Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería Mecánica, Mecánica Eléctrica y Mecatrónica

"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE IDENTIFICACIÓN DE VEHÍCULOS UTILIZANDO REDES NEURONALES ARTIFICIALES, APLICADO A LA UNIVERSIDAD CATÓLICA DE SANTA MARÍA"

Tesis presentada por el bachiller:

Pinto Vizcarra, Christian Alexander

Para optar por el título profesional de:

Ingeniero Mecatrónico

Asesor:

Ing. Mestas Ramos, Sergio

Arequipa – Perú

2019

Universidad Católica de Santa María

(51 54) 382038 Fax:(51 54) 251213 ucsm@ucsm.edu.pe http://www.ucsm.edu.pe Apartado:1350

AREQUIPA - PERÚ

ESCUELA PROFESIONAL DE INGENIERÍA MECANICA, MECÁNICA ELÉCTRICA Y MECATRÓNICA

INFORME DICTAMINATORIO

V I S T O

EL BORRADOR DE TESIS TITULADO:

"DISEÑO E IMPLEMENTACION DE UN SISTEMA DE IDENTIFICACION DE VEHICULOS UTILIZANDO REDES NEURONALES ARTIFICIALES APLICADO A LA UNIVERSIDAD CATOLICA DE SANTA MARIA"

Presentado por el Bachiller:

PINTO VIZCARRA, Christian Alexander

Nuestro **DICTAMEN** es:

Favorable

OBSERVACIONES:

Ninguna

Arequipa, 26 / Junio 2019

ING. SERGIO MESTAS RAMOS

ING. MARCELO QUISPE CCACHUCO

AGRADECIMIENTOS

Agradezco a Dios por guiarme durante mi vida y estar presente en cada paso, ayudándome a soportar cualquier adversidad y mirar siempre hacia adelante, otorgando su sabiduría en las decisiones tomadas.

Agradezco a mis padres Jorge y Doris, porque me criaron para ser un hombre de bien y me apoyaron a lo largo del camino, con sus consejos y paciencia para sobrellevar los obstáculos. Fueron un ejemplo de aprendizaje constante.

Agradezco a mi hermano Diego, que estuvo siempre ahí para ayudar en cualquier necesidad, además de darme ánimos para continuar y no rendirme.

Agradezco a mi profesor Sergio Mestas, que me ayudó y me aconsejó durante todo el proceso, prestándome su tiempo y conocimientos para desarrollar y aplicar lo aprendido durante mi vida universitaria.

Estaré agradecido por las personas que no menciono aquí, pero que prestaron su ayuda para lograr el desarrollo de este proyecto.

RESUMEN

El presente trabajo de investigación consiste en el diseño e implementación de un sistema de identificación vehicular aplicable a la Universidad Católica Santa María, ubicada en el distrito de Yanahuara de la provincia de Arequipa.

El objetivo principal es integrar en un software 2 metodologías importantes en la actualidad, como son el procesamiento digital de imágenes (PDI) y el reconocimiento óptico de caracteres (OCR), con el fin de contribuir con una alternativa de solución a un problema álgico de nuestra sociedad: la seguridad.

De esta manera, se realizó un estudio de técnicas de programación, obteniéndose como resultado el procesamiento morfológico para la detección de placas vehiculares, con resultados de 91% de acierto. De este porcentaje se pasa a la identificación por medio del entrenamiento de RNA, con resultados de 100% y tiempos de respuesta de 1.2 segundos, los cuales son mejores que con el método convencional: coeficiente de correlación de Pearson (86%), y que tiene tiempos promedio de 5.5 segundos.

Este software es aplicable a cualquier cámara con resolución mínima de 480p, sin necesidad de hardware con costos adicionales, lo que lo hace una alternativa más económica y versátil respecto a otras del mercado.

Palabras Clave: procesamiento digital de imágenes, reconocimiento óptico de caracteres, red neuronal artificial, procesado morfológico.

ABSTRACT

This research work consists in the design and implementation of a vehicle identification system applicable to the Santa María Catholic University, located in the Yanahuara district of the province of Arequipa.

The main objective is to integrate into today's software 2 important methodologies, such as digital image processing (PDI) and optical character recognition (OCR), in order to contribute with an alternative solution to an allergic problem of Our society: security.

In this way, a study of programming techniques was carried out, obtaining as a result the morphological processing for the detection of vehicle plates, with results of 91% of success. This percentage is passed to the identification through RNA training, with results of 100% and response times of 1.2 seconds, which are better than with the conventional method: Pearson's correlation coefficient (86%), and that It has average times of 5.5 seconds.

This software is applicable to any camera with a minimum resolution of 480p, without the need for additional cost hardware, which makes it a more economical and versatile alternative compared to others in the market.

Keywords: digital image processing, optical character recognition, artificial neural network, morphological processing.

INTRODUCCION

El desarrollo profesional de un ingeniero se basa en el uso de dos facultades importantes: “el ingenio y la creatividad” que, combinados con el conocimiento y experiencia adquiridos en la Universidad, llevan a tener nuevas perspectivas del mundo y el contexto industrial que lo rodea. Esto lo lleva a crear soluciones o mejorar tecnologías para satisfacer necesidades específicas de aplicación para uno o varios usuarios.

Hoy en día, un campo que ha adquirido mayor importancia en nuestra sociedad es la video vigilancia, un sistema de seguridad utilizada en instituciones públicas y privadas, con el fin de buscar la tranquilidad del usuario respecto a su propiedad en cualquier circunstancia. Una aplicación de este sistema es el acceso vehicular, realizado manualmente por un personal de seguridad, con la ayuda de cámaras que registran los ingresos y salidas a las instituciones.

Siguiendo este contexto, se tienen herramientas como los sistemas de reconocimiento inteligente, que permiten actualmente obtener una mayor cantidad de características a partir de imágenes, además de la comunicación con bases de datos de usuarios, sin necesidad de la interacción de una persona.

El presente proyecto busca aprovechar algunas de estas tecnologías, representadas por las técnicas de procesamiento de imagen y de diseño de redes neuronales artificiales, creando un software compatible con las cámaras de seguridad presentes en la institución. La integración de ambos componentes creará un sistema que detecte el vehículo, localice su placa y permita la obtención de sus caracteres junto con información relacionada a una base de datos del usuario.

INDICE DE CONTENIDO

INDICE DE TABLAS

INDICE DE GRÁFICOS Y FIGURAS

INDICE DE ABREVIATURAS Y SIMBOLOS

LISTA DE ABREVIATURAS

CMYK: Cyan-Magenta-Yellow-Key

DHCP: Dynamic Host Configuration Protocol (Protocolo de Configuración de Anfitrión Dinámico

EE: Elemento Estructurante

EP: Elemento de Proceso

GUI: Graphic User Interface (Interfaz Gráfica de Usuario)

IA: Inteligencia Artificial

OCR: Optical Character Recognition (Reconocimiento Optico de Caracteres)

PDI: Procesamiento Digital de Imágenes

POE: Power Over Ethernet (Alimentación a través de Ethernet)

RGB: Red-Green-Blue

RNA: Redes Neuronales Artificiales

UCSM: Universidad Católica de Santa María

LISTA DE SIMBOLOS

px: pixel

seg.: segundos

CAPITULO I: MARCO METODOLOGICO

1. Tema de Investigación

"Diseño e implementación de un sistema de identificación de vehículos utilizando redes neuronales artificiales, aplicado a la Universidad Católica de Santa María".

¿Cómo se puede implementar un sistema de identificación vehículos en la UCSM y qué ventajas ofrece el uso de redes neuronales artificiales (RNA) en el diseño de este sistema?

2. Hipótesis

Es posible diseñar un sistema de identificación de vehículos en tiempo real aplicable al ingreso de vehículos mayores (autos, camionetas, minivan, etc.) en la Universidad Católica de Santa María, el cual sea independiente de un software y hardware de coste muy elevado, que además incluya tecnologías modernas como las RNA, cuyos tiempos de mejorar los porcentajes de aciertos con tiempos de respuesta más pequeños.

3. Objetivos

3.1. Objetivo General

- Desarrollar e implementar un sistema que realice la identificación de placas vehiculares de para el acceso de personal y validarlo en el ingreso de estacionamiento de la UCSM.

3.2. Objetivos Específicos

- Definir el método de adquisición de imágenes y las características necesarias.
- Aplicar el procesamiento digital de imágenes para la localización y segmentación de placas de vehículos mayores a partir de capturas en tiempo real.
- Desarrollar el algoritmo de las RNA para su aplicación en el reconocimiento de placas vehiculares según una base de datos.
- Integrar y evaluar los sistemas de captura, reconocimiento e identificación de vehículos.

4. Justificación

La presente investigación se enfocará en desarrollar un sistema de reconocimiento vehicular aplicable a nuestra Universidad Católica de Santa María. Es por ello que se presentará las 2 bases que muestran la importancia que acarrea: metodología de PDI y rendimiento práctico.

La metodología de PDI (procesamiento digital de imágenes) incluye una combinación de técnicas, las cuales abarcan localización de áreas (placa vehicular) y reconocimiento de caracteres (OCR). Estas se combinarán con el entrenamiento de redes neuronales artificiales (RNA) para ver sus resultados en carga computacional, tiempo y precisión, los cuales se compararán con el método convencional: coeficiente de correlación de Pearson.

El rendimiento práctico se basa en evaluar el sistema tanto en su confiabilidad de identificación, como en la capacidad de reproducirse en cámaras sin sistema de identificación integrado por hardware, reduciendo el costo total necesario para aplicar esta tecnología en instituciones que no cuenten con recursos altos o necesiten un mayor número de cámaras.

Por lo tanto, uniendo el aprendizaje inherente en las RNA y la capacidad de reconocimiento de la cámara por software, se busca presentar una alternativa confiable de seguridad para cumplir con las necesidades del usuario de una institución o residencia privada.

5. Descripción del problema

En la actualidad, debido a la gran cantidad de vehículos en circulación, una de las necesidades más recurrentes en los establecimientos públicos y privados, es el control de ingreso y salida de estos vehículos y, por ende, de los usuarios propietarios de estos. En el caso de la Universidad Católica de Santa María, cuenta con un sistema de identificación de personal docente y universitario para el ingreso a sus instalaciones, en el cual se observa la posibilidad de ampliar dicho reconocimiento a los vehículos que hacen uso del servicio del estacionamiento.

La siguiente tesis pretende indagar y profundizar en dicha posibilidad, mediante el diseño e puesta a prueba de un sistema de identificación vehicular, enfocado al reconocimiento de placas vehiculares utilizando técnicas de reconocimiento óptico de caracteres (OCR) como son las redes neuronales artificiales (RNA). La propuesta implica una investigación y el diseño del software necesarios para aplicar dicha técnica, dividido en etapas; así como el hardware y comunicación necesaria entre el usuario y la interfaz para su implementación y testeo de rendimiento.

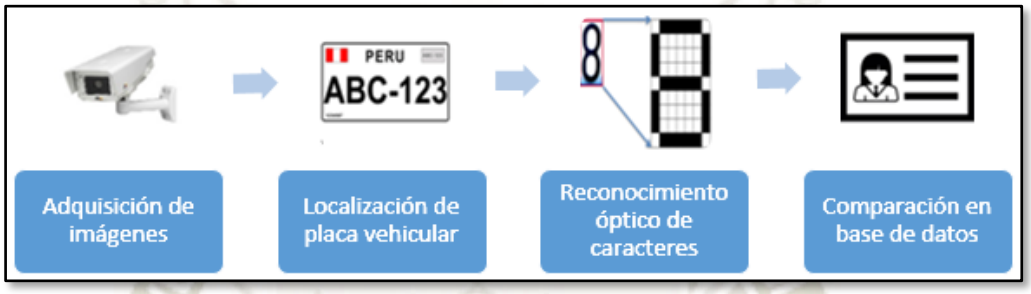


Figura 1. Esquema General del Sistema de Identificación de Vehículos
Fuente: Elaboración Propia

Se considera que las 2 etapas intermedias afectan en gran medida la efectividad de nuestro sistema; por lo que es necesario definir condiciones y bases de funcionamiento, como las mostradas en la Figura 2, para utilizar técnicas de procesamiento digital específicas, las cuales cada una de las tareas y compararlas entre sí, para mejorar el rendimiento general del sistema.

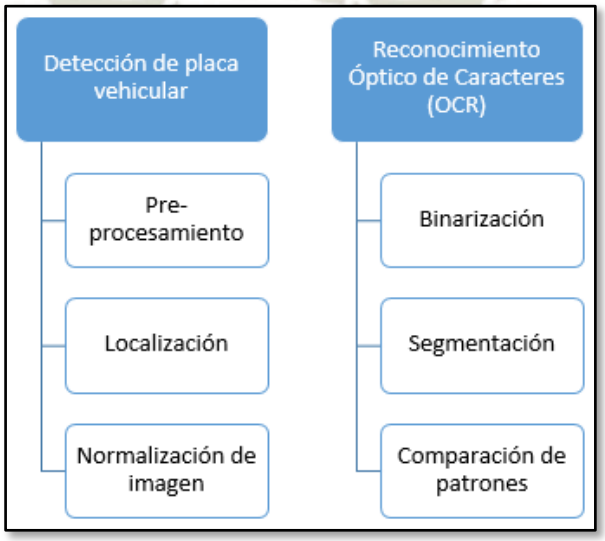


Figura 2. Procesos pertenecientes a las Etapas 2 y 3
Fuente: Elaboración Propia

6. Antecedentes

6.1. Antecedentes Locales

Baluarte Hurtado, Roberto (2014) **"Diseño e implementación de un sistema de video inteligente para el seguimiento de patrones" – Tesis de grado, Universidad Católica de Santa María – Arequipa, Perú**

Este trabajo de tesis desarrollado en la ciudad de Arequipa presenta un desarrollo de sistema de video inteligente enfocado al seguimiento de objetos, basándose en el procesamiento de imágenes, la aplicación de lógica difusa como controlador y un mecanismo de direccionamiento como actuador. El sistema se desarrolla en la plataforma Linux y el lenguaje de programación Python.

Se muestra en el trabajo el desarrollo de la programación necesaria para el seguimiento de un objeto en el plano de la cámara, a través de características resaltantes de este que lo diferencian del entorno. Este seguimiento de patrones se puede utilizar para lograr un mejor enfoque en la localización de placas vehiculares a partir de una cámara de video vigilancia.

6.2. Antecedentes Nacionales

Mundaca Vidarte, George Antonio (2016) **"Detección de caracteres de placas de automóviles mediante técnicas de visión artificial" – Tesis de grado, Universidad de Piura – Piura, Perú**

El proyecto desarrollado en la ciudad de Piura pretende realizar la identificación de caracteres en placas de automóviles, utilizando algoritmos de visión artificial, en los cuales incluye técnicas de variación de gradientes, filtro Sobel y transformada de Bottom-hat. Además de ello utiliza correcciones de perspectiva con el uso de transformadas de Hough, recurriendo para la OCR el método de correlación de Pearson.

Este proyecto logra tiempos de computación de 2.69 segundos para localización e identificación de placas vehiculares, las cuales son estáticas y a una distancia prudente; sin embargo, indican la posibilidad de mejora del algoritmo de OCR utilizando redes neuronales artificiales (RNA) para aumentar el porcentaje de éxito a un valor mayor al 77% conseguido en dicho proyecto.

6.3. Antecedentes Internacionales

García García, Pedro Pablo (2018). **"Reconocimiento de imágenes utilizando redes neuronales" – Proyecto de Maestría, Universidad Complutense de Madrid – Madrid, España**

Este proyecto desarrollado en la ciudad de Madrid, España, se enfoca en el uso de Redes Neuronales Artificiales (RNA) para la extracción de patrones que caracterizan una imagen, pudiendo ser identificado según los datos almacenado en servicio web.

Entre las funciones principales de este proyecto está:

- Extraer información de las imágenes almacenadas en el servidor web.
- Realizar la captura y obtener las características de estas.
- Identificar la imagen a partir de las características y su comparación con la base de datos.
- Guardar la imagen detectada para su uso en próximas identificaciones.

Un teléfono móvil realiza la adquisición de imágenes, para lo cual utiliza librerías de OpenCV en Visual Studio. Los resultados fueron acertados, permitiendo cambios en orientación y posición, pero dando un indicativo de mejora a partir de necesitar una base de datos más amplia y variada, además de las condiciones de operación como iluminación y fondo de la imagen.

7. Alcances

El presente proyecto pretende desarrollar un sistema de identificación de vehículos aplicando las redes neuronales para el reconocimiento de caracteres. Su rango de vehículos no incluye vehículos menores (motos, motonetas, moto taxi, etc.).

El proyecto será implementado y evaluado con pruebas en la Universidad Católica de Santa María ubicada en el distrito Yanahuara, de la provincia de Arequipa, departamento de Arequipa. Se utilizará una base de datos para el reconocimiento de vehículos con la obtención de información adicional, respecto a sus características y su propietario.

CAPITULO II: MARCO TEÓRICO

1. Conceptos Básicos de Imágenes Digitales

La imagen digital se puede definir como una matriz numérica bidimensional, preferentemente binaria (1 y 0), la cual representa una imagen en el campo de la informática y capaz de ser procesada por cualquier dispositivo digital.

La unidad mínima de visualización de una imagen digital es el pixel, las cuales poseen información de valor y posición particular. La cantidad de píxeles indican el grado de calidad o detalle que se puede observar en distintos niveles de aproximación o "zoom".

1.1. Características de una imagen

La imagen digital obtenida ya sea por computadora o dispositivo de captura, presenta 3 características fundamentales que determinan la calidad de la imagen. Mediante ello se pueden realizar comparaciones, para la elección de métodos de captura y procesamiento que se adecúen mejor a las necesidades que el usuario requiera. Estas características son:

1.1.1. Resolución de imagen digital

La resolución de una imagen digital nos indica el número de píxeles que la conforman, la cual se expresa normalmente como "(N° píxeles de ancho) x (N° píxeles de alto)", por ejemplo: 400x300 o 800x600. Este valor permite conocer el nivel de calidad de una imagen digital, además de la "relación de aspecto" (4:3 en ambos casos).

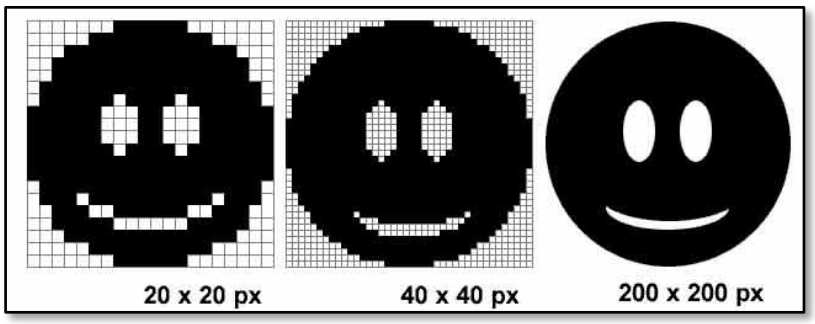


Figura 3. Diferentes resoluciones para una misma imagen.
Recuperado de http://www.popartplay.com/wp-content/uploads/2009/06/resolution.jpg

Tabla 1
Resolución de una imagen

Resolución	Dimensiones
0.5 MP	640 x 480 px
1 MP	1280 x 720 px
2 MP	1600 x 1200 px
4 MP	2560 x 1440 px

Fuente: Elaboración Propia

1.1.2. Profundidad de color

La profundidad de color indica el número de bits de información (gama de colores) destinados a cada pixel, y que afecta directamente al tamaño que ocupará una imagen digital en una memoria, al igual que la resolución de imagen. Esta característica también es conocida como profundidad de bit o profundidad de pixel. Según sea el nivel de profundidad utilizado, al conjunto de bit denominado "byte" se le asigna un color o canal, el cual variará su intensidad según el valor de los bits que lo componen. En la Fig. 4 se observa la combinación de 3 canales con diferentes valores para obtener un color específico.

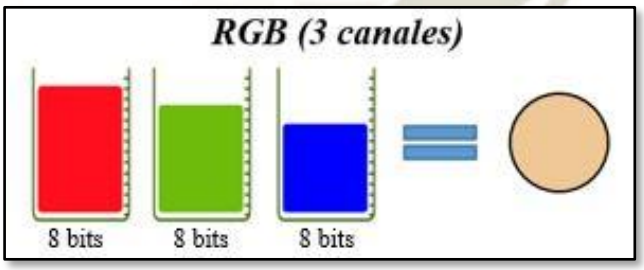


Figura 4. Color representado mediante una profundidad de 24 bits (RGB).
Fuente: Elaboración Propia

1.1.3. Tamaño de archivo

Esta característica indica el espacio utilizado para almacenar una imagen digital. Se obtiene a partir del producto del número de píxeles de ancho por el número de píxeles de alto y por la profundidad de color designada para el archivo.

Este tamaño se varía eligiendo diferentes valores de resolución y profundidad, para adecuar una imagen al espacio que dispongamos en una memoria o el límite de información que nos permita enviar un sitio Web. Se muestra un ejemplo en la tabla a continuación:

Tabla 2
Tamaño de una imagen

Resolución	Profundidad (bits)	Tamaño del archivo
400 x 300	1	120000 Bits = 15000 Byte = 14.65 KByte
480 x 320	1	153600 Bits = 19200 Byte = 18.75 KByte
400 x 300	4	480000 Bits = 60000 Byte = 58.59 KByte
480 x 320	4	614400 Bits = 76800 = 75 KByte

Fuente: Elaboración Propia

1.2. Clasificación de las Imágenes Digitales

La resolución de imagen puede trabajarse de 2 formas: estática o dinámica, dando lugar a la imagen de mapa de bits o la imagen vectorial, respectivamente.

1.2.1. Imagen de Mapa de Bits

La imagen de mapa de bits o ráster consta de puntos independientes entre sí llamados píxeles con valores de color asignados y los cuales están limitados por la resolución de la imagen. Es ampliamente utilizado en dispositivos digitales de captura de imágenes, como escáneres y cámaras, por su gran gama de tonalidades que plasman la realidad en imágenes de gran detalle.

Sin embargo, no permite la separación de elementos de la imagen como objetos geométricos independientes. Además, pierde detalle al aplicársele ampliaciones (zoom), debido a su aspecto dentado (“pixelado”). En la Fig. 5 se observa un ejemplo de este fenómeno.

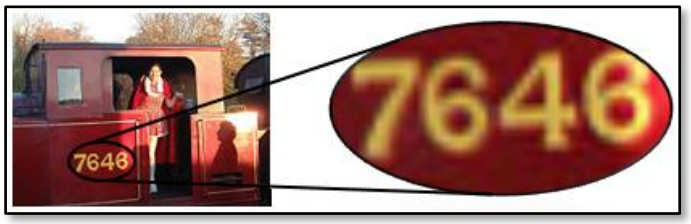


Figura 5. Imagen de Mapa de Bits y fenómeno de “pixelado”.
Fuente: Elaboración Propia

1.2.2. Imagen Vectorial

La imagen vectorial está formada por elementos geométricos representados por líneas que siguen una ecuación matemática para obtener sus atributos de forma, posición, etc. Esta característica les permite conversar su calidad durante la ampliación o reducción de la imagen, además de mover o cambiar las propiedades de cada elemento sin afectar al resto. Como resultado de su definición matemática, el detalle de la imagen es independiente de la resolución y la imagen ocupa un espacio menor al tipo ráster. Por ello la imagen vectorial es útil para el área de diseño de figuras geométricas, como se observa en la Fig. 6, pero defectuoso para la representación de la realidad en una imagen digital, por su falta de detalles.

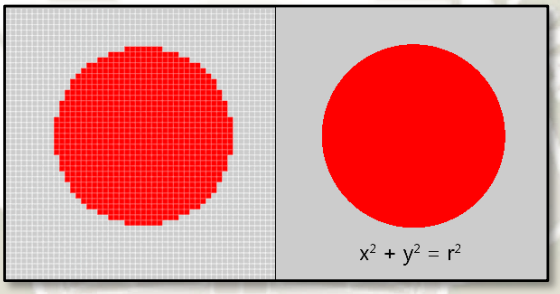


Figura 6. Círculo representado por imágenes ráster (izquierda) y vectorial (derecha).
Recuperado de: https://curiosoando.com/wp-content/uploads/2014/02/bitmap_vs_vector.png

La conversión de imagen de mapa de bits a vectorial se denomina vectorización y es posible realizarla por medios manuales o computacionales, mientras que el proceso inverso llamado rasterización es más sencillo de realizar. Ciertas propiedades que posean cada uno de estos tipos de imágenes se pierden al realizar la conversión, por lo que no es posible recuperar por completo la imagen después de ejecutar ambos procedimientos.

1.3. Formatos de Imágenes de Mapa de Bits

Las imágenes digitales se pueden almacenar en distintos formatos. Cada uno de ellos tiene sus propias características de calidad, tamaño, colores, etc., reconocidos en la computadora por la extensión del archivo. En la Fig. 7 se observan los formatos más utilizados.

Figura 7. Formatos de imagen y sus respectivos íconos.
Recuperado de: http://www.iconhot.com/icon-pack/file-icons-vs-2.html

1.3.1. BMP (Bits Maps Protocole)

Formato estándar de Windows, reconocido por la mayoría de programas de PC y capaz de almacenar gran cantidad de información de la imagen (16 millones de colores o 24 bits), manteniendo la calidad de esta. A pesar de ello genera archivos de gran tamaño, incapaces de leerse por ordenadores Macintosh ni ser soportados por la Web, por lo que el formato es de uso orientado a aplicaciones de Windows.

1.3.2. GIF (Graphics Interchange Format)

Formato desarrollado por la compañía Compuserve para comprimir imágenes digitales sin pérdida de datos. Posee una gama de 256 colores (8 bits), que junto a su reducido tamaño lo hace apropiado para realizar dibujos y transmitirlos por la Web. Además de ello permite la creación de animaciones y transparencia. Sin embargo, no es capaz de plasmar correctamente imágenes fotográficas por su reducida cantidad de colores disponibles.

1.3.3. JPG - JPEG (Joint Photographic Experts Group)

Formato ampliamente utilizado para fotografía y publicación Web, debido a su amplia gama de colores (16 millones de colores o 24 bits). La característica principal es su alto nivel de compresión que permite obtener archivos pequeños de baja calidad o de gran tamaño con una calidad alta. Sin embargo, la compresión genera pérdidas de calidad una vez realizada, lo cual lo hace un proceso irreversible después de la creación de archivo JPEG. Las cámaras fotográficas utilizan este tipo de formato para la extracción de sus imágenes digitales.

1.3.4. TIF - TIFF (Tagged Image File Format)

Formato compatible con ordenadores Windows o Macintosh con capacidad de compresión LZW (sin pérdida de información), ideal para almacenar imágenes digitales de alta calidad. Este nivel de calidad se debe a un rango de 48 bits para la paleta de colores. Es utilizado por los escáneres, además de ser compatible con la mayoría de programas de edición de imágenes. A pesar de ello su uso en la Web está restringido por el elevado tamaño de archivo que se genera.

1.3.5. PNG (Portable Network Graphic)

Este formato combina características de los formatos JPEG y GIF. Trabaja con una amplitud de colores similar al JPEG, pero con una capacidad de compresión sin pérdida de calidad. También permite el uso de transparencia como el formato GIF, pero no animaciones. Si bien los archivos son de mayor tamaño al formato GIF, conservan un peso reducido para su uso en la Web y otros programas de edición de imágenes digitales.

1.4. Modos de color

"El modo de color o modo de imagen determina la combinación de los colores en función del número de canales de un modelo de color. Los diferentes modos de color dan lugar a diferentes niveles de detalle de color y tamaño de archivo" (Adobe Corporation, 2014).

La profundidad de color representa su característica principal, la cual nos indica la cantidad de colores que maneja la imagen digital, diferenciando los distintos modos de color y los usos que se le dan a cada uno, según el detalle que se requiera. En la Tabla 1 se observa los distintos niveles de profundidad con los modos de color respectivos. Se debe considerar que algunos modos de color pueden trabajar en distintas profundidades de color, siendo el tamaño del archivo final un factor importante para realizar dicha selección.

Tabla 3
Profundidades y modos de color

Profundidad de Color	Bits/Pixel	Modo de color
2 Colores (Blanco y Negro)	1	Mapa de Bits
16 Colores/Grises	4	Escala de Grises
256 Colores/Grises	8 (1 Byte)	Indexado, Escala de Grises, Duotono
65.536 Colores	16 (2 Bytes)	High Color (R, G, B, alpha), Escala de Grises
16.777.216 Colores	24 (3 Bytes)	"Color Verdadero", RGB, LAB (Espacios de Color)
4.294.967.296 Colores	32 (4 Bytes)	CMYK

Fuente: Elaboración Propia

Cada modo de color presenta similitudes en la base de colores utilizados. Por ello se procederá a explicar los más utilizados para el procesamiento de imágenes por su simplicidad y compatibilidad con programas como Matlab, Visual Studio, etc.

1.4.1. Modo RGB y CMYK

Los nombres vienen de los colores base que utilizan para la composición de sus 16 milones de colores. Estos se dividen en 3 o 4 canales, siendo para el RGB el rojo-verde-azul y para el CMYK el cyan-magenta-amarillo-negro, cada uno con 8 bits asignados a los valores de intensidad, que varían de 0 al 255. Utilizado en formatos de imagen de gran calidad, como JPEG, BMP y PNG.

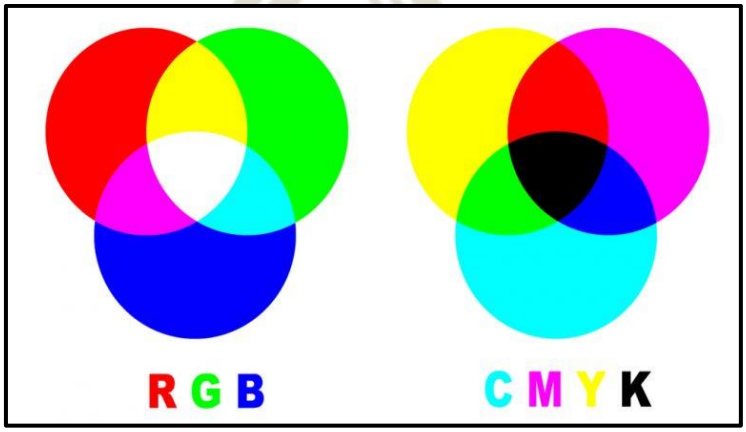


Figura 8. Colores primarios de la luz
Fuente: *Elaboración Propia*

1.4.2. Modo Indexado

Desarrollado como un solo canal de 8 bits de información, los cuales permiten elegir entre 256 colores, lo que reduce el tamaño de la imagen digital. Estos colores son discretos, es decir no varían de intensidad unos de otros, sino que funcionan como una paleta con su respectiva etiqueta cada uno. Funciona en formatos de bajo tamaño como PNG y GIF.

Figura 9. Modo indexado y su paleta de colores
Fuente: *Elaboración Propia*

1.4.3. Modo de Escala de Grises

Presenta 8 bits de información agrupado en un solo canal que representa la intensidad de gris de cada pixel de la imagen digital. Se utilizan 256 tonos que varían desde 0 (blanco) y 255 (negro). Existen casos en los que su profundidad varía desde 4 hasta 16 bits de información.

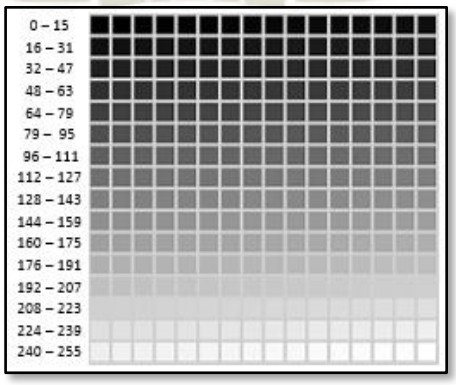


Figura 10. Modo de Escala de Grises y su paleta de colores
Fuente: *Elaboración Propia*

1.4.4. Modo de Mapa de Bit (Binarizado)

El modo más simple debido a su profundidad de 1 bit de información, el cual representa los 2 valores de color básicos (blanco y negro). Generan archivos del menor tamaño posible, con píxeles blancos o negros puros. Utilizado para la impresión de logos corporativos, reconocimiento de formas o elementos, y creación de base de datos de imágenes.

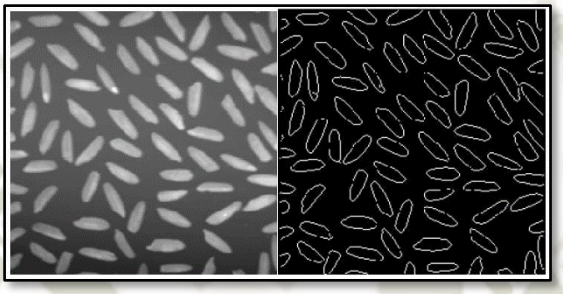


Figura 11. Binarización de una imagen de escala de grises
Recuperado de:
https://www.researchgate.net/publication/267240432_Deteccion_de_bordes_mediante_el_algoritmo_de_Canny

2. Procesamiento Digital de Imágenes (PDI)

El procesamiento digital de imágenes (PDI) es un área de la ingeniería encargado de la manipulación y análisis de una imagen digital por medio de un ordenador. Abarca conocimientos combinados de diferentes disciplinas como matemáticas, físicas, ciencia de la computación e ingeniería eléctrica (señales). Su desarrollo se enfoca en dar solución a 3 problemáticas relacionadas a las imágenes digitales:

- Digitalización: La adquisición y codificación de imágenes al formato digital para su lectura, almacenamiento, impresión o transferencia.
- Visualización: Mejora o recuperación de la información de la imagen para facilitar su interpretación humana o por software.
- Descripción: Extracción y caracterización de los elementos que conforman una imagen para su aplicación en “visión de máquina” (Machine Vision).

2.1. Modelo de PDI

El PDI abarca un conjunto de etapas, cada una de ellas con herramientas o técnicas que se ejecutarán según sea la aplicación deseada. Se considera 5 etapas principales en el PDI (Wainschenker, Massa, & Tristán, 2011), sin embargo se agrega una 6ta denominada "interpretación", como se observa en la Fig. 12.

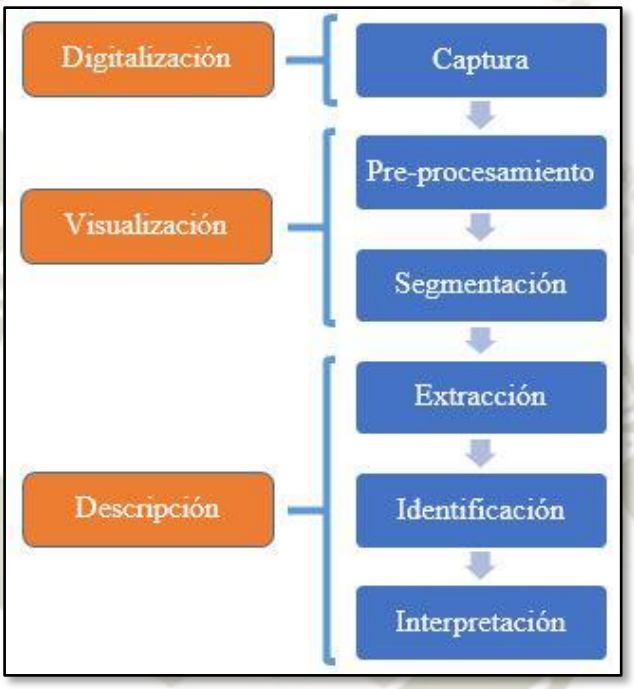


Figura 12. Modelo de PDI en 3 niveles de aplicación.
Fuente: *Elaboración Propia*

- Captura o adquisición: de una imagen utilizando un dispositivo digital (escáner, cámara), considerando características del archivo como modo de color, resolución, formato, etc.
- Pre-procesamiento: abarca 2 grupos de técnicas. Estas son la mejora de la calidad de imagen (reducción de ruido, contrastes) y la eliminación o recorte de las áreas que no sean de interés.
- Segmentación: reconocer y separar los componentes de la imagen como elementos independientes para el estudio individual de ellos.
- Extracción: búsqueda, selección y cuantización de características (forma, brillo, color) que diferencien cada elemento obtenido de la imagen para la clasificación.

- Identificación: clasifica cada elemento de la imagen en grupos con características en común, para etiquetarlos o asignarles un nombre (letras, monedas).
- Interpretación: a partir de un análisis de los resultados, asocia un significado a los grupos de elementos clasificados (palabras, dinero) según la problemática estudiada.

2.2. Software de PDI

Actualmente el mercado nos ofrece una variedad de software de procesamiento digital de imágenes. Estos varían según el concepto libre o comercial, directo o basado en código, entre otros factores. A continuación, se muestran algunos de los más utilizados:

- Adobe Photoshop: Líder en aplicaciones de edición de imágenes por su compatibilidad con variedad de formatos de imágenes, su facilidad de uso, popularidad y herramientas diversas. http://www.adobe.com/es/products/photoshop.html?promoid=PC1PQQ5T&mv=other#x
- Matlab: Programa de desarrollo de software que presenta las herramientas "Image Processing Toolbox" y "Image Acquisition Toolbox". En ellas se encuentran fórmulas, aplicaciones de PDI ya desarrolladas y que simplifican el tamaño del código base. https://www.mathworks.com/products/matlab.html
- PhotoStudio: Desarrollado por ArcSoft. Programa para la edición profesional de fotos e imágenes, en el nivel de visualización del modelo mencionado anteriormente. http://www.arcsoft.com/
- Mathematica: Desarrollado por Wolfram. Posee librerías de procesamiento de imagen para captura, visualización y extracción de características. http://www.wolfram.com/mathematica/
- OpenCV: Librería de computación visual de código libre, con interfaces para trabajar con lenguajes como Python, C, C++, Java, etc. http://opencv.org/

2.3. Aspectos Generales del PDI

El PDI está basado en 2 fundamentos generales: relaciones entre píxeles y ruido de la imagen (García Santillán, 2008). Su análisis permite la selección de las técnicas que permitan mejoras en la calidad de imagen o una mejor descripción de los elementos que la componen.

2.3.1. Relaciones entre píxeles

Los píxeles determinan su posición en la imagen mediante coordenadas (x,y) limitadas por la resolución. Estas coordenadas permiten el análisis de los píxeles como una matriz en la cual se forman conjuntos que representan los elementos visibles que componen la imagen. Para dichos conjuntos se forman relaciones de conectividad y relaciones de distancia (Platero Dueñas, 2013).

En el caso de conectividad se quiere agrupar píxeles adyacentes con un valor igual (binario con unos y ceros) o cercano (escala de grises con su tonalidad) como un mismo objeto proyectado. Para ello se analiza un píxel "p" con coordenadas (x, y) el cual presenta vecinos divididos en 3 tipos de vecindad como se observa en la Fig. 13.

- 4-vecinos: denotada $N_4(p)$ y representado por los píxeles adyacentes horizontal ("y-1" y "y+1") y verticalmente ("x-1" y "x+1") al pixel *p*.
- 4-vecinos diagonales: denotada $N_D(p)$ y representado por los 4 píxeles adyacentes diagonalmente ("x ± 1" y "y ± 1") al pixel *p*.
- 8-vecinos: denotada $N_8(p)$ y representado por la unión de las vecindades $N_4(p)$ y $N_8(p)$.

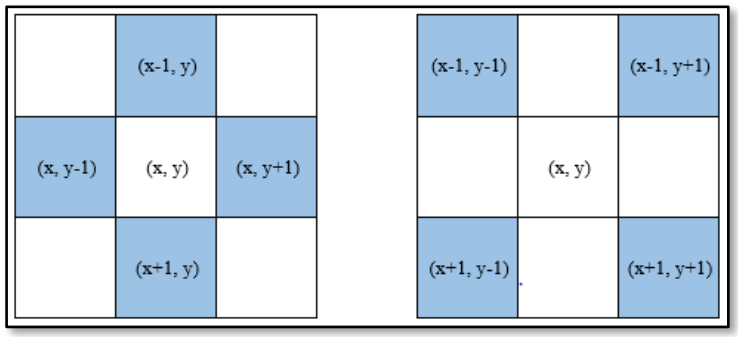


Figura 13. Tipos de vecindad de 4 píxeles: vertical/horizontal y diagonal
Fuente: *Elaboración Propia*

Para el caso de la relación por distancia entre píxeles, los cuales presentan 3 formas de cálculo (Platero Dueñas, 2013). La primera de ellas es la geométrica o euclídea, la cual considerando 2 puntos *p* y *q* con coordenadas *(x, y)* y *(s, t)* respectivamente:

$$D_E(p,q) = \sqrt{(x-s)^2 + (y-t)^2} \qquad [2.1]$$

En la segunda forma se considera movimientos horizontales y verticales. Este se denomina distancia rectangular o Manhattan, definida como:

$$D_M(p,q) = |x-s| + |y-t| \qquad [2.2]$$

Por último, está la distancia "tablero de ajedrez" o Tchebychev, que utiliza las vecindades como unidad. A continuación, su fórmula:

$$D_T(p,q) = \max(|x-s|, |y-t|) \qquad [2.3]$$

Como fuente de comparación entre estas tres fórmulas, se muestra en la Fig. 14 una matriz de 5x5 píxeles que indica la distancia de cada píxel respecto al centro, calculado con cada método.

$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$
$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$
2	1	0	1	2
$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$
$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

Figura 14. Tipos de distancias: geométrica (1ra), rectangular (2da) y tablero ajedrez (3ra)
Fuente: *(Platero Dueñas, 2013)*

2.3.2. Ruido en imágenes digitales

El ruido, según su definición, es cualquier perturbación que sufre una señal en el proceso de adquisición o transmisión o almacenamiento. Por lo tanto, para el PDI, el ruido digital indica una variación en el valor (brillo o color) de los píxeles que conforman una imagen digital producido por el dispositivo de entrada (cámara, escáner) o medio de transmisión.

El ruido digital, según la naturaleza matemática de su variación, se clasifica en 3 tipos: (Departamento de Computación, 2015)

- Aditivo: Conocido como Gaussiano, es el más común y que produce menores variaciones en la imagen, pero que afecta a todos los píxeles. Producido generalmente por dispositivos de adquisición (cámaras, sensores). Se define en la ecuación [2.4] como la distorsión de una imagen f(x, y) por una señal de ruido gaussiano de distribución normal n(x, y).

$$g(x,y) = f(x,y) + n(x,y) \qquad [2.4]$$

- Impulsivo: O efecto "sal y pimienta". Sin abarcar a la totalidad de los píxeles, estos toman valores muy altos (blancos o sal) o bajos (negros o pimienta), dejando de tener relación con su valor real. Suele ocurrir en medios de distribución. Se puede modelar bajo la ecuación [2.5], considerando un ruido impulsivo i(x, y) y un factor "p" de valor 0 o 1.

$$g(x,y) = (1-p) * f(x,y) + p * i(x,y) \qquad [2.5]$$

- Multiplicativo: Se puede encontrar en imágenes de radares y ecografías. El ruido que afecta a la imagen es de naturaleza uniforme. Esencialmente un factor constante que altera el valor de todos los píxeles de la imagen. Para ello se considera al factor m(x, y) constante en la ecuación [2.6].

$$g(x,y) = f(x,y) * m(x,y) \qquad [2.6]$$

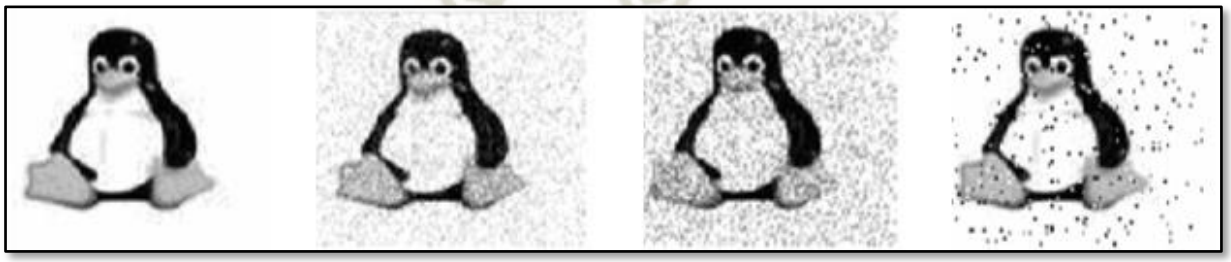


Figura 15. Tipos de ruido: original (1ra), aditivo (2da), impulsivo (3ra) y multiplicativo (4ta)
Fuente: (García Santillán, 2008)

3. Herramientas PDI en MatLab

MATLAB dispone de una amplia variedad de programas de apoyo especializados, denominados "Toolbox", que extienden en gran medida el número de funciones incorporadas en el programa principal. Estas "cajas de herramientas" están agrupadas para abarcar distintos rubros de la programación y simulación, siendo uno de ellos el procesamiento de imágenes.

3.1. Image Processing Toolbox

Image Processing Toolbox proporciona un conjunto completo de algoritmos estándar de referencia y apps de flujo de trabajo para el procesamiento, el análisis y la visualización de imágenes, así como para el desarrollo de algoritmos (MathWorks). Entre sus principales prestaciones se encuentran:

- Adquisición y visualización
- Pre-procesamiento de imágenes
- Análisis de imágenes
- Segmentación de imágenes
- Registro de imágenes
- Flujos de trabajo de procesamiento de imágenes 3D
- Aceleración de algoritmos e implementación

3.2. Representación de la imagen

"En MatLab una imagen a escala de grises es representada por medio de una matriz bidimensional de *m x n* elementos en donde *n* representa el número de píxeles de ancho y *m* el número de píxeles de largo. Por otro lado una imagen de color RGB es representada por una matriz tridimensional *m x n x p*, donde *m* y *n* tienen la misma significación que en el caso anterior, mientras *p* representa la gama de colores, que para RGB es la tonalidad de rojo, verde y azul" (Cuevas Jimenez & Zaldivar Navarro, s.f.).

3.3. Lectura, escritura y visualización de imágenes

Se utiliza el comando *imread* para leer una imagen desde un archivo de formato gráfico. Para ello se utiliza la siguiente sintaxis:

imread(‘nombre del archivo’)

Donde el *nombre del archivo* es una cadena de caracteres conteniendo el nombre completo de la imagen con su respectiva extensión. Se deben considerar los formatos listados en la tabla 3 como válidos para el software de MatLab.

Tabla 4
Formatos de Imagen Admisibles en MatLab

Formato	Definición	Extensión
TIFF	Tagged Image File Format	.tiff
JPEG	Join Photograph Expert Group	.jpg
GIF	Graphics Interchange Format	.gif
BMP	Bit Mapped Picture	.bmp
PNG	Portable Network Graphics	.png
XWD	X Window Dump	.xwd

Fuente: Elaboración Propia

Si se desea realizar la visualización o procesamiento de una imagen en MatLab, se tiene que almacenar dicha imagen en una variable que la represente. Como ejemplo, considerando *foto.png* como un archivo gráfico de formato PNG, y una variable de Matlab *matriz_img*, entonces se deberá escribir en la línea de comandos:

matriz_img = imread(‘foto.png’)

A partir de ello se puede conocer las dimensiones de la imagen, con el comando *size*. Para almacenar dicha información en una matriz de 1x2, se utiliza la siguiente sintaxis.

[a, b] = size (‘foto.png’)

Para el caso que se desee grabar el contenido de una variable en un archivo gráfico, se utiliza el comando *imwrite*, cuya sintaxis es la siguiente:

imwrite(variable, 'nombre del archivo')

Donde *variable* representa la imagen contenida en una variable de MatLab, mientras que el *nombre del archivo* representará el título del archivo junto con la extensión específica del formato deseado, el cual debe pertenecer a uno de los mencionados en la Tabla 3.

Si consideramos una variable gráfica *imagen2*, la cual se desea almacenar en un archivo de formato TIFF con el nombre *foto2*, entonces el comando escrito debería ser:

imwrite(imagen2, 'foto2.tiff')

En el caso de la visualización, el comando *imshow* mostrará la imagen en una ventana de MatLab. También se puede utilizar el comando *figure*, para generar una nueva ventana sin eliminar una ya creada. Entonces para mostrar ambas figuras *imagen* e *imagen2*, se escribiría lo siguiente en la ventana de comandos:

imshow(imagen)
figure, imshow(imagen2)

Figura 16. Ejemplo de visualización en MatLab
Fuente: *Elaboración Propia*

3.4. Pre-procesamiento de una imagen digital

3.4.1. Acceso a información de píxel

El píxel es la unidad de información en una imagen digital, siendo el acceso a este una de las operaciones más comunes. Para ello con conocer la posición del píxel es posible obtener su valor, el cual en escala de grises se representa en un solo plano, mientras que en RGB se utilizan 3 planos.

Para la obtención del valor requerido de un píxel de coordenadas *(x, y)* de una variable gráfica *matriz_img* se pueden utilizar 3 métodos:

- Elemento de una matriz: considerando la variable como matriz se puede obtener los valores de cada uno de sus elementos con la siguiente sintaxis en caso de escala de grises:

  gray = matriz_img(x, y)

  O en caso del modo de color RGB, considerar el plano de color *n* (1, 2 y 3 para rojo, verde y azul respectivamente):

  rgb = matriz_img(x, y, n)

- Comando *impixel*: permite obtener el valor del píxel requerido indistintamente del modo de color utilizado en la imagen, siendo un número escalar en caso de escala de grises o una matriz de 1x3 para el modo de color RGB:

  color = matriz_img(imagen, x, y)

- Comando *improfile*: se utiliza para obtener el perfil de colores de un segmento de la imagen, logrado a partir de haber ejecutado el comando *imshow* previamente. Con ello obtenemos una o tres (RGB) señales a lo largo del segmento seleccionado por el puntero, el cual cambia a +. No requiere de datos de posición, solo la selección del segmento mediante el mouse.

Para la escritura de valores en píxeles, simplemente se le asigna al elemento de la matriz un valor escalar o matricial (*r*, *g* y *b*) según sea el modo de color, mediante el comando del caso 1:

matriz_img(x, y) = [r g b]

3.4.2. Conversión entre modos de color

Dependiendo de la operación que se vaya a realizar a la imagen, se necesitará de un modo de color en particular, para lo cual se utiliza una conversión del modo original a otro admisible por el programa MatLab. Sin embargo, puede ocurrir que se desconozca el modo de color de la imagen, por lo cual se utilizan los comandos listados en la siguiente tabla:

Tabla 5
Comandos informativos sobre imágenes digitales

Comando	Descripción
Isbw	Regresa un valor verdadero (1) si la imagen es binaria
Isgray	Regresa un valor verdadero (1) si la imagen es escala de grises
Isind	Regresa un valor verdadero (1) si la imagen es indexada
Isrgb	Regresa un valor verdadero (1) si la imagen es RGB
imfinfo	Regresa información sobre la imagen

Fuente: (García Santillán, 2008)

Si se conoce el modo de color de la imagen, entonces se puede realizar la conversión para elegir la más adecuada, aunque en el proceso se puede perder información, lo que lo hace reversible en caso no se guarden como variables separadas. En la Tabla 5 se observan los comandos más utilizados para dichas conversiones y sus descripciones.

Tabla 6
Comandos de conversión de modos de color

Comando	Descripción
gray2ind - ind2gray	Convierte una imagen en escala de grises a una imagen indexada o viceversa
rgb2gray	Convierte una imagen RGB a una imagen en escala de grises
rgb2ind – ind2rgb	Convierte una imagen RGB a una imagen indexada o viceversa
imbinarize	Binariza una imagen (escala de grises) mediante la umbralización

Recuperado de: https://es.mathworks.com/products/image.html

3.4.3. Selección de una sección

Para el corte de una sección de una variable gráfica *I*, se utiliza el comando *imcrop* cuya sintaxis puede ser en caso de seleccionar manualmente una región:

I2 = imcrop(I)

En cambio, si la región es un rectángulo de valores *rec = [$x_{inicial}$ $y_{inicial}$ ancho alto]*:

I2 = imcrop(I, rec)

Siendo $x_{inicial}$ y $y_{inicial}$ la posición de la esquina superior izquierda de la región deseada.

3.5. Operadores Geométricos

Estas herramientas modifican las posiciones y relaciones espaciales que existen entre los píxeles. El toolbox permite la transformación mediante 2 comandos: *imrotate* y *imresize*.

Un factor importante para realizar de una transformación geométrica es la interpolación. Este representa el proceso por el cual se calculan los valores del mapa de colores de los píxeles creados a partir de una operación. En imágenes que presentan modos de color con múltiples planos (por ejemplo, RGB), se debe considerar que el valor en cada plano es calculado independiente del resto.

Considerando la imagen digital como una matriz bidimensional, la interpolación que genera MatLab será de la misma naturaleza, calculando el valor de cada píxel como un promedio de los píxeles que tienen posiciones próximas a este. La cantidad de píxeles considerados está determinada por los métodos de interpolación utilizados:

- Vecino más cercano (comando *nearest*): Método básico con un tiempo de procesamiento bajo. Se elige el píxel más cercano al punto interpolado, o también se considera como aumentar el tamaño de cada píxel de la imagen.
- Bilineal (comando *bilinear*): Considera los valores de los píxeles que rodean el punto interpolado en una vecindad de 2x2. Se promedia el valor de estos 4 píxeles generando un suavizado en la imagen a costa de un tiempo de procesamiento más alto.

- Bicúbica (comando *bicubic*): Funciona de manera similar al bilineal, pero aumentando la cantidad de píxeles a 16 o vecindad 4x4. Sacrifica bastante tiempo de procesamiento a cambio de una mayor calidad de imagen.

3.5.1. Amplificación y Reducción

El cambio de tamaño se realiza con el comando *imresize*, sea para el aumento o disminución de la imagen. Considerando *I* la imagen original y *I2* la imagen nueva, la sintaxis es la siguiente:

I2 = imresize(I, escala, método)

Se utilizan 2 datos adicionales para el comando. El primero de ellos es la *escala*, la cual se representa en 2 formas: escalar como un factor que multiplica el tamaño de la imagen (por ejemplo *0.5* para la mitad); y vectorial para determinar la resolución o tamaño final de la imagen (matriz *[alto ancho]* con valores en píxeles). El segundo de los datos es el método de interpolación explicado anteriormente y el cual debe ser colocado entre comillas (*'bicubic'* por ejemplo).

3.5.2. Rotación

El giro de una imagen se realiza con el comando *imrotate*. Considerando las variables *I* y *I2* anteriores, la sintaxis se escribe:

I2 = imrotate(I, giro, método)

Se utilizan 2 datos adicionales para el comando. El primero de ellos es el giro, en grados, al cual se desea rotar la imagen. El segundo de los datos es el método de interpolación explicado anteriormente y el cual debe ser colocado entre comillas (*'bicubic'* por ejemplo).

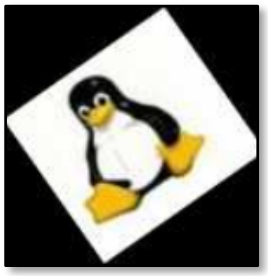


Figura 17. Fenómeno del fondo negro en imagen rotada
Fuente: *(García Santillán, 2008)*

Se debe considerar que al usar valores de giro no múltiplos de 90, la imagen representada por una matriz, presentará píxeles de fondo que asumirán el color negro por defecto (Fig. 17).

3.6. Filtrado Espacial

El filtraje espacial es un conjunto de técnicas que tienen por objetivo modificar o mejorar la imagen. Se considera como una operación de vecindario, pues utiliza algoritmos que configuran matrices máscaras que recogen los datos de los píxeles vecinos de la imagen original para calcular el valor de cada píxel en una imagen procesada.

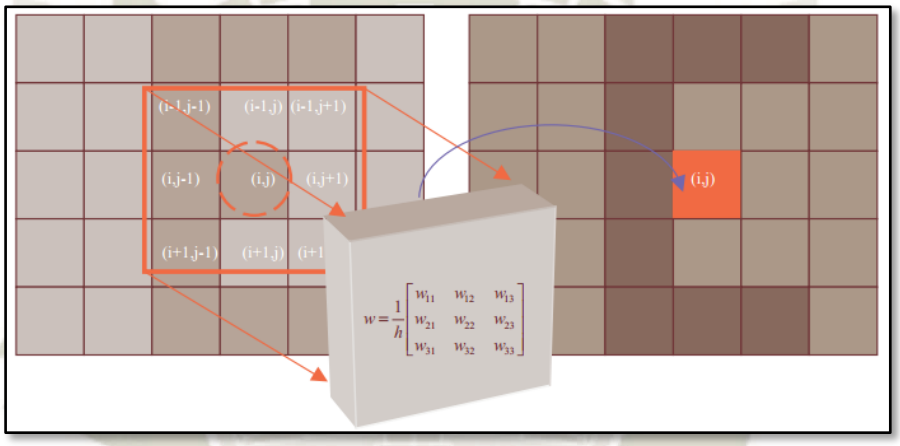


Figura 18. Filtrado espacial utilizando una máscara de 3x3
Fuente: *(Cuevas Jimenez & Zaldivar Navarro, s.f.)*

Estos filtros se dividen en 2 tipos: filtros de paso bajo y filtros de paso alto, los cuales actúan según sea el objetivo que se espera de la imagen procesada (García Santillán, 2008).

3.6.1. Filtros de paso bajo

También llamados filtros suavizantes, son utilizados para lograr una imagen un poco borrosa y reducir el ruido digital. Según su naturaleza de la operación, se utilizan 2 métodos principales:

- Filtro de promedio (comando imfilter): Tiene naturaleza lineal, cuyo principio es el promediado de los píxeles que forman la vecindad del píxel designado. Su uso está enfocado al suavizado de la imagen digital, por ejemplo, ante el ruido gaussiano. Mediante el comando *fspecial* se puede especificar el tamaño de vecindad usado para el filtro.

Ejemplo: Creación del filtro *F* de vecindad 5x5 y filtrado por promedio de la imagen *I*

F = fspecial('average', 5)

I2 = imfilter(I, F)

- Filtro de mediana (comando medfilt2): Tiene naturaleza no lineal y su principio se basa en reemplazar el valor de un píxel por el de la mediana de los valores del conjunto del mismo y los ocho adyacentes (vecindad 3x3). Utilizado para la reducción de ruido, en especial el impulsivo, permitiendo conservar el resaltado de los bordes.

  Ejemplo: Filtrado de mediana de la imagen *I*

  I2 = medfilt2(I, F)

En las figuras a continuación, se observa la diferencia entre ambos filtros para tratar 2 tipos de ruidos: aditivo (gaussiano) e impulsional (salt and pepper). Se concluye la preferencia por el filtro de promedio para el caso de ruido aditivo que altera la totalidad de los píxeles, debido a su naturaleza lineal; mientras que para el caso del impulsional que afecta una parte del conjunto de píxeles, basta con un filtrado de mediana para eliminar los valores extremos de color.

Figura 19. Filtros de paso bajo en una imagen con ruido gaussiano
Fuente: *Elaboración Propia*

Figura 20. Filtros de paso bajo en una imagen con ruido impulsional
Fuente: *Elaboración Propia*

3.6.2. Filtros de paso alto

También llamados filtros atenuantes, son utilizados para intensificar detalles y cambios de alta frecuencia, atenuando espacios con intensidades uniformes. Según la aplicación, se dividen en:

- Realce de bordes: Se encarga de resaltar los píxeles con valores de color diferentes a sus vecinos, por lo que es necesario realizar un filtrado de ruido anteriormente para evitar incrementar su efecto en la imagen. En MatLab está implementado por el comando *filter2*, para el cual se necesita determinar las características del filtro por el comando *firpm*.

  Ejemplo: Suma de una imagen *I* con su respectivo realce *Ir* por un filtro *f*

  Ir = filter2(f, I)
  I2 = I + Ir

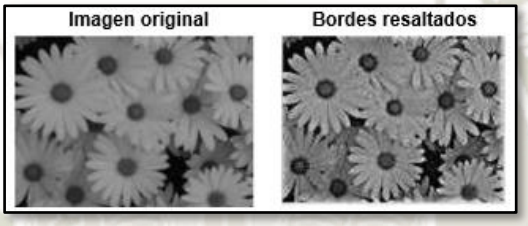


Figura 21. Ejemplo de filtrado de paso alto para resalte de bordes
Fuente: *(García Santillán, 2008)*

- Detección de contornos: Los contornos representan los límites de los objetos que forman una imagen. Se caracterizan por los cambios bruscos en la tonalidad de colores de los píxeles. En MatLab está implementado por el comando *edge*, cuya sintaxis es la siguiente:

  B = edge(I, método)

  Se debe especificar el *método* o algoritmo utilizado para la detección: Sobel, Prewitt, Robert, Canny. En la figura a continuación se muestra la diferencia entre 2 métodos muy utilizados.

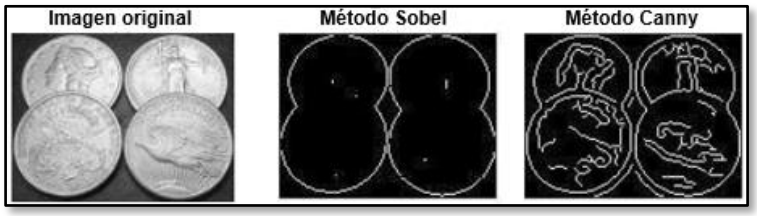


Figura 22. Detección de contornos en Imágenes por algoritmos Sobel y Canny
Fuente: *(García Santillán, 2008)*

3.7. Procesado Morfológico

La morfología matemática es una técnica de procesado no lineal de la imagen, interesada en la geometría de los objetos. Las operaciones morfológicas proporcionan información sobre la forma o estructura de una imagen (García Santillán, 2008).

Según sea el modo de color utilizado se presentan 3 tipos: binaria (más frecuente), intensidad de gris y policromática. También se deben considerar 3 factores que intervienen en el análisis y procesado morfológico:

- Conjuntos (imágenes)
- Elementos Estructurantes (formas)
- Operadores y/o Filtros Morfológicos

3.7.1. Conjuntos

El lenguaje utilizado en las técnicas morfológicas proviene de la teoría de conjuntos, según el cual cada conjunto de píxeles con valores similares representa la forma de los objetos de una imagen digital. Según el modo de color analizado, los conjuntos variarán en tamaño y definición.

En una imagen binaria, los conjuntos existentes están representados por puntos en un espacio bidimensional, donde cada píxel es un punto de coordenadas (x, y) el cual se clasifica según su igualdad de valor con sus píxeles adyacentes (negro o blanco) o según 2 planos definidos según:

$$\text{Objeto: } A = \{(x,y) \mid f(x,y) = 1\} \qquad [2.7]$$

$$\text{Fondo: } B = \{(x,y) \mid f(x,y) = 0\} \qquad [2.8]$$

En una imagen de escala de gris, los conjuntos están representados por puntos en un espacio tridimensional, en el cual se utilizan 2 componentes para la posición (x, y), mientras que el componente faltante representa la intensidad de gris del objeto, por lo cual la cantidad de objetos aumenta y se utilizan rangos de valores para la selección de píxeles con intensidades similares.

3.7.2. Elementos Estructurantes (EE)

El elemento estructurante es una figura geométrica utilizado como sonda o patrón en la examinación de la imagen inicial. Este presenta tamaño y forma definido por el usuario según la obtención de formas que se desea reconocer. El comando de Matlab *strel* crea un elemento estructurante, sea bidimensional o tridimensional, que se guarda en una variable para su uso en operadores o filtros morfológicos. La sintaxis es la siguiente:

EE = strel(forma, parámetros)

Ejemplos: Creación de formas según el comando *strel*

Línea (de 10 a 45°): ee1 = strel('line', 10, 45)
Cuadrado (5x5): ee2 = strel('square', 5)
Rectángulo (10x5): ee3 = strel('rectangle', 5, 10)
Disco (radio 10): ee4 = strel('disk', 10)

3.7.3. Operadores Morfológicos

Transforman una imagen binaria o en escala de grises, mediante el cambio de tamaño o forma de un conjunto de píxeles. Presentan 3 clasificaciones (Alba, Martín, Cid, & Mora, 2006):

- Extensivo: El resultado contiene a la imagen original.
- Anti extensivo: El resultado está contenido en la imagen original
- Idempotente: Al aplicar el operador 2 o más veces, el resultado sigue siendo el mismo.

Matlab presenta 4 operadores morfológicos, cuyo impacto en la imagen original depende directamente del EE utilizado. Estos son:

- Dilatación (comando *imdilate*): Expande los píxeles que conforman un conjunto de la imagen digital, o también definido como "el valor máximo del entorno de vecindad definido por el elemento estructurante" (Platero Dueñas, 2013). Su sintaxis está definida por:

I2 = imdilate(I, EE)

La dilatación es extensiva, debido a que agrega puntos del fondo que presentan vecindad al borde de un objeto. Por ello se utiliza para rellenar agujeros o bahías que sean más pequeños que el EE. En la figura a continuación se observa la representación gráfica de una dilatación.

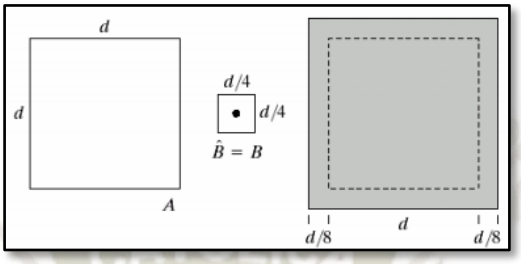


Figura 23. Dilatación de un conjunto A mediante el EE denominado B
Fuente: *(Gonzalez & Woods, 2002)*

- Erosión (comando *imerode*): Reduce (adelgaza) los píxeles que conforman un conjunto en una imagen digital, también interpretado como "el valor mínimo de la imagen en el entorno de vecindad definido por el elemento estructurante" (Platero Dueñas, 2013). Su sintaxis es:

  I2 = imerode(I, EE)

  La erosión es anti extensiva, debido a la reducción de tamaño de la imagen original. Se utiliza para eliminar elementos (islas o protuberancias) más pequeños que el EE. Está relacionado a la dilatación, pero no es su inversa. En la figura se observa su representación gráfica.

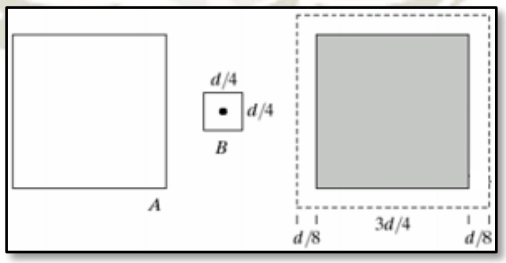


Figura 24. Erosión de un conjunto A mediante el EE denominado B
Fuente: *(Gonzalez & Woods, 2002)*

- Apertura (comando *imopen*): Combinación de una erosión seguida de una dilatación. Se realiza desplazando el EE por el interior del conjunto y eliminando las regiones de píxeles inalcanzables. En Matlab su sintaxis es la siguiente:

  I2 = imopen(I, EE)

La apertura es un operador anti extensivo, debido a que cumple la función de la erosión, pero evitando la reducción significativa de la imagen original. Es utilizado para redondear contornos, separar objetos y eliminar píxeles aislados. Puede ser idempotente al realizar numerosas veces. Se observa una representación gráfica en la siguiente figura:

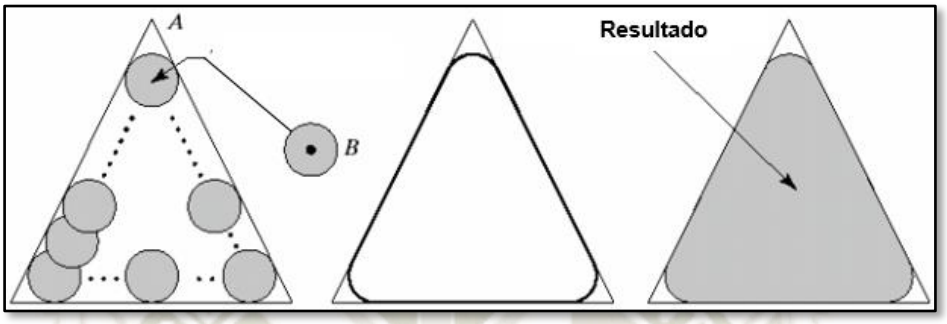


Figura 25. Apertura de un conjunto A mediante el EE denominado B
Fuente: *(Gonzalez & Woods, 2002)*

- Cierre (comando *imclose*): Combinación de una dilatación seguida de una erosión. Se realiza desplazando el EE por el exterior del conjunto y adicionando las regiones de píxeles inalcanzables. En Matlab su sintaxis es la siguiente:

I2 = imclose(I, EE)

El cierre es una operación extensiva, cumple la función de dilatación, pero adicionando la posibilidad de conectar objetos vecinos. Es utilizado para alisar contornos, rellenar agujeros y conectar elementos. Esta operación es idempotente y se puede realizar en par con la apertura, sin importar el orden. A continuación, una representación gráfica del cierre:

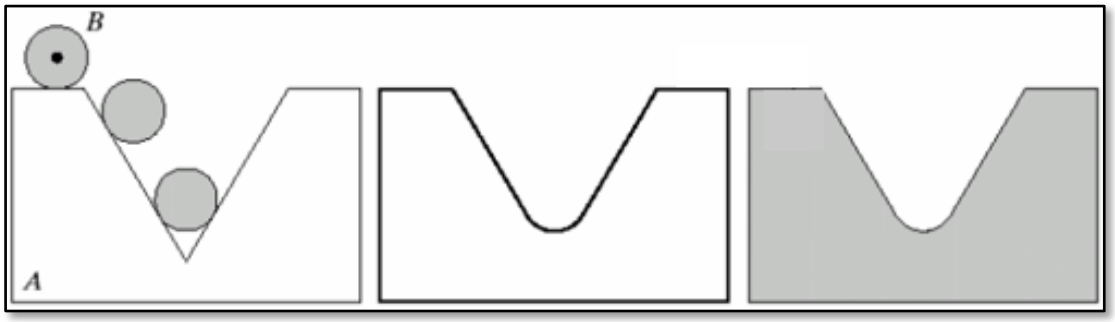


Figura 26. Cierre de un conjunto A mediante el EE denominado B
Fuente: *(Gonzalez & Woods, 2002)*

3.7.4. Filtros Morfológicos

Los filtros morfológicos tienen por función resaltar objetos o conjuntos de píxeles de un color distinto al fondo de la imagen según un EE definido. Estos se pueden realizar en modos de color binario o de escala de grises. Además, se debe considerar que al realizarse se pierde información, por lo que es irreversible. Se dividen en 2 tipos:

- Positivo (comando *imtophat*): Resalta detalles de una imagen en presencia de sombras (blanco sobre negro). Su sintaxis y ejemplo se muestran a continuación:

I2 = imtophat(I, EE)

Figura 27. Filtrado Morfológico Top Hat
Recuperado de: https://www.mathworks.com/help/pdf_doc/images/images_tb.pdf

- Negativo (comando *imbothat*): Resalta conjuntos oscuros sobre fondos claros (negro sobre blanco). Su sintaxis y ejemplo se muestra a continuación:

I2 = imbothat(I, EE)

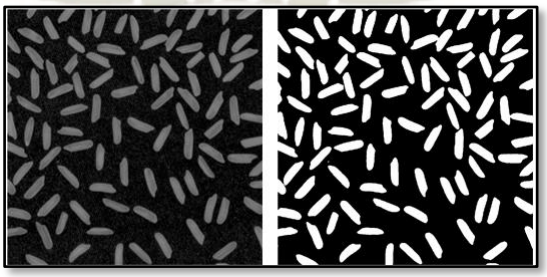


Figura 28. Filtrado Morfológico Bottom Hat
Recuperado de: https://www.mathworks.com/help/pdf_doc/images/images_tb.pdf

Matlab permite el uso de una herramienta general en imágenes binarias, que realiza operaciones o filtros morfológicos. Esta es el comando *bwmorph*, cuya sintaxis es:

I2 = bwmorph(I, operación)

Se puede recurrir a la siguiente tabla para las distintas secuencias de caracteres de *operación*.

Tabla 7
Operaciones del comando bwmorph

Operación	Descripción
'bothat'	Filtro morfológico 'Bottom Hat'
'clean'	Elimina píxeles aislados que no formen conjuntos
'close'	Operación morfológica 'Cierre'
'dilate'	Operación morfológica 'Dilatación'
'erode'	Operación morfológica 'Erosión'
'fill'	Rellena píxeles aislados que no formen conjuntos
'open'	Operación morfológica 'Apertura'
'tophat'	Filtro morfológico 'Top Hat'

Recuperado de: https://www.mathworks.com/help/pdf_doc/images/images_tb.pdf

4. Reconocimiento Óptico de Caracteres (OCR)

El OCR es un procedimiento que permite reconocer la parte textual de una imagen digitalizada a través de un escáner o aplicación, obteniendo como resultado un archivo de texto editable y utilizable en otra operación por un programa que reconozca y edite textos (Parker, 2010).

El reconocimiento de los caracteres necesita de una comparación de los caracteres segmentados de una imagen y la comparación de cada uno de ellos con un alfabeto traducido en patrones o plantillas de los posibles caracteres.

4.1. Problemática y Situación Actual

En una situación ideal, el OCR trabajará con una imagen de solo 2 niveles de gris claramente diferenciados, segmentado los caracteres necesarios y transformándolos según un "alfabeto" a un texto que exprese lo mismo de la imagen. Sin embargo, las imágenes obtenidas por dispositivos o programas no son perfectas, por lo que el OCR atraviesa ciertas barreras en su desarrollo:

- Fondo de la imagen: Varios niveles de grises que no corresponden a los caracteres.
- Resolución: Variable según el tipo de imagen y dispositivo, afectan a los píxeles a evaluar,
- Posición del texto: Genera errores en la evaluación del tamaño del carácter y su comparación.
- Ruido: Este provocado por el dispositivo o el programa, genera que se produzcan caracteres unidos o cortados que afectan la segmentación y el reconocimiento.

Aun así, este mecanismo es utilizado para la automatización de procesos que requieren la lectura de gran cantidad de imágenes y su traducción a textos, por ejemplo: identificación de códigos postales, matrículas de vehículos, tarjetas de personal, etc.

4.2. Metodología

El OCR tiene una metodología cimentada en 4 pasos:

- Binarización de la imagen
- Fragmentación o segmentación en caracteres
- Adelgazamiento de componentes
- Comparación de patrones

Se debe considerar en los 4 pasos listados la entrada como una imagen digitalizada ya sea a través de un dispositivo o programa (Adquisición), además de una salida de códigos que puede tomar la forma de caracteres aislados (Identificación) o de un texto (Interpretación).

4.2.1. Binarización

Como su nombre indica, consiste en pasar una imagen a un modo de color de mapa de bit, que indica el uso de un solo bit de información (blanco o negro), reduciendo el volumen de información y se preserven la información esencial de los caracteres. Su implementación requiere de realizar 2 pasos que dependen del modo de color utilizado en la imagen:

- De modo a color (RGB, Indexado) al modo de Escala de Grises: Se utiliza una fórmula que recoge la intensidad de colores y con el uso de pesos se transforma en intensidad de grises.

$$Igris = Peso1*Irojo + Peso2*Iverde + Peso3*Iazul$$

- De modo de Escala de Grises al modo de Blanco y Negro: A través del thresholding, que define un valor umbral de intensidad de gris que separa los valores de negro y blanco.

4.2.2. Segmentación

Es la operación que descompone la imagen en diferentes elementos conexos que la componen, siendo estos trozos de imagen con píxeles adyacentes entre sí, en forma de un recuadro. Su implementación es la más complicada, habiendo múltiples métodos y lógicas para su aplicación:

- Métodos explícitos (unidades físicas): se recorre la matriz ubicando un pixel negro. A partir de ello se recorre las columnas y filas hasta completar un recuadro que encierre los píxeles conexos entre sí ("border following"). Obtiene todos los elementos, pero no necesariamente de dimensiones similares para su comparación.
- Métodos implícitos (unidades lógicas): utiliza los renglones o cuadros con dimensiones especificadas previamente para la búsqueda de píxeles contiguos. Tiende a ser más rápido, debido a que encierra solo los caracteres cuyas plantillas se utilicen en la comparación, pero requiere de un cálculo previo para las dimensiones de dichos recuadros.

4.2.3. Adelgazamiento

Consiste en el borrado sucesivo de puntos de contorno de cada carácter, con el fin de reducir la información a la necesaria para su clasificación. Es necesario realizar un barrido en paralelo a todos los píxeles, eliminando los excedentes todos a la vez. Se definen los siguientes tipos:

- Punto Simple: La vecindad de 8 forma un solo componente conexo adyacente al punto P.
- Punto Final: Solo presenta un solo vecino negro en su vecindad de 8.
- Punto Aislado (Ruido): No presenta vecinos negros. Se debe considerar su eliminación.

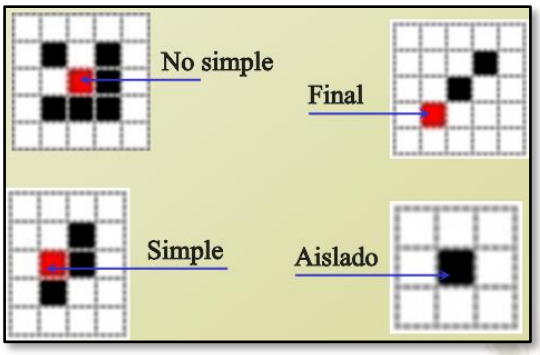


Figura 29. Clasificación de píxeles según su vecindad
Fuente: *(León Jiménez, 2014)*

4.2.4. Comparación

En la última etapa se comparan los caracteres obtenidos con los patrones o plantillas de caracteres teóricos que conforman el "alfabeto" o base de datos que define el texto a reconocer. Mientras mayor sea la precisión y cantidad de plantillas, será más exacta la comparación e clasificación de los caracteres, pero requerirá de un tiempo mayor para su procesamiento.

- Métodos geométricos o estadísticos
- Métodos estructurales
- Métodos neuromiméticos
- Métodos markovianos
- Métodos basados en IA

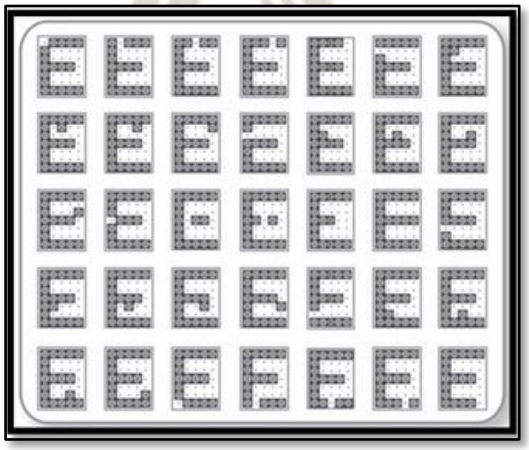


Figura 30. Patrones relacionados al carácter "E", con un 2.86% de ruido
Fuente: *(Polanco Carrillo, Álvarez Gerónimo, & Moreno Vega, 2015)*

5. Redes Neuronales Artificiales (RNA)

Las Redes Neuronales son un campo muy importante dentro de la Inteligencia Artificial. Inspirándose en el comportamiento conocido del cerebro humano (principalmente el referido a las neuronas y sus conexiones), trata de crear modelos artificiales que solucionen problemas difíciles de resolver mediante técnicas algorítmicas convencionales.

Es conocido como RNA el modelo computacional simplificado que trata de imitar el comportamiento de las neuronas en el cerebro humano para el procesamiento de información.

5.1. Modelo de Neurona Artificial

El modelo estándar desarrollado por Rumelhart y McClelland en los libros "Procesamiento Distribuido Paralelo" (o PDP) en 1986, define una neurona artificial como un elemento de proceso que, a partir de un conjunto de entradas o vector xi (i=1...n), genera una única salida y.

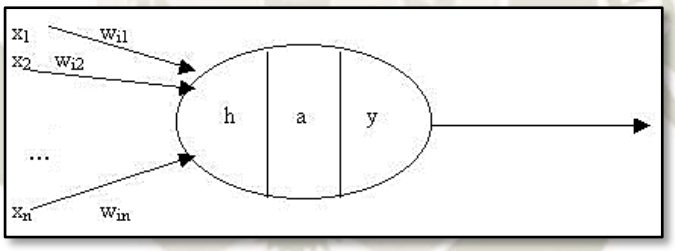


Figura 31. Modelo de una Neurona Artificial
Fuente: *(Romero, 2010)*

Esta neurona artificial consta de los siguientes elementos:

- **Conjunto de entradas o vector de entradas (x)**: de n componentes
- **Conjunto de pesos sinápticos (wij)**: Representan la interacción entre la neurona anterior "j" y la posterior "i" a la sinapsis o "enlace".
- **Regla de propagación d(wij,xj(t))**: proporciona el potencial postsináptico, hi(t).
- **Función de activación ai(t)=f(ai(t-1)**: proporciona el estado de activación de la neurona en función del estado anterior y del valor postsináptico.
- **Función de salida Fi(t)**: proporciona la salida yi(t), en función del estado de activación.

Las señales de entrada y salida pueden ser señales binarias (0,1), bipolares (-1,1), números enteros o continuos, variables borrosas, etc.

La regla de propagación tiene distintos tipos, los cuales pueden ser:

- Suma ponderada del producto escalar del vector de entrada y el vector de pesos.

$$h_i(t) = \sum w_{ij} \cdot x_j$$

- Distancia euclídea entre ambos vectores.

$$h_i(t) = \sum (x_j \cdot w_i)^2$$

- Distancia de Voronoi
- Distancia de Mahalanobis

La función de activación normalmente no considera la "historia" o estados anteriores de la neurona, sino el valor del potencial hi(t). Generalmente se representa en una función determinista, continua y creciente. Las funciones más utilizadas son las mostradas a continuación:

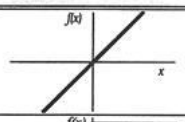
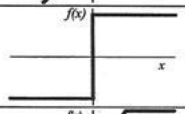
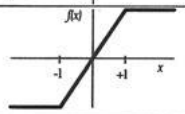
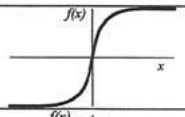
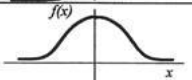
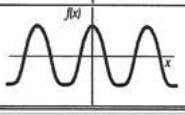
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

Figura 32. Funciones de Activación para RNA
Fuente: *(Romero, 2010)*

La función de salida suele ser la identidad. En algunos casos es un valor umbral (la neurona no se activa hasta que su estado supera un determinado valor).

Con todo esto, el modelo de neurona queda bastante simplificado:

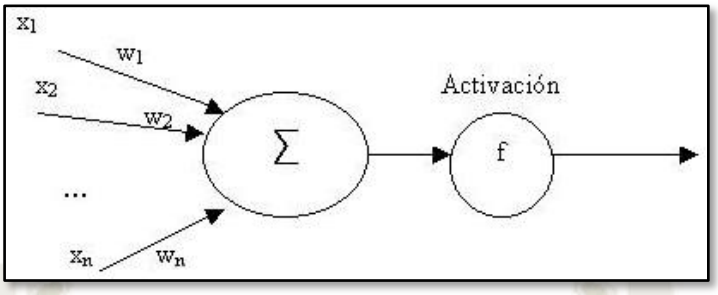


Figura 33. Modelo de Neurona Simplificado
Fuente: *(Romero, 2010)*

5.2. Red Neurona Artificial

Una red neuronal artificial (RNA) como perceptrón fue desarrollada por Rosenblatt en 1957, sin embargo se puede definir (Hecht-Nielssen, 1990) como un grafo con las restricciones:

a. Los nodos se llaman elementos de proceso (EP).
b. Los enlaces se llaman conexiones y funcionan como caminos unidireccionales instantáneos
c. Cada EP puede tener cualquier número de conexiones.
d. Todas las conexiones que salgan de un EP deben tener la misma señal.
e. Los EP pueden tener memoria local.
f. Cada EP posee una función de transferencia que, en función de las entradas y la memoria local produce una señal de salida y/o altera la memoria local.
g. Las entradas a la RNA llegan del mundo exterior, mientras que sus salidas son conexiones que abandonan la RNA.

5.3. Arquitectura de las RNA

La arquitectura de una RNA es el modelo o patrón de conexiones de la red. Es por ello que las conexiones sinápticas son direccionales, es decir, la información sólo se transmite en un sentido.

En general, las neuronas suelen agruparse en unidades estructurales llamadas capas. Dentro de una capa, las neuronas suelen ser del mismo tipo. Se pueden distinguir tres tipos de capas:

- De entrada: reciben datos o señales procedentes del entorno.
- De salida: proporcionan la respuesta de la red a los estímulos de la entrada.
- Ocultas: no reciben ni suministran información al entorno (procesamiento interno de la red).

Generalmente las conexiones se realizan entre neuronas de distintas capas, pero puede haber conexiones intracapa o laterales y conexiones de realimentación que siguen un sentido contrario al de entrada-salida.

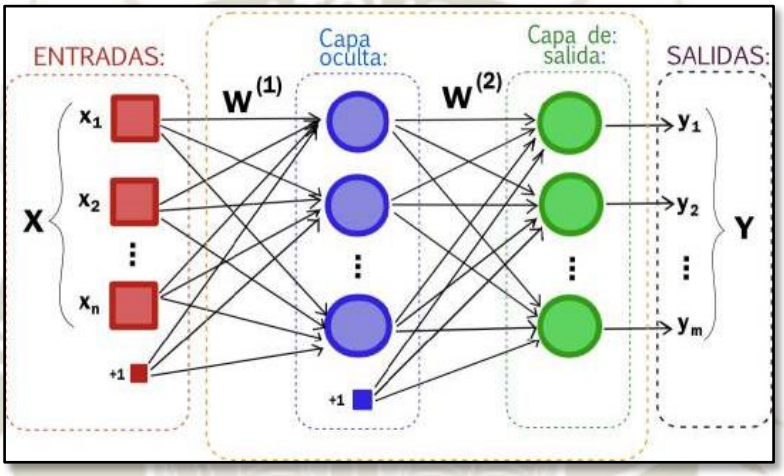


Figura 34. Arquitectura de RNA multicapa
Fuente: *(Romero, 2010)*

5.4. Tipos de RNA Multicapa

Las redes multicapa están formadas por varias capas de neuronas (2,3...). Estas redes se pueden a su vez clasificar atendiendo a la manera en que se conexionan sus capas.

Usualmente, las capas están ordenadas por el orden en que reciben la señal desde la entrada hasta la salida y están unidas en ese orden. Ese tipo de conexiones se denominan conexiones feedforward o hacia delante.

Por el contrario, existen algunas redes en que algunas capas están fuera del orden normal, conectando la salida a la entrada en el orden inverso en que viajan las señales de información. Las conexiones de este tipo se llaman conexiones hacia atrás, feedback o retroalimentadas.

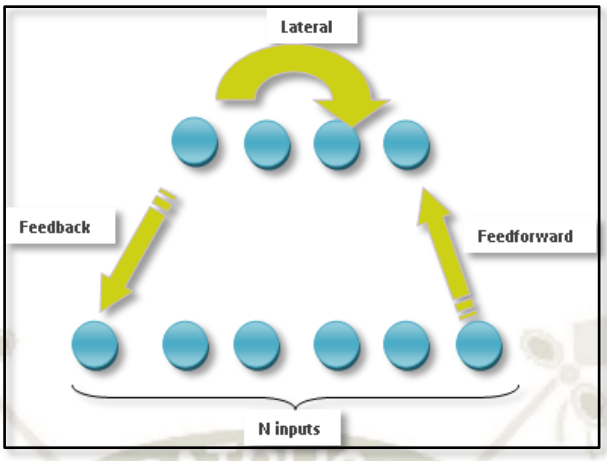


Figura 35. Tipos de conexiones en RNA multicapa
Fuente: *(Romero, 2010)*

5.5. Aprendizaje de las RNA

Es el proceso por el que una RNA actualiza los pesos (y, en algunos casos, la arquitectura) con el propósito de que la red pueda llevar a cabo de forma efectiva una tarea determinada.

Hay tres conceptos fundamentales en el aprendizaje:

- Paradigma de aprendizaje: información de la que dispone la red.
- Regla de aprendizaje: principios que gobiernan el aprendizaje.
- Algoritmo de aprendizaje: procedimiento numérico de ajuste de los pesos.

Existen dos paradigmas fundamentales de aprendizaje:

- Supervisado: la red trata de minimizar un error entre la salida que calcula y la salida deseada (conocida), de modo que la salida calculada termine siendo la deseada.
- No supervisado o auto-organizado: la red conoce un conjunto de patrones sin conocer la respuesta deseada. Debe extraer rasgos o agrupar patrones similares.

El método de aprendizaje define el tiempo y carga computacional necesario para entrenar a una red neuronal. Estos han ido evolucionando con los años, pero se pueden agrupar en 3 grupos: gradiente, Newton y Levenberg-Marquardt. Se comparan estos métodos en la siguiente figura, considerando velocidad (speed) y memoria necesaria (memory).

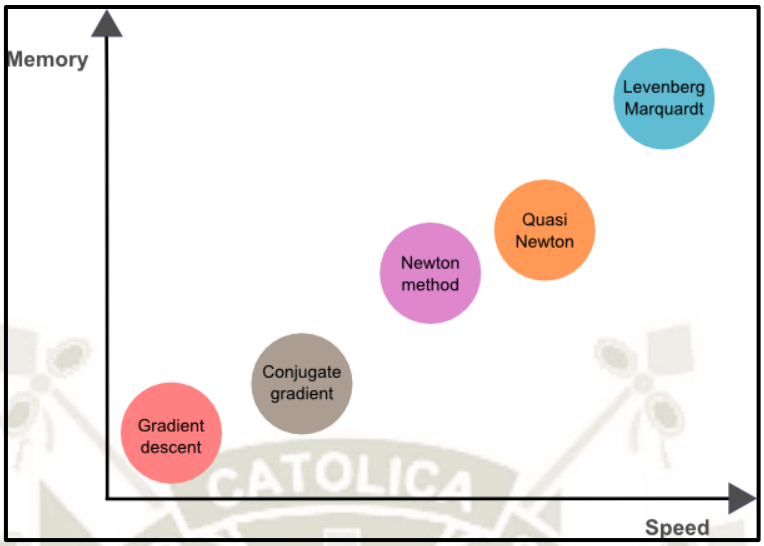


Figura 36. Comparación de métodos de aprendizaje
Recuperado de: http://www.cs.us.es/~fsancho/?e=165

- Descenso de gradiente: el más simple y más conocido, conocido como de primer orden. Tiene como objetivo reducir el vector gradiente, a partir de calcular un punto (peso) w_{i+1} de un punto anterior w_i siguiendo una tasa de entrenamiento v_i, según la fórmula:

$$w_{i+1} = w_i - g_i v_i$$

  Requiere poco cálculo, pero presenta un gran problema para funciones con errores muy grandes, debido a que se eleva en gran medida las iteraciones y pueden no ser muy precisas.

- Método de Newton: conocido como de segundo orden, requiere calcular la matriz Hessiana, para ello hace uso del desarrollo de Taylor de orden 2, y buscar aproximar la gradiente a 0:

$$w_{i+1} = w_i - {H_i}^{-1} g_i$$

  Utiliza menos cantidad de iteraciones, pero tiene un cálculo computacional muy alto para obtener tanto la Hessiana como su inversa.

- Gradiente Conjugado: Siguiendo el método del descenso de gradiente, pero agregando un parámetro conjugado (y_i) a la gradiente para dar una dirección de entrenamiento, la cual se restablece para evitar acumulación de errores. No requiere cálculo de Hessiana.

$$d_{i+1} = g_{i+1} + d_i \gamma_i$$

- Cuasi-Newton: bajo el esquema de Newton, pero en lugar de calcular la inversa (Gi) directamente, se busca una aproximación para cada iteración con la ayuda de las primeras derivas de la función de error. Este se utiliza como método por defecto para este cálculo.

$$w_{i+1} = w_i - (G_i g_i)v$$

- Algoritmo de Levenberg-Marquardt: se utiliza para funciones de error expresado como suma de errores cuadráticos. Hace uso del vector gradiente y la matriz Jacobiana.

$$f(w) = \sum_{i=0}^{m} e_i^2\,(w) \qquad \text{y} \qquad J_{ij}(w) = \frac{\partial e_i}{\partial w_j}$$

6. Implementación de Cámaras de Vigilancia

En la actualidad, las cámaras de video vigilancia se han convertido en un recurso muy utilizado y requerido por los usuarios. La razón de ello es principalmente por los actos delictivos que azotan nuestro país, proveyendo una sensación de seguridad y tranquilidad mucho mayor. Sin embargo, su uso se ha extendido a diversos ámbitos públicos y privados de nuestra comunidad, debido a su versatilidad en el reconocimiento de elementos del ambiente: objetos y/o personas.

Es debido a la tecnología creciente en el tiempo que las nuevas cámaras proveen aplicaciones que complementan e integran diferentes soluciones para ofrecer un mejor servicio, entre ellas:

- Reconocimiento facial
- Reconocimiento de placas vehiculares
- Seguimiento de objetos para rastreo y seguridad
- Integración con sistemas de control de acceso, alarmas de fuego, etc.

6.1. Cámaras Analógicas

Una cámara analógica es tradicionalmente utilizada en los sistemas CCTV (Circuito cerrado de televisión), que tienen como principio la supervisión de grupos de ambientes y elementos.

Se le llama “circuito cerrado” debido a que todos sus componentes están conectados, logrando una cantidad limitada de espectadores que pueden acceder a estas imágenes.

Sus principales componentes son:

- Cámaras: dispositivos que captan la imagen del lugar protegido, pueden ser fijas o móviles.
- Monitor: componente externo que permite proyectar las imágenes captadas en la cámara.
- Medio de transmisión: puede ser cable coaxial o par entrelazado (mayores distancias), realizando también la alimentación del circuito.
- Dispositivo de almacenamiento de video: para uso local en DVR (Digital Video Recorder) o para redes IP en NVR (Network Video Recorder).

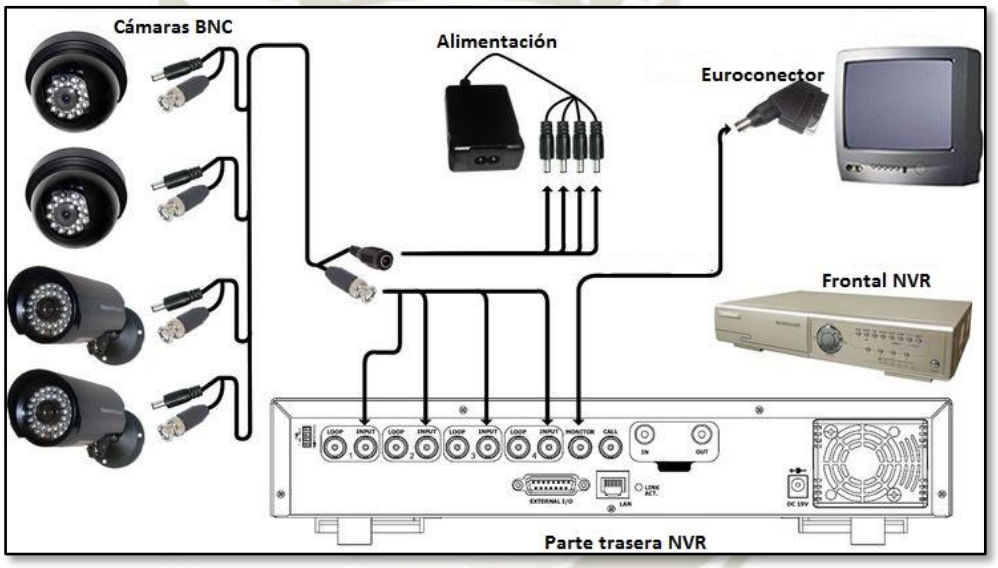


Figura 37. Esquema de conexión cámaras analógicas
Recuperado de: http://zoominformatica.com/blog/diferencias-entre-camaras-ip-y-sistemas-cctv/

El precio de las cámaras analógicas suele ser más económica debido a la tecnología utilizada, que se reduce a captar y enviar al grabador para su procesamiento, el cual es obligatorio para el circuito, lo cual aumenta el gasto en la instalación.

6.2. Cámaras IP

La video vigilancia IP es una tecnología que permite la transmisión de imágenes de una cámara directamente al intranet o internet sin la necesidad de un ordenador para el procesamiento de datos. Además de ello ofrece una calidad de imagen superior (HD 720p o 1080p), además de una accesibilidad desde cualquier lugar donde se cuente con una red o acceso a internet.

Entre las principales tecnologías que ofrece está:

- Conexión inalámbrica: necesitando solo la alimentación para su visualización.
- POE (Power Over Ethernet): el uso de conmutadores Ethernet simplifica la instalación de una cámara IP, debido a que envía datos y corriente eléctrica por un mismo cable.
- Firewall: impide el acceso indebido a la red de vigilancia IP.
- Software de análisis de vídeo: permite análisis automáticos de las imágenes en función de los parámetros previamente definidos por el usuario, que permiten grabaciones y transmisiones personalizadas, lo cual permite un mayor control de almacenamiento y consumo de ancho de banda desde una manera centralizada o remota gracias al protocolo IP.
- Aplicaciones adicionales: reconocimiento facial, detector de movimiento, infrarojo, etc.

Este tipo de cámaras se acoplan perfectamente en empresas o lugares donde se posee una infraestructura de red ya instalada, dado que se aprovecha el puerto de red más cercano disponibles y puede ser instalado con la necesidad de un solo cable de red. La cantidad de cámaras está influenciada por el número de puntos de red disponible.

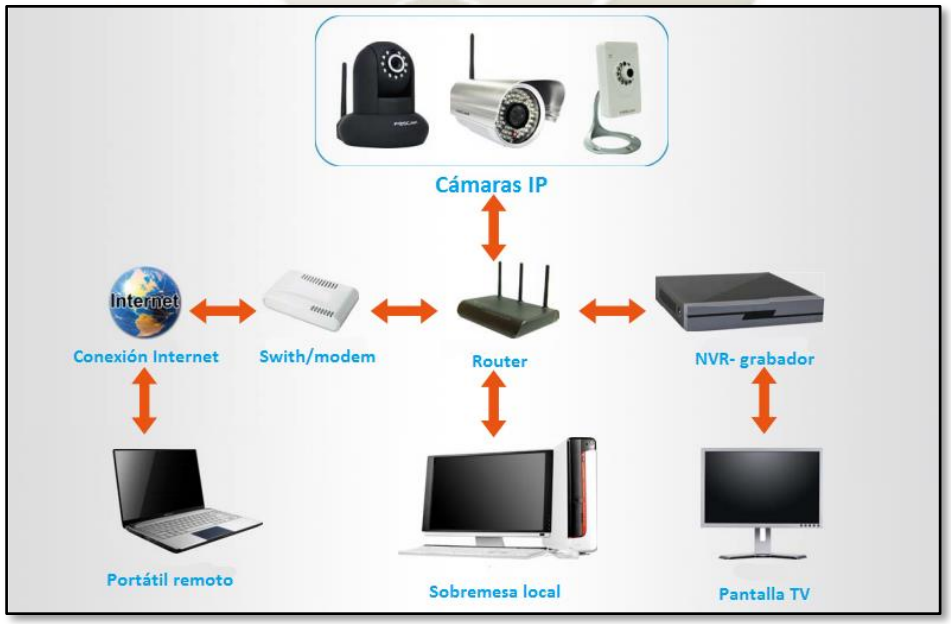


Figura 38. Esquema de conexión cámaras IP
Recuperado de: http://zoominformatica.com/blog/diferencias-entre-camaras-ip-y-sistemas-cctv/

6.3. Comparación de alternativas

Tabla 8

Comparación de Cámaras Analógicas y Cámaras IP

Cámaras Analógicas o CCTV	Cámaras IP
• Las cámaras son analógicas en sí y lo que nos permite verlas por Internet es el DVR o un servidor de video. • Las cámaras por sí solas no detectan movimiento y no son "inteligentes". • En digital se requiere hacer un cableado "separado" para video, para audio y para corriente eléctrica. • La calidad de video es buena, en ocasiones al digitalizarla se afecta un poco.	• Las cámaras IP son digitales y se pueden ver por Internet conectándolas directamente a un router, por ser un recurso de red. • Son "inteligentes", incorporando funciones de alarma, detección de movimiento, etc. • Algunas cámaras de alta gama son POE permitiendo transmitir tanto el video, audio y la corriente eléctrica por el mismo cable de red. • Suelen manejar una excelente calidad de video, debido a su forma digital nativa.

Fuente: Elaboración Propia

CAPITULO III: DISEÑO E INGENIERÍA

1. Esquema de Diseño del Sistema

El diseño de un sistema en Ingeniería comprende un proceso de toma de decisiones, a menudo secuencial, en la cual los conocimientos de ciencias físicas y matemáticas son aplicados a transformar recursos en forma óptima para satisfacer deseos o exigencias de un proyecto.

Con la finalidad de implementar un sistema de identificación vehicular, se empieza por dividir el sistema en subsistemas o etapas, los cuales presentan características a considerar. Esto se muestra en el siguiente gráfico:

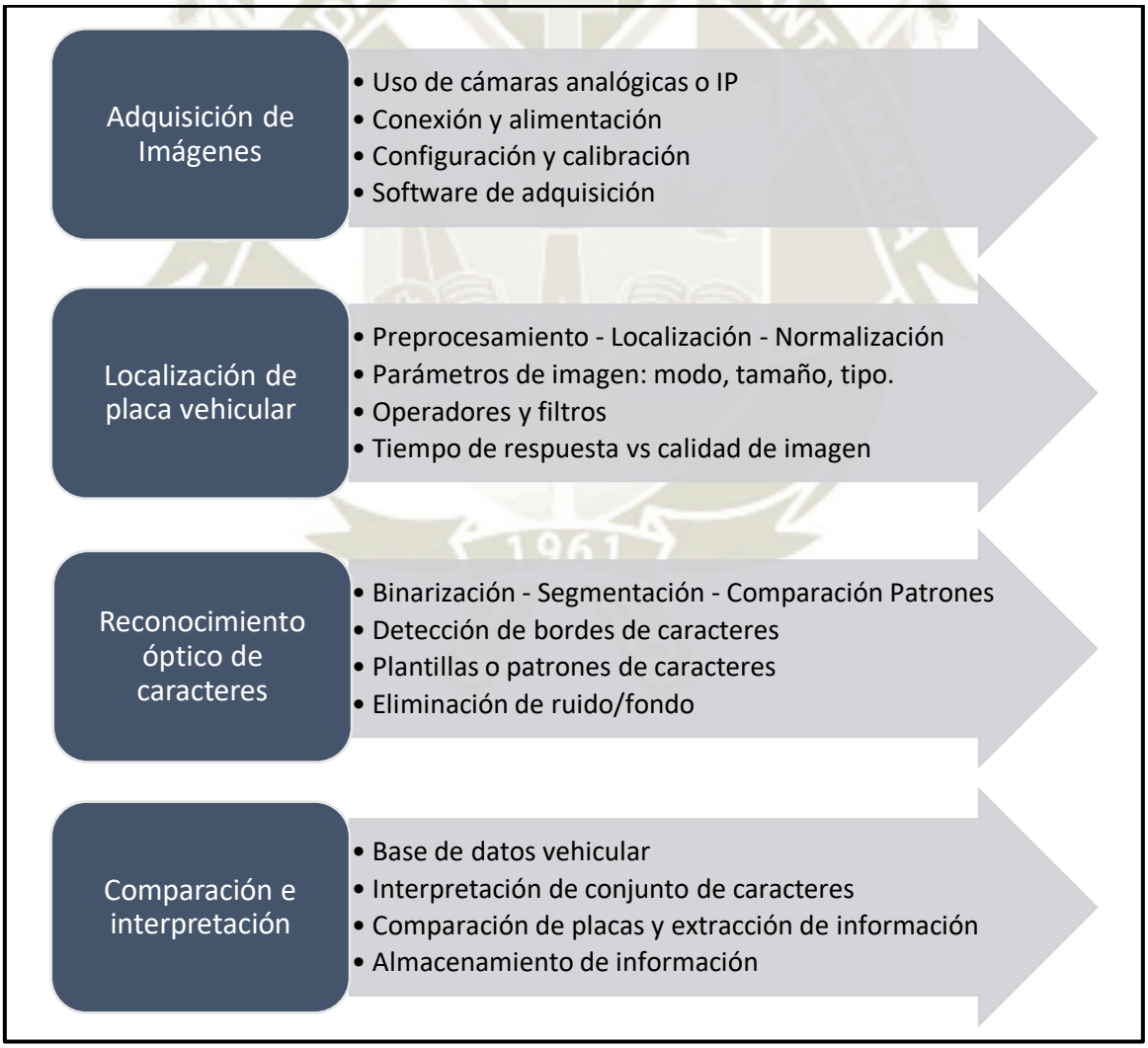


Figura 39. Modelo del sistema y características
Fuente: Elaboración Propia

1.1. Cuadro de Requerimientos

El cuadro de requerimientos es una herramienta que permite reunir las condiciones que el usuario o el diseñador quiere plasmar en un sistema. Además, se utilizó características globales que permiten dividir en deseos o exigencias según los objetivos del proyecto.

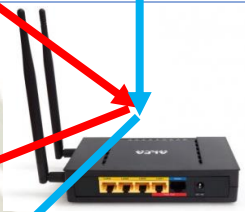
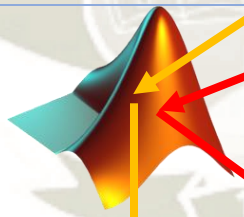
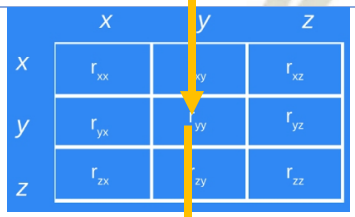
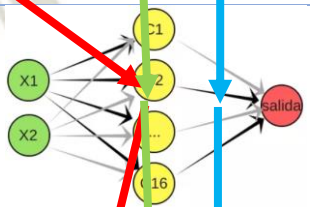
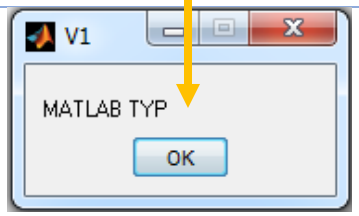
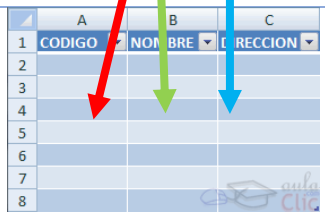
Tabla 9
Cuadro de Requerimientos

Características	Deseo o Exigencia	Condiciones
Aplicación	E	El sistema debe identificar el ingreso de vehículos mayores (autos, camionetas, etc.) en la Universidad Católica de Santa María.
	E	Se debe poder visualizar información del vehículo y su propietario.
Imagen	E	Se debe observar imágenes en tiempo real para su procesamiento.
	E	La imagen debe ser digital debido al tamaño y versatilidad.
	D	Se necesita una calidad decente de resolución (mínimo 720p).
Software	E	Debe ser versátil y compatible con la adquisición de la imagen.
	E	Dispondrá de herramientas para el Procesamiento Digital de Imágenes
	D	Permitirá la creación y enseñanza de RNA sin necesidad de programas adicionales
Comunicación	E	La comunicación debe ser constante y con el menor retraso posible.
	E	La red de conexiones no debe ser compleja y debe permitir la adición de mayor cantidad de elementos.
	D	Utilizar la conexión POE como alimentación y comunicación.
Base de Datos	E	Su tamaño debe ser reducido y fácil de entender y editar.
	D	La información recopilada debe ser accesible para el software.
	D	Las plantillas deben simular un gran número de condiciones de disposición y tamaño de caracteres.
Seguridad	E	La información o parámetros de la cámara o imagen no deben ser editables sin autorización de un usuario certificado.
	D	Los errores de ejecución del software no deben generar pérdida de información ya procesada.
Montaje	E	El sistema debe ser sencillo de instalar y de transportar.
	D	El sistema tiene que ser accesible desde cualquier ordenador
Costo	E	Los costos de cámara, cableado y comunicación no deben ser elevados
	D	Estos elementos deben ser accesibles a cualquier usuario.

1.2. Matriz de Selección

La siguiente matriz contiene los principales elementos que conforman el sistema de identificación de vehículos, mostrando opciones y sus características en el desarrollo del proyecto.

Tabla 10
Matriz de Alternativas

FUNCION	***ALTERNATIVAS***		
	1	***2***	***3***
Adquisición de Imágenes	 Analógica	 IP Alámbrica	 IP Inalámbrica
Método de Comunicación	 NVR	 Inyector POE	 Switch Router
Software de Programación	 MatLab	 Visual Studio	
Herramientas de OCR	 Método geométrico	 Método basado en IA	
Tipos de Salida	 *Interfaz*	 *Excel*	

Fuente: Elaboración Propia

A continuación, se procederá a evaluar las 4 alternativas obtenidas de la tabla anterior con valoraciones en 3 escalas para los siguientes criterios:

Tabla 11
Matriz de Selección

Criterios	Propuesta A (verde)	Propuesta B (naranja)	Propuesta C (rojo)	Propuesta D (azul)
Aplicación (calidad imagen)	Regular	Buena	Buena	Regular
Autonomía (duración y confiabilidad)	Buena	Regular	Buena	Mala
Software de programación	Bueno	Regular	Regular	Bueno
Adquisición y programación de Toolbox	Regular	Fácil	Fácil	Regular
Seguridad (programa)	Buena	Regular	Regular	Buena
Montaje de equipos	Malo	Regular	Regular	Bueno
Costo total	Alto	Bajo	Regular	Alto
RESULTADO	Descartado	Seleccionada Opción 1	Seleccionada Opción 2	Descartado

Fuente: Elaboración Propia

Según lo observado en la matriz de selección, se seleccionaron 2 opciones para su desarrollo durante el proyecto, debido a sus ventajas sobre las otras 2 alternativas. Estas comparten el software de programación MatLab y el tipo de cámara IP alámbrica, el cual es más utilizado en la actualidad para aplicaciones de seguridad por periodos de tiempo extensos.

2. Adquisición de Imágenes

En este apartado se busca como objetivo obtener imágenes del mundo real con una calidad decente para su procesamiento por medio de un software. El software elegido fue el MatLab por su facilidad de trabajo, además del conocimiento y experiencia que se tiene sobre sus herramientas.

Se consideró el siguiente método de trabajo para la creación de un programa que permitiera la adquisición de imágenes:

- Lectura de imágenes almacenadas en archivos.
- Pruebas con el tipo de conexión IP
- Sistemas de visualización de cámaras IP instaladas.

Cada uno de estas etapas se trabajó mediante un programa con código particular, que permitía su aplicación para pruebas en los siguientes subsistemas de nuestro proyecto, validando la versatilidad de estas herramientas según los requerimientos del usuario.

2.1. Lectura de Imágenes en Archivos

La primera etapa corresponde al método más sencillo de visualización de imágenes, debido a la facilidad de trabajo con archivos almacenados, además de su reutilización para la configuración de parámetros en el software o hardware.

Las características deseadas para este tipo de sistema se citan a continuación:

- Imagen de Mapa de Bits: el más utilizado por las cámaras actuales. Sencillo de trabajar.
- Formato de imagen compatible con MatLab y obtenible mediante cámaras: JPEG, PNG, etc.
- Modo de Color RGB o escala de grises: capaz de mostrar diferencias entre elementos.
- Resoluciones máximas de 1280x720, debido al tamaño que ocupan estos archivos y que aumentan considerablemente el procesamiento de pixel por pixel.
- Imágenes correctamente capturadas, evitando problemas que son posible evitar como: posiciones complejas o mal enfocadas de los vehículos.

Para la creación de los programas se necesitó la instalación de bibliotecas, enfocadas a la adquisición y procesamiento de archivos de imágenes, que en MatLab son definidos como:

- "Image Acquisition Toolbox" proporciona funciones y bloques que habilitan la conexión de cámaras industriales y científicas a MATLAB y Simulink. Esto incluye una app que te permiten detectar y configurar propiedades de hardware de manera interactiva. Esta librería habilita modos de adquisición tales como procesamiento en bucles, activación por hardware, adquisición de fondos y sincronización en la adquisición por múltiples dispositivos.
- "Image Processing Toolbox" proporciona un conjunto completo de algoritmos estándar de referencia y apps de flujo de trabajo para el procesamiento, el análisis y la visualización de imágenes, así como para el desarrollo de algoritmos. Puede llevar a cabo segmentación de imágenes, mejora de imágenes, reducción de ruido, transformaciones geométricas, registro de imágenes y procesamiento de imágenes 3D.

Las primeras herramientas a utilizar para este fin son los comandos "imread" y "imshow". Para ello se necesitará contar con 2 condiciones: el nombre y ubicación de los archivos. En ambos casos son variables controlables por el usuario. Un ejemplo simple que lo representa es el siguiente:

```
imagen = imread('carro.jpg')        % Lee el archivo de nombre "carro"
imshow(imagen)                      % Muestra la variable imagen
```

La ubicación de los archivos debe estar disponible, para ello en caso de no contar con las imágenes en la carpeta de instalación de MATLAB, utilizar la herramienta "Set Path" 

ubicado en la categoría de "Environment" del cajetín de "Home" para agregar dicha ubicación.

Otros comandos útiles para la visualización o almacenamiento de imágenes a procesar son:

- Cambio de resolución "imresize"
- Guardar imagen "savefig" o "imwrite"
- Información de la imagen "imfinfo"

2.2. Conexión IP y pruebas

La segunda etapa es la visualización de imágenes en tiempo real a partir de la conexión IP incorporada en cámaras y otros dispositivos. Entre las características resaltantes están:

- Transmisión de imágenes a través de internet o intranet sin necesidad de un ordenador.
- Accesibilidad desde cualquier ordenador con acceso a internet o la red local.
- Calidad de imagen superior: valores de 720p o mayores.

En primer lugar, se necesita de una cámara IP para la grabación de video, sin embargo, por motivos de comodidad y versatilidad, se necesitaba realizar pruebas en un equipo portable. Para cumplir dicho fin, se investigó la posibilidad de visualizar imágenes grabadas por el celular utilizando esta tecnología. La respuesta a ello fue la aplicación "IP Webcam", desarrollada por Pavel Khlebovich, para dispositivos Android 4.1 o superiores.

Según la descripción disponible en Play Store, "IP Webcam convierte su celular en un cámara en red con múltiples opciones de visualización. Funciona en cualquier plataforma de ordenador, además de reproductores de vídeo como VLC Player o el navegador web. Emite a través de una red WiFi sin conexión a internet".

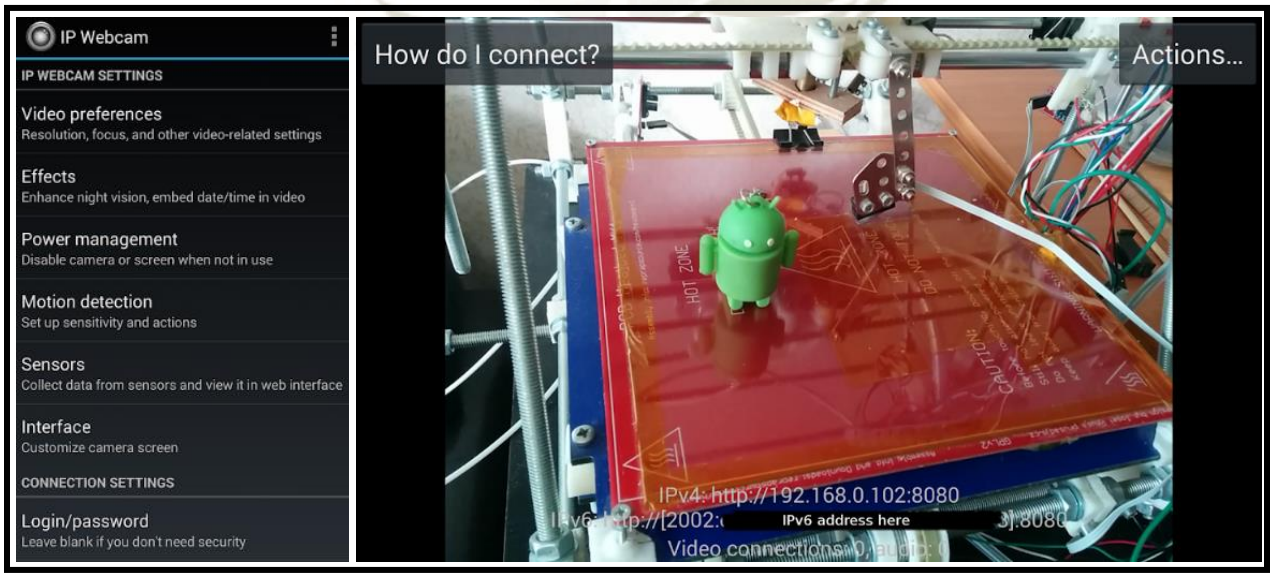


Figura 40. Configuración y visualización de la app "IP Webcam"
Recuperado de: https://www.appbrain.com/dev/Pavel+Khlebovich/

Al ser una aplicación gratuita, no permite la generación de múltiple direcciones IP de visualización, sin embargo, se puede conectar a una red local a través de WiFi. La dirección por defecto es "192.168.43.1:8080", para declarar una red de clase C, la cual permite una conexión de hasta un máximo de 254 hosts o usuarios de red, incluyendo la de la cámara IP.

Luego de declarar la dirección de red a utilizar, es necesario realizar su lectura por medio de un ordenador. Dicha visualización se realiza de manera sencilla por los siguientes 2 métodos:

- Navegador web: Ingrese la dirección IP y puerto que utiliza la cámara la barra de direcciones.
- Reproductor VLC: Ubicar la herramienta "Abrir ubicación de red" en la pestaña "Medio" e ingresar la dirección IP y puerto de la cámara.

Sin embargo, el método deseado para el procesamiento de las imágenes es el software MATLAB. Para ello se utiliza 2 procedimientos según la necesidad del usuario:

2.2.1. Visualización de vídeo

En este procedimiento se considera la visualización completa de la transmisión de la cámara IP. Para ello se utilizará el comando "ipcam" para crear un objeto que represente dicha cámara. Este comando requiere de 2 datos mínimos a ingresar: la dirección URL de transmisión y el formato de video utilizado por la cámara. Sin embargo, es posible generar un usuario y contraseña de acceso. A continuación, un ejemplo del uso de este comando:

```
cam = ipcam('http://172.28.17.193/video.mjpeg', '', '', 'Timeout', 20)

cam =

Display Summary for ipcam:

            URL: 'http://172.28.17.193/video.mjpeg'
       Username: ''
       Password: ''
        Timeout: 20
```

Figura 41. Creación de un objeto de cámara IP en MATLAB
Recuperado de: https://www.mathworks.com/help/pdf_doc/images/images_tb.pdf

Como se observa en la imagen, un dato más a considerar es la variable "Timeout", la cual define la cantidad de tiempo en segundos que el programa espera por la información a trasmitir, o dicho de otra manera, el tiempo máximo de espera para conectarse a la cámara. Por defecto su valor es 10 segundos, pero puede ser modificado ya sea en la creación del objeto o después de dicha acción, por ejemplo: "cam.Timeout = 25", siendo "cam" el objeto referido a la cámara.

La dirección URL generalmente es entregada por el software de la cámara o puede ser configurada en la misma, pero empezará con el comando "http" que abarca MJPEG o RTSP stream. Sin embargo, la diferencia entre estos formatos de video no es siempre accesible a simple vista, por lo que es necesario obtenerlo por otros métodos, algunos de ellos se citan a continuación:

- Buscar en la interfaz de la cámara web, también considerar la página web del fabricante.
- En caso del navegador web Mozilla Firefox, al ingresar a la dirección URL, hacer clic derecho en el buscador y seleccionar la opción "Ver información de imagen/página".
- Reproductor VLC: Ingresar la dirección URL sin el formato de video en la herramienta "Abrir ubicación de red" de la pestaña "Medio", luego ir a la herramienta "Ver información de códec" en la pestaña "Herramientas". Chequear el formato de video utilizado.

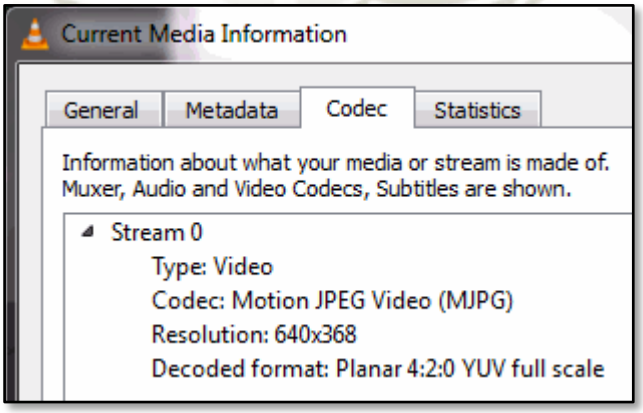


Figura 42. Formato de video obtenido a través de VLC Player
Recuperado de: https://www.mathworks.com/help/pdf_doc/images/images_tb.pdf

La siguiente tabla muestra algunas modelos de cámaras IP populares y la sintaxis de su dirección URL. Se puede sustituir la dirección web de la cámara en lugar de "<IP address:port>".

Tabla 12
Tabla de direcciones URL según marca de cámara

Marca	N° de Modelo (ejemplo)	Dirección URL
FOSCAM	FI9821W V2	http://<IP address:port>/cgi-bin/CGIStream.cgi?cmd=GetMJStream
DLINK	DCS-2132L	http://<IP address:port>/video1.mjpg
TRENDNET	TV-IP572WI	http://<IP address:port>/video/mjpg.cgi
VIVOTEK	IB8168	http://<IP address:port>/video.mjpg
AXIS	0519-004	http://<IP address:port>/mjpg/video.mjpg
SONY	SNC-CH110	http://<IP address:port>/mjpeg

Fuente: Elaboración Propia

Para visualizar la cámara se utilizan los siguientes comandos:

- preview(cam): Visualiza el video transmitido por la cámara en una ventana emergente.
- closePreview(cam): Cierra la ventana de visualización, sin eliminar la transmisión.
- img = snapshot: Adquiere la imagen actual de la cámara y la almacena en una variable.
- imshow(img): Muestra la imagen almacenada en la variable.
- clear cam: Liberar la variable, deteniendo la transmisión.

Entre las ventajas de este método está la pérdida nula de información debido a que se observa el video de transmisión completo, además de permitir la creación de usuarios autorizados, restringiendo la entrada a este. Aun así, se debe considerar que el tamaño de información es muy alto por su naturaleza de video, aunque puede ser mitigado cambiando la resolución de la cámara, además de generar errores constantes en caso de pérdida de conexión con la cámara.

2.2.2. Visualización de imágenes continuas

Este método está centrado en la reducción de información que genera utilizar imágenes capturadas en intervalos de tiempo en vez del formato de video adquirido por la cámara. Esto se traduce a menores tiempos de procesamiento y cantidad de recursos necesarios por el ordenador para mantener el programa funcionando.

Para ello es necesario utilizar el comando “imread”, sin embargo, la principal diferencia con respecto a la visualización de imágenes almacenadas, es que se utilizará la dirección URL en vez del nombre del archivo, agregando al final el formato de imagen entregado por la cámara o configurado en la interfaz de esta. Además de ello es necesario generar una matriz de imagen para insertar la información en ella cada vez que se cumple un intervalo de tiempo de captura.

El siguiente programa simple demuestra este método:

```
URL = 'http://192.168.1.90/jpg/image.jpg';        % Dirección de la cámara
info = imread(URL);                               % Adquisición de información
imagen = image(info);                             % Crea la matriz de imagen
while(1)                                          % Creación de un bucle
  info = imread(URL);
  set(imagen,'CData',info);                       % Inserta la información en la matriz
  imshow(imagen);                                 % Muestra la imagen en una ventana
  pause(10)                                       % Genera un intervalo de 10 seg.
end
```

2.3. Sistema de Visualización con Cámaras IP

La tercera etapa es la selección de una cámara IP que sea la base de nuestro programa. Como se indicó anteriormente, se desea utilizar la tecnología IP por ser configurable y accesible por cualquier ordenador con acceso a Internet, además de entregar buena calidad de imagen.

En el mercado se encontraron distintas cámaras con esta tecnología, las cuales varían sus precios según la marca y las características que poseen. Entre las principales marcas que son accesibles en rango de calidad de imagen y precio se encuentran: TP-Link, Dahua y Hikvision.

Gracias a la ayuda de la Universidad se pudo acceder a una cámara Axis modelo M1145-L, la cual cumple de sobremanera los requerimientos planteados, además de agregar funcionalidades que mejoran el rendimiento de la cámara para distintas situaciones. Se procede a mencionar algunas de las más resaltantes:

- Calidad de vídeo desde 160x90 hasta 1920x1080 (HDTV) con 2 MP.
- Capacidad de zoom digital con autoenfoque.
- Alimentación a través de Ethernet (POE), además de E/S digitales.
- Funcionalidad día y noche, con el uso de infrarrojos.
- Tamaño y peso reducidos (114x75x46 mm y 250 g)

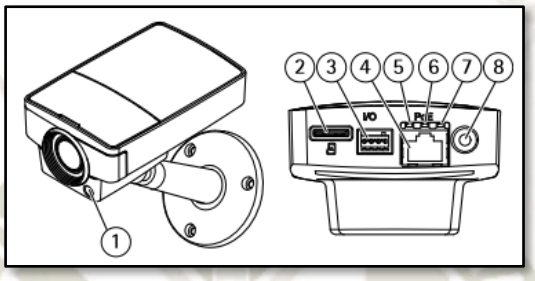


Figura 43. Cámara Axis M1145-L
Recuperado de: https://www.axis.com/files/datasheet/ds_m1145l_1667957_es_1703.pdf

Las principales conexiones de esta cámara se muestran en la siguiente imagen:

1. Sensor de Luz
2. Ranura para Tarjeta microSD
3. Conector de E/S
4. Conector de Red
5. LED de Red
6. LED de Estado
7. LED de Poder
8. Botón de Control

La alimentación a través de Ethernet, o también denominada POE (Power over Ethernet) es una tecnología que permite incorporar alimentación eléctrica a una conexión LAN estándar. Esto permite que un dispositivo de red (switch, router, teléfono, cámara IP) sea suministrado de alimentación eléctrica por medio del mismo cable utilizado para la conexión de red. Para ello se cuentan con diferentes herramientas para establecer dicha conexión.

- Inyector de alimentación: Suministra la energía necesaria a todo el kit. Se obtiene la alimentación necesaria a través del enchufe de pared y el cable Ethernet se puede conectar al ordenador o a un switch para el envío de datos al servidor. Se obtiene un único cable que entrega voltaje y datos hasta el dispositivo de red.

- Unidad Terminal o Splitter: Realiza el proceso inverso al inyector de alimentación, recibiendo de un switch POE una señal de alimentación y datos en un solo cable de conexión, para dividirla en 2 cables que entreguen cada señal a un dispositivo sin conexión POE.
- Switch POE: Cumple la función de un inyector POE, pero dispone de múltiples terminales para una mayor cantidad de dispositivos y con una sola fuente de alimentación.

Se explica en la imagen 3 métodos de conexión por POE, utilizado para diferentes casos:

- Primer caso: Inyector POE y switch para la conexión de una cámara IP con opción POE.
- Segundo caso: Switch POE para la conexión de múltiples cámaras IP con alimentación POE.
- Tercer caso: Inyector POE y Splitter para cámaras sin opción POE.

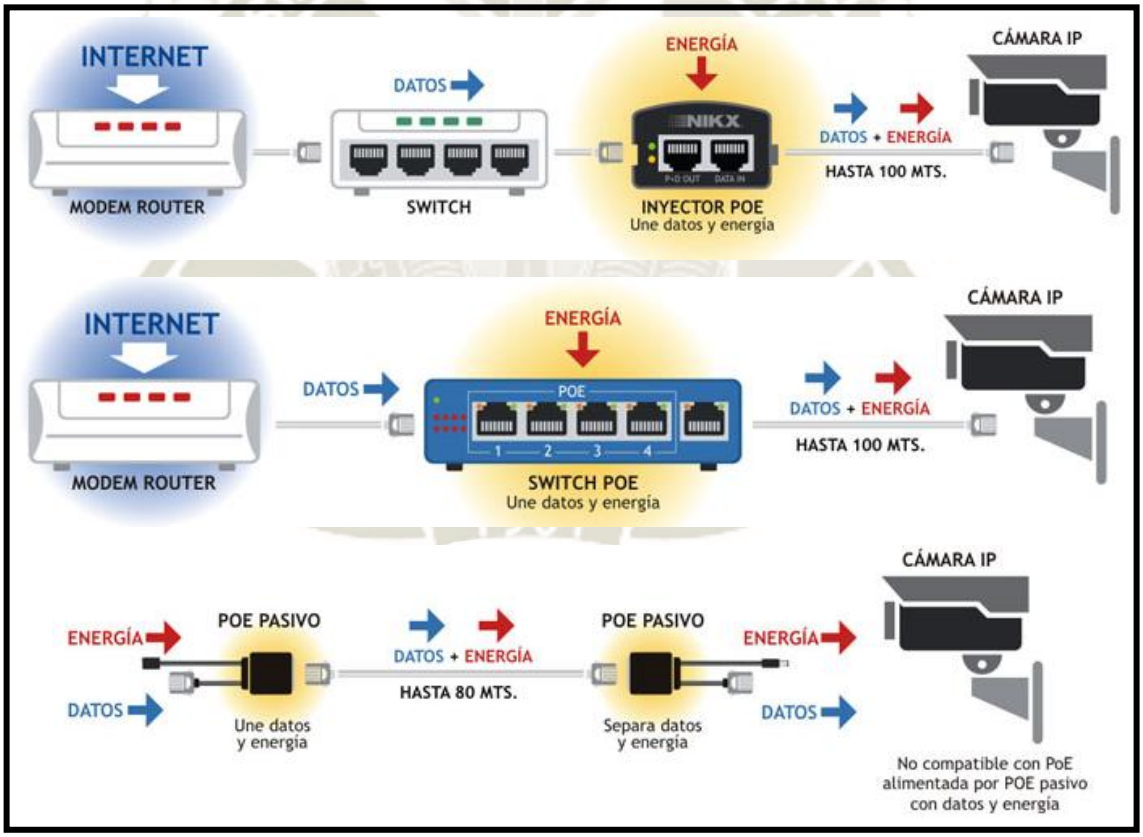


Figura 44. Métodos de conexión POE
Recuperado de: http://empretel.com.mx/content/55-que-es-y-como-funciona-un-poe-

Este tipo de cámara requiere un programa a instalar para ser utilizadas o configuradas desde un ordenador o dispositivo móvil. Axis provee 2 herramientas a esta necesidad.

2.3.1. AXIS IP Utility

"AXIS IP Utility detecta y muestra los dispositivos de Axis de su red. La aplicación también se utiliza para establecer manualmente una dirección IP estática y para acceder a la página inicial de la unidad para realizar otras configuraciones." (AXIS COMMUNICATIONS, s.f.)

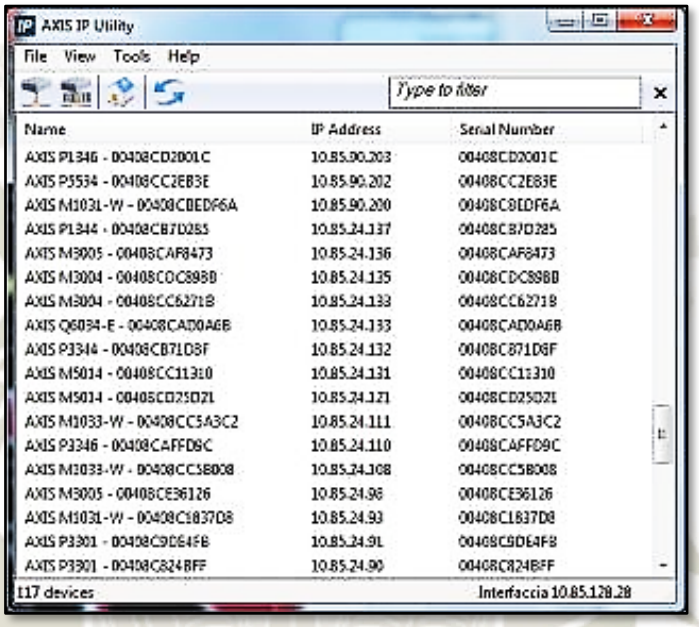


Figura 45. Entorno del programa Axis IP Utility
Recuperado de: https://www.axis.com/files/datasheet/ds_m1145l_1667957_es_1703.pdf

En el entorno inicial de AXIS IP Utility se mostrarán automáticamente todas las cámaras disponibles. A partir de allí se puede realizar distintas acciones:

- Acceder al dispositivo: Doble clic en cualquier cámara abrirá la página web de configuración.
- Asignar una IP: Clic derecho en la cámara de la lista y seleccione "Asignar nueva dirección IP al dispositivo seleccionado".

2.3.2. AXIS Device Manager

"AXIS Device Manager es una herramienta que pone a su disposición una forma sencilla, económica y segura de gestionar dispositivos. Dispone de instaladores de seguridad y administradores del sistema, una herramienta muy eficaz para gestionar todas las tareas principales de instalación, seguridad y mantenimiento." (AXIS COMMUNICATIONS, s.f.)

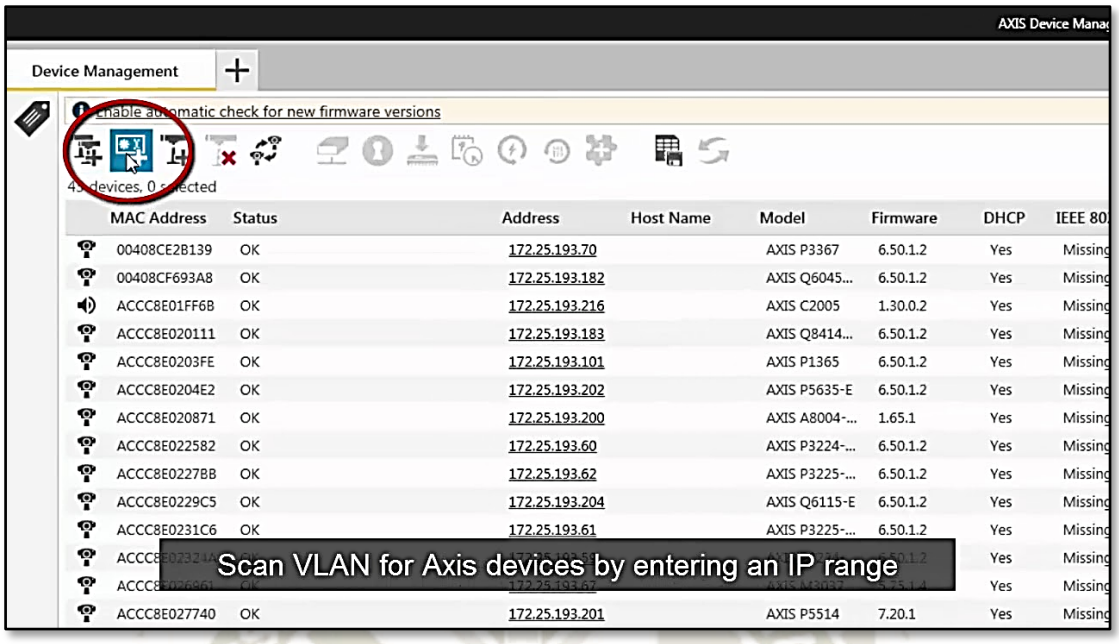


Figura 46. Entorno del programa Axis Device Manager
Recuperado de: https://www.axis.com/files/datasheet/ds_m1145l_1667957_es_1703.pdf

AXIS Device Manager busca dispositivos de AXIS automáticamente en la red y los agrega a la lista. A partir del entorno inicial se pueden realizar distintas acciones según las necesidades:

- Conexión con un servidor: En el Menú principal – Servidores – Nueva conexión se puede elegir las opciones "Servidor remoto" para unirse a un servidor en la red, o en su defecto conectarse directamente al equipo a través de "Esta computadora".
- Agregar dispositivos a la red: A través de la herramienta "Gestión de dispositivos" se puede elegir agregar de forma manual de una lista con "Agregar dispositivos", en un rango de IP con "Agregar dispositivos del rango de IP declarado" o una IP específica con "Agregar dispositivo desde dirección".
- Acceder al dispositivo: Haga doble clic en el nombre en la lista abrirá la página de configuración en el navegador.
- Asignar una IP: Haga clic derecho en la cámara de la lista y seleccione "Asignar dirección IP al dispositivo seleccionado" con el símbolo  . Introducir la dirección IP, la máscara de subred y el router predeterminado.

2.4. AXIS Network Camera

Corresponde a la página de configuración por defecto vinculada a la dirección IP de cada cámara. Estas cámaras trabajan con un usuario principal "root" el cual tiene total control y debe ser reservado para tareas de control y programación. La contraseña de esta cuenta se encuentra en el manual entregado con la cámara, se puede configurar de ser necesario. Además de ello es recomendable crear un usuario con privilegios limitados para evitar accesos no deseados.

Axis Network Camera cuenta con 3 pestañas principales: "Live View" (vista en vivo), "Setup" (configuración) y "Help" (ayuda). Se explicará a continuación las herramientas más utilizadas:

2.4.1. Live View (vista en vivo)

El primero de ellos es la ventana principal de visualización y tiene herramientas generales para controlar la transmisión en directo. Se puede configurar según las necesidades del usuario. Los principales elementos son:

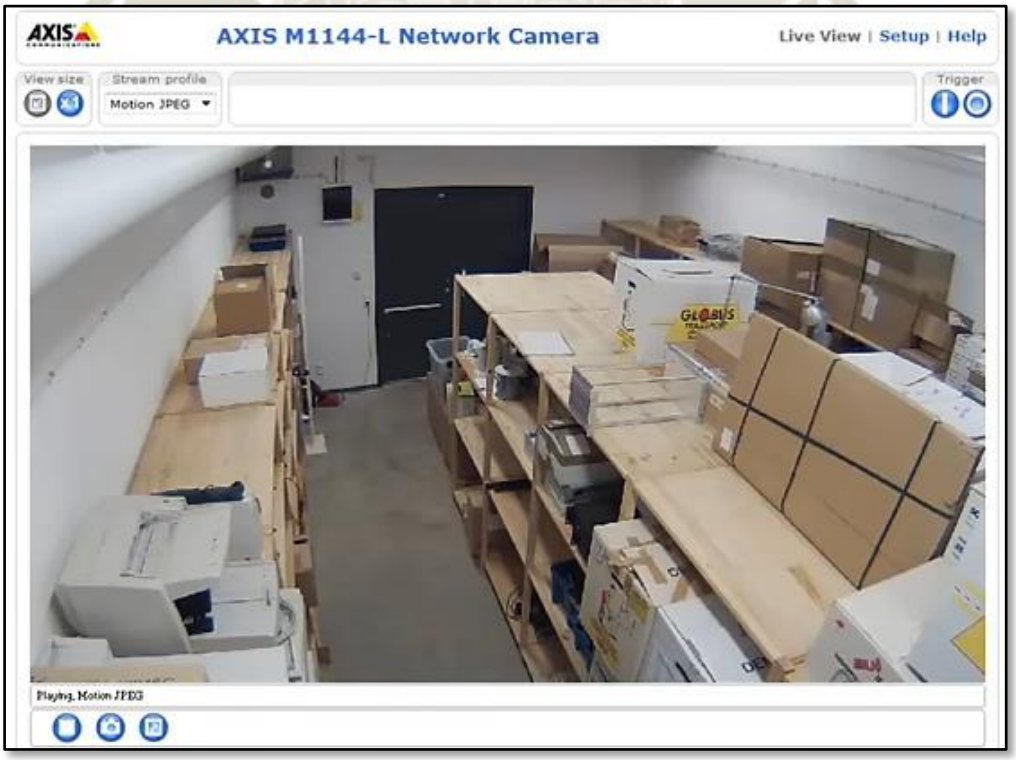


Figura 47. Pestaña de vista en vivo en la página AXIS
Recuperado de: https://www.axis.com/files/datasheet/ds_m1145l_1667957_es_1703.pdf

- View size (Ver tamaño): Ubicada en la esquina superior izquierda, permite mostrar la imagen en su tamaño original o escalarla a la ventana del navegador.
- Stream profile (Perfil de transmisión): Al lado de la herramienta anterior, permite la selección del tipo de video que transmite la cámara. Se puede crear perfiles adicionales además del configurado por defecto.
- Trigger: En la esquina superior derecha, permite activar una acción desde la ventana de transmisión. Esta acción se configura en la pestaña de "Setup".
- Barra de Control: Ubicado en la zona inferior, suele tener herramientas generales de transmisión de video. Entre estas se encuentran: Play (Reproducir), Stop (Parar), Snapshot (Captura de imagen), Record (Grabar), View Full Screen (Vista de Pantalla Completa).

2.4.2. Setup (configuración)

El segundo de ellos es la ventana de configuración y tiene herramientas específicas para controlar distintas características y funciones de la cámara. En nuestro caso nos enfocaremos en las que engloba la sección de "Basic Setup" (Configuración básica), que cuenta con:

- Users: Permite la creación y edición de permisos para los usuarios que acceden a la cámara. Un administrador puede configurar otros usuarios con nombre y contraseñas, además de inicios de sesión anónimo del espectador, con acceso a la página de Live View.
  - Nivel Administrador: tiene acceso libre a todas las funciones. Puede añadir, modificar o remover otros usuarios sin ninguna restricción.
  - Nivel Operador: tiene acceso a todas las funciones, a excepción de configuración de eventos, privacidad de red o subir aplicaciones de archivos.
  - Nivel Observador: tiene acceso a la página de Live View, restringiendo la página "Setup".
- TCP/IP: Las cámaras AXIS soportan dirección IP versión 4 y 6, ambas pueden ser activadas simultáneamente, pero solo se utilizará una para la transmisión.

- Dirección IPv4: Es configurada por defecto de forma automática vía DHCP (Dynamic Host Configuration Protocol), aunque también es configurable vía programas AXIS IP Utility o AXIS Device Manager.
- Dirección IPv6: Si está activa esta opción, se generará una dirección acorde a la configuración del router de red. Cualquier modificación se realiza desde el router.

- Date & Time: Muestra la fecha y hora actual en formato reloj de 24 horas y 12 horas. También permite modificar estos datos según 3 métodos.
  - Sincronizar con el tiempo del ordenador: Con esta opción, se modifica la fecha y hora una vez y no se actualizará automáticamente con el ordenador.
  - Sincronizar con un server NTP: Con esta opción, se actualizará la fecha y hora de manera continua, es necesario crear un server DNS en caso se use una cuenta anfitriona.
  - Configurar manualmente
- Video stream: Configura el video de transmisión a través de 3 pestañas.
  - La pestaña "Imagen" permite modificar resolución, compresión (cambia la calidad a cambio de menor ancho de banda y tamaño de archivo), rotación y velocidad de frame.
  - La segunda pestaña video "H.264" es un estándar de compresión de video que proporciona transmisiones de video de alta calidad a bajas velocidades de bits. Esta trasmisión consiste en tipos de fotogramas I-frames y P-frames. Un I-frame es una imagen completa, mientras que los P-frame solo contienen las diferencias entre I-frame.
  - La tercera pestaña video "MJPEG" provee una calidad buena de imagen, la cual es constante a cambio de mayor ancho de banda, además de mejorar los detalles ante entornos de baja iluminación o mayor complejidad de colores.
- Focus & Zoom: Configura el enfoque y zoom en la cámara. Esto es posible por 2 métodos.
  - Automático: Seleccionar el zoom deseado y luego utilizar el botón "Realizar autoenfoque" (Perform autofocus).

- Manual: Seleccionar el zoom deseado, luego clic en "Abrir iris" para brinda el mejor enfoque en la zona reducida de la imagen. Usar el slider para cambiar el tamaño de la zona y el enfoque necesario. Finalmente habilitar el iris y bloquear el autoenfoque.

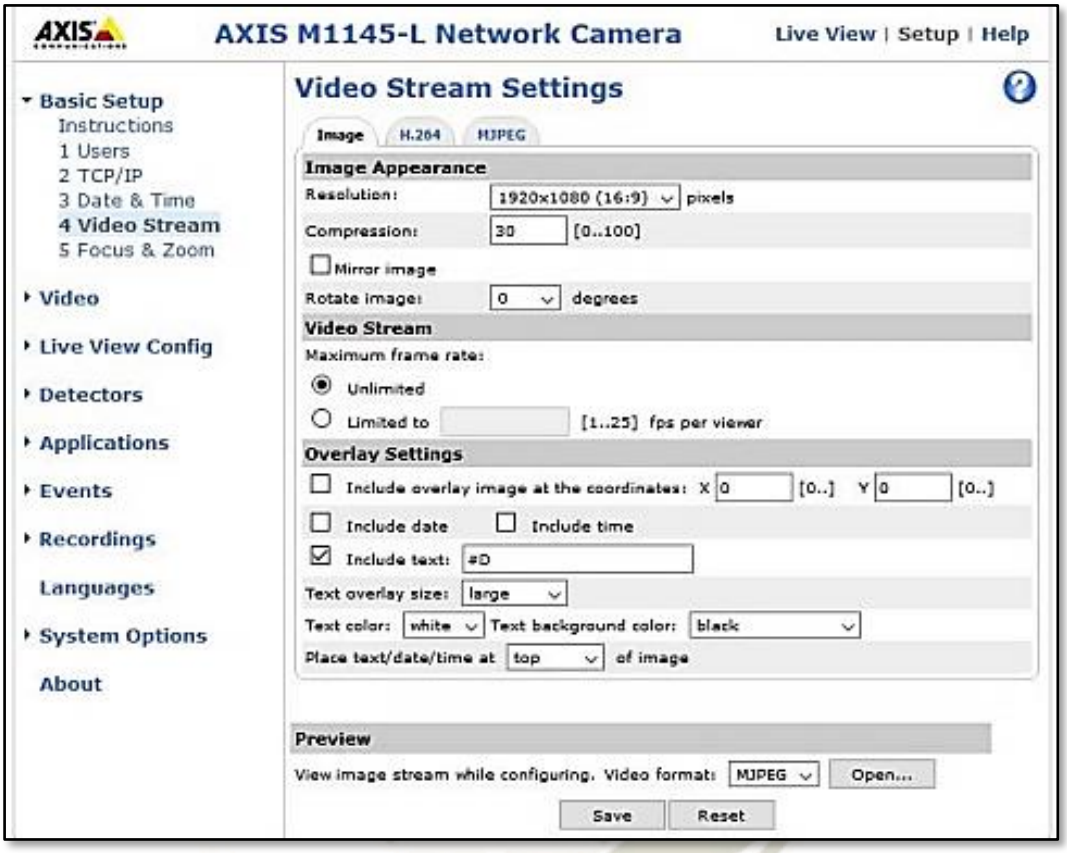


Figura 48. Pestaña de configuración en la página AXIS
Fuente: Elaboración Propia

2.5. Parámetros de proyecto

Como se indicó anteriormente, se utilizó el modelo AXIS M1145-L de cámara IP, por su gran versatilidad y accesibilidad, además de ser portable y contar con un equipo en la Universidad. A continuación, se resumirá los parámetros utilizados para nuestro proyecto:

- Conexión: se utilizó la configuración con inyector POE y ordenador, debido a que al utilizar un solo computador no se necesita de un router para acceder a la transmisión.
- Dirección IP: Se generó una dirección IPv4 manualmente con el programa AXIS IP Utility, La dirección utilizada fue "`http://192.168.1.90`", accesible a través del ordenador.

- Tipo de visualización: Se decidió por la visualización de imagen, por ello el formato JPG mostrado en la dirección "`http://192.168.1.90/jpg/image.jpg`", el cual es leído por el software MATLAB para el procesamiento.

3. Localización de placa vehicular

La localización de la placa vehicular a partir de una imagen es la segunda etapa en el proyecto. El objetivo es extraer la zona de la imagen que contenga los caracteres necesarios para la identificación del vehículo y con una calidad decente para su procesamiento posterior. En este aspecto se debe considerar que la placa no siempre se encontrará en la misma posición, pero si con una orientación y tamaño similares, debido a que la cámara trabajará de forma estacionaria y el punto de ingreso de vehículos no varía en ningún momento.

Existen múltiples formas de realizar esta etapa, pero la manera utilizada en este proyecto es el uso de un procesamiento de formas (morfológicos) y el filtrado espacial de los pixeles que conforman una imagen. Esto requirió de 3 bloques de programación secuenciales:

3.1. Pre-Procesamiento

Debido a la naturaleza de este método, se requiere reducir la cantidad de datos por píxel a la esencial. Los pasos adicionales son relativos y dependientes de la precisión que se requiera.

- Escala de grises: la primera necesidad es la de reducir la paleta de colores a escala de grises, necesario para las operaciones posteriores. Comando utilizado: **"rgb2gray"**.
- Reducción de la resolución de la imagen: en nuestro caso se utilizó 400x300 debido a que permite trabajar con una calidad decente y el tamaño de imágenes utilizadas es menor a 1 MB, por lo que la información a procesar es mínima. Comando utilizado: **"imresize"**.
- Recorte de una sección de la imagen: se realiza con la intención de reducir los espacios en los cuales no se puede ubicar la placa vehicular debido a su disposición estándar en casi todos los vehículos. Comando utilizado: **"imcrop"**.

- Filtro Morfológico: utilizamos el filtro morfológico "bottom hat" debido a que permite resaltar zonas muy claras, mejorando su visibilidad y debilitando la percepción del resto. Comandos utilizados: **"strel"** para la creación de un elemento estructurante de patrón para el filtro y **"imbothat"** como filtro morfológico de estilo bottom hat.
- Binarización: por último, se realiza una nueva reducción de la paleta de colores a la binaria, que permite resaltar aún más los detalles claros y eliminando por completo las zonas intermedias de escala de gris. Comando utilizado: **"im2bw"**.

El siguiente ejemplo muestra este método en una imagen tomada de una cámara celular:

Figura 49. Imagen original de un vehículo
Fuente: Elaboración Propia

Figura 50. Imagen recortada y binarizada durante el pre-procesamiento
Fuente: Elaboración Propia

Los parámetros utilizados en los comandos mostrados anteriormente se pueden observar en el capítulo de ANEXOS. Cada uno de estos parámetros se obtuvo mediante el proceso de prueba-error, debido a que no existe un estándar y las condiciones de ambiente e imagen varían. Es posible

reducir la información de los exteriores de la placa sin perder los datos referentes a esta. Aun así, este proceso por sí solo no permite localizar la placa vehicular, como se ve en la imagen.

3.2. Procesamiento Morfológico

El procesado morfológico es el segundo paso en la localización, además de ser la más importante. La cantidad de parámetros y operadores aumenta, debido a que se debe eliminar la mayor cantidad de información irrelevante de la imagen pre-procesada. Como se explicó anteriormente, este procesamiento depende de 3 factores: conjunto que representa la imagen, los elementos estructurantes y los operadores.

En nuestro caso el procedimiento realizado es la unión de las diversas herramientas siguiendo un orden que permite resaltar los elementos de la placa y eliminar los demás conjuntos, aun así, existen diversas formas de ordenar los operadores para realizar el mismo objetivo.

- Cierre: este operador permite conectar elementos cercanos entre sí, rellenando agujeros y alisando los contornos. Lo que se logra es unir los elementos que representan los caracteres de la placa y generar bloques claros en un fondo negro.

  Comandos utilizados: **“strel”** para la creación de un patrón para el operador (“disco” de preferencia) y **“imclose”** como operador morfológico de cierre.
- Apertura: este operador cumple la labor de filtro, eliminando los píxeles aislados en la imagen, pudiendo elegir el tamaño según el patrón utilizado. En nuestro caso se debe realizar pruebas para definir el tamaño óptimo que permita englobar la placa y eliminar el ruido.

  Comandos utilizados: **“strel”** para la creación de un patrón para el operador (“disco” de preferencia) y **“imopen”** como operador morfológico de apertura.
- Dilatación: mediante este operador se realiza el último paso en el resaltado de los bloques significativos en una imagen, dilatando los bloques anteriormente reducidos por la apertura y mostrando una menor cantidad de conjuntos para la selección posterior.

Comandos utilizados: **“strel”** para la creación de un patrón para el operador (“línea” de preferencia) y **“imdilate”** como operador morfológico de dilatación.

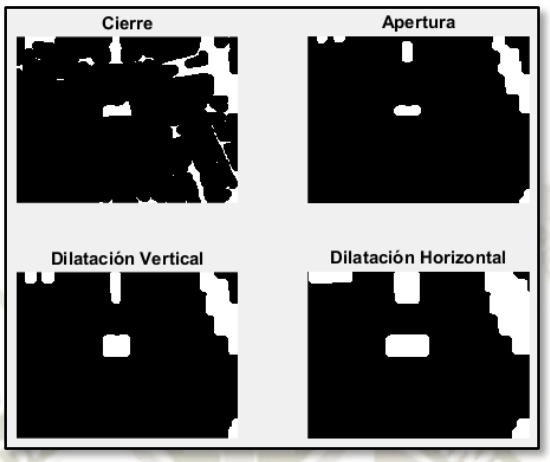


Figura 51. Imagen obtenida después de realizar el procesamiento morfológico
Fuente: Elaboración Propia

Como se observa en la imagen, se han generado 6 bloques de datos como resultado del procesamiento morfológico anterior. Esto varía de imagen a imagen y dependerá de los parámetros utilizados para la creación de los elementos estructurantes de patrón. Sin embargo, este resultado permite detectar a simple vista la ubicación de la placa vehicular en la imagen.

3.3. Interpretación y Clasificación

El último paso es el análisis del resultado, el cual no es el ideal, pero nos da una posición relativa del objeto deseado. Por ello se necesita una etapa de clasificación en la cual consideramos las dimensiones y posiciones esperadas o “coherentes” de cualquier placa vehicular.

En primer lugar, se debe separar los distintos conjuntos en variables, indicando su posición y dimensiones. Esto se logra con el comando **“bwlabel”** el cual crea regiones etiquetando los distintos conjuntos de píxeles conectados. Luego de ellos se extraen los datos de cada uno de ellos con el comando **“regionprops”**, cuyas opciones son:

- ‘Area’: Número real de píxeles en la región, mostrado como un escalar.
- ‘BoundingBox’: Rectángulo más pequeño que contiene a una región, regresada con un vector 1x2Q, donde Q es el número de dimensiones de la imagen.

- 'Centroid': Centro de masa de la región, regresada como un vector 1xQ, donde las coordenadas se ordenan según x-y-z.
- 'Orientation': Ángulo entre el eje "x" y el eje mayor de la elipse que contiene a los elementos de la imagen. El valor es en grados entre -90 y 90 grados.
- 'Perimeter': Distancia alrededor de la frontera de la región, como un escalar. Cuenta los píxeles adyacentes a los que corresponden a la frontera de la región.

Para nuestro procedimiento, la propiedad a utilizar es la de **BoundingBox**, la cual extrae las dimensiones del cajetín que encierra cada conjunto, indicando la ubicación de la esquina superior izquierda y su ancho y alto, siguiente el orden ["X" "Y" "ancho X" "ancho Y"]. El resultado es mostrado a continuación:

```
cuadro =

    1.5     0.5    40    11
   71.5    57.5    39    20
   78.5     0.5    24    29
  160.5     0.5    41    75
  185.5   133.5    16    18
```

Figura 52. Datos de los bloques generados en el procesado morfológico
Fuente: Elaboración Propia

Se puede extraer la ubicación de la placa utilizando el comando "imcrop" en la imagen inicial. Para ello se utilizan las dimensiones que corresponden al segundo elemento reconocido.

4. Reconocimiento de Caracteres

En el tercer bloque, el objetivo es reconocer los caracteres a partir de una imagen que representa la placa vehicular. El procedimiento a utilizar es similar al anterior, debido a que se deben extraer cada caracter por separado para su reconocimiento. Se necesitará de 3 etapas para lograr esto.

4.1. Binarización, filtrado y recorte

De manera similar al utilizado en la localización de la placa, se necesita trabajar con una imagen con la menor cantidad de ruido y con regiones bien diferenciadas. Aun así, según la precisión necesitada se utilizará otros comandos.

Inicialmente se recurre al comando **“imbothat”** nuevamente para reducir el ruido y el comando “**im2bw**” para convertir el resultado a un modo de color de blanco y negro. Esto permite obtener regiones bien diferenciadas unas de otras, pero conlleva a un problema a solucionar.

Al ser una imagen obtenida de otra previa, la cantidad de píxeles se ve reducida, provocando que se generen regiones pequeñas que no corresponden a ningún caracter, esto mediante el efecto de “pixelado”. Sin embargo, con la ayuda del comando **“bwareaopen”**, se puede establecer un tamaño mínimo de área como patrón para filtrar dichas regiones. En consecuencia, el código utilizado se generaría de la siguiente manera.

```
EE = strel('disk',”tamaño”);        %Elemento Estructurante
Imagen = imbothat(“Imagen”,EE);     %Filtrado de los píxeles
Imagen = im2bw(“Imagen”);           %Binarizado de la Imagen
Imagen = bwareaopen(“Imagen”,”tamaño”);%Filtrado de regiones
```

Finalmente se necesita eliminar los bordes de la imagen para resaltar únicamente los caracteres. Se pueden utilizar diferentes métodos, pero al ser la imagen una matriz de píxeles, se puede eliminar las columnas y filas que contengan un número indicado de píxeles blancos. El resultado de todo este procedimiento es mostrado a continuación:

Figura 53. Procesamiento de la Placa Vehicular
Fuente: Elaboración Propia

Según la precisión utilizada en cada comando, se puede distinguir también los caracteres que representan al texto de nacionalidad en la parte superior y el guion que separa los 2 grupos de caracteres. Para nuestra aplicación no utilizaremos estos valores, por lo que los eliminamos.

4.2. Segmentación de Caracteres

Como segundo paso se requiere la separación de cada uno de los caracteres como elementos independientes para su reconocimiento. El procedimiento es el mismo que el utilizado en la ubicación de la placa, recurriendo a los comandos **"bwlabel"** y **"regionprops"** con la opción de "BoundingBox". Como se muestra a continuación se extrae la totalidad de caracteres y las dimensiones de los recuadros que engloban cada uno de ellos.

```
n3 =

     6


stats3 =

         13.5          3.5           38           73
         58.5          0.5           33           77
         99.5          0.5           34           76
        170.5          0.5           32           75
        210.5          0.5           31           71
        252.5          0.5           35           71
```

Figura 54. Extracción de cantidad y dimensiones de los caracteres de la placa
Fuente: Elaboración Propia

Finalmente es necesario realizar la independización de estos elementos, con el comando **"imcrop"** y utilizando los valores que representan a cada uno de los caracteres, mostrados como filas en la matriz "stats3". Considerar luego de ello la estandarización de los tamaños de cada conjunto, con el comando **"imresize"** para establecer plantillas con dimensiones definidas. En la siguiente imagen, se muestran las plantillas extraídas a partir de la imagen de la placa.

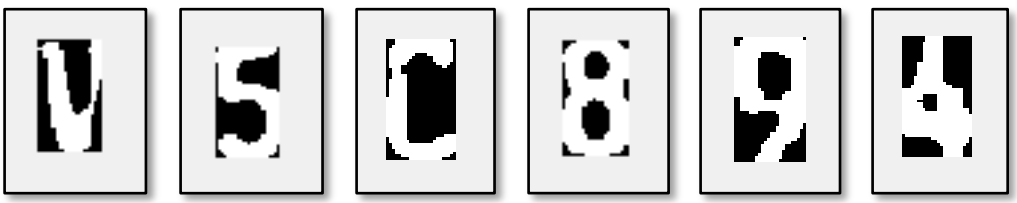


Figura 55. Ejemplo de imágenes de caracteres extraídos
Fuente: Elaboración Propia

4.3. Comparación de Caracteres

El tercer paso es la identificación de cada carácter mediante la comparación de sus píxeles con una plantilla de letras y números estandarizados y predefinidos en una matriz de datos. Con este objetivo, se necesita resaltar 2 bloques diferentes que se realizan en diferentes secciones del programa, pero que requieren de los mismos parámetros.

4.3.1. Creación de la Matriz de Datos

Al inicio del programa es necesario definir las plantillas de letras y números, las cuales se pueden extraer de la binarización de imágenes de la Internet o en su defecto, creadas como una matriz de datos. Para el primer caso se necesitará utilizar el comando **"imread"** para leer la imagen como una matriz, mientras que el segundo necesita desarrollar cada matriz por separado.

En ambos casos es necesario considerar que las dimensiones de las matrices deben ser igual para permitir una comparación de todos los píxeles. Cada una de estas matrices se debe almacenar bajo variables, para finalmente ser integradas en una matriz general con ayuda del comando **"mat2cell"** como se muestra en el código a continuación:

```
mat1 = [a b ... y z uno dos ... nueve cero]
mat2 = mat2cell(mat1, "alto" ,[repmat("ancho",1,"n° elementos")])
```

Donde se muestra en la matriz "mat1" la matriz general formada por cada uno de los caracteres sin ninguna separación entre sus datos, para luego en la matriz "mat2" se segmenta cada carácter en submatrices que representan los datos de cada elemento.

4.3.2. Comparación de Patrones por correlación

Como última etapa, se realiza la comparación de los patrones extraídos de la placa vehicular y la matriz de caracteres desarrollada al inicio del programa. El comando de comparación con mayor uso en aplicaciones es **"corr2"** el cual calcula el coeficiente de correlación entre 2 matrices de dimensiones iguales. La fórmula utilizada para ello es:

$$r = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \overline{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \overline{B})^2\right)}}$$

Donde $\bar{A}$ y $\bar{B}$ son el promedio de valores de píxeles que conforman ambas matrices. A partir de comparar cada carácter con cada patrón en la matriz general, se generará 36 datos de correlación.

En la siguiente imagen, se observa la matriz "letras" que muestra la posición relativa de cada carácter dentro de la matriz general, representada como variables entero o integer (int), mientras que en la segunda se interpreta cada posición como una variable de tipo carácter o character (char):

```
letras =

    22    31     3    34    35    30
```

```
car =

V5C894
```

Figura 56. Matriz de posiciones e identificación de caracteres
Fuente: Elaboración Propia

Sin embargo, el método de correlación describe cada carácter según su parecido a una plantilla, mas no identifica al carácter de la matriz, puesto que estos valores deben ser interpretados por el usuario o por medio de programación. Otra desventaja es la necesidad de realizar un gran número de comparaciones por cada caracter, aumentando el tiempo de procesado y la carga computacional.

4.3.3. Comparación de Patrones por RNA

Las redes neuronales son otra opción para el reconocimiento de patrones y la interpretación de los mismos. El software MATLAB ofrece una herramienta llamada "**nnstart**" que abre una interfaz gráfica (GUI) para la generación y entrenamiento de redes. Esta interfaz presenta 4 comandos para resolver distintos problemas:

- "nftool": Ajuste de datos, con una red feedforward de dos capas entrenada con el concepto de Levenberg-Marquardt.
- "nprtool": Reconocimiento de patrones, con una red feedforward de dos capas con neuronas de salida sigmoidea.

- "ntstool": Series de tiempo dinámicas, con una red feedforward de dos capas.
- "nctool": Agrupamiento, mediante el uso de un mapa auto-organizado. Este mapa forma una representación comprimida del espacio de entradas.

La opción utilizada es **"nprtool",** debido a que genera resultados menos ambiguos con la función de salida sigmoidea. Observamos el tipo de red neuronal que se genera con este comando:

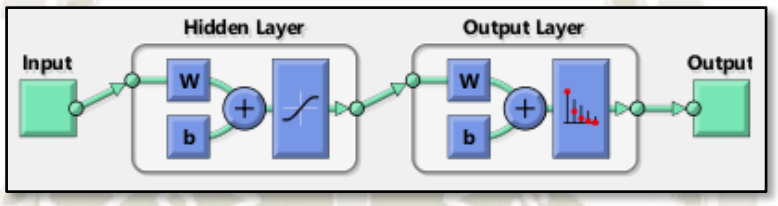


Figura 57. Esquema de arquitectura de RNA por MATLAB
Fuente: Elaboración Propia

Al continuar, observaremos la ventana de ingreso de entradas y salidas, la cual permite ingresar variables del espacio de trabajo ("workspace"), así como matrices guardadas en archivos tipo ".mat". Además, permite la selección de disposición de las muestras en filas o columnas. Se debe considerar que el número de muestras de entrada debe ser igual al número de muestra de salida.

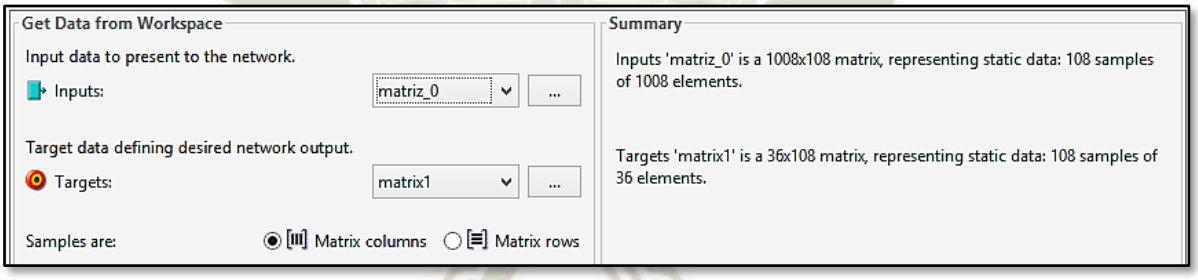


Figura 58. Ventana de ingreso de entradas y resultados deseados en MATLAB
Fuente: Elaboración Propia

Luego de ello se deberá elegir el porcentaje de muestras que servirán como entrenamiento, validación y testeo. De esta manera se calcula que tomando como base 36 caracteres (números y letras), se necesitan como mínimo 108 muestras para realizar la generación y comprobación del mismo. Esto se logra utilizando la matriz generada en la sección anterior, pero convirtiendo las matrices en vectores de datos, además de utilizar más plantillas para cada carácter.

Como pasos finales se elige el número de capas, siendo recomendado utilizar el mismo número de valores de salida definidos (36 para nuestra plantilla) y luego realizar el entrenamiento. Como resultado de este proceso se obtendrá una ventana similar a la mostrada:

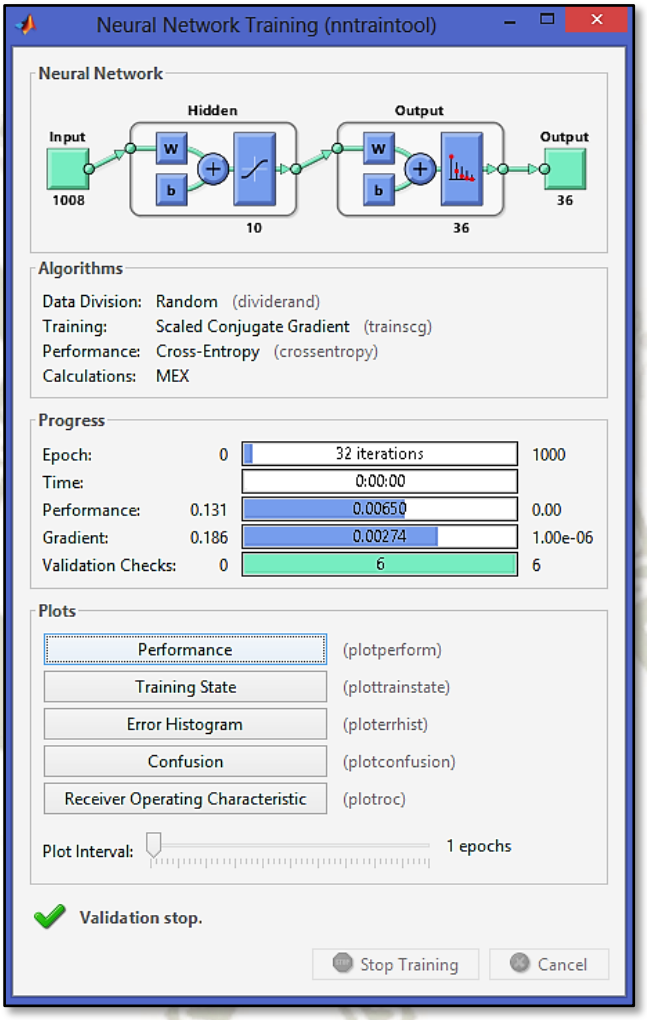


Figura 59. Ventana de evaluación de entrenamiento de RNA en MATLAB
Fuente: Elaboración Propia

Se observan 4 secciones importantes que caracterizan a la red neuronal generada:

- Red Neuronal: Muestra la estructura de la red neuronal, con la cantidad de entradas y salidas, capas ocultas y de salida, además de las funciones utilizadas para cada neurona.
- Algoritmos: Entrega los métodos de entrenamiento utilizado como: división de datos (entrenamiento, validación y testeo), tipo de entrenamiento (gradiente de conjugado y escalado) y desempeño (entropía cruzada – elimina desviaciones entre datos).

- Progreso: Incluye resultados del entrenamiento como: iteraciones de entrenamiento (epoch), tiempo necesario para el entrenamiento, desempeño, gradiente y número de validaciones.
- Gráficos: Muestras aspectos de la red que permitan valorar su funcionamiento. Incluye:
  - Desempeño: actuación de la entropía cruzada o desviación de datos, entre una distribución de datos dada "q" y la verdadera distribución "p". Es deseado reducir lo mayor posible este valor. Se mide en variables aleatorias continuas:

$$-\int_X p(x)\,\log q(x)\,dx$$

  - Estado de entrenamiento: indica la desviación del valor de la gradiente para minimizar la función error, la cual se logra modificando los pesos a partir de la ecuación mostrada, donde "p" es el factor de aprendizaje que modifica en mayor o menor medida los pesos. Es deseado reducir este valor hasta lograr una desviación más horizontal respecto a la cantidad de iteraciones.

$$E_T(w+\Delta w) \approx E_T(w) + \Delta w * \nabla E_T(w)$$

$$\Delta w = -\rho \Delta E_T(w) \qquad \rho > 0$$

  - Histograma de error: muestra el error como la diferencia entre los targets (resultados deseados) y las salidas, dada una cierta cantidad de iteraciones. Se busca que las desviaciones estén más cercanas a cero y sean lo más cerradas, demostrando una mayor consistencia y desviaciones corregibles en una mayor cantidad de datos.

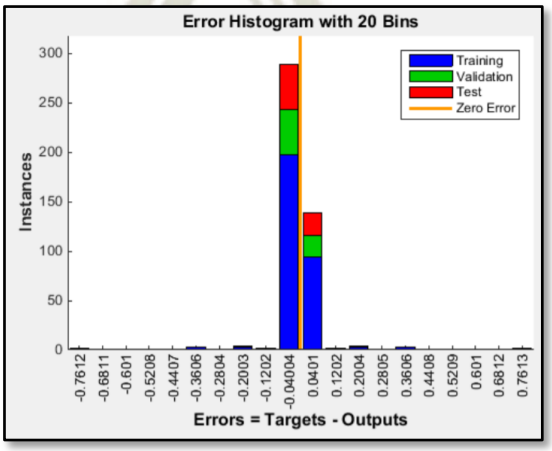


Figura 60. Histograma de error de una RNA entrenada
Fuente: Elaboración Propia

- Matriz de confusión y características del funcionamiento del receptor (ROC).

Luego de continuar en la interfaz principal, se nos permite exportar la red neuronal generada en 3 formas de despliegue:

- Aplicación: Función que trabaja tanto con matrices como arreglos de celdas para el compilador y herramientas de creación de MATLAB.
- Código: Función que solo trabaja con matrices, sin soporte para arreglos de celdas y es usado en herramientas de codificación de MATLAB.
- Simulink: Diagrama de bloques de representa el comportamiento de la red neuronal y también es desplegable en el codificador de esta herramienta.

5. Comparación e interpretación de placas

El cuarto y último bloque del programa es el agrupamiento de los caracteres obtenidos en el bloque anterior y asociar dicho conjunto a un usuario real, extrayendo información perteneciente a este. Según lo mencionando es necesario crear una base de datos inicial de trabajo que permita no solo visualizar el registro de usuarios disponible, sino además ampliar dicho registro o editar alguno de ellos en caso la información ya no sea válida.

5.1. Creación de la base de datos

Para la creación de una base de datos, Microsoft Office presenta 2 alternativas válidas de trabajo: Microsoft Excel y Microsoft Access. Por motivos de accesibilidad se utilizará el Microsoft Excel el cual presenta herramientas más que necesarias para este objetivo.

El primer paso es definir la información a tratar como parte de cada usuario, la cual debe ser almacenada en un cuadro simple como se muestra en la siguiente imagen:

	A	B	C	D
1	Matrícula	Propietario	Marca	Modelo
2	V7P400	Miguel Ojeda	Hyundai	Grand I10
3	V2S571	Janeth Esquivel	Dongfeng	E6380LF
4	V5C894	Rosa Galdos	Nissan	Navara
5	V2C402	César Condori	Ford	Escape
6	V7K117	Joselin Collantes	Suzuki	Swift
7	V8L118	José Paz	Toyota	Fortuner
8	A1C432	Adolfo Salazar	Toyota	Corolla XLI
9	X2P341	David Meléndez	Toyota	RAV 4

Figura 61. Base de datos simple desarrollada en Microsoft Excel
Fuente: Elaboración Propia

Este documento debe ser guardado en una ubicación conocida y accesible para el programa de MATLAB. Considerar nombrar también la hoja en la cual se ha desarrollado la tabla, además de evitar dejar filas en blanco entre usuarios para que el MATLAB reconozca la tabla entera.

5.2. Importación de tabla a MATLAB

El siguiente paso es la importación de los datos de nuestra tabla al entorno de MATLAB, lo cual mediante el comando **"xlsread"**. La sintaxis se puede observar en la página de MathWorks:

```
[num,txt,raw] = xlsread(filename,sheet,xlRange)
```

Donde:

- Num: Extrae los datos numéricos de la tabla.
- Txt: Extrae los datos tipo texto de la tabla
- Raw: Extrae todos los datos de la tabla
- Filename: Nombre del archivo que contiene a la tabla
- Sheet: Nombre de la hoja que contiene a la tabla
- xlRange: Nombre generado al rango que contiene a la tabla

Como ejemplo se muestra el código utilizado para extraer los datos de nuestra tabla:

```
[~, base, ~] = xlsread('Base de datos.xls','REGISTRO');
```

5.3. Comparación de datos de placa vehicular

Por último, falta comparar el conjunto de caracteres que conforman la placa vehicular, para ello se necesita convertir tanto la matriz de caracteres obtenidos por OCR como la de nuestra base de datos a tipo carácter, con el comando **"char"**. A partir de ello se puede comparar los resultados y extraer la información que corresponde a la fila del usuario identificado.

```
car = char(car);                          %Conversión a tipo caracter
base = char(base(2,1));                   %Conversión a tipo caracter
if size(car) == size(base);               %Igualdad de n° caracteres
for i=1:1:h
n = base(i,1));                           %Extraer cada placa
if n==car                                 %Comparación de conjuntos
inf1 = base(i,1)); inf2 = base(i,2)); %Extraer la información de
inf3 = base(i,3)); inf4 = base(i,4)); la fila seleccionada
end; end
```

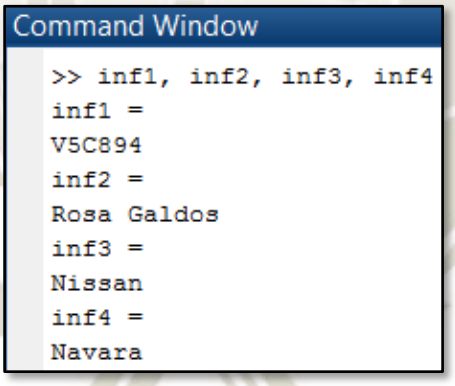


Figura 62. Extracción de información del personal en MATLAB
Fuente: Elaboración Propia

Como se observa la información extraída tiene el formato "carácter" y dependerá de la cantidad de información agregada a la base de datos, por lo cual cualquier cambio se debe realizar en el documento de Excel, el cual se puede realizar de forma manual o con ayuda de formularios.

5.4. Creación de formulario para base de datos

Los formularios de Excel son herramientas para ingresar datos a la hojas de manera sencilla y accesible para cualquier usuario. Añade la posiblidad de colocar instrucciones precisas y recuadros necesarios, evitando de esta manera errores en la captura de información.

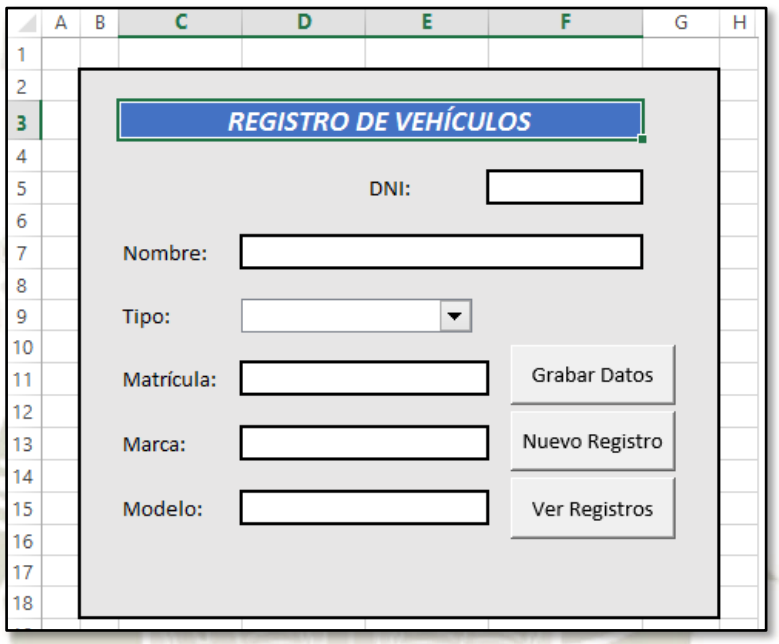


Figura 63. Hoja de formulario desarrollada en Microsoft Excel
Fuente: Elaboración Propia

Para nuestro caso se utilizó una hoja de cálculo con controles de formulario. Los datos considerados para ingresar son: Nombre, Tipo de Usuario, Matrícula, Marca y Modelo. El dato mostrado en la parte superior es la identificación del usuario, la cual puede variar (código, DNI, etc.). Por último, se colocaron 3 botones para realizar las funciones de grabar, generar un nuevo registro (limpiar la hoja) y observar la lista de registros de los usuarios.

Además de la hoja del Formulario, se agregó 2 hojas más que interactúan con los botones mencionados anteriormente:

- AUXILIAR: Incluye una lista con las opciones de "tipo de usuario" y la tabla que recoge los datos del usuario ingresados en el formulario y los copia a la hoja de REGISTRO.
- REGISTRO: Tabla que contiene todos los usuarios registrados en el formulario.

	A	B	C	D	E
1					
2		TIPO		DATOS GRABAR	CELDAS
3				CÓDIGO	P - 0
4		Autoridad		Matrícula	0
5		Administrativo		Propietario	0
6		Docente		Tipo	Docente
7		Estudiante		Marca	0
8		Trabajador		Modelo	0
9					

	A	B	C	D	E	F	G
1							
2		BASE DE DATOS					
3		DNI	Matrícula	Propietario	Tipo	Marca	Modelo
4		P - 100	V7P400	Miguel Ojeda	Docente	Hyundai	Grand I10
5		P - 101	V2S571	Janeth Esquivel	Administrativo	Dongfeng	E6380LF
6		P - 102	V5C894	Rosa Galdos	Autoridad	Nissan	Navara
7		P - 103	V2C402	César Condori	Estudiante	Ford	Escape
8		P - 104	V7K117	Joselin Collantes	Docente	Suzuki	Swift
9		P - 105	V8L118	José Paz	Administrativo	Toyota	Fortuner
10		P - 106	A1C432	Adolfo Salazar	Trabajador	Toyota	Corolla XLI
11		P - 107	X2P341	David Meléndez	Estudiante	Toyota	RAV 4
12		P - 108	J4U93J	Christian Pinto Vizcarra	Estudiante	Hyundai	Tucson
13		P - 109	C7K123	María Rivera Chavez	Docente	Mitsubishi	ASX 2WD
14		P - 110	V7O357	Carla Aguilar Salas	Docente	Suzuki	S-Cross
15		P - 111	D0J208	Wilbert Zeballos Gonzal	Autoridad	Volkswagen	Tiguan Track & Field

Figura 64. Hojas "auxiliar" y "registro" desarrollada en Microsoft Excel
Fuente: Elaboración Propia

Los botones son controles ActiveX los cuales contienen funciones creadas en un módulo del Visual Basic para Aplicaciones (VBA). Algunas de estas funciones se explican a continuación:

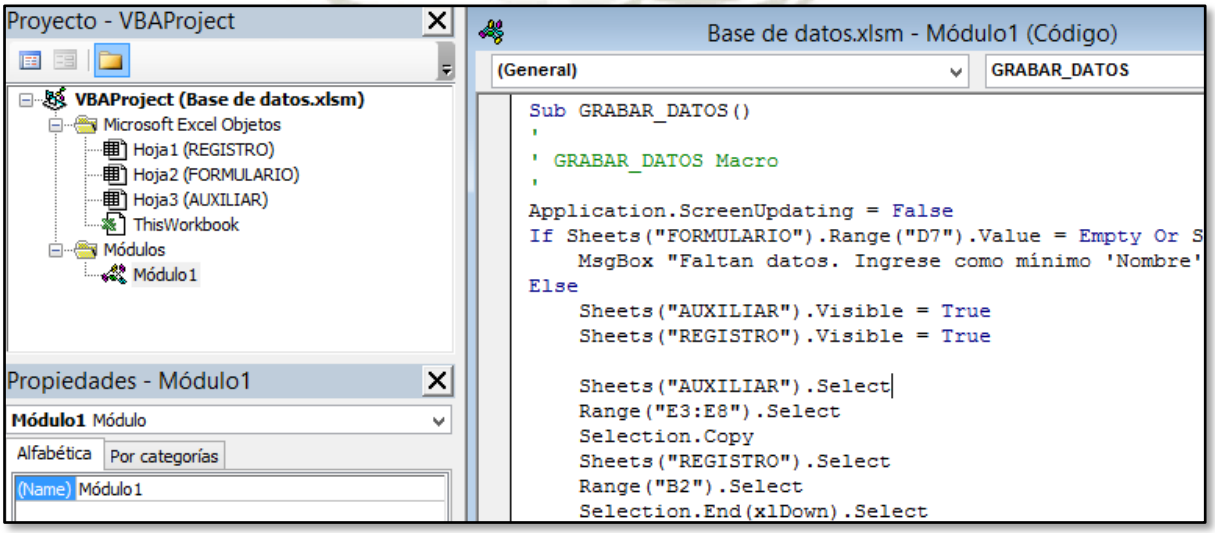


Figura 65. Creación de un módulo para funciones desarrollada en Visual Basic
Fuente: Elaboración Propia

- Application.ScreenUpdating: Cambiar la propiedad de actualización automática
- MsgBox: Generar una caja de mensaje
- Selection.PasteSpecial: Realizar un copiado especial de un rango de celdas
- ClearContents: Limpiar un rango de celdas

CAPITULO IV: RESULTADOS

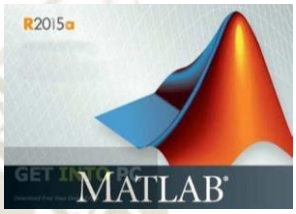
En este capítulo se mostrarán los programas desarrollados durante el proyecto, además de indicar los software y componentes físicos que se necesitaron para lograr su funcionamiento.

1. Lista de hardware y software

Tabla 13
Lista de componentes seleccionados

Cantidad	Descripción	Imagen Referencial
01	Cámara IP Axis modelo M1145-L	
01	Adaptador e inyector POE Ubiquiti modelo POE-24-12W 24V	
01	Trípode con rótula de 3 ejes Tripod, peso: 1.75 kg, altura: 1.61 m	
02	Cables de red categoría 5E con conectores RJ45 (5 y 10 m)	
01	Router Wifi TP-Link 450 Mbps modelo Tl-WR940N	
01	Laptop Toshiba Modelo Satellite S855-S5382	

Tabla 14
Lista de software utilizados

Versión	Descripción	Imagen Referencial
1.14.17.678	***IP Webcam*** Convertir un dispositivo Android en una cámara IP para acceder desde cualquier ordenador.	
R2015a	***MATLAB*** Herramienta de software matemático que ofrece un entorno de desarrollo con lenguaje propio.	
4.0.4.0	***AXIS IP Utility*** Detecta, muestra y configura manualmente un dispositivo Axis con una IP estática en la red.	
4.35.004	***AXIS Camera Management Client*** Herramienta de instalación y gestión de dispositivos de video, además de aplicaciones y seguridad	

En la Tabla 10, se seleccionaron los componentes en base a su relación costo/calidad, pero con más importancia en la accesibilidad, hecho que se muestra con la cámara AXIS que fue prestada por la Universidad para su uso en este proyecto.

En la Tabla 11, se consiguió software que sean de instalación sencilla y fáciles de utilizar para realizar las funciones que necesitamos, además de contar con compatibilidad para un mayor número de dispositivos tanto de celular, portátil y cámara IP.

2. Primer programa: Android IP Webcam

El objetivo inicial era lograr la transmisión de imágenes por medio de una dirección Web sin necesidad de una conexión física entre cámara y ordenador. La mejor manera de ejemplificarlo es por medio de un celular y una aplicación: IP Webcam.

Para realizar la transmisión de una forma más cómoda y accesible, se requirió de la herramienta GUI para diseñar la interfaz de este programa. Esta consta de un input de texto, 3 botones de acción y 2 recuadros de imagen para realizar las siguientes funciones:

- Input de Texto: Ingresar la dirección IP del servidor creado por la aplicación, en formato de imagen, lo cual reduce sustancialmente la cantidad de información a transmitir. En caso de necesitar el video como opción, simplemente se elimina el "\photo.jpg" de la dirección.
- Botón 1: Empieza la transmisión de imágenes desde el servidor al primer recuadro de imagen, extrayendo la dirección del input y luego crea un bucle que muestra continuamente la información. Por lo tanto, no es posible cambiar la dirección durante la transmisión.
- Botón 2: Corta la transmisión del servidor al GUI, mediante el uso de una variable que detiene el bucle de la transmisión y además limpia el recuadro de la transmisión. En este momento se puede cambiar la dirección IP de ser necesario.
- Botón 3: Realiza una captura de imagen de la transmisión, mostrándola en el recuadro de la derecha. Se debe considerar que esta imagen dependerá de que la transmisión se encuentre activa, por lo que en caso que la transmisión no haya iniciado o se haya apagado, este botón no realizará ninguna acción perceptible, además de lanzar una advertencia.

El código utilizado para el desarrollo de este programa se muestra en el apartado de Anexos, donde se muestran las variables y funciones utilizadas para cumplir con las acciones mostradas anteriormente. Este programa puede ser compilado en una aplicación para ser independiente del software MATLAB y utilizarse en otros ordenadores. A continuación, su diagrama de flujo:

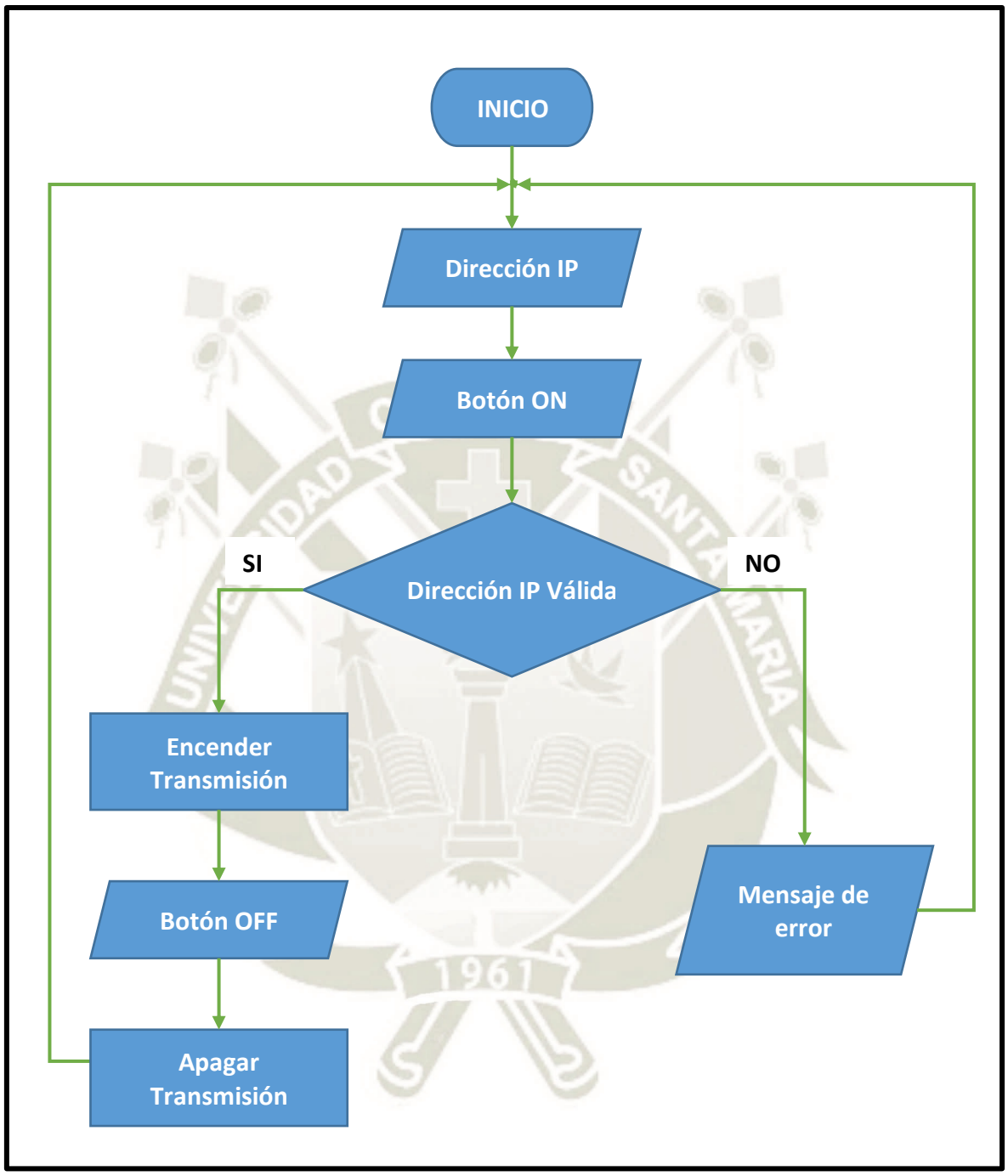


Gráfico 66. Diagrama de Flujo del 1er Programa
Fuente: Elaboración Propia

Como un adicional del programa, se puede utilizar con otras aplicaciones, diferentes a IP Webcam, que realizan la acción de transmisión pública (“streaming”), puesto que solo depende de la dirección IP como limitante para acceder a dicho servidor.

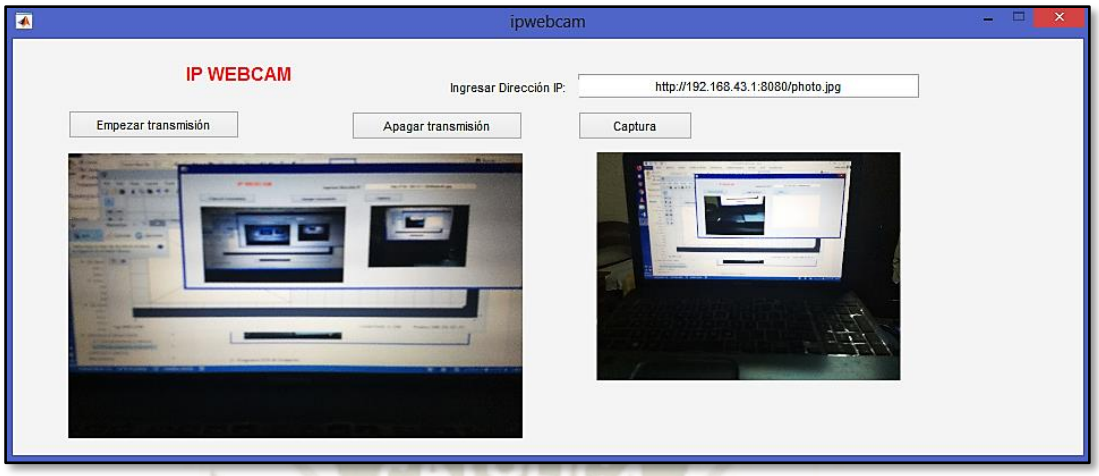


Figura 67. Prueba de funcionamiento del 1er Programa
Fuente: Elaboración Propia

Por último, se procedió a probar los límites de rango para esta aplicación. El primer caso fue la transmisión sin el uso de routers, logrando una distancia de 20 metros de forma local, ocurriendo caídas de fps (de 30 hasta 5 frames por segundo) y pérdidas de transmisión. Con el uso de routers se elimina la caída de fps, además de ampliar el rango hasta distancias de 100 metros.

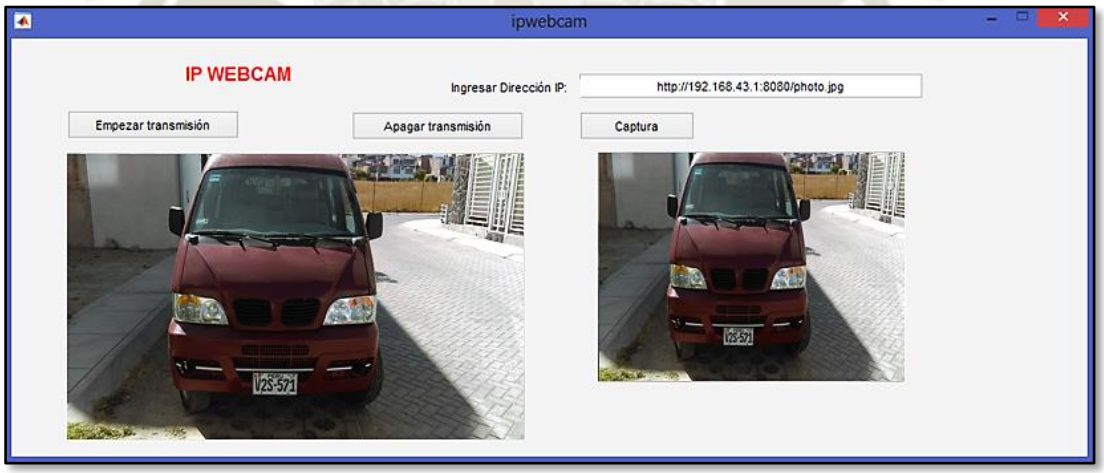


Figura 68. Captura de un vehículo estacionado con ayuda de un router inalámbrico
Fuente: Elaboración Propia

Sin embargo, también es de considerar que existen aplicaciones que con el uso de datos pueden realizar una retransmisión por medio de la nube para lograr un rango superior. Este es el caso de "Ivideon" que sirve como un servidor que conecta desde dispositivos móviles hasta ordenadores como mayor cantidad de opciones de personalización y seguridad.

3. Segundo programa: OCR de Imágenes

En este programa, el objetivo es cumplir con el desarrollo del código que representa los 2 bloques principales del OCR (Reconocimiento óptico de caracteres):

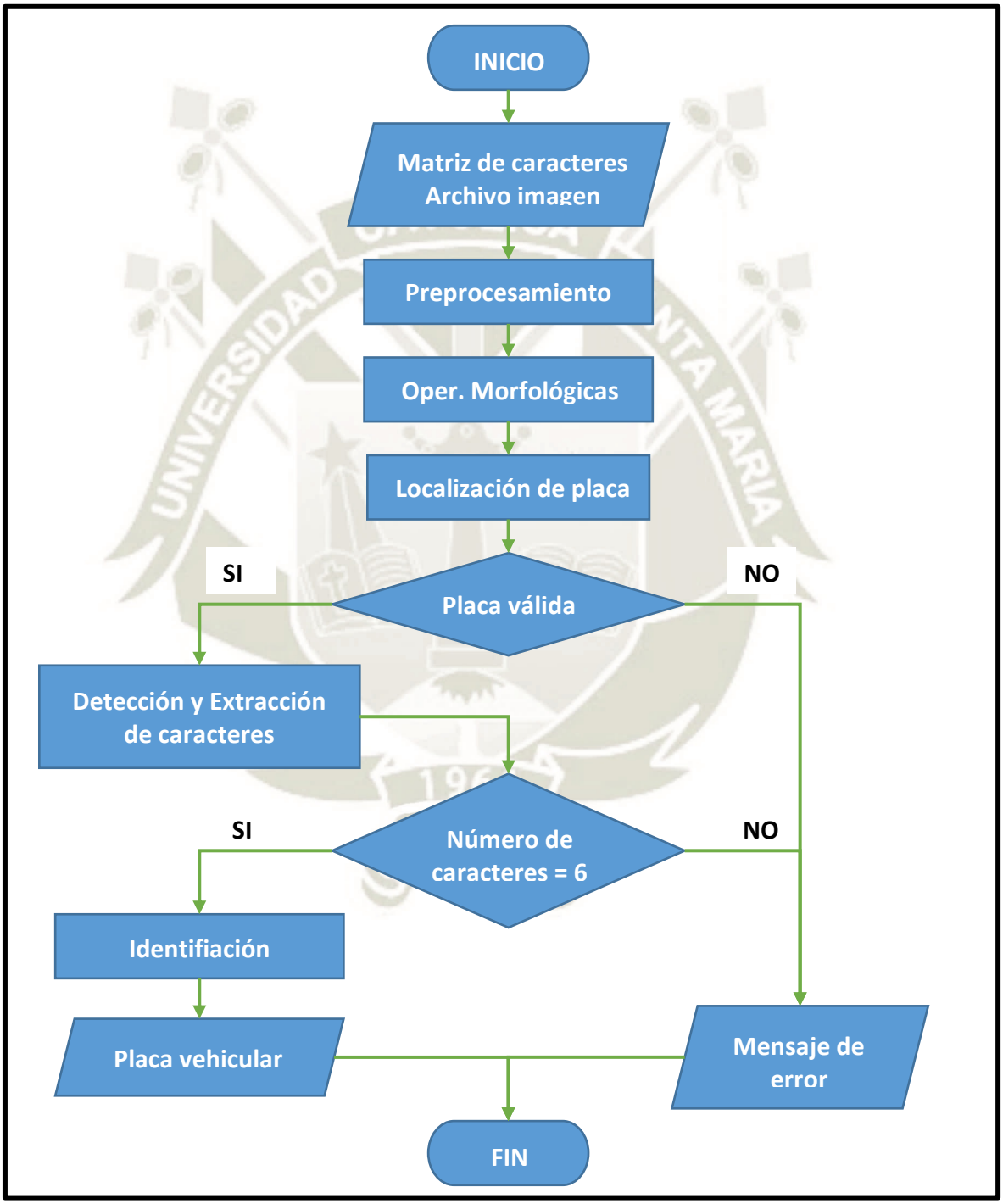


Gráfico 69. Diagrama de Flujo del 1er Programa
Fuente: Elaboración Propia

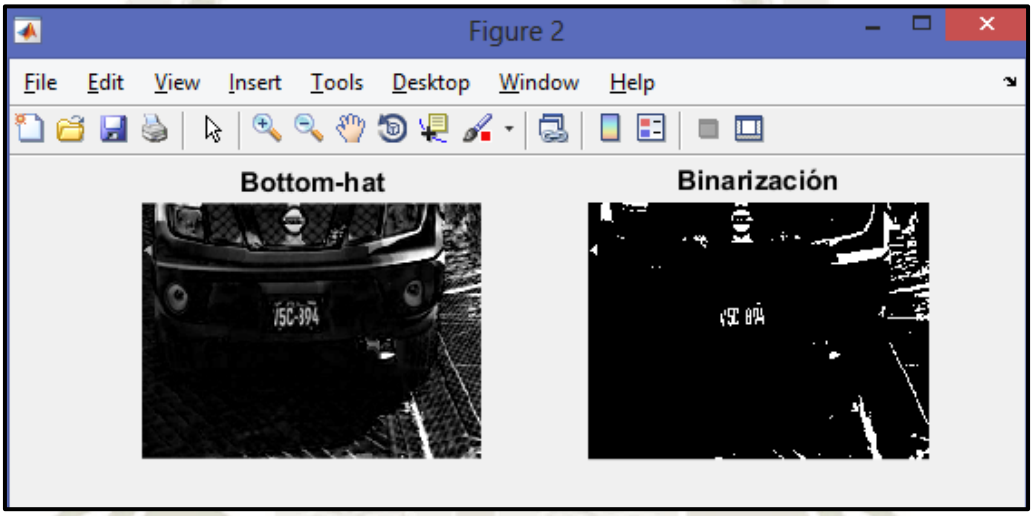
- Localización de placa vehicular (subcapítulo 3.3): ingresa una imagen capturada mediante una cámara y entregará la imagen recortada con la placa vehicular.
- Reconocimiento de caracteres (subcapítulo 3.4): recibe la placa vehicular en formato escala de grises y entregará cada carácter agrupado en una matriz de tipo texto.

El código utilizado para el desarrollo de este programa se muestra en el apartado de Anexos, donde se muestra al inicio la matriz general de caracteres utilizada como patrones para la comparación, estas son formadas por submatrices de 42x24 píxeles por cada caracter. Sin embargo, este programa no cuenta con una interfaz GUI, por lo que es dependiente del software MATLAB y requiere tener las imágenes de prueba junto al código para ser utilizado en otros ordenadores.

3.1. Bloque 1: Localización de placa vehicular

Este apartado se dividió en tres ventanas para mostrar el procedimiento utilizado y descrito en el capítulo anterior. Cada ventana muestra una operación que recoge la imagen anterior como entrada y entrega una imagen resultante mostrada en la parte inferior. A continuación, se explicará el enfoque utilizado en cada una de las ventanas:

- Recorte de información: se reduce la cantidad de datos presentes en la imagen original, cambiando el modo de color RGB a escala de grises y recortando parte de la imagen que no ofrece ninguna información útil para nuestro proyecto.
- Resaltar los detalles importantes: mediante el uso del filtro “bottom-hat” resaltamos aún más los detalles más oscuros en fondos claros, mientras que con la binarización eliminamos todos los elementos que no presenten ningún tipo de resalte en la imagen.
- Extracción del conjunto de la placa: utilizamos los operadores morfológicos siguiendo el orden cierre → apertura → dilatación vertical → dilatación horizontal, con el fin de obtener los recuadros que puedan obtener la placa vehicular, luego se utiliza una condición de posición y tamaño para reconocer el recuadro correcto.



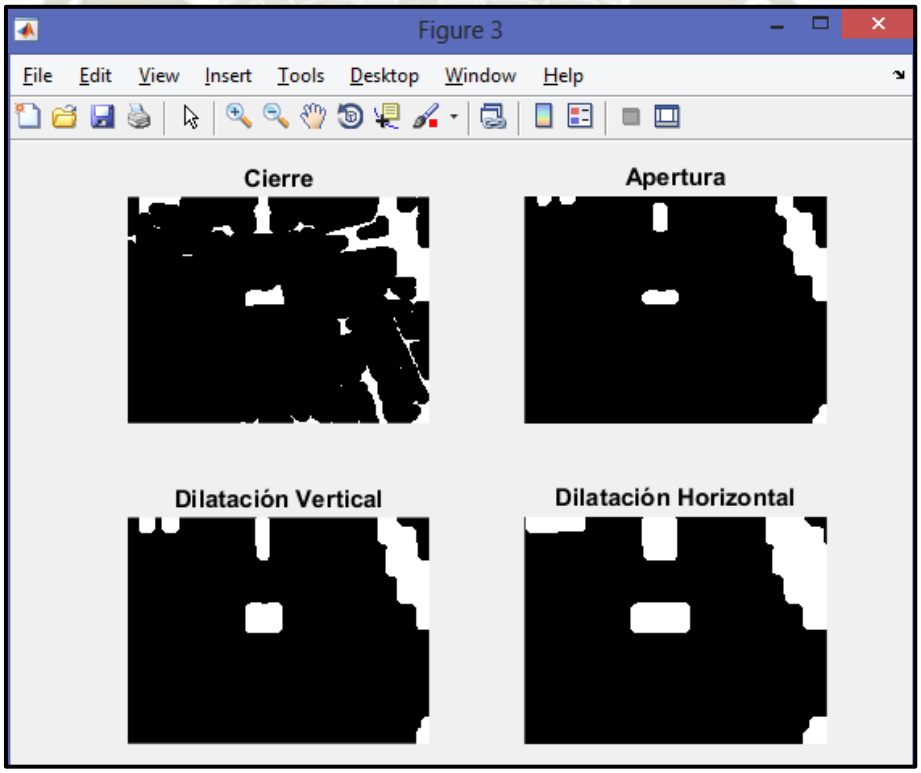


Figura 70. Bloque 1 de OCR: Localización de placa vehicular
Fuente: Elaboración Propia

3.2. Bloque 2: Reconocimiento de caracteres

Este apartado se dividió en dos ventanas y 2 variables para mostrar el procedimiento utilizado y descrito en el capítulo anterior. Cada ventana muestra una operación que recoge la imagen anterior como entrada y entrega una imagen resultante mostrada en la parte inferior, mientras que las últimas 2 variables recogen los números que simbolizan cada carácter y los interpretan, en una palabra. Se procede a explicar el enfoque utilizado en cada una de las ventanas:

- Pre-procesamiento de la placa: a partir de la placa obtenida a partir de las dimensiones de recorte en el bloque anterior, se necesita eliminar los espacios que dificultan la segmentación de la imagen. Esto se logra a partir del binarización (filtrado) del color de la imagen y la eliminación del fondo de color blanco que engloba a la placa.
- Extracción de los caracteres: con los comandos “bwlabel” y “regionprops”, separamos los caracteres, mostrando cada carácter en una matriz estándar de 42x24 píxeles.

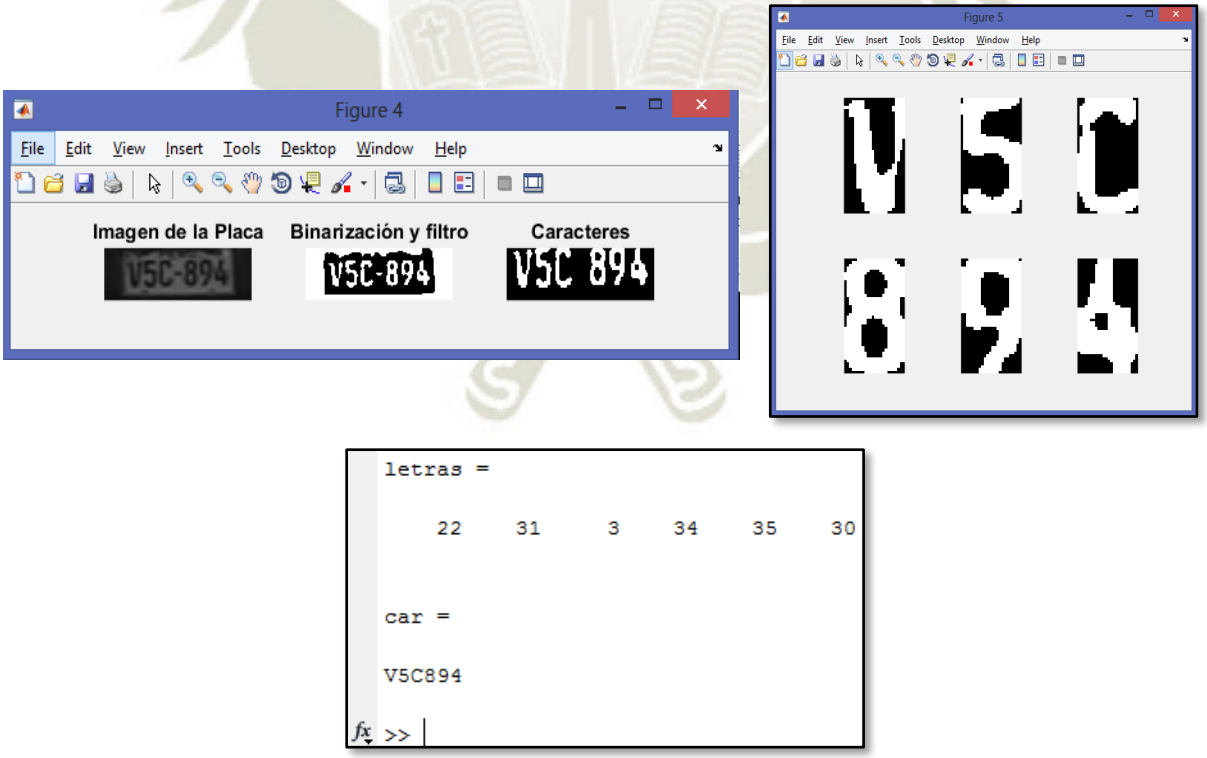


Figura 71. Bloque 2 de OCR: Reconocimiento de caracteres
Fuente: Elaboración Propia

3.3. Restricciones

Entre las restricciones físicas de este programa tenemos:

- La cámara se encuentra en posición frontal y superior a matrícula del vehículo.
- La cámara se encuentra en un ángulo respecto a la horizontal de la matrícula que no supera los 25°, para evitar cambios en la forma de los caracteres
- No se realizarán pruebas en vehículos menores: motos, moto taxis, etc.
- No se consideran toma de imágenes nocturnas.

3.4. Pruebas de funcionamiento

A partir del ejemplo mostrado anteriormente, se realizaron pruebas de este programa con un mayor número de imágenes, cumpliendo requisitos mínimos de información: resolución 480p, modo de color RGB, placa vehicular completa y sin rotaciones exageradas (mayor a 10°).

La característica variable entre las pruebas fue la resolución, la cual es directamente proporcional al tiempo de procesamiento necesario por el programa. En el siguiente gráfico se muestra la proporción:

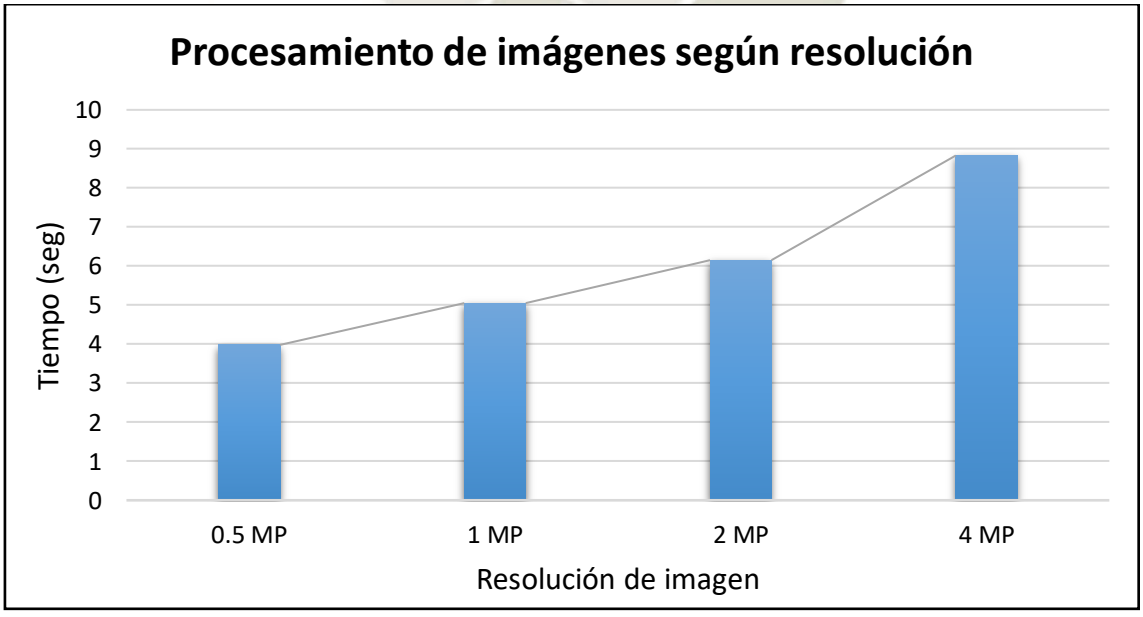


Gráfico 72. Variación de tiempo de procesamiento según la resolución
Fuente: *Elaboración Propia*

Las pruebas se realizaron siguiendo las restricciones mencionadas y considerando en la mayoría de casos la resolución 2MP. A continuación, algunos resultados con sus tiempos de procesamiento:

Tabla 15
Resultados y tiempos de respuesta

Imagen	Resultado	Tiempo de procesamiento
	letras = 22 31 3 34 35 30 car = V5C894	5.9171 seg
	letras = 22 33 24 27 27 33 car = V7X117	6.3461 seg
	letras = 22 28 19 31 33 27 car = V2S571	6.3396 seg
	letras = 22 34 12 27 27 34 car = V8L118	5.8231 seg.

Fuente: Elaboración Propia

Para reducir estos tiempos se considerarán algunos aspectos para los subsiguientes programas:

- Reducir la cantidad de ventanas e imágenes a mostrar.
- Crear y cargar la variable de matriz del alfabeto de forma independiente al código.

4. Tercer programa: GUI de Reconocimiento e Identificación

En el desarrollo del tercer programa, se tuvo como objetivo realizar el reconocimiento en una interfaz que permitiera realizar el procedimiento con un solo botón y con la capacidad de ser reproducible para varias imágenes. Esta mostraría no solo la imagen de la placa extraída, sino que además obtendríamos las matrices que representaban cada carácter y la placa resultante de este reconocimiento.

Otro de los apartados agregados es la adición de 3 botones que sirven para la identificación del usuario asociado a la placa vehicular, cada uno cumple una función específica:

- BUSCAR: Se encarga de buscar la palabra obtenida del reconocimiento (placa vehicular) en un documento Excel de nombre "Base_de_datos.xlsm" y mostrar a continuación la información de "Propietario", "Marca" y "Modelo".
- REGISTRAR: Guarda la placa vehicular y la información asociada, además de la hora que corresponde dicho registro, en una variable de tipo matriz denominada "T".
- EXPORTAR EXCEL: Genera un documento Excel a partir de la variable "T" con los datos descritos anteriormente. El nombre por defecto es "Libro Nuevo".

El código utilizado para el desarrollo de este programa se muestra en el apartado de Anexos, donde se muestran las variables y funciones utilizadas para cumplir con las acciones mostradas anteriormente. Este programa puede ser compilado en una aplicación para ser independiente del software MATLAB y utilizarse en otros ordenadores. A continuación, se muestra el diagrama de flujo que sigue la GUI.

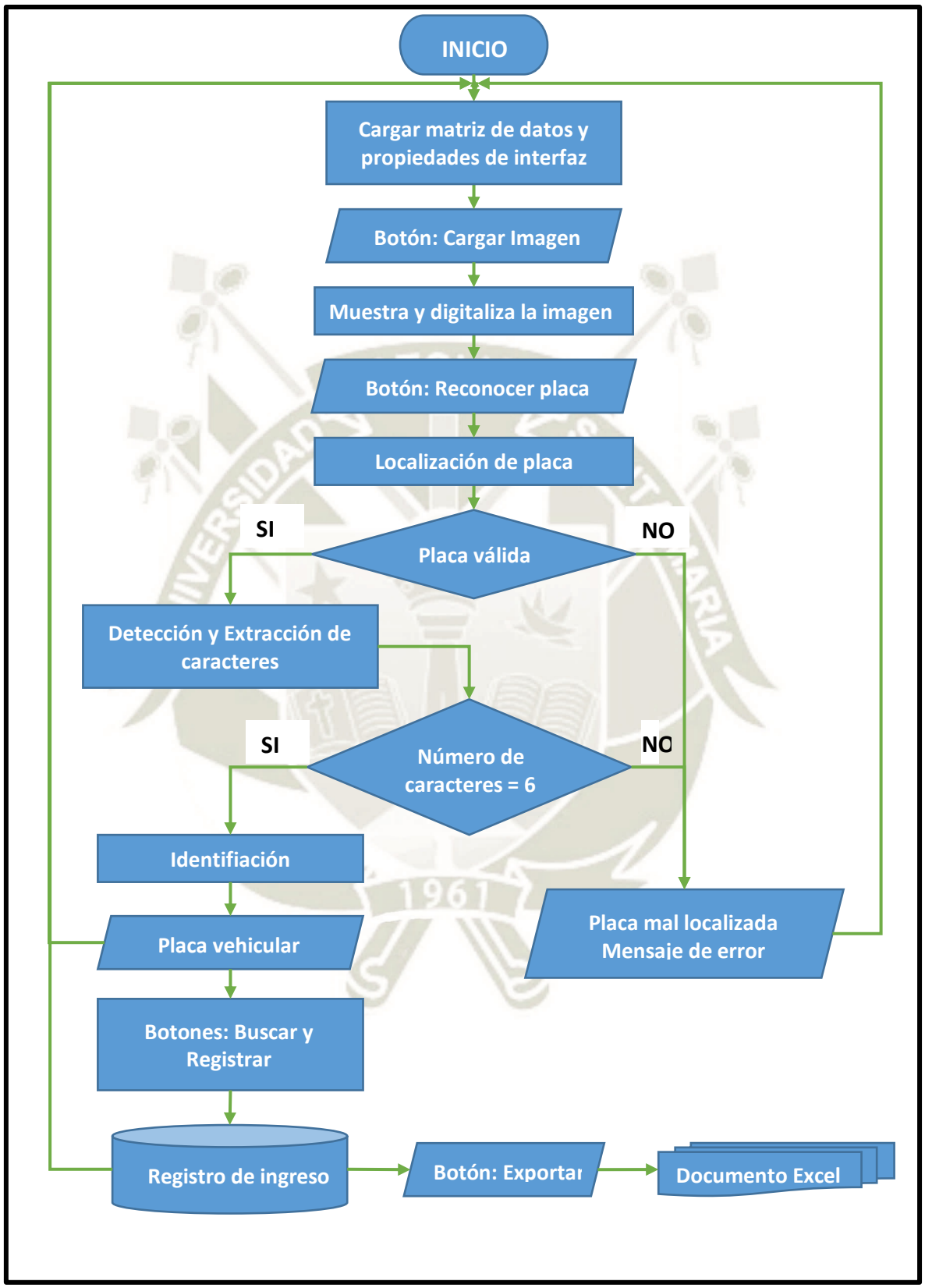


Gráfico 73. Diagrama de Flujo del 3er Programa
Fuente: Elaboración Propia

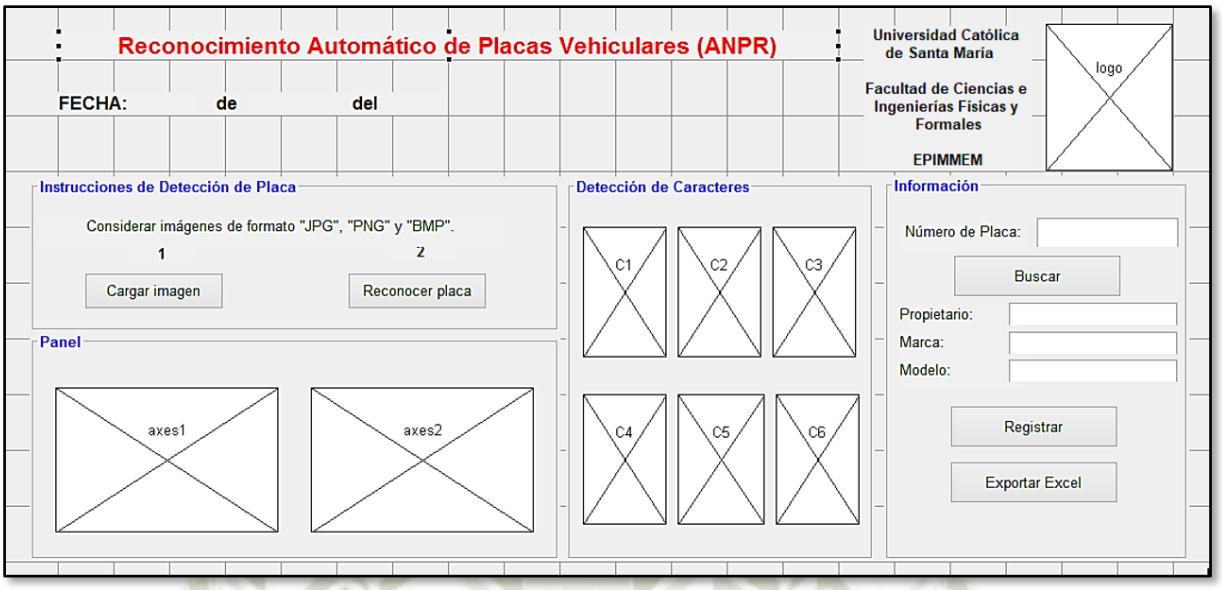


Figura 74. Interfaz Gráfica de MATLAB para el 3er programa
Fuente: Elaboración Propia

4.1. Restricciones

Entre las restricciones físicas de nuestro sistema tenemos:

- La cámara se encuentra en posición frontal y superior a matrícula del vehículo.
- La cámara se encuentra en un ángulo respecto a la horizontal de la matrícula que no supera los 25°, para evitar cambios en la forma de los caracteres
- No se realizarán pruebas en vehículos menores: motos, moto taxis, etc.
- No se consideran toma de imágenes nocturnas sin la iluminación adecuada.
- Es necesario tener en todo momento una PC encendida conectada a la cámara.

En el código del GUI se desarrollaron algunas restricciones, saltando un mensaje de error que permita un uso correcto y no comprometa el funcionamiento continuo del programa:

- Las matrices con los patrones para el alfabeto deben estar en una carpeta accesible para MATLAB, y ser añadidos durante la compilación a una aplicación.
- Formato de Imagen: acepta como válidos las extensiones JPG, PNG y BMP con cualquier resolución de tamaño mayor a 360p para un mejor resultado.

- El reconocimiento de la placa no se realizará si no se ha cargado ninguna imagen.
- En caso de no lograr ubicar la placa vehicular el programa detendrá la segmentación y comparación de caracteres, a fin de evitar un error de tipo bucle sin fin.
- El posicionamiento de la placa vehicular en la imagen debe ser aproximadamente en bloque que representan los 2/3 de la zona inferior y del centro de la imagen, a modo de enfocar el procesamiento en una zona específica y reduciendo el tiempo.

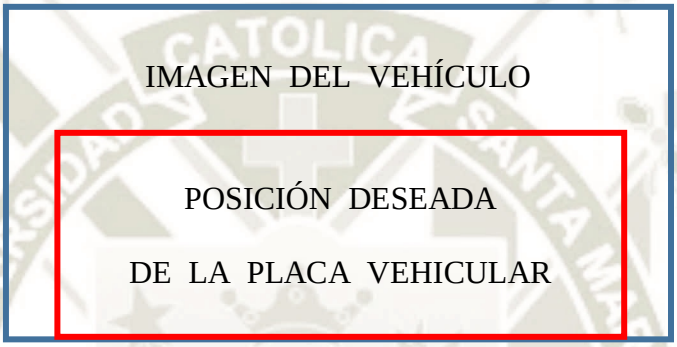


Gráfico 75. Posicionamiento deseado de la placa vehicular
Fuente: Elaboración Propia

- El programa pondrá como condición la búsqueda de la placa antes de realizar algún registro.
- Exportar no eliminará el registro extraído, por lo que para empezar un nuevo registro es necesario cerrar el programa.

4.2. Pruebas de funcionamiento

Figura 76. Pruebas del tercer programa con imágenes de vehículos
Fuente: Elaboración Propia

Como se observa en las imágenes anteriores, se logró con éxito el reconocimiento de placas vehiculares. Los tiempos estimados fueron menores de los mostrados en el programa anterior, pero agregando la búsqueda y registro de los mismos, resultan en tiempos similares. La tabla con el registro de las pruebas realizadas en un día se pudo observar tanto en el software de MATLAB como en el documento Excel extraído, estos cuentan con datos de "Propietario", "Placa", "Hora" y "Minutos". El resultado se muestra a continuación:

```
T =

    'Propietario'       'Placa'     'Hora'     'Minuto'
    '_____'             '_____'     '_____'    '_____'
    'José Paz'          'V8L118'    [   20]    [   25]
    'David Meléndez'    'X2P341'    [   20]    [   26]
    'César Condori'     'V2C402'    [   20]    [   31]
```

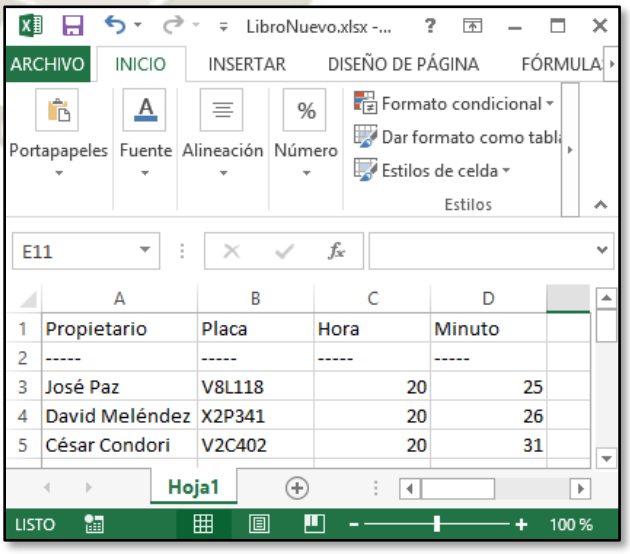


Figura 77. Registro de placas identificada con el programa (MATLAB y EXCEL)
Fuente: Elaboración Propia

5. Cuarto programa: Sistema de Identificación de Vehículos

El cuarto y último programa debe cumplir con las exigencias y deseos descritos en el cuadro de requerimientos integrándolo en una sola interfaz, la cual reúne características mostradas en los 3 programas anteriores, pero añadiendo un mayor orden y facilidad de uso para cualquier persona que lo utilice, sin necesidad de tener conocimientos del código interno.

Además de ello se probará el método de reconocimiento de caracteres descrito anteriormente: las "Redes Neuronales". La razón principal de su uso es demostrar su eficacia en la reducción de los tiempos de procesamiento de la imagen del vehículo y de esta manera reducir el uso de la CPU, permitiendo un funcionamiento más dinámico y fluido de los comandos. La interfaz y comandos utilizados se muestran y explican a continuación:

- Zona superior: Se ubican 2 objetos dinámicos. La primera es la "fecha" que se actualiza automáticamente, y la segunda es el botón de "Cambiar logo" para cambiar la imagen que aparece por defecto.
- Panel "Configuración": En esta zona se modifica la dirección IP además de poder encender y apagar la cámara a voluntad. Se debe considerar que el bloque de texto estará bloqueado cuando la cámara se encuentre encendida. La dirección admite formato de imagen o video.
- Panel "Cámara": Presenta una pantalla de imagen, además de 3 herramientas. La de la izquierda es el zoom de la cámara, realizado en el centro de la misma. La del medio es el botón que realiza la detección de la placa vehicular y el reconocimiento de sus caracteres. La última es la capacidad de elegir el modo de trabajo automático o manual. Esto se explicará en el panel de información.
- Panel "Placa Detectada": Aquí se mostrará la imagen de la placa detectada con el botón del panel anterior. En caso no reconocer una imagen clara, no se mostrará ningún resultado.

- Panel “Información”: Muestra tanto el número de la placa vehicular, además de permitir tanto la búsqueda o ingreso de datos del personal asociado a ella. Se utiliza además 3 botones que realizan las funciones de registrar la placa y usuario, exportar el documento Excel con la tabla de registro, y mostrar el avance del registro trabajado.

Figura 78. Interfaz y tabla de registro del sistema de identificación vehicular
Fuente: Elaboración Propia

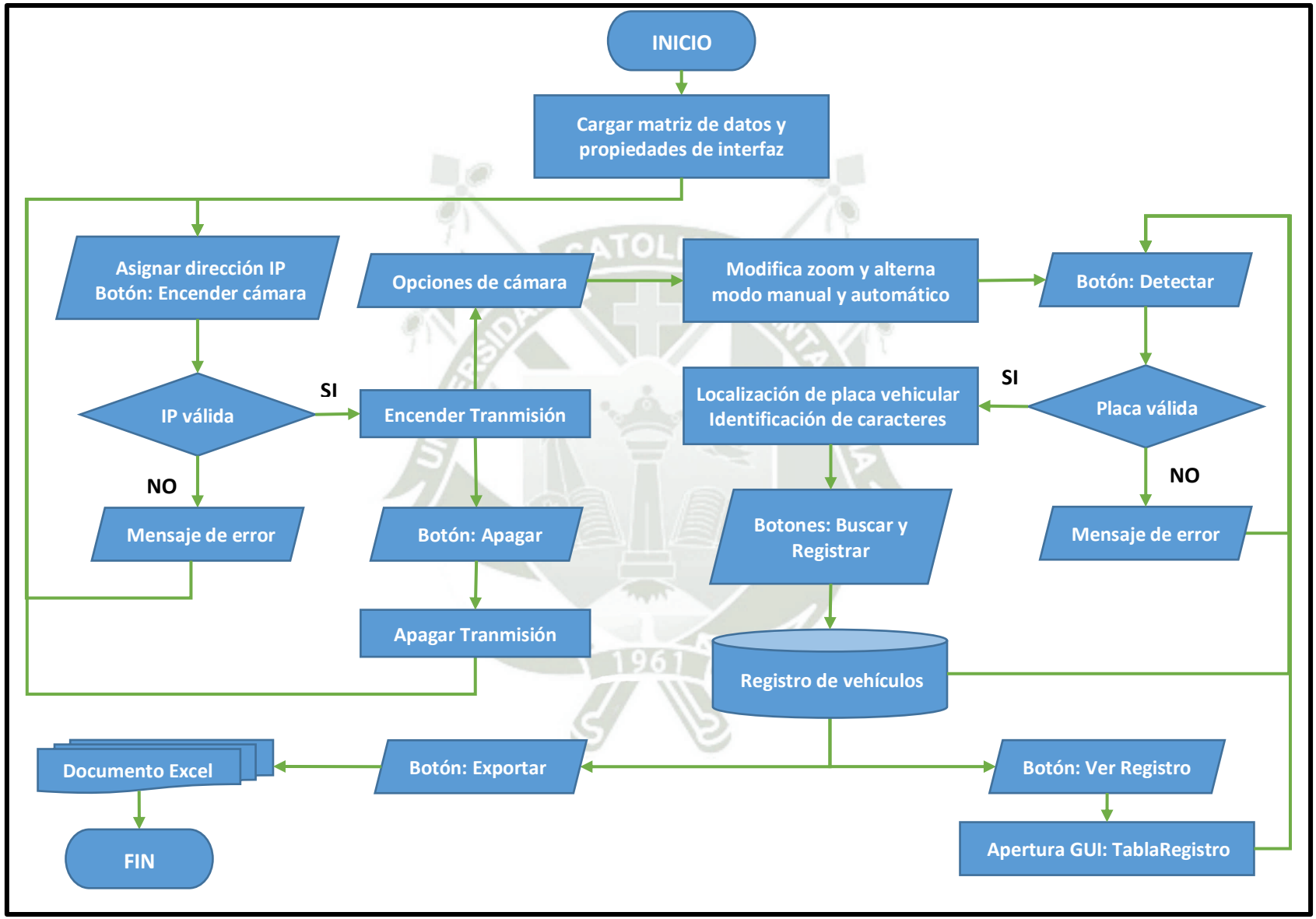


Gráfico 79. Diagrama de Flujo del 4to Programa
Fuente: Elaboración Propia

El código utilizado para el desarrollo de este programa se muestra en el apartado de Anexos, donde se muestran las variables y funciones utilizadas para cumplir con las acciones mostradas anteriormente. Este programa puede ser compilado en una aplicación para ser independiente del software MATLAB y utilizarse en otros ordenadores.

5.1. Funcionamiento del programa

- Utilizar el instalador "MyAppInstaller.mcr" el cual instalará todo lo necesario para poder ejecutar el programa, sin necesidad del software MATLAB, ni carpetas adicionales.

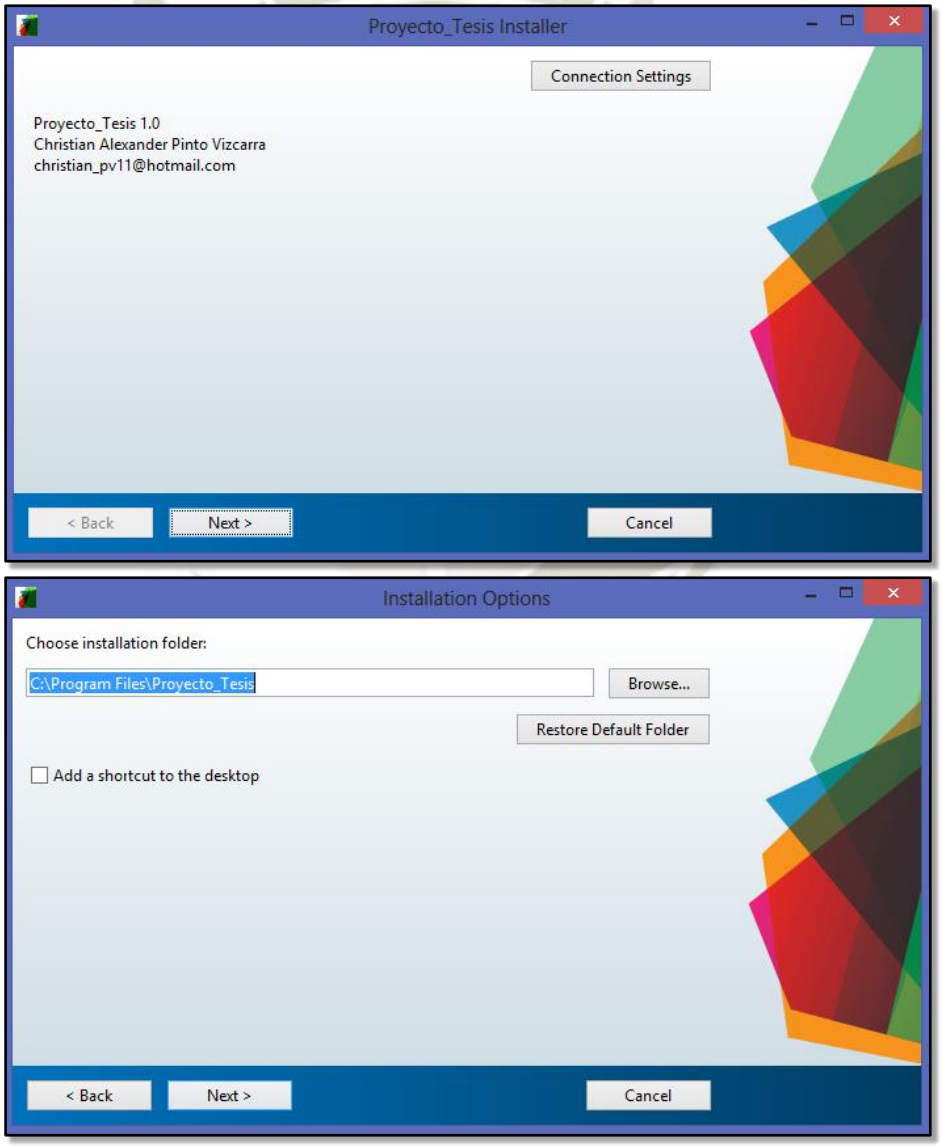
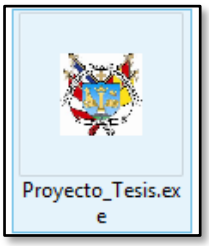


Figura 80. Instalador del programa "Proyecto_Tesis"
Fuente: Elaboración Propia

- Realizar el montaje de la cámara IP y conectar dicha cámara a un router o PC por medio de un inyector POE. Encender la cámara IP.
- Encontrar y enlazar la PC a la cámara IP utilizada con la ayuda de programas Axis Utility (cámara de marca Axis) o Axis Device Manager (cámara de otra marca).
- Definir la dirección IP a utilizar. Considerar que esta dirección sea única y no se encuentre siendo utilizada por otros dispositivos.
- En caso de la cámara Axis, se pueden definir algunos parámetros en la página web Axis Network Camera, además de también modificar la dirección IP.
- Ejecutar el ícono del programa "Proyecto_Tesis.exe".



- Una vez ejecutado el programa, modificar la dirección IP a la que hemos designado. Luego de ello apretar el botón "Encender Cámara".
- Luego de ello modificar el zoom y la opción de detección.

Figura 81. Ejemplo de uso del programa "Proyecto_Tesis"
Fuente: *Elaboración Propia*

- En caso de Manual, utilizar los botones “Buscar” y “Registrar”, mientras que, en modo Automático, con presionar Detectar se reconocerá la placa, pero no se podrá hacer uso de registrar los datos de un usuario que no se encuentre en la base de datos, por lo que solo se guardará la placa y hora de ingreso.

5.2. Restricciones

Entre las restricciones físicas de nuestro sistema tenemos:

- La cámara se encuentra en posición frontal y superior a matrícula del vehículo.
- La cámara se encuentra en un ángulo respecto a la horizontal de la matrícula que no supera los 25°, para evitar cambios en la forma de los caracteres
- No se realizarán pruebas en vehículos menores: motos, moto taxis, etc.
- No se consideran toma de imágenes nocturnas sin la iluminación adecuada.
- Es necesario tener en todo momento una PC encendida conectada a la cámara.

En el código del GUI se desarrollaron algunas restricciones, saltando un mensaje de error que permita un uso correcto y no comprometa el funcionamiento continuo del programa:

- Formato de Imagen: extensiones JPG, PNG y BMP con resoluciones mayores a 360p.
- La posición de la placa vehicular en la imagen debe estar en un bloque que representan los 2/3 de la zona inferior y del centro de la imagen, para un mayor enfoque.
- El reconocimiento de la placa no se realizará si no se ha cargado ninguna imagen.
- En caso de no lograr ubicar la placa vehicular el programa detendrá la segmentación y comparación de caracteres, a fin de evitar un error de tipo bucle sin fin.
- El programa pondrá como condición la búsqueda de la placa antes de realizar el registro o exportar algún documento Excel.
- Exportar no eliminará el registro extraído, por lo que para empezar un nuevo registro es necesario cerrar el programa.

5.3. Pruebas Iniciales y Finales

Se realizaron 3 tipos de pruebas, de las que se extrajo ideas que llevaron a la interfaz mostrada anteriormente. Se explicarán a continuación las condiciones y resultados de los mismos:

5.3.1. Primera Interfaz

El día 04 de diciembre del 2017 se realizó pruebas en el estacionamiento de San Jerónimo de la Universidad Católica de Santa María, entre las 7:30 a 9:00 a.m., para revisar la comunicación y eficacia del programa para reconocer placas de vehículos con una cámara estática en un trípode. Se utilizó la página de Axis Network Camera y el programa de MATLAB. Los resultados fueron:

- Total, de vehículos muestreados: 8.
- Cantidad de vehículos detectados: 5, de los cuales se reconoció 4 grupos de caracteres.
- Cantidad de vehículos no detectados: 3, de los cuales 1 placa se encontraba en mala posición.
- Método utilizado de comparación de caracteres: coeficiente de correlación.
- Tiempo estimado de detección de placa vehicular: 6 seg.
- Número de base de datos inicial: 0.

Figura 82. Ejemplo de placa mal posicionada durante prueba
Fuente: Elaboración Propia

A partir de estos resultados, se refinó segmentos de código, con el fin de:

- Aumentar la base de datos hasta 20 considerando los vehículos observados durante el día.
- Mejorar el procesamiento de la imagen para reducir ligeramente los tiempos y a la vez mejorar la precisión de la detección de placas.

- Agregar una herramienta para modificar el zoom de la cámara, debido a que la imagen mostrada era por defecto la utilizada en la página web.
- Eliminar por completo la necesidad de la página web, con ayuda del software IP Utility que permitía generar la dirección IP deseada, además de conectar la cámara y el ordenador.
- Empezar la implementación del método de comparación de caracteres por redes neuronales, utilizando solo 2 grupos de muestras para su entrenamiento.

Figura 83. Detección correcta de la placa de un vehículo estacionado
Fuente: Elaboración Propia

5.3.2. Segunda Interfaz

En esta interfaz se mejoró el código, permitiendo la independencia del programa con la página web de Axis. Además, se integró una base de datos inicial y la opción de realizar zoom a la imagen mostrada por la cámara estática colocada en un trípode. Las pruebas se realizaron durante 3 días (6-8 de diciembre) en el ingreso de San Jerónimo con horarios similares. Los resultados fueron:

- Total, de vehículos muestreados: 20.
- Cantidad de vehículos detectados: 15, de los cuales se reconoció 13 grupos de caracteres.
- Cantidad de vehículos no detectados: 5, de los cuales 1 placa se encontraba manchada.
- Método utilizado de comparación de caracteres: coeficiente de correlación.
- Tiempo estimado de detección de placa vehicular: 5.4 seg.
- Número de base de datos inicial: 20.

Figura 84. Ejemplo de placa con muestras de daño
Fuente: Elaboración Propia

Estos resultados se mostraban más satisfactorios, pero también mostraban deficiencias del programa e ideas para mejorar dichas condiciones:

- Tiempos: se necesita reducir el tiempo entre detecciones, debido a que 5.4 segundo resulta aún muy alto para el alto tránsito de vehículos. La mejora se enfoca en el uso de la RNA.
- Precisión: para mejorar la detección de placas se encontró que a una distancia y zoom específicos el reconocimiento mejoraba, debido a que la cantidad de información a procesar era menor y se resaltaba aún más la placa vehicular.

- Autonomía: la necesidad de tres botones independientes para detectar la placa vehicular, buscar el resultado en la base de datos y registrarlo; genera complicaciones cuando el tráfico de vehículos aumenta. Sin embargo, esto es útil para generar nuevos usuarios que no se encuentren registrados. Por lo tanto, se dispondrá de una herramienta de realizar las 3 funciones con un solo botón, sin quitar la posibilidad de realizar cada proceso por separado.

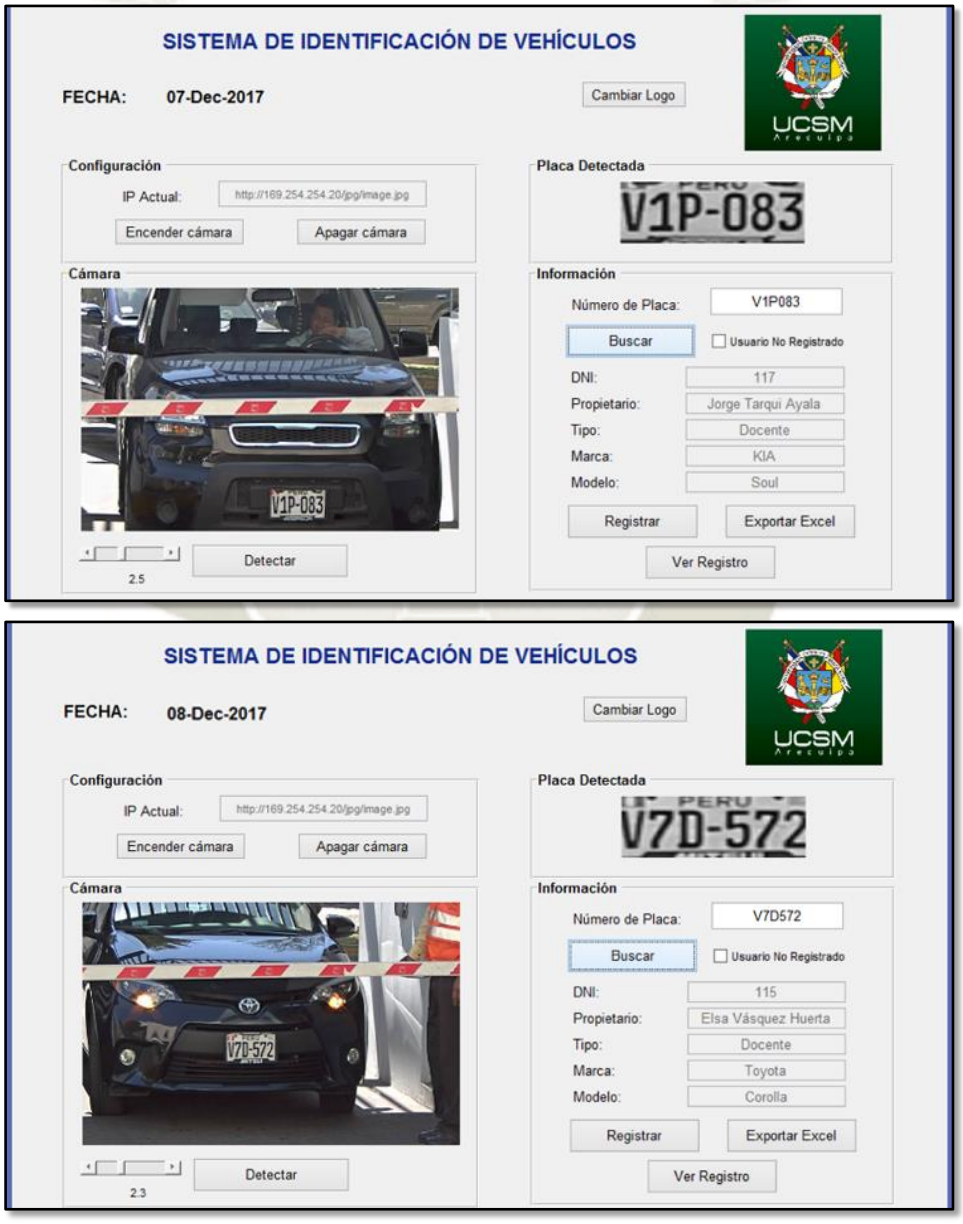


Figura 85. Pruebas de funcionamiento de la segunda interfaz del 4to programa
Fuente: Elaboración Propia

Las gráficas de resultados obtenidos se muestran a continuación:

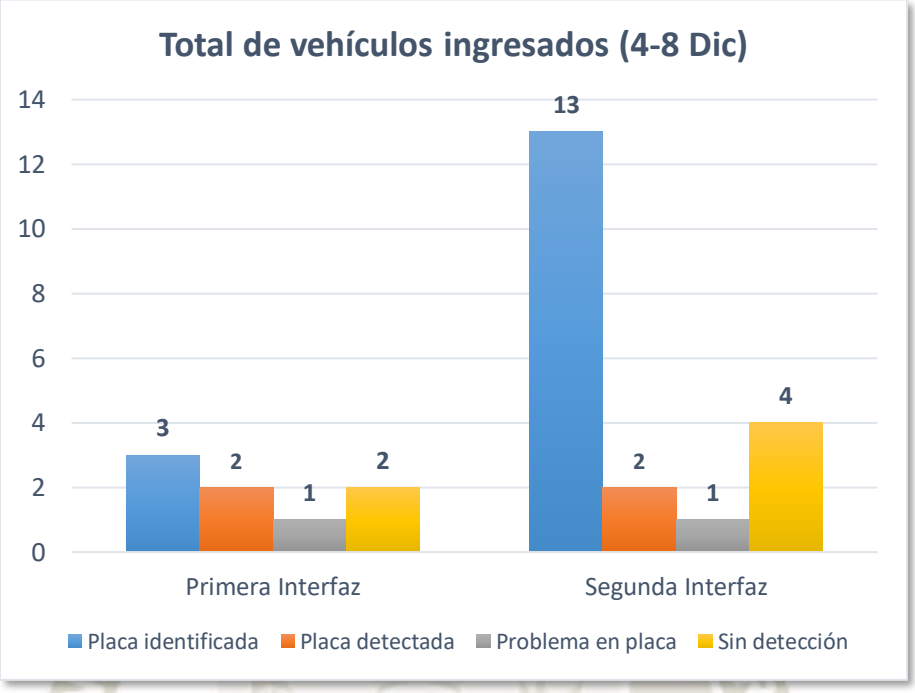


Gráfico 86. Resultados obtenidos de las pruebas entre la 1ra y 2da interfaz
Fuente: *Elaboración Propia*

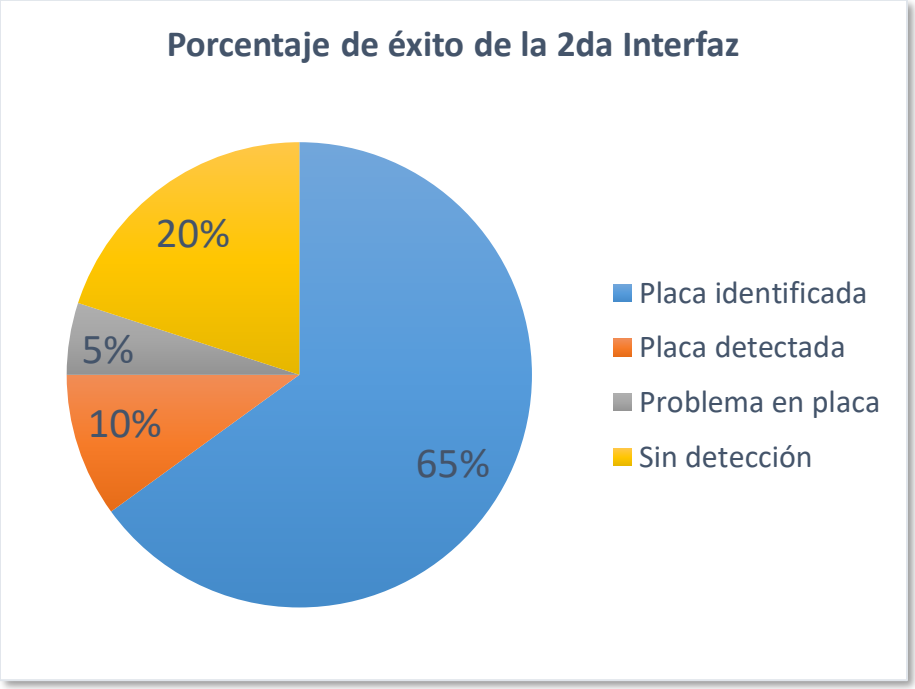


Gráfico 87. Representación del estado final de las pruebas de la 2da Interfaz
Fuente: *Elaboración Propia*

5.3.3. Tercera Interfaz

Para la última interfaz se hicieron mejoras y reducciones de código para acelerar el procesamiento y a la vez mejorar la precisión de la identificación. El principal agregado fue la adición de las RNA para observar su eficacia respecto al uso del coeficiente de correlación.

Como se explicó anteriormente, con 3 grupos de muestras se puede realizar un entrenamiento factible para el reconocimiento de caracteres, pero debido a la necesidad de reducir aún más la cantidad de errores, se intentó realizar este entrenamiento con 8 grupos de muestras, lo cual representan un total de 288 plantillas de caracteres, las cuales presentaban ruidos impulsionales de un máximo de 5% en sus valores.

Las características utilizadas para las RNA fueron:

- Porcentaje de muestras de entrenamiento: 50%
- Número de capas: 2, la capa oculta y la capa de salida
- Número de neuronas: 10 en la capa oculta y 1 en la capa de salida
- Número de salidas: 36 (los caracteres disponibles)
- Tipo de entrenamiento: Gradiente Conjugado

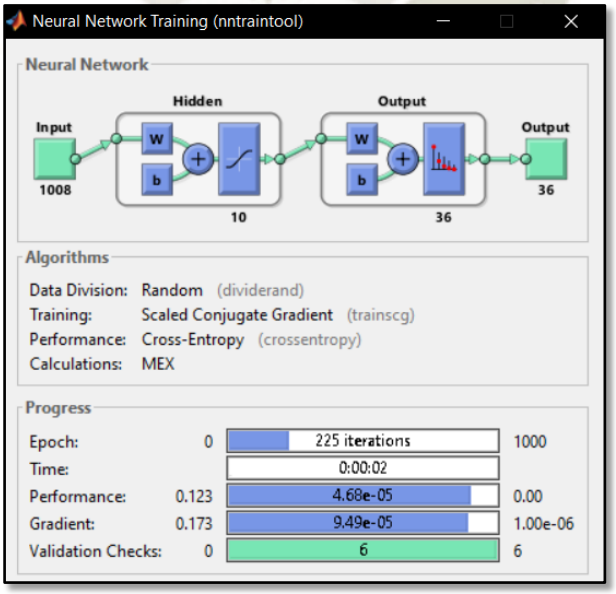


Figura 88. Red Neuronal Entrenada
Fuente: Elaboración Propia

Luego de realizar el entrenamiento se presentan gráficos para evaluar el performance, basado en distintos criterios, a continuación, los resultados:

- Rendimiento: El valor de la entropía cruzada cae hasta un valor de 6.1459e-05 en la iteración N° 219, siendo muy bajo considerando los valores utilizados en la matriz de "1" y "0". Este dato es regular tanto para los datos de entrenamiento, como validación y testeo, con lo cual muestra consistencia para abordar muestras externas a la creación de la RNA.

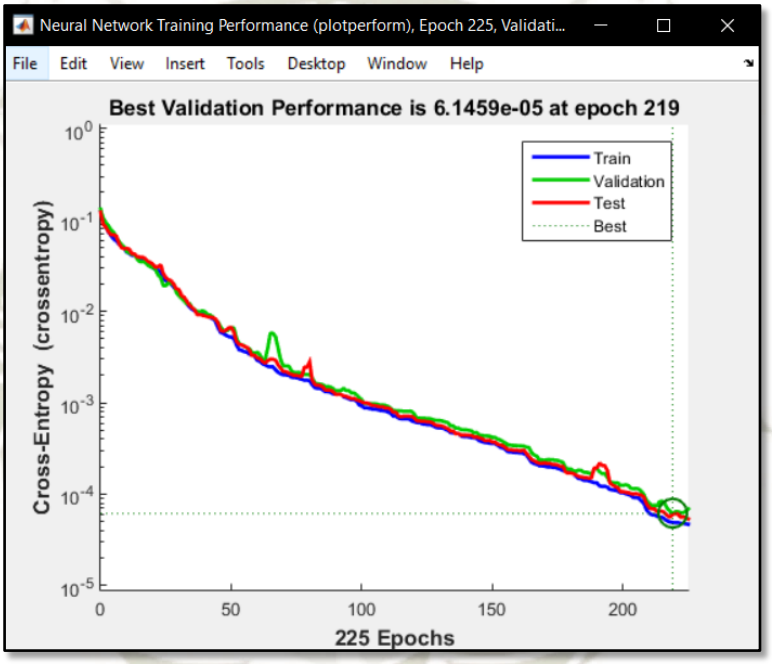


Gráfico 89. Gráfico de Rendimiento (entropía cruzada)
Fuente: Elaboración Propia

- Estado de entrenamiento: En este caso la gradiente de modificación de pesos llega hasta el valor de 9.4867e-05 en la iteración 225, donde su comportamiento se vuelve más horizontal o de poca variación. En comparación al ejemplo mostrado anteriormente de 108 muestras, con una gradiente final de 0.00274, este valor se ha reducido casi 30 veces más, lo cual demuestra su gran capacidad de aprendizaje.

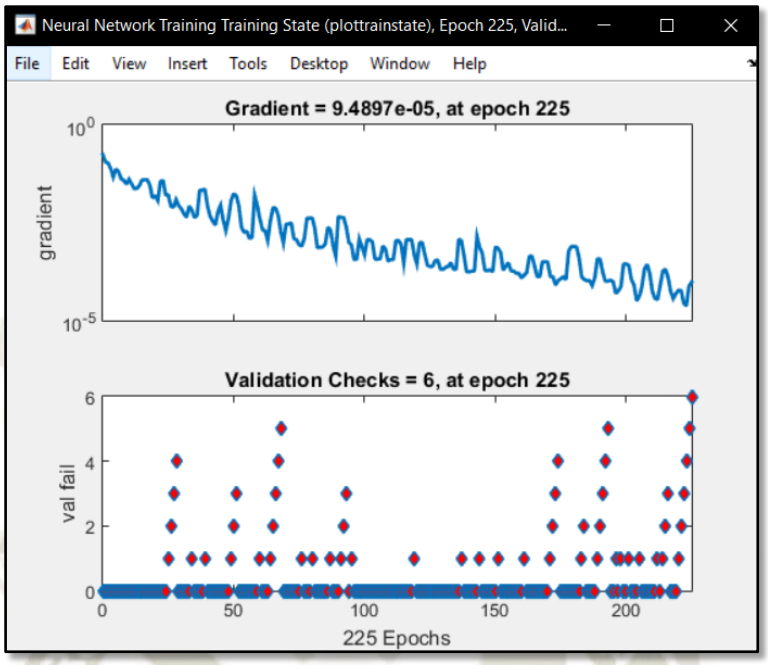


Gráfico 90. Gráfico de Rendimiento (estado de entrenamiento)
Fuente: Elaboración Propia

- Histograma de error: este resultado muestra la gran consistencia de la RNA para entregar resultados similares con errores menores bajo diferentes instancias, puesto que se obtiene un error cercano a 4.44e-05 bajo un porcentaje mayor al 90% de las probabilidades entregadas.

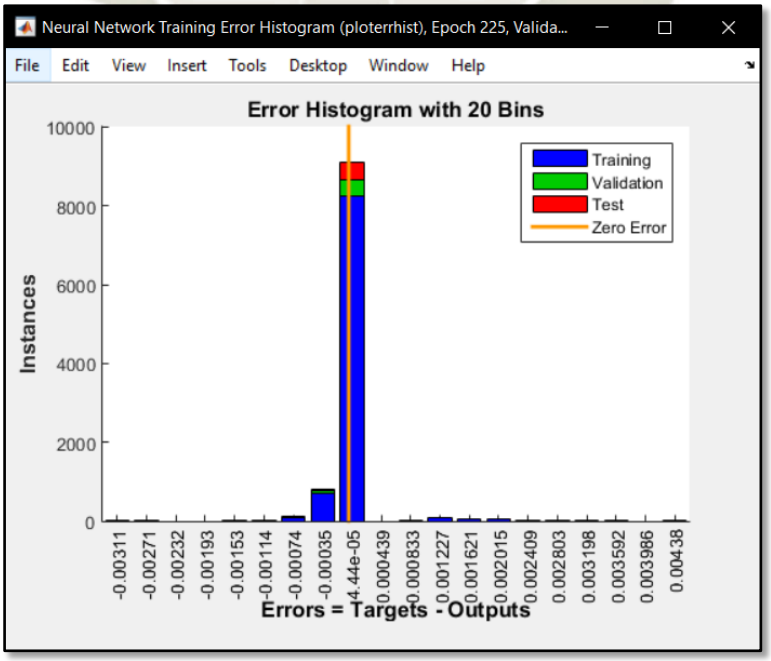


Gráfico 91. Gráfico de Rendimiento (histograma de error)
Fuente: Elaboración Propia

Continuando con la interfaz, se integró los modos “manual” y “automático”, reduciendo los tiempos y dando menos trabajo al usuario. A partir de ahí, se procedió a realizar pruebas bajo las mismas condiciones de las anteriores, durante el día 13 de diciembre en dos horarios distintos, con la disposición estática de la cámara en un trípode, ajustando la opción de zoom de la misma.

La primera prueba realizada en la mañana, entre las 8:00 y 9:30 am, contó con el uso de los 2 métodos mencionados: coeficiente de correlación y RNA. Los resultados obtenidos fueron:

Tabla 16
Comparación de métodos en pruebas de día

Coeficiente de Correlación	RNA
• Total de muestras: 25	• Total de muestras: 25
• N° Aciertos: 20	• N° Aciertos: 22
• N° Fallos de Localización: 2	• N° Fallos de Localización: 2
• N° Fallos de OCR: 3	• N° Fallos de OCR: 1
• Tiempo de Procesamiento: 5.5 seg	• Tiempo de Procesamiento: 1.2 seg
• Líneas de Código adicionales: 100	• Líneas de Código adicionales: 10

La segunda prueba realizada en la noche, entre las 6:15 y 7:45 pm, contó con el uso de los 2 métodos mencionados: coeficiente de correlación y RNA. Los resultados obtenidos fueron:

Tabla 17
Comparación de métodos en pruebas de noche

Coeficiente de Correlación	RNA
• Total de muestras: 8	• Total de muestras: 8
• N° Aciertos: 5	• N° Aciertos: 7
• N° Fallos de Localización: 1	• N° Fallos de Localización: 1
• N° Fallos de OCR: 2	• N° Fallos de OCR: 0
• Tiempo de Procesamiento: 5.2 seg	• Tiempo de Procesamiento: 1.1 seg
• Líneas de Código adicionales: 100	• Líneas de Código adicionales: 10

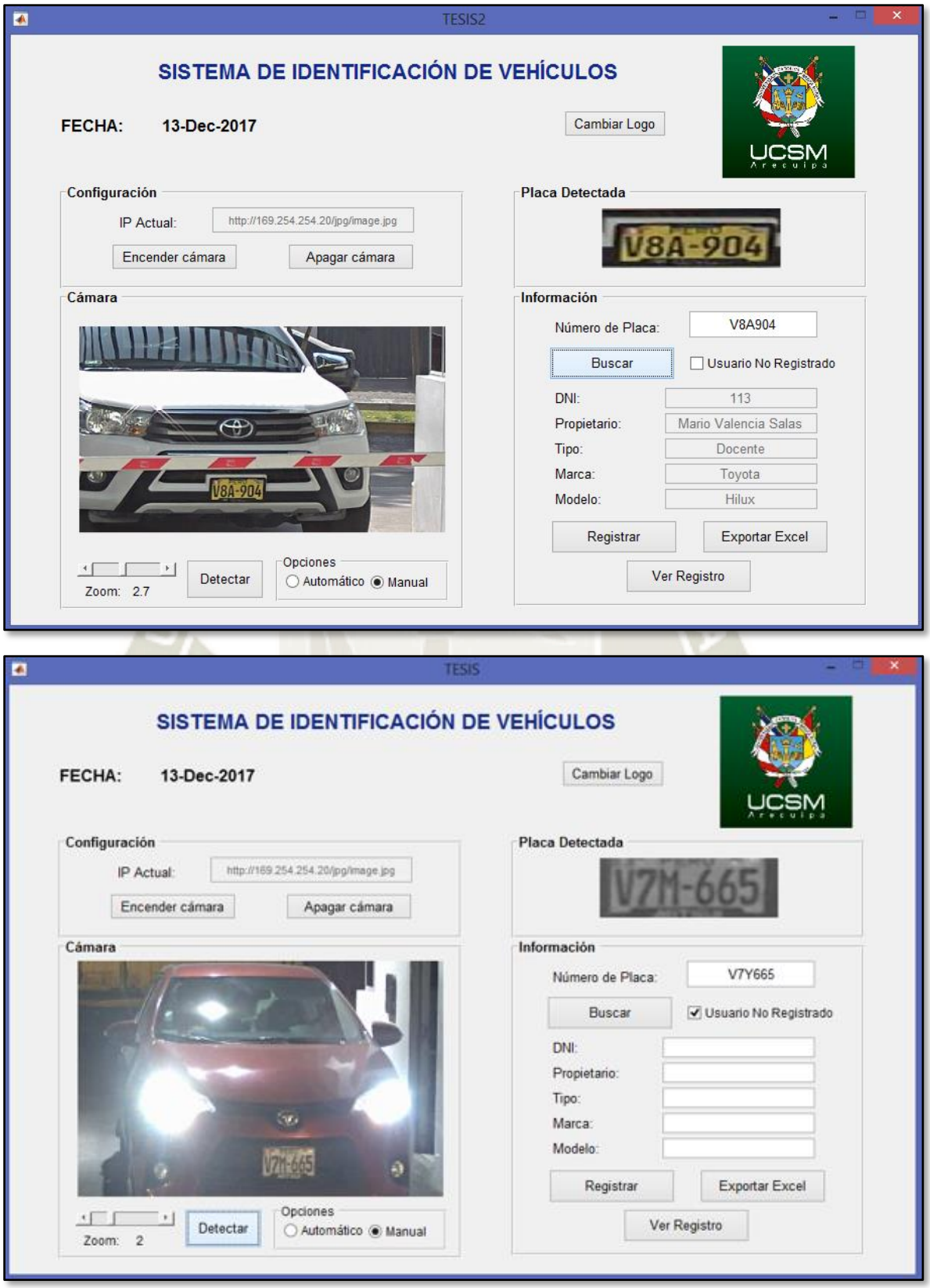


Figura 92. Pruebas realizadas el día 13/12/2017 (día y noche)
Fuente: Elaboración Propia

Las gráficas de resultados obtenidos se muestran a continuación:

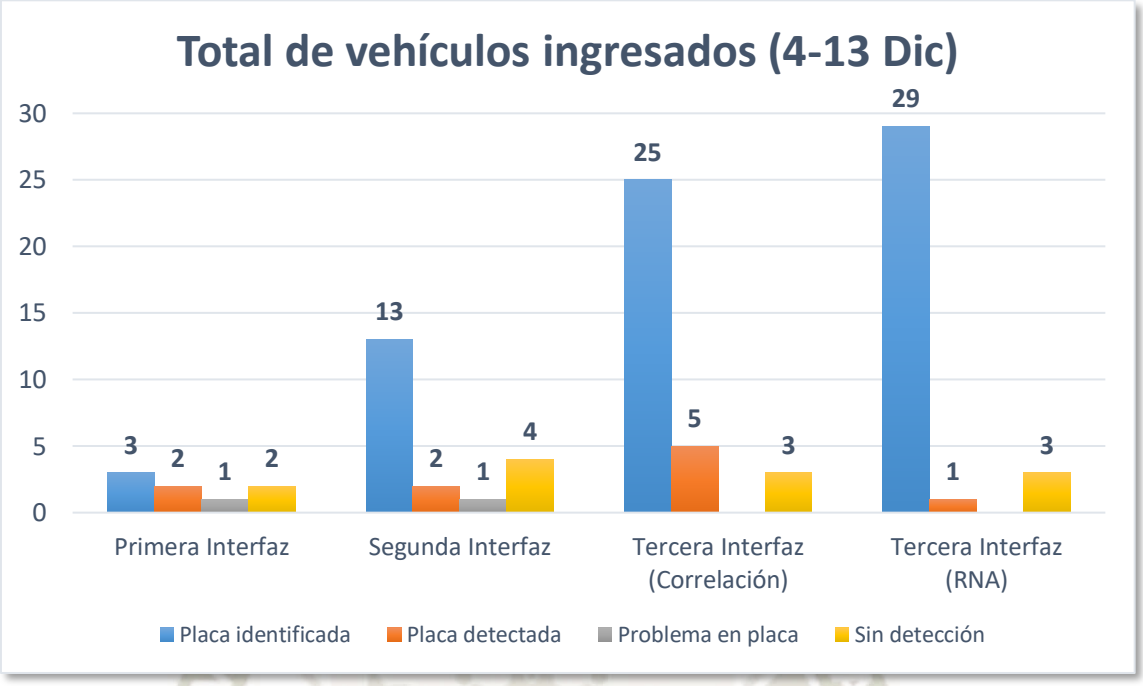


Gráfico 93. Resultados obtenidos de las pruebas entre las 3 interfaces
Fuente: *Elaboración Propia*

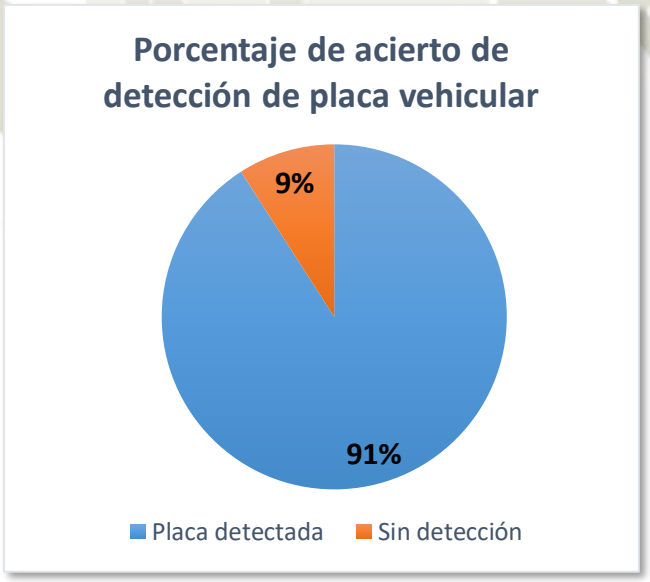


Gráfico 94. Resultados de la detección de placas por procesado morfológico
Fuente: *Elaboración Propia*

Sin considerar el incidente de la placa con el marco obstaculizando, un 9% no se pudo detectar. Sin embargo, del 91% detectado que paso al reconocimiento por RNA, el 100% fue identificado, en comparación al 78% conseguido por método de correlación.

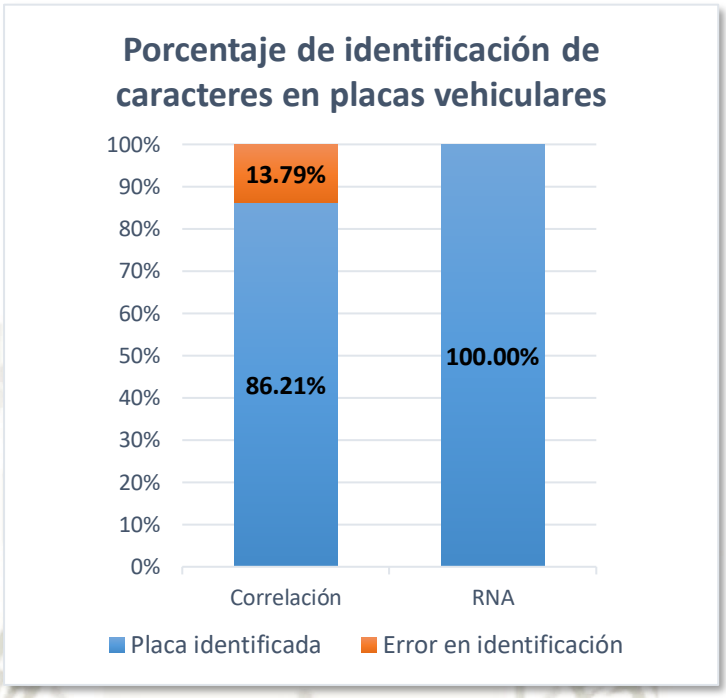


Gráfico 95. Resultados de identificación de caracteres por método de correlación y RNA
Fuente: *Elaboración Propia*

CONCLUSIONES

- Se desarrolló e implementó un sistema de identificación de placas vehiculares enfocado al ingreso de vehículos en la Universidad Católica de Santa María, el cual logró un porcentaje de reconocimiento de 91% en un día completo de prueba.

- Se estableció como valores mínimos de hardware una cámara IP de 480p para su compatibilidad con esta interfaz, el cual debe estar conectado a un router WiFi. Esto resulta en bajos costos de adquisición, al depender únicamente del software instalado en uno o varios ordenadores. Este programa es un archivo ejecutable independiente de Matlab.

- El método de procesamiento digital de imágenes en tiempo real, basado en morfología de elementos y entrenamiento de RNA, permitió obtener tiempos de procesamiento de 1.2 segundos, los cuales son menores hasta un 20% de los obtenidos por métodos de correlación (5.5 segundos) y a proyectos con la misma aplicación (2.69 segundos), debido al bajo cálculo computacional.

- La RNA es una metodología muy versátil, debido a su capacidad de aprendizaje constante, que logran resultados de hasta 100% de identificación de caracteres, mientras que técnicas como la correlación, las cuales solo describen (valoran) la comparación de matrices, obtienen resultados de 86% de acierto.

RECOMENDACIONES

- El sistema está enfocado a vehículos mayores, que presentan la matrícula en la zona delantera del vehículo, la cual a su vez debe ser nítida y sin obstáculos (marcos o protectores) que dificulten la visualización de los caracteres. Asimismo, la distancia de trabajo óptima de la cámara es de 8 a 10 m, ampliable con el uso del zoom, pero sin perder calidad de imagen.

- Se debe considerar un posicionamiento frontal de la cámara, con un ángulo no mayor a 25°, además de proveer una buena iluminación durante la noche. Asimismo, considerar la alimentación y protección continua del ordenador que ejecutará el programa durante el día.

- Realizar pruebas de funcionamiento en caso de cambio de cámara IP. Aspectos como resolución, calidad de zoom y seguridad, son configurables por software, sin embargo, la alimentación y transmisión de datos dependerá del tipo de conexionado utilizado.

- El uso de una base de datos integrada para identificación de personal puede permitir agregar información de un usuario con acceso temporal. De esta manera se lleva un mejor control para aplicaciones con un nivel de seguridad más libre.

- La cantidad de muestras requerida en el entrenamiento de la RNA para nuestro programa, varía según la precisión y tiempos de respuesta solicitados por el usuario. Considerar que ambas variables son inversamente proporcionales, por lo que un análisis previo podía resultar idóneo para indicar el número con el que se va a trabajar.

BIBLIOGRAFIA

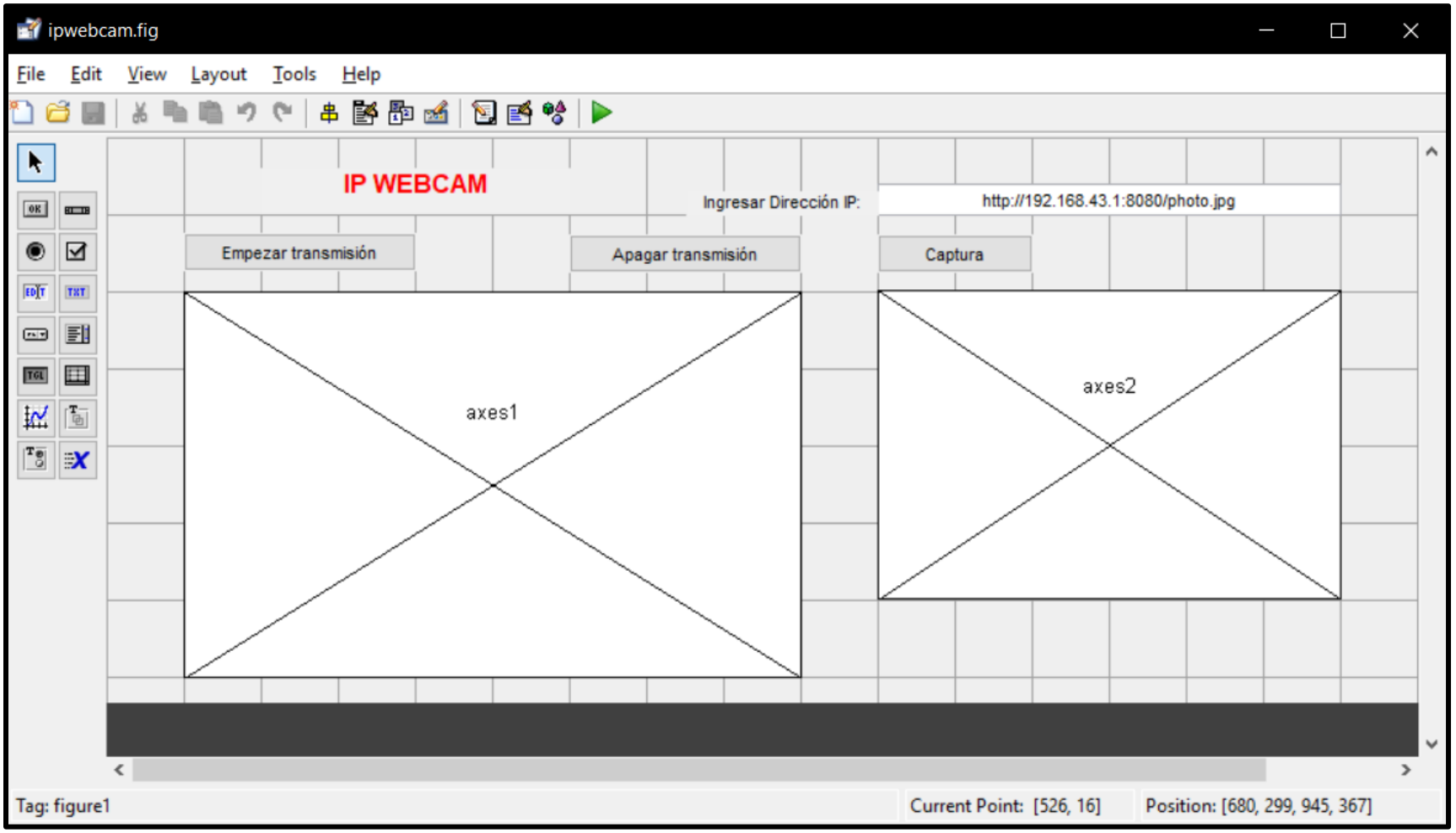
- Adobe Corporation. (2014, Enero). *Adobe Photoshop: Ayuda y tutoriales* [archivo PDF]. Recuperado de: https://helpx.adobe.com/archive/es/photoshop/cc/2015/photoshop_reference.pdf
- Alba, J. L., Martín, F., Cid, J., & Mora, I. (2006, Abril). *Morfología Matemática: Aplicación a procesado de imágenes binarias y monocromáticas* [archivo PPTX]. España: Universidad de Vigo, Universidad Carlos III, Universidad Rey Juan Carlos. Recuperado de: https://docplayer.es/59675063-Morfologia-matematica.html
- Axis Communications (2017, Marzo). *AXIS M1145-L Network Camera: Datasheet* [archivo PDF]. Recuperado de: https://www.axis.com/files/datasheet/ds_m1145l_t10054669_en_1703.pdf
- Axis Communications (2017, Febrero). *AXIS M1145-L Network Camera: Manual* [archivo PDF]. Recuperado de: https://www.axis.com/files/manuals/um_m1145l_57268_en_1702.pdf
- Axis Communications (s.f.). *AXIS Device Manager* [archivo PDF]. Recuperado de: https://www.axis.com/files/datasheet/ds_device_manager_t10091713_es_1806.pdf
- Axis Communications (s.f.). *AXIS IP Utility* [sitio Web]. Recuperado de: https://www.axis.com/es-pe/support/downloads/axis-ip-utility
- Baluarte Hurtado, R. (2015, Enero). *Diseño e Implementación de un Sistema de Video Inteligente para el Seguimiento de Patrones* [archivo PDF]. Tesis de grado publicada. Arequipa, Perú: Universidad Católica de Santa María.
- Cuevas Jimenez, E. V., & Zaldivar Navarro, D. (s.f.). *Visión por Computador utilizando MatLab y el Toolbox de Procesamiento Digital de Imágenes* [archivo PDF]. Guadalajara, México. Recuperado de: https://www.academia.edu/7448239/Visi%C3%B3n_por_Computador_utilizando_MatLAB_Y_el_Toolbox_de_Procesamiento_Digital_de_Im%C3%A1genes
- Departamento de Computación (2015). *Ruido y Filtrado* [archivo PPTX]. Buenos Aires, Argentina. Recuperado de: https://docplayer.es/26292005-Ruido-y-filtrado-clase-de-taller-i-del-28-10-2015.html

- Zoom Informática Blog (2016, Agosto). *Diferencias entre cámaras IP y sistemas CCTV* [sitio Web]. Recuperado de: http://zoominformatica.com/blog/diferencias-entre-camaras-ip-y-sistemas-cctv/
- Empretel S.A. de C.V. (s.f.). *¿Qué es y cómo funciona un PoE?* [sitio Web]. Recuperado de: http://empretel.com.mx/content/55-que-es-y-como-funciona-un-poe
- García Santillán, I. D. (2008, Enero). *Visión Artificial y Procesamiento Digital de Imágenes usando Matlab* [archivo PDF]. Libro de investigación publicado. Ibarra, Ecuador: Pontificia Universidad Católica del Ecuador.
- Gonzalez, R. C., & Woods, R. E. (2002). *Digital Image Processing (2da Ed.)* [archivo PDF]. New Jersey, EEUU: Editorial Prentice Hall.
- Hecht-Nielssen (1990). *Neurocomputing: picking the human brain* [archivo PDF]. Massachusetts, EEUU: IEEE Spectrum, Editorial Addison–Wesley.
- Khlebovich, P. (2017, Octubre). *IP Webcam App for Android* [sitio Web]. Recuperado de: https://www.appbrain.com/dev/Pavel+Khlebovich/
- León Jiménez, M. (2014). *Proyecto Fin de Carrera: Generic OCR* [archivo PPTX]. Recuperado de: http://slideplayer.es/slide/39761/
- MathWorks (s.f.). *Image Acquisition Toolbox: User Guide* [archivo PDF]. Recuperado de: https://www.mathworks.com/help/pdf_doc/imaq/imaq_ug.pdf
- MathWorks (s.f.). *Image Processing Toolbox: User Guide* [archivo PDF]. Recuperado de: https://www.mathworks.com/help/pdf_doc/images/images_tb.pdf
- Mundaca Vidarte, G. A. (2016, Diciembre). *Detección de caracteres de placas de automóviles mediante técnicas de visión artificial* [archivo PDF]. Tesis de grado publicada. Piura, Perú: Universidad de Piura.

- Nieto Bonilla, D. (2010). *Sistema de Reconocimiento de Kanjis Japoneses basado en Procesamiento Digital de Imágenes aplicado a Dispositivos Móviles* [archivo PDF]. Tesis de grado publicada. México: Universidad de las Américas Puebla.
- Parker, J. R. (2010). *Algorithms for Image Processing and Computer Vision (2da Ed.)* [archivo PDF]. Libro de investigación publicado. Indiana, EEUU: Editorial John Wiley & Sons Inc.
- Platero Dueñas, C. (2008). *Apuntes de Visión Artificial; Introducción a la visión artificial* [archivo PDF]. Libro de investigación publicado. Madrid, España: Universidad Politécnica de Madrid.
- Polanco Carrillo, F., Álvarez Gerónimo, D., & Moreno Vega, V. (2015, Setiembre). *Implementación mediante hardware de una Red Neuronal Artificial para Reconocimiento de Caracteres* [archivo PDF]. Artículo de investigación publicado. La Habana, Cuba: Universidad Tecnológica de la Habana.
- Rebaza, J. V. (2007, Octubre). *Detección de bordes mediante el algoritmo de Canny* [archivo PDF]. Artículo de investigación publicado. La Libertad, Perú: Universidad Nacional de Trujillo.
- Alonso Romero, L. (2010). *Reconocimiento de patrones con redes neuronales* [archivo PDF]. Libro de Investigación publicado. Salamanca, España: Universidad de Salamanca, Facultad de Ciencias.
- Salazar Márquez, M. (2014). *Desarrollo de un Algoritmo para la Localización Automática de Placas Vehiculares Peruanas usando Técnicas de Procesamiento de Imágenes* [archivo PDF]. Tesis de grado publicada. Lima, Perú: Pontificia Universidad Católica del Perú.
- Wainschenker, R., Massa, J. M., & Tristán, P. (2011). *Procesamiento Digital de Imágenes* [archivo PPTX]. Curso de Taller publicado. Buenos Aires, Argentina: Universidad Nacional del Centro de la Provincia de Buenos Aires. Recuperado de: http://www.academia.edu/5378686/Procesamiento_Digital_de_Im%C3%A1genes

ANEXOS

Anexo 1: Interfaz Android IP Webcam



Anexo 2: Código – 1er Programa Android IP Webcam

```
function varargout = ipwebcam(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ipwebcam_OpeningFcn, ...
                   'gui_OutputFcn',  @ipwebcam_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function ipwebcam_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;
```

```
%% DATOS DE LA INTERFAZ
global ip
guidata(hObject, handles);
axes(handles.axes1);       box off;
set(gca,'xcolor',get(gcf,'color')); set(gca,'xtick',[]);
set(gca,'ycolor',get(gcf,'color')); set(gca,'ytick',[]);
axes(handles.axes2);       box off;
set(gca,'xcolor',get(gcf,'color')); set(gca,'xtick',[]);
set(gca,'ycolor',get(gcf,'color')); set(gca,'ytick',[]);
ip = 0;

function varargout = ipwebcam_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
```

```
%% ENCENDER Y APAGAR LA TRANSMISIÓN DE LA CAMARA
function ON_Callback(hObject, eventdata, handles)
global ss ip
ip = 1; axes(handles.axes1); axis off;
url = get(handles.DIRECCION,'String');
ss = imread(url); fh = image(ss);
while(1) ss = imread(url); set(fh,'CData',ss); drawnow;
end

function OFF_Callback(hObject, eventdata, handles)
global ip
ip = 0; axes(handles.axes1); cla(gca,'reset'); box off;
set(gca,'xcolor',get(gcf,'color')); set(gca,'xtick',[]);
set(gca,'ycolor',get(gcf,'color')); set(gca,'ytick',[]);
```

```
%% REALIZA LA CAPTURA DE LA IMAGEN
function Captura_Callback(hObject, eventdata, handles)
global ss ip
if ip == 1
    axes(handles.axes2); himage = imshow(ss);
else
    msg = msgbox('Encender la transmisión primero','ANPR');
end

function DIRECCION_Callback(hObject, eventdata, handles)
function DIRECCION_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Anexo 3: Código – 2do Programa OCR de Imágenes

```
clear all; close all; clc
```

```
%% BASE DE DATOS DE CARACTERES
% Cargamos las todas las imágenes binarias de letras (A-Z) preelaboradas en variables
diferentes.
a=imread('A1.bmp'); b=imread('B1.bmp'); c=imread('C1.bmp'); d=imread('D1.bmp');
e=imread('E1.bmp'); f=imread('F1.bmp'); g=imread('G1.bmp'); h=imread('H1.bmp');
i=imread('I1.bmp'); j=imread('J1.bmp'); k=imread('K1.bmp'); l=imread('L1.bmp');
m=imread('M1.bmp'); n=imread('N1.bmp'); o=imread('O1.bmp'); p=imread('P1.bmp');
q=imread('Q1.bmp'); r=imread('R1.bmp'); s=imread('S1.bmp'); t=imread('T1.bmp');
u=imread('U1.bmp'); v=imread('V1.bmp'); w=imread('W1.bmp'); x=imread('X1.bmp');
y=imread('Y1.bmp'); z=imread('Z1.bmp');

% Cargamos las imágenes binarias de números (0-9) preelaboradas en variables diferentes.
uno=imread('1a.bmp');    dos=imread('2a.bmp');   tres=imread('3a.bmp');
cuatro=imread('4a.bmp'); cinco=imread('5a.bmp'); seis=imread('6a.bmp');
siete=imread('7a.bmp');  ocho=imread('8a.bmp');  nueve=imread('9a.bmp');
cero=imread('0a.bmp');

% Creamos un arreglo de matrices con las imágenes de letras y números preelaborados
alfabeto = [a b c d e f g h i j k l m n o p q r s t u v w x y z uno dos tres cuatro cinco seis
siete ocho nueve cero];
caracteres = mat2cell(alfabeto,42,[repmat(24,1,36)]);
```

```
%% LECTURA DE IMAGENES
a = 'Placa4.jpg';
I = imread(a);
subplot(1,3,1), imshow(I), title('Original')
```

```
%% PREPROCESAMIENTO
% Escala de Grises y Escalado
info=imfinfo(a);
if (info.ColorType == 'truecolor')  % Determinar el formato (color o escala de grises)
    I=rgb2gray(I);
else
    I=I;
end
I2 = imresize(I,[300 500],'bicubic');         % Nueva imagen de 400x300
subplot(1,3,2), imshow(I2), title('Escala de Grises')

% Recortado
I3 = imcrop(I2,[150 150 200 150]);   % Corte estándar para disminuir datos
subplot(1,3,3), imshow(I3), title('Imagen recortada')

% Filtro Morfológico (Bottom Hat)
bh = strel('disk',5);                        % Bajar el valor si es necesario
I4 = imbothat(I3,bh)*3;                      % Filtro Bottom Hat
figure, subplot(1,2,1), imshow(I4), title('Bottom-hat')

% Binarización
I5 = im2bw(I4); subplot(1,2,2), imshow(I5), title('Binarización')
```

```
%% OPERACIONES MORFOLOGICAS
% Cierre
Oc = strel('disk',5);      I6 = imclose(I5,Oc);
figure, subplot(2,2,1), imshow(I6), title('Cierre')

% Apertura
Oa = strel('disk',5);      I7 = imopen(I6,Oa);
subplot(2,2,2), imshow(I7), title('Apertura')

% Dilatación Vertical
Odv = strel('line',10,90); I8 = imdilate(I7,Odv);
subplot(2,2,3), imshow(I8), title('Dilatación Vertical')

% Dilatación Horizontal
Odh = strel('line',15,0);  I9 = imdilate(I8,Odh);
subplot(2,2,4), imshow(I9), title('Dilatación Horizontal')
```

```
%% EXTRACCION DE PLACA VEHICULAR
L = bwlabel(I9);                % Crea regiones (etiqueta componentes conectados)
n = max(max(L))                 % Cantidad de elementos
stats = regionprops(L, 'all');  % Obtiene la región

% Encontrar figura centrada
for i=1:n
    cuadro(i,:) = stats(i).BoundingBox;
end
cuadro;
if n==1
    cuadroF = stats(1).BoundingBox
else
    ii = 1;    vm = 0;
    for i=1:n
        if 100>cuadro(i,1) && (cuadro(i,1)+cuadro(i,3))>100 && cuadro(i,3)
        if cuadro(i,2)>vm
            vm = cuadro(i,2);           ii = i;
        end
        end
    end
    cuadroF = stats(ii).BoundingBox
end

% Eliminar los espacio negros de barras metálicas
I10 = imcrop(I9,(cuadroF)); T = size(I10); hr = T(2); vr = T(1); n=0;
if (cuadroF(3)>75)||(cuadroF(4)>40)
    condsize=0;
    while(condsize==0)
        xir = 0; yir = 0; c11 = 1; c12 = 1; c21 = 1; c22 = 1; c3 = 1; c4 = 1;
        xfr = hr; yfr = vr; xir1 = 0; xir2 = 0;
        for i=(round(hr/2)-2):-1:1
            if I10(round(vr/2)-1,i)==0 && c11 == 1, xir1 = i+1; c11 = 0; end
            if I10(round(vr/2)+1,i)==0 && c12 == 1, xir2 = i+1; c12 = 0; end
            if xir1 > xir2, xir=xir1; else, xir=xir2; end
        end
        xfr1 = hr; xfr2 = hr;
        for i=(round(hr/2)+2):1:hr
            if I10(round(vr/2)-1,i)==0 && c21 == 1, xfr1 = i-1; c21 = 0; end
            if I10(round(vr/2)+1,i)==0 && c22 == 1, xfr2 = i-1; c22 = 0; end
            if xfr1 > xfr2, xfr=xfr2; else, xfr=xfr1; end
        end
        if xir==0 || xfr == hr
            for i=(round(vr/2)-2):-1:1
            if I10(i,round(hr/2))==0 && c3 == 1, yir = i+1, c3 = 0; end
            end
            for i=(round(vr/2)+2):1:vr
            if I10(i,round(hr/2))==0 && c4 == 1, yfr = i-1, c4 = 0; end
            end
        else
            for i=(round(vr/2)-1):-1:1
            if I10(i,xir-2)==1 && c3 == 1, yir = i+1, c3 = 0; end
            end
            for i=(round(vr/2)+1):1:vr
            if I10(i,xfr+2)==1 && c4 == 1, yfr = i-1, c4 = 0; end
            end
        end
        cuadroF = [cuadroF(1)+xir cuadroF(2)+yir xfr-xir-1 yfr-yir-1];
        I10 = imcrop(I9,(cuadroF));
        T = size(I10); hr = T(2); vr = T(1);
        if (sum(I10(:,1))+sum(I10(:,hr)))>(2*vr-5) && (sum(I10(1,:))+sum(I10(vr,:)))>(2*hr-5)
            condsize=1
        else
            cuadroF = cuadroF+[1 1 -2 -2];
            I10 = imcrop(I9,(cuadroF));
            T = size(I10); hr = T(2); vr = T(1);
        end
    end
end
```

```
I10 = imcrop(I3,(cuadroF));
I10 = imresize(I10,[100 300],'bicubic');
figure, subplot(1,3,1), imshow(I10), title('Imagen de la Placa')

% Binarización y filtrado de la Placa
bh1 = strel('disk',100);    I11 = imbothat(I10,bh1)*2.7;
I11 = im2bw(I11);           I11 = bwareaopen(I11,40);
subplot(1,3,2), imshow(I11), title('Binarización y filtro')
```

```
%% DETECCION DE CARACTERES
% Recorta bordes horizontales
for i=1:300;     I11v(1,i) = sum(I11(:,i));  end,    I11v;
cond1 = 0; i = 1;
while(cond1==0)
    if 90>I11v(1,i)
        h1 = i;         cond1 = 1;
    end
    i=i+1;
end
cond1 = 0; i = 300;
while(cond1==0)
    if 90>I11v(1,i)
        h2 = i;         cond1 = 1;
    end
    i=i-1;
end
h1;h2;

I12 = imcrop(I11,[h1+1 1 h2-h1-2 100]);
I12 = imresize(I12,[100 300],'bicubic');
subplot(1,3,3), imshow(I12), title('Caracteres')

% Recorta bordes verticales
for i=1:100;     I12h(i,1) = sum(I12(i,:));  end,    I12h;
cond1 = 0; i = 1; c1 = 0;
while(cond1==0)
    if c1==I12h(i,1)
        ii = i;         cond1 = 1;
    end
    i=i+1;
    if i==50
        i=1;            c1=c1+1;
    end
end
ii;
I12 = imcrop(I12,[1 ii 300 100-ii+1]);
I12 = imresize(I12,[100 300],'bicubic');
subplot(1,3,3), imshow(I12), title('Caracteres')

for i=1:100;     I12h(i,1) = sum(I12(i,:));  end,    I12h;
cond1 = 0; ii = 1;
while(cond1==0)
    if I12h(ii,1)>100
        v1 = ii;         cond1 = 1;
    end
    if I12h(ii+1,1)==300
        cond1 = 0;
    end
    ii=ii+1;
end
cond1 = 0; i = 100;
while(cond1==0)
    if 250>I12h(i,1)
        v2 = i;         cond1 = 1;
    end
    i=i-1;
end

I12 = imcrop(I12,[1 v1+1 300 v2-v1-2]);
I12 = imresize(I12,[100 300],'bicubic');
subplot(1,3,3), imshow(I12), title('Caracteres')
```

```
% Recorte de la Placa (Eliminar letras de nacionalidad)
I12 = bwareaopen(I12,1000);
L2 = bwlabel(I12);                  % Crea regiones (etiqueta componentes conectados)
n2 = max(max(L2))                   % Cantidad de elementos
stats2 = regionprops(L2, 'all');    % Obtiene las Estadísticas de las regiones
subplot(1,3,3), imshow(I12), title('Caracteres')
n3 = 1;
while (n3~=6)
    for i = 1:n2
        E = stats2(i).BoundingBox;
        if E(3)>=30 && E(4)>=40
            if E(3)>=50
                ni = round(E(3)/45)
                for i=1:(ni-1)
                    Error = round(E(1))+round(E(3)/ni)*i
                    I12(:,(Error))=0;
                end
            end
        end
    end
    I12 = bwareaopen(I12,800);
    L2 = bwlabel(I12);                  % Crea regiones (etiqueta componentes conectados)
    n2 = max(max(L2))                   % Cantidad de elementos
    stats2 = regionprops(L2, 'all');    % Obtiene las Estadísticas de las regiones
    subplot(1,3,3), imshow(I12), title('Caracteres')
    n3 = 1;
    for i = 1:n2
        E = stats2(i).BoundingBox;
        if E(3)>=30 && E(4)>=40
                stats3(n3,:)=E;
                n3=n3+1;
        end
    end
    n3=n3-1
    stats3
end
    subplot(1,3,3), imshow(I12), title('Caracteres')
```

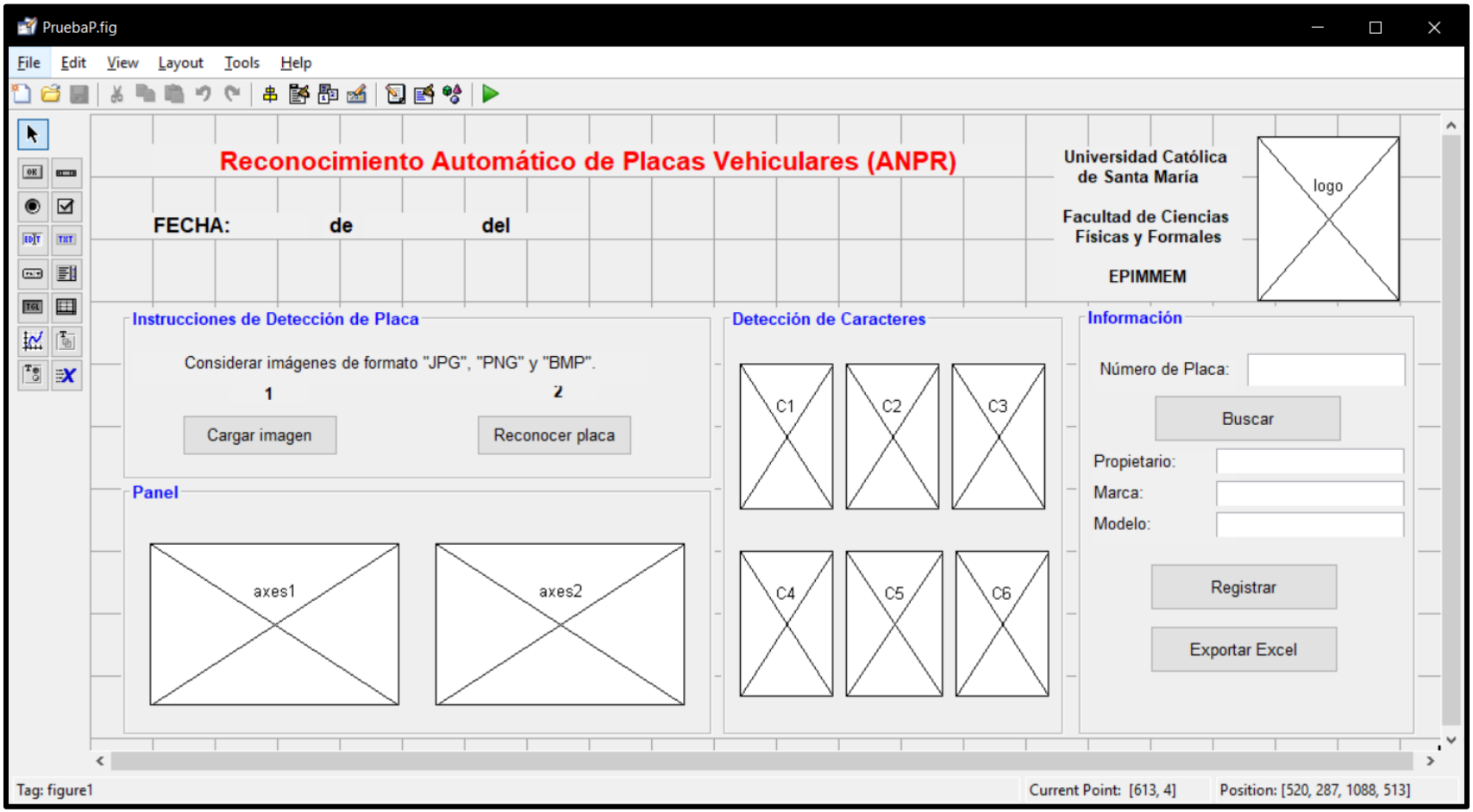
```
%% EXTRACCION DE CARACTERES
for fila=1:n3
    E = stats3(fila,:);
    if E(4)>50
        C = imcrop(I12,E);
        Caracter = imresize(C,[42 24]);   %[192 117]
        cond = 0;
            if (sum(Caracter(1,:))==0) || (sum(Caracter(1,:))==24)
                C = imcrop(Caracter,[1 4 24 39]); Caracter = imresize(C,[42 24]); cond = 1;
            end
            if (sum(Caracter(42,:))==0) || (sum(Caracter(42,:))==24)
                C = imcrop(Caracter,[1 1 24 39]); Caracter = imresize(C,[42 24]); cond = 1;
            end
            if (sum(Caracter(:,1))==0) || (sum(Caracter(:,1))==42)
                C = imcrop(Caracter,[4 1 21 42]); Caracter = imresize(C,[42 24]); cond = 1;
            end
            if (sum(Caracter(:,24))==0) || (sum(Caracter(:,24))==42)
                C = imcrop(Caracter,[1 1 21 42]); Caracter = imresize(C,[42 24]); cond = 1;
            end
        figure, imshow(Caracter);
        % Trabajamos solo con las Letras de la placa
        % Ciclo que calcula los coeficientes de correlación
        R = [Caracter];
        RC = mat2cell(R,42,[24]);
        plc = RC{1,1};                  % Toma la letra
        pos = 1; temp = 0;

        while pos <= 36                 % Recorre todos los caracteres
            temp = caracteres{1,pos};   % Toma un caracter de la matriz
            co = corr2(temp,plc);       % Calcula el coeficiente de correlación
            % Para placas peruanas
```

```
                if fila == 1 && pos >= 27
                    letra(fila,pos) = 0;        % Guarda las correlaciones en una matriz
                elseif fila >= 5 && pos <= 26
                    letra(fila,pos) = 0;
                else
                    letra(fila,pos) = co;
                end
            pos = pos+1;
        end
        letra;
        maxs=max(letra,[],2);                    % Calcula la correlación más alta de cada fila
        [posx posy]=find(letra==maxs(fila,1));  % Busca la fila y la columna de la correlación
más alta
        letras(fila)=posy;                       % Guarda la posición de la correlación más alta
en el arreglo "letras"
    else
        letras(fila)=37;
    end
end
letras
```

```
%% IDENTIFICACION DE CARACTERES
% Cambio de posición por caracter
for lt = 1:n3 % son 3 letras
    if letras(lt)== 1,       car(lt)='A';    elseif letras(lt) == 2,     car(lt)='B';
    elseif letras(lt) == 3,  car(lt)='C';    elseif letras(lt) == 4,     car(lt)='D';
    elseif letras(lt) == 5,  car(lt)='E';    elseif letras(lt) == 6,     car(lt)='F';
    elseif letras(lt) == 7,  car(lt)='G';    elseif letras(lt) == 8,     car(lt)='H';
    elseif letras(lt) == 9,  car(lt)='I';    elseif letras(lt) == 10,    car(lt)='J';
    elseif letras(lt) == 11, car(lt)='K';    elseif letras(lt) == 12,    car(lt)='L';
    elseif letras(lt) == 13, car(lt)='M';    elseif letras(lt) == 14,    car(lt)='N';
    elseif letras(lt) == 15, car(lt)='O';    elseif letras(lt) == 16,    car(lt)='P';
    elseif letras(lt) == 17, car(lt)='Q';    elseif letras(lt) == 18,    car(lt)='R';
    elseif letras(lt) == 19, car(lt)='S';    elseif letras(lt) == 20,    car(lt)='T';
    elseif letras(lt) == 21, car(lt)='U';    elseif letras(lt) == 22,    car(lt)='V';
    elseif letras(lt) == 23, car(lt)='W';    elseif letras(lt) == 24,    car(lt)='X';
    elseif letras(lt) == 25, car(lt)='Y';    elseif letras(lt) == 26,    car(lt)='Z';
    elseif letras(lt) == 27, car(lt)='1';    elseif letras(lt) == 28,    car(lt)='2';
    elseif letras(lt) == 29, car(lt)='3';    elseif letras(lt) == 30,    car(lt)='4';
    elseif letras(lt) == 31, car(lt)='5';    elseif letras(lt) == 32,    car(lt)='6';
    elseif letras(lt) == 33, car(lt)='7';    elseif letras(lt) == 34,    car(lt)='8';
    elseif letras(lt) == 35, car(lt)='9';    elseif letras(lt) == 36,    car(lt)='0';
    else
    car(lt)='-';
end
lt=lt+1;
end
car
```

Anexo 4: Interfaz – 3er Programa GUI de Reconocimiento e Identificación



Anexo 5: Código – 3er Programa GUI de Reconocimiento e Identificación

```
function varargout = PruebaP(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PruebaP_OpeningFcn, ...
                   'gui_OutputFcn',  @PruebaP_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% --- Executes just before PruebaP is made visible.
function PruebaP_OpeningFcn(hObject, ~, handles, varargin)

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

% Update handles structure
guidata(hObject, handles);
```

```
%% BASE DE DATOS DE CARACTERES
global caracteres
% Cargamos las todas las imágenes binarias de letras (A-Z) preelaboradas en variables
diferentes.
a=imread('A1.bmp'); b=imread('B1.bmp'); c=imread('C1.bmp'); d=imread('D1.bmp');
e=imread('E1.bmp'); f=imread('F1.bmp'); g=imread('G1.bmp'); h=imread('H1.bmp');
i=imread('I1.bmp'); j=imread('J1.bmp'); k=imread('K1.bmp'); l=imread('L1.bmp');
m=imread('M1.bmp'); n=imread('N1.bmp'); o=imread('O1.bmp'); p=imread('P1.bmp');
q=imread('Q1.bmp'); r=imread('R1.bmp'); s=imread('S1.bmp'); t=imread('T1.bmp');
u=imread('U1.bmp'); v=imread('V1.bmp'); w=imread('W1.bmp'); x=imread('X1.bmp');
y=imread('Y1.bmp'); z=imread('Z1.bmp');

% Cargamos las imágenes binarias de números (0-9) preelaboradas en variables diferentes.
uno=imread('1a.bmp'); dos=imread('2a.bmp');
tres=imread('3a.bmp'); cuatro=imread('4a.bmp');
cinco=imread('5a.bmp'); seis=imread('6a.bmp');
siete=imread('7a.bmp'); ocho=imread('8a.bmp');
nueve=imread('9a.bmp'); cero=imread('0a.bmp');

% Creamos un arreglo de matrices con las imágenes de letras y números preelaborados
alfabeto = [a b c d e f g h i j k l m n o p q r s t u v w x y z uno dos tres cuatro cinco seis
siete ocho nueve cero];
caracteres = mat2cell(alfabeto,42,[repmat(24,1,36)]);
```

```
%% DEFINIMOS VARIABLES
% Establecemos el día de operación
reloj = clock;
if reloj(2)==1; mes = 'Enero'; end
if reloj(2)==2; mes = 'Febrero'; end
if reloj(2)==3; mes = 'Marzo'; end
if reloj(2)==4; mes = 'Abril'; end
if reloj(2)==5; mes = 'Mayo'; end
if reloj(2)==6; mes = 'Junio'; end
if reloj(2)==7; mes = 'Julio'; end
if reloj(2)==8; mes = 'Agosto'; end
if reloj(2)==9; mes = 'Setiembre'; end
if reloj(2)==10; mes = 'Octubre'; end
if reloj(2)==11; mes = 'Noviembre'; end
if reloj(2)==12; mes = 'Diciembre'; end
set(handles.Day,'String',reloj(3));
set(handles.Month,'String',mes);
```

```
set(handles.Year,'String',reloj(1));

global T fil x xx
T = {'Propietario', 'Placa', 'Hora', 'Minuto'; '-----','-----','-----','-----'};
fil = 3;
x = 0; xx = 0;
logo = imread('logo.png');
axes(handles.logo); imshow(logo);

% --- Outputs from this function are returned to the command line.
function varargout = PruebaP_OutputFcn(~, ~, handles)
varargout{1} = handles.output;
```

```
%% SUBIR IMAGEN A GUI
% --- Executes on button press in Cargar.
function Cargar_Callback(~, ~, handles)
global I,
axes(handles.axes2); cla('reset'); set(handles.axes2,'Visible','off');
axes(handles.C1); cla('reset'); set(handles.C1,'Visible','off');
axes(handles.C2); cla('reset'); set(handles.C2,'Visible','off');
axes(handles.C3); cla('reset'); set(handles.C3,'Visible','off');
axes(handles.C4); cla('reset'); set(handles.C4,'Visible','off');
axes(handles.C5); cla('reset'); set(handles.C5,'Visible','off');
axes(handles.C6); cla('reset'); set(handles.C6,'Visible','off');
[f,p] = uigetfile({'*.png;*.jpg;*.bmp;*.tif','Supported images';...
                   '*.png','Portable Network Graphics (*.png)';...
                   '*.jpg','J-PEG (*.jpg)';...
                   '*.bmp','Bitmap (*.bmp)';...
                   '*.tif','Tagged Image File (*.tif,)';...
                   '*.*','All files (*.*)'});
I = imread([p f]);
axes(handles.axes1);set(handles.axes1,'Visible','on');
himage = imshow(I);
title('Vehículo');
```

```
%% RECONOCIMIENTO DE PLACA VEHICULAR
% --- Executes on button press in Reconocer.
function Reconocer_Callback(hObject, eventdata, handles)
% hObject    handle to Reconocer (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global I caracteres
%% Escala de Grises y Escalado
I=rgb2gray(I);
I2 = imresize(I,[300 500],'bicubic');          % Nueva imagen de 400x300

%% Recortado
I3 = imcrop(I2,[150 150 200 150]);    % Corte estándar para disminuir datos

%% Filtro Morfológico (Bottom Hat)
bh = strel('disk',5);                          % Bajar el valor si es necesario
I4 = imbothat(I3,bh)*3;                        % Filtro Bottom Hat

%% Binarización
I5 = im2bw(I4);

%% Operaciones
% Cierre
Oc = strel('disk',5); I6 = imclose(I5,Oc);
% Apertura
Oa = strel('disk',5); I6 = imopen(I6,Oa);
% Dilatación Vertical
Odv = strel('line',10,90); I6 = imdilate(I6,Odv);
% Dilatación Horizontal
Odh = strel('line',15,0); I6 = imdilate(I6,Odh);

%% Extracción de placa
L = bwlabel(I6);                % Crea regiones (etiqueta componentes conectados)
n = max(max(L))                 % Cantidad de elementos
stats = regionprops(L, 'all');  % Obtiene la región
% Encontrar figura centrada
```

```
for i=1:n
    cuadro(i,:) = stats(i).BoundingBox;
end
cuadro;
if n==1
    cuadroF = stats(1).BoundingBox
else
    ii = 1;
    vm = 0;
    for i=1:n
        if 100>cuadro(i,1) && (cuadro(i,1)+cuadro(i,3))>100 && cuadro(i,3)
        if cuadro(i,2)>vm
            vm = cuadro(i,2);
            ii = i;
        end
        end
    end
    cuadroF = stats(ii).BoundingBox
end

% Eliminar los espacio negros de barras metalicas
I7 = imcrop(I6,(cuadroF)); T = size(I7); hr = T(2); vr = T(1); n=0;
if (cuadroF(3)>75)||(cuadroF(4)>40)
    condsize=0;
    while(condsize==0)
        xir = 0; yir = 0; c11 = 1; c12 = 1; c21 = 1; c22 = 1; c3 = 1; c4 = 1;
        xfr = hr; yfr = vr;
        xir1 = 0; xir2 = 0;
        for i=(round(hr/2)-2):-1:1
            if I7(round(vr/2)-1,i)==0 && c11 == 1, xir1 = i+1; c11 = 0; end
            if I7(round(vr/2)+1,i)==0 && c12 == 1, xir2 = i+1; c12 = 0; end
            if xir1 > xir2, xir=xir1; else, xir=xir2; end
        end
        xfr1 = hr; xfr2 = hr;
        for i=(round(hr/2)+2):1:hr
            if I7(round(vr/2)-1,i)==0 && c21 == 1, xfr1 = i-1; c21 = 0; end
            if I7(round(vr/2)+1,i)==0 && c22 == 1, xfr2 = i-1; c22 = 0; end
            if xfr1 > xfr2, xfr=xfr2; else, xfr=xfr1; end
        end
        if xir==0 || xfr == hr
            for i=(round(vr/2)-2):-1:1
            if I7(i,round(hr/2))==0 && c3 == 1
                yir = i+1, c3 = 0;
            end
            end
            for i=(round(vr/2)+2):1:vr
            if I7(i,round(hr/2))==0 && c4 == 1
                yfr = i-1, c4 = 0;
            end
            end
        else
            for i=(round(vr/2)-1):-1:1
            if I7(i,xir-2)==1 && c3 == 1
                yir = i+1, c3 = 0;
            end
            end
            for i=(round(vr/2)+1):1:vr
            if I7(i,xfr+2)==1 && c4 == 1
                yfr = i-1, c4 = 0;
            end
            end
        end

        cuadroF = [cuadroF(1)+xir cuadroF(2)+yir xfr-xir-1 yfr-yir-1];
        I7 = imcrop(I6,(cuadroF));
        T = size(I7); hr = T(2); vr = T(1);
        if (sum(I7(:,1))+sum(I7(:,hr)))>(2*vr-5) && (sum(I7(1,:))+sum(I7(vr,:)))>(2*hr-5)
            condsize=1
        else
            cuadroF = cuadroF+[1 1 -2 -2];
            I7 = imcrop(I6,(cuadroF));
```

```
            T = size(I7); hr = T(2); vr = T(1);
        end
    end
end

I7 = imcrop(I3,(cuadroF));
I7 = imresize(I7,[100 300],'bicubic');
set(handles.axes2,'Visible','on');
axes(handles.axes2), imshow(I7), title('Placa detectada')

%% Binarización de la Placa
bh1 = strel('disk',100);
I8 = imbothat(I7,bh1)*2.7;
I8 = im2bw(I8);
I8 = bwareaopen(I8,40);

%% Detección de Caracteres
% Recorta bordes horizontales
for i=1:300;     I8v(1,i) = sum(I8(:,i));  end,    I8v;
cond1 = 0; i = 1;
while(cond1==0)
    if 90>I8v(1,i)
        h1 = i;         cond1 = 1;
    end
    i=i+1;
end
cond1 = 0; i = 300;
while(cond1==0)
    if 90>I8v(1,i)
        h2 = i;         cond1 = 1;
    end
    i=i-1;
end
h1;h2;

I8 = imcrop(I8,[h1+1 1 h2-h1-2 100]);
I8 = imresize(I8,[100 300],'bicubic');

% Recorta bordes verticales
for i=1:100;     I8h(i,1) = sum(I8(i,:));  end,    I8h;
cond1 = 0; i = 1; c1 = 0;
while(cond1==0)
    if c1==I8h(i,1)
        ii = i;         cond1 = 1;
    end
    i=i+1;
    if i==50
        i=1;         c1=c1+1;
    end
end
I8 = imcrop(I8,[1 ii 300 100-ii+1]);
I8 = imresize(I8,[100 300],'bicubic');

for i=1:100;     I8h(i,1) = sum(I8(i,:));  end,    I8h;
cond1 = 0; ii = 1;
while(cond1==0)
    if I8h(ii,1)>100
        v1 = ii;         cond1 = 1;
    end
    if I8h(ii+1,1)==300
        cond1 = 0;
    end
    ii=ii+1;
end
cond1 = 0; i = 100;
while(cond1==0)
    if 250>I8h(i,1)
        v2 = i;         cond1 = 1;
    end
    i=i-1;
end
```

```
v1;v2;

I8 = imcrop(I8,[1 v1+1 300 v2-v1-2]);
I8 = imresize(I8,[100 300],'bicubic');

%% Recorte de la Placa (Eliminar letras de nacionalidad)
I9 = bwareaopen(I8,1000);
L2 = bwlabel(I9);                   % Crea regiones (etiqueta componentes conectados)
n2 = max(max(L2))                    % Cantidad de elementos
stats2 = regionprops(L2, 'all');     % Obtiene las Estadísticas de las regiones

n3 = 1;
while (n3~=6)
    for i = 1:n2
        E = stats2(i).BoundingBox;
        if E(3)>=30 && E(4)>=40
            if E(3)>=50
                ni = round(E(3)/45)
                for i=1:(ni-1)
                    Error = round(E(1))+round(E(3)/ni)*i
                    I9(:,(Error-1:1:Error))=0;
                end
            end
        end
    end
    I9 = bwareaopen(I9,800);
    L2 = bwlabel(I9);                   % Crea regiones (etiqueta componentes conectados)
    n2 = max(max(L2))                    % Cantidad de elementos
    stats2 = regionprops(L2, 'all');     % Obtiene las Estadísticas de las regiones
    n3 = 1;
    for i = 1:n2
        E = stats2(i).BoundingBox;
        if E(3)>=30 && E(4)>=40
                stats3(n3,:)=E;
                n3=n3+1;
            end
        end
    n3=n3-1
end

%% Ubicación del caracter
for fila=1:n3
    E = stats3(fila,:);
    if E(4)>50
        C = imcrop(I9,E);
        Caracter = imresize(C,[42 24]);   %[192 117]
        cond = 0;
            if (sum(Caracter(1,:))==0) || (sum(Caracter(1,:))==24)
                C = imcrop(Caracter,[1 4 24 39]); Caracter = imresize(C,[42 24]); cond = 1;
            end
            if (sum(Caracter(42,:))==0) || (sum(Caracter(42,:))==24)
                C = imcrop(Caracter,[1 1 24 39]); Caracter = imresize(C,[42 24]); cond = 1;
            end
            if (sum(Caracter(:,1))==0) || (sum(Caracter(:,1))==42)
                C = imcrop(Caracter,[4 1 21 42]); Caracter = imresize(C,[42 24]); cond = 1;
            end
            if (sum(Caracter(:,24))==0) || (sum(Caracter(:,24))==42)
                C = imcrop(Caracter,[1 1 21 42]); Caracter = imresize(C,[42 24]); cond = 1;
            end
        if fila == 1, set(handles.C1,'Visible','on'); axes(handles.C1); imshow(Caracter); end
        if fila == 2, set(handles.C2,'Visible','on'); axes(handles.C2); imshow(Caracter); end
        if fila == 3, set(handles.C3,'Visible','on'); axes(handles.C3); imshow(Caracter); end
        if fila == 4, set(handles.C4,'Visible','on'); axes(handles.C4); imshow(Caracter); end
        if fila == 5, set(handles.C5,'Visible','on'); axes(handles.C5); imshow(Caracter); end
        if fila == 6, set(handles.C6,'Visible','on'); axes(handles.C6); imshow(Caracter); end
        % Trabajamos solo con las Letras de la placa
        % Ciclo que calcula los coeficientes de correlación
        R = [Caracter];
        RC = mat2cell(R,42,[24]);
        plc = RC{1,1};                  % Toma la letra
        pos = 1; temp = 0;
```

```
        while pos <= 36                 % Recorre todos los caracteres
            temp = caracteres{1,pos};   % Toma un caracter de la matriz
            co = corr2(temp,plc);       % Calcula el coeficiente de correlación
            % Para placas peruanas
                if fila == 1 && pos >= 27
                    letra(fila,pos) = 0;        % Guarda las correlaciones en una matriz
                elseif fila >= 4 && pos <= 26
                    letra(fila,pos) = 0;
                else
                    letra(fila,pos) = co;
                end
            pos = pos+1;
        end

        maxs=max(letra,[],2);                    % Calcula la correlación más alta de cada fila
        [posx posy]=find(letra==maxs(fila,1));  % Busca la fila y la columna de la correlación
más alta
        letras(fila)=posy;                       % Guarda la posición de la correlación más alta
en el arreglo "letras"
    else
        letras(fila)=37;
        C = imcrop(I12,[E(1) E(2)-30 E(3) E(4)+60]);
        Caracter = imresize(C,[42 24]);
        if fila == 3, set(handles.C3,'Visible','on'); axes(handles.C3); imshow(Caracter); end
        if fila == 4, set(handles.C4,'Visible','on'); axes(handles.C4); imshow(Caracter); end
    end
end
letras;

%% Cambio de posición por caracter
for lt = 1:n3 % son 3 letras
    if letras(lt)== 1,          car(lt)='A';
    elseif letras(lt) == 2,     car(lt)='B';
    elseif letras(lt) == 3,     car(lt)='C';
    elseif letras(lt) == 4,     car(lt)='D';
    elseif letras(lt) == 5,     car(lt)='E';
    elseif letras(lt) == 6,     car(lt)='F';
    elseif letras(lt) == 7,     car(lt)='G';
    elseif letras(lt) == 8,     car(lt)='H';
    elseif letras(lt) == 9,     car(lt)='I';
    elseif letras(lt) == 10,    car(lt)='J';
    elseif letras(lt) == 11,    car(lt)='K';
    elseif letras(lt) == 12,    car(lt)='L';
    elseif letras(lt) == 13,    car(lt)='M';
    elseif letras(lt) == 14,    car(lt)='N';
    elseif letras(lt) == 15,    car(lt)='O';
    elseif letras(lt) == 16,    car(lt)='P';
    elseif letras(lt) == 17,    car(lt)='Q';
    elseif letras(lt) == 18,    car(lt)='R';
    elseif letras(lt) == 19,    car(lt)='S';
    elseif letras(lt) == 20,    car(lt)='T';
    elseif letras(lt) == 21,    car(lt)='U';
    elseif letras(lt) == 22,    car(lt)='V';
    elseif letras(lt) == 23,    car(lt)='W';
    elseif letras(lt) == 24,    car(lt)='X';
    elseif letras(lt) == 25,    car(lt)='Y';
    elseif letras(lt) == 26,    car(lt)='Z';
    elseif letras(lt) == 27,    car(lt)='1';
    elseif letras(lt) == 28,    car(lt)='2';
    elseif letras(lt) == 29,    car(lt)='3';
    elseif letras(lt) == 30,    car(lt)='4';
    elseif letras(lt) == 31,    car(lt)='5';
    elseif letras(lt) == 32,    car(lt)='6';
    elseif letras(lt) == 33,    car(lt)='7';
    elseif letras(lt) == 34,    car(lt)='8';
    elseif letras(lt) == 35,    car(lt)='9';
    elseif letras(lt) == 36,    car(lt)='0';
    else
    car(lt)='-';
end
```

```
lt=lt+1;
end
set(handles.Placa,'String',car);
```

%% ESTABLECE PROPIEDADES DE ELEMENTOS DE LA GUI

```
% --- Executes during object creation, after setting all properties.
function Placa_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function prop_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function marca_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function modelo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function N_placa_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function N_prop_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function N_marca_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function N_modelo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

%% BUSCA LA PLACA IDENTIFICADA EN BASE DE PERSONAL

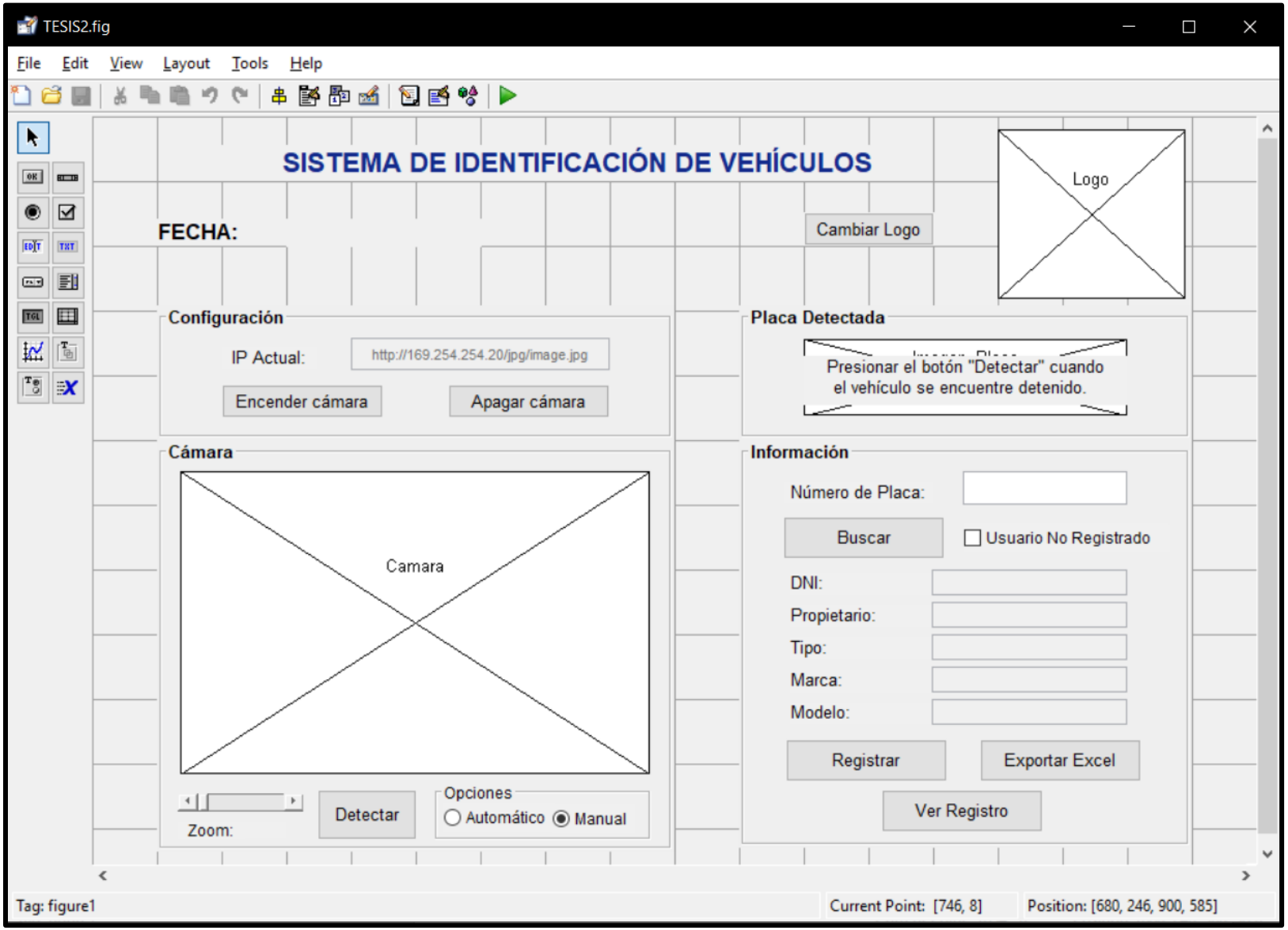
```
% --- Executes on button press in Buscar.
function Buscar_Callback(hObject, eventdata, handles)
global xx x
[n1, base, n2] = xlsread('Base_de_datos.xlsx',1);
h = size(base); h = h(1,1);
car = get(handles.Placa,'String'); car = char(car);
x = 0;
t = char(base(2,1));
if size(car) == size(t)
for i=2:1:h
    n = char(base(i,1));
    if n==car
        inf1 = char(base(i,2));        set(handles.prop,'String',inf1);
        inf2 = char(base(i,3));        set(handles.marca,'String',inf2);
        inf3 = char(base(i,4));        set(handles.modelo,'String',inf3);
        x = 1;
```

```
    end
end
end
if x==1;
    msg = msgbox('Personal encontrado.','ANPR');
    xx = 1;
else
    msg = msgbox('No se encuentra en la base de datos.','ANPR');
    set(handles.prop,'String','');
    set(handles.marca,'String','');
    set(handles.modelo,'String','');
    xx = 0;
end
```

```
%% REGISTRO DE ENTRADA DE PERSONAL
% --- Executes on button press in Registrar.
function Registrar_Callback(hObject, eventdata, handles)
global xx fil x T
% if h == 0
% else
%     fil = h+1
% end
if xx == 0
    msg = msgbox('Buscar una placa antes de registrar','ANPR');
else
    if x == 0
        msg = msgbox('Este usuario ya ha sido registrado. Por favor elegir otro.','ANPR');
    else
        t_placa = get(handles.Placa,'String');
        t_prop = get(handles.prop,'String');
        format short g
        time = clock;
        hour = time(4);
        minute = time(5);
        T(fil,:) = {t_prop, t_placa, hour, minute}
        fil = fil+1;
        x = 0;
        msg = msgbox('El usuario ha sido registrado con éxito','ANPR');
    end
end

% --- Executes on button press in Exportar.
function Exportar_Callback(hObject, eventdata, handles)
global T
filename = 'LibroNuevo.xlsx';
xlswrite(filename,T,1,'A1')
end
```

Anexo 6: Interfaz – 4to Programa Sistema de Identificación de Vehículos



Anexo 7: Código – 4to Programa Sistema de Identificación de Vehículos

```
function varargout = TESIS(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @TESIS_OpeningFcn, ...
                   'gui_OutputFcn',  @TESIS_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% --- Executes just before TESIS is made visible.
function TESIS_OpeningFcn(hObject, ~, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
```

```
%% CONFIGURACIÓN INICIAL
% Logo
logo = imread('UCSM.png'); axes(handles.Logo); imshow(logo);
% Fecha y Hora
set(handles.Fecha,'String',date);
% Camara
axes(handles.Camara);      box off;
set(gca,'xcolor',get(gcf,'color')); set(gca,'xtick',[]);
set(gca,'ycolor',get(gcf,'color')); set(gca,'ytick',[]);
global zoom
zoom = 1; set(handles.Zoom,'Value',1); set(handles.N_Zoom,'String',1);

% Tabla
global x1 x2 fil T
T = {'---','---','---','---','---','---','---'};
fil = 1; x1 = 0; x2 = 0; % x1 : Personal encontrado, x2 : Personal registrado

% --- Outputs from this function are returned to the command line.
function varargout = TESIS_OutputFcn(~, ~, handles)
varargout{1} = handles.output;

% --- Executes on button press in CambiarLogo.
function CambiarLogo_Callback(~, ~, handles)
[f,p] = uigetfile({'*.png;*.jpg;*.bmp;*.tif','Supported images';...
                   '*.png','Portable Network Graphics (*.png)';...
                   '*.jpg','J-PEG (*.jpg)';...
                   '*.bmp','Bitmap (*.bmp)';...
                   '*.tif','Tagged Image File (*.tif,)';...
                   '*.*','All files (*.*)'});
I = imread([p f]);
axes(handles.Logo); imshow(I);

%% HERRAMIENTA ZOOM
% --- Executes on slider movement.
function Zoom_Callback(hObject, eventdata, handles)
global zoom
zoom = get(hObject,'Value');
zoom = round(zoom,1);
set(handles.N_Zoom,'String',zoom);

%% HERRAMIENTA MANUAL O AUTOMATICO
% --- Executes on button press in Opt1.
function Opt1_Callback(hObject, eventdata, handles)
set(handles.I_Placa,'Enable','off');
set(handles.Buscar,'Enable','off');
```

```
% --- Executes on button press in Opt2.
function Opt2_Callback(hObject, eventdata, handles)
set(handles.I_Placa,'Enable','on');
set(handles.Buscar,'Enable','on');
```

```
%% PROPIEDADES DE ELEMENTOS DE GUI
% --- Executes during object creation, after setting all properties.
function IP_CreateFcn(hObject, ~, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function I_Propietario_CreateFcn(hObject, ~, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function I_Marca_CreateFcn(hObject, ~, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function I_Modelo_CreateFcn(hObject, ~, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function I_Tipo_CreateFcn(hObject, ~, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function I_DNI_CreateFcn(hObject, ~, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function Zoom_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes during object creation, after setting all properties.
function I_Placa_CreateFcn(hObject, ~, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
%% BUSQUEDA Y REGISTRO DE PERSONAL IDENTIFICADO
% --- Executes on button press in Buscar.
function Buscar_Callback(~, ~, handles)
global x2 x1
% x1 : Personal encontrado
% x2 : Personal registrado
set(handles.Usuario,'Value',0);
[dni, base, ~] = xlsread('Base de datos.xlsm','REGISTRO');
base([1 2],:)=[]; base(:,1)=[];
h = size(base); h = h(1,1);
car = get(handles.I_Placa,'String'); car = char(car);
x1 = 0; t = char(base(2,1));
if size(car) == size(t)
for i=1:1:h
    n = char(base(i,1));
    if n==car
```

```
        inf1 = dni(i,1);
        set(handles.I_DNI,'Enable','off'); set(handles.I_DNI,'String',inf1);
        inf2 = char(base(i,2));
        set(handles.I_Propietario,'Enable','off'); set(handles.I_Propietario,'String',inf2);
        inf3 = char(base(i,3));
        set(handles.I_Tipo,'Enable','off'); set(handles.I_Tipo,'String',inf3);
        inf4 = char(base(i,4));
        set(handles.I_Marca,'Enable','off'); set(handles.I_Marca,'String',inf4);
        inf5 = char(base(i,5));
        set(handles.I_Modelo,'Enable','off');set(handles.I_Modelo,'String',inf5);
        x1 = 1;
    end
end
end
if x1==1;
    msgbox('Personal encontrado en la base de datos.','Registro');
    x2 = 1;
else
    msgbox('No se encuentra en la base de datos.','Registro');
    set(handles.I_DNI,'String','');
    set(handles.I_Propietario,'String','');
    set(handles.I_Tipo,'String','');
    set(handles.I_Marca,'String','');
    set(handles.I_Modelo,'String','');
    x2 = 0;
end

% --- Executes on button press in Registrar.
function Registrar_Callback(hObject, ~, handles)
global x2 fil x1 T
if x2 == 0
    msgbox('Buscar una placa antes de registrar','Instrucción');
else
    if x1 == 0
        msgbox('Este usuario ya ha sido registrado. Por favor elegir otro.','Registro');
    else
        t_placa = get(handles.I_Placa,'String');
        t_dni = get(handles.I_DNI,'String');
        t_prop = get(handles.I_Propietario,'String');
        t_tipo = get(handles.I_Tipo,'String');
        t_marca = get(handles.I_Marca,'String');
        t_modelo = get(handles.I_Modelo,'String');
        format short g
        hour = datestr(datetime('now'));
        T(fil,:) = {t_placa, t_dni, t_prop, t_tipo, t_marca, t_modelo, hour(13:20)};
        fil = fil+1;
        x1 = 0;
        msgbox('El usuario ha sido registrado con éxito','Registro');
    end
end

% --- Executes on button press in Exportar.
function Exportar_Callback(hObject, ~, handles)
global T
fecha = char(datetime('today','Format','dd-MM-yyyy'));
filename = strcat('Registro_',fecha,'.xlsx');
T2 = {'Matrícula','DNI','Propietario','Tipo','Marca','Modelo','Hora de Ingreso'; '-----','-----
','-----','-----','-----','-----','-----'};
T3 = [T2(1:end,:);T(1:end,:)];
xlswrite(filename,T3,1,'B2');
winopen(filename);
```

```
%% ENCENDIDO Y APAGADO DE LA TRANSMISION
% --- Executes on button press in ON.
function ON_Callback(hObject, ~, handles)
global img ip zoom
ip = 1;
axes(handles.Camara);
url=get(handles.IP,'String');
img = imread(url);
fh = image(img);
```

```
axis off;
set(handles.IP,'Enable','off')
while(ip == 1)
    img = imread(url); img = imcrop(img,[round(1920*0.5*(1-1/zoom)) round(1080*0.5*(1-1/zoom))
round(1920/zoom) round(1080/zoom)]);
    img = imresize(img,[1080 1920],'bicubic');
    set(fh,'CData',img); drawnow;
end

% --- Executes on button press in OFF.
function OFF_Callback(~, ~, handles)
global ip
ip = 0;
axes(handles.Camara); cla(gca,'reset'); box off;
set(gca,'xcolor',get(gcf,'color')); set(gca,'xtick',[]);
set(gca,'ycolor',get(gcf,'color')); set(gca,'ytick',[]);
set(handles.IP,'Enable','on')
```

```
%% IDENTIFICACION DE CARACTERES EN PLACA VEHICULAR
% --- Executes on button press in Detectar.
function Detectar_Callback(~, ~, handles)
global img ip I caracteres
if ip == 1
    I = img;
    set(handles.Ins,'Visible','off');

    %% Escala de Grises y Escalado
    I=rgb2gray(I);
    I2 = imresize(I,[300 500],'bicubic');       % Nueva imagen de 400x300

    %% Recortado
    I3 = imcrop(I2,[150 150 200 150]);          % Corte estándar para disminuir datos

    %% Filtro Morfológico (Bottom Hat)
    bh = strel('disk',5);                       % Bajar el valor si es necesario
    I4 = imbothat(I3,bh)*3;                     % Filtro Bottom Hat

    %% Binarización
    I5 = im2bw(I4);

    %% Operaciones
    % Cierre
    Oc = strel('disk',5); I6 = imclose(I5,Oc);

    % Apertura
    Oa = strel('disk',5); I6 = imopen(I6,Oa);

    % Dilatación Vertical
    Odv = strel('line',10,90); I6 = imdilate(I6,Odv);

    % Dilatación Horizontal
    Odh = strel('line',15,0); I6 = imdilate(I6,Odh);

    %% Extracción de placa
    L = bwlabel(I6);                % Crea regiones (etiqueta componentes conectados)
    n = max(max(L));                % Cantidad de elementos
    stats = regionprops(I6, 'all');  % Obtiene la región
    % Encontrar figura centrada
    if n == 0
        msgbox('No se puede encontrar una placa.','ANPR');
    else
    cuadro = zeros(n,4);
    for i=1:n
        cuadro(i,:) = stats(i).BoundingBox;
    end
    if n==1
        cuadroF = stats(1).BoundingBox;
    else
        ii = 1;         vm = 0;
        for i=1:n
            if 100>cuadro(i,1) && (cuadro(i,1)+cuadro(i,3))>100 && cuadro(i,3)
```

```
            if cuadro(i,2)>vm
                vm = cuadro(i,2);                ii = i;
            end
            end
        end
        cuadroF = stats(ii).BoundingBox;
    end

    % Eliminar los espacio negros de barras metalicas
    I7 = imcrop(I6,(cuadroF)); T = size(I7); hr = T(2); vr = T(1);
    if (cuadroF(3)>75)||(cuadroF(4)>40)
        condsize=0;
        while(condsize==0)
            xir = 0; yir = 0; c11 = 1; c12 = 1; c21 = 1; c22 = 1; c3 = 1; c4 = 1;
            xfr = hr; yfr = vr;
            xir1 = 0; xir2 = 0;
            for i=(round(hr/2)-2):-1:1
                if I7(round(vr/2)-1,i)==0 && c11 == 1, xir1 = i+1; c11 = 0; end
                if I7(round(vr/2)+1,i)==0 && c12 == 1, xir2 = i+1; c12 = 0; end
                if xir1 > xir2, xir=xir1; else xir=xir2; end
            end
            xfr1 = hr; xfr2 = hr;
            for i=(round(hr/2)+2):1:hr
                if I7(round(vr/2)-1,i)==0 && c21 == 1, xfr1 = i-1; c21 = 0; end
                if I7(round(vr/2)+1,i)==0 && c22 == 1, xfr2 = i-1; c22 = 0; end
                if xfr1 > xfr2, xfr=xfr2; else xfr=xfr1; end
            end
            if xir==0 || xfr == hr
                for i=(round(vr/2)-2):-1:1
                if I7(i,round(hr/2))==0 && c3 == 1
                    yir = i+1; c3 = 0;
                end
                end
                for i=(round(vr/2)+2):1:vr
                if I7(i,round(hr/2))==0 && c4 == 1
                    yfr = i-1; c4 = 0;
                end
                end
            else
                for i=(round(vr/2)-1):-1:1
                if xir > 2; xir1 = xir-2; else xir1 = 1; end
                if I7(i,xir1)==1 && c3 == 1
                    yir = i+1; c3 = 0;
                end
                end
                for i=(round(vr/2)+1):1:vr
                if hr-1 > xfr; xfr1 = xfr+2; else xfr1 = hr; end
                if I7(i,xfr1)==1 && c4 == 1
                    yfr = i-1; c4 = 0;
                end
                end
            end
            cuadroF = [cuadroF(1)+xir cuadroF(2)+yir xfr-xir-1 yfr-yir-1];
            I7 = imcrop(I6,(cuadroF));
            T = size(I7); hr = T(2); vr = T(1);
            if (sum(I7(:,1))+sum(I7(:,hr)))>(2*vr-5) && (sum(I7(1,:))+sum(I7(vr,:)))>(2*hr-5)
                condsize=1;
            else
                cuadroF = cuadroF+[1 1 -2 -2];
                I7 = imcrop(I6,(cuadroF));
                T = size(I7); hr = T(2); vr = T(1);
            end
        end
    end

    I7 = imcrop(I3,(cuadroF));    I7 = imresize(I7,[100 300],'bicubic');
    set(handles.Imagen_Placa,'Visible','on');    axes(handles.Imagen_Placa), imshow(I7);

    %% Binarización de la Placa
    bh1 = strel('disk',100);
    I8 = imbothat(I7,bh1)*2.7;
```

```
I8 = im2bw(I8);
I8 = bwareaopen(I8,40);

%% Detección de Caracteres
% Recorta bordes horizontales
I8v = zeros(100,300);
for i=1:300;     I8v(1,i) = sum(I8(:,i));  end,    %I8v;
cond1 = 0; i = 1;
while(cond1==0)
    if 90>I8v(1,i)
        h1 = i;            cond1 = 1;
    end
    i=i+1;
end
cond1 = 0; i = 300;
while(cond1==0)
    if 90>I8v(1,i)
        h2 = i;            cond1 = 1;
    end
    i=i-1;
end
I8 = imcrop(I8,[h1+1 1 h2-h1-2 100]);
I8 = imresize(I8,[100 300],'bicubic');

% Recorta bordes verticales
I8h = zeros(100,300);
for i=1:100;     I8h(i,1) = sum(I8(i,:));  end,    %I8h;
cond1 = 0; i = 1; c1 = 0;
while(cond1==0)
    if c1==I8h(i,1)
        ii = i;
        cond1 = 1;
    end
    i=i+1;
    if i==50
        i=1;
        c1=c1+1;
    end
end

I8 = imcrop(I8,[1 ii 300 100-ii+1]);
I8 = imresize(I8,[100 300],'bicubic');

for i=1:100;     I8h(i,1) = sum(I8(i,:));  end,    %I8h;
cond1 = 0; ii = 1;
while(cond1==0)
    if I8h(ii,1)>100
        v1 = ii;
        cond1 = 1;
    end
    if I8h(ii+1,1)==300
        cond1 = 0;
    end
    ii=ii+1;
end
cond1 = 0; i = 100;
while(cond1==0)
    if 250>I8h(i,1)
        v2 = i;
        cond1 = 1;
    end
    i=i-1;
end
I8 = imcrop(I8,[1 v1+1 300 v2-v1-2]);
I8 = imresize(I8,[100 300],'bicubic');

%% Recorte de la Placa (Eliminar letras de nacionalidad)
I9 = bwareaopen(I8,1000);
L2 = bwlabel(I9);                    % Crea regiones (etiqueta componentes conectados)
n2 = max(max(L2));                     % Cantidad de elementos
stats2 = regionprops(I9, 'all');    % Obtiene las Estadísticas de las regiones
```

```
n3 = 1;
ne = 0;
while (n3~=6)
    for i = 1:n2
        E = stats2(i).BoundingBox;
        if E(3)>=30 && E(4)>=40
            if E(3)>=50
                ni = round(E(3)/45);
                for i=1:(ni-1)
                    Error = round(E(1))+round(E(3)/ni)*i;
                    I9(:,(Error-1:1:Error))=0;
                end
            end
        end
    end
    I9 = bwareaopen(I9,800);
    L2 = bwlabel(I9);                   % Crea regiones (etiqueta componentes conectados)
    n2 = max(max(L2));                    % Cantidad de elementos
    stats2 = regionprops(I9, 'all');    % Obtiene las Estadísticas de las regiones
    n3 = 1;
    stats3 = zeros(6,4);
    for i = 1:n2
        E = stats2(i).BoundingBox;
        if E(3)>=30 && E(4)>=40
            stats3(n3,:)=E;
            n3=n3+1;
        end
    end
    n3=n3-1;
    ne=ne+1;
    if ne == 6; n3=6; end
end

if ne == 6
    msgbox('No se puede encontrar una placa.','ANPR');
else

%% Ubicación del caracter
letras = zeros(6);    letra = zeros(6,36);
for fila=1:n3
    E = stats3(fila,:);
    if E(4)>50
        C = imcrop(I9,E);            Caracter = imresize(C,[42 24]);
            if (sum(Caracter(1,:))==0) || (sum(Caracter(1,:))==24)
                C = imcrop(Caracter,[1 4 24 39]); Caracter = imresize(C,[42 24]);
            end
            if (sum(Caracter(42,:))==0) || (sum(Caracter(42,:))==24)
                C = imcrop(Caracter,[1 1 24 39]); Caracter = imresize(C,[42 24]);
            end
            if (sum(Caracter(:,1))==0) || (sum(Caracter(:,1))==42)
                C = imcrop(Caracter,[4 1 21 42]); Caracter = imresize(C,[42 24]);
            end
            if (sum(Caracter(:,24))==0) || (sum(Caracter(:,24))==42)
                C = imcrop(Caracter,[1 1 21 42]); Caracter = imresize(C,[42 24]);
            end
        if fila == 1, C1 = Caracter; end
        if fila == 2, C2 = Caracter; end
        if fila == 3, C3 = Caracter; end
        if fila == 4, C4 = Caracter; end
        if fila == 5, C5 = Caracter; end
        if fila == 6, C6 = Caracter; end

        % Aplicamos la Red Neuronal basada en 288 muestras
        [pos,Xf,Af] = RNA(Caracter(:));
        [~, posicion] = ismembertol(1,pos,1E-3);
        letras(fila) = posicion;
    end
end

%% Cambio de posición por caracter
```

```
    car = ['','','','','',''];
    for lt = 1:n3 % son 3 letras
        if letras(lt)== 1,          car(lt)='A'; elseif letras(lt) == 2,     car(lt)='B';
        elseif letras(lt) == 3,     car(lt)='C'; elseif letras(lt) == 4,     car(lt)='D';
        elseif letras(lt) == 5,     car(lt)='E'; elseif letras(lt) == 6,     car(lt)='F';
        elseif letras(lt) == 7,     car(lt)='G'; elseif letras(lt) == 8,     car(lt)='H';
        elseif letras(lt) == 9,     car(lt)='I'; elseif letras(lt) == 10,    car(lt)='J';
        elseif letras(lt) == 11,    car(lt)='K'; elseif letras(lt) == 12,    car(lt)='L';
        elseif letras(lt) == 13,    car(lt)='M'; elseif letras(lt) == 14,    car(lt)='N';
        elseif letras(lt) == 15,    car(lt)='O'; elseif letras(lt) == 16,    car(lt)='P';
        elseif letras(lt) == 17,    car(lt)='Q'; elseif letras(lt) == 18,    car(lt)='R';
        elseif letras(lt) == 19,    car(lt)='S'; elseif letras(lt) == 20,    car(lt)='T';
        elseif letras(lt) == 21,    car(lt)='U'; elseif letras(lt) == 22,    car(lt)='V';
        elseif letras(lt) == 23,    car(lt)='W'; elseif letras(lt) == 24,    car(lt)='X';
        elseif letras(lt) == 25,    car(lt)='Y'; elseif letras(lt) == 26,    car(lt)='Z';
        elseif letras(lt) == 27,    car(lt)='1'; elseif letras(lt) == 28,    car(lt)='2';
        elseif letras(lt) == 29,    car(lt)='3'; elseif letras(lt) == 30,    car(lt)='4';
        elseif letras(lt) == 31,    car(lt)='5'; elseif letras(lt) == 32,    car(lt)='6';
        elseif letras(lt) == 33,    car(lt)='7'; elseif letras(lt) == 34,    car(lt)='8';
        elseif letras(lt) == 35,    car(lt)='9'; elseif letras(lt) == 36,    car(lt)='0';
        else
        car(lt)='-';
        end
    end
    set(handles.I_Placa,'String',car);
    end
    end
else
    msgbox('Encender la transmisión primero','ANPR');
end

OPT = get(handles.Opt2,'Value')
if OPT == 0
global x2 x1
% x1 : Personal encontrado
% x2 : Personal registrado
set(handles.Usuario,'Value',0);

[dni, base, ~] = xlsread('Base de datos.xlsm','REGISTRO');
base([1 2],:)=[]; base(:,1)=[];
h = size(base); h = h(1,1);
car = get(handles.I_Placa,'String');car = char(car);
x1 = 0;t = char(base(2,1));
if size(car) == size(t)
for i=1:1:h
    n = char(base(i,1));
    if n==car
        inf1 = dni(i,1);
        set(handles.I_DNI,'Enable','off'); set(handles.I_DNI,'String',inf1);
        inf2 = char(base(i,2));
        set(handles.I_Propietario,'Enable','off'); set(handles.I_Propietario,'String',inf2);
        inf3 = char(base(i,3));
        set(handles.I_Tipo,'Enable','off'); set(handles.I_Tipo,'String',inf3);
        inf4 = char(base(i,4));
        set(handles.I_Marca,'Enable','off'); set(handles.I_Marca,'String',inf4);
        inf5 = char(base(i,5));
        set(handles.I_Modelo,'Enable','off');set(handles.I_Modelo,'String',inf5);
        x1 = 1;
    end
end
end
if x1==1;
    msgbox('Personal encontrado en la base de datos.','Registro');
    x2 = 1;
else
    msgbox('No se encuentra en la base de datos.','Registro');
    set(handles.I_DNI,'String','');
    set(handles.I_Propietario,'String','');
    set(handles.I_Tipo,'String','');
    set(handles.I_Marca,'String','');
    set(handles.I_Modelo,'String','');
```

```
    x2 = 0;
end
end
```

```
%% INFORMACIÓN DE USUARIO Y EL REGISTRO VIGENTE
% --- Executes on button press in Registro.
function Registro_Callback(~, ~, handles)
TablaRegistro

% --- Executes on button press in Usuario.
function Usuario_Callback(hObject, ~, handles)
global x1 x2
if get(hObject,'Value') == 1
    set(handles.I_DNI,'Enable','on');
    set(handles.I_Propietario,'Enable','on');
    set(handles.I_Tipo,'Enable','on');
    set(handles.I_Marca,'Enable','on');
    set(handles.I_Modelo,'Enable','on');
    x1 = 1;
    x2 = 1;
else
    set(handles.I_DNI,'Enable','off');
    set(handles.I_Propietario,'Enable','off');
    set(handles.I_Tipo,'Enable','off');
    set(handles.I_Marca,'Enable','off');
    set(handles.I_Modelo,'Enable','off');
end
```

Anexo 8: Hoja de Datos de Cámara IP

Cámara de red AXIS M1145-L

Cámara	
Sensor de imagen	CMOS RGB de barrido progresivo de 1/2,9"
Lente	Varifocal, iris tipo P, 3-10,5 mm, F1.4 Campo de visión horizontal: 95°-34° Campo de visión vertical: 47°-19° Filtro de infrarrojos incorporado y enfoque/zoom motorizados
De día y de noche	Filtro de infrarrojos extraíble automáticamente
Iluminación mínima	**Color:** 0,4 lux, F1.4 **B/N:** 0,08 lux, F1.4, 0 lux con la iluminación con infrarrojos activada
Velocidad de obturación	De 1/28000 s a 2 s con 50 Hz De 1/33500 s a 2 s con 60 Hz
Movimiento horizontal/vertical y zoom	PTZ digital
Vídeo	
Compresión de vídeo	H.264 (MPEG-4 Parte 10/AVC), perfiles Base y Main Motion JPEG
Resoluciones	De 1920x1080 a 160x90
Velocidad de imagen	25/30 imágenes por segundo en todas las resoluciones con una frecuencia de línea de alimentación de 50/60 Hz
Retransmisión de vídeo	Múltiples secuencias de vídeo configurables individualmente en H.264 y Motion JPEG Frecuencia de imagen y ancho de banda controlables VBR/MBR H.264
Parámetros de la imagen	Compresión, color, brillo, nitidez, contraste, balance de blancos, compensación de contraluz, duplicación de imágenes, rotación (incluido formato pasillo), superposición de texto e imágenes, máscara de privacidad, rotación: 0°, 90°, 180°, 270° amplio rango dinámico (WDR) con contraste dinámico, control y zonas de exposición, configuración más precisa del comportamiento con poca luz
Red	
Seguridad	Protección por contraseña, filtrado de direcciones IP, cifrado HTTPS[a], control de acceso a la red IEEE 802.1X[a] autenticación Digest, registro de acceso de usuarios, gestión centralizada de certificados
Protocolos compatibles	IPv4/v6, HTTP, HTTPS, SSL/TLS, QoS Layer 3 DiffServ, FTP, CIFS/SMB, SMTP, Bonjour, UPnP™, SNMPv1/v2c/v3 (MIB-II), DNS, DynDNS, NTP, RTSP, RTP, TCP, UDP, IGMP, RTCP, ICMP, DHCP, ARP, SOCKS, SSH
Integración de sistemas	
Interfaz de programación de aplicaciones	API abierta para la integración de software, incluida VAPIX® y la plataforma de aplicaciones de cámaras AXIS, especificaciones disponibles en *www.axis.com* AXIS Video Hosting System (AVHS) con conexión con un solo clic ONVIF® Profile S y ONVIF® Profile G, las especificaciones están disponibles en *www.onvif.org*
Analíticas	**Incluido** AXIS Video Motion Detection, alarma antimanipulación activa **Compatible** AXIS Digital Autotracking, AXIS Perimeter Defender, AXIS Cross Line Detection Compatibilidad con la plataforma de aplicaciones de cámaras AXIS que permite la instalación de aplicaciones de terceros, consulte *www.axis.com/acap*
Activadores de evento	Análisis, eventos de almacenamiento local, entrada externa, programación de hora
Acciones de evento	Carga de archivos a través de FTP, HTTP y recurso compartido de red y correo electrónico Notificación por correo electrónico, HTTP y TCP, memoria de vídeo previa y posterior a la alarma, activación de salida externa, grabación de vídeo en el almacenamiento local, ir a una posición PTZ predefinida, ronda de vigilancia Activación de iluminación de infrarrojos
Retransmisión de datos	Datos de eventos
General	
Carcasa	color: blanco NCS S 1002-B aluminio y PC/siloxano, 42 % de plástico reciclado
Sostenibilidad	Sin PVC
Memoria	256 MB de RAM, 128 MB de Flash
Alimentación	Alimentación a través de Ethernet (PoE) IEEE 802.3af/802.3at Tipo 1 Clase 3 Máx. 10,5 W
Conectores	RJ45 10BASE-T/100BASE-TX PoE. 1 entrada y 1 salida de alarma
Iluminación con infrarrojos	OptimizedIR, LED de consumo energético muy bajo y con un ángulo de iluminación e intensidad regulables. Alcance de hasta 15 m.
Almacenamiento	Compatibilidad con tarjetas microSD/microSDHC/microSDXC Compatible con cifrado de tarjeta SD Compatible con grabación en almacenamiento conectado a la red (NAS) Para conocer las recomendaciones de tarjetas SD y NAS, consulte *www.axis.com*
Condiciones de funcionamiento	de 0 °C a 50 °C Humedad relativa del 20 al 80 % (sin condensación)
Homologaciones	**EMC** EN 55022 Clase B, EN 61000-6-1, EN 61000-6-2, EN 61000-3-2,EN 61000-3-3, EN 55024, FCC Parte 15 Subparte B Clase B, ICES-003 Clase B, VCCI Clase B, C-tick AS/NZS CISPR 22 Clase B, KCC KN22 Clase B, KN24 **Seguridad** IEC/EN/UL 60950-1 **Medio ambiente** EN 50581 **Otros** IEC/EN 62471
Dimensiones	Altura: 46 mm Longitud: 114 mm Anchura: 75 mm
Peso	260 g
Accesorios incluidos	Soporte, guía de instalación, descodificador de Windows (1 licencia de usuario)
Accesorios opcionales	Montajes Axis, carcasas y armarios Axis, midspans Axis, iluminadores Axis Para conocer más accesorios, consulte *www.axis.com*
Software de gestión de vídeo	AXIS Companion, AXIS Camera Station, software de gestión de vídeo de socios desarrolladores de aplicaciones de Axis disponible en *www.axis.com/techsup/software*
Idiomas	Inglés, alemán, francés, español, italiano, ruso, chino simplificado, japonés, coreano, portugués, chino tradicional
Garantía	Garantía Axis de 3 años y opción de garantía AXIS ampliada, visite *www.axis.com/warranty*

a. *Este producto incluye software desarrollado por OpenSSL Project para su uso en el kit de herramientas OpenSSL. (www.openssl.org) y software criptográfico escrito por Eric Young (eay@cryptsoft.com).*

Responsabilidad medioambiental:

www.axis.com/environmental-responsibility