UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO

DE INGENIERO EN MECATRÓNICA

TEMA:

“SISTEMA DE GESTIÓN Y AUTOMATIZACIÓN DEL ACCESO A UN PARQUEADERO”

AUTOR: Alex Javier Cuzco Cualchi

DIRECTOR: MSc. Cosme Damián Mejía Echeverría

IBARRA-ECUADOR

2020

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	172680542-5		
APELLIDOS Y NOMBRES:	CUZCO CUALCHI ALEX JAVIER		
DIRECCIÓN:	PEDRO MONCAYO-TUPIGACHI		
EMAIL:	ajcuzcoc@utn.edu.ec		
TELEFONO FIJO:	(02) 2119 234	**TELÉFONO MÓVIL:**	(+593) 962960950

DATOS DE LA OBRA	
TÍTULO:	"SISTEMA DE GESTIÓN Y AUTOMATIZACIÓN DEL ACCESO A UN PARQUEADERO."
AUTOR:	CUZCO CUALCHI ALEX JAVIER
FECHA:	NOVIEMBRE, 2020
SOLO PARA TRABAJOS DE GRADOS	
PROGRAMA:	PREGRADO
TÍTULO POR EL QUE OPTA:	Ingeniero en Mecatrónica
DIRECTOR:	MSc. Cosme Damián Mejia Echeverría

CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 8 días del mes de junio de 2021

EL AUTOR:

Cuzco Cualchi Alex Javier

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

En calidad de director de trabajo de grado "SISTEMA DE GESTIÓN Y AUTOMATIZACIÓN DEL ACCESO A UN PARQUEADERO", presentando por el egresado CUZCO CUALCHI ALEX JAVIER, para optar por el título de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, diciembre del 2020

MsC. Cosme Mejía

DIRECTOR DE TESIS

AGRADECIMIENTO

A mis padres, Lucia y Juan por su apoyo incondicional, por su comprensión, paciencia, amor y confianza depositada en mi para realizarme profesionalmente y cumplir mis metas, todas las acciones buenas que he realizado son un reflejo del esfuerzo conjunto.

A mi hermano Fabricio y hermana Dayana, por su cariño, apoyo, compañía y comprensión en cada etapa de mi vida.

A mi numerosa y muy querida familia, quienes siempre han guiado mis pasos y aplaudido mis pequeños logros, absolutamente todos fueron esenciales en este camino recorrido y que ahora culmino con éxito. Sin ustedes no lo hubiera logrado.

A Katherine, por su cariño e importante apoyo incondicional ayudándome a crecer, trazando nuevas metas y enseñándome a soñar en grande.

A la Universidad Técnica del Norte, por acogerme como su estudiante durante todo el tiempo de mi formación profesional en esta prestigiosa academia, ha sido un soporte invaluable, en todos los éxitos cosechados.

A mis compañeros; Eric, Danny, Cristian, Brayan y Anita, que con su amistad invaluable han sido un pilar fundamental dentro y fuera del aula de clase.

DEDICATORIA

Dedico este logro en primer lugar a mis padres Lucia y Juan, por inculcarme desde niño que el estudio y el conocimiento es la mejor herencia que pueden dejarnos como padres; además de, que por su amor, poyo y esfuerzo se me ha permitido hoy cumplir otro sueño.

A todos mis amigos y amigas que conocí en el transcurso de mi carrera con los que aprendí y compartí innumerables experiencias en esta etapa, ayudándome a crecer de manera profesional y personal.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

RESUMEN

El presente trabajo de investigación responde al desarrollo de un sistema de reconocimiento de placas vehiculares basado en técnicas de visión por computador, cuyo objetivo es automatizar el ingreso y salida de vehículos de un parqueadero e introducir una opción tecnológica a través del internet de las cosas permitiendo agilizar un proceso que dependía del control de un operario; la automatización representa la optimización del tiempo, seguridad a los usuarios y manejo de información mediante la web. El estudio parte de un algoritmo de reconocimiento de objetos enfocado en los vehículos para obtener la información de la placa vehicular. El sistema capta la información existente en el exterior, reconoce el vehículo, procesa la información proveniente del algoritmo mediante un ordenador para obtener los datos de las placas, que se almacenarán en una base de datos alojada en la web. Se desarrolló una aplicación móvil en la plataforma Android Studio compatible con gran parte de los dispositivos móviles actuales, la aplicación permite el manejo y visualización de la información existente en la base que se divide en: registro de entrada y salida de los vehículos y el número de lugares disponibles de un parqueadero en tiempo real por medio de dispositivos inteligentes. Se ejecuta un plan de pruebas para evaluar al sistema en cada uno de sus módulos tomado en cuenta diferentes variables como: condiciones meteorológicas, posición y distancia de la cámara y la unidad de procesamiento de la información, para la validación del sistema de reconocimiento de placas vehiculares.

PALABRAS CLAVE: visión por computador, algoritmo de reconocimiento de objetos, vehículo, parqueadero, aplicación móvil.

ABSTRACT

This research work responds to the development of a vehicle license plate recognition system based on computer vision techniques, whose objective is to automate the entry and exit of vehicles from a parking lot and introduce a technological option through the internet of things allowing streamline a process that depended on the control of an operator; automation represents the optimization of time, user safety and information management through the web. The study is based on an object recognition algorithm focused on vehicles to obtain vehicle license plate information. The system captures the information existing outside, recognizes the vehicle, processes the information from the algorithm using a computer to obtain the data from the license plates that will be stored in a database located on the web. A mobile application was developed on the Android Studio platform compatible with a large part of current mobile devices, the application allows the management and visualization of the existing information in the base that is divided into: entry and exit registration of vehicles and the number of available places of a parking lot in real time through smart devices. A test plan is executed to evaluate the system in each of its modules, taking into account different variables such as: meteorological conditions, position and distance of the camera and the information processing unit, for the validation of the plate recognition system. vehicular.

KEY WORDS: computer vision, object recognition algorithm, vehicle, parking, mobile app.

CAPÍTULO I

1. DESCRIPCIÓN DEL PROBLEMA

1.1 Planteamiento del problema

La congestión vehicular en los núcleos urbanos a nivel mundial es una grave crisis que de a poco se ha convertido en una problemática para el desarrollo de las ciudades. El uso del suelo, la expansión urbana y la migración han causado serios impactos en la gestión de movilidad, evidenciándose un déficit en el servicio de transporte público, así como un aumento en el parque automotor, lo que ha derivado en una problemática en la gestión de movilidad al momento de circular por los anillos céntricos de las ciudades [1].

En la ciudad de Ibarra, a partir del último censo poblacional 2010, se ha observado un acelerado crecimiento demográfico, el cual se manifiesta con la creciente demanda de vehículos particulares, unidades de transporte público y taxis [2]. Esta situación ha puesto en alerta a la Unidad Municipal de Tránsito que es la institución encargada en la cuidad de Ibarra del control, señalización y organización de tránsito para que tome cartas en el asunto y de soluciones a los problemas que enfrenta por el acelerado crecimiento del parque automotor en la cuidad [3].

Por ende, este crecimiento del parque automotriz y la carencia de soluciones viables al problema de tráfico vehicular ha desencadenado que en las carreteras y calles principales en las ciudades generando grandes puntos de aglomeración y contaminación.

Actualmente, los sistemas destinados a gestionar el acceso de vehículos a parqueaderos carecen de un mecanismo de identificación automático de sus usuarios, y normalmente requieren la presencia de un operario para la seguridad del lugar.

Por ello es necesario implementar un sistema inteligente destinado a la gestión y automatización del acceso vehicular a través de una cámara como mecanismo de identificación de vehículos brindado confort y seguridad a los mismos, además se reducirá el tráfico vehicular en lugares próximos al parqueadero automatizado, aportando a la conservación y cuidado del ambiente con la disminución de emisiones de CO_2 al aire.

1.2 Objetivos

1.2.1 Objetivo General.

Implementar un sistema de gestión y automatización del acceso a un parqueadero.

1.2.2 Objetivos Específicos.

- Determinar las condiciones adecuadas para la adquisición de imágenes a utilizar en visión por computador.
- Adquirir imágenes para su procesamiento digital.
- Extraer las características de las placas de los vehículos útiles para el reconocimiento e identificación de datos.
- Crear una aplicación móvil para visualizar la disponibilidad del parqueadero.
- Ejecutar un plan de pruebas que permitan evaluar al sistema en cada uno de sus módulos.

1.3 Alcance

La propuesta del proyecto de trabajo de titulación abarca el desarrollo de un sistema que permita la detección y conteo vehicular basado en técnicas de visión por computador, el cual enviará los datos del registro vehicular a una base de datos para su posterior almacenamiento y monitoreo.

La aplicación móvil será orientada a 2 áreas de interés: el usuario del parqueadero podrá observar la disponibilidad de este, a través de un Smartphone, por otra parte, el operario o administrador tendrá acceso a datos de su interés para analizar la concurrencia de clientes.

1.4 Justificación

El desarrollo de un sistema de gestión y automatización del acceso a un parqueadero, basado en técnicas de visión por computador y reconocimiento de placas vehiculares es relevante en los siguientes aspectos:

En el ámbito de la movilidad, las personas necesitan trasladarse de forma rápida y segura en el centro de la ciudad para evitar el congestionamiento vehicular, una de las soluciones es tener acceso a una aplicación que muestre si existe o no, espacios disponibles en los distintos parqueaderos, siendo este un factor de alta importancia en la movilidad de las personas.

Según el VII censo nacional realizado en el año 2010, de los 14.483.499 habitantes que tiene el Ecuador, la población del cantón Ibarra alcanza el 181.175. El 10% de esta ciudad utilizan vehículos particulares para movilizarse de acuerdo con [2], convirtiéndolos en beneficiarios directos de los sistemas de parqueadero inteligente.

En el aspecto tecnológico, la creación de sistemas inteligentes (Visión por computador) podría generar nuevos temas de desarrollo útiles en diferentes aplicaciones.

En el aspecto de seguridad vehicular los parqueaderos son muy importantes para garantizar las condiciones de los vehículos que se encuentran en los sitios de parqueos, evitando futuros incidentes o accidentes.

CAPÍTULO II

2. MARCO TEÓRICO

2.1 Visión por computador

La visión por computador es una de las aplicaciones que se derivan de la inteligencia artificial como describe en [4], definida como el proceso de obtención, caracterización e interpretación de imágenes capturadas del mundo externo.

La visión por computador tiene como objetivo modelar matemáticamente los procesos de percepción visual de los seres humanos permitiendo simular las capacidades visuales mediante un ordenador, además permite la detección automática de la estructura y características principales de un mundo dinámico en 3 dimensiones a partir de una o varias imágenes bidimensionales del mundo [5].

2.1.1 Componentes principales de visión por computador.

A medida que la tecnología continua evolucionando y permitiendo mejorar las herramientas informáticas tanto en hardware como en software facilitando las necesidades que surgen habitualmente, de esta forma los dispositivos y herramientas que actúan y forman parte del campo de visión artificial han permitido la creación de sensores, placas computadoras, algoritmos, librerías informáticas entre otras, para mejorar el procesamiento de imágenes sin limitaciones que se presentan en el proceso por ejemplo la velocidad de captura de imagen, tiempo que tarda del procesamiento de la información y reconocimiento de caracteres [6].

Un sistema de visión por computador está conformado básicamente por subsistemas que cumple dos funciones [6]:

- Captar la información de la escena real mediante la obtención de una imagen.
- Procesar las imágenes para extraer la información que contiene.

Los dispositivos importantes para cumplir con estas funciones son: fuentes de iluminación, cámaras, software y hardware adecuado para analizar y procesar las imágenes como se muestra en la Ilustración 1.

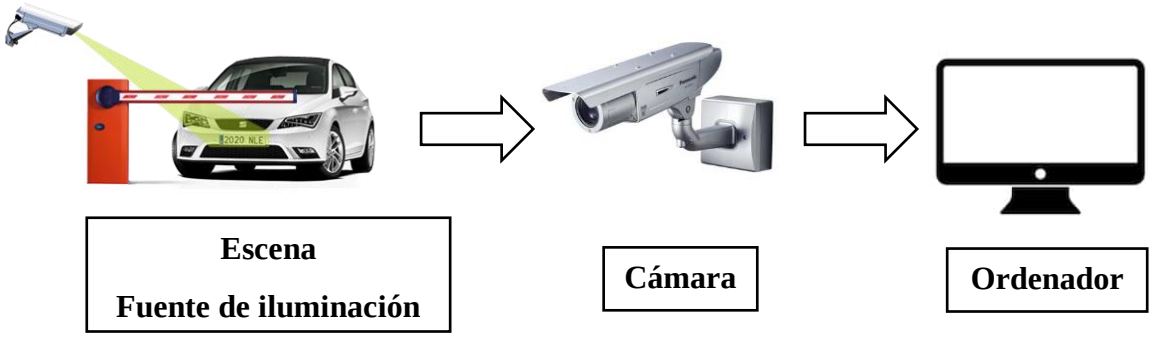


Ilustración 1: Componentes de visión por computador

Fuente: [6]

2.1.1.1 Escena (Fuente de Iluminación).

De acuerdo con el entorno de aplicación del sistema se debe implementar una fuente de iluminación adecuada, una mala iluminación puede generar anormalidades en la imagen obtenida, difíciles de eliminar con la aplicación de filtros en el caso de sombras, el bajo o alto nivel de contraste, contornos no muy definidos entre otras características [7].

Las fuentes de iluminación se muestran en la Ilustración 2:

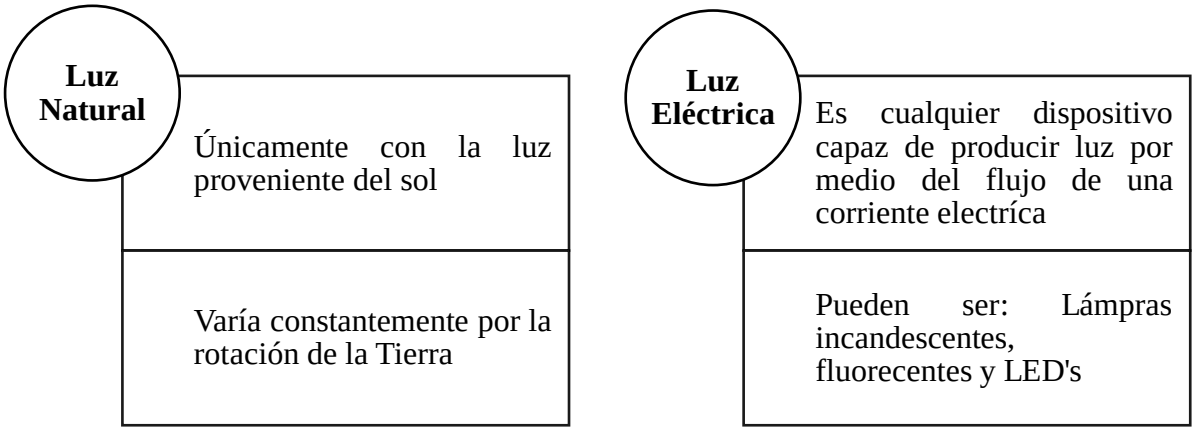


Ilustración 2: Fuentes de Iluminación

Fuente: [7]

2.1.1.2 Cámara.

La cámara en el sistema de visión por computador cumple la función de ingreso de la información y se denomina “Ojos del sistema”, debido a que el dispositivo capta la escena del

medio en el cual se está aplicando. Las cámaras óptimas para el cumplimento de los objetivos del sistema cuentan con características mejoradas de una cámara normal, las mismas que ofrecen [6] :

- Resolución HD
- Visión infrarroja (Para el enfoque nocturno)
- Enfoque automático (Para detección de objetos)
- Protección de exteriores (Garantiza el funcionamiento del sistema en diferentes condiciones externas)

2.1.1.3 Ordenador

El ordenador con un software y hardware debidamente complementados con herramientas para el procesamiento de imágenes es el encargado de procesar la información recopilada por las cámaras, extraer e interpretar la misma mediante algoritmos de reconocimiento de objetos, entre las principales tareas que debe realizar el ordenador en un sistema de visión por computador se citan las siguientes de acuerdo con [6] :

- Detección y pre procesado de la imagen
- Segmentación de caracteres
- Reconocimiento de caracteres

2.1.2 Procesamiento digital de imágenes.

El procesamiento digital tiene como objetivo mejorar la calidad de la imagen para hacer más simple el procesamiento e interpretación en la detección de objetos de interés dentro de la imagen obtenida apreciado en [8], las mejoras son las siguientes:

- Remover defectos
- Remover problemas por movimiento o desenfoque
- Mejorar ciertas propiedades como color, contraste, estructura, etc.

- Agregar “colores falsos” a imágenes monocromáticas

En la ilustración 3 se muestra un ejemplo de procesamiento de imágenes. El objetivo es mejorar la imagen obtenida del entorno, la cual es obscura y se obtiene como salida la misma imagen con mejor calidad aplicando un mejoramiento en el contraste.

Ilustración 3: Procesamiento digital de imágenes. (a) Imagen obscura debido a que su rango de grises es reducido, (b) Ecualización del rango de grises.

Fuente: [8]

2.1.2.1 Etapas para el procesamiento digital de imágenes.

En la Ilustración 4 se muestra las 5 etapas necesarias para el procesamiento digital de imágenes:

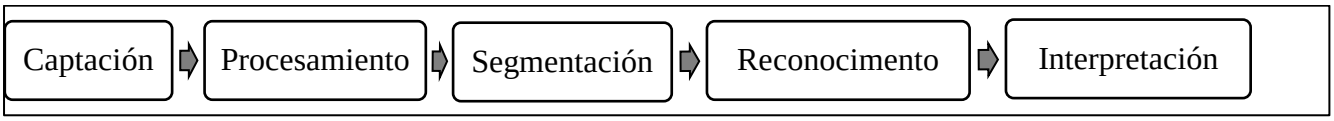


Ilustración 4: Etapas para el procesamiento digital de imágenes

Fuente: [9]

La etapa de captación es el proceso mediante el cual se obtiene una imagen digital, en la etapa de procesamiento se incluyen técnicas como reducción de ruidos, aplicación de filtros y realce de detalles, la segmentación es el proceso que divide una imagen en objetos de interés, llamado también como proceso de etiquetado [9].

2.2 Algoritmos de detección de objetos

El proceso de la detección de objetos se hace a través de un sistema inteligente capaz de extraer información de un entorno y luego implementar los algoritmos adecuados para su procesamiento y correcta interpretación, independientemente de factores como la iluminación, color, posición entre otros, y así reconocer todas las posibles variaciones que el objeto o imagen pueda representar [10].

2.2.1 Sift.

Este algoritmo fue desarrollado para el reconocimiento de objetos. SIFT detecta un punto característico de un objeto el mismo que puede ser reconocido con la invariancia de la iluminación, escala rotación y transformaciones afines. De acuerdo con [10] las ventajas que ofrece este método son:

- No se necesita transformar a coordenadas polares, para una precisión alta en la segmentación de imágenes.
- Debido a su invarianza a la iluminación, escala y rotación se espera que sea confiable cuando se use en la adquisición de imágenes con las siguientes condiciones:
    - Conveniencia del usuario.
    - Aplicabilidad a los ambientes no cooperativos.

2.2.2 Momentos de HU

Los momentos de HU y los relacionados con la invariancia han sido extensivamente analizados en el reconocimiento de patrones de imágenes en una gran variedad de aplicaciones.

Los momentos invariantes fueron introducidos por primero vez por Hu. HU derivó seis momentos ortogonales y un momentos asimétrico-basados todos estos en el algebra, los mismos que no son dependientes de la rotación, tamaño y posición, pero si son dependientes de una proyección paralela. Los momentos invariantes han sido probados para ser medidos

adecuadamente para la coincidencia de imágenes a pesar de su rotación, escalamiento y rotación. Los momentos invariantes han sido extensivamente aplicados al reconocimiento de imágenes, al registro de imágenes y a la reconstrucción de estas [11].

2.2.3 Redes Neuronales

Las redes neuronales son estructuras pensadas para simular las estructuras y el comportamiento del sistema nervioso, por esta razón una neurona artificial posee entradas, salidas y un estado. Además de estar conectadas a otras neuronas siendo las salidas de estas las entradas de las demás [12].

Cada neurona realiza una operación matemática, y además de agrupan en capas lo que constituye un esquema de red neuronal mostrado en la Ilustración 5. De acuerdo con [12] un sistema neuronal o conexionista se compone de lo siguiente:

- Un conjunto de procesadores elementales
- Un patrón de conectividad o arquitectura
- Una dinámica de activaciones
- Una regla o dinámica de aprendizaje
- En entorno donde opera

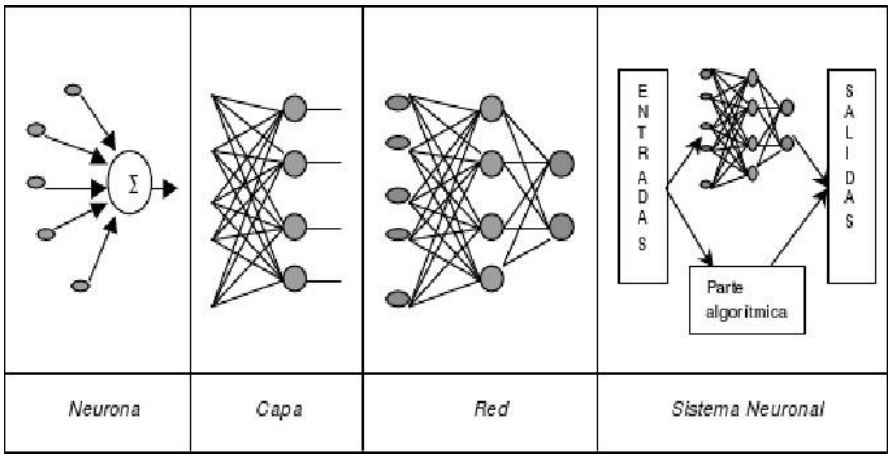


Ilustración 5: Esquema de una red neuronal

Fuente: [12]

2.2.3.1 Características de una Red Neuronal.

De acuerdo con [12] una red neuronal consta de las siguientes características:

- *Procesamiento paralelo*: Si queremos procesar una imagen, un ordenador tarda algunos minutos en recorrer toda la imagen. Por otro lado, el cerebro necesita unos pocos segundos en procesar millones de pixeles, ya que las neuronas que interviene en el sistema trabajan en paralelo.
- *Memoria distribuida*: en un ordenador la información se almacena en posiciones continuas bien definidas, en una red neuronal se almacena en las sinapsis de la red. Si una se daña perdemos la información.
- *Aprendizaje adaptativo*: la capacidad de aprender tareas en base al entrenamiento elimina la necesidad de elaborar modelos a priori o indicar funciones de distribución de probabilidad.
- *Auto organización*: Las redes neuronales utilizan su capacidad de aprendizaje adaptativo para organizar la información que reciben durante el aprendizaje/operación [12].

2.2.4 Lógica difusa.

La lógica difusa nació en 1965 siendo Lofti Zadeh su creador. A este científico se le atribuyen muchas contribuciones en el desarrollo de la transformada Z para sistemas lineales discretos. La lógica difusa según [13], difiere de la logia clásica debido a que esta última maneja de forma imprecisa la clasificación de valores. Esta además permite una clasificación más amplia donde la pertenencia a un grupo es gradual.

La lógica difusa como tal se refiere a toda la serie de teorías para modelación, control, algoritmos computacionales e inteligencia artificial que se basan en un razonamiento de

incertidumbre o imprecisión, donde la palabra clave se basa en límites que no son cortantes [13].

2.3 Librerías para el procesamiento de imágenes

Actualmente existe gran variedad de herramientas para visión por computador, a continuación, se enlista las más utilizadas y se procede a realizar su concerniente análisis.

2.3.1 Open CV (Open Source Computer Visión).

Open CV es una biblioteca de software de visión abierta y software de automático aprendizaje. El objetivo de su construcción e implementación es proporcionar una infraestructura común para aplicaciones de visión por computadora y para acelerar el uso de la percepción de la maquina en los productos comerciales. Al ser un producto con licencia BSD, OpenCV facilita a las personas y empresas a utilizar y modificar el código [14].

Open Source Computer Visión es una biblioteca que cuenta con más de 2.500 algoritmos optimizado, incluyen un conjunto completo de algoritmos de visión artificial y de aprendizaje automático tanto clásico como avanzado [14].

2.3.2 JavaCV.

Es una interfaz para usar OpenCV bajo Java. De hecho, comparte la misma documentación que OpenCV. Ha quedado relegada a un segundo plano con la salida de la última versión de OpenCV, la 2.4.4, que soporta Java [15].

2.3.3 PyTorch.

De acuerdo con [16] PyTorch es una biblioteca para programas de Python que facilita la construcción de proyectos de aprendizaje profundo. Enfatiza la flexibilidad y permite que los modelos de aprendizaje profundo se expresen en Python idiomático. Esta accesibilidad y facilidad de uso encontraron a los primeros usuarios en la comunidad de investigación, y en los

años transcurridos desde el lanzamiento de la biblioteca, se ha convertido en una de las herramientas de aprendizaje profundo más destacadas para una amplia gama de aplicaciones.

2.3.4 LTI-Lib.

El LTI-Lib es una biblioteca orientada a objetos con algoritmos y estructuras de datos utilizados frecuentemente en el procesamiento de imágenes y visión por computador. Ha sido desarrollado en la Cátedra de Informática Técnica (Lehrstuhl fuer Technische Informatik) LTI en la Universidad Tecnológica de Aquisgrán, como parte de muchos proyectos de investigación en visión artificial relacionados con robótica, reconocimiento de objetos y reconocimiento de gestos [17].

2.3.5 Comparativa de librerías de procesamiento de imágenes

Se realiza un análisis de ventajas y desventajas de cada librería de procesamiento de imágenes planteada para obtener conclusiones de cuál es la adecuada dependiendo del medio de aplicación como se puede apreciar en Tabla 1.

Tabla 1.

Ventajas y desventajas de librerías para el procesamiento de imágenes.

Nombre	Ventajas	Desventajas
OpenCV	• Dirigido hacia aplicaciones de visión por ordenador en tiempo real. • Esta bajo la Licencia de BSD. • Cuenta con más de 2500 algoritmos optimizados para visión artificial y aprendizaje automático tanto clásico como avanzado. • Tiene interfaces C++, Python, Java y MATLAB. • Es compatible con Windows, Linux, Android y Mac OS. • Está escrito nativamente en C ++.	• Posee compresiones y descompresiones complejas y costosas. • No incluye transparencias ni animaciones. • La información perdida no se recupera. Si trabajamos con un JPEG guardando en disco tras cada operación, la imagen se va degradando
PyTorch	• Contiene algoritmos de procesamiento de imágenes y extracción de características. • Esta distribuido bajo la licencia BSD. • Es modular.	• Se limita a trabajar únicamente con el lenguaje C ++. • Sus algoritmos son muy básicos.
LTI-Lib	• Está bajo los términos de la Licencia Pública GNU. • Orientada a objetos. • Tiene módulos para despeñar diferentes funciones. • Consta de algoritmos y estructuras de datos utilizados en el procesamiento de imágenes y la visión por computadora.	• Se limita a trabajar únicamente con el lenguaje C ++. • Funciona solo en las plataformas • GCC en Linux y Visual C ++ en Windows NT.

Fuente: [18]

2.4 Sistemas de Parqueo Inteligente

De acuerdo con [19] el objetivo principal de los sistemas de parqueo inteligente es ayudar a los conductores a encontrar plazas de aparcamiento disponibles de manera rápida y eficiente a través de tecnologías de la información y comunicaciones. La idea se presenta diez años atrás aproximadamente, y en los últimos cinco años se ve observa cada vez más sistemas de parqueo inteligente en las principales ciudades, con la capacidad de conectarse a internet.

Con el fin de obtener el estado de las plazas de parqueo, se instalan sensores fijos o móviles en calles, parqueaderos cubiertos, etc., para detectar los eventos vehiculares. Se puede utilizar una amplia variedad de sensores, entre ellos: infrarrojos, magnetómetros, tubo neumático de carretera, sensores magnéticos, sensores ultrasónicos, sensores de peso en movimiento, RFID y procesadores de imagen de video [20].

2.4.1 Sistemas de información de guía parqueo (IGP).

Según [21], uno de los primeros ejemplos de parqueadero inteligente, son los sistemas de información de guía de parqueo (IGP). Sistemas de IGP intentaron optimizar la búsqueda de aparcamiento mediante el control de estacionamiento disponible de forma dinámica, pueden incluir ya sea en toda la ciudad o simplemente la instalación de aparcamiento.

En IGP los sensores de detección de vehículos se instalan habitualmente en las entradas, salidas y/o espacio de estacionamiento individual para detectar ocupación de los vehículos. Las lecciones aprendidas mediante la evaluación y el modelado de estos sistemas muestran que, con el fin de ser eficaz, los mensajes deben mostrar información adecuada que se adapte a las necesidades del conductor [21].

2.4.2 Sistemas basados en tránsito (TB).

Basándose en [22], las lecciones aprendidas de los sistemas de IGP, los sistemas basados en tránsito tienen como objetivo aumentar el uso del transporte y de los ingresos y

reducir la marcha del vehículo, el uso de combustible y la contaminación del aire. Muchos sistemas basados en tránsito (TB) implementados en países como Francia, Alemania, Irlanda, Japón, Suiza, el Reino Unido y Estados Unidos, son muy parecidos a los de la IGP.

2.4.3 Sistemas de pago inteligente.

Los dos sistemas IGP y sistemas TB, proporcionan a sus usuarios información valiosa acerca de la disponibilidad de estacionamiento y, en el caso de la TB, su viaje. De acuerdo con [23], un tipo de sistema inteligente de estacionamiento que no se centra en la disponibilidad, pero que apuntan a la optimización de los procesos de estacionamiento, es un sistema de pago inteligente. En estos sistemas, el proceso de pago es completamente automatizado para evitar lo retrasos que se producen durante los métodos de pago convencionales [23].

2.4.4 Sistemas E-parking.

Un cuarto tipo de sistema de aparcamiento inteligente, se llama E-parking. En un E-parking los usuarios pueden consultar la disponibilidad y/o reservar una plaza de estacionamiento. El sistema se puede acceder a través de numerosos métodos, tales como SMS, Smartphone, entre otros dispositivos conectados a internet. Esto permite a los usuarios ver información actualizada antes y durante su recorrido. Este sistema también está equipado con un procedimiento de reserva, así es capaz de evitar que dos conductores llegan al mismo momento en el mismo lugar [23].

2.4.5 Sistemas Parqueo Automatizado

Según [23], la forma final de un parqueo inteligente es un Sistema de Parqueo Automatizado. En este sistema, cada paso del proceso de aparcamiento está automatizado. El conductor se detiene en una plataforma, deja su vehículo bloqueado y las máquinas se encarguen del resto. Este tipo de aparcamiento obviamente permite almacenar muchos vehículos en un espacio relativamente pequeño debido a que los conductores no necesitan el

espacio adicional para conducir, para entrar y salir de sus vehículos. Además, se evita que los conductores se pasean en medio del espacio de estacionamiento, aumentando la seguridad.

2.5 Software Libre

El software libre ofrece al usuario cuatro libertades: libertad de uso, de estudio y modificación de distribución y redistribución de las versiones modificadas [24].

Adicionalmente existen licencias para software libre que lo garantizan y que le dan una cobertura legal, como por ejemplo la GPL, Licencia Pública General [25].

2.6 Hardware Libre

De acuerdo con la asociación de Hardware de fuentes abiertas (Open Source Hardware Association / Hardware Libre) es aquel hardware cuya disponibilidad de diseño es de acceso público con el objetivo de que cualquier persona se sienta en la libertad de analizar, modificar, distribuir o crear nuevos diseños a partir del diseño original [26].

2.7 Placas vehiculares del ecuador

2.7.1 Reglamento de las placas vehiculares.

Cada vehículo y sus similares, para hacer uso de las vías del país, debe de contar con títulos habilitantes como la licencia y la matrícula correspondiente, además de portar placas de identificación vehicular, que son regulados y autorizadas única y exclusivamente por la Agencia Nacional de Regulación y Control del Transporte Terrestre, Tránsito y Seguridad Vial, de acuerdo con lo establecido en la "LEY DE TRANSPORTE TERRESTRE TRANSITO Y SEGURIDAD VIAL" [27].

Las placas de identificación vehicular son otorgadas por las entidades oficiales: Agencia Nacional de Regulación y Control del Transporte Terrestre, Tránsito y Seguridad Vial, por las Unidades Administrativas Regionales o Provinciales, o por los gobiernos autónomos

descentralizados, las cuales son ubicadas en la parte anterior y posterior del vehículo, en los lugares destinados especialmente por el fabricante del automotor.

2.7.2 Reglamento para la elaboración, entrega y control de placas de identificación vehicular.

Todos los vehículos deberán portar dos placas de identificación vehicular (Art 1).

Las placas de identificación vehicular serán emitidas exclusivamente por la ANT, y entregadas por sus Unidades Administrativas Regionales o Provinciales por los GAD's que hayan asumido la competencia, o por el organismo de tránsito debidamente autorizado, conforme la Ley de acuerdo con las siguientes características y dimensiones, como se muestra en la Ilustración 6:

a) Sus dimensiones serán de 404 mm de largo por 154 mm de alto
b) En la esquina superior izquierda llevarán impreso a color el logotipo de la Agencia Nacional de Tránsito
c) Las letras y números impresos medirán 38 mm de ancho por 75 mm de alto
d) En la parte superior, en un 16 campo de 146 mm de ancho por 27 mm de alto, estará impresa la palabra ECUADOR en letras mayúsculas.
e) La identificación de la placa se realizará mediante una serie alfanumérica, que tendrá un guion de separación entre los números y las letras de un ancho de 20mm.

Las características de seguridad serán definidas exclusivamente por la Dirección Ejecutiva de la ANT [27].

Ilustración 6: Placa vehicular Ecuador

Fuente: [27]

La primera letra identificará la provincia donde se registra la placa vehicular como se indica en la Tabla 2.

Tabla 2

Provincias del Ecuador

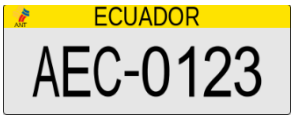
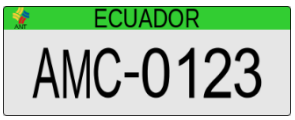
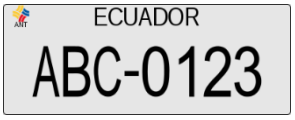
N°	Provincia	Letra
1	Azuay	A
2	Bolívar	B
3	Cañar	U
4	Carchi	C
5	Cotopaxi	X
6	Chimborazo	H
7	El Oro	O
8	Esmeraldas	E
9	Galápagos	W
10	Guayas	G
11	Imbabura	I
12	Loja	L
13	Los Ríos	R
14	Manabí	M
15	Morona Santiago	V
16	Napo	N
17	Pastaza	S
18	Pichincha	P
19	Orellana	Q
20	Sucumbíos	K
21	Tungurahua	T
22	Zamora Chinchipe	Z
23	Santa Elena	Y
24	Sto. Domingo de los Tsáchilas	J

Fuente: [27]

La segunda letra de la placa y franja de color en la parte superior de las matrículas se establecen de acuerdo con el servicio que presta el vehículo mostrado en la Tabla 3.

Tabla 3

Placas especificación por Servicio

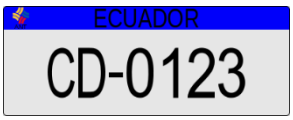
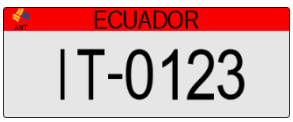
Tipo	Segunda letra	Color	Ejemplo	Imagen
Vehículos comerciales (Taxi o autobus)	A,U,Z	Naranja	AAB-0123	
Vehículos gubernamentales	E	Oro	PEB-0123	
Vehículos de uso oficial	X	Oro	GXA-0123	
Vehículos de los Gobiernos Autónomos Descentralizados (Regiones, Provincias, Cantones, Parroquias)	M	Verde limón	AMC-0123	
Vehículo particular (privado)	Cualquiera menos las anteriores	Blanco-plateado	ABC-0123	

Fuente: [27]

Las placas vehiculares especiales son destinadas para vehículos de organismos internacionales o vehículos que estén por un tiempo temporal en el país, y se distinguen por usar solo dos letras, ordenándose como se muestra en la Tabla 4.

Tabla 4

Placas especificación servicios especiales.

Tipo	Letras	Color	Ejemplo	Imagen
Vehículos de servicio diplomático	CC (Cuerpo Consular) CD (Cuerpo Diplomático) OI (Organismo Internacional) AT (Asistencia Técnica)	Azul	CC-0123	
Vehículos de internación temporal	IT	Rojo	IT-0123	

Fuente: [27].

2.8 Tecnología móvil

Según [28], la tecnología móvil está directamente ligada a la telefonía móvil y es a la que nos vamos a referir para la presente investigación.

"Las comunicaciones móviles sin duda alguna han experimentado un enorme crecimiento desarrollándose diversas tecnologías y sistemas para dar servicios de comunicación inalámbrica. En el Ecuador el servicio móvil celular inicia a finales de 1993 con la entrada en el mercado de CONECEL S.A., manteniéndose el dominio de las empresas Bellsouth y Porta hasta el año 2003, cuando entro en operación una tercera operadora denominada Alegro" [28].

Todas estas aplicaciones se ejecutan dentro de un ecosistema, existiendo varios factores que lo afectan como se muestran el Ilustración 7:

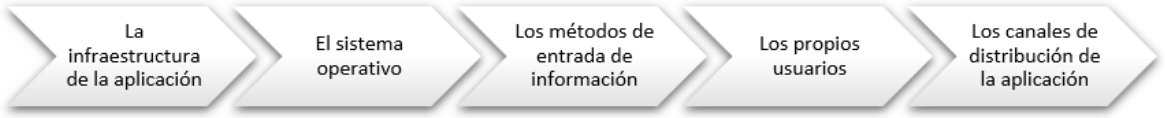


Ilustración 7: Ecosistema de una base de datos

Fuente: [28]

CAPÍTULO III

3. METODOLOGÍA

3.1 Diseño conceptual

Esta fase de gran importancia es el punto de partida del proyecto debido a que se especifican las características y funciones del sistema de gestión y automatización del acceso a un parqueadero. Se describen los atributos que se esperan del sistema en función a los requerimientos del proceso que debe ejecutar para conseguir los datos deseados por la administración del parqueadero.

3.2 Despliegue de la Función de la Calidad (QFD)

El Despliegue de la Función de Calidad (QFD) es uno de los métodos que frecuentemente se emplea en el diseño conceptual para captar las demandas reales del mercado y plasmarlas como objetivos de diseño, mismos que deben permanecer presentes en todas las fases posteriores para obtener un producto final (o en este caso, un sistema informático) que responda realmente a las expectativas del cliente [29].

3.2.1 Desarrollo del QFD.

Retomando el argumento del párrafo anterior, es correcto asumir que la primera tarea del QFD es detectar los requerimientos del cliente y enlistarlas como procede a continuación.

3.2.1.1 Requerimientos del cliente

De acuerdo con los requerimientos del proyecto de titulación definidos el sistema de gestión y automatización del acceso a un parqueadero está orientada a la obtención de información de entrada y salida de los usuarios, registro de placa vehicular en una base de datos, manipulación de la información; en virtud de aquello, el objetivo es entregar dicha información al usuario final de una manera automática.

- Facilidad de detección de vehículos.
- Conteo y reporte de datos de vehículos.
- Almacenamiento de placas en DataBase.
- Almacenamiento de datos en disco externo.
- Gestión de usuario en aplicación móvil.
- Fácil de utilizar.
- Evitar el uso excesivo de cable para la instalación.

3.2.1.2 Lista de parámetros técnicos

Los parámetros técnicos son unidades físicas medibles elaboradas por el ingeniero, a partir de las demandas del cliente.

- Algoritmo de detección de vehículos.
- Capacidad de generar datos estadísticos de flujo de vehículos.
- Visión por computador y Base de datos.
- Capacidad de almacenamiento
- Manejo de aplicación móvil.
- Interfaz amigable con el usuario.
- Sistema inalámbrico para transmisión de datos.

3.2.1.3 Matriz QFD resultante

La casa de la calidad fue elaborada mediante software para facilitar los cálculos necesarios de esta herramienta, se puede observar en el Anexo 1.

3.2.1.4 Resultados de la casa de la calidad

Los requerimientos funcionales para considerar son los siguientes:

- Algoritmos de visión por computador: El algoritmo será de alta precisión con la capacidad de detectar al vehículo de forma automática.
- Capacidad de conteo y generación de datos estadísticos del flujo vehicular: Almacenar datos de clientes para su manipulación, por ejemplo, hora de ingreso, hora de llegada y disponibilidad de lugares de parqueo.
- Procesamiento mediante una placa computadora: el sistema realizará el procesamiento de imágenes mediante una placa computadora con una cámara adecuada al medio de trabajo del parqueadero.
- Plataforma de desarrollo de aplicación móvil: El desarrollo de la aplicación permitirá el manejo y visualización de la información mediante un dispositivo móvil compatible con el software de programación.
- Facilidad para utilizar el sistema: la funcionalidad del sistema no debe presentar complejidad para el usuario.
- Transmisión inalámbrica de datos: La transmisión de datos de forma inalámbrica evitará la presencia de cableado excesivo debido a las dimensiones de los parqueaderos.

3.3 Estructuración sistemática de funciones

En ingeniería mecatrónica, es común desglosar sistemáticamente las funciones de un producto o sistema en proceso de diseño para entender de mejor manera el objetivo de este.

La función principal del sistema de gestión y automatización del acceso a un parqueadero es detectar el ingreso y salida de los vehículos, guardar la información obtenida de manera automática y presentar al usuario final la disponibilidad de lugares en el parqueadero mediante una aplicación, en la Ilustración 8, se evidencia las entradas y salidas desde una perspectiva global del sistema.

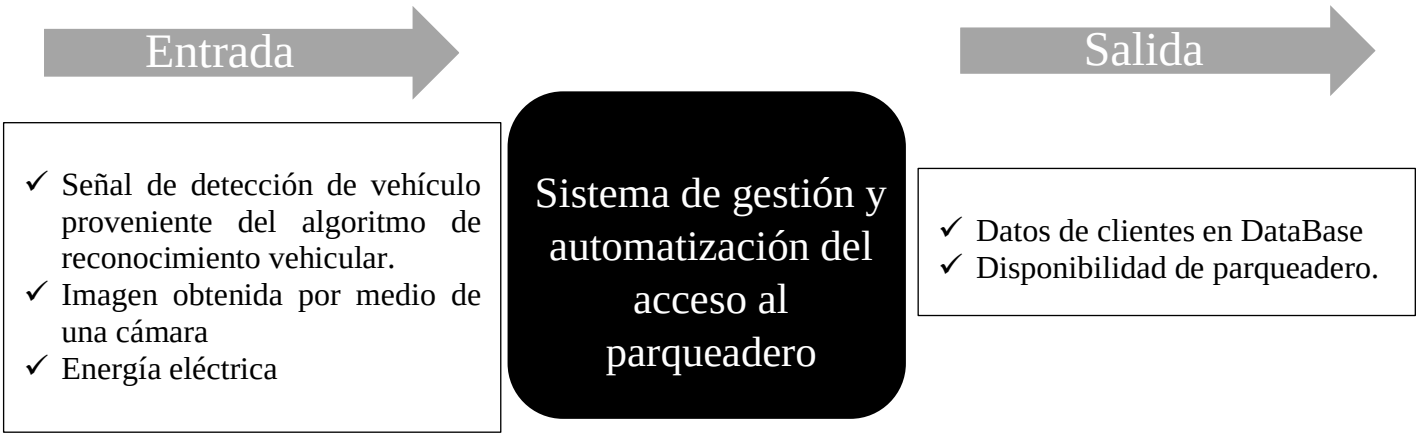


Ilustración 8: Caja negra del sistema de gestión y automatización del acceso a un parqueadero.

El proceso completo que ejecuta el sistema de gestión y automatización del acceso a un parqueadero a partir de su ciclo inicial y su ciclo final desglosado en subfunciones encerradas en un módulo.

Este sistema de organización mostrado en la Ilustración 9, permite analizar cada etapa independiente, de esta manera, se genera varias alternativas de funcionamiento y desarrollo para cada subfunción, en efecto, se conseguirá varias soluciones o caminos para el desarrollo del sistema.

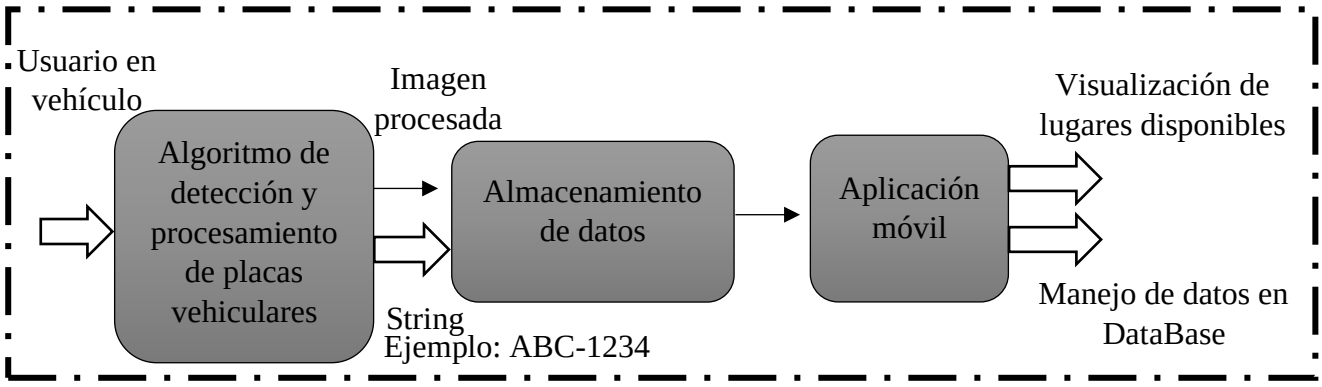


Ilustración 9: Estructura funcional del sistema de gestión y automatización de acceso a un parqueadero

Desglose de funciones y subfunciones del sistema de gestión y automatización de acceso a un parqueadero.

El bloque 1 corresponde al algoritmo de detección y procesamiento de placas vehiculares (software libre), el cual es el encargado de captar el objeto de interés desde el exterior por medio de una cámara para su posterior procesamiento mediante un ordenador.

El bloque 2 corresponde al procesamiento de imágenes captadas por el algoritmo de detección y procesamiento de placas vehiculares, para la obtención del String de la placa vehicular para posteriormente almacenarlo en una base de datos.

El bloque 3 es el encargado de almacenar los datos de los vehículos en una base de datos alojada en la web conocida como 000WebHost, según [30] es un Hosting Web gratuito con 1.5 GB de almacenamiento, y con la capacidad de crear 2 bases de datos MySQL, para su manejo y visualización de la información.

El bloque 4 es el encargado de mostrar la información en un dispositivo móvil mediante una aplicación con dos funciones principales:

1. Mostrar el historial de ingreso y salida de los vehículos a un parqueadero.
2. Mostrar los espacios disponibles en el parqueadero.

3.3.1 Alternativas para cada subfunción.

El sistema de sistema de gestión y automatización del acceso a un parqueadero comprende cuatro subfunciones para ejecutar el proceso de reconocimiento de placas vehiculares, a continuación, se proponen alternativas de solución para cada uno:

3.3.1.1 Sistemas de Procesamiento.

El sistema de procesamiento de la información es una placa computadora que posee un software y hardware debidamente complementados con herramientas para el procesamiento de

imágenes; las principales tareas que debe realizar la placa computadora dentro de un sistema de visión artificial se citan las siguientes:

- Pre procesado de la imagen
- Segmentación
- Descripción y reconocimiento.

La opción más favorable para el correcto procesamiento y manejo de la información es utilizar software de desarrollo Open Source, el mismo que puede ser instalado en las distintas placas computadoras si así lo admite, como se nuestra en la Tabla 5

Tabla 5

Unidades de procesamiento para parqueadero inteligente

Placa computadora	Software	Alimentación
Laptop (Recursos tecnológicos adecuados)	Windows Ubuntu Linux OS X	120 VAC
RaspBerry Pi	**PiDora** Arch Linux Ubuntu MATE Android OpenSure Raspbian	12 VDC
Arduino Yún	**Arduino Linux**	12 VDC

3.3.1.2 Sistema de detección de vehículos.

La función principal del sistema de detección de vehículos consiste en obtener una imagen atreves de una cámara o video con el objetivo de captar una imagen adecuada para su posterior procesamiento digital con el fin de mejorar la calidad de esta haciendo más simple el manejo e interpretación de objetos de interés.

En la Tabla 6 se detallan las características de las cámaras de video adecuadas para el procesamiento de imágenes disponibles en el mercado.

Tabla 6

Características de cámaras de video

Cámaras	Características principales	Alimentación
Cámara IP	IP privada Resolución desde 2 megapíxeles, hasta HD Receptor de corriente (power socket) Conector LAN: Conexión WIFI	120 VAC
Cámara RaspBerry Pi	Lente de distancia focal fija Resolución nativa de 8MP, Resolución fotos de 3280x2464 Resolución vídeo 1080p30, 720p60 y 640x480p90	Raspberry pi
Cámara Web	Resolución por lo general baja, aproximadamente 640 X 480 píxeles Su diseño es muy específico para Aplicaciones de entretenimiento y en algunos casos como cámara de vigilancia.	Computador
Cámara de seguridad	Sensor CMOS Resolución Máxima: 720P Día / Noche real (Filtro IR Cut). Distancia IR: 20mts. Protección: IP66 Alimentación: 12 VCD / 4W Aplicación: Interior / Exterior IP66	12 VDC

3.3.1.3 Almacenamiento de datos.

Para el almacenamiento de datos existe múltiples opciones de almacenamiento de la información digital con el objetivo de:

- Conservar y preservar los soportes de información.
- Conservar los dispositivos de procesamiento y lectura de esos soportes.
- Migrar de unos soportes de información a otros.
- Describir esa información para que se pueda recuperar de forma precisa.

Los soportes de almacenamiento de información digital posibles a utilizar se pueden destacar en la siguiente Tabla 7.

Tabla 7

Soportes de almacenamiento de información digital.

Soportes de almacenamiento	Características principales
CD y DVD	Vida estimada de entre 10 y 35 años Capacidad limitada claramente insuficiente Degradación y su vulnerabilidad a situaciones de humedad, altas temperaturas
Unidades de discos duros (Hard Drive Disk o HDD)	Discos duros mecánicos Mayor capacidad de almacenamiento Necesidad de mantenimiento, la degradación (vida media de 5 años)
Unidades de estado sólido (Solid State Drive o SSD)	**Guardan la información en microchips con memorias flash interconectadas** **Más rápidos en el procesamiento, se calientan menos, son más pequeños**
Almacenamiento basado en tecnologías cloud computing:	**La información es alojada en servidores externos** **Conlleva gestión de copias de seguridad y redundancia de la información** **La información es recuperable**

3.3.1.4 Aplicación móvil

La aplicación móvil para el sistema de gestión y automatización del acceso a un parqueadero comprende:

- Difusión de la información
- Canal de comunicación con los clientes

La plataforma en la que se desarrollará la aplicación móvil puede ser las siguientes:

- Kivy
- Android Studio
- App Inventor

3.4 Alternativas de solución

Las alternativas de solución guardan estrecha relación con los resultados de la casa de la calidad, estas se generan después de emplear la técnica de la matriz morfológica como se observa en la Tabla 8.

3.4.1 Solución Amarilla.

Este diseño se compone de un ordenador de placa única “Raspberry Pi” como sistema de procesamiento de la información, la misma que recibe los datos emitidos por su módulo de cámara infrarroja Raspberry Pi, posteriormente los datos serán almacenados en internet con acceso remoto desde cualquier parte del mundo, con conectividad a una aplicación móvil desarrollada en Android Studio para el manejo de la información.

3.4.2 Solución Azul.

El diseño contiene una laptop como sistema de procesamiento de la información, que recibe los datos emitidos por una cámara Web que posteriormente serán almacenados en la nube, y podrán ser visualizados mediante una aplicación móvil desarrollada en la plataforma de programación Android Studio para el manejo de la información.

3.4.3 Solución Roja.

El diseño contiene una laptop como sistema de procesamiento de la información, que recibe los datos emitidos por una Cámara Wifi, que posteriormente serán almacenados en un disco duro, y podrán ser visualizados mediante una aplicación móvil desarrollada en la plataforma de programación Kivy para el manejo remoto de la información.

3.4.4 Solución Naranja.

El diseño se compone de un ordenador de placa única “Arduino YUN” como sistema de procesamiento de la información, la misma que recibe los datos emitidos por una cámara de seguridad convencional, posteriormente los datos serán almacenados en un disco de estado sólido, con conectividad a una aplicación móvil desarrollada en MIT APP Inventor, para el manejo de la información.

Tabla 8

Matriz morfológica

SISTEMAS	ALTERNATIVAS
Sistemas de Procesamiento	 Laptop  Raspberry Pi  Arduino Yun
Sistema de detección de vehículos	 Cámara Wifi  Cámara Web  Cámara Raspberry Pi  Cámara de seguridad
Almacenamiento de datos	 CD/DVD  Disco duro  Solid State Drive o SSD  Cloud
Aplicación móvil	kivy Android Studio MIT APP INVENTOR

3.5 Selección de solución

Se va a emplear el "Método ordinal de criterios ponderados" Recomendado por [31] para decidir entre diversas soluciones, el cual permite obtener globales suficientemente significativos sin la necesidad de evaluar los parámetros de cada propiedad y sin tener que estimar numéricamente el peso de cada criterio. Ver Tabla 9.

Tabla 9

Evaluación del peso específico de cada criterio

Criterios	Plataforma de desarrollo de aplicación móvil	Respuesta del Sistema	Transmisión de Datos	Algoritmo de detección de objetos	Capacidad de procesamiento	∑ + 1	Ponderación
Plataforma de desarrollo de aplicación móvil		0	0	0	0	1	0.0666
Respuesta del Sistema	1		0.5	0	0	2.5	0.1667
Transmisión de Datos	1	0.5		0.5	0	3	0.2
Algoritmo de detección de objetos	1	1	0.5		0	3.5	0.2333
Capacidad de procesamiento	1	1	1	1		5	0.333
					Suma	15	1

Capacidad de procesamiento > Algoritmo de detección de objetos > Transmisión de Datos=Respuesta del Sistema > Plataforma de desarrollo de aplicación móvil

Tabla 10

Evaluación del peso específico del criterio plataforma de desarrollo de aplicación móvil.

Capacidad de procesamiento	Solución amarilla	Solución roja	Solución azul	Solución naranja	∑+1	Ponderación
Solución amarilla		1	1	0.5	3.5	0.35
Solución roja	0		1	0.5	2.5	0.25
Solución azul	0	0		0	1	0.1
Solución naranja	0.5	0.5	1		3	0.3
				Suma	10	1

S. Amarilla > S. Roja > S. Azul < S. Naranja

Tabla 11

Evaluación del peso específico del criterio respuesta del sistema.

Algoritmo de detección de objetos	Solución amarilla	Solución roja	Solución azul	Solución naranja	∑+1	Ponderación
Solución amarilla		0	0	0.5	1.5	0.15
Solución roja	1		0.5	1	3.5	0.35
Solución azul	1	0.5		1	3.5	0.35
Solución naranja	0.5	0	0		1.5	0.15
				Suma	10	1

S. Amarilla < S. Roja = S. Azul > S. Naranja

Tabla 12

Evaluación del peso específico del criterio transmisión de datos

Transmisión de Datos	Solución amarilla	Solución roja	Solución azul	Solución naranja	∑+1	Ponderación
Solución amarilla		1	1	0.5	3.5	0.35
Solución roja	0		0.5	0	1.5	0.15
Solución azul	0	0.5		0	1.5	0.15
Solución naranja	0.5	1	1		3.5	0.35
				Suma	10	1

S. Amarilla > S. Roja = S. Azul < S. Naranja

Tabla 13

Evaluación del peso específico del criterio algoritmo de detección de objetos.

Respuesta del Sistema	Solución amarilla	Solución roja	Solución azul	Solución naranja	∑+1	Ponderación
Solución amarilla		1	0.5	1	3.5	0.35
Solución roja	0		0	0.5	1.5	0.15
Solución azul	0.5	1		1	3.5	0.35
Solución naranja	0	0.5	0		1.5	0.15
				Suma	10	1

S. Amarilla > S. Roja < S. Azul > S. Naranja

Tabla 14

Evaluación del peso específico del criterio capacidad de procesamiento.

Plataforma de desarrollo de aplicación móvil	Solución amarilla	Solución roja	Solución azul	Aplicación móvil	∑+1	Ponderación
Solución amarilla		1	1	1	4	0.4
Solución roja	0		0.5	0.5	2	0.2
Solución azul	0	0.5		0.5	2	0.2
Solución naranja	0	0.5	0.5		2	0.2
				Suma	10	1

S. Amarilla > S. Roja = S. Azul = S. Naranja

Tabla 15

Tabla de conclusiones.

Conclusiones	Plataforma de desarrollo de aplicación móvil	Respuesta del sistema	Transmisión de Datos	Algoritmo de detección de objetos	Capacidad de procesamiento	∑	Ponderación
Solución amarilla	0.35*0.07	0.15*0.17	0.35*0.2	0.35*0.23	0.4*0.33	0.3325	1
Solución roja	0.25*0.07	0.35*0.17	0.15*0.2	0.15*0.23	0.2*0.33	0.2075	4
Solución azul	0.1*0.07	0.35*0.17	0.15*0.2	0.35*0.23	0.2*0.33	0.243	2
Solución naranja	0.3*0.07	0.15*0.17	0.35*0.2	0.15*0.23	0.2*0.33	0.217	3

La solución amarilla es la que posee la mejor prioridad, la solución azul es la segunda mejor puntuada, pero aun así existe una diferencia significativa en relación con la amarilla.

Por lo tanto, el sistema de reconocimiento de placas vehiculares está conformado por un ordenador de placa única “Raspberry Pi”, como unidad de procesamiento de la información,

recibe los datos emitidos por su módulo de cámara infrarroja, esta información será almacenada en internet con acceso remoto desde cualquier parte del mundo y se conectará mediante una aplicación móvil desarrollada en Android Studio para el manejo de la información.

Después de aplicar la solución recomendada (amarilla), se obtiene como resultados que el ordenador "Raspberry Pi" no tiene la capacidad de procesamiento para el tratamiento de imágenes en tiempo real, por lo tanto, se utilizara la solución 2 que es la siguiente mejor puntuada en la cual se utiliza un ordenador como unidad de procesamiento para el correcto funcionamiento del sistema.

3.6 Desarrollo del sistema de reconocimiento y almacenamiento de placas vehiculares

En esta sección se puntualizan los criterios de diseño del sistema desarrollado; se presentan los procesos de entrenamiento de un sistema de reconocimiento de objetos (S.R.O) mediante técnica de visión artificial desarrollado por el autor para la identificación de vehículos y sus correspondientes placas vehiculares. El procesamiento de imágenes mediante un ordenador para obtener la codificación de las placas. Se desarrolla una base de datos para la administración y almacenamiento de la información con la finalidad de que sea visualizada mediante una aplicación móvil por los usuarios.

3.6.1 Sistemas de reconocimiento de placas vehiculares.

Se desarrolla dos algoritmos de reconocimiento de objetos, detallados a continuación:

1. S. R. O pre entrenado por el autor mediante Machine Learning.
2. Real-time vehicle detection in Python (Software Libre, adaptable a los requerimientos del sistema de reconocimiento de placas vehiculares)

3.6.1.1 S. R. O pre entrenado por el autor mediante Machine Learning.

Es importante conocer la diferencia entre un clasificador de objetos y un reconocedor de objetos.

- Clasificación de imagen: algoritmo el cual se encarga de clasificar toda una imagen dentro de una categoría. Por ejemplo, le damos una foto de un auto y solo nos menciona que el objeto predominante en la foto es un auto.
- Reconocimiento de objetos: algoritmo el cual se encarga de detectar varios elementos dentro de una imagen y clasificarlos.

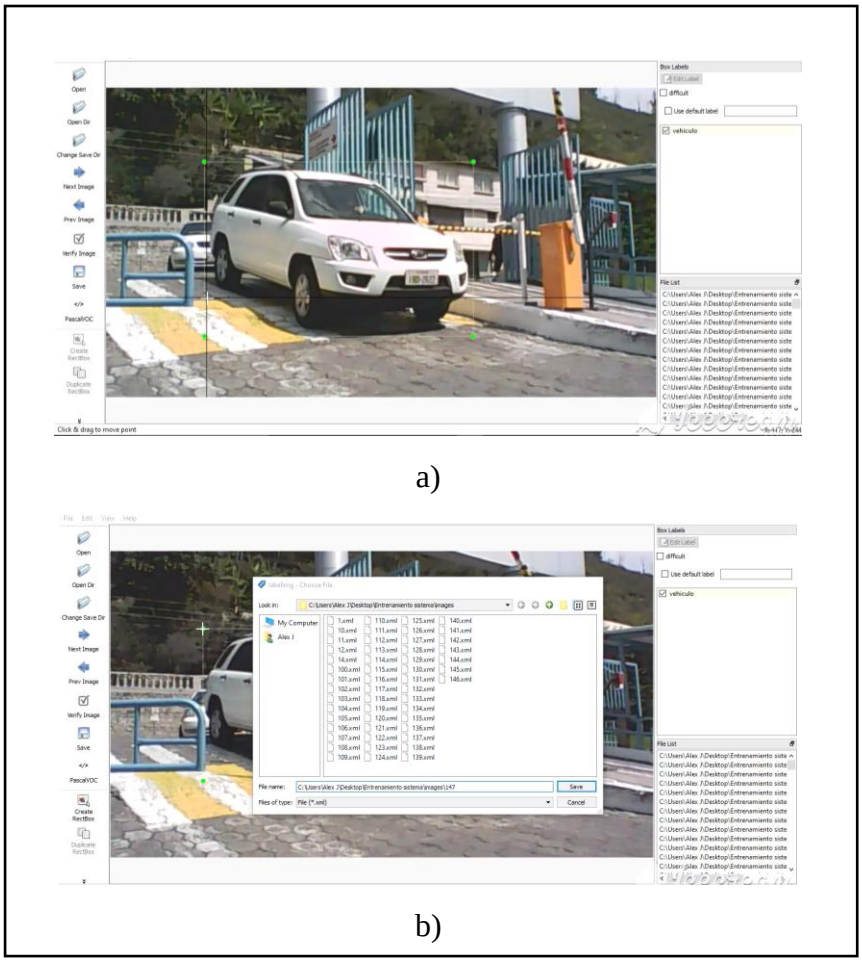
Requerimientos para el funcionamiento del sistema

- Python 3.6
- Tensorflow
- Numpy
- Pandas
- Matplotlib
- OpenCV (Open Source Computer Vision)
- Tarjeta de gráficos (Recomendada para poder hacer un entrenamiento de manera rápida, aunque es posible hacerlo sin GPU, puede llegar a tardar horas o días sin una tarjeta de gráficos NVIDIA)

Procedimiento para el entrenamiento del sistema

- ✓ ***Preparación de imágenes***

Después de una selección de imágenes adecuadas, se preparan para el entrenamiento, es decir marcar una sección en la imagen con las coordenadas donde están los objetos que se busca detectar, se utilizó el software TFrecord como se adjunta en la Ilustración 10.



a)

b)

Ilustración 10: Software TFrecord. a) Selección del área de interés de la imagen procesada. b) Archivos xml con coordenadas.

EL software TFrecord permite de forma fácil convertir los datos a un archivo XML necesario para poder entrenar el algoritmo.

Para la preparación de los datos se necesitaron 450 imágenes para el óptimo funcionamiento del sistema, se utilizó el algoritmo de visión artificial YOLO (You only look once, «sólo se mira una vez») que detecta y clasifica objetos en tiempo real.

Sobre la misma imagen se puede seleccionar distintos elementos de interés para que el programa los detecte, por ejemplo, en la misma imagen se puede marcar camionetas, autos, semáforos, letreros, pasos peatonales, entre otros.

- ✓ ***Entrenamiento.***

El sistema de reconocimiento de objetos necesita aplicar una derivación de MACHINE LEARNING para crear un sistema que aprenda de manera automatizada con la capacidad de identificar patrones complejos en las imágenes seleccionadas mediante un algoritmo [32].

Configuración de entrenamiento (faster_rcnn_resnet101_coco.config)

El modelo de entrenamiento de código libre faster_rcnn_resnet101_coco.config, se utilizó para encontrar el objeto de estudio en las imágenes que se van a entrenar. (Anexo 2)

Finalmente terminado el entrenamiento es necesario congelar el modelo terminado para poder realizar las pruebas correspondientes.

3.6.1.2 Real-time vehicle detection in Python (Software Libre, adaptable a los requerimientos del sistema de reconocimiento de placas vehiculares)

La detección de vehículos es una de las funciones más utilizadas por empresas y organizaciones en la actualidad. Esta tecnología utiliza visión por computadora para detectar diferentes tipos de vehículos en tiempo real a través de una cámara, en control de tráfico, rastreo de automóviles, creación de sensores de estacionamiento y muchos más, como se puede observar en la Ilustración 11:

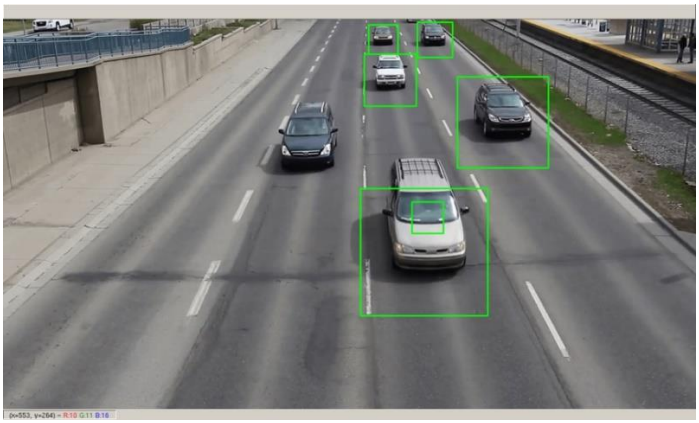


Ilustración 11: Modelo Real-time vehicle detection in Python

Fuente: [33]

En este proyecto, se desarrolla un sistema de reconocimiento de vehículos en Python para obtener la codificación de las placas vehiculares.

Es necesario instalar la biblioteca OpenCV (Open Source Computer Vision) que está diseñada para ayudar a los desarrolladores a realizar tareas relacionadas con la visión por computadora.

Procedimiento para el uso de un sistema de reconocimiento de objetos pre entrenado

El procedimiento según Venkatesh Chandra para la detección de vehículos en tiempo real en el lenguaje Python es el que se describe a continuación [33].

1. Primero se crea un ambiente de programación en el lenguaje Python para la instalación de las librerías necesarias para el correcto funcionamiento del sistema.
2. Importar la librería OpenCV desde un script en cualquier Entorno de Desarrollo Integrado (IDE) en Python.
3. Importar librerías adicionales para el óptimo funcionamiento del procesamiento de imágenes.
4. Configurar la entrada de la cámara web a utilizar

El video que capta el ordenador a través de la cámara Web, se divide en cuadros y se lee un cuadro a la vez. En cada cuadro, detectamos la ubicación del automóvil en el cuadro utilizando la interfaz de programación de aplicaciones (API).

Para cada automóvil detectado, se ubica las coordenadas, se dibuja un rectángulo alrededor y se muestra el video al usuario.

El código completo se muestra en el Anexo 3.

La primera sección del código detecta los vehículos en el cuadro y almacena sus coordenadas (ejes x, y, y el ancho y alto del vehículo).

La segunda sección dibuja un rectángulo alrededor del área donde se detecta el vehículo y muestra el texto "Vehículo" sobre el rectángulo. Puede cambiar la fuente del texto y el código (0, 0, 255) es el código de color del rectángulo y el texto en secuencia B-G-R.

La imagen resultante (fotograma) se muestra al espectador y el bucle continúa ejecutándose hasta que el usuario presiona la tecla (Q) en el teclado para finalizar.

Después de varias pruebas ejecutadas con las dos versiones de programas de reconocimiento de objetos, se opta por utilizar el segundo método: *Real-time vehicle detection in Python (Software Libre, adaptable a los requerimientos del sistema de reconocimiento de placas vehiculares),* debido a un tiempo de respuesta ágil y rápido y permite optimizar el proceso de obtención de la placa vehicular para el correcto funcionamiento de los siguientes módulos.

3.7 Procesamiento de la información a partir del algoritmo Real-time vehicle detection in Python.

A partir de la imagen original captada por el sistema Real-time vehicle detection in Python empieza el procesamiento de la imagen en la región de interés, que es en donde se ubica la placa vehicular.

Paso 1:

Detección de vehículo (Código en Anexo 4)

Paso 2:

Aplicación de filtros en la imagen del vehículo para determinar el área de contornos de la placa vehicular (Código en Anexo5) como se puede observar en la Ilustración 12.

Ilustración 12: Aplicación de filtros en la imagen del vehículo

Paso 3:

Aplicación de un bucle para detectar los contornos de la placa vehicular y generar una nueva ventana con la imagen de la placa únicamente, mostrado en la Ilustración 13. (Código en Anexo 6)

Ilustración 13: Aplicación de un bucle para detectar los contornos de la placa vehicular.

Paso 4:

Recortar el área de interés para aplicar filtros de eliminación de ruido. Ilustración 14 (Código en Anexo 7)

Ilustración 14: Ventana de placa vehicular aplicando filtros.

Paso 5:

Se aplica la librería Pytesseract que es una herramienta de OCR (Reconocimiento Óptico de Caracteres) para Python, para obtener el String de la placa vehicular. Como se puede observar en la Ilustración 16. (Código en Anexo 8)

Paso 6:

Aplicación de filtros para obtener el correcto String de la placa vehicular. (código en Anexo 9)

Código completo adjunto en Anexo 10.

3.8 Base de datos 000WebHost

Posterior a obtener el String de la placa vehicular mediante el sistema de reconocimiento de objetos y una Webcam, se procede a desarrollar una Base de Datos en 000WebHost Ilustración 15, que es un Hosting Web más grande de la red especialmente en la modalidad gratuita [34].

Ilustración 15: 000 WebHost

Fuente: [34]

Pasos para crear la base de datos en 000WebHost.

Paso 1:

Crear una cuenta gratuita en la página oficial de 000WebHost con capacidad de almacenamiento de 1.5 GB como se puede ver en la Ilustración 16.

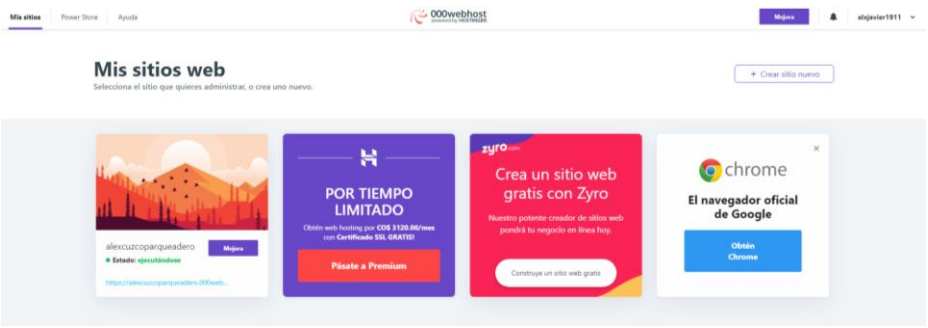


Ilustración 16: Cuenta Gratuita en 000WebHost

Fuente: [34]

Paso 2:

Mediante una de sus herramientas, se administra una base de datos gratuita.

Este sitio web permite crear hasta 2 bases de datos como se muestra en la Ilustración 17.

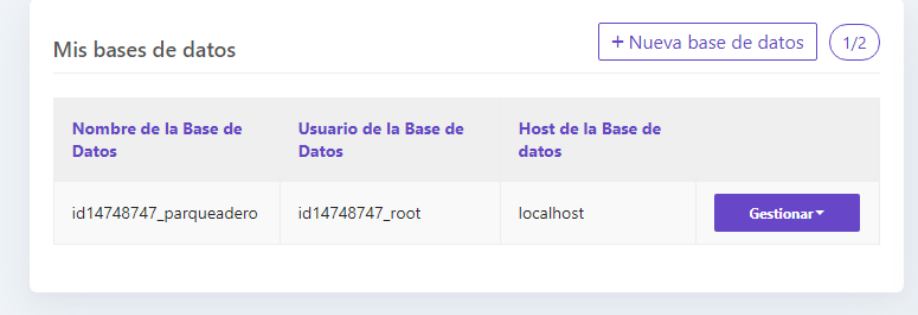


Ilustración 17: Administrador de base de datos en 000WebHost

Paso 3:

Se crea una base de datos mediante la herramienta gratuita phpMyAdmin que permite acceder a todas las funciones de la base de datos MySQL mediante una interfaz web intuitiva como se muestra en la Ilustración 18.

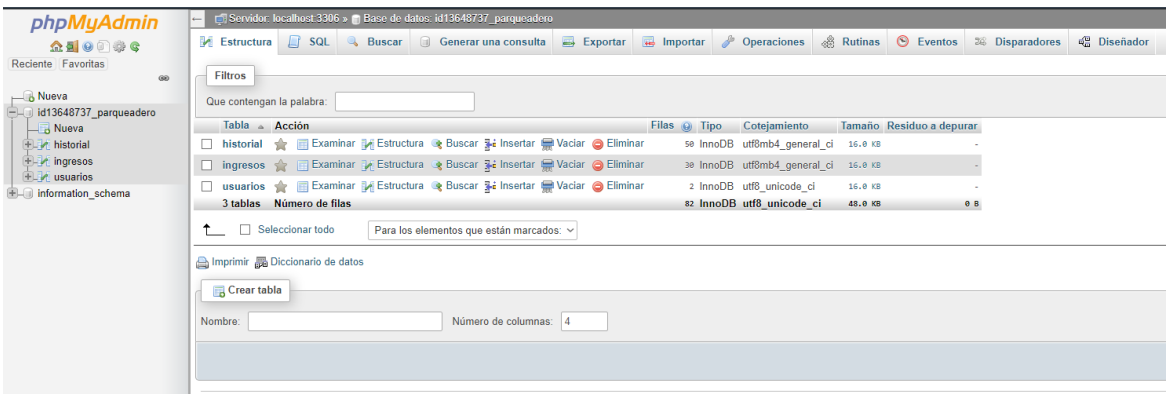


Ilustración 18: Aplicación phpMyAdmin

Paso 4:

Conexión con el sistema de reconocimiento de placas vehiculares para el almacenamiento del String en la tabla correspondiente, (Código en Anexo 11)

3.9 Aplicación móvil

Luego de obtener el String de la placa vehicular y almacenarlo en una base de datos es necesario la visualización y manejo de la información mediante una interfaz gráfica en una aplicación móvil.

El desarrollo de la aplicación móvil fue en Android Studio por la facilidad de programación y la compatibilidad con dispositivos móviles actuales, como se puede apreciar en la Ilustración 19.

Ilustración 19: Interfaz de aplicación móvil

Funciones principales:

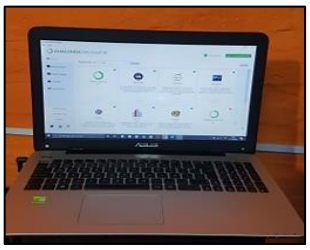
- Consulta del historial de entrada y salida de vehículos con fecha y hora correspondiente. (Código en el Anexo 12)
- Visualización de lugares disponibles en un parqueadero.

3.10 Desarrollo de diagramas de funcionamiento y bloques del sistema

En la Ilustración 20 se describe el diagrama de funcionamiento del sistema de reconocimiento de placas vehiculares.

1. El modelo pre entrenado es el que capta la información del exterior a través de una cámara.
2. Un ordenador con la capacidad para el procesamiento de imágenes con el objetivo de obtener la codificación de una placa vehicular.
3. Almacenamiento de la información en la web mediante una base de datos alojados en la nube.
4. Aplicación para el manejo y visualización de la información obtenida por el sistema.

Sistema de reconocimiento de objetos a través de una webcam



Ordenador con tarjeta gráfica

Aplicación móvil para usuarios

Base de datos alojados en la nube

Ilustración 20: Diagrama de funcionamiento del sistema de reconocimiento de placas vehiculares

El diagrama de la ilustración 20, muestra 4 bloques principales para el correcto funcionamiento del sistema, la obtención de la placa vehicular y posteriormente el manejo de la información a través de una aplicación móvil para generar una óptima gestión y automatización del acceso a un parqueadero.

El bloque 1 corresponde al sistema Real-time vehicle detection in Python (software libre), el cual es el encargado de captar el objeto de interés desde el exterior por medio de una cámara para su posterior procesamiento mediante un ordenador.

El bloque 2 corresponde al procesamiento de datos e imágenes captadas por el sistema Real-time vehicle detection in Python para la obtención del String de la placa vehicular para posteriormente almacenarlo en una base de datos.

El bloque 3 es el encargado de almacenar los datos de los vehículos en una base de datos alojada en la web conocida como 000WebHost, que según [34], es un Hosting Web gratuito con 1.5 GB de almacenamiento, y con la capacidad de crear 2 bases de datos MySQL, para su manejo y visualización de la información.

El bloque 4 es el encargado de mostrar la información mediante un dispositivo móvil con dos funciones principales:

1. Mostrar el historial de ingreso y salida de los vehículos a un parqueadero.
2. Mostrar los espacios disponibles en el parqueadero.

En la Ilustración 21 se muestra es diagrama de bloques del sistema desarrollado.

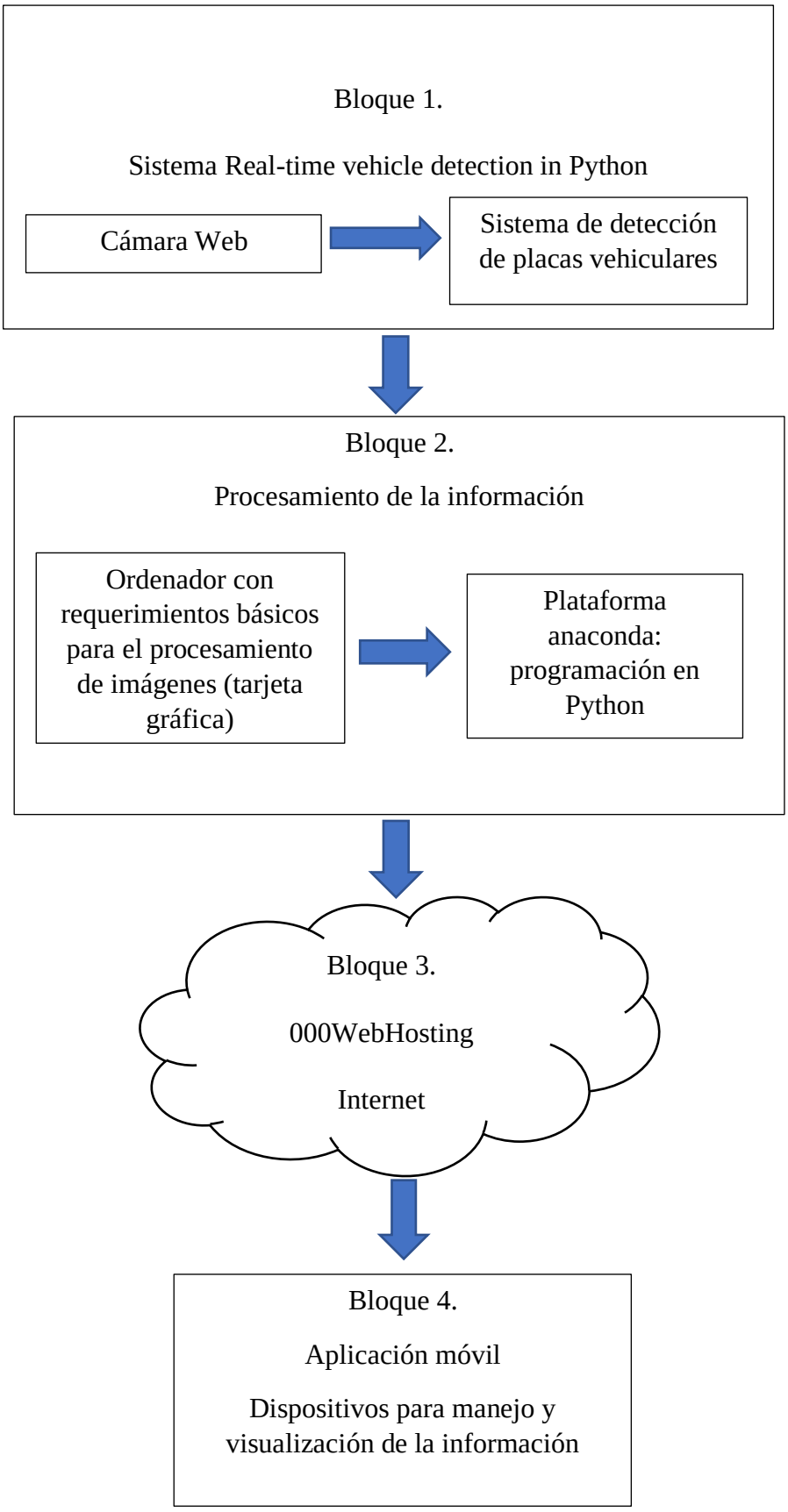


Ilustración 21: Diagrama de bloques del sistema desarrollado

CAPÍTULO IV

4. RESULTADOS

4.1 Análisis de resultados

4.1.1 Análisis del algoritmo de reconocimiento de placas vehiculares

Como se describió en el diagrama de bloques de la Ilustración 21 la cámara web y el sistema Real-time vehicle detection in Python corresponden al bloque 1, en el cual se configuró una cámara web como ojos del sistema para captar la información de interés del exterior, además se adaptó el código fuente de uso libre para la detección de vehículos de acuerdo con los requerimientos del trabajo de titulación.

Obteniendo de esta manera un sistema funcional con la capacidad de detectar los vehículos, procesar la información obtenida y detectar la placa vehicular después de varias pruebas realizadas.

Como resultado del S. R. O pre entrenado por el autor mediante Machine Learning, se obtiene los siguientes datos:

Tabla 16

Detección de vehículo a 1 metro de distancia. (S. R. O pre entrenado por el autor mediante Machine Learning.)

S. R. O pre entrenado por el autor mediante Machine Learning.	Placa vehicular	Detección de vehículo	Tiempo transcurrido (segundos)
Prueba 1	PJQ-512	**NO**	-
Prueba 2	PIX-499	**NO**	-
Prueba 3	TDH-398	**NO**	-
Prueba 4	PDA-8188	**NO**	-
Prueba 5	POA-818	**NO**	-

La información de la tabla 16 muestra el resultado de detección del vehículo en tiempo real con el tiempo transcurrido en segundos, se puede apreciar que las pruebas realizadas a 1 metro de distancia de separación entre el vehículo y la cámara de reconocimiento del sistema no muestra un resultado satisfactorio, no se obtiene ningún reconocimiento del vehículo.

Tabla 17

Detección de vehículo a 3 metros de distancia. (S. R. O pre entrenado por el autor mediante Machine Learning.)

S. R. O pre entrenado por el autor mediante Machine Learning.	Placa vehicular	Detección de vehículo	Tiempo transcurrido (segundos)
Prueba 1	PJQ-512	**SI**	**10**
Prueba 2	PIX-499	**NO**	**-**
Prueba 3	TDH-398	**NO**	**-**
Prueba 4	PDA-8188	**SI**	**12**
Prueba 5	POA-818	**NO**	**-**

La información de la tabla 17 muestra el resultado de detección del vehículo en tiempo real con el tiempo transcurrido en segundos, se puede apreciar que las pruebas realizadas a 3 metros de distancia de separación entre el vehículo y la cámara de reconocimiento del sistema aun no son totalmente satisfactorias, obteniendo 2 de 5 reconocimientos. El tiempo promedio de respuesta es de 11 segundos.

Tabla 18

Detección de vehículo a 5 metros de distancia. (S. R. O pre entrenado por el autor mediante Machine Learning.)

S. R. O pre entrenado por el autor mediante Machine Learning.	Placa vehicular	Detección de vehículo	Tiempo transcurrido (segundos)
Prueba 1	PJQ-512	**NO**	**-**
Prueba 2	PIX-499	**NO**	**-**
Prueba 3	TDH-398	**NO**	**-**
Prueba 4	PDA-8188	**SI**	**14**
Prueba 5	POA-818	**NO**	**-**

La información de la tabla 18 muestra el resultado de detección del vehículo en tiempo real con el tiempo transcurrido en segundos, se puede apreciar que las pruebas realizadas a 5 metros de distancia de separación entre el vehículo y la cámara de reconocimiento del sistema decrecieron en relación con los reconocimientos a 3 metros de distancia; obteniendo 1 único reconocimiento.

Como resultado del sistema Real-time vehicle detection in Python, se obtiene los siguientes datos:

Tabla 19

Detección de vehículo a 1 metros de distancia. (Sistema Real-time vehicle detection in Python)

Sistema Real-time vehicle detection in Python	Placa vehicular	Detección de vehículo	Tiempo transcurrido (segundos)
Prueba 1	PJQ-512	**NO**	**-**
Prueba 2	PIX-499	**SI**	**6**
Prueba 3	TDH-398	**NO**	**-**
Prueba 4	PDA-8188	**NO**	**-**
Prueba 5	POA-818	**SI**	**5**

La información de la tabla 19 muestra el resultado de detección del vehículo en tiempo real utilizando el algoritmo de detección de vehículos del sistema Real-time vehicle detection in Python obteniendo 2 de 5 reconocimientos con la distancia de 1 metro de separación entre el vehículo y la cámara de reconocimiento del sistema.

Tabla 20

Detección de vehículo a 3 metros de distancia. (Sistema Real-time vehicle detection in Python)

Sistema Real-time vehicle detection in Python	Placa vehicular	Detección de vehículo	Tiempo transcurrido (segundos)
Prueba 1	PJQ-512	**SI**	**3**
Prueba 2	PIX-499	**SI**	**2**
Prueba 3	TDH-398	**SI**	**3**
Prueba 4	PDA-8188	**SI**	**3**
Prueba 5	POA-818	**SI**	**3**

La información que muestra la tabla 20 es totalmente satisfactoria con 5 reconocimientos de los vehículos en las 5 pruebas realizadas a 3 metros de separación entre el vehículo y la cámara de reconocimiento del sistema. Resultado obtenido con el algoritmo de detección de vehículos del sistema Real-time vehicle detection in Python.

El tiempo promedio transcurrido es de 2.8 segundos para el reconocimiento del vehículo.

Tabla 21

Detección de vehículo a 5 metros de distancia. (Sistema Real-time vehicle detection in Python)

Sistema Real-time vehicle detection in Python	Placa vehicular	Detección de vehículo	Tiempo transcurrido (segundos)
Prueba 1	PJQ-512	**SI**	**7**
Prueba 2	PIX-499	**NO**	**-**
Prueba 3	TDH-398	**SI**	**8**
Prueba 4	PDA-8188	**NO**	**-**
Prueba 5	POA-818	**NO**	**-**

La información de la tabla 21 muestra el resultado de detección del vehículo en tiempo real utilizando el algoritmo de detección de vehículos del sistema Real-time vehicle detection in Python obteniendo 2 de 5 reconocimientos con la distancia de 5 metros de separación entre el vehículo y la cámara de reconocimiento del sistema.

En la Ilustración 22 se muestra el tiempo promedio de respuesta del Sistema Real-time vehicle detection in Python vs S. R. O pre entrenado por el autor mediante Machine Learning a 3 metros de separación del vehículo y la cámara del sistema.

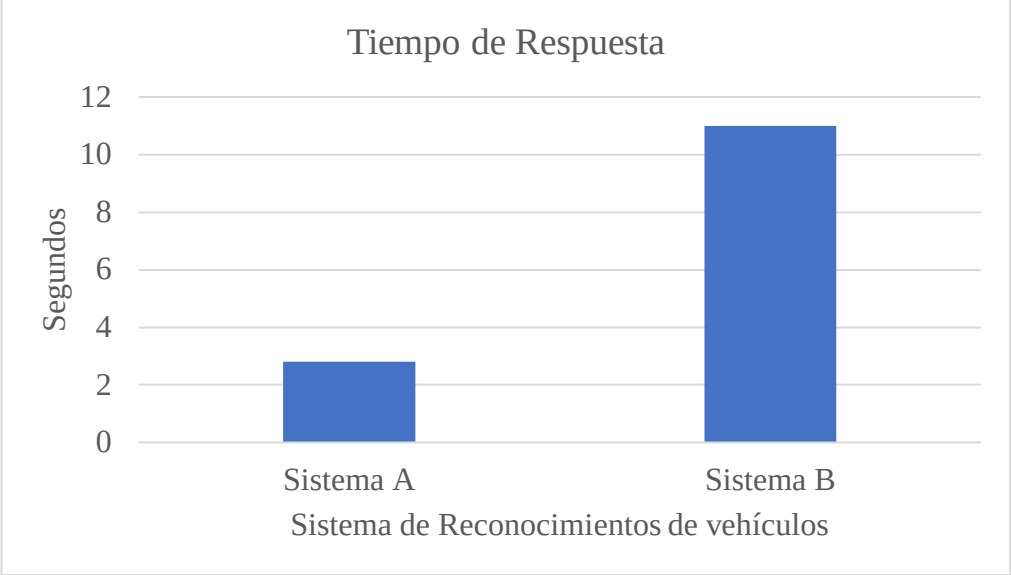


Ilustración 22: Tiempo de respuesta de los sistemas A y B.

De acuerdo con el resultado obtenido mediante las pruebas en los 2 tipos de sistemas, Sistema Real-time vehicle detection in Python (Sistema A) y S. R. O pre entrenado por el autor mediante Machine Learning. (Sistema B), se define que el mejor sistema para el reconocimiento de vehículos es el sistema A, con un tiempo de respuesta promedio de 2.8 segundos, con respecto al tiempo de respuesta del sistema B con 11 segundos.

Parámetros considerados para el correcto funcionamiento del sistema:

- Inclinación de la cámara con respecto al suelo: 45 grados aproximadamente.
- Altura de la cámara: 90 cm aproximadamente con respecto al suelo.

Por lo tanto, en base al mejor tiempo de respuesta se utiliza el Sistema Real-time vehicle detection in Python (Sistema A) para el desarrollo del proyecto tomando en cuenta todos los parámetros sugeridos en las pruebas de campo, para garantizar el correcto funcionamiento del sistema en todos sus módulos.

4.1.2 Análisis de configuración de base de datos

Posterior a la obtención del String de la placa vehicular es necesario almacenar esta información para su manejo y visualización.

El método más viable fue mediante la nube, que permitió alojar los datos de forma gratuita, el Hosting Web 000WebHost se vinculó de manera fácil y ágil con la aplicación phpMyAdmin que es en donde se almacena la tabla de datos de las placas vehiculares con fecha y hora de ingreso o salida correspondiente.

El 100 % de la información obtenida por el sistema de reconocimiento de placas vehiculares fue almacenada en la base de datos, alojada en 000WebHost.

Además, no existe interferencia (errores de conexión) al consultar los datos desde el dispositivo móvil debido a su fácil programación.

4.1.3 Análisis de aplicación móvil

La aplicación móvil facilitó la visualización de los datos alojados en la web mediante 000WebHost, al ser desarrollada en la plataforma de software libre Android Studio, es compatible con la mayoría de los dispositivos móviles actuales, de acuerdo con las pruebas realizadas con el dispositivo Samsung Galaxy S7 Edge mostrado en la tabla 22.

Tabla 22

Pruebas de enlace de aplicación móvil y base de datos.

Aplicación móvil	Conexión con base datos	Datos Guardados.	Placa vehicular registrada
Prueba1	**SI**	**SI**	**PBU-1481**
Prueba2	**NO**	**NO**	**-**
Prueba3	**SI**	**SI**	**PIX-499**
Prueba4	**SI**	**SI**	**IBE-1481**
Prueba5	**SI**	**SI**	**ICO-4332**
Prueba 6	**SI**	**SI**	**PJQ-512**
Prueba 7	**SI**	**SI**	**PIX-499**
Prueba 8	**NO**	**NO**	**-**
Prueba 9	**SI**	**SI**	**TDH-398**
Prueba 10	**SI**	**SI**	**POA-8188**

De acuerdo con los datos obtenidos en la tabla 22 muestra un margen de error de conectividad del 20 %, durante las pruebas ejecutadas.

La aplicación es de fácil uso e interactiva con el usuario de acuerdo con el desarrollo del autor y la facilidad de programación que ofrece la plataforma de desarrollo Android Studio, además tiene la capacidad de modificar o eliminar datos existentes en la base de datos desde el dispositivo móvil y así optimizar tiempo y recursos.

4.2 Experimentación y pruebas

4.2.1 Plan de Pruebas

Se realizaron 20 pruebas de campo en dos condiciones de clima muy comunes; clima nublado (A), y clima soleado (B), con el Sistema Real-time vehicle detection in Python

(Sistema A), para evaluar el sistema en cada uno de sus módulos, y se obtuvo los siguientes resultados:

Tabla 23

Pruebas de campo en clima nublado (A)

Código	Placa	Fecha	Hora	Acierto
1	PBU-1481	26/8/2020	10H32	Si
G2	PJQ-512	31/8/2020	16H31	Si
3	PIX-499	2/9/2020	11H33	No
4	IBE-1481	10/9/2020	11H15	Si
5	ICO-4332	10/9/2020	10H37	Si
6	TDH-398	14/9/2020	10H49	Si
7	PDA-8188	24/9/2020	15H30	Si
8	PAO-818	24/9/2020	15H03	No
9	IBA-8566	25/9/2020	12h06	Si
10	IBC-9837	25/9/2020	08H45	Si

La información de la Tabla 23 muestra los datos obtenidos mediante el sistema de reconocimiento de placas vehiculares, consta de código que es el orden de registro de ingreso o salida del vehículo en el sistema, seguido del String de la placa vehicular captada por la cámara y procesado mediante el ordenador con fecha y hora correspondiente. Esta información es almacenada en el Hosting Web 000WebHost en una base de datos, además muestra el margen de error del funcionamiento del sistema en las condiciones de clima nublado con un 20% de error, como se puede apreciar en la Ilustración 23.

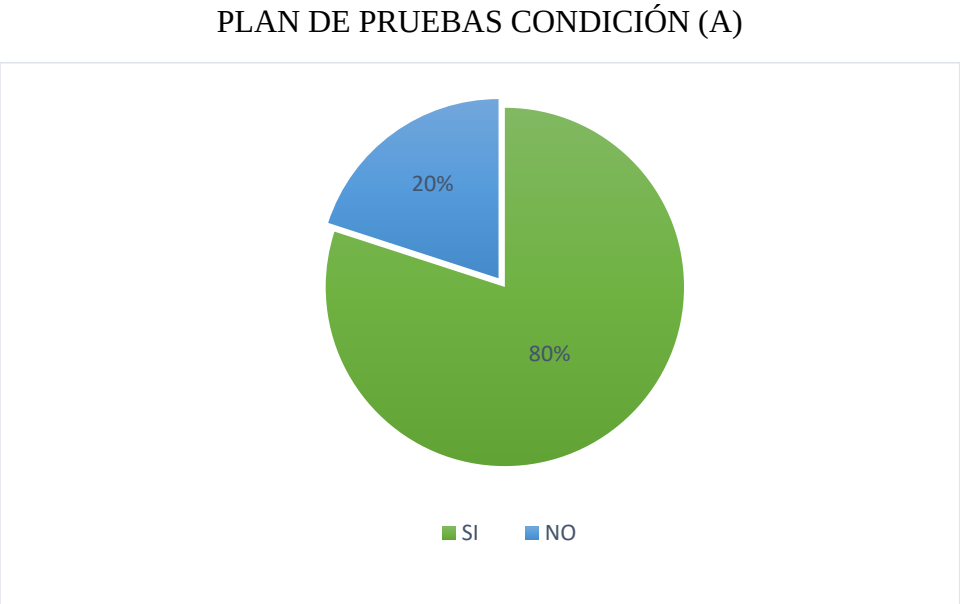


Ilustración 23: Porcentaje de error en plan de prueba (A)

Tabla 24

Pruebas de campo en el clima soleado (B)

Código	Placa	Fecha	Hora	Acierto
1	PBU-1481	26/8/2020	10H32	Si
2	PJQ-512	31/8/2020	16H31	Si
3	POA-818	2/9/2020	11H33	No
4	IBE-1481	10/9/2020	11H15	Si
5	ICO-4332	10/9/2020	10H37	Si
6	TDH-398	14/9/2020	10H49	Si
7	PDA-8188	24/9/2020	15H30	Si
8	IDE-000	24/9/2020	15H03	No
9	IBA-8566	25/9/2020	12h06	Si
10	IBC-9837	25/9/2020	08H45	Si

La información de la Tabla 24 muestra los datos obtenidos mediante el sistema de reconocimiento de placas vehiculares, consta de código que es el orden de registro de ingreso o salida del vehículo en el sistema, seguido del String de la placa vehicular captada por la

cámara y procesado mediante el ordenador con fecha y hora correspondiente. Esta información es almacenada en el Hosting Web 000WebHost en una base de datos, además muestra el margen de error del funcionamiento del sistema en las condiciones de clima soleado con un 20% de error, como se puede apreciar en la Ilustración 24.

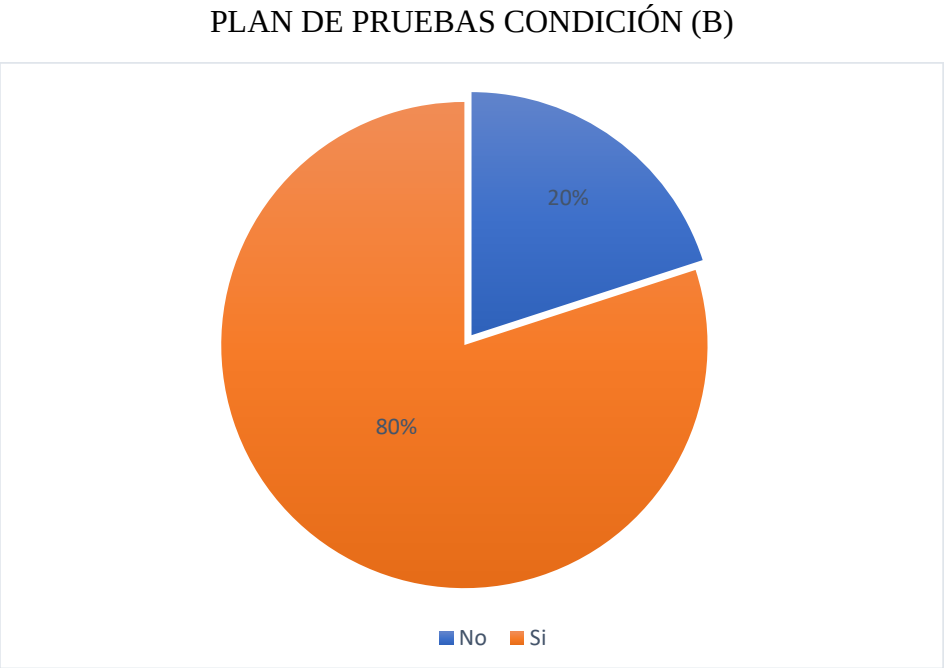


Ilustración 24: Porcentaje de error en plan de prueba (B)

Mediante el plan de pruebas se ha evaluado el funcionamiento del sistema en cada uno de sus módulos con los siguientes resultados:

En el módulo 1 que corresponde al funcionamiento de sistema Real-time vehicle detection in Python (software libre), se obtuvo las fotografías adecuadas mediante la cámara web con una posición de 90 cm sobre el nivel del suelo y además con una inclinación de 45° con vista a el ingreso o salida del vehículo.

En el módulo 2 corresponde al procesamiento de las imágenes captadas por el sistema Real-time vehicle detection in Python, en el cual se aplica varios filtros para la obtención del String de la placa vehicular, por medio de la librería Teseeract de software libre.

Una vez obtenido el String de la placa vehicular se almacena en una base de datos denominada parqueadero con información como fecha y hora de ingreso o salida de los vehículos todos estos datos correspondientes al módulo 3.

Finalmente, en el módulo 4, de acuerdo con las pruebas de enlace entre la aplicación móvil y la base de datos con un 20 % de margen de error en la conectividad.

Mediante esta aplicación se pudo realizar consultas de los datos existentes de ingreso y salida de los usuarios con sus respectivas fecha y hora, además de mostrar la disponibilidad de espacios libres de un parqueadero.

CONCLUSIONES

- Las condiciones adecuadas para la adquisición de imágenes son: distancia entre la cámara y el vehículo que debe ser de 3 metros aproximadamente. La posición de la cámara debe tener una inclinación de 45° aprox. Y finalmente las condiciones climatológicas nublado y soleado, afectan en 20 % el funcionamiento del sistema, debido a la variación de la luz.
- Se decide emplear una cámara web sobre las de seguridad, cámara wifi o RaspBerrypi para detectar los vehículos debido a su facilidad de programación y compatibilidad con el ordenador y software de desarrollo.
- Las características útiles para el reconocimiento e identificación de datos son el tamaño y las líneas de los contornos de las placas vehiculares.
- La aplicación móvil implementada logra visualizar la disponibilidad de espacios en el parqueadero, desde la información que se dispone previamente en una base de datos, lo cual optimiza el tiempo y facilita el manejo de la información por el usuario.
- La eficiencia del sistema es del 85% de acuerdo con las pruebas realizadas en las condiciones climatológicas soleado y nublado, tomando en cuenta los parámetros de: distancia de 3 metros de separación entre el vehículo y la cámara del sistema e inclinación de la cámara.

RECOMENDACIONES

- Realizar investigaciones de dispositivos de conexión inalámbrica para optimizar el funcionamiento del sistema de reconocimiento de placas vehiculares.
- Es importante se realice la investigación de algoritmos de reconocimiento de objetos que permitan disminuir el tiempo de respuesta del sistema.
- Implementar dispositivos de alta resolución para disminuir la distancia entre el vehículo y la cámara del sistema y así optimizar el tiempo de respuesta.
- Desarrollo de una aplicación móvil con la opción de pago del servicio de parqueadero, mediante dinero electrónico.

BIBLIOGRAFÍA

[1] M. D. M. L. QUINTANA ANDRÉS, «Sistema de visión artificial para conteo de objetos en movimiento. El Hombre y la máquina, num. 40,» Redalyc, pp. 87-101, 2012.

[2] Instituto Nacional de Estadística y Censos del Ecuador, «Instituto Nacional de Estadística y Censos del Ecuador,» 2010. [En línea]. Available: https://www.ecuadorencifras.gob.ec/institucional/home/. [Último acceso: 07 Enero 2020].

[3] Gobierno Autónomo Descentralizado de Ibarra, «Alcaldía de Ibarra,» 5 julio 2019. [En línea]. Available: https://www.ibarra.gob.ec/site/blog/2019/07/05/municipio-de-ibarra-articula-acciones-frente-a-la-problematica-de-movilidad-humana-en-el-canton/. [Último acceso: 10 01 2020].

[4] G. d. i. EDMANS, «Técnicas y algoritmos básicos de visión artificial: Silicon Develop,» 8 Mayo 2010. [En línea]. Available: https://silicondevelop.files.wordpress.com/2010/05/tecnicas-y-algoritmos-basicos-de-vision-artificial.pdf. [Último acceso: 2020 01 12].

[5] C. V. Á. G. ,. F. J. P. M. Juan Romero, Inteligencia Artificial y computación Avanzada, Santiago de Compostela: Fundación Alfredo Brañas, 2007.

[6] H. F. Cañadas Betancourt, « Prototipo de un sistema de un siste,a de adquisición de imágenes de vehículos, detección y reconocimiento automático de los caracteres de la placa en tiempo real por medio de visipon artificial, aplicado al control vehicular: Escuela Politécnica Nacional,» 2011. [En línea]. Available: https://doi.org/10.1007/s13398-014-0173-7.2. [Último acceso: 18 Enero 2020].

[7] S. d. I. d. Transporte, «Ministerio de Transporte y Obras Públicas del Ecuador,» 2013. [En línea]. Available: https://www.obraspublicas.gob.ec/wp-

content/uploads/downloads/2013/12/01-12-2013_Manual_NEVI-12_VOLUMEN_2A.pdf. [Último acceso: 20 Enero 2020].

[8] F. G. O. Zamora, «Uiversidad de Alicante, Departamento de Física, Ingeniería en Sistemas y Teoría de la Señal,» Mayo 2002. [En línea]. [Último acceso: 20 Enero 2020].

[9] A. S. CALLE, Aplicaciones de la Visión Artificial y la Biométrica Informática, Madrid: Dykinson, 2005.

[10] I. I. y. P. L. Abdelmalik Moujahid, «Departamento de Ciencias de la Computación e Inteligencia Artificial,» 2011. [En línea]. Available: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t9knn.pdf.

[11] S. C. a. I. I. Systems, «Soft Computing and Intelligent Information Systems,» 2003. [En línea]. Available: https://sci2s.ugr.es/docencia/algoritmica/Practicas%20-%20SI.pdf. [Último acceso: 15 Febrero 2020].

[12] G. L. Pedro Viñuela, Redes de Neuronas Artificiales. Un enfoque Práctico, Pearson Prentice Hall, 2003.

[13] Y. P. S. Y. Mohamed Haniza, «Object detection using geometric invariant moment: Engineering, School of Computer and Communication,» 2006. [En línea]. Available: http://dspace.unimap.edu.my/bitstream/handle/123456789/11347/ajas361876-1878.pdf?sequence=1&isAllowed=y. [Último acceso: 8 Marzo 2020].

[14] J. G. G. A. V. M Arévalo, «OpenCV. La Librerí a Open Source de Visión Artificial,» Linux Free Magazine, vol. 6ta Edición, pp. 143-145, 2005.

[15] J. I. A. GUADALUPE, «ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO,» 2017. [En línea]. Available:

http://dspace.espoch.edu.ec/bitstream/123456789/6158/1/20T00812.pdf. [Último acceso: 1 Marzo 2020].

[16] L. A. Eli Stevens, Deep Learning with PyTorch, Shelter Island: Manning, 2019.

[17] P. D. Pablo Alvarado, «LTI Image Processing Library: Developer's Guide,» 21 Julio 2004. [En línea]. Available: http://www.ie.tec.ac.cr/palvarado/ltilib-2/styleguide/en/DevelopersGuide.pdf. [Último acceso: 4 Marzo 2020].

[18] E. I. Jiva, «Departamento de Ingeniería de Sistemas y Automática de la Universidad Politécnica de Valencia,» Julio 2019. [En línea]. Available: https://riunet.upv.es/bitstream/handle/10251/130010/Jiva%20-%20Desarrollo%20de%20la%20teleoperación%20de%20robots%20industriales%20y%20colaborativos%20mediante%20técnicas%20av....pdf?sequence=1&isAllowed=y. [Último acceso: 4 octubre 2020].

[19] T. S. Lin, «Centre pour la Comunication Scientifique Directe,» 17 Diciembre 2015. [En línea]. Available: https://www.researchgate.net/publication/290182056_Smart_Parking_Network_Infrastructure_and_Urban_Service. [Último acceso: 4 Abril 2020].

[20] Y. L. E. T. N. N. a. Z. R. M. Idris, «Science Alert - Car Park System: A Review of Smart Parking System and its Technology,» 2009. [En línea]. Available: https://scialert.net/abstract/?doi=itj.2009.101.113. [Último acceso: 5 Abril 2020].

[21] S. A. S. Caroline Rodier, «Institute of Transportation Studies, UC Berkeley,» 12 Enero 2004. [En línea]. Available: https://www.researchgate.net/publication/46439569_Transit-Based_Smart_Parking_in_the_San_Francisco_Bay_Area_an_Assessment_of_User_Demand_and_Behavioral_Effects. [Último acceso: 8 Abril 2020].

[22] S. A. S. Caroline Rodier, «ParkingDemonstration, Institute of transportation Studies-mart Parking Management Field Test: A Bay Area Rapid Transit (BART) District; Report, Final,» 6 Enero 2008. [En línea]. Available: https://www.researchgate.net/publication/46440061_Smart_Parking_Management_Field_Test_A_Bay_Area_Rapid_Transit_BART_District_Parking_Demonstration_Final_Report. [Último acceso: 25 Abril 2020].

[23] S. A. S. Caroline Rodier, «Institute of transportation Studies-mart Parking Management Field Test: A Bay Area Rapid Transit (BART) District ParkingDemonstration; Final Report,» 06 Enero 2008. [En línea]. Available: https://www.researchgate.net/publication/46440061_Smart_Parking_Management_Field_Test_A_Bay_Area_Rapid_Transit_BART_District_Parking_Demonstration_Final_Report. [Último acceso: 26 Enero 2020].

[24] F. d. S. L. d. Europa, «Fundación del Software Libre de Europa Empowering users to control technology,» 1 Enero 2020. [En línea]. Available: https://fsfe.org/about/basics/freesoftware.es.html. [Último acceso: 8 Mayo 2020].

[25] F. a. GNU, «El sistema operativo GNU,» 15 Septiembre 2019. [En línea]. Available: https://www.gnu.org/licenses/licenses.es.html. [Último acceso: 20 Mayo 2020].

[26] O. S. H. Association, «Open Source Hardware Association,» WordPress, [En línea]. Available: https://www.oshwa.org/definition/spanish/. [Último acceso: 26 Mayo 2020].

[27] D. E. 1196, «REGLAMENTO GENERAL PARA LA APLICACION DE LA LEY ORGANICA DE TRANSPORTE TERRESTRE, TRÁNSITO Y SEGURIDAD VIAL,» 25 Junio 2012. [En línea]. Available: https://www.obraspublicas.gob.ec/wp-content/uploads/downloads/2015/03/Decreto-Ejecutivo-No.-1196-de-11-06-2012-

REGLAMENTO-A-LA-LEY-DE-TRANSPORTE-TERRESTRE-TRANSITO-Y-SEGURIDAD-VIA.pdf. [Último acceso: 4 Abril 2020].

[28] P. R. Vargas, «Superintendencia de Telecomunicaciones,» Agosto 2007. [En línea]. Available: https://www.imaginar.org/docs/historia_telecomunicaciones.pdf. [Último acceso: 9 Junio 2020].

[29] C. C. D. O. Olaya Erika, «Despliegue de la función calidad (QFD): beneficios y limitaciones detectados en su aplicación al diseño de prótesis mioeléctrica de mano,» Ingeniería e Investigación , vol. 25, pp. 3-6, 2005.

[30] R. E. Montoya, «Introducción al desarrollo web,» 2013. [En línea]. Available: https://core.ac.uk/download/pdf/16376139.pdf. [Último acceso: 2 octubre 2020].

[31] C. Riba, Diseño concurrente, Barcelona:: Edicions UPC, 2002.

[32] S. Raschka, Python Machine Learning, Birmingham: Packt Publishing Ltd., 2017.

[33] V. Chandra, «Vehicle Detection in Real Time and Recorded Videos in Python - Windows and macOS,» 1 Enero 2020. [En línea]. Available: https://medium.com/analytics-vidhya/vehicle-car-detection-in-real-time-and-recorded-videos-in-python-windows-and-macos-c5548b243b18. [Último acceso: 10 Agosto 2020].

[34] 0. WebHost, «000 WebHost,» [En línea]. Available: https://www.000webhost.com. [Último acceso: 3 10 2020].

[35] phpmyadmin.net, «phpmyadmin.net,» [En línea]. Available: https://www.phpmyadmin.net. [Último acceso: 4 Octubre 2020].

[36] I. N. d. E. y. Censos, «Población y Geografía,» INEC, 13 Julio 2020. [En línea]. Available: https://www.ecuadorencifras.gob.ec/?s=geografia. [Último acceso: 7 Noviembre 2020].

ANEXOS

Anexo 1

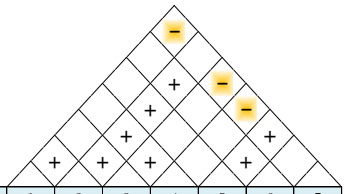
Casa de la calidad

Anexo 1

Correlaciones	
Positivo	+
Negativo	−
No Correlacion	

Relaciones	
Fuerte	●
Moderado	○
Débil	▽

Dirección de mejora	
Maximizar	▲
Objetivo	◇
Minimizar	▼



N° de Columna			1	2	3	4	5	6	7						
Dirección de Mejora										EVALUACIÓN COMPETITIVA DEL CLIENTE					
N° de Fila	Tabla de Peso	Peso Relativo	Importancia del Cliente	Relación Máxima	REQUERIMIENTOS DEL CLIENTE (Explícitos e Implícitos) / REQUERIMIENTOS FUNCIONALES	Algoritmo de detección de vehículos.	Capacidad de generar datos estadísticos de flujo de vehículos	Visión por computador y Base de datos	Capacidad de almacenamiento	Manejo de aplicación móvil	Interfaz amigable con el usuario	Sistema inalámbrico para transmisión de datos	**Parqueadero automático**	Parqueadero inteligente modelo TGW-003	Parqueadero inteligente modelo TGW007-1	0 1 2 3 4 5	N° de Fila
1		17%	5	9	Facilidad de detección de vehículos	●	○	●	▽				5	4	4		1
2		17%	5	9	Conteo y reporte de datos de vehículos	●	●	●	○	▽		▽	5	5	5		2
3		14%	4	9	Almacenamiento de placas en DataBase	○	○	●	○	▽		●	4	4	0		3
4		3%	1	9	Almacenamiento de datos en disco externo.				●				3	1	2		4
5		14%	4	9	Gestión de usuario en aplicación móvil					●	○	▽	5	0	0		5
6		17%	5	9	Fácil de utilizar					○	●		5	3	3		6
7		17%	5	9	Evitar el uso excesivo de cable para la instalación	▽						●	5	1	1		7
					Objetivos Para el Requisito Funcional	Algoritmos de visión por computador 2	Disponibilidad de parqueadero 4	Procesamiento mediante una placa computadora 1	Disco externo	Plataforma de desarrollo de aplicación móvil	Fácil de utilizar	Transmisión inalámbrica 3					
					Relación Máxima	9	9	9	9	9	9	9					
					Clasificación de Importancia Técnica	368,97	248,28	434,48	141,38	206,9	196,55	310,34					
					Peso Relativo	19%	13%	23%	7%	11%	10%	16%					
					Tabla de Peso												
EVALUACIÓN TÉCNICA COMPETITIVA					Parqueadero automático	5	5	5	5	5	4	5					
					Parqueadero inteligente modelo TGW-003	4	4	4	0	0	3	0					
					Parqueadero inteligente modelo TGW007-1	4	5	4	0	0	4	0					
					0 1 2 3 4 5												
					N° de Columna	1	2	3	4	5	6	7					

Anexo 2

Configuración de entrenamiento (faster_rcnn_resnet101_coco.config)

```
item {
  id: 1
  name: 'Automóvil'
}
model {
  faster_rcnn {
    num_classes: 1 (Aquí ponemos el número de objetos a detectar)
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }    }
train_config: {
  batch_size: 1
  gradient_clipping_by_norm: 10.0
  fine_tune_checkpoint: "modelo/model.ckpt"
  from_detection_checkpoint: true
  num_steps: 200000
  data_augmentation_options {
    random_horizontal_flip {
    } }}
train_input_reader: {
  tf_record_input_reader {
    input_path: "TFRecords/entrenamiento.record"
  }
  label_map_path: "configuracion/label_map.pbtxt"
}
eval_input_reader: {
  tf_record_input_reader {
```

```
  input_path: "TFRecords/test.record"
 }
 label_map_path: "configuracion/label_map.pbtxt"
 shuffle: false
 num_readers: 1
 num_epochs: 1
}
```

Anexo 3

Real time vehicle detection in Python

```
while True:
ret, frames = cap.read()                    # leer fotogramas de un video
gray = cv2.cvtColor(frames, cv2.COLOR_BGR2GRAY) #convertir a escala de grises
cars = car_cascade.detectMultiScale( gray, 1.1, 1) # detector vehículos dentro de la imagen
for (x,y,w,h) in cars:     # Dibujar un rectángulo en cada carro
cv2.rectangle(frames,(x,y),(x+w,y+h),(0,0,255),2)
font = cv2.FONT_HERSHEY_DUPLEX
 cv2.putText(frames, 'Car', (x + 6, y — 6), font, 0.5, (0, 0, 255), 1)
 cv2.imshow('Car Detection', frames)    # Mostrar marcos en una ventana
 if cv2.waitKey(Q) == 13:         # Espere que la tecla Q detenga el programa
 break
```

Anexo 4

Detección de vehículo

```
cars_cascade = cv2.CascadeClassifier('haarcascade_car.xml')
cap = cv2.VideoCapture(0)
def detect_cars(frame):
   cars = cars_cascade.detectMultiScale(frame, 1.15, 4)
   for (x, y, w, h) in cars:
      plate = frame[y:y + h, x:x + w]
```

```
        cv2.rectangle(frame, (x, y), (x+w,y+h), color=(0, 255, 0), thickness=2)
        cv2.imshow('car',plate)
        cv2.imwrite("carplate.jpg", plate)
          return frame
```

Anexo 5

Aplicación de filtros en la imagen del vehículo.

```
        img = cv2.imread('carplate.jpg',cv2.IMREAD_COLOR)
        img = cv2.resize(img, (620,480) )
        placaencontrada=False
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #convierte a escala de grises
        gray = cv2.bilateralFilter(gray, 11, 17, 17) #Desenfoque para reducir el ruido
        edged = cv2.Canny(gray, 30, 200) #Realizar detección de bordes
        # encontrar contornos en la imagen de bordes, mantener solo el más grande
        cnts = cv2.findContours(edged.copy(), cv2.RETR_TREE,
    cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:10]
        screenCnt = None
```

Anexo 6

Bucle para detectar los contornos de la placa vehicular

```
    for c in cnts:
        # aproximar el contorno
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.018 * peri, True)
        # si nuestro contorno aproximado tiene cuatro puntos, entonces se ha encontrad
    nuestra placa      vehicular
        if len(approx) == 4:
          screenCnt = approx
          break
```

```
if screenCnt is None:
    detected = 0
    print ("No existe contorno de placa")
else:
    detected = 1
if detected == 1:
    cv2.drawContours(img, [screenCnt], -1, (0, 255, 0), 3)
# Enmascarar la parte que no sea la placa de matrícula
mask = np.zeros(gray.shape,np.uint8)
new_image = cv2.drawContours(mask,[screenCnt],0,255,-1,)
new_image = cv2.bitwise_and(img,img,mask=mask)
```

Anexo 7

Recortar área de interés de la placa vehicular.

```
# recortar
(x, y) = np.where(mask == 255)
(topx, topy) = (np.min(x), np.min(y))
(bottomx, bottomy) = (np.max(x), np.max(y))
Cropped = gray[topx:bottomx+1, topy:bottomy+1]
#FILTROS
ret,thresh1 = cv2.threshold(Cropped,127,255,cv2.THRESH_BINARY)
ret,thresh3 = cv2.threshold(thresh1,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(Cropped,127,255,cv2.THRESH_TOZERO)
ret, thresh5 =
cv2.threshold(Cropped,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
blur = cv2.GaussianBlur(Cropped,(5,5),0)
ret, thresh6 =
cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
#Eliminacion de ruido
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh1,cv2.MORPH_OPEN,kernel, iterations = 2)
# Encuentra el área del fondo
sure_bg = cv2.dilate(opening,kernel,iterations=3)
```

```
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)
```

Anexo 8

Leer matrícula

```
#Lee la matrícula
text = pytesseract.image_to_string(Cropped, config='--psm 11')
text2 = pytesseract.image_to_string(thresh1, config='--psm 11')
text3 = pytesseract.image_to_string(thresh3, config='--psm 11')
text4 = pytesseract.image_to_string(thresh4, config='--psm 11')
text5 = pytesseract.image_to_string(thresh5, config='--psm 11')
text6 = pytesseract.image_to_string(thresh6, config='--psm 11')
if text [0] == "{":
    text = text.replace("{", "I")
if text [0] == "/":
    text = text.replace("/", "I")
if text [0] == "1":
    text = text.replace("1", "I")
if text2 [0] == "{":
    text2 = text2.replace("{", "I")
if text2 [0] == "/":
    text2 = text2.replace("/", "I")
if text2 [0] == "1":
    text2 = text2.replace("1", "I")
if text3 [0] == "{":
    text3 = text3.replace("{", "I")
if text3 [0] == "/":
    text3 = text3.replace("/", "I")
if text3 [0] == "1":
    text3 = text3.replace("1", "I")
if text4 [0] == "{":
    text4 = text4.replace("{", "I")
```

```
if text4 [0] == "/":
    text4 = text4.replace("/", "I")
if text4 [0] == "1":
    text4 = text4.replace("1", "I")
if text5 [0] == "{":
    text5 = text5.replace("{", "I")
if text5 [0] == "/":
    text5 = text5.replace("/", "I")
if text5 [0] == "1":
    text5 = text5.replace("1", "I")
if text6 [0] == "{":
    text6 = text6.replace("{", "I")
if text6 [0] == "/":
    text6 = text6.replace("/", "I")
if text6 [0] == "1":
    text6 = text6.replace("1", "I")
```

Anexo 9

Aplicación de filtros para String.

```
#Longitud de texto por cada filtro
longitud1 = len(text)
longitud2 = len(text2)
longitud3 = len(text3)
longitud4 = len(text4)
longitud5 = len(text5)
longitud6 = len(text6)
#Procedemos a normalizar y validar cada uno de los filtros
print("Número de placa detectado es:",text)
print("1ER FILTRO")
if longitud1 > 5:
    comp11 = text [0].isalpha()
    comp12 = text [1].isalpha()
    comp13 = text [2].isalpha()
```

```
        comp14 = text [4].isdigit()
        comp15 = text [5].isdigit()
        comp16 = text [6].isdigit()
    if comp11 == True:
            print("1 parametro valido")
            if comp12 == True:
                print("2 parametro valido")
                if comp13 == True:
                    print("3 parametro valido")
                    if text [3] == "-":
                        print("Gion conseguido")
                        if comp14 == True:
                            print("4 parametro valido")
                            if comp15 == True:
                                print("5 parametro valido")
                                if comp16 == True:
                                    print("6 parametro valido")
                                    valides1 = True
                                else:
                                    print("6 parametro no valido")
                                    valides1 = False
                            else:
                                print("5 parametro no valido")
                                valides1 = False
                        else:
                            print("4 parametro no valido")
                            valides1 = False
                    else:
                        print("no es placa nacional")
                        valides1 = False
                else:
                    print("3 parametro no valido")
                    valides1 = False
            else:
```

```
            print("2 parametro no valido")
            valides1 = False
        else:
            print("1 parametro no valido")
            valides1 = False
    else:
        print("el primer filtro no es valido")
        valides1 = False
    print("Número de placa detectado 2 es:",text2)
    print("2DO FILTRO")
    if longitud2 > 5:
        comp21 = text2 [0].isalpha()
        comp22 = text2 [1].isalpha()
        comp23 = text2 [2].isalpha()
        comp24 = text2 [4].isdigit()
        comp25 = text2 [5].isdigit()
        comp26 = text2 [6].isdigit()
        if comp21 == True:
            print("1 parametro valido")
            if comp22 == True:
                print("2 parametro valido")
                if comp23 == True:
                    print("3 parametro valido")
                    if text2 [3] == "-":
                        print("Gion conseguido")
                        if comp24 == True:
                            print("4 parametro valido")
                            if comp25 == True:
                                print("5 parametro valido")
                                if comp26 == True:
                                    print("6 parametro valido")
                                    valides2 = True
                                else:
                                    print("6 parametro no valido")
```

```
                    valides2 = False
                else:
                    print("5 parametro no valido")
                    valides2 = False
            else:
                print("4 parametro no valido")
                valides2 = False
        else:
            print("no es placa nacional")
            valides2 = False
    else:
        print("3 parametro no valido")
        valides2 = False
else:
    print("2 parametro no valido")
    valides2 = False
else:
    print("1 parametro no valido")
    valides2 = False
else:
    print("el segundo filtro no es valido")
    valides2 = False
print("Número de placa detectado 3 es:",text3)
print("3RO FILTRO")
if longitud3 > 5:
    comp31 = text3 [0].isalpha()
    comp32 = text3 [1].isalpha()
    comp33 = text3 [2].isalpha()
    comp34 = text3 [4].isdigit()
    comp35 = text3 [5].isdigit()
    comp36 = text3 [6].isdigit()
    if comp31 == True:
        print("1 parametro valido")
        if comp32 == True:
```

```
            print("2 parametro valido")
            if comp33 == True:
                print("3 parametro valido")
                if text3 [3] == "-":
                    print("Gion conseguido")
                    if comp34 == True:
                        print("4 parametro valido")
                        if comp35 == True:
                            print("5 parametro valido")
                            if comp36 == True:
                                print("6 parametro valido")
                                valides3 = True
                            else:
                                print("6 parametro no valido")
                                valides3 = False
                        else:
                            print("5 parametro no valido")
                            valides3 = False
                    else:
                        print("4 parametro no valido")
                        valides3 = False
                else:
                    print("no es placa nacional")
                    valides3 = False
            else:
                print("3 parametro no valido")
                valides3 = False
        else:
            print("2 parametro no valido")
            valides3 = False
    else:
        print("1 parametro no valido")
        valides3 = False
else:
```

```
    print("el tercer filtro no es valido")
    valides3 = False
print("Número de placa detectado 4 es:",text4)
print("4TO FILTRO")
if longitud4 > 5:
    comp41 = text4 [0].isalpha()
    comp42 = text4 [1].isalpha()
    comp43 = text4 [2].isalpha()
    comp44 = text4 [4].isdigit()
    comp45 = text4 [5].isdigit()
    comp46 = text4 [6].isdigit()
    if comp41 == True:
        print("1 parametro valido")
        if comp42 == True:
            print("2 parametro valido")
            if comp43 == True:
                print("3 parametro valido")
                if text4 [3] == "-":
                    print("Gion conseguido")
                    if comp44 == True:
                        print("4 parametro valido")
                        if comp45 == True:
                            print("5 parametro valido")
                            if comp46 == True:
                                print("6 parametro valido")
                                valides4 = True
                            else:
                                print("6 parametro no valido")
                                valides4 = False
                        else:
                            print("5 parametro no valido")
                            valides4 = False
                    else:
                        print("4 parametro no valido")
```

```
                valides4 = False
            else:
                print("no es placa nacional")
                valides4 = False
        else:
            print("3 parametro no valido")
            valides4 = False
    else:
        print("2 parametro no valido")
        valides4 = False
else:
    print("1 parametro no valido")
    valides4 = False
else:
    print("el cuarto filtro no es valido")
    valides4 = False
print("Número de placa detectado 5 es:",text5)
print("5TO FILTRO")
if longitud5 > 5:
    comp51 = text5 [0].isalpha()
    comp52 = text5 [1].isalpha()
    comp53 = text5 [2].isalpha()
    comp54 = text5 [4].isdigit()
    comp55 = text5 [5].isdigit()
    comp56 = text5 [6].isdigit()
    if comp51 == True:
        print("1 parametro valido")
        if comp52 == True:
            print("2 parametro valido")
            if comp53 == True:
                print("3 parametro valido")
                if text5 [3] == "-":
                    print("Gion conseguido")
                    if comp54 == True:
```

```
                print("4 parametro valido")
                if comp55 == True:
                    print("5 parametro valido")
                    if comp56 == True:
                        print("6 parametro valido")
                        valides5 = True
                    else:
                        print("6 parametro no valido")
                        valides5 = False
                else:
                    print("5 parametro no valido")
                    valides5 = False
            else:
                print("4 parametro no valido")
                valides5 = False
        else:
            print("no es placa nacional")
            valides5 = False
    else:
        print("3 parametro no valido")
        valides5 = False
else:
    print("2 parametro no valido")
    valides5 = False
else:
    print("1 parametro no valido")
    valides5 = False
else:
    print("el quinto filtro no es valido")
    valides5 = False
print("Número de placa detectado 6 es is:",text6)
print("6TO FILTRO")
if longitud6 > 6:
    comp61 = text6 [0].isalpha()
```

```
comp62 = text6 [1].isalpha()
comp63 = text6 [2].isalpha()
comp64 = text6 [4].isdigit()
comp65 = text6 [5].isdigit()
comp66 = text6 [6].isdigit()
if comp61 == True:
    print("1 parametro valido")
    if comp62 == True:
        print("2 parametro valido")
        if comp63 == True:
            print("3 parametro valido")
            if text6 [3] == "-":
                print("Gion conseguido")
                if comp64 == True:
                    print("4 parametro valido")
                    if comp65 == True:
                        print("5 parametro valido")
                        if comp66 == True:
                            print("6 parametro valido")
                            valides6 = True
                        else:
                            print("6 parametro no valido")
                            valides6 = False
                    else:
                        print("5 parametro no valido")
                        valides6 = False
                else:
                    print("4 parametro no valido")
                    valides6 = False
            else:
                print("no es placa nacional")
                valides6 = False
        else:
            print("3 parametro no valido")
```

```
                valides6 = False
            else:
                print("2 parametro no valido")
                valides6 = False
        else:
            print("1 parametro no valido")
            valides6 = False
    else:
        print("el sexto filtro no es valido")
        valides6 = False
    if valides1 == True:
    comprobacion()
        #resultado1
        if coincidencia > 1:
            print("El resultado con un porcentaje de:",(coincidencia+1*100/6))
            print("es:",text)
            valides2 = False
            valides3 = False
            valides4 = False
            valides5 = False
            valides6 = False
            placaencontrada = True
            recordTuple = (text)
    if valides2 == True:
        comprobacion2()
        #resultado2
if coincidencia2 >= 1:
            print("El resultado con un porcentaje de:",(coincidencia2+1)*100/6)
            print("es:",text2)
            valides3 = False
            valides4 = False
            valides5 = False
            valides6 = False
            placaencontrada = True
```

```
        recordTuple = (text2)
if valides3 == True:
  comprobacion3()
   #resultado3
   if coincidencia3 >= 1:
        print("El resultado con un porcentaje de:",(coincidencia3+1)*100/6)
        print("es:",text3)
        valides3 = False
        valides4 = False
        valides5 = False
        valides6 = False
        placaencontrada = True
        recordTuple = (text3)
if valides4 == True:
   comprobacion4()
   #resultado4
 if coincidencia4 >= 1:
        print("El resultado con un porcentaje de:",(coincidencia4+1)*100/6)
        print("es:",text4)
        valides4 = False
        valides5 = False
        valides6 = False
        placaencontrada = True
        recordTuple = (text4)
if valides5 == True:
   comprobacion5()
   #resultado4
     if coincidencia5 >= 1:
        print("El resultado con un porcentaje de:",(coincidencia5+1)*100/6)
        print("es:",text5)
        valides6 = False
        placaencontrada = True
        recordTuple = (text5)
if valides6 == True:
```

```
        comprobacion6()
        #resultado4
            if coincidencia6 >= 1:
            print("El resultado con un porcentaje de:",(coincidencia6+1)*100/6)
            print("es:",text6)
            placaencontrada = True
            recordTuple = (text6)
    if valides6 == True:
            print("NECESITAMOS OTRA FOTO")
    #print("l1:",text4 [0])
    #print("l2:",text4 [1])
    #print("l3:",text4 [2])
    #print("l4:",text4 [3])
    #print("l5:",text4 [4])
    #print("l6:",text4 [5])
    #print("l7:",text4 [6])
    #print("l8:",text4 [7])
    if placaencontrada == True:
        print("fotografia si")
webbrowser.open_new("https://parqueaderoapp.000webhostapp.com/parqueadero/agregar.php?placa="+ recordTuple)
    else:
        print("fotografia no")
    cv2.imshow('image',img)
    cv2.imshow('Cropped',Cropped)
    cv2.imshow('img_binary',thresh1)
    cv2.imshow('img_trunc',thresh3)
    cv2.imshow('tozero',thresh4)
    cv2.imshow('binaryinv',thresh5)
    cv2.imshow('Otsus Binarization',thresh6)
    print("tesis")
    return
#Comparamos los validos
    def comprobacion():
```

```
    global coincidencia
    coincidencia =0
    busca = text.find(text2)
    if busca == 0:
        coincidencia = coincidencia+1
    busca = text.find(text3)
    if busca == 0:
        coincidencia = coincidencia+1
    busca = text.find(text4)
    if busca == 0:
        coincidencia = coincidencia+1
    busca = text.find(text5)
    if busca == 0:
        coincidencia = coincidencia+1
    busca = text.find(text6)
    if busca == 0:
        coincidencia = coincidencia+1
    return
def comprobacion2():
    global coincidencia2
    coincidencia2 =0
    busca = text2.find(text)
    if busca == 0:
        coincidencia2 = coincidencia2+1
    busca = text2.find(text3)
    if busca == 0:
        coincidencia2 = coincidencia2+1
    busca = text2.find(text4)
    if busca == 0:
        coincidencia2 = coincidencia2+1
    busca = text2.find(text5)
    if busca == 0:
        coincidencia2 = coincidencia2+1
    busca = text2.find(text6)
```

```
    if busca == 0:
        coincidencia2 = coincidencia2+1
    return
def comprobacion3():
    global coincidencia3
    coincidencia3=0
    busca = text3.find(text)
    if busca == 0:
        coincidencia3 = coincidencia3+1
    busca = text3.find(text2)
    if busca == 0:
        coincidencia3 = coincidencia3+1
    busca = text3.find(text4)
    if busca == 0:
        coincidencia3 = coincidencia3+1
    busca = text3.find(text5)
    if busca == 0:
        coincidencia3 = coincidencia3+1
    busca = text3.find(text6)
    if busca == 0:
        coincidencia3 = coincidencia3+1
    print(coincidencia3)
    return
def comprobacion4():
    global coincidencia4
    coincidencia4=0
    busca = text4.find(text)
    if busca == 0:
        coincidencia4 = coincidencia4+1
    busca = text4.find(text2)
    if busca == 0:
        coincidencia4 = coincidencia4+1
    busca = text4.find(text3)
    if busca == 0:
```

```
        coincidencia4 = coincidencia4+1
    busca = text4.find(text5)
    if busca == 0:
        coincidencia4 = coincidencia4+1
    busca = text4.find(text6)
    if busca == 0:
        coincidencia4 = coincidencia4+1
     return
def comprobacion5():
    global coincidencia5
    coincidencia5=0
    busca = text5.find(text)
    if busca == 0:
        coincidencia5 = coincidencia5+1
    busca = text5.find(text2)
    if busca == 0:
        coincidencia5 = coincidencia5+1
    busca = text5.find(text3)
    if busca == 0:
        coincidencia5 = coincidencia5+1
    busca = text5.find(text4)
    if busca == 0:
        coincidencia5 = coincidencia5+1
    busca = text5.find(text6)
    if busca == 0:
        coincidencia5 = coincidencia5+1
 return
def comprobacion6():
    global coincidencia6
    coincidencia6=0
    busca = text6.find(text)
    if busca == 0:
        coincidencia6 = coincidencia6+1
    busca = text6.find(text2)
```

```
    if busca == 0:
        coincidencia6 = coincidencia6+1
    busca = text6.find(text3)
    if busca == 0:
        coincidencia6 = coincidencia6+1
    busca = text6.find(text4)
    if busca == 0:
        coincidencia6 = coincidencia6+1
    busca = text6.find(text5)
    if busca == 0:
        coincidencia6 = coincidencia6+1
    return
```

Anexo 10

Código completo del sistema de reconocimiento de placas vehiculares.

#Detección de vehículo

```
cars_cascade = cv2.CascadeClassifier('haarcascade_car.xml')
cap = cv2.VideoCapture(0)
def detect_cars(frame):
    cars = cars_cascade.detectMultiScale(frame, 1.15, 4)
    for (x, y, w, h) in cars:
        plate = frame[y:y + h, x:x + w]
        cv2.rectangle(frame, (x, y), (x+w,y+h), color=(0, 255, 0), thickness=2)
        cv2.imshow('car',plate)
        cv2.imwrite("carplate.jpg", plate)
            return frame
```

#Aplicación de filtros en la imagen del vehículo.

```
    img = cv2.imread('carplate.jpg',cv2.IMREAD_COLOR)
    img = cv2.resize(img, (620,480) )
    placaencontrada=False
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #convierte a escala de grises
    gray = cv2.bilateralFilter(gray, 11, 17, 17) #Desenfoque para reducir el ruido
```

```
    edged = cv2.Canny(gray, 30, 200) #Realizar detección de bordes
    # encontrar contornos en la imagen de bordes, mantener solo el más grande
    cnts = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:10]
    screenCnt = None
#Bucle para detectar los contornos de la placa vehicular
    for c in cnts:
        # aproximar el contorno
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, 0.018 * peri, True)
        # si nuestro contorno aproximado tiene cuatro puntos, entonces se ha encontrad nuestra placa vehicular
        if len(approx) == 4:
            screenCnt = approx
            break
    if screenCnt is None:
        detected = 0
        print ("No existe contorno de placa")
    else:
        detected = 1
    if detected == 1:
        cv2.drawContours(img, [screenCnt], -1, (0, 255, 0), 3)
    # Enmascarar la parte que no sea la placa de matrícula
    mask = np.zeros(gray.shape,np.uint8)
    new_image = cv2.drawContours(mask,[screenCnt],0,255,-1,)
    new_image = cv2.bitwise_and(img,img,mask=mask)
#Recortar área de interés de la placa vehicular.
    # recortar
    (x, y) = np.where(mask == 255)
    (topx, topy) = (np.min(x), np.min(y))
    (bottomx, bottomy) = (np.max(x), np.max(y))
```

```
            Cropped = gray[topx:bottomx+1, topy:bottomy+1]
            #FILTROS
            ret,thresh1 = cv2.threshold(Cropped,127,255,cv2.THRESH_BINARY)
            ret,thresh3 = cv2.threshold(thresh1,127,255,cv2.THRESH_TRUNC)
            ret,thresh4 = cv2.threshold(Cropped,127,255,cv2.THRESH_TOZERO)
            ret, thresh5 =
        cv2.threshold(Cropped,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
            blur = cv2.GaussianBlur(Cropped,(5,5),0)
            ret, thresh6 =
        cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
            #Eliminacion de ruido
            kernel = np.ones((3,3),np.uint8)
            opening = cv2.morphologyEx(thresh1,cv2.MORPH_OPEN,kernel, iterations = 2)
            # Encuentra el área del fondo
            sure_bg = cv2.dilate(opening,kernel,iterations=3)
           dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
            ret, sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)
#Lee la matrícula
            text = pytesseract.image_to_string(Cropped, config='--psm 11')
            text2 = pytesseract.image_to_string(thresh1, config='--psm 11')
            text3 = pytesseract.image_to_string(thresh3, config='--psm 11')
            text4 = pytesseract.image_to_string(thresh4, config='--psm 11')
            text5 = pytesseract.image_to_string(thresh5, config='--psm 11')
            text6 = pytesseract.image_to_string(thresh6, config='--psm 11')
            if text [0] == "{":
                text = text.replace("{", "I")
            if text [0] == "/":
                text = text.replace("/", "I")
           if text [0] == "1":
                text = text.replace("1", "I")
           if text2 [0] == "{":
                text2 = text2.replace("{", "I")
           if text2 [0] == "/":
```

```
        text2 = text2.replace("/", "I")
    if text2 [0] == "1":
        text2 = text2.replace("1", "I")
    if text3 [0] == "{":
        text3 = text3.replace("{", "I")
    if text3 [0] == "/":
        text3 = text3.replace("/", "I")
    if text3 [0] == "1":
        text3 = text3.replace("1", "I")
   if text4 [0] == "{":
        text4 = text4.replace("{", "I")
    if text4 [0] == "/":
        text4 = text4.replace("/", "I")
    if text4 [0] == "1":
        text4 = text4.replace("1", "I")
    if text5 [0] == "{":
        text5 = text5.replace("{", "I")
    if text5 [0] == "/":
        text5 = text5.replace("/", "I")
    if text5 [0] == "1":
        text5 = text5.replace("1", "I")
    if text6 [0] == "{":
        text6 = text6.replace("{", "I")
    if text6 [0] == "/":
        text6 = text6.replace("/", "I")
    if text6 [0] == "1":
        text6 = text6.replace("1", "I")
#Aplicación de filtros para String.
            #Longitud de texto por cada filtro
        longitud1 = len(text)
        longitud2 = len(text2)
        longitud3 = len(text3)
```

```
    longitud4 = len(text4)
    longitud5 = len(text5)
    longitud6 = len(text6)
    #Procedemos a normalizar y validar cada uno de los filtros
    print("Número de placa detectado es:",text)
    print("1ER FILTRO")
    if longitud1 > 5:
        comp11 = text [0].isalpha()
        comp12 = text [1].isalpha()
        comp13 = text [2].isalpha()
        comp14 = text [4].isdigit()
        comp15 = text [5].isdigit()
        comp16 = text [6].isdigit()
if comp11 == True:
            print("1 parametro valido")
            if comp12 == True:
                print("2 parametro valido")
                if comp13 == True:
                    print("3 parametro valido")
                    if text [3] == "-":
                        print("Gion conseguido")
                        if comp14 == True:
                            print("4 parametro valido")
                            if comp15 == True:
                                print("5 parametro valido")
                                if comp16 == True:
                                    print("6 parametro valido")
                                    valides1 = True
                                else:
                                    print("6 parametro no valido")
                                    valides1 = False
                            else:
                                print("5 parametro no valido")
                                valides1 = False
```

```
                else:
                    print("4 parametro no valido")
                    valides1 = False
            else:
                print("no es placa nacional")
                valides1 = False
        else:
            print("3 parametro no valido")
            valides1 = False
    else:
        print("2 parametro no valido")
        valides1 = False
else:
    print("1 parametro no valido")
    valides1 = False
else:
    print("el primer filtro no es valido")
    valides1 = False
print("Número de placa detectado 2 es:",text2)
print("2DO FILTRO")
if longitud2 > 5:
    comp21 = text2 [0].isalpha()
    comp22 = text2 [1].isalpha()
    comp23 = text2 [2].isalpha()
    comp24 = text2 [4].isdigit()
    comp25 = text2 [5].isdigit()
    comp26 = text2 [6].isdigit()
    if comp21 == True:
        print("1 parametro valido")
        if comp22 == True:
            print("2 parametro valido")
            if comp23 == True:
                print("3 parametro valido")
                if text2 [3] == "-":
```

```
                print("Gion conseguido")
                if comp24 == True:
                    print("4 parametro valido")
                    if comp25 == True:
                        print("5 parametro valido")
                        if comp26 == True:
                            print("6 parametro valido")
                            valides2 = True
                        else:
                            print("6 parametro no valido")
                            valides2 = False
                    else:
                        print("5 parametro no valido")
                        valides2 = False
                else:
                    print("4 parametro no valido")
                    valides2 = False
            else:
                print("no es placa nacional")
                valides2 = False
        else:
            print("3 parametro no valido")
            valides2 = False
    else:
        print("2 parametro no valido")
        valides2 = False
else:
    print("1 parametro no valido")
    valides2 = False
else:
    print("el segundo filtro no es valido")
    valides2 = False
print("Número de placa detectado 3 es:",text3)
print("3RO FILTRO")
```

```
if longitud3 > 5:
    comp31 = text3 [0].isalpha()
    comp32 = text3 [1].isalpha()
    comp33 = text3 [2].isalpha()
    comp34 = text3 [4].isdigit()
    comp35 = text3 [5].isdigit()
    comp36 = text3 [6].isdigit()
    if comp31 == True:
        print("1 parametro valido")
        if comp32 == True:
            print("2 parametro valido")
            if comp33 == True:
                print("3 parametro valido")
                if text3 [3] == "-":
                    print("Gion conseguido")
                    if comp34 == True:
                        print("4 parametro valido")
                        if comp35 == True:
                            print("5 parametro valido")
                            if comp36 == True:
                                print("6 parametro valido")
                                valides3 = True
                            else:
                                print("6 parametro no valido")
                                valides3 = False
                        else:
                            print("5 parametro no valido")
                            valides3 = False
                    else:
                        print("4 parametro no valido")
                        valides3 = False
                else:
                    print("no es placa nacional")
                    valides3 = False
```

```
            else:
                print("3 parametro no valido")
                valides3 = False
        else:
            print("2 parametro no valido")
            valides3 = False
    else:
        print("1 parametro no valido")
        valides3 = False
else:
    print("el tercer filtro no es valido")
    valides3 = False
print("Número de placa detectado 4 es:",text4)
print("4TO FILTRO")
if longitud4 > 5:
    comp41 = text4 [0].isalpha()
    comp42 = text4 [1].isalpha()
    comp43 = text4 [2].isalpha()
    comp44 = text4 [4].isdigit()
    comp45 = text4 [5].isdigit()
    comp46 = text4 [6].isdigit()
    if comp41 == True:
        print("1 parametro valido")
        if comp42 == True:
            print("2 parametro valido")
            if comp43 == True:
                print("3 parametro valido")
                if text4 [3] == "-":
                    print("Gion conseguido")
                    if comp44 == True:
                        print("4 parametro valido")
                        if comp45 == True:
                            print("5 parametro valido")
                            if comp46 == True:
```

```
                        print("6 parametro valido")
                        valides4 = True
                    else:
                        print("6 parametro no valido")
                        valides4 = False
                else:
                    print("5 parametro no valido")
                    valides4 = False
            else:
                print("4 parametro no valido")
                valides4 = False
        else:
            print("no es placa nacional")
            valides4 = False
    else:
        print("3 parametro no valido")
        valides4 = False
else:
    print("2 parametro no valido")
    valides4 = False
else:
    print("1 parametro no valido")
    valides4 = False
else:
    print("el cuarto filtro no es valido")
    valides4 = False
print("Número de placa detectado 5 es:",text5)
print("5TO FILTRO")
if longitud5 > 5:
    comp51 = text5 [0].isalpha()
    comp52 = text5 [1].isalpha()
    comp53 = text5 [2].isalpha()
    comp54 = text5 [4].isdigit()
    comp55 = text5 [5].isdigit()
```

```
comp56 = text5 [6].isdigit()
if comp51 == True:
    print("1 parametro valido")
    if comp52 == True:
        print("2 parametro valido")
        if comp53 == True:
            print("3 parametro valido")
            if text5 [3] == "-":
                print("Gion conseguido")
                if comp54 == True:
                    print("4 parametro valido")
                    if comp55 == True:
                        print("5 parametro valido")
                        if comp56 == True:
                            print("6 parametro valido")
                            valides5 = True
                        else:
                            print("6 parametro no valido")
                            valides5 = False
                    else:
                        print("5 parametro no valido")
                        valides5 = False
                else:
                    print("4 parametro no valido")
                    valides5 = False
            else:
                print("no es placa nacional")
                valides5 = False
        else:
            print("3 parametro no valido")
            valides5 = False
    else:
        print("2 parametro no valido")
        valides5 = False
```

```
    else:
        print("1 parametro no valido")
        valides5 = False
else:
    print("el quinto filtro no es valido")
    valides5 = False
print("Número de placa detectado 6 es is:",text6)
print("6TO FILTRO")
if longitud6 > 6:
    comp61 = text6 [0].isalpha()
    comp62 = text6 [1].isalpha()
    comp63 = text6 [2].isalpha()
    comp64 = text6 [4].isdigit()
    comp65 = text6 [5].isdigit()
    comp66 = text6 [6].isdigit()
    if comp61 == True:
        print("1 parametro valido")
        if comp62 == True:
            print("2 parametro valido")
            if comp63 == True:
                print("3 parametro valido")
                if text6 [3] == "-":
                    print("Gion conseguido")
                    if comp64 == True:
                        print("4 parametro valido")
                        if comp65 == True:
                            print("5 parametro valido")
                            if comp66 == True:
                                print("6 parametro valido")
                                valides6 = True
                            else:
                                print("6 parametro no valido")
                                valides6 = False
                        else:
```

```
                    print("5 parametro no valido")
                    valides6 = False
                else:
                    print("4 parametro no valido")
                    valides6 = False
            else:
                print("no es placa nacional")
                valides6 = False
        else:
            print("3 parametro no valido")
            valides6 = False
    else:
        print("2 parametro no valido")
        valides6 = False
else:
    print("1 parametro no valido")
    valides6 = False
else:
    print("el sexto filtro no es valido")
    valides6 = False
if valides1 == True:
comprobacion()
    #resultado1
    if coincidencia > 1:
        print("El resultado con un porcentaje de:",(coincidencia+1*100/6))
        print("es:",text)
        valides2 = False
        valides3 = False
        valides4 = False
        valides5 = False
        valides6 = False
        placaencontrada = True
        recordTuple = (text)
if valides2 == True:
```

```
        comprobacion2()
        #resultado2
if coincidencia2 >= 1:
            print("El resultado con un porcentaje de:",(coincidencia2+1)*100/6)
            print("es:",text2)
            valides3 = False
            valides4 = False
            valides5 = False
            valides6 = False
            placaencontrada = True
            recordTuple = (text2)
    if valides3 == True:
      comprobacion3()
        #resultado3
        if coincidencia3 >= 1:
            print("El resultado con un porcentaje de:",(coincidencia3+1)*100/6)
            print("es:",text3)
            valides3 = False
            valides4 = False
            valides5 = False
            valides6 = False
            placaencontrada = True
            recordTuple = (text3)
    if valides4 == True:
        comprobacion4()
        #resultado4
     if coincidencia4 >= 1:
            print("El resultado con un porcentaje de:",(coincidencia4+1)*100/6)
            print("es:",text4)
            valides4 = False
            valides5 = False
            valides6 = False
            placaencontrada = True
            recordTuple = (text4)
```

```
    if valides5 == True:
        comprobacion5()
        #resultado4
          if coincidencia5 >= 1:
            print("El resultado con un porcentaje de:",(coincidencia5+1)*100/6)
            print("es:",text5)
            valides6 = False
            placaencontrada = True
            recordTuple = (text5)
    if valides6 == True:
        comprobacion6()
        #resultado4
            if coincidencia6 >= 1:
            print("El resultado con un porcentaje de:",(coincidencia6+1)*100/6)
            print("es:",text6)
            placaencontrada = True
            recordTuple = (text6)
  if valides6 == True:
            print("NECESITAMOS OTRA FOTO")
    #print("l1:",text4 [0])
    #print("l2:",text4 [1])
    #print("l3:",text4 [2])
    #print("l4:",text4 [3])
    #print("l5:",text4 [4])
    #print("l6:",text4 [5])
    #print("l7:",text4 [6])
    #print("l8:",text4 [7])
    if placaencontrada == True:
        print("fotografia si")
webbrowser.open_new("https://parqueaderoapp.000webhostapp.com/parqueadero/agregar.php?placa="+ recordTuple)
    else:
        print("fotografia no")
    cv2.imshow('image',img)
```

```
    cv2.imshow('Cropped',Cropped)
    cv2.imshow('img_binary',thresh1)
    cv2.imshow('img_trunc',thresh3)
    cv2.imshow('tozero',thresh4)
    cv2.imshow('binaryinv',thresh5)
    cv2.imshow('Otsus Binarization',thresh6)
    print("tesis")
    return
#Comparamos los validos
    def comprobacion():
        global coincidencia
        coincidencia =0
        busca = text.find(text2)
        if busca == 0:
            coincidencia = coincidencia+1
        busca = text.find(text3)
        if busca == 0:
            coincidencia = coincidencia+1
        busca = text.find(text4)
        if busca == 0:
            coincidencia = coincidencia+1
        busca = text.find(text5)
        if busca == 0:
            coincidencia = coincidencia+1
        busca = text.find(text6)
        if busca == 0:
            coincidencia = coincidencia+1
        return
    def comprobacion2():
        global coincidencia2
        coincidencia2 =0
        busca = text2.find(text)
        if busca == 0:
            coincidencia2 = coincidencia2+1
```

```
    busca = text2.find(text3)
    if busca == 0:
        coincidencia2 = coincidencia2+1
    busca = text2.find(text4)
    if busca == 0:
        coincidencia2 = coincidencia2+1
    busca = text2.find(text5)
    if busca == 0:
        coincidencia2 = coincidencia2+1
    busca = text2.find(text6)
    if busca == 0:
        coincidencia2 = coincidencia2+1
    return
def comprobacion3():
    global coincidencia3
    coincidencia3=0
    busca = text3.find(text)
    if busca == 0:
        coincidencia3 = coincidencia3+1
    busca = text3.find(text2)
    if busca == 0:
        coincidencia3 = coincidencia3+1
    busca = text3.find(text4)
    if busca == 0:
        coincidencia3 = coincidencia3+1
    busca = text3.find(text5)
    if busca == 0:
        coincidencia3 = coincidencia3+1
    busca = text3.find(text6)
    if busca == 0:
        coincidencia3 = coincidencia3+1
    print(coincidencia3)
    return
def comprobacion4():
```

```
    global coincidencia4
    coincidencia4=0
    busca = text4.find(text)
    if busca == 0:
        coincidencia4 = coincidencia4+1
    busca = text4.find(text2)
    if busca == 0:
        coincidencia4 = coincidencia4+1
    busca = text4.find(text3)
    if busca == 0:
        coincidencia4 = coincidencia4+1
    busca = text4.find(text5)
    if busca == 0:
        coincidencia4 = coincidencia4+1
    busca = text4.find(text6)
    if busca == 0:
        coincidencia4 = coincidencia4+1
     return
def comprobacion5():
    global coincidencia5
    coincidencia5=0
    busca = text5.find(text)
    if busca == 0:
        coincidencia5 = coincidencia5+1
    busca = text5.find(text2)
    if busca == 0:
        coincidencia5 = coincidencia5+1
    busca = text5.find(text3)
    if busca == 0:
        coincidencia5 = coincidencia5+1
    busca = text5.find(text4)
    if busca == 0:
        coincidencia5 = coincidencia5+1
    busca = text5.find(text6)
```

```
    if busca == 0:
        coincidencia5 = coincidencia5+1
 return
def comprobacion6():
    global coincidencia6
    coincidencia6=0
    busca = text6.find(text)
    if busca == 0:
        coincidencia6 = coincidencia6+1
    busca = text6.find(text2)
    if busca == 0:
        coincidencia6 = coincidencia6+1
    busca = text6.find(text3)
    if busca == 0:
        coincidencia6 = coincidencia6+1
    busca = text6.find(text4)
    if busca == 0:
        coincidencia6 = coincidencia6+1
    busca = text6.find(text5)
    if busca == 0:
        coincidencia6 = coincidencia6+1
    return
```