOriginal] =
            estimateGeometricTransform(mp1,mp2,'similarity');
357         axes(handles.axes3);
            showMatchedFeatures(c12,vim,inlierPtsOriginal,inlierPtsDistorted);
358
359     elseif(kk==13)
360         warndlg('MODELO: AVEO')
361         disp('MODELO: AVEO')
362         %pc13=detectSURFFeatures(c13);
363         pc1=detectSURFFeatures(c13);
364         figure(1), subplot(1,2,1);imshow(c13);hold on; plot(pc1.selectStrongest(100))
365         pvim=detectSURFFeatures(vim);
366         figure(1), subplot(1,2,2);imshow(vim);hold on; plot(pvim.selectStrongest(100))
367         [f1,vpc1]=extractFeatures(c13,pc1);
368         [f2,vpim2]=extractFeatures(vim,pvim);
369         ip=matchFeatures(f1,f2);
370         mp1=vpc1(ip(:,1));
371         mp2=vpim2(ip(:,2));
372         figure(2);showMatchedFeatures(c13,vim,mp1,mp2,'falsecolor');
373         [tform,inlierPtsDistorted,inlierPtsOriginal] =
            estimateGeometricTransform(mp1,mp2,'similarity');
374         axes(handles.axes3);
            showMatchedFeatures(c13,vim,inlierPtsOriginal,inlierPtsDistorted);
```

```
375
376   else
377       disp('ERROR EN LA BASE DE DATOS')
378   end
379
380   % --- Executes on button press in pushbutton4.
381   function pushbutton4_Callback(hObject, eventdata, handles)
382   % hObject    handle to pushbutton4 (see GCBO)
383   % eventdata  reserved - to be defined in a future version of MATLAB
384   % handles    structure with handles and user data (see GUIDATA)
385   close(Modelo);
386   Menu;
387
388
389   % --- Executes on button press in pushbutton5.
390   function pushbutton5_Callback(hObject, eventdata, handles)
391   % hObject    handle to pushbutton5 (see GCBO)
392   % eventdata  reserved - to be defined in a future version of MATLAB
393   % handles    structure with handles and user data (see GUIDATA)
394   close all;
395   clear all;
396   clc;
397
```