ESTUDIO Y DESARROLLO DE SOLUCIÓN ANPR PARA UN ACCESO VEHICULAR

GRADO EN INGENIERÍA EN SISTEMAS INDUSTRIALES

UNIVERSIDAD EUROPEA DE MADRID

Escuela de Arquitectura, Ingeniería y Diseño

ESTUDIO Y DESARROLLO DE SOLUCIÓN ANPR PARA ACCESO VEHICULAR

Autor:

Pablo Javier de Paz García.

Director del proyecto:

Diego Ortega Sanz.

Junio 2024

TÍTULO: ESTUDIO Y DESARROLLO DE SOLUCIÓN ANPR PARA ACCESO VEHICULAR.

AUTOR: PABLO JAVIER DE PAZ GARCÍA.

DIRECTOR DEL PROYECTO: DIEGO ORTEGA SANZ.

FECHA: 5 DE JUNIO DE 2024.

RESUMEN

Este proyecto abarca la detallada explicación del procedimiento y las decisiones de ingeniería tomadas a lo largo del desarrollo de una solución de reconocimiento automático de matrículas (ANPR).

Se ha realizado un estudio comparativo entre los métodos de visión artificial más empleados en la detección y lectura de matrículas, con el fin de determinar el método más óptimo en la actualidad y desarrollar un algoritmo ANPR eficiente, rápido y preciso.

Se ha desarrollado una aplicación ANPR que incluye interfaz gráfica de usuario para monitorizar y controlar los accesos y salidas de un recinto industrial privado. Además, se ha desarrollado una propuesta de embarcado en hardware.

Por último, se ha definido un posible producto económicamente competitivo con las alternativas disponibles en el mercado.

Palabras clave: Visión artificial, Lector de matrículas, ANPR, OCR, Aprendizaje automático.

ABSTRACT

This project includes a detailed explanation of the procedures followed and the engineering challenges overcome throughout the development of an automatic number plate recognition solution (ANPR).

A comparative study of the most commonly used computer vision methods for license plate reading has been conducted, with the final goal of determining the most optimal method available today, in order to develop an efficient, fast and precise ANPR algorithm.

An ANPR application that includes a human-machine interface has also been developed. The purpose of this application is to monitor and control the entrances and exits of vehicles to a private industrial site. A proposal for a hardware-embedded solution has also been made.

Lastly, a proposal for an economically competitive product, in comparison to the alternatives available in the market, has been developed.

Key words: Computer vision, License plate reader, ANPR, OCR, Machine Learning.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

1.1. Objeto y alcance del proyecto.

En este proyecto se busca desarrollar una solución de reconocimiento automático de matrículas diseñada para ser empleada en una barrera automática fija. Esta solución ANPR (del inglés “Automatic Number Plate Recognition”) debe obtener la información de una matrícula y permitir o denegar el acceso o salida de vehículos a un entorno industrial privado como una fábrica o una empresa.

El primer objetivo de este proyecto consiste en desarrollar un algoritmo de reconocimiento de matrículas. El resultado de aplicar este algoritmo sobre una imagen será una cadena de texto con la información de la matrícula, la cual puede ser empleada en procesos externos al sistema de acceso para optimizar los procesos de un complejo industrial.

Para desarrollar el algoritmo se va a emplear Python 3.9. Para determinar los algoritmos óptimos para la aplicación, se va a realizar un estudio comparativo sobre la eficacia y el rendimiento de los distintos métodos de detección y de lectura de matrículas, haciendo especial énfasis en el análisis de los sistemas de visión basados en redes neuronales en comparación con los métodos tradicionales.

Por tanto se busca comparar la eficiencia de los métodos tradicionales, de los métodos de reconocimiento de patrones y de los métodos basados en inteligencia artificial para las dos etapas del algoritmo.

Una vez desarrollado el algoritmo en cascada final se pretende comprobar sus características mediante el análisis de ejemplos de funcionamiento realistas.

El siguiente objetivo de este proyecto será plantear una aplicación ANPR diseñada para una barrera vehicular doble, que sirve como entrada y salida de vehículos para un recinto industrial con un único acceso. Esta aplicación debe permitir el acceso inmediatamente a vehículos previamente registrados. Además, debe ofrecer a un operario la decisión de permitir o denegar el acceso al recinto. Todos los accesos y salidas deben ser registrados para su posterior uso o análisis.

Por último, se va a estudiar una posible solución de embarcado en hardware para este escenario, seguido del desarrollo de unos posibles presupuestos de fabricación, compra e instalación del producto final.

El proyecto no abarca la construcción de una maqueta funcional, por lo que todo el desarrollo se va a realizar empleando un hardware personal e imágenes obtenidas con antelación.

1.2.Objetivos específicos.

En este apartado se van a definir a modo de lista los objetivos específicos del proyecto, con el fin de plantear los límites en el desarrollo del mismo.

Además, este apartado sirve para poder llevar un control del progreso y de los pasos a seguir durante el desarrollo. Más adelante, en el capítulo de conclusiones, se va a evaluar el cumplimiento de estos objetivos.

Los objetivos específicos de este proyecto son los siguientes:

- Desarrollar un algoritmo de visión artificial propio en Python 3 para la detección y lectura de matrículas con las siguientes características:
    - El algoritmo debe ser capaz de detectar y leer, como mínimo, las matrículas de camiones, furgonetas y turismos.
    - El algoritmo debe ser capaz de detectar y leer, como mínimo, las matrículas delanteras en vehículos europeos, canadienses, estadounidenses, australianos y neozelandeses.
    - El algoritmo debe ser capaz de funcionar en entornos de baja visibilidad como lluvia o durante la noche, así como en entornos desfavorables de detección como matrículas sucias o daño de las lentes.
    - La distancia de lectura será constante. Se indicará a los vehículos donde se deben detener mediante señalizaciones físicas para mantener una distancia homogénea hasta la matrícula.
    - El algoritmo debe ser diseñado para que el lector de matrículas se encuentre a una altura de entre 0.2 metros y 1.5 metros.
    - El algoritmo debe ser veloz. El objetivo es reconocer una matrícula en 1 segundo o menos.
    - El algoritmo debe ser fiable. El objetivo es tener un 80% de precisión o más.
    - El algoritmo ofrece como resultado una cadena de texto con la información de la matrícula detectada.

- Para desarrollar el algoritmo de detección de matrículas se va a realizar un estudio comparativo de los métodos más comunes. Por tanto, se van a desarrollar y comparar algoritmos basados en los siguientes métodos:
    - Métodos tradicionales.
    - Métodos basados en reconocimiento de patrones.
    - Métodos basados en aprendizaje automático.

- Para desarrollar el algoritmo de lectura de caracteres se va a realizar un estudio comparativo de los métodos más comunes. Por tanto, se van a desarrollar y comparar algoritmos basados en los siguientes métodos:
    - Métodos basados en reconocimiento de patrones.
    - Métodos basados en aprendizaje automático.
    - Métodos basados en soluciones OCR de código abierto, como Tesseract o EasyOCR.

- Determinar el algoritmo en cascada final óptimo según los resultados del estudio comparativo de los algoritmos de detección y lectura de caracteres.

- Desarrollar una aplicación ANPR que registre la fecha y hora de entrada y salida de vehículos a un determinado recinto con carácter industrial.
    - Diseñar una interfaz gráfica interactiva para la aplicación ANPR.
    - Implementar el algoritmo ANPR en conjunto con la interfaz gráfica.

- Plantear una solución de hardware y plantear el proyecto como un producto comercializable.
    - Plantear hardware local sobre el que se va a realizar el procesamiento de imágenes.
    - Plantear un presupuesto de compra de los elementos necesarios para el desarrollo del sistema ANPR.
    - Plantear un precio de venta de los productos y de un servicio de instalación asociado al sistema ANPR.

1.3. Justificación y motivación.

Una de las aplicaciones de la visión artificial más comunes e integradas en la vida cotidiana son sin duda los sistemas de reconocimiento de matrículas.

Entre ellos, destacan en número los encontrados en sistemas de control de velocidad de carreteras y autovías, en los sistemas de control y supervisión de aparcamiento o tráfico y en los accesos a recintos privados como aparcamientos o zonas residenciales privadas. En este proyecto nos centraremos en este último grupo de sistemas ANPR instalados en barreras automáticas.

Aunque la inteligencia artificial aplicada en la visión artificial lleva existiendo desde la década de los 90, en los últimos años se ha comenzado a popularizar el uso de las inteligencias artificiales en un gran número de campos. Esto, debido al gran aumento en el poder computacional de los hardwares modernos, ha llevado a un gran desarrollo en la potencia, eficiencia y precisión de las inteligencias artificiales.

Por este motivo, en este proyecto se busca comparar la eficiencia de los nuevos sistemas ANPR (Automatic Number Plate Recognition) basados en una rama de la inteligencia artificial llamada aprendizaje automático en comparación con los métodos tradicionales, así como aplicar los conocimientos de automatización adquiridos a lo largo de mis carrera universitaria.

Además, los precios de los sistemas ANPR disponibles en el mercado actual son muy elevados, por lo que se busca tratar de crear una solución de bajo coste.

Por otro lado, mi motivación personal es diferente. Desde que comencé a interesarme por el mundo de la robótica, sin duda el campo que más me ha fascinado es el de la visión artificial. La infinidad de posibilidades que presentan los sistemas basados en visión artificial, junto con la compatibilidad de estos sistemas en el mundo de la automatización industrial me han atraído a este campo.

El otro motivo personal por el que he decidido desarrollar un lector de matrículas es diferente. Para acceder en un vehículo al recinto residencial donde vivo, es necesario pasar por una barrera automática con lector de matrículas. Este lector es anticuado y por norma general no suele funcionar a la primera.

Por lo tanto y dado que este problema se solapa con los conocimientos adquiridos durante mis años de estudios universitarios, he decidido tratar de desarrollar un algoritmo ANPR desde cero para comprobar si realmente hoy en día están justificados los problemas que presenta el sistema mencionado.

CAPÍTULO 2. MARCO TEÓRICO.

En este capítulo se van a explicar en detalle los conceptos y conocimientos mínimos y esenciales necesarios para comprender el proyecto en su totalidad.

Por otro lado, se va a realizar un breve análisis de otros proyectos similares.

Además, se van a explicar brevemente las normativas regionales aplicables a este proyecto.

2.1. Sistemas ANPR.

2.1.1. ¿Qué es un sistema ANPR?

Los sistemas de reconocimiento automático de matrículas son también conocidos como NPR, LPR, ANPR o ALPR (Automatic Number / License Plate Recognition). Otros nombres por los que se puede nombrar a este tipo de sistemas son AVI (Automatic Vehicle Identification) o CPR (Car Plate Recognition).

Aunque pueden llegar a existir diferencias de uso o principio de funcionamiento entre estos nombres mencionados, en esencia significan lo mismo, un algoritmo, aparato o sistema capaz de detectar y leer matrículas de vehículos.

Este tipo de sistemas emplean técnicas de visión artificial para detectar y localizar una matrícula en una determinada imagen, sin necesidad de intervención humana. Una vez encontrada la matrícula se emplean sistemas de reconocimiento de caracteres llamados OCR (Optical Character Recognition).

Para un humano es sencillo identificar una matrícula y reconocer los caracteres de la misma. En cambio, lo que para un humano es una tarea sencilla se convierte en un reto para un algoritmo. Enfrentar este reto y diseñar un sistema ANPR ha abierto un sinfín de posibilidades a la hora de automatizar tareas tediosas, difíciles e incluso potencialmente peligrosas. [1].

Las aplicaciones principales de este tipo de sistemas son:

- Control del estacionamiento indebido.
- Control de las infracciones de velocidad o de tránsito.
- Control de pagos en peajes.
- Control y encuestas de origen-destino en viajes de larga duración.
- Control del acceso a recintos privados o controlados.
- Control en fronteras.
- Control de vehículos robados.

Por norma general estos sistemas se componen de tres partes físicas diferenciadas:

- La unidad de captura de imagen o vídeo.
- La unidad de iluminación.
- La unidad de procesamiento.

Las características de estas unidades dependen completamente de su aplicación. A continuación se va a entrar en detalle en cada una de las partes que componen un sistema completo de reconocimiento de matrículas.

Unidad de captura de imágen.

La unidad de captura de imagen o video es posiblemente el componente que más puede variar entre las distintas aplicaciones y versiones de los sistemas ANPR.

Las características de las cámaras empleadas son lo primero que se debe determinar en un sistema de lectura de matrículas, ya que todo el algoritmo de procesamiento se va a adaptar alrededor de dichas características, que se encuentran determinadas por la aplicación escogida para el sistema ANPR.

Los puntos más relevantes a la hora de elegir las características de la cámara que se va a emplear son los siguientes:

- La calidad de la imagen necesaria.
- Tipo de luz a captar (infrarroja, natural).
- La velocidad de captura de imágenes (en fotogramas por segundo).
- Si la aplicación necesita imagen en color o puede ser monocroma.
- Los métodos de conexión y transmisión de datos empleados.
- El tipo de sensor empleado en la captura de imagen (sensor CMOS).
- Global shutter o Rolling shutter (en función de si el vehículo se encuentra detenido o en movimiento).

Por ejemplo, una cámara empleada en un radar de tráfico debe tener unos fotogramas por segundo máximos muy bajos, ya que únicamente se necesita tomar una sola imagen de la parte delantera o trasera del vehículo detectado. La calidad de la imagen debe ser buena para identificar los detalles a una distancia de 2-4 metros, el tipo de captura de imágen debe de ser global shutter para obtener una imagen sin artefactos generados por un rolling shutter, etc.

En este tipo de aplicaciones donde no es realmente necesario disponer de una imagen a color, se pueden emplear cámaras monocromas, que al disponer de un único canal de color (escala de grises) producen imágenes mucho más rápidas de procesar. Cabe destacar que en el caso de utilizar una inteligencia artificial para el procesamiento de la imagen puede ser necesario la imagen a color completa.

En caso de emplear iluminación infrarroja la cámara debe ser también infrarroja y por tanto la imagen producida será en escala de grises.

Además, en caso de estar situada en exteriores o en entornos húmedos, es recomendable que la cámara empleada tenga un grado de protección IP relativamente elevado. Algunos fabricantes recomiendan emplear no menos de un IP-55.

Generalmente, las cámaras clásicas de vigilancia CCTV (Closed Circuit Television) que emplean encapsulados tipo bullet o tipo domo ya han sido diseñadas para exteriores y para cumplir este tipo de necesidades, y como se verá más adelante.

Unidad de iluminación.

La unidad de iluminación proporciona la luz necesaria para que la cámara sea capaz de obtener imágenes consistentes y claras donde toda la información sea visible. Generalmente la fuente de iluminación está incorporada en el hardware de la cámara, aunque es posible utilizar una fuente de luz externa.

De nuevo, el tipo de iluminación requerida varía en función de la aplicación escogida. Por norma general, en exteriores se emplea iluminación infrarroja para obtener imágenes consistentes independientemente de la iluminación ambiente o de la contaminación lumínica provocada por los mismos vehículos o por elementos reflectantes externos.

En entornos más controlados, como sistemas en instalaciones subterráneas o en interiores, se pueden emplear técnicas de iluminación tradicionales como iluminación directa o iluminación difusa.

Unidad de procesamiento.

Por último tenemos la unidad de procesamiento de los sistemas ANPR. Esta unidad procesa la información obtenida mediante la cámara empleada en la unidad de captura de imagen. Por tanto, es imprescindible conocer las características de la cámara que se va a emplear antes de comenzar a desarrollar un algoritmo de visión.

Por norma general la unidad de procesamiento se encuentra integrada en el hardware de la propia cámara, aunque existen algunos tipos de instalaciones con un gran número de cámaras que emplean un procesador externo.

La imagen de las cámaras es transmitida hasta este procesador, el cual puede encontrarse a grandes distancias de la ubicación de la cámara.

Ya que las condiciones meteorológicas, atmosféricas y físicas del entorno pueden cambiar constantemente, este algoritmo de visión artificial suele estar basado en métodos relacionados con inteligencia artificial, ya que estos presentan una gran adaptabilidad a los cambios en el entorno.

Sin embargo, este tipo de soluciones suelen ser lentas y requieren un procesador mucho más potente de lo que requeriría una solución basada en métodos tradicionales, por lo que el precio de estos sistemas se ha disparado en los últimos años.

Como ya se ha mencionado con anterioridad en los apartados 1.1 y 1.2, en este proyecto se va a trabajar en el desarrollo del algoritmo empleado en la unidad de procesamiento.

En este caso se va a desarrollar un algoritmo considerando unas características de cámara previamente definidas, como se verá más adelante en el apartado 3.1.3.

2.1.2. Ejemplos de sistemas existentes.

Existe una gran cantidad de fabricantes de sistemas ANPR en el mundo. Estos fabricantes venden cámaras de reconocimiento de matrículas con un procesador incorporado, de manera que el producto únicamente necesita ser brevemente configurado para ofrecer como resultado la información de las matrículas de la forma que el cliente desee.

De la misma manera, en España existe una gran cantidad de proveedores. Se van a mencionar brevemente los proveedores más conocidos e influyentes, así como de las soluciones de hardware y software que han decidido incorporar.

Uniview.

Este fabricante chino de cámaras de vigilancia se ha abierto paso en el mercado europeo gracias a sus reducidos precios y a su gran calidad de producto. Hasta el momento tienen sedes en 17 países y continúan creciendo y expandiendo su influencia por el mundo.

Se introdujo al mercado de cámaras ANPR hace cinco años, y desde entonces ha ofrecido una solución barata de cámaras de reconocimiento de matrículas. Esta apuesta con el mercado de cámaras ANPR les ha llevado a quintuplicar los beneficios en 5 años, por lo que se les considera un referente en este campo.

Su solución para ANPR consiste en una cámara con procesador y fuente de luz incorporados. Las cámaras producidas son ligeras, resistentes y duraderas. Además del reconocimiento de matrículas, estas cámaras permiten ver y grabar la imagen de la cámara, de manera que también pueden funcionar como cámaras de vigilancia.

Figura 2.1: *2MP ANPR Bullet Camera, de Uniview.*

El 2MP ANPR Bullet Camera de Uniview es el modelo ANPR más vendido en todo el mundo. Este sistema de visión tiene una cámara tradicional con un sensor CMOS y 4 LEDs de alta intensidad para iluminar en exteriores hasta 50 metros.

Las imágenes que esta cámara es capaz de obtener tienen una resolución de 1080p y son en color. Estas cámaras se encuentran continuamente procesando el video hasta que detectan una matrícula en la imagen.

Por defecto, estas cámaras tienen una velocidad de procesamiento de 25 fotogramas por segundo, algo verdaderamente impresionante en el mercado, especialmente para un sistema ANPR a escala global.

Este modelo destaca por ser capaz de detectar matrículas de todo el mundo. Gracias a ser una empresa propiedad del gobierno de China tienen acceso a una gran base de datos de matrículas de este país, por lo que han sido capaces de adueñarse por completo del mercado en Asia.

ESSistemas.

Otro fabricante muy importante en España es la empresa española ESSistemas. Este fabricante de cámaras de vigilancia ofrece una solución muy similar a la vista anteriormente por Uniview. [2].

Las cámaras ANPR de este fabricante tienen un encapsulado de tipo bullet y se especializan en radares de velocidad, por lo que la característica fundamental que las diferencia de las vistas anteriormente es la velocidad de captura de imagen. Ya que capturan imágenes en movimiento, estas cámaras emplean captura de imagen de tipo global shutter en sus sensores CMOS.

Además, dado que deben captar imágenes con una gran precisión en un momento indicado, no es tan necesario obtener un flujo constante de vídeo como lo es obtener la imagen de gran calidad en un momento determinado.

Figura 2.2: *Cámara Pro Bullet LPR X12 Motorizada 120fps H.265+ 2Mpx, de ESSistemas.*

Dado que los vehículos de los que se debe obtener la información se encuentran a un gran distancia de la cámara (alrededor de 5 metros), la imagen resultante tiene una calidad de 2 Mp, aunque únicamente puede tomar 1 imagen por segundo.

El algoritmo de reconocimiento de matrículas que tiene integrado analiza imágenes de una enorme calidad, por lo que es comprensible que tenga tanto tiempo de retardo entre capturas de imagen.

2.2. Fundamentos de visión artificial.

Para un humano, ver un objeto y reconocerlo es una acción casi inmediata y que realizamos involuntariamente. En cambio, para un ordenador el simple hecho de distinguir un objeto en una imagen se vuelve una tarea muy compleja.

Para un ordenador, una imagen es simplemente un conjunto de píxeles sin relación. La visión artificial dota a cualquier ordenador con la capacidad de identificar estas relaciones entre píxeles de manera precisa y eficiente.

La visión artificial es una tecnología empleada para reconocer de forma automática las imágenes y describir su contenido de manera precisa y eficiente. [3]

2.2.1. ¿Qué es una imagen?

Un píxel es la unidad más pequeña en la que se divide una imagen, y está descrito por sus características en color y brillo. Una imagen es una matriz ordenada de píxeles. En cambio, un vídeo es una representación sucesiva de imágenes en un intervalo regular conocido como FPS (fotogramas por segundo).

Las imágenes pueden ser tanto a color como en escala de grises. El color de una imagen se describe mediante tres canales únicos, mientras que una imagen en escala de grises únicamente necesita un canal por pixel.

Estos canales pueden tomar valores desde 0 hasta 255. En el caso de las imágenes a color, ya que existen tres canales de color diferentes, se pueden crear hasta 17 millones de colores diferentes por cada píxel.

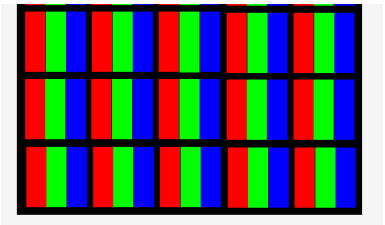


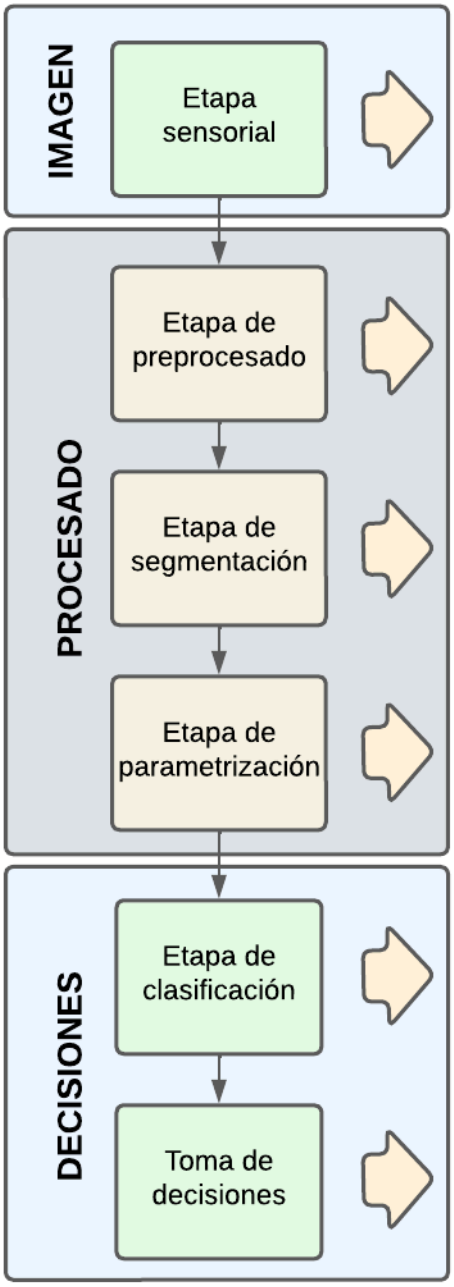
Figura 2.3: *Píxeles de una imagen en RGB. de IONOS.*

Existen varios formatos diferentes para la representación del color. El más conocido y utilizado es sin duda el RGB (Red Green Blue), pero existen otros como por ejemplo el CYGM (Cyan Yellow Green Magenta), el HSV (Hue Saturation Value) o el HSL (Hue Saturation Lightness).

Todas estas formas de representar color tienen sus utilidades en distintos campos, pero en visión artificial se emplean generalmente RGB y HSL, dado que son los más comunes y presentan más ventajas a la hora de representar y analizar imágenes a color.

2.2.2. Etapas de un sistema de visión artificial.

Un sistema de visión artificial se divide fundamentalmente en tres etapas: La adquisición de la imagen, el procesamiento de la imagen y la toma de decisiones (visión computacional). [4].



En esta etapa se da la adquisición de las imágenes.

En esta etapa se realizan las transformaciones lineales y el filtrado sobre la imagen.

En esta etapa se utilizan técnicas y algoritmos para aislar los objetos a identificar del resto de la imagen.

En esta etapa se obtienen los parámetros y características que describen el objeto a reconocer.

En esta etapa se realiza el reconocimiento de la imagen, donde se identifica qué es el objeto analizando los parámetros obtenidos.

Por último, en esta etapa se interpretan los resultados obtenidos y se actúa según los mismos

2.2.3. Técnicas de visión artificial.

En este apartado se van a enumerar y explicar brevemente las distintas técnicas de visión artificial, incluyendo desde los métodos más tradicionales como el reconocimiento de patrones y las técnicas de procesado de imagen hasta los métodos más modernos basados en aprendizaje automático.

No se va a entrar en detalle sobre cuáles son los principios matemáticos en los que se basan los distintos algoritmos y técnicas, sino que se va a explicar de manera simplificada en qué consisten y qué es lo que se busca obtener aplicándolos.

Se va a dar especial atención a las técnicas empleadas en este proyecto, junto con aquellas que se consideran de obligado conocimiento en el ámbito de la visión artificial.

De la misma forma, se van a omitir aquellas técnicas de visión artificial que se consideren fuera de lugar o demasiado complejas como para poder hacer una mención breve de ellas.

Conocimientos básicos esenciales.

Todos los algoritmos y las técnicas que se van a mencionar tienen en común que se emplean para modificar una imagen o para extraer información de ella.

Es decir, dado que una imágen es una matriz tridimensional con valores de color para cada canal, tal y como se ha explicado en el apartado 2.2.1. El objetivo de las técnicas de visión artificial es obtener o modificar los valores de los píxeles de una imagen para facilitar su posterior procesamiento. [5].

Dado que los píxeles de una imagen se encuentran ordenados dentro de sus correspondientes matrices, es posible acceder individualmente a su valor o editarlo en la medida en la que se desee.

En visión artificial, en los métodos tradicionales se suele tratar con imágenes en escala de grises, ya que únicamente disponen de un canal de color sobre el que trabajar. Esto permite reducir los cálculos necesarios y aumentar la velocidad de procesado. Por tanto, existen múltiples herramientas que permiten extraer una imagen monocroma a partir de una imagen en color.

Otro tipo de imágenes en visión artificial son las imágenes binarizadas. Estas imágenes están compuestas por únicamente dos valores de color, blanco o negro, de ahí el nombre. Estas imágenes tienen un uso particularmente útil en técnicas de visión que se verán más adelante.

Una herramienta comúnmente empleada para analizar las características de una imagen en escala de grises es el histograma. [6].

Un histograma es una representación gráfica del número de píxeles de cada uno de los valores de gris. Los histogramas son una herramienta muy útil para conocer las propiedades de brillo y contraste de una imagen.

Si la distribución de píxeles por niveles de gris es uniforme, se dice que la imagen tiene mucho contraste. Si en cambio la distribución no es uniforme, se dice que tiene poco contraste.

Por otro lado, si la cantidad de píxeles se agrupa en uno de los lados del histograma, se dice que la imagen tiene mucho o poco brillo.

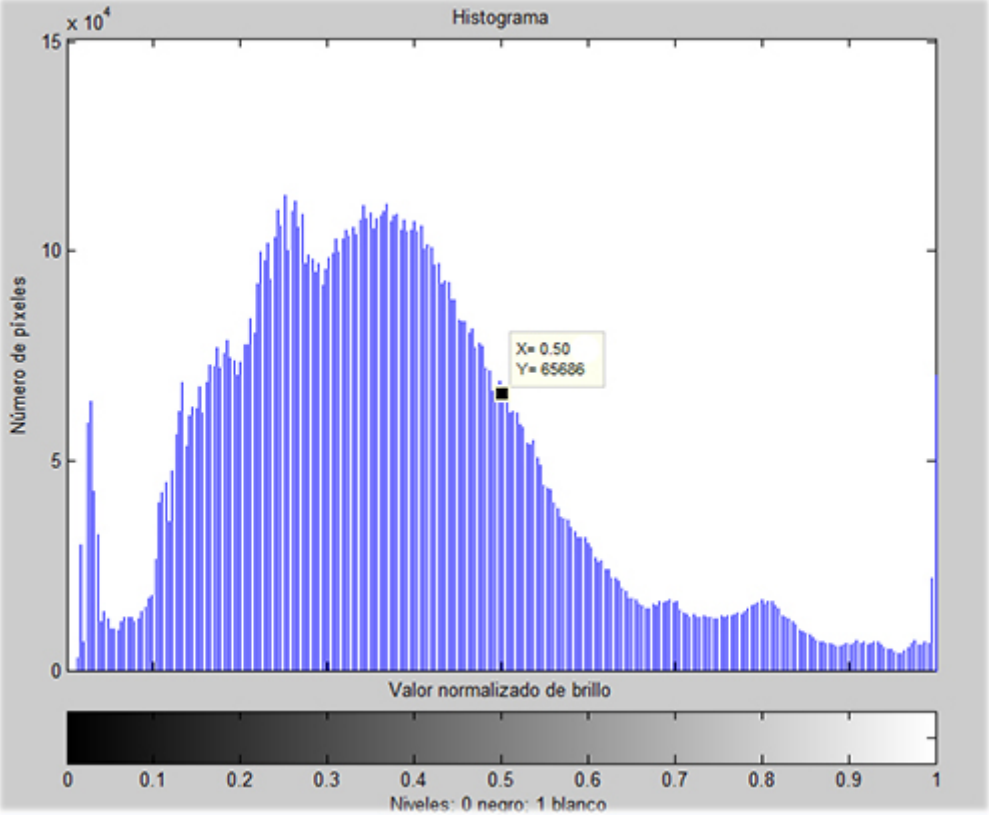


Figura 2.4: *Ejemplo de un histograma con bajo brillo y bajo contraste. De Fotoigual.com.*

También se pueden realizar transformaciones puntuales para aumentar el contraste, como por ejemplo ecualizar el histograma.

Muchos de los algoritmos complejos que se van a explicar a continuación extraen la información de un píxel con el fin de modificar los valores de los píxeles circundantes en función de los parámetros contenidos en una matriz. A esta matriz se la conoce como máscara o kernel.

Generalmente, las máscaras se aplican de manera selectiva a una región en concreto de la imagen, aunque se puede aplicar una máscara del tamaño que se desee e incluso aplicarla a una imagen.

Por ejemplo, se puede emplear para aumentar o disminuir los valores de brillo en una imagen. También se puede emplear para sustituir los valores de los píxeles de una imagen por otros predefinidos, para por ejemplo dibujar una figura en una imagen, como un círculo o un polígono.

Transformaciones geométricas.

En las transformaciones geométricas se determina una nueva posición de cada píxel por separado, basándose en la posición anterior de dicho píxel para realizar este cambio. [4].

Se dividen en tres categorías: Transformaciones rígidas, transformaciones afines y transformaciones proyectivas. Para explicar visualmente los tipos de transformación, se va a partir de la figura 2.5.

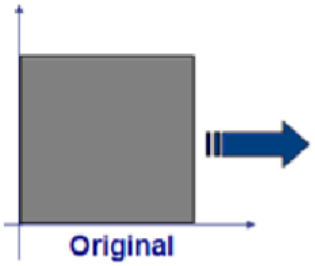


Figura 2.5: *Figura original para la explicación de las transformaciones. De La Fuente & Trespaderne, 2013.*

Las transformaciones rígidas preservan los ángulos y las distancias entre elementos. Los tipos de transformaciones rígidas que existen son los siguientes:

- Traslación.
- Rotación.
- Reflexión.

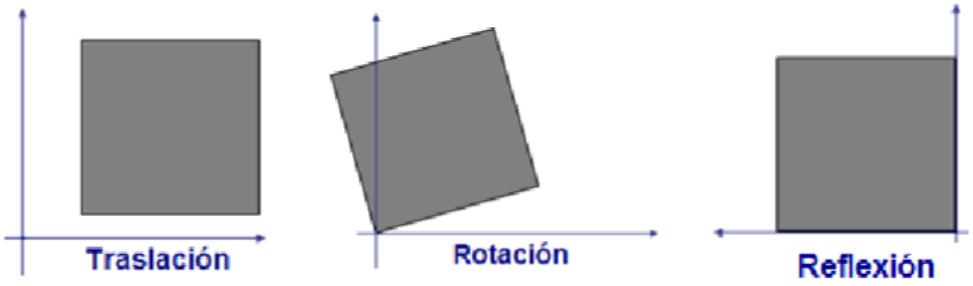


Figura 2.6: *Transformaciones rígidas. De La Fuente & Trespaderne, 2013.*

Las transformaciones afines preservan los paralelismos entre elementos, pero no preservan los ángulos. Los tipos de transformaciones afines que existen son los siguientes:

- Escalado: Isotrópico o anisotrópico (si mantiene la relación de aspecto tras el escalado o no).
- Cizalladura.
- Similitud.

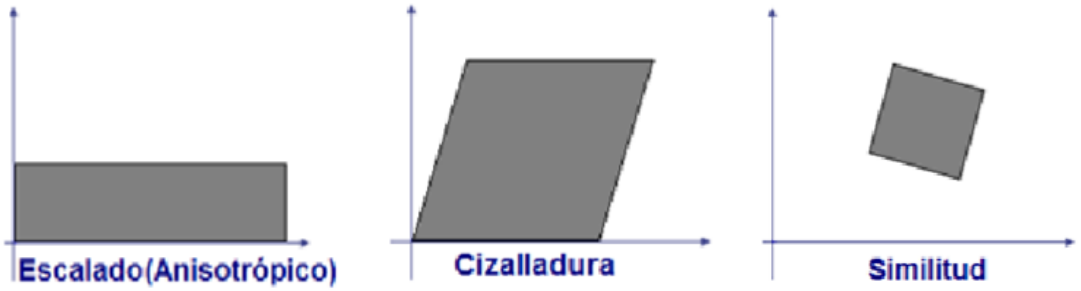


Figura 2.7: *Transformaciones afines. De La Fuente & Trespaderne, 2013.*

Las transformaciones proyectivas únicamente preservan la colinealidad entre los puntos de la imagen original, pero no preservan ni paralelismos ni ángulos.

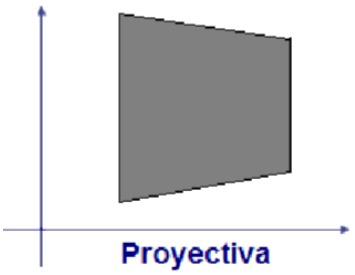


Figura 2.8: *Transformación proyectiva. De La Fuente & Trespaderne, 2013.*

Se pueden realizar un sinfín de combinaciones de transformaciones, para dar lugar al resultado deseado.

Un problema importante que surge al realizar transformaciones es que, en ocasiones, la posición resultante de un píxel puede no ser un valor entero, es decir, puede no caer sobre la retícula cuadrada de la imagen transformada.

Para resolver esto, se emplean algoritmos de interpolación, que calculan el color de cada celda de la retícula de la imagen resultante en función de los píxeles que la rodean.

Los tres tipos más comunes de interpolación son los siguientes:

- Vecino más próximo: Rápida y sencilla, pero resulta en imágenes escalonadas y con bordes pixelados.
- Bilineal: Se basa en aquellos píxeles situados en una matriz de 2x2 que rodean a la celda de la retícula. Buen resultado.
- Bicúbica: Similar a la bilineal, pero con una matriz de 4x4. Muy buen resultado.

Prácticamente siempre que se realice una transformación geométrica será necesario aplicar un algoritmo de interpolación.

Filtros.

Uno de los problemas más frecuentes que aparecen a la hora de analizar una imagen es el ruido. El ruido son píxeles aleatorios que toman un valor inesperado por varios motivos. Los dos tipos principales de ruido son el Gaussiano y el Sal y Pimienta, mostrados en la figura 2.9.

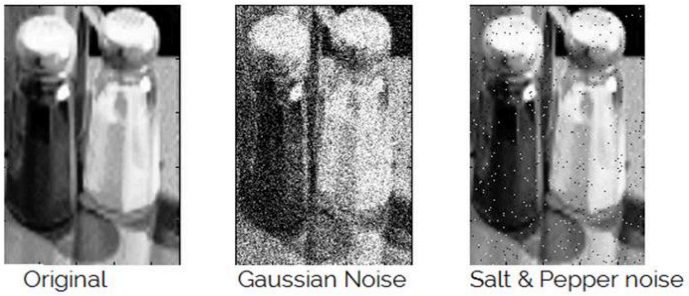


Figura 2.9: *Ruidos de Sal y Pimienta y Gaussiano. De devendrapratapyadav.github.io.*

Para solucionar este problema y eliminar este ruido, se hace uso de filtros. Los filtros se pueden emplear también para modificar las características de una imagen. Por ejemplo, se puede hacer uso de filtros realzantes o suavizantes para cambiar las propiedades de la imagen en conjunto. [4].

Fundamentalmente se dividen en dos tipos: Filtros lineales y no lineales.

Los filtros lineales hacen uso de una máscara o kernel que se aplica a todos los píxeles de la imagen, teniendo en cuenta todos los píxeles vecinos. A este proceso se le llama convolución de una máscara. Los filtros lineales son:

Filtros pasa bajo: Filtro suavizante. Tienden a hacer la imagen más borrosa.
Filtros pasa alto: Filtro realzante.
Filtros Gaussianos: Buenos contra el ruido Gaussiano.

En cambio, los filtros de suavizado no lineales son particularmente eficaces contra el ruido de Sal y Pimienta, en especial el filtro de la mediana.

Segmentación.

Como ya se ha explicado en el apartado 2.2.2, la segmentación consiste en separar los elementos de interés del fondo de una imagen. Para ello, se hace uso de detectores de líneas y bordes.

Los detectores de líneas son empleados especialmente para buscar una línea en una orientación específica. Por tanto, existen detectores de líneas horizontales, verticales y diagonales.

Los detectores de bordes buscan líneas continuas en las imágenes que separan regiones de distinto color, iluminación o textura.

Dentro de los detectores de bordes, el más común y más eficiente es sin duda el detector de Canny. Este detector ofrece los mejores resultados en prácticamente cualquier imagen, ya que se pueden variar ciertos parámetros para alterar la sensibilidad en la detección. Un ejemplo de su funcionamiento se muestra en la figura 2.10.

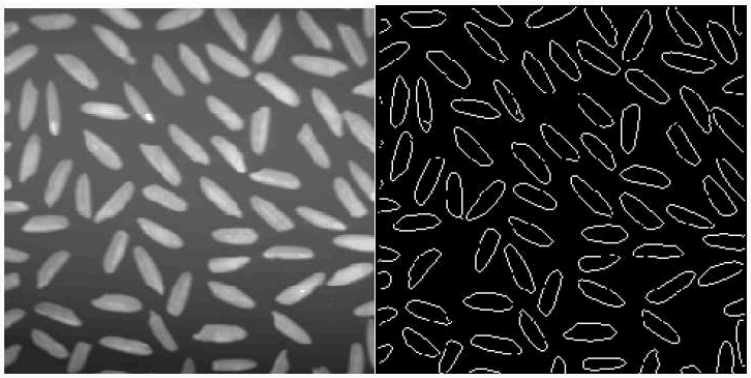


Figura 2.10: *Aplicación del algoritmo de detección de bordes de Canny. Por Jorge Valverde-Rebaza (2007).*

Una vez aplicado Canny, se pueden emplear buscadores de contornos cerrados para encontrar objetos en las imágenes. Estos contornos se clasifican por tamaño, expresado en píxeles cuadrados.

Aplicando algoritmos más complejos, se pueden establecer relaciones de padre-hijo entre los distintos contornos, es decir, se puede obtener una lista de qué contornos están inscritos dentro de otros.

Tratamientos morfológicos.

Los tratamientos morfológicos son un tipo especial de filtro que se emplea para cerrar contornos abiertos o para eliminar artefactos o ruido de las imágenes. Se emplean casi exclusivamente en imágenes binarizadas.

Para mostrar los efectos de estos tratamientos morfológicos, se hará uso de la figura 2.11.

Figura 2.11: *Imagen de ejemplo para tratamientos morfológicos. De docs.opencv.org.*

Los dos tipos fundamentales de tratamientos morfológicos son los siguientes:

Erosión: Consiste en aplicar un filtro a la imagen binarizada, donde se erosionan los contornos de una figura sólida, como se muestra en la figura 2.12.

Figura 2.12: *Operación de erosión sobre imagen de ejemplo. De docs.opencv.org.*

Dilatación: Es lo opuesto de la erosión. Consiste en aplicar un filtro a la imagen binarizada, donde se comprueba si un píxel negro se encuentra rodeado completamente de píxeles de igual valor. Si es así, no se hace nada. Si no es así, se cambia el valor de ese píxel a blanco. De esta manera se pueden aumentar contornos y cerrar espacios aislados en el interior de los contornos.

Figura 2.13: *Operación de dilatación sobre imagen de ejemplo. De docs.opencv.org.*

Estos dos tratamientos morfológicos en su conjunto dan lugar a técnicas muy interesantes, como son el Open y el Close.

Open: Dilatación + Erosión. Aplicando esta técnica, se pueden eliminar píxeles aislados en el interior de una figura, sin prácticamente cambiar la forma de la imagen.

Figura 2.14: *Operación de open sobre imagen de ejemplo con ruido. De docs.opencv.org.*

Close: Erosión + Dilatación. Aplicando esta técnica, se pueden eliminar píxeles aislados en el exterior de una figura, sin prácticamente cambiar la forma de la imagen.

Figura 2.15: *Operación de close sobre imagen de ejemplo con ruido. De docs.opencv.org.*

Reconocimiento de patrones (Pattern Matching o Template Matching).

El reconocimiento de patrones consiste en buscar una imagen o un patrón predefinido en el interior de otra imagen más grande. Existen dos tipos tradicionales pattern matching: Geométrico y de Correlación. [7].

El pattern matching geométrico busca los bordes de los objetos predefinidos en la imagen, lo que permite detectar objetos aunque se encuentren escalados o rotados.

Por otro lado, el pattern matching de correlación se basa en los niveles de gris del objeto predefinido para buscar coincidencias en la imagen deseada. Para emplear este método, es necesario mantener unas condiciones de iluminación constantes para evitar cambiar estos niveles de gris.

Para solucionar este problema, los algoritmos de pattern matching actuales devuelven sus resultados con un valor de confianza asignado.

Machine learning (Inteligencia artificial).

Hoy en día, la tendencia en las técnicas de visión artificial es el uso de inteligencia artificial, que ha demostrado ser una de las técnicas más revolucionarias de las últimas décadas.

La gran ventaja de los sistemas basados en inteligencia artificial es que permiten una gran flexibilidad en las condiciones ambientales de un sistema de visión, ya que no se basan simplemente en reconocimiento de patrones. En su lugar, una red neuronal es capaz de aprender a identificar con mucha precisión objetos, lo que es especialmente útil en imágenes que nunca ha visto.

En visión artificial se trabaja con una de las ramas de la inteligencia artificial especializada en el análisis de datos a gran escala, llamada Aprendizaje Automático o Machine Learning (ML). Esta técnica hace uso de modelos estadísticos en redes neuronales para identificar patrones en los datos y ofrecer un resultado evaluado con un grado de confianza. [8].

En este documento se van a emplear los nombres de aprendizaje automático, machine learning, inteligencia artificial y redes neuronales indistintamente.

Las redes neuronales tratan de imitar la arquitectura de un cerebro humano, y están compuestas por nodos interconectados llamados neuronas. Cada neurona se encuentra conectada con neuronas de la capa anterior y de la capa siguiente, y tienen una función de peso asociada que le permite enviar datos a la neurona de la capa siguiente si se cumplen los parámetros adecuados.

Una red neuronal simple está compuesta por una capa de entradas, una capa oculta y una capa de salidas.

En cambio, el Machine Learning está basado en un tipo de red neuronal llamado aprendizaje profundo (Deep Learning), que tiene varias capas ocultas interconectadas de forma no lineal. Algunas arquitecturas de Deep Learning pueden llegar a tener hasta cientos de capas ocultas. [9].

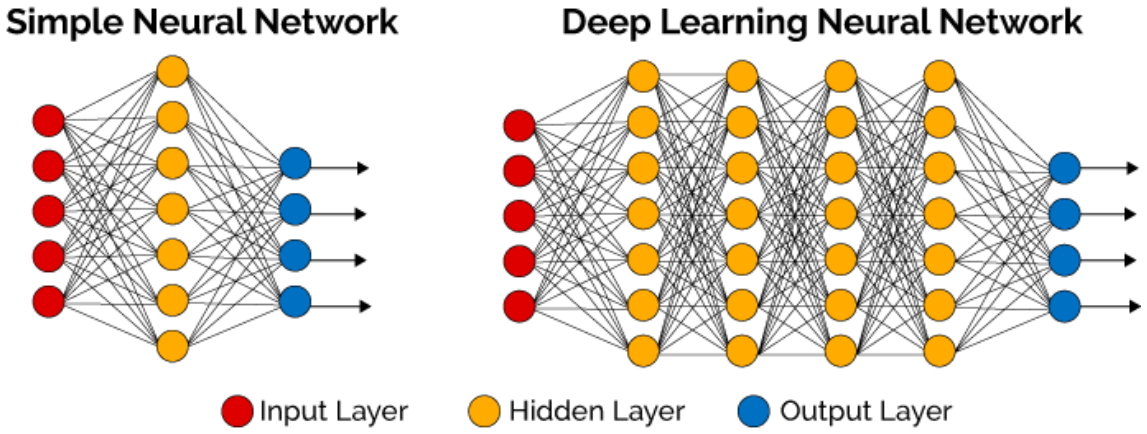


Figura 2.16: *Esquema de redes neuronales simples y de aprendizaje profundo. De Medium.com por Stacey Roman.*

Se distinguen principalmente tres tipos de redes neuronales que se pueden aplicar al campo de la visión artificial: Las redes neuronales artificiales (ANN), las redes neuronales convolucionales (CNN) y las Redes neuronales recurrentes (RNN). [10].

De estas tres arquitecturas, sin duda el tipo más popular y eficiente en el reconocimiento y análisis de imágenes son las redes convolucionales.

Estas redes se componen de varias capas donde se aplican filtros convolucionales a las imágenes, de manera que se puedan extraer de forma eficiente los parámetros descriptivos de los objetos a detectar.

Para que la red neuronal sea capaz de realizar la detección de objetos, se debe entrenar un modelo de predicción durante varias generaciones utilizando un banco de imágenes etiquetadas. Para etiquetar una imagen se señala una región de la imagen en la que se encuentre el objeto que se desea detectar.

A continuación se debe dividir el conjunto de imágenes en subconjuntos de entrenamiento, validación y comprobación. El conjunto de entrenamiento es con el que va a realizarse el entrenamiento de la red neuronal, el conjunto de validación se utiliza para que la red neuronal reciba el feedback sobre el entrenamiento y el conjunto de comprobación se utiliza una vez terminado el entrenamiento para terminar de evaluar los resultados.

Generalmente se distribuyen los conjuntos con un 60-80% para entrenamiento, 10-20% para validación y 10-20% para comprobación.

El modelo se entrena generalmente en una GPU (Graphics Processing Unit), la unidad gráfica de un ordenador convencional. El motivo es que la velocidad de procesamiento de cálculos en paralelo de una GPU es mucho mayor que la de una CPU (Central Processing Unit). Además, existen algunas TPU (Tensor Processing Unit) que permiten un entrenamiento aún más veloz.

Un problema muy común que puede ocurrir es el “overfitting” o sobreajuste. Este inconveniente consiste en un entrenamiento demasiado intenso de la red sobre un mismo set de imágenes, lo que lo vuelve malo para identificar imágenes diferentes a aquellas con las que se ha entrenado el modelo. Para evitar este problema, se puede reducir el número de épocas o iteraciones de entrenamiento.

Una vez entrenado el modelo de una red neuronal, se puede emplear las veces que sean necesarias, puesto que se puede convertir a otros formatos de redes neuronales como ONNX. Incluso, se puede continuar entrenando a partir del mismo modelo para tratar de mejorar la eficiencia del de dicho modelo.

Los resultados de una detección de un modelo de predicción basado en aprendizaje automático se suelen representar sobre la imagen original dibujando recuadros que engloban dicha detección, llamados “Bounding Boxes”. De esta manera se pueden valorar los resultados de forma visual.

Algoritmos NMS (Non Maximum Suppression).

Un problema común que surge a la hora de realizar detecciones, ya sea con un modelo de red neuronal o con reconocimiento de patrones, son las múltiples detecciones de un mismo objeto en la imagen, como se muestra en la figura 2.17.

Figura 2.17: *Múltiples detecciones de un mismo coche. De Jatin Prakash, 2021. En LearnOpenCV.com.*

Este problema dificulta en gran medida la detección de múltiples objetos en una imagen, puesto que no se puede contar el número de detecciones para determinar el número total de objetos. Para solucionar este problema es necesario aplicar un algoritmo NMS (Non Maximum Suppression). [11].

Este tipo de algoritmos eliminan las múltiples detecciones de un mismo objeto. Para ello, se toma una detección como referencia, y se calcula cuánto porcentaje de la bounding box de esta detección se solapa con las bounding boxes del resto de detecciones.

A continuación, se determina un umbral para el porcentaje máximo de tolerancia en la coincidencia de ambas bounding boxes. Se rechazan las detecciones que superen este umbral, de manera que se eliminan aquellas muy cercanas a la detección de referencia. El siguiente paso es guardar la detección de referencia y pasar a tomar la siguiente detección no eliminada como la de referencia.

Este proceso se repite hasta que no queden detecciones sin asignar como referencia o sin eliminar. El resultado es una lista con las detecciones reales en la imagen, como se puede ver en la figura 2.18.

Figura 2.18: *Algoritmo NMS aplicado sobre la imagen de un coche con múltiples detecciones. De Jatin Prakash, 2021. En LearnOpenCV.com.*

2.3. Recursos de software empleados en este proyecto.

En este apartado se mencionan y se explican los principales recursos de software empleados en el desarrollo de este proyecto.

Entre ellos se encuentran entornos de programación, librerías de código abierto, software para el entrenamiento de redes neuronales y bases de datos de imágenes para el entrenamiento de dichas redes.

Entornos de programación de PyCharm y Google Colab.

Para el desarrollo de este proyecto se va a trabajar con los entornos de programación de Pycharm y Google Colab, donde se emplearán Python 3.11 y CLI (Command Line Interface).

PyCharm Community Edition es un entorno de programación de Python 3, donde se va a desarrollar el cuerpo del código de los sistemas de detección y lectura de caracteres, así como la aplicación de registro. El motivo de la elección de PyCharm es su flexibilidad a la hora de trabajar en distintos proyectos simultáneamente, pudiendo compartirse las librerías entre proyectos.

Para los entrenamientos de las redes neuronales, así como para realizar algunas pruebas de funcionamiento con dichas redes, se va a emplear Google Colab. Este entorno emplea CLI y Python 3, y es particularmente eficiente a la hora de entrenar redes neuronales de aprendizaje profundo, ya que emplea los recursos de GPU de los PCs en el backend de Google. Esto optimiza en gran medida los tiempos de entrenamiento, y ofrece una compatibilidad completa con Google Drive para guardar los archivos resultantes.

OpenCV, Imutils y NumPy.

OpenCV e Imutils son dos librerías de código abierto muy populares en el análisis y el procesamiento de imágenes. Estas librerías emplean un gran repertorio de algoritmos, funciones e interfaces que permiten realizar complejos y extensos análisis de visión artificial sobre imágenes en unas pocas líneas de código, lo que simplifica en gran medida el desarrollo de cualquier algoritmo.

Individualmente presentan una gran cantidad de ventajas, pero ambas librerías pueden funcionar en conjunto para obtener resultados que por separado serían prácticamente imposibles. [12]. [13].

Por otro lado, NumPy es una librería que permite realizar una enorme cantidad de operaciones matemáticas con vectores y matrices, incluso en varias dimensiones. Esto facilita enormemente el trabajo con imágenes, dado que estas no son más que una matriz que representa la información de cada píxel.

Custom TkInter.

Custom TkInter es una librería de código abierto ampliamente empleada en el desarrollo de interfaces de usuario con una sencilla implementación en el código. La versión de Python 3 de esta librería ofrece una gran cantidad de posibilidades, y gracias a su amplia documentación y uso existen muchos recursos sobre los que instruirse en su funcionamiento. [14].

En este proyecto se va a emplear Custom TkInter en el desarrollo de una interfaz de usuario aplicable sobre el programa de lectura y registro de matrículas.

YOLO VISION.

En este proyecto se va a trabajar con un tipo de CNN conocido como YOLO (You Only Look Once). Esta arquitectura de red neuronal obtiene su nombre de su forma particular de procesar las imágenes. [15].

Otras CNN leen la imagen varias veces a la hora de procesarla, mientras que la arquitectura de YOLO le permite leer la imagen una única vez incluso para la detección de varias clases de elementos distintos.

Esto conlleva una gran mejora en la velocidad de inferencia de las imágenes en comparación con otras redes neuronales de análisis de imágen, como pueden ser por ejemplo SSD (Single Shot Detection) o Retina Net.

Gracias a estas características y a que es un software de código libre, YOLO se ha vuelto la red neuronal más popular en el análisis de imágenes, y no ha parado de mejorar su rendimiento y velocidad con los años.

Figura 2.19: *Logo de YOLO VISION, de Ultralytics.*

La versión con la que se va a trabajar en este proyecto es YOLO v8.0.196, una de las últimas versiones de esta red neuronal convolucional. Esta versión es particularmente estable a la hora de relacionarla con las librerías de análisis de imágenes ya mencionadas, como OpenCV e Imutils.

Además, esta versión permite una implementación completa mucho más cómoda y eficiente, así como muchas mejoras en la calidad del producto para los usuarios.

Roboflow y Dataset Ninja.

Roboflow y Dataset Ninja son unas herramientas públicas empleadas en el desarrollo, etiquetamiento y publicación de bases de datos de imágenes para ser empleadas en el entrenamiento de modelos de detección de objetos. [16].

Estas herramientas disponen de una gran cantidad de bancos de imágenes distintas, que permiten recopilar una gran cantidad de imágenes ya etiquetadas y listas para el entrenamiento de la red neuronal.

Además, Roboflow permite realizar la distribución de entrenamiento y validación de una base de datos de forma automática, así como implementar un modelo completo al repertorio de la base de datos.

En este proyecto se va a emplear Roboflow para recopilar, etiquetar y juntar bases de datos de imágenes de matrículas y los caracteres que las componen. Por otro lado, se va a emplear Dataset Ninja para añadir más imágenes al dataset.

EasyOCR y TesseractOCR.

EasyOCR y Tesseract son las dos librerías OCR de código abierto más conocidas y empleadas en una gran variedad de proyectos y aplicaciones.

Estas librerías hacen uso de modelos de detección basados en redes neuronales para realizar la detección de los caracteres de una imagen. Además de recoger la información leída en una cadena de texto, estas librerías ofrecen más información, como por ejemplo la ubicación de los caracteres en la imagen o la confianza en la correcta detección del texto.

En este proyecto se van a emplear como comparativa a los algoritmos que se van a desarrollar en los apartados 3.2 y 3.3.

Package Installer for Python (PIP).

Otro recurso relevante del que se va a hacer uso en el proyecto es el Package Installer for Python.

Este addon de Python permite instalar y desinstalar librerías empleando una única línea de código, donde se puede especificar la versión que se desea descargar, así como descargar todas las librerías dependientes asociadas.

2.4. Matrículas.

Aunque el color, las dimensiones, la tipografía e incluso el alfabeto cambie entre matrículas de todo el mundo, todas tienen características en común que se van a emplear en este proyecto para poder desarrollar el algoritmo ANPR.

En esencia, una matrícula es una placa metálica identificativa con forma rectangular, única para cada vehículo y compuesta por al menos cuatro caracteres y números denominados glifos.

Partiendo desde esta definición universal de matrícula, se va a enumerar las características específicas de los distintos tipos de matrículas que se van a identificar en este proyecto: europeas, estadounidenses, canadienses y australianas. El motivo de esta elección de países es la gran cantidad de imágenes de vehículos disponibles en internet.

Matrículas europeas.

En primer lugar, se tienen las matrículas europeas. [17]. Estas matrículas se rigen por una normativa establecida por el parlamento europeo, como se explica más adelante en el apartado 2.6.

Las matrículas europeas tienen un distintivo rectangular identificativo en la parte izquierda, donde se muestra la bandera de la unión europea junto con un código alfabético para indicar el país de procedencia. En cuanto a las dimensiones y tipografía, se muestra un resumen en la tabla 2.1.

Europa		
Matrícula	**Dimensiones**	**Tipografía**
Turismo	520 x 110 mm	DIN 1451
Camión	520 x 200 mm	DIN 1451
Motocicleta	180 x 200 mm	DIN 1451

Tabla 2.1: *Características de las matrículas europeas. De coceurope.eu y leewardpro.com.*

En la figura 2.20 se muestra un ejemplo de una matrícula europea, en concreto una matrícula de Portugal.

Figura 2.20: *Matrícula de Portugal. De matriculasdelmundo.com.*

Matrículas de EE.UU.

Por otro lado, tenemos las matrículas de NorteAmérica. Por resumir este apartado, se van a ignorar todas las normativas especiales para cada estado de EEUU, que pueden variar levemente la establecida por convenio entre Canadá y Estados Unidos. [18].

EE.UU		
Matrícula	**Dimensiones**	**Tipografía**
Turismo	300 x 150 mm	Driver Gothic, Penitentiary Gothic
Camión	300 x 150 mm	Driver Gothic, Penitentiary Gothic
Motocicleta	180 x 100 mm	Driver Gothic, Penitentiary Gothic

Tabla 2.2: *Características de las matrículas de EE.UU. De matriculasdelmundo.com y leewardpro.com.*

En EE.UU, las matrículas tienen un estilo, color, dibujo o incluso tipografía diferentes. Además, en la parte superior de las matrículas se muestra el nombre del estado de matriculación, mientras que en la parte inferior se muestra un texto o frase representativos de dicho estado.

En la figura 2.21 se muestran algunas de las variantes de matrículas en Estados Unidos, una para cada estado.

Figura 2.21: *Matrículas de EE.UU. De worldlicenseplates.com.*

Matrículas de Canadá.

En Canadá, las matrículas presentan unas características en tamaño y tipografía muy similares a las de EE.UU. Esto se debe a un trato realizado por ambos países para unificar la norma de las matrículas de ambos países. [19].

Canadá		
Matrícula	**Dimensiones**	**Tipografía**
Turismo	300 x 150 mm	Driver Gothic, Penitentiary Gothic
Camión	300 x 150 mm	Driver Gothic, Penitentiary Gothic
Motocicleta	180 x 100 mm	Driver Gothic, Penitentiary Gothic

Tabla 2.3: *Características de las matrículas de Canadá. De matriculasdelmundo.com y leewardpro.com.*

De la misma manera que en Estados Unidos, las matrículas en Canadá presentan dibujos en función de la provincia en la que se ha matriculado el vehículo.

En la figura 2.22 se muestran las matrículas de algunos de las provincias y territorios de Canadá.

Figura 2.22: *Matrículas de Canadá. De worldlicenseplates.com.*

Matrículas de Australia y Nueva Zelanda.

En Australia y Nueva Zelanda las matrículas son más similares a las europeas en tamaño y estilo, aunque presentan algunas diferencias en estilo entre las distintas provincias. [20].

Australia		
Matrícula	**Dimensiones**	**Tipografía**
Turismo	372 x 134 mm	SAA Series “B”
Camión	372 x 134 mm	SAA Series “B”
Motocicleta	254 x 100 mm	SAA Series “B”

Tabla 2.4: *Características de las matrículas de Australia. De matriculasdelmundo.com y leewardpro.com.*

Figura 2.23: *Matrículas de Australia. De worldlicenseplates.com.*

Nueva Zelanda		
Matrícula	**Dimensiones**	**Tipografía**
Turismo	360 x 125 mm	Licenz
Camión	360 x 125 mm	Licenz
Motocicleta	260 x 100 mm	Licenz

Tabla 2.5: *Características de las matrículas de Nueva Zelanda. De matriculasdelmundo.com y leewardpro.com.*

Figura 2.24: *Matrículas de Nueva Zelanda. De worldlicenseplates.com.*

2.5. Estado del arte.

El primer trabajo relevante respecto al proyecto actual es de autoría de Kusumadewi, Ira (2019). [21]. El artículo científico llamado “License Number Plate Recognition using Template Matching and Bounding Box Method” fue publicado en la revista Journal of Physics en 2019 gracias a su interesante solución para un sistema ANPR.

El objetivo de este proyecto fue el desarrollo de un sistema de reconocimiento de matrículas basado en reconocimiento de patrones empleando el uso de imágenes como plantillas.

Aunque este sistema fue diseñado específicamente para vehículos de Indonesia, el acercamiento al problema y su solución han sido un gran pilar sobre el que basar este proyecto. Lo más destacable de este proyecto, en mi opinión, fue su empleo de métodos tradicionales para el tratamiento de la imagen y la simplicidad en la explicación del funcionamiento del algoritmo.

El segundo trabajo que ha servido como guía general para el desarrollo de los algoritmos basados en Machine Learning proviene de Kasper-Eulaers, Margrit & Hahn, Nico & Berger, Stian & Sebulonsen, Tom & Myrland, Øystein & Kummervold, Per (2021). [22]. El nombre de este proyecto es “Short Communication: Detecting Heavy Goods Vehicles in Rest Areas in Winter Conditions Using YOLOv5”.

El objetivo de este proyecto es desarrollar un algoritmo basado en YOLO que permita identificar los espacios libres en un parking al aire de vehículos pesados en invierno. El algoritmo es capaz de detectar y contar los vehículos en dicho parking, con el fin de plantear una solución para aquellos conductores buscando un aparcamiento en las condiciones de un invierno duro.

Este trabajo ha servido para entender la forma de entrenar y aplicar una red neuronal desde cero, así como comprender en detalle la forma de interpretar los datos que ofrece como resultado dicha red neuronal.

El tercer trabajo del cual se ha obtenido una gran información es de autoría de Elias Ccoto Huallpa, Abel Angel Sullon Macalupu, Jorge Eddy Otazu Luque, Jorge Sánchez-Garces (2022) [23]. El trabajo tiene por título “Determinación del mejor algoritmo de detección de matrículas en ambientes controlados y no controlados”.

En este documento, publicado en la revista RISTI, se realiza un breve análisis teórico sobre los posibles métodos eficaces en el reconocimiento de matrículas para controlar los robos en las grandes ciudades de Perú.

Este análisis teórico ha servido para motivar el desarrollo de los métodos tradicionales de detección, ya que en este artículo se menciona un muy buen resultado de los mismos.

Por último, se ha empleado como guía orientativa para los primeros pasos el trabajo de Jorge Navacerrada (2017). [24]. El proyecto tiene como título “Sistema de detección de matrículas con Open CV”.

En esta memoria se ha descrito el desarrollo de un sistema de detección de matrículas haciendo uso de algoritmos SURF.

El proyecto tiene un estado del arte muy extenso y claro, donde las explicaciones sobre los problemas encontrados y las decisiones de ingeniería tomadas han hecho de este un recurso muy orientativo, especialmente en las primeras etapas de desarrollo.

2.6. Normativa aplicable.

De acuerdo con la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, las matrículas pasan a ser datos de carácter personal susceptibles a la ley de protección de datos. [25].

Esta ley se aplica en Europa de la misma manera, debido a la validación de la regulación europea 2016/679, donde las matrículas pasan a ser protegidas por la GDPR (General Data Protection Regulation). [26].

Por tanto, es imperativo consultar la normativa vigente en cada país o región donde se desee instalar este sistema de control de acceso basado en ANPR.

En el caso de España, disponer de un registro de matrículas vuelve al organismo poseedor de dicha información susceptible a la Ley Orgánica de Protección de Datos (LOPD).

Si se desea realizar una instalación de un sistema ANPR en España, se debe cumplir con la regulación establecida en la Ley Orgánica 15/1999. En este decreto se regula concretamente el uso de sistemas ANPR en barreras de acceso a recintos privados, como se explica en profundidad en el informe 297/2012 de la AEPD (Agencia Española de Protección de Datos). [27]. [28].

En este informe se expresa claramente que se podrá disponer de un registro de matrículas, siempre y cuando la imagen obtenida no sea conservada, sino que sea la cadena de texto extraída de la imagen lo que se almacene en un documento externo.

Además, en dicho informe se hace especial hincapié en el hecho de que sólo se permite la obtención de estos datos personales si son obtenidos para una finalidad determinada, explícita y legítima.

En este caso, la finalidad de la obtención de los datos es la siguiente:

- Proveer a la compañía, edificio o fábrica de un registro de matrículas asociado a las entradas y salidas del recinto, con la finalidad de mejorar los tiempos de los procesos empleados.

- Proveer a la compañía, edificio o fábrica de un registro de matrículas asociado a las entradas y salidas del recinto, con la finalidad de emplearse en vigilancia, para la erradicación o mitigación de comportamientos fraudulentos.

Por tanto y en conformidad con todas normativas aplicables por la Ley Orgánica 15/1999 y la Ley Orgánica 3/2018, el registro de los datos de matrículas que accedan a los recintos privados de los organismos implicados será legal en su totalidad.

Por otro lado, existen leyes asociadas a la posición, dimensiones y tipografía de las matrículas.

Estas leyes normalizan dichas características, de manera que en este proyecto puede ser muy útil tenerlas presentes. Dichas características se han mencionado en el apartado 2.4.

El principal inconveniente es que dicha normativa varía en función del país. Por ejemplo, todos los países de la Unión Europea se rigen por el reglamento 1003/2010. [29]. [30].

Esto significa que todos estos países deben, como mínimo, cumplir con dicha norma, que regula la posición, dimensiones, materiales, tipografía, altura máxima y altura mínima de las matrículas.

En Europa, esta altura mínima del borde inferior de la matrícula es de 0,30 metros, mientras que la altura máxima del borde superior puede ser de 1,20 metros o en casos excepcionales de 2,00 metros. La tipografía y las recomendaciones para OCR siguen la norma ISO/IEC 30116:2016. [31].

En España, esta ley europea se ha adoptado en el Real Decreto 885/2020, que regula con las mismas restricciones el tamaño y posición de las matrículas españolas. [32]

CAPÍTULO 3. DESARROLLO DE ALGORITMO ANPR.

En este capítulo se relatan todos los pasos necesarios para el desarrollo del algoritmo ANPR, así como todas las decisiones de ingeniería tomadas a lo largo del proceso de desarrollo.

En este capítulo se incluyen la elección de las características de una cámara para la captura de imagen, el desarrollo de los algoritmos de detección y OCR, una comparativa de los resultados para determinar los mejores algoritmos a emplear y el desarrollo del algoritmo en cascada final.

Se van a definir las propiedades en cuanto a la posición y propiedades de hardware de la cámara, para facilitar el desarrollo de los algoritmos.

Se va a desarrollar un algoritmo en cascada, que en primer lugar detecte la presencia de una matrícula en una imagen y calcule su ubicación, para posteriormente leer los datos de la matrícula en la región indicada.

Para la explicación de los algoritmos de detección y OCR se realizarán ejemplos sobre las mismas imágenes fijas como ejemplo.

Es importante recordar que un vídeo no es más que una serie de imágenes representadas sucesivamente, por lo que el algoritmo final simplemente va a leer el fotograma actual del vídeo de entrada, va a realizar la detección y lectura de la matrícula y se va a pasar al siguiente fotograma de la imagen, continuando este ciclo por cada iteración del código.

3.1. Captura de imagen.

En primer lugar, antes de comenzar con el desarrollo de los algoritmos se deben conocer las características del entorno donde se va a aplicar el algoritmo, así como la localización y las propiedades de la cámara a emplear. De esta manera se puede ajustar el algoritmo a las condiciones de operación, para asegurar un funcionamiento más controlado del mismo.

3.1.1. Entorno de trabajo.

Como ya se ha mencionado anteriormente, el sistema ANPR se va a diseñar para funcionar en una barrera automática que permite acceso a un entorno industrial. Por tanto, se asume que el sistema se encontrará en el exterior, como el ejemplo mostrado en la figura 3.1.

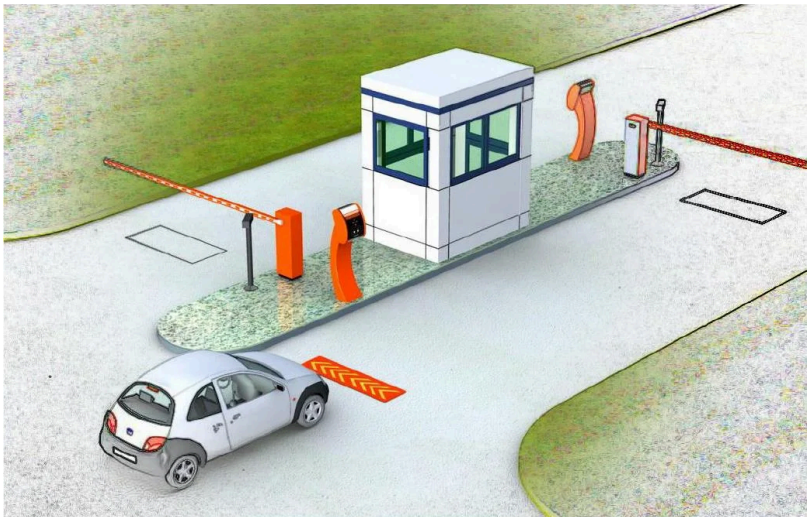


Figura 3.1: *Doble barrera automática. De Lorenzo, Jorge, en issuu.com.*

Esto significa que el sistema se encontrará expuesto a las condiciones ambientales, ya sean climatológicas como la lluvia y niebla o externas como la intervención humana o animal, entre otros muchos factores.

La entrada se encontrará iluminada durante la noche y las horas de poca luz, por farolas o focos. Las zonas donde el vehículo se debe detener para que se realice la lectura de la matrícula se encontrará marcada con indicaciones en el asfalto.

Además, es probable que el recinto disponga de varios accesos, por lo que se ha de tener en cuenta este hecho a la hora de desarrollar la lógica del sistema.

3.1.2. Posición de la cámara.

En segundo lugar se va a definir la localización de la cámara.

Puesto que el sistema ANPR se va a encontrar situado en una barrera de acceso y se desconoce la longitud de los vehículos que van a atravesar dicha barrera, la solución más sencilla es captar la matrícula delantera del vehículo.

Para ello, se han de colocar unas marcas en el suelo que indican al conductor del vehículo una zona aproximada donde debe detenerse. La cámara debe encontrarse lo más cerca posible de la barrera, orientada de tal manera que se vea claramente la matrícula de cualquier camión o turismo.

Se debe permitir un margen de error amplio en la ubicación esperada de la matrícula, para tener en cuenta los errores humanos cometidos por los conductores.

La cámara se situará a una altura de entre 0.3 y 0.8 metros, e idealmente debe captar matrículas a no más de 5 metros de distancia, pudiéndose hacer uso de herramientas de zoom o lentes para ajustar correctamente el encuadre deseado. La cámara puede encontrarse incorporada en el armario eléctrico que contiene el mecanismo que levanta la barrera, anclado a una pared, sobre un poste o en un soporte especializado.

Al mantener la matrícula lo más cerca posible de la cámara, se reduce enormemente el efecto de los fenómenos meteorológicos del exterior.

Un sencillo ejemplo de una instalación con estas características se muestra en la figura 3.2.

Figura 3.2: *Barrera automática y cámaras. De adfabsgates.co.uk.*

3.1.3. Propiedades de la cámara.

En tercer lugar se van a definir las características de la cámara. Las características relevantes para el desarrollo del algoritmo son la calidad de la imagen, el color, los fotogramas por segundo, el tipo de luz que se detecta y el tipo de obturador. A continuación se van a explicar las decisiones de ingeniería tomadas sobre estos parámetros.

Color.

En los algoritmos de visión artificial tradicionales generalmente se hace uso de imágenes en escala de grises para optimizar la velocidad de procesamiento y deshacerse de la “irrelevante” información que se aporta mediante el color. En caso de aplicar algoritmos tradicionales para la detección o OCR, lo más óptimo es sin duda emplear imágenes en escala de grises.

Por otro lado, si se pretende emplear algoritmos basados en modelos de detección de redes neuronales, es relevante mantener las imágenes a color. Esto es debido a que dichas arquitecturas de redes neuronales hacen uso de estas características de color para realizar las predicciones, aunque pueden funcionar perfectamente con imágenes en grises.

Para las pruebas de algoritmos tradicionales se va a convertir las imágenes a color en escala de grises, mientras que para las pruebas de modelos de predicción de YOLO se hará uso de imágenes a color.

Por tanto, si la cámara es monocroma o a color dependerá del algoritmo a emplear, así como del tipo de luz a captar, tal y como se explica más adelante.

Calidad de la imagen.

Se pretende obtener la mayor cantidad de información posible de las imágenes, como por ejemplo bordes definidos, poco ruido, buena nitidez y buen contraste en la imagen. Esto ayuda a desarrollar los algoritmos, dado que obtener de más calidad en la imagen sin duda fomenta las capacidades de detección de formas o líneas en algoritmos tradicionales.

Por otro lado, la arquitectura de YOLO trabaja con imágenes de 640 x 640 píxeles, por lo que lo importante es obtener imágenes claras, nítidas y con un alto contraste.

En conclusión, obtener una imagen de buena calidad es un paso necesario para obtener un buen rendimiento de los algoritmos. Esto significa obtener imágenes claras, nítidas, con alto contraste y poco ruido. En caso de requerir una cámara de peor calidad pero de mejor precio, se pueden mejorar estas características en las etapas de preprocesado, pero el objetivo es disponer de una buena imagen de partida.

Tipo de luz a detectar.

En caso querer operar el algoritmo en condiciones de baja luminosidad, se debe hacer uso de cámaras infrarrojas. Esto conlleva una pérdida completa de la información de color de la imagen.

Dado que la información que ofrece disponer de una imagen a color puede ser relevante para los algoritmos, se considera la posibilidad de utilizar los focos y farolas de la instalación para evitar el uso de estas cámaras infrarrojas, aunque de esta manera se pierda mucha calidad en la imagen.

Por estos motivos y para obtener lo mejor de ambos sistemas, se ha decidido emplear una cámara híbrida, que durante el día funcione como una cámara de vídeo estándar y durante las horas de menos luz funcione como una cámara infrarroja. Es necesario incorporar linternas infrarrojas en la misma cámara para proveer al sistema de una iluminación consistente y robusta.

Velocidad de captura de vídeo.

Los algoritmos de detección y de OCR tienen unos altos tiempos de procesado, por lo que disponer de muchos fotogramas por segundo para el procesado no es estrictamente necesario.

Si se suma este hecho a que no es necesaria una respuesta inmediata del algoritmo, puesto que se plantea que la ejecución del mismo sea de 1 segundo o menos, se determina que la velocidad de captura de vídeo debe ser baja. Por tanto, dicha velocidad de captura debe ser de alrededor de 30 fotogramas por segundo.

Tipo de obturador.

El obturador funciona como una cortina que permite el paso de la luz hacia el sensor. Los obturadores más baratos son de tipo “rolling shutter” y presentan deformidades en las imágenes de objetos captados en movimiento.

Para solucionar estos problemas, se hace uso de un tipo de obturador llamado “global shutter”. Este obturador permite captar imágenes de objetos en movimiento sin estos artefactos y deformidades.

En este caso, los vehículos que pretenden cruzar la barrera han de encontrarse quietos o desplazarse muy lentamente. Por este motivo, las deformidades generadas por el uso de un obturador “rolling shutter” son ínfimas, y serían únicamente relevantes en caso de querer realizar metrología.

Dado que no se busca realizar metrología sobre las imágenes captadas, el obturador a escoger para esta aplicación será de tipo “Rolling Shutter”.

Propiedades físicas.

Puesto que el sistema ANPR puede ser instalado en exteriores, es importante que la cámara sea resistente ante los distintos eventos climatológicos, así como la humedad y el polvo.

Por tanto, una cámara expuesta al exterior debe tener como mínimo un grado de protección IP 65. El primer dígito representa la resistencia ante polvo, y el segundo ante humedad.

Un IP65 tiene la máxima resistencia ante polvo y alta resistencia a la humedad, hasta el límite de no poder sumergirse en agua, como se muestra en la figura 3.3.

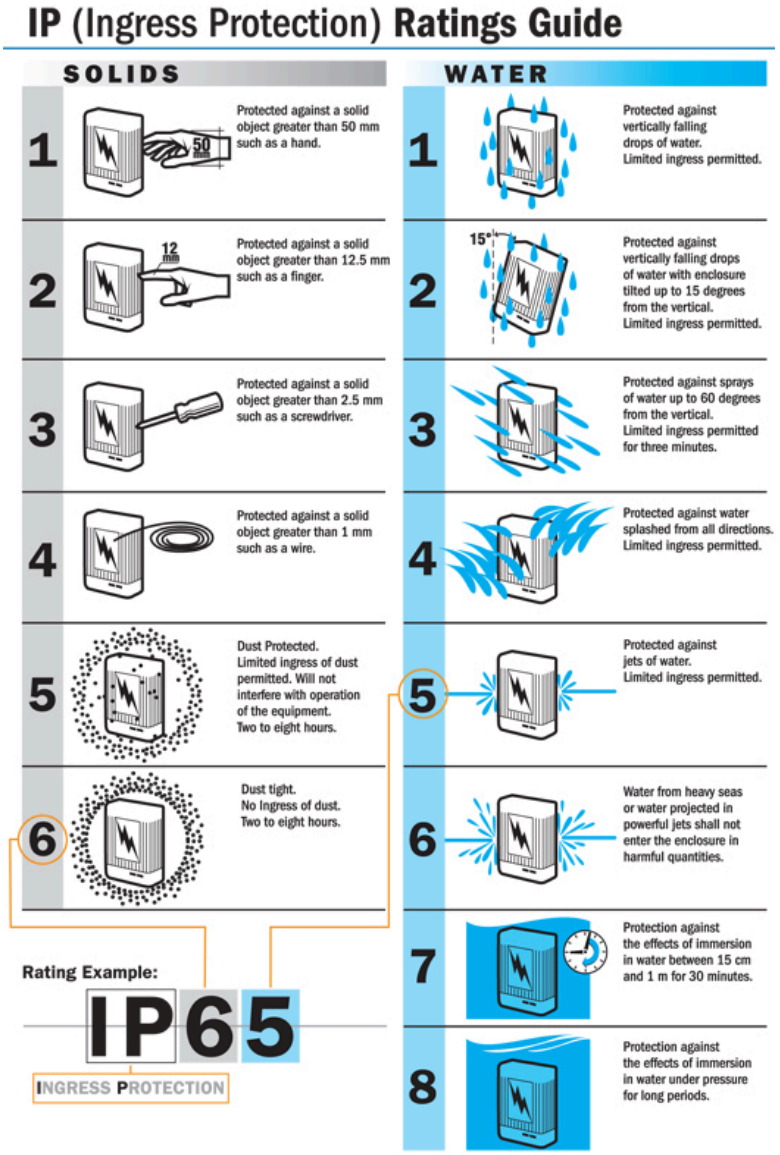


Figura 3.3: *Grado de protección IP 65. De BlueSeaSystems en blusesea.com.*

3.2. Algoritmos de detección de matrículas.

Una vez obtenida la imagen, el siguiente paso para desarrollar el algoritmo ANPR es determinar si existe una matrícula en la imagen captada. En caso afirmativo, el algoritmo debe encontrar la ubicación de dicha matrícula.

El resultado de este segmento del programa será un recorte de la región donde se encuentra la matrícula en la imagen original, para facilitar el procesamiento de la información de la misma.

En todos los casos se va a emplear la librería de OpenCV para el preprocesado, tratamiento y lectura de las imágenes. Se va a desarrollar el código en Python 3, en el entorno de PyCharm Community Edition.

A continuación se van a desarrollar tres algoritmos de detección distintos:

- Un algoritmo de detección basado en métodos tradicionales.
- Un algoritmo de detección basado en reconocimiento de patrones.
- Un algoritmo de detección basado en aprendizaje automático.

Para realizar la explicación de los algoritmos de detección, se hará uso de la imagen mostrada en la figura 3.4 a modo de ejemplo.

Figura 3.4: *Misteriosa matrícula "COVID19". De Eugene Boisvert, en abc.net.*

3.2.1. Algoritmo de detección de matrículas basado en métodos tradicionales.

En primer lugar, se va a desarrollar el algoritmo de detección de matrículas basado en métodos tradicionales. Como el nombre indica, este algoritmo va a hacer uso de métodos de análisis de imágenes tradicionales para determinar la ubicación de una matrícula en la imagen captada.

Se va a hacer uso de la librería de OpenCV para aplicar las técnicas tradicionales de procesado de imágenes. [33].

Antes de comenzar con el desarrollo se van a realizar una serie de hipótesis sobre la eficacia y eficiencia de este tipo de algoritmo.

Se puede asumir que mediante este método se va a obtener un procesamiento muy rápido de la imagen, y se asume que el algoritmo no será extremadamente robusto ante cambios de iluminación, posicionamiento del vehículo en la imagen o elementos externos como suciedad en la matrícula.

Aunque este sistema presenta tanta inestabilidad ante cambios en el entorno, se puede afinar la configuración ajustando los parámetros necesarios para cada instalación física, de manera que se reduzcan en gran medida las variables ambientales.

Esto significa que se debe dedicar un tiempo de trabajo para la recolección de muestras en cada sistema ANPR, con el fin de realizar los ajustes necesarios en los parámetros para optimizar el algoritmo.

Dado que en este algoritmo no se va utilizar ningún modelo de predicción ni ninguna herramienta de reconocimiento de patrones, la etapa de preprocesamiento se convierte en la etapa más importante del algoritmo de detección.

Es extremadamente importante proveer a la etapa de procesamiento una imagen clara, sin ruido, y con los ajustes necesarios para facilitar a los segmentos de procesamiento y segmentación del algoritmo.

Tras desarrollar varias de versiones del algoritmo, se ha llegado a una versión final que implementa una gran variedad de técnicas de preprocesado que asegura una posible detección, siempre y cuando se ajusten los parámetros adecuados para adaptarse al entorno deseado.

Desarrollo del segmento del algoritmo.

Preprocesado.

El primer paso para el desarrollo del algoritmo es realizar un preprocesado de la imagen. El preprocesado va a seguir las siguientes etapas:

- Conversión de la imagen a grises.
- Transformación afín.
- Aplicación de una máscara predefinida.
- Aplicación de filtros de ruido y filtros suavizantes.
- Aplicación del algoritmo de detección de bordes de Canny.
- Aplicación de un Open personalizado.

A continuación se explican detalladamente cada uno de los pasos, así como el razonamiento detrás de cada paso y las decisiones de ingeniería tomadas en cada momento.

En primer lugar, se carga la imagen al programa junto con las librerías de “cv2” y “numpy”. Se va a utilizar para la explicación del algoritmo la imagen mostrada anteriormente en la figura anterior.

El siguiente paso una vez cargada la imagen consiste en convertirla a escala de grises. Esta conversión se realiza para eliminar los canales de color, ya que no aportan mucho valor en los métodos de detección tradicionales. Además, reducir de tres canales de color a uno conlleva una optimización en los tiempos de procesado de la imagen.

El resultado de la conversión a grises se muestra en la figura 3.5.

Figura 3.5: *Conversión a escala de grises aplicada a la imagen de entrada. Elaboración propia.*

El siguiente paso consiste en aplicar una transformación afín a la imagen para orientar la matrícula hasta una posición horizontal.

En la realidad, si la matrícula no se encuentra en posición horizontal en la imagen en primer lugar, sería necesario realizar una transformación de perspectiva para lograr una imagen rectangular con la matrícula. Esto es debido a las propiedades de una cámara a la hora de tomar una imagen de un objeto en tres dimensiones.

Tras incontables horas tratando de implementar esta transformación proyectiva, encontré varios inconvenientes.

El principal problema es que la transformación proyectiva se emplea exclusivamente para rescatar una región de una imagen. Por tanto, si se aplicase sobre la figura completa, el resto de la imagen quedaría irreconociblemente deformada y desproporcionada. Dado que no se conoce la ubicación de la matrícula, es imposible realizar la transformación proyectiva.

Es por esto que se ha decidido emplear una transformación afín en su lugar, ya que se mantienen las relaciones de paralelidad entre todos los puntos de la imagen, deformando por igual todos los puntos de la imagen.

Para conocer los parámetros de la transformación afín, es necesario conocer la orientación del vehículo con respecto de la cámara, algo particular para cada instalación.

Estos parámetros se pueden determinar con una imagen de ejemplo tomada en la ubicación exacta de la lectura.

Una vez se conocen estos datos, gracias a la librería de OpenCV se aplican los comandos necesarios para determinar la matriz de transformación, así como el comando “cv2.warpAffine” para aplicar esta transformación a la imagen. El resultado de la transformación afín se muestra en la figura 3.6.

Figura 3.6: *Transformación afín aplicada a la imagen de entrada. Elaboración propia.*

Una vez aplicada la transformación, se va a aplicar una máscara personalizada a la imagen para que el algoritmo pueda ejecutarse sobre la zona en la que es más probable que se encuentre una matrícula. [34].

De esta manera, se logra evitar detectar otros polígonos rectangulares pertenecientes a otras partes del vehículo, además de mejorar la velocidad de procesamiento al eliminar una gran cantidad de píxeles.

Esta región debe cambiar en cada instalación física del sistema ANPR, pero se puede asumir que gracias a las características en la captura de imagen definidas en el apartado 3.1, todas las matrículas a detectar se encontrarán en el interior de una región específica de la imagen.

Se han creado distintas máscaras para esta etapa. En la figura 3.7 se muestran algunos ejemplos de dichas máscaras.

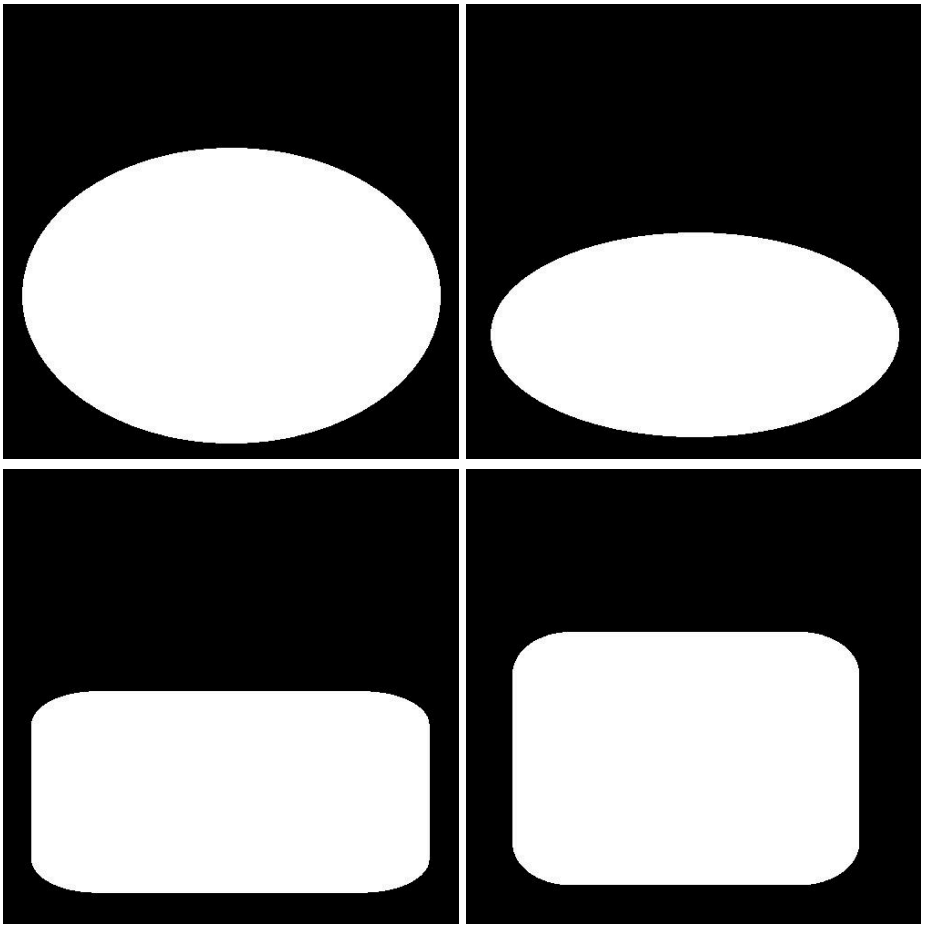


Figura 3.7: *Máscaras para el preprocesado del sistema de detección. Elaboración propia.*

Estas máscaras se redimensionan para encajar sobre la imagen en la que se está aplicando el algoritmo, y mediante el uso del comando “cv2.bitWiseAnd” se aplica sobre la imagen actual, resultando en la figura 3.8.

Figura 3.8: *Imagen con la máscara de preprocesado aplicada. Elaboración propia.*

Una vez que disponemos de la imagen en grises, con la transformación afín y con la máscara aplicada, se procede a aplicar filtros de ruido y filtros suavizantes.

El objetivo de aplicar estos filtros es reducir las anomalías en píxeles aleatorios de la imagen (filtros de ruido), así como homogeneizar los bordes y eliminar los bordes menos prominentes.

Con esto se concluye la etapa de preprocesado. Se ha preparado una imagen con características uniformes lista para ser procesada.

Segmentación.

La siguiente etapa del algoritmo es la segmentación. En esta etapa se busca detectar la matrícula en la imagen y aislarla del resto de elementos de la misma.

Para ello, se comienza por aplicar el buscador de bordes de Canny a la imagen preprocesada, con el fin de encontrar los bordes definidos por la placa de la matrícula.

Los parámetros de este detector de bordes dependen de las condiciones físicas de cada sistema específico, y se deben ajustar en cada instalación física particular.

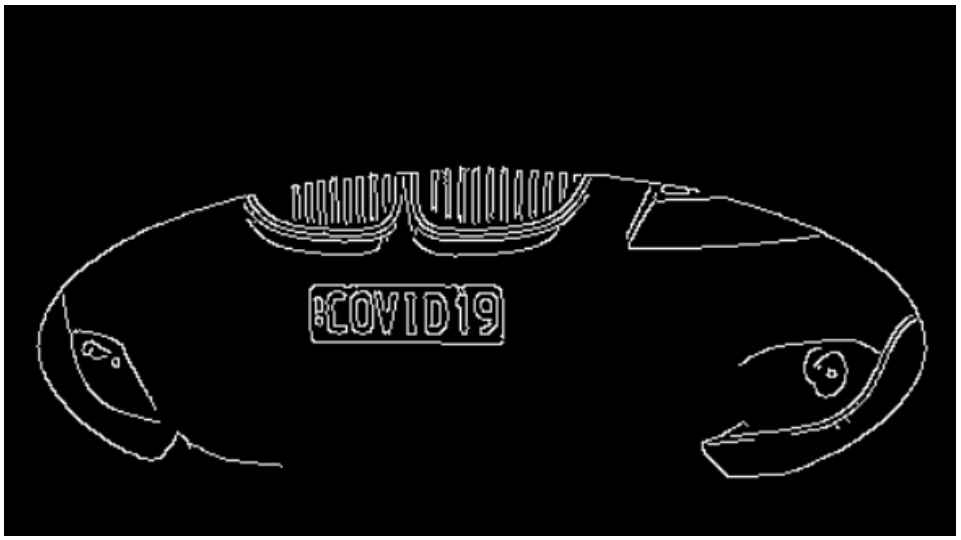


Figura 3.9: *Imagen con el detector de bordes Canny. Elaboración propia.*

Una vez aplicado el detector de bordes, se va a aplicar una operación de Open, es decir, una dilatación seguida de una erosión.

En este caso se ha personalizado la operación de Open para que se realicen varias iteraciones de dilatación, seguidas de una erosión. De esta manera se logra cerrar líneas que no sean continuas, además de aumentar el grosor de todos los bordes. Con esto se logra obtener muchas líneas rectas, que convierten muchas de las regiones cerradas en figuras poligonales.

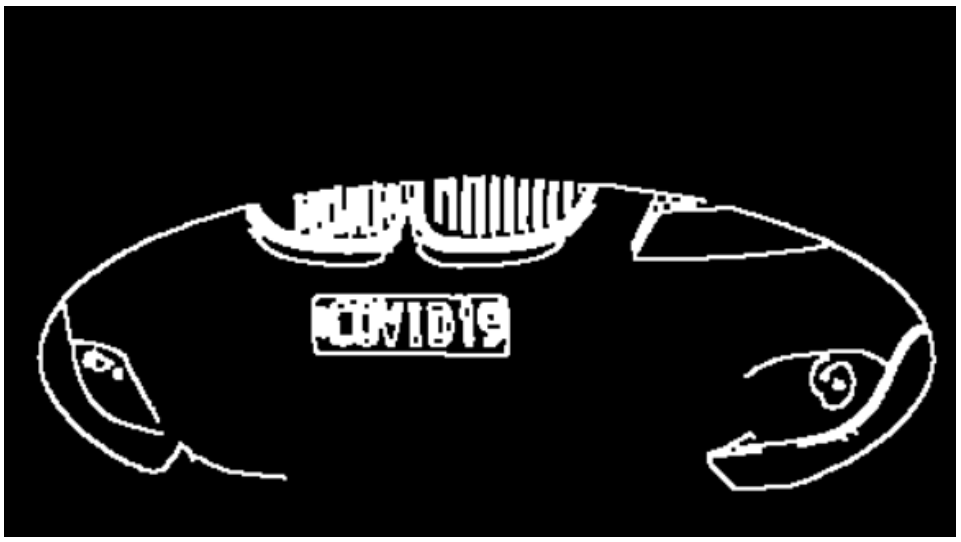


Figura 3.10: *Imagen con el detector de bordes Canny con Open. Elaboración propia.*

Con esto se ha conseguido cerrar el recinto rectangular definido por la matrícula. Gracias a ello, se puede aplicar un algoritmo de OpenCV para localizar estos contornos cerrados, simplificarlos hasta polígonos empleando el comando “cv2.approxPolyDP” y clasificarlos en función del área de cada contorno (en píxeles cuadrados) en orden descendente.

Una vez se dispone de esta lista con los contornos, se realizan varias iteraciones donde se comprueban las características físicas de estos contornos.

Las características a tener en cuenta son las siguientes:

- El número de vértices tiene que ser igual a cuatro (para ubicar el rectángulo de la matrícula).
- Las longitudes de los lados de dicho cuadrilátero deben ser similares dos a dos (los lados horizontales similares y los lados verticales similares).
- Cada segmento del cuadrilátero debe cumplir unos requisitos de longitud mínimos en píxeles.
- La relación de aspecto entre los lados verticales y horizontales debe ser de entre 3 y 7.

Aplicando estos parámetros para cada detección de polígonos, idealmente se debe de obtener un polígono de cuatro vértices que contiene la matrícula.

Si representamos los puntos obtenidos en un lienzo negro, obtenemos la máscara representada en la figura 3.11.

Figura 3.11: *Máscara de la detección de la matrícula, elaboración propia.*

Si aplicamos esta máscara sobre la imagen con la transformación afín, aplicando los mismos principios que en el paso en el que aplicamos la máscara de preprocesado, se obtiene el siguiente resultado, mostrado en la figura 3.12.

Figura 3.12: *Máscara con la detección de la matrícula aplicada a la imagen original. Elaboración propia.*

Por último, se recorta la región que contiene la detección en la imagen, obteniendo de esta manera la matrícula, tal y como se muestra en la figura 3.13.

Figura 3.13: *Matrícula detectada y recortada mediante métodos tradicionales. Elaboración propia.*

3.2.2. Algoritmo de detección de matrículas basado en reconocimiento de patrones.

En segundo lugar se va a desarrollar el algoritmo de detección basado en reconocimiento de patrones. Esto significa que se van a emplear una gran cantidad de imágenes recortadas de matrículas para buscar una coincidencia en la imagen de entrada.

Se va a hacer uso de las librerías de OpenCV y Numpy para implementar las funciones de “template matching”, además de emplearse para el preprocesado y la segmentación de la imagen.

Antes de comenzar con el desarrollo, se va a realizar una breve predicción de los resultados que se esperan obtener con este algoritmo.

En cuanto a la velocidad, se espera que el algoritmo sea considerablemente lento. Esto es debido a que se va a realizar un reconocimiento de patrones sobre la imagen completa, donde se van a buscar más de 90 matrículas diferentes.

Esto supone analizar una enorme cantidad de píxeles para realizar una única detección, lo que debería conllevar tiempos de procesado muy elevados.

Por otro lado, se espera que este algoritmo no sea particularmente preciso. Esta suposición se basa en que un mínimo cambio en las dimensiones, iluminación, color u orientación de la matrícula en la imagen puede causar un fallo en la detección, provocando muchos falsos negativos.

Aun así, el algoritmo debería de ser muy sencillo de implementar una vez se obtenga el banco de imágenes de las matrículas que se van a buscar.

Búsqueda del banco de imágenes.

El primer paso consiste en encontrar el banco de imágenes de matrículas que se van a buscar sobre la imagen de entrada.

Todas las matrículas que se van a emplear en este algoritmo se han obtenido de recursos online gracias a páginas web de coleccionistas de matrículas. Algunas de estas páginas web que recopilan matrículas son “matriculasdelmundo.com”, “worldlicenseplates.com” y “plate-planet.com”.

Como se ha visto anteriormente en el apartado 2.4, existen una gran cantidad de matrículas distintas, incluso en un mismo país. Este es el caso de las matrículas de Estados Unidos, Canadá y Australia. Entre estos tres países se suman un total de 70 matrículas diferentes en circulación.

Algunas de estas matrículas estadounidenses, canadienses y australianas se muestran a continuación.

Figura 3.14: *Matrículas europeas y neozelandesas. De worldlicenseplates.com y matriculasdelmundo.com.*

En cambio, en otros países o regiones como Europa y Nueva Zelanda, se ha definido un estándar de matrícula que se debe cumplir en todos los vehículos del territorio, ya sea en tamaño, tipografía o color. En la figura 3.15 se muestran algunas de estas matrículas.

Figura 3.15: *Matrículas europeas y neozelandesas. De worldlicenseplates.com y matriculasdelmundo.com.*

En total se han recogido 91 matrículas diferentes con las que se va a realizar la detección de patrones.

Desarrollo del segmento del algoritmo.

Para la explicación del algoritmo se va a hacer uso de la imagen mostrada anteriormente en la figura anterior.

preprocesado.

El segundo paso consiste en realizar un preprocesado de la imagen de entrada. Este preprocesado consiste en dos etapas diferenciadas: el preprocesamiento de la imagen de entrada y el preprocesamiento de las imágenes de las matrículas que se van a buscar sobre la imagen de entrada.

Para la imagen de entrada:

- Conversión de la imagen a grises.
- Transformación afín de la imagen.
- Aplicación de filtros de ruido.

Para las imágenes con las que realizar el reconocimiento de patrones:

- Conversión de las 91 matrículas a grises.
- Ajuste del tamaño de las imágenes de las matrículas.

A continuación se va a realizar una explicación detallada de cada una de ellas, así como de las decisiones de ingeniería tomadas en cada momento.

En primer lugar, se carga la imagen de entrada y se convierte a escala de grises mediante el uso del comando “cv2.cvtColor()”. Con esto se logra reducir el tamaño de la imagen a una tercera parte eliminando los canales de color, lo que mejora la velocidad de procesamiento del algoritmo.

Además, mediante este proceso se elimina la información que aporta el color, que en el caso de los métodos de detección de patrones no es demasiado relevante. El resultado se muestra en la figura 3.16.

Figura 3.16: *Imagen de entrada convertida a grises. Elaboración propia.*

En segundo lugar, se va a realizar una transformación afín sobre la imagen de entrada. Tal y como se ha explicado en el apartado 3.2.1, lo correcto es aplicar una transformación proyectiva para convertir la matrícula a una forma rectangular perfecta.

Sin embargo, en la realidad no se puede aplicar una transformación proyectiva sobre una imagen completa, dado que se utiliza exclusivamente para rescatar una región previamente detectada de una imagen.

Por este motivo, se ha tomado la decisión de aplicar una transformación afín sobre la imagen completa. Esto ajusta la matrícula hasta una posición horizontal.

Sobre la imagen resultante se aplican filtros para eliminar el ruido. Ahora la imagen es lo suficientemente buena como para poder realizar un reconocimiento de patrones sobre la misma. En la figura 3.17 se muestra la imagen tras el preprocesado.

Figura 3.17: *Imagen de entrada tras el filtrado y la transformación afín. Elaboración propia.*

Por otro lado, se procesan las imágenes con las que se va a realizar el reconocimiento de patrones. Este preprocesado comienza convirtiendo cada una de las imágenes de matrículas a escala de grises, como se muestra en la figura 3.18.

Figura 3.18: *Matrículas convertidas a escala de grises. Elaboración propia.*

El último paso consiste en escalar las matrículas hasta el tamaño relativo que deberían tener en la imagen que se va a analizar.

Dado que se conocen las características de las instalaciones físicas como se ha visto en el apartado 3.1.3, este tamaño objetivo se puede calcular haciendo uso de algunas imágenes de ejemplo obtenidas en cada instalación física particular.

segmentación.

El siguiente paso consiste en segmentar la matrícula del resto de la imagen empleando reconocimiento de patrones. Para ello, se va a hacer uso del comando “cv2.matchTemplate(imagen 1, imagen 2, cv2. TM _ CCOEFF _ NORMED)”.

Este comando busca en la imagen de entrada una imagen plantilla dada, y ofrece como resultado un mapa de puntos a los que se ha asignado un valor de confianza dado. El mapa de puntos se puede representar como una imagen, mostrada en la figura3.19.

Figura 3.19: *Mapa de coincidencias resultante de la detección basada en reconocimiento de patrones. Elaboración propia.*

A continuación se hace uso del comando de Numpy “numpy.where()” para encontrar todos los puntos de la imagen anterior que superen un umbral de confianza previamente establecido.

En el caso de este algoritmo se han obtenido los mejores resultados con un umbral de 0.45, aunque los resultados han sido generalmente malos, como se explica más adelante.

A continuación, se debe repetir este proceso de segmentación para cada una de las 91 imágenes de matrículas. El resultado de este bucle es un array con las posiciones de cada detección y su confianza asociada.

El último paso consiste en representar la bounding box correspondiente al punto con mayor índice de confianza, como se muestra en la figura 3.20.

Figura 3.20: *Bounding box con la detección de la matrícula en la imagen de ejemplo mediante reconocimiento de patrones. Elaboración propia.*

Para obtener una imagen de la matrícula detectada, se recorta la región contenida por esta bounding box. El resultado final se muestra en la figura 3.21.

Figura 3.21: *Recorte de la matrícula en la imagen de ejemplo obtenida mediante reconocimiento de patrones. Elaboración propia.*

3.2.3. Algoritmo de detección de matrículas basado en aprendizaje automático de modelos de detección.

En tercer y último lugar, se va a desarrollar el algoritmo de detección basado en inteligencia artificial, en concreto basado en YOLOv8. Esto significa que se va a utilizar un modelo de predicción basado en la arquitectura de red neuronal de YOLO para realizar las detecciones de matrículas en la imagen dada.

Se va a hacer uso de las librerías de Ultralytics y OpenCV para el desarrollo del algoritmo.

Antes de comenzar con el desarrollo, se va a hacer una breve predicción de los resultados de este algoritmo.

Se puede asumir que los algoritmos basados en inteligencia artificial son muy precisos en las detecciones realizadas, y son extremadamente flexibles ante cambios las condiciones ambientales del entorno del sistema ANPR.

Los inconvenientes de este método deberían ser el tiempo de procesado, la necesidad de un buen conjunto de imágenes de entrenamiento etiquetadas y la potencia de software necesaria para realizar el entrenamiento de la red neuronal.

Búsqueda del banco de imágenes.

El primer paso para desarrollar el algoritmo es buscar un banco de imágenes para poder entrenar la red neuronal. Para ello, se han buscado en Roboflow y Dataset Ninja proyectos de bases de datos de matrículas.

Se han encontrado datasets de matrículas de Europa, EEUU, Emiratos Árabes, Rusia, India, UK y Sudamérica. En estos bancos de imágenes se han incluido imágenes de camiones, motocicletas y furgonetas, con el fin de obtener una predicción más robusta en una gran variedad de vehículos y formas de matrícula.

Una vez reunidas las imágenes, se han analizado una por una las casi 5.500 imágenes de matrículas, eliminando aquellas que no se creían convenientes y etiquetando todas las matrículas que no se encontraban etiquetadas.

Aunque el lector de matrículas que se plantea en este proyecto busca leer únicamente la matrícula frontal, se ha tomado la decisión de incluir las imágenes de las matrículas traseras de vehículos para incrementar el tamaño del banco de datos. Se espera que esta decisión no afecte a la capacidad de detección de la red neuronal.

Una vez etiquetadas y analizadas manualmente todas las imágenes, se procede a mezclar todos los distintos bancos de imágenes de datos hasta crear un dataset variado de casi 5.500 imágenes.

Es importante añadir imágenes donde no hay matrículas para enseñar a la red neuronal que no siempre existen matrículas en las imágenes.

A continuación se debe dividir este dataset en los conjuntos de entrenamiento, validación y comprobación.

Para determinar la proporción en la que se debe dividir cada conjunto, se han tomado algunas recomendaciones de la documentación de Roboflow.

En estos documentos, se recomienda que para un set de más de 3.000 imágenes empleado en el entrenamiento de un modelo de detección de una única clase de objeto, el reparto más adecuado es de 70% para entrenamiento, 20% para validación y 10% para comprobación. Se pueden realizar pequeños ajustes del 5% en caso de querer buscar optimizar el entrenamiento.

Por tanto, se ajustó el reparto en el conjunto en la página de Roboflow, como se muestra en la figura 3.22.

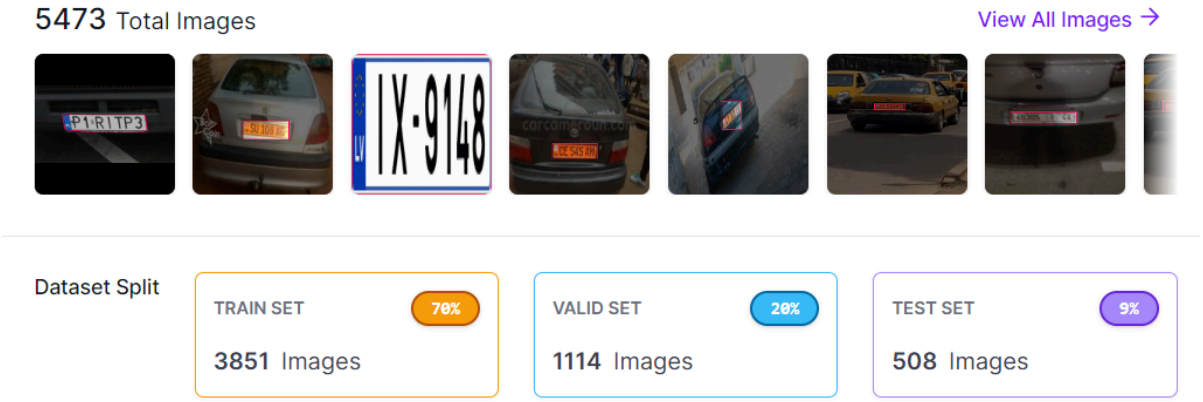


Figura 3.22: *Banco de imágenes de matrículas repartido en entrenamiento, validación y comprobación. Elaboración propia en Roboflow.com.*

Entrenamiento del modelo de YOLOv8.

En un primer lugar se intentó entrenar el modelo de la red neuronal en mi propio hardware, en una tarjeta gráfica GEForce RTX 3060 Laptop, a través del entorno de PyCharm. Para ello, es necesario instalar el software de CUDA, una aplicación de Nvidia que permite emplear la GPU para el entrenamiento de modelos de redes neuronales.

Una vez se dispone de CUDA, se debe instalar el paquete de dependencias de ultralytics. En este caso, ya que estamos empleando Python 3, se utiliza la aplicación de “pip” para instalar los paquetes de “ultralytics” y de “roboflow”.

Una vez instaladas las dependencias, se debe descargar el dataset desde roboflow hasta el directorio del proyecto y emplear los comandos de YOLO para comenzar el entrenamiento. [35].

Por algún motivo que todavía se desconoce, el entrenamiento de la red neuronal fallaba nada más comenzar. Tras una larga investigación, se ha determinado que este fallo se puede deber a una incompatibilidad de CUDA con la última versión de los controladores de Nvidia de mi GPU, o a una incompatibilidad de la GPU con las últimas versiones de TorchVision, pero al final no se ha podido resolver este problema.

Por este motivo, se tomó la decisión de tratar de entrenar la red neuronal en Google Colab. Esta plataforma basada en Python 3 y CLI permite acceso a una GPU del backend de Google que dispone del software de CUDA incorporado, lo que permite realizar el entrenamiento de manera muy eficiente.

Por tanto, se volvió a instalar las dependencias de Ultralytics y Roboflow en el entorno virtual de Google Colab, y tras descargar el banco de imágenes se pudo realizar el entrenamiento satisfactoriamente.

Se tomó la decisión de entrenar el modelo comenzando desde un modelo vacío, para optimizar los tiempos de inferencia de la red neuronal. Además, se ha decidido entrenar el modelo durante 140 épocas o iteraciones, para asegurar un entrenamiento extenso y correcto. El tiempo de entrenamiento fue de aproximadamente 6 horas.

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances
139/140	2.48G	0.4938	0.2567	0.9912	12
	Class	Images	Instances	Box(P	R
	all	1114	1124	0.903	0.939
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances
140/140	2.48G	0.4875	0.2557	0.9886	11
	Class	Images	Instances	Box(P	R
	all	1114	1124	0.903	0.94

Figura 3.23: *Resultado de la consola de Google Colab tras finalizar el entrenamiento del modelo de detección. Elaboración propia en Google Colab.*

En caso de que el modelo sufriera un “overfitting” o sobreajuste sobre el modelo de entrenamiento, YOLOv8 automáticamente detecta este evento y devuelve como resultado el modelo que mejor resultado ha dado a lo largo del entrenamiento.

El resultado de este entrenamiento es un archivo llamado “DetMatriculasV1.pt” que contiene el modelo de detección de matrículas.

Gracias a emplear el argumento “plots=True”, al terminar el entrenamiento se generan una serie de gráficas y esquemas que permiten evaluar el rendimiento del entrenamiento.

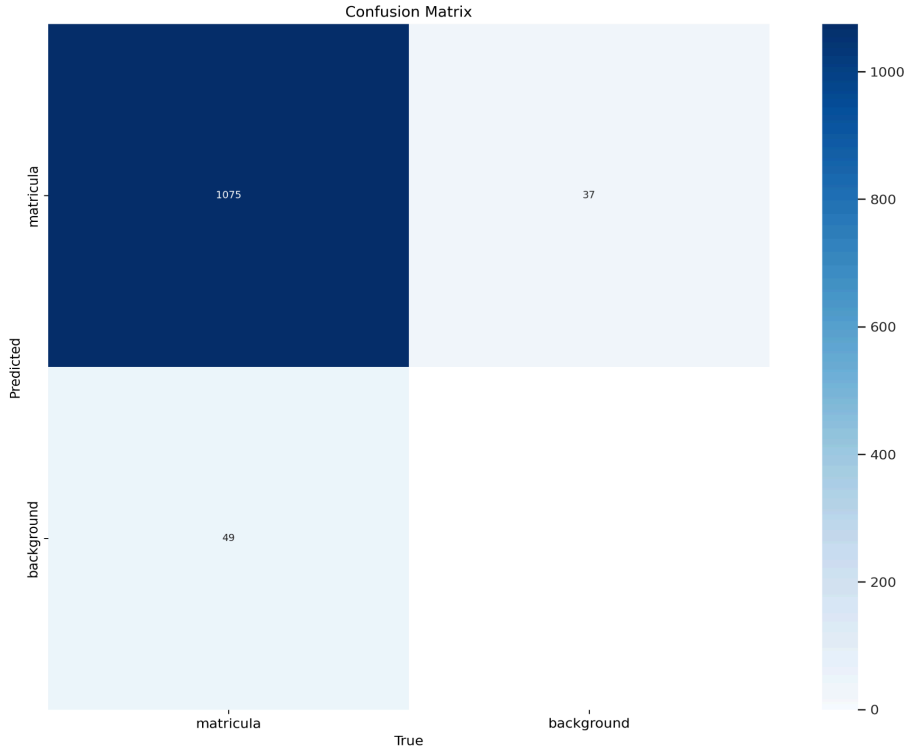


Figura 3.24: *Matriz de confusión de los resultados del entrenamiento del modelo de detección. De Ultralytics YOLOv8.*

En esta matriz de confusión mostrada en la figura 3.24 se muestran los resultados en la predicción de la red neuronal sobre el set de validación de 1.114 imágenes.

Esto significa que se tiene un acierto en la predicción en el 96% de los casos, se genera una falsa detección en el 2% de los casos y no se detectan matrículas cuando las hay el 2% de los casos.

En la figura 3.25 se pueden ver distintas gráficas que muestran cómo mejoran los resultados en las detecciones y predicciones del modelo a lo largo de las épocas de entrenamiento.

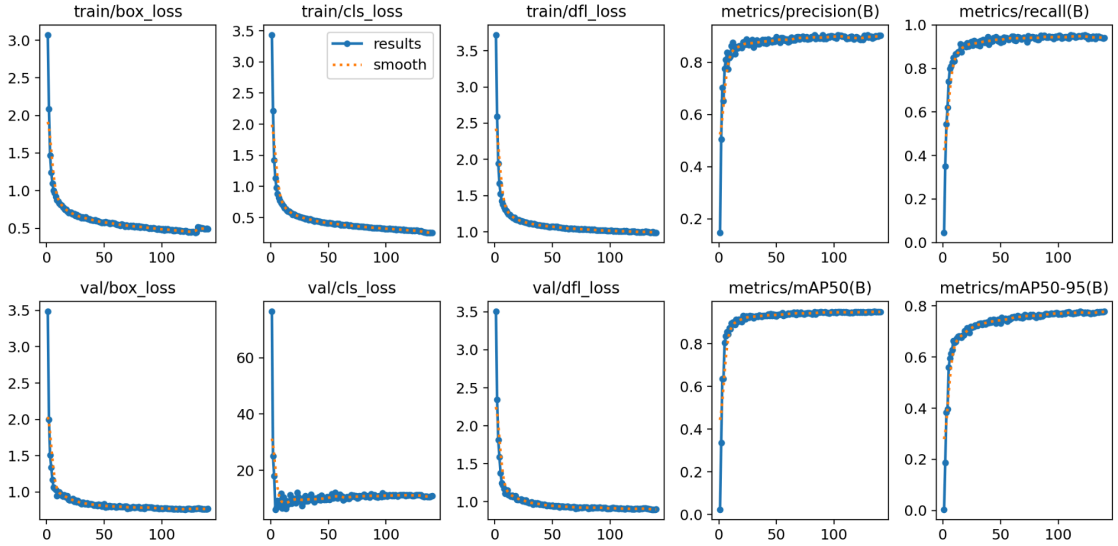


Figura 3.25: *Gráficas con los resultados del entrenamiento del modelo. De Ultralytics YOLOv8.*

También se puede observar cómo se aplanan las curvas de mejora, indicando que se ha establecido una cantidad de épocas de entrenamiento adecuada y que se ha hecho uso de un banco de imágenes amplio, robusto y consistente.

Desarrollo del segmento del algoritmo.

Para explicar el desarrollo del algoritmo de forma visual, se va a hacer uso de la imagen de ejemplo mostrada anteriormente en la figura anterior.

El primer paso en el desarrollo del algoritmo consiste en importar todas las librerías necesarias: “cv2”, “ultralytics” y “numpy”. Después, se lee la imagen en la que se quiere realizar la detección y se aplica el modelo de predicción entrenado con YOLO v8.

Tras aplicar el modelo, se obtiene una estructura con todas las características de los objetos detectados en la imagen, incluidas las coordenadas de la bounding box y las coordenadas de los centros de los objetos detectados.

Generalmente, el siguiente paso es hacer uso de un algoritmo NMS para eliminar las múltiples detecciones de un mismo elemento en la imagen. En YOLO v8 no es necesario hacer uso de estos algoritmos, ya que se aplican automáticamente durante la detección.

En este caso, nos importan los datos de la bounding box, para poder extraer la región útil de la imagen. Si se utiliza el comando resultados.plot() se obtiene la imagen mostrada en la figura 3.26, en la cual se muestra la bounding box de la detección sobre la imagen original.

Figura 3.26: *Imagen con una detección de matrícula mediante YOLOv8. Elaboración propia.*

En la figura anterior, se puede comprobar cómo el modelo de detección asigna una confianza del 86% a su predicción.

Este valor de confianza en la predicción se considera acertado a partir de un umbral arbitrario establecido por el usuario, en este caso un umbral del 25% de confianza.

Esta confianza en la predicción tiene una correlación directa con la precisión en la detección, de manera que una predicción con un 25% de confianza tiene un 80% de posibilidades de contener una matrícula.

Esta relación entre precisión y confianza sigue la función mostrada en la figura 3.27.

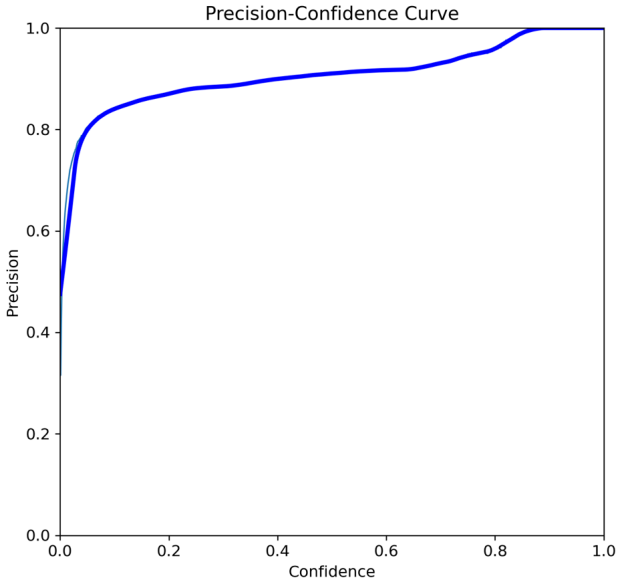


Figura 3.27: *Gráfica con la relación Precisión-Confianza en la detección. Elaboración propia empleando YOLOv8 de ultralytics.*

El último paso consiste en recortar la zona de la detección delimitada por la bounding box, obteniendo de esta manera una imagen que contiene únicamente la matrícula, tal y como se muestra en la figura 3.28.

Figura 3.28: *Imagen con la matrícula recortada, detectada mediante YOLOv8. Elaboración propia.*

3.3. Algoritmos de reconocimiento de caracteres.

Una vez determinada la ubicación de la matrícula, únicamente resta leer los datos de la misma. Para ello, se va a emplear un sistema de reconocimiento automático de caracteres, comúnmente conocido como OCR (Optical Character Recognition).

El resultado de este segmento del programa será una cadena de texto con la información obtenida de la matrícula.

En todos los casos se va a emplear la librería de OpenCV para el preprocesado, tratamiento y lectura de las imágenes. Se va a desarrollar el código en Python 3, en el entorno de PyCharm Community Edition.

A continuación se van a desarrollar cuatro algoritmos para el reconocimiento de los caracteres:

- Algoritmo OCR basado en reconocimiento de patrones.
- Algoritmo OCR basado en YOLO v8.
- 2 algoritmos basados en soluciones OCR de código libre.

Para realizar la explicación de los algoritmos se hará uso de la imagen obtenida como resultado en los ejemplos explicativos de los apartados 3.2.1., 3.2.2. y 3.2.3. Esta imagen se muestra en la figura 3.29.

Figura 3.29: *Imagen con la matrícula preprocesada y recortada. Elaboración propia.*

3.3.1. Algoritmo OCR basado en reconocimiento de patrones.

En primer lugar, se va a desarrollar el algoritmo OCR basado en reconocimiento de patrones. Esto significa que se van a obtener las fuentes de texto empleadas en cada país, para posteriormente aislar cada glifo en una imagen independiente y emplear reconocimiento de patrones para buscar cada glifo en la imagen de la matrícula obtenida en los algoritmos desarrollados en el apartado 3.2.

Se va a hacer uso de las funciones de “template matching” de la librería de OpenCV para realizar la detección de los glifos contenidos en la imagen de la matrícula.

Antes de comenzar el desarrollo, se va a hacer una breve predicción de los resultados que se espera obtener con este algoritmo.

Como ya se ha visto en el apartado 3.2.2 sobre el sistema de detección basado en reconocimiento de patrones, no se obtuvo un buen resultado debido a lo poco controlado que se encuentra el entorno en el algoritmo de detección, así como al gran tamaño de las imágenes donde realizar el reconocimiento de patrones.

Sin embargo, las imágenes sobre las que se va a trabajar en este apartado han sido recortadas y transformadas geométricamente, de manera que son mucho más controladas y limpias de anomalías y ruido.

Haciendo uso de algoritmos de preprocesado, se espera obtener unas imágenes muy buenas sobre las que trabajar. Esto debería mejorar en gran medida el resultado obtenido al aplicar el algoritmo de reconocimiento de patrones.

Además, dado que las imágenes de la matrícula y los glifos son mucho más pequeñas que las del algoritmo de detección, se puede deducir que los tiempos de procesado serán mucho más rápidos que en el apartado 3.2.2, aunque se tengan que procesar el mismo número de glifos o más.

En gran medida, se van a seguir pasos muy similares a los relatados en el desarrollo del algoritmo de detección de matrículas basado en reconocimiento de patrones visto en el apartado 3.2.2.

Búsqueda de las fuentes de texto.

Puesto que se busca leer matrículas de Europa, Estados Unidos, Canadá, Australia y Nueva Zelanda, el primer paso consiste en obtener las fuentes de texto de las matrículas de estos países.

Como ya se ha investigado y desarrollado en el apartado 2.4, las fuentes que necesitamos son: “Din 1451” para Europa, “Driver Gothic” para Estados Unidos y Canadá, “SAA Series B” para Australia y “Licenz” para Nueva Zelanda.

Estas fuentes de texto se muestran en las figuras X.

Figura 3.30: *Fuente de texto “Din 1451”. De Ludwig Goller, 1936, en leewardpro.com.*

Figura 3.31: *Fuente de texto “Driver Gothic”. De Park Griffin, 2008, en leewardpro.com.*

Figura 3.32: *Fuente de texto “SAA Series B”. De URW Staff, 1980, en leewardpro.com.*

Figura 3.33: *Fuente de texto “Licenz”. De David Buck, 2006, en leewardpro.com.*

Una vez obtenidas las fuentes, se hace uso de la página web “Free Fonts Download” para descargar cada uno de los glifos de estas fuentes. [36].

Como se puede ver en las figuras anteriores, la gran mayoría de los glifos son muy similares en todas las fuentes de texto. Las claras excepciones son los caracteres D, I, O, Q y W, así como los números 0, 1, 4, 6 y 9.

Además, existen otras fuentes consideradas obsoletas pero que todavía se encuentran en circulación en coches antiguos, particularmente comunes en algunas regiones europeas, estadounidenses y australianas.

Por tanto, para optimizar el algoritmo y no realizar reconocimiento de patrones con cada uno de los 36 glifos de cada fuente, se ha decidido crear una pequeña base de datos de 48 glifos donde se han añadido todas las variantes de los caracteres y números que presentan variaciones notables.

Desarrollo del segmento del algoritmo.

En este algoritmo se va a trabajar a modo de ejemplo con la imagen mostrada en la figura anterior.

Preprocesado.

El primer paso para el desarrollo del algoritmo es el preprocesado de la imagen de entrada de la matrícula. Para

- Escalado de la imagen.
- Conversión de la imagen a grises.
- Aplicación de filtros de ruido y filtros suavizantes.

A continuación se explican detalladamente cada uno de los pasos, así como el razonamiento detrás de cada paso y las decisiones de ingeniería tomadas en cada momento.

En primer lugar, se carga la imagen de la matrícula al programa. Esta imagen es de unas dimensiones desconocidas, ya que dependen de cada sistema de vídeo y algoritmos de detección y recorte empleados.

Por este motivo se ha tomado la decisión de escalar la imagen de entrada hasta que los glifos de la matrícula sean del mismo tamaño que las imágenes de los glifos importadas de las fuentes de texto.

Este proceso requiere conocer las características del sistema ANPR donde se va a trabajar, de manera que se pueda configurar el escalado de la imagen. En este caso, se va a ajustar manualmente este parámetro de escala para adaptarse a cada ejemplo.

En segundo lugar, se convierte la imagen en color a una imagen en escala de grises mediante el uso del comando “cv2.cvtColor”, seguido de la aplicación de un filtro de suavizado para homogeneizar la imagen escalada mediante el comando “cv2.GaussianBlur”. El resultado se muestra en la figura 3.34.

Figura 3.34: *Imagen de la matrícula convertida a gris y filtrada. Elaboración propia.*

Por otro lado, se han de cargar todas las imágenes que contienen los glifos de la fuente de texto personalizada.

Para ello, se leen los nombres de todas las imágenes contenidas en la carpeta de la fuente de texto personalizada, para posteriormente introducir todas estas imágenes en una lista de Python.

De la misma manera que con la imagen de la matrícula, se hace uso de varios bucles para convertir cada imagen a grises y posteriormente binarizar cada una de ellas. Unos pocos resultados se muestran en la figura 3.35 a modo de ejemplo ilustrativo.

Figura 3.35: *Glifos binarizados. Elaboración propia.*

Segmentación.

El siguiente paso consiste en aplicar un algoritmo de binarización adaptativo llamado algoritmo de Otsu.

El algoritmo de Otsu convierte la imagen en escala de grises en una imagen con únicamente dos colores, blanco y negro, tal y como se muestra en la figura 3.36.

Figura 3.36: *Imagen de la matrícula binarizada. Elaboración propia.*

Como se puede comprobar, los glifos de la matrícula se encuentran completamente separados en tamaño y color del resto de elementos de la imagen.

En este momento se puede emplear un algoritmo de “blob” para contar el número de glifos que superen cierto umbral de cantidad de píxeles, pero se ha considerado este paso redundante dado que el siguiente paso consiste en aplicar el algoritmo de reconocimiento de patrones, en el cual se cuentan los elementos de la matrícula de forma indirecta y automática.

Parametrización.

Por último, únicamente resta aplicar el reconocimiento de patrones sobre la imagen procesada de la matrícula, en la que se debe realizar una búsqueda independiente para cada glifo.

La búsqueda consiste en guardar aquellas detecciones que superen un umbral de confianza predeterminado, en este caso de 0.7. En caso de conseguir una detección, lo más probable es que se encuentren una gran cantidad de coincidencias que superen el umbral.

Un ejemplo de este comportamiento se comprueba en la figura 3.37, donde se ha aplicado el reconocimiento de patrones buscando la letra “C”. En este ejemplo se dibujan las bounding boxes de alrededor de 30 puntos que superan el umbral de certeza en la detección. Esta múltiple detección es denotada por el grosor de la bounding box, que debería de ser de dos píxeles, en lugar de 20.

Figura 3.37: *Imagen de la matrícula con múltiples detecciones. Elaboración propia.*

Esto supone un gran problema, ya que se necesita obtener una única detección por cada glifo en la imagen. La solución consiste en crear y aplicar un algoritmo NMS (Non Maximum Suppression).

Para esta aplicación se ha diseñado un algoritmo NMS simplificado, donde se eliminan las detecciones que se encuentran muy cerca de otras. Esta distancia en píxeles es fija y conocida, dado que se conocen las dimensiones de la matrícula (tras haberla escalado) y por tanto las dimensiones de los glifos que contiene. En este caso esta distancia es de 50 píxeles horizontalmente y 100 píxeles verticalmente.

Tras aplicar este algoritmo NMS personalizado, se eliminan las detecciones múltiples que se encuentren muy cerca unas de otras. El resultado se muestra en la figura 3.38.

Figura 3.38: *Imagen de la matrícula con las múltiples detecciones eliminadas. Elaboración propia.*

El siguiente paso consiste en aplicar el reconocimiento de patrones para todos los glifos de la fuente personalizada.

Para ello, se introduce todo el código anterior en un bucle. Tras ajustar el valor del umbral de confianza para evitar los falsos positivos, se obtiene el siguiente resultado, mostrado en la figura 3.39.

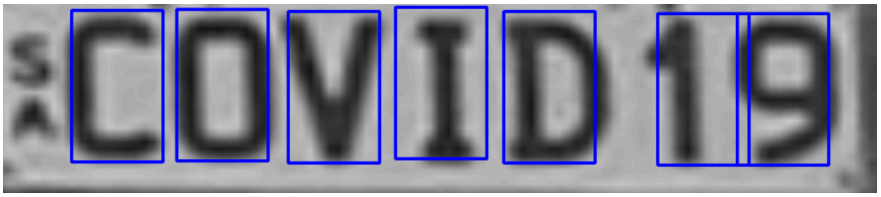


Figura 3.39: *Imagen de la matrícula con las detecciones de cada glifo. Elaboración propia.*

El último paso consiste en asignar a cada detección el valor del glifo con el que se ha detectado, y ordenar las detecciones de arriba a abajo y de izquierda a derecha.

Esto se lleva a cabo mediante el uso de un algoritmo que ordena los puntos y redacta la cadena de texto con el contenido de la imagen, la cual se muestra por pantalla. El resultado se muestra en la figura 3.40.

```
Número de glifos cargados: 48

Matrícula:  C0VID19
```

Figura 3.40: *Resultado del algoritmo OCR mediante pattern matching, en la consola de Python. Elaboración propia.*

3.3.2. Algoritmo OCR basado en aprendizaje automático de modelos de detección.

En segundo lugar, se va a desarrollar el algoritmo OCR basado en YOLOv8. Esto significa que se va a utilizar un modelo de predicción basado en redes neuronales para realizar las detecciones de cada glifo en las imágenes obtenidas por los algoritmos de detección desarrollados en el apartado 3.2.

Se va a hacer uso de la librería de OpenCV para el procesado de los resultados, así como de las librerías de ultralytics para la detección mediante la red neuronal.

Antes de comenzar con el desarrollo, se va a hacer una breve predicción de los resultados que se espera obtener con este algoritmo. De la misma manera que con el algoritmo de detección basado en YOLO v8 desarrollado en el apartado 3.2.3, se espera un algoritmo muy flexible ante los cambios en las condiciones de la imagen de la matrícula.

Se espera que, mediante el uso de un banco de datos de imágenes variado, el algoritmo sea capaz de detectar los glifos de matrículas en imágenes de muy mala calidad, ya que al fin y al cabo el algoritmo se va a aplicar a imágenes muy pequeñas obtenidas de recortes de otras imágenes.

Por otro lado, se espera que la principal desventaja en el uso de este algoritmo sean los tiempos de procesado de las imágenes, que en sistemas basados en inteligencia artificial suelen ser altos, pero en el caso de un algoritmo OCR lo son aún más. Esto se debe a que se han de buscar 36 glifos diferentes en la imagen.

En gran medida, se van a seguir pasos muy similares a los relatados en el desarrollo del algoritmo de detección basado en YOLOv8.

Búsqueda del banco de imágenes.

Debido a que se han de detectar 36 clases de objetos (26 letras y 10 números) el banco de datos necesario para el entrenamiento de este algoritmo OCR es sustancialmente más complejo de etiquetar que el visto en el apartado 3.2.3.

Gracias a la herramienta Roboflow, se han encontrado varios proyectos de bancos de datos adecuados para este entrenamiento. Estas imágenes contienen matrículas en distintas orientaciones, de todo tipo de vehículos, nacionalidades y tipografías. Tras reunir el dataset de imágenes, se han retirado aquellas que se han juzgado poco prácticas o fuera de lugar para este proyecto.

Tras varias horas de verificar y clasificar el banco de datos, se ha obtenido un dataset de 5.000 imágenes de matrículas etiquetadas con números y letras.

El siguiente paso es realizar el reparto de los conjuntos de entrenamiento, validación y comprobación.

En este caso, según la documentación de roboflow, el reparto óptimo se encuentra en el rango de 65-75% de entrenamiento, 15-25% de validación y 5-15% de comprobación. Para este modelo en particular, se ha elegido un reparto de 75%, 20% y 5%, mostrado en la figura 3.41.

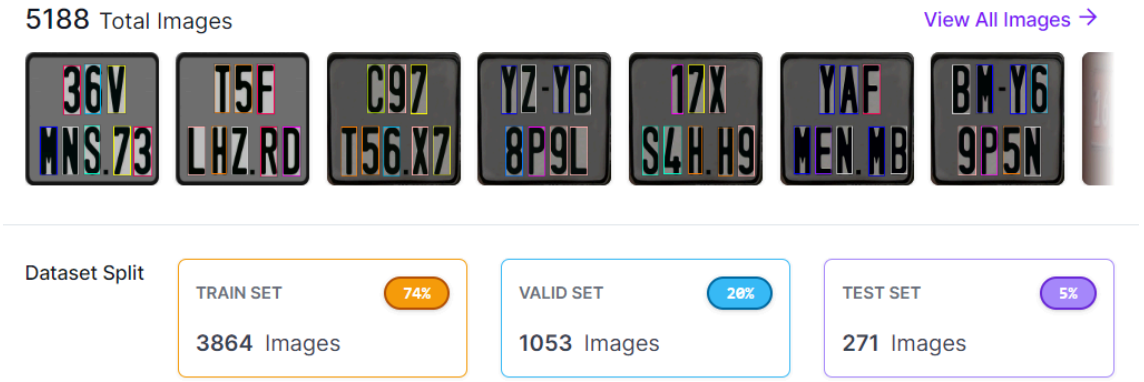


Figura 3.41: *Banco de imágenes de matrículas para OCR repartido en entrenamiento, validación y comprobación. Elaboración propia en Roboflow.com.*

Entrenamiento del modelo de YOLOv8.

El siguiente paso es comenzar con el entrenamiento del modelo de YOLOv8. Para ello, se ha recurrido al entorno de Google Colab, donde se han instalado todas las dependencias de ultralytics para el entrenamiento, así como las librerías de Roboflow para acceder al banco de datos de imágenes.

Una vez instaladas las dependencias y descargado el dataset en el entorno virtual, se procedió con el entrenamiento del modelo. Se tomó la decisión arbitraria de entrenar el modelo para 150 épocas, basándose en el entrenamiento para el modelo de detección de matrículas visto en el apartado 3.2.3.

Al comenzar con el entrenamiento y realizar unos cálculos preliminares sobre la duración del mismo, se llegó a la conclusión de que el entrenamiento tomaría entre 25 y 35 horas. Esto es debido al gran volumen de imágenes del banco de datos, además del gran número de clases que se pretende detectar.

Existen varios problemas con esta duración del entrenamiento. La primera es que el entorno de Google Colab permite una conexión continua durante un máximo de 10 horas. A partir de este momento, el entorno virtual se desconecta y todos los datos no descargados se pierden, incluido el progreso de entrenamiento del modelo.

Además, se rechaza la conexión durante las próximas 20 horas al entorno web para ese cuaderno de Google Colab.

Esto significa que para llevar a cabo el entrenamiento se debe dividir la carga computacional en, como mínimo, cuatro sesiones de 10 horas, repartidas a lo largo de una semana, recuperando en el último momento el modelo de la última época entrenada para continuar con el entrenamiento en la sesión siguiente.

Para optimizar este tiempo, se tomó la decisión de comprar el acceso al entorno de Google Colab Pro. Este entorno permite acceso a una GPU del backend de Google muy potente, llamada V100 GPU. Por lo que he podido averiguar, esta GPU es entre 6 y 10 veces más potente que la empleable en la versión sin pagos del entorno.

Tras probar a entrenar el modelo con el entorno de Google Colab Pro, se llegó a la conclusión de que en este caso el entrenamiento duraría aproximadamente 6 horas, un tiempo mucho más reducido que en el entorno sin pagos.

Tras cinco horas y media de ejecución del código, se completó satisfactoriamente el entrenamiento, como se muestra en la figura 3.42.

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances
149/150	2.53G	0.7924	0.5934	1.1	48
	Class	Images	Instances	Box(P	R
	all	1053	7539	0.974	0.934

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances
150/150	2.52G	0.7913	0.5944	1.105	33
	Class	Images	Instances	Box(P	R
	all	1053	7539	0.974	0.934

150 epochs completed in 5.679 hours.

Figura 3.42: *Resultado de la consola de Google Colab tras finalizar el entrenamiento del modelo OCR. Elaboración propia en Google Colab.*

Haciendo uso del argumento “plots=True” al ejecutar el modelo de la red, se han obtenido una serie de gráficas que permiten evaluar el rendimiento del entrenamiento y del modelo resultante.

La primera gráfica que se va a interpretar es la matriz de confusión del modelo que contiene todos los resultados en la predicción del set de validación.

Los datos que se comparan son la cantidad de veces que se confunde o acierta la predicción de un glifo en particular, ya sea con otras clases, con el fondo o consigo mismo.

La matriz de confusión se muestra en la figura 3.43.

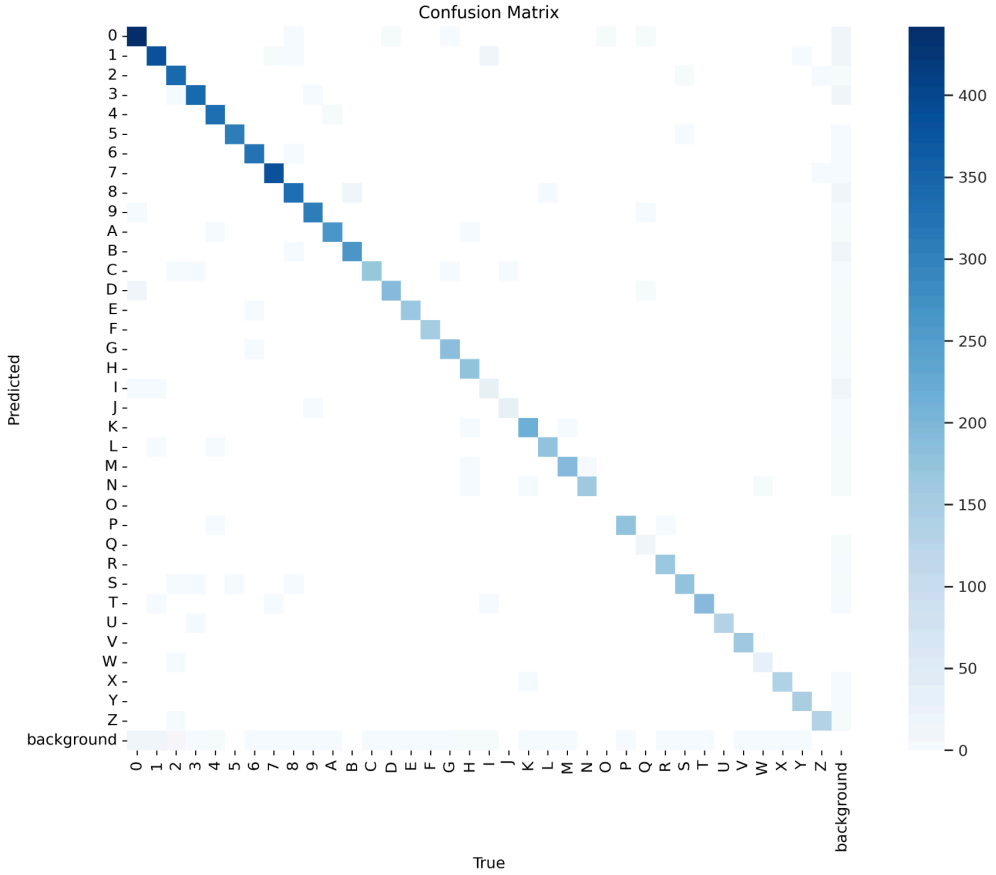


Figura 3.43: *Matriz de confusión de los resultados del entrenamiento del modelo OCR. De Ultralytics YOLOv8.*

Esta matriz de confusión ofrece varias revelaciones interesantes. La primera es que existen ciertos caracteres que son mucho menos comunes que los demás, entre ellos destacan los caracteres “I”, “J”, “O”, “Q” y “W”. Esto se debe a que en la inmensa mayoría de países, no se emplean dichos caracteres para evitar confusión con otros glifos similares, como por ejemplo la confusión entre el número 0 y las letras Q y O, la confusión entre el número 1 y las letras I y J, y la confusión entre las V y las W.

Por otro lado, se ve que los glifos más comunes son los números y las primeras letras del alfabeto. Tiene sentido que todos los números sean más comunes, ya que en comparación existen 10 dígitos y 26 letras. En cambio, las letras A y B son mucho más comunes que el resto.

Esto se puede deber a la estructura de las matrículas con letras. Si, por ejemplo, la primera matrícula española es 0000 AAA, las próximas 6 millones de matrículas serán de la forma XXXX AXX. Dado que son las primeras letras, es normal que en un glifo con base 26 los primeros glifos sean relativamente más comunes.

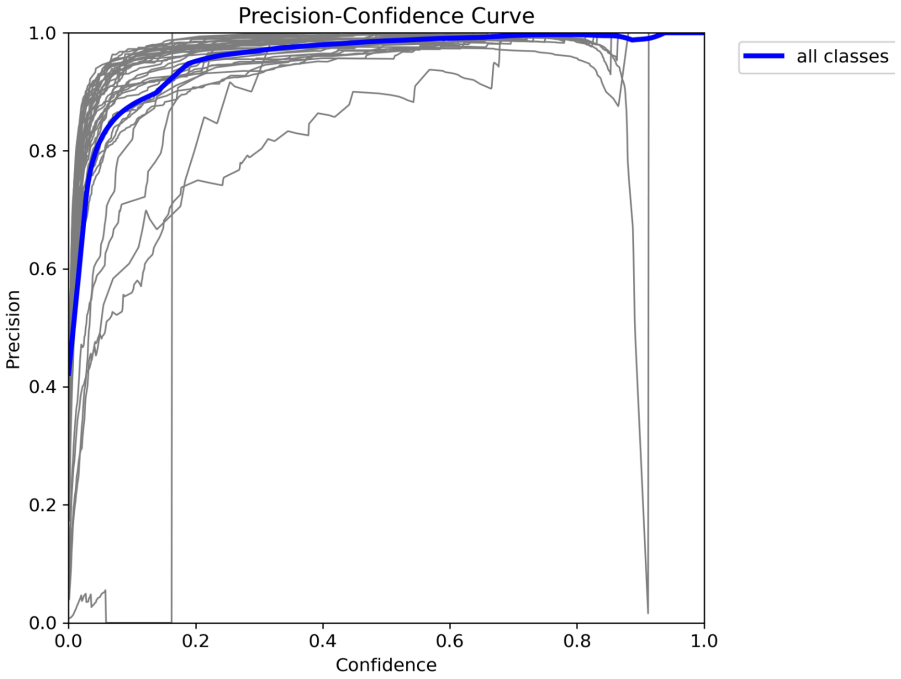


Figura 3.44: *Gráfica de precisión-confianza de los resultados del entrenamiento del modelo OCR. De Ultralytics YOLOv8.*

En la gráfica de precisión confianza mostrada en la figura anterior, se puede ver como existe una muy buena correlación entre la confianza en la predicción y la precisión real con la que se ha acertado la predicción.

Existen ciertas anomalías en aquellos glifos menos comunes, O, Q, I, J y W, dado que existen muy pocas matrículas con estos caracteres y son extremadamente difíciles de identificar, no solo para un modelo de YOLOv8, sino que resulta una tarea complicada incluso para los humanos.

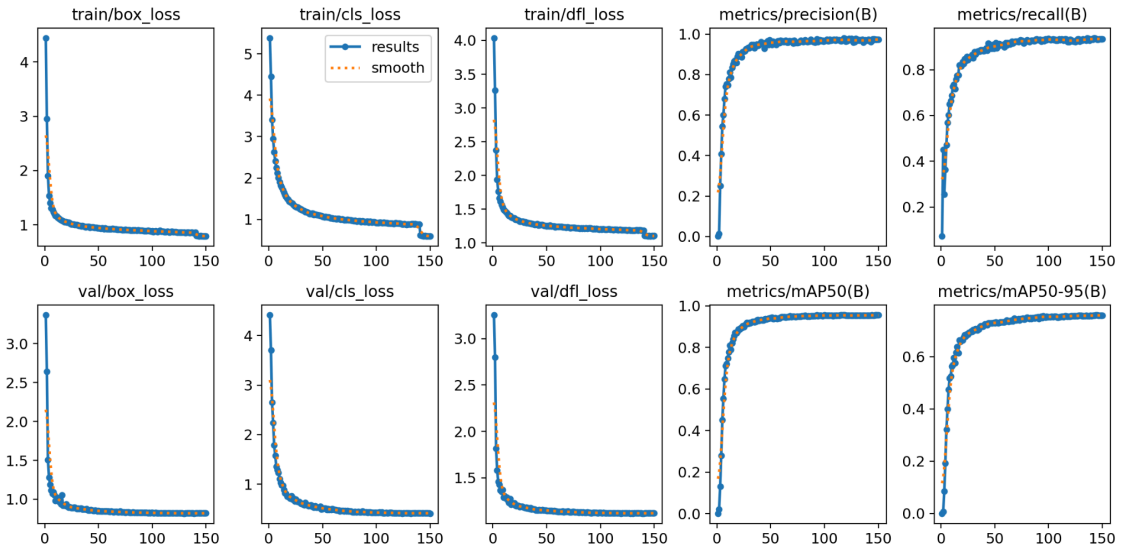


Figura 3.45: *Conjunto de gráficas de los resultados del entrenamiento del modelo OCR. De Ultralytics YOLOv8.*

En la figura anterior se puede ver como todos los parámetros del entrenamiento mejoran en gran medida a medida que avanzan las épocas de entrenamiento.

Se puede observar que a partir de la época 100, las mejoras en la eficiencia para cada iteración se vuelven ínfimas, a excepción de las últimas 10 iteraciones, donde se realiza una integración especial que mejora enormemente el rendimiento en la predicción.

Todo esto indica que se ha establecido una cantidad de épocas de entrenamiento adecuada.

Desarrollo del segmento del algoritmo.

Gracias al entrenamiento, se ha obtenido un modelo entrenado llamado "OCRV1.pt". A continuación se va a desarrollar el código que va a permitir extraer una cadena de caracteres con la información de la matrícula.

Las librerías que se van a emplear son OpenCV para el procesado de la imagen y Ultralytics.

En primer lugar, se lee la imagen con la matrícula recortada obtenida en el algoritmo de detección. Este proceso de recortar la imagen mejora en gran medida los tiempos de procesado de la detección de los glifos en la matrícula.

Para facilitar la comprensión en la detección, se va a escalar la imagen hasta unos valores que permitan visualizar las detecciones.

En la explicación del algoritmo se va a emplear la imagen de la figura 3.46.

Figura 3.46: *Imagen con la matrícula "COVID19" preprocesada y recortada. Elaboración propia.*

Una vez obtenida la imagen, se carga el modelo de YOLOv8 obtenido en el entrenamiento y se aplica sobre la imagen escalada.

El resultado es una estructura que contiene las bounding boxes que rodean a cada detección, así como a qué clase pertenece cada una de ellas.

Haciendo uso de la función plot incorporada en la librería de YOLO, se representan dichas bounding boxes sobre la imagen original, obteniendo de esta manera la imagen de la figura 3.47.

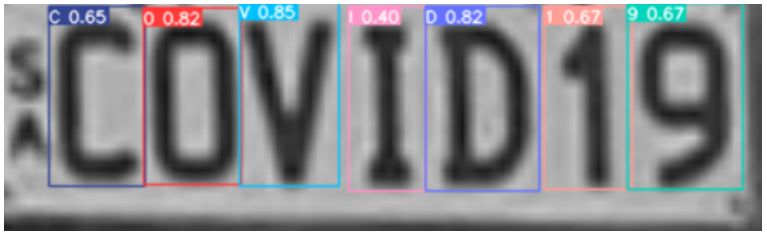


Figura 3.47: *Imagen con la matrícula "COVID19" con las detecciones representadas. Elaboración propia.*

Una vez detectados los glifos, es necesario hacer uso de un breve algoritmo de clasificación para ordenar las detecciones de izquierda a derecha. Una vez logrado, se traducen los valores identificativos de las clases detectadas que representan los números y letras que conocemos. Es decir, si se detecta una clase "11" en la imagen, se traduce como la letra B mediante el uso de un diccionario de Python.

Una vez logrado, se convierte la lista ordenada de detecciones a una cadena de texto, tal y como se muestra en la figura 3.48.

```
Matrícula:  C0VID19
```

Figura 3.48: *Cadena de texto extraída de la matrícula en la consola de Python. Elaboración propia.*

Se ha adaptado el algoritmo para leer matrículas de cualquier longitud, y de hasta dos filas de caracteres, como la imagen de la figura 3,49.


```
Matrícula:  4315BHH
```

Figuras 3.49 y 3.50: *Matrícula de dos filas y la cadena de texto extraída de ella en la consola de Python. Elaboración propia.*

3.3.3. Algoritmos OCR de código abierto.

Para comprobar y validar la eficacia y eficiencia de los algoritmos desarrollados en los apartados anteriores, se va a hacer uso de dos librerías OCR de código abierto: EasyOCR y Tesseract.

Al hacer uso de estas librerías, se puede implementar el algoritmo de reconocimiento de caracteres en unas pocas líneas de código, obteniendo resultados predecibles, robustos y con una gran variedad de información sobre las detecciones.

En este apartado se van a implementar ambas librerías en dos sencillos programas de Python, haciendo uso de comandos de OpenCV y del entorno de PyCharm.

Además, se va a evaluar el tiempo promedio de procesamiento, la precisión en la detección y las ventajas e inconvenientes que presentan.

Antes de comenzar, se van a hacer una serie de suposiciones sobre las características de ambos algoritmos.

Puesto que tanto EasyOCR y Tesseract están basadas en modelos de predicción de redes neuronales, se espera que los resultados sean muy precisos, especialmente al tratarse de fuentes de texto estandarizadas y con las que muy probablemente se hayan entrenado estos modelos de predicción.

Por otro lado, se espera que los tiempos de procesamiento sean relativamente altos. Esto se debe a que generalmente los algoritmos de predicción basados en redes neuronales exigen unos altos pesos computacionales.

Sin embargo, se espera que estos tiempos se reduzcan considerablemente, debido a que son arquitecturas de redes entrenadas y personalizadas para el reconocimiento de caracteres.Por tanto, se espera que las redes neuronales de estos algoritmos OCR se encuentren muy optimizadas.

A continuación se van a implementar las librerías. Para ello, se va a utilizar a modo de ejemplo la imagen de la matrícula procesada mostrada en la figura anterior.

Preprocesado en ambos algoritmos.

Dado que ambos algoritmos OCR están basados en principios similares y presentan una gran cantidad de similitudes en funcionamiento, los pasos a seguir en el preprocesado se van a compartir en gran medida.

Esta etapa de preprocesado comienza cargando la imagen de la matrícula, convirtiéndola a escala de grises y escalando la imagen hasta un tamaño uniforme (generalmente más grande que la imagen original).

Este paso es necesario debido a que al trabajar con imágenes de un tamaño más grande los algoritmos ofrecen un mejor comportamiento a la hora de detectar los caracteres, además de eliminar falsas detecciones debidas a confusiones con el fondo de la imagen.

Además, se aplica un leve filtro suavizante sobre la imagen para homogeneizar bordes y difuminar el ruido.

El siguiente paso es una binarización adaptativa de imagen para separar los glifos del fondo, lo cual además elimina artefactos e información innecesaria en el fondo de la imagen.

El resultado del preprocesado se muestra en la figura 3.51.

Figura 3.51: *Imagen de la matrícula tras el preprocesado. Elaboración propia.*

A continuación se va a procesar esta imagen mediante EasyOCR y Tesseract. Se van a implementar ambas librerías en el entorno de PyCharm y se van a explicar tanto los métodos como las decisiones de ingeniería tomadas en cada momento.

EasyOCR

Para instalar la librería en el entorno de PyCharm, es tan sencillo como hacer uso del comando de pip.

El siguiente paso es indicar mediante el comando “easyocr.Reader” el lenguaje del que se pretende leer, además de indicar si se va a hacer uso de la GPU para la detección.

En este caso no se pretende leer en ningún lenguaje, por lo que se especifica un lenguaje que disponga de todos los caracteres y números del mundo occidental, como por ejemplo el inglés.

Para hacer uso de la GPU, es necesario instalar el software de CUDA, y configurarlo para la tarjeta gráfica de Nvidia de la que se disponga.

Una vez indicados los parámetros y configurado el software de CUDA, es tan sencillo como hacer uso del método “.readtext()” y extraer la cadena de texto del atributo correspondiente en el resultado.

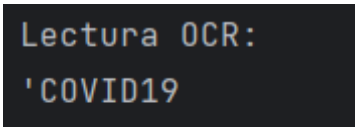


Figura 3.52: *Resultado de la lectura OCR de la matrícula mediante EasyOCR. Elaboración propia.*

Como se puede comprobar en la figura anterior, la cadena de texto extraída de la consola de Python es errónea, ya que el algoritmo ha detectado uno de los blobs de texto de la imagen como un apóstrofe.

Para eliminar este carácter, se va a hacer uso de un método de Python que permite detectar los caracteres especiales en una cadena de texto, para posteriormente eliminarlos. Este método es “.isalnum()”, y se aplica individualmente sobre cada carácter de la cadena de texto

El algoritmo resultante es increíblemente eficaz, ya que elimina todos los caracteres detectados en los artefactos de las imágenes.

El resultado de la matrícula final tras el procesado se muestra en la figura 3.53.

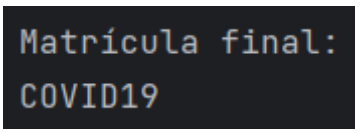


Figura 3.53: *Resultado de la lectura OCR de la matrícula mediante EasyOCR (filtrada). Elaboración propia.*

Tesseract

Para instalar la librería en el entorno de PyCharm, es tan sencillo como hacer uso del comando de pip.

El siguiente paso consiste en descargar e instalar el propio software de Tesseract, para que el programa pueda tener acceso a los modelos de predicción del algoritmo.

Un problema que se ha encontrado a la hora de aplicar el modelo de detección, es que se ha tenido que especificar manualmente la ruta de la instalación del programa en el ordenador, mediante el comando:

“pytesseract.pytesseract.tesseract_cmd = ‘ruta’ ”

A continuación, se aplica el método “pytesseract.image_to_string()” para extraer la cadena de texto de la imagen binarizada de la matrícula.

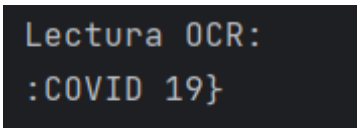


Figura 3.54: *Resultado de la lectura OCR de la matrícula mediante Tesseract. Elaboración propia.*

Como se puede comprobar en la imagen anterior, ocurre lo mismo que en el programa de EasyOCR. El algoritmo es capaz de reconocer los caracteres correctamente, pero interpreta los artefactos de la imagen como caracteres especiales.

Para eliminarlos se aplica el método “.isalnum()” sobre cada elemento de la cadena de texto para identificar los caracteres especiales, y en caso de ser el caso se elimina ese elemento de la lista.

El resultado de la operación se muestra en la figura 3.55.

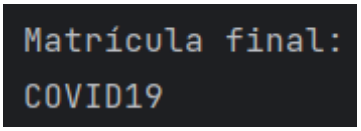


Figura 3.55: *Resultado de la lectura OCR de la matrícula mediante Tesseract (filtrada). Elaboración propia.*

3.4. Evaluación de algoritmos, comparación de resultados y elección de los algoritmos óptimos para la aplicación.

En este apartado, en primer lugar se van a evaluar los tres algoritmos desarrollados en los apartados 3.2.1, 3.2.2 y 3.2.3.

Para ello, se va a realizar un estudio sobre el mismo hardware sobre las ventajas y los inconvenientes que presenta cada algoritmo.

Posteriormente se va a realizar una comparación objetiva entre las características de los algoritmos de detección, con el fin de determinar cuál es el más adecuado para la aplicación.

En segundo lugar, se van a evaluar los algoritmos desarrollados en los apartados 3.3.1, 3.3.2 y 3.3.3 de la misma forma.

Para ello, se va a realizar un breve estudio con el mismo hardware sobre las ventajas e inconvenientes que presenta cada algoritmo, con el fin de determinar el algoritmo más adecuado para la aplicación.

Posteriormente se va a realizar una comparación entre las características de los algoritmos OCR, con el fin de determinar cuál es el más adecuado para la aplicación.

Además, para obtener una perspectiva general del rendimiento de ambos algoritmos, se va a comparar la eficacia de aquellos desarrollados por mí con las mejores alternativas de código abierto disponibles actualmente, Tesseract y EasyOCR.

3.4.1. Evaluación de los algoritmos de detección de matrículas.

A continuación, para cada algoritmo de detección se va a realizar una prueba de funcionamiento sobre 50 imágenes con matrículas en posiciones similares a las que tomaría una cámara de un lector de matrículas en una barrera automática.

Algunas de estas imágenes tienen parte de la matrícula obstruida o tienen pésimas condiciones de luminosidad.

El objetivo de esta prueba es comprobar de manera objetiva la robustez del algoritmo ante imágenes en condiciones tanto favorables como desfavorables.

Evaluación del algoritmo basado en métodos tradicionales.

El algoritmo ha logrado un 82% de precisión en las predicciones y localizaciones de las matrículas, con un tiempo promedio de 9.3 milisegundos de procesamiento.

Las imágenes sobre las que no se ha logrado obtener la matrícula han sido aquellas con parte de la matrícula obstruida y una gran cantidad de imágenes de camiones. Esto es debido a que los camiones presentan muchas formas poligonales en su cara delantera, las cuales se confunden con el rectángulo de la matrícula y son muy difíciles de distinguir.

Este resultado es muy positivo, ya que con un poco más de tiempo y recursos se podría mejorar la mala detección en camiones, realizando ajustes dinámicos sobre los tamaños de las matrículas o empleando reconocimiento de patrones para confirmar que la detección en la imagen sea una matrícula.

Es muy relevante destacar que para hacer funcionar el algoritmo sobre todas las imágenes se han tenido que realizar algunos ajustes sobre ciertos parámetros, para adaptarse a los requerimientos prácticos de cada imagen.

Los parámetros que se deben ajustar son los siguientes:

- La máscara del preprocesado.
- Los puntos necesarios para realizar la transformación afín.
- Los parámetros del detector de bordes.
- Los parámetros de la operación Open personalizada.
- El parámetro de aproximación de polígonos, épsilon.
- Los umbrales del tamaño de la matrícula en la imagen.

También es importante destacar que sería interesante ver el comportamiento de este algoritmo en una prueba de campo, ya que se han tenido que ajustar manualmente los parámetros para cada imagen. En cambio, en una instalación

ANPR fija dichos parámetros se mantendrían fijos y es posible que la precisión en la detección resulte ser inferior.

Este problema podría ser solucionable implementando un bucle donde se ejecute el algoritmo completo varias veces empleando varios parámetros distintos, aunque esto incrementará en gran medida el tiempo de procesado. Considerando lo extremadamente veloz que es el algoritmo, esto no supondría ningún problema.

En la tabla 3.1 se recogen las características finales de este algoritmo de detección basado en métodos tradicionales.

Características del algoritmo de detección basado en métodos tradicionales	
T de procesado:	9.3 ms
Precisión en test:	82%
Preprocesado:	Extenso y particular para cada sistema.
Desventajas:	-Requiere adaptar el algoritmo a las características de cada emplazamiento. -Inflexible ante cambios en las condiciones ambientales. -Particularmente malo reconociendo matrículas de camiones.
Ventajas:	-Extremadamente veloz y simple. -Requiere muy pocas imágenes de prueba.

Tabla 3.1: *Características del algoritmo de detección basado en métodos tradicionales. Elaboración propia.*

Evaluación del algoritmo basado en reconocimiento de patrones.

El algoritmo ha logrado detectar correctamente la ubicación de la matrícula en 12 de las 50 imágenes, un 24% de precisión. Además, el tiempo promedio de procesado de la imagen para obtener una detección ha sido de 270.9 milisegundos.

Este resultado es muy malo. Se ha experimentado con varias soluciones potenciales para este algoritmo, pero tras largas horas de plantear mejoras y tratar de aplicarlas se ha determinado que el algoritmo no es apto para su uso.

Este mal rendimiento se debe a varios factores. Por un lado, el algoritmo ha confundido en una gran cantidad de ocasiones el fondo de la imagen con una matrícula. Estos falsos positivos pueden deberse a que el pattern matching requiere de una coincidencia muy alta en forma y color entre las imágenes para ofrecer un resultado preciso.

En este caso, un leve cambio en la iluminación, posición e incluso si la tipografía empleada en la matrícula de la imagen es muy diferente de la empleada en las plantillas ha llevado a un mal funcionamiento del algoritmo.

Se ha tratado de realizar el reconocimiento de patrones con las imágenes a color, pero esto no ha presentado ninguna mejora notable en la precisión, sino que ha incrementado el tiempo de procesado promedio hasta los 600 ms al tener que procesar tres canales de color distintos. Otra solución para este problema es disponer de más imágenes de matrículas para buscar, pero esto lleva a otro problema diferente. Considerando que se ha de realizar este reconocimiento de patrones una gran cantidad de veces, incrementar este número de iteraciones necesarias conlleva peores tiempos de procesado.

Las características del algoritmo de detección basado en reconocimiento de patrones se recogen en la tabla 3.2.

Características del algoritmo de detección basado en reconocimiento de patrones.	
T de procesado:	270.9 ms
Precisión en test:	24%
Desventajas:	-Preprocesado extenso y particular para cada sistema. -Altos tiempos de procesado. -Poco flexible ante leves cambios en las condiciones ambientales. -Requiere un gran banco de imágenes de matrículas.
Ventajas:	-Algoritmo simple de programar. -Requiere pocas imágenes de la instalación física para funcionar.

Tabla 3.2: *Características del algoritmo basado en reconocimiento de patrones. Elaboración propia.*

Evaluación del algoritmo basado en modelos de detección.

El algoritmo ha logrado un 98% de precisión en las predicciones y localizaciones de las matrículas, con un tiempo promedio de 81 milisegundos de procesamiento.

Este resultado es excepcionalmente positivo, considerando que dentro de las 50 imágenes sobre las que se ha realizado la prueba hay muchas imágenes en condiciones de luminosidad y visibilidad bajas. La única predicción en la que se ha fallado ha sido una de las imágenes más confusas, donde apenas se distingue la matrícula, por lo que se considera un entorno extremadamente desfavorable.

Las características de este algoritmo de detección basado en la arquitectura de red neuronal de YOLOv8 se resumen de manera simplificada en la tabla 3.3.

Características del algoritmo de detección basado en YOLOv8	
T de procesado:	81.1 ms
Precisión en test:	98 %
Desventajas:	-Requiere de un dataset grande y variado para el entrenamiento. -Tiene unos altos requisitos de hardware.
Ventajas:	-Flexible ante cambios en condiciones ambientales. -Extremadamente preciso en la detección. -No requiere de ningún preprocesado de las imágenes.

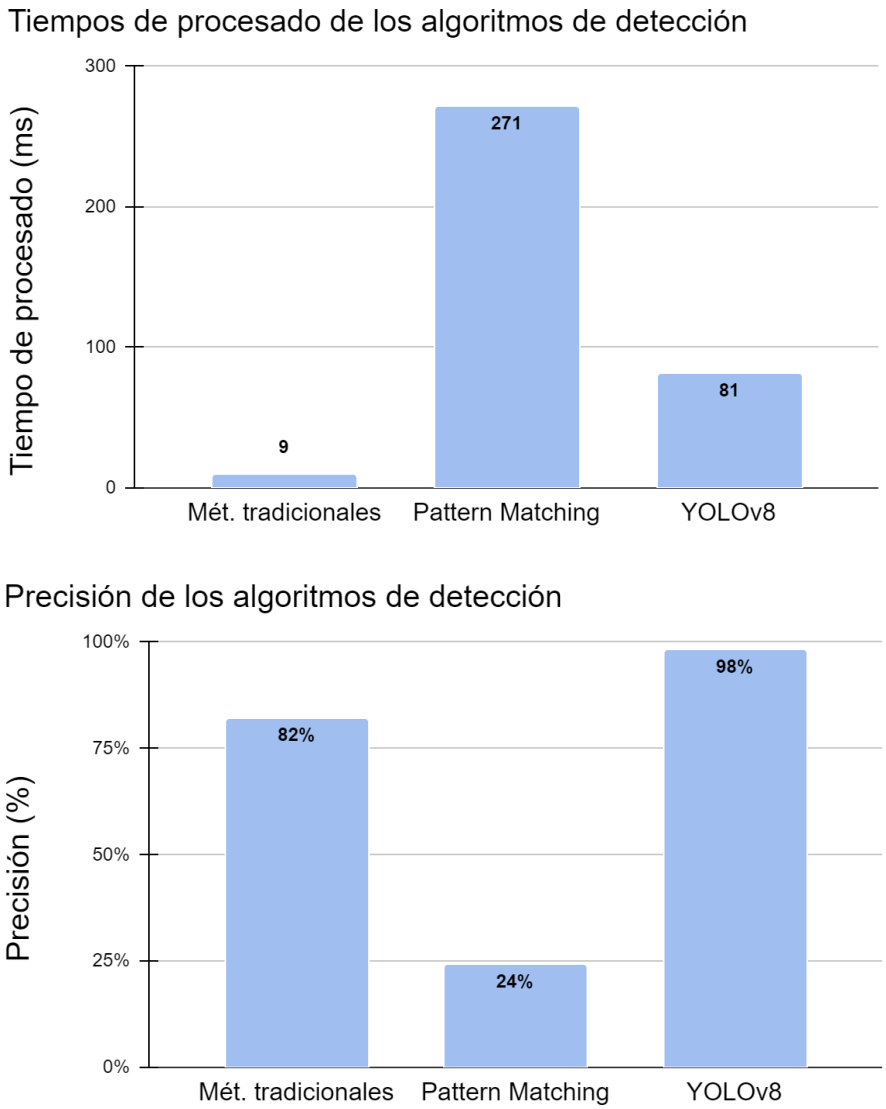
Tabla 3.3: *Características del algoritmo de detección basado en IA. Elaboración propia.*

3.4.2. Comparación de las características de los algoritmos de detección y elección del algoritmo óptimo para la aplicación.

Antes de comenzar se va a hacer una declaración importante. Dado que los resultados obtenidos con el algoritmo de detección basado en reconocimiento de patrones han sido extremadamente malos, queda claro antes de comenzar que este algoritmo no va a ser elegido.

Sin embargo, se van a comparar sus características con los otros dos algoritmos para ofrecer una perspectiva global del rendimiento de los mismos.

Es tan importante dejar claros los problemas que han hecho que un algoritmo falle como señalar los puntos fuertes y las decisiones que han ofrecido un mejor resultado.



Figuras 3.56 y 3.57: *Comparación en tiempos de procesamiento y precisión de los algoritmos de detección de matrículas. Elaboración propia.*

Comparación en ventajas y desventajas generales.		
Algoritmo	**Ventajas**	**Inconvenientes**
Métodos tradicionales.	-Extremadamente veloz y simple. -Requiere de muy pocas imágenes de prueba de cada instalación física para el funcionamiento.	-Requiere adaptar el algoritmo a las características de cada emplazamiento. -Inflexible ante cambios en las condiciones ambientales. -Pobre reconociendo matrículas de camiones.
Reconocimiento de patrones.	-Preprocesado extenso y particular para cada sistema. -Poco flexible ante leves cambios en las condiciones ambientales. -Requiere un gran banco de imágenes de matrículas.	-Algoritmo simple de programar. -Requiere pocas imágenes de la instalación física para funcionar.
YOLOv8.	-Robusto y flexible ante cambios en condiciones ambientales. -No necesita preprocesado en las imágenes.	-Requiere de una base de datos de imágenes grande y variada para el entrenamiento. -Tiene unos altos requisitos de hardware.

Tabla 3.4: *Comparación en ventajas y desventajas generales de los algoritmos de detección de matrículas. Elaboración propia.*

Todos los resultados de los algoritmos de detección de matrículas se han recogido en la tabla y figuras anteriores.

En velocidad de procesamiento, sin duda el algoritmo más rápido es el basado en métodos tradicionales, debido a la simplicidad del programa.

El algoritmo basado en YOLOv8 es significativamente más lento, pero el resultado sigue siendo excepcionalmente rápido para ser una solución basada en modelos de predicción de redes neuronales.

Dado que se busca obtener un algoritmo con un tiempo de procesado de menos de 1 segundo, ambas opciones son viables puesto que dejan un amplio margen de tiempo para la segunda mitad del algoritmo, la parte OCR.

En cuanto a precisión en las detecciones, el algoritmo más preciso y robusto es sin duda el algoritmo basado en YOLOv8. Esto no es una sorpresa, ya que los algoritmos basados en redes neuronales tienden a ser muy certeros y flexibles.

En cuanto a las ventajas y las desventajas de los algoritmos, lo más destacable es el contraste entre el extremo preprocesado de las imágenes en el algoritmo de métodos tradicionales en comparación con el inexistente preprocesado en el algoritmo de IA, si se excluye el trabajo previo de recopilación de imágenes y entrenamiento.

También es importante destacar la flexibilidad en los cambios ambientales que permite el algoritmo basado en IA en comparación con la baja flexibilidad que tolera el algoritmo basado en métodos tradicionales.

Elección del algoritmo de detección óptimo para la aplicación.

Dado que ambos algoritmos, de métodos tradicionales e IA, son viables, se va a tener que tomar la decisión basándose en circunstancias realistas.

En una aplicación real, es importante que el tiempo de instalación y puesta a punto sea bajo. El algoritmo de inteligencia artificial destaca en esta característica al ahorrar tiempo y dinero de operarios cualificados que deben configurar el sistema basado en métodos tradicionales.

Además, es necesario recordar que las condiciones ambientales en las que debe funcionar el algoritmo, previamente mencionadas en el apartado 3.1, son muy variables al encontrarse en el exterior.

Por otro lado, aunque ambos algoritmos se han considerado viables, claramente el algoritmo basado en modelos de detección es más fiable que el basado en métodos tradicionales. Al tratarse de un algoritmo pensado para funcionar en una barrera de acceso a un recinto privado, un algoritmo más preciso incrementa directamente la seguridad en dicho recinto.

Por estos motivos, se va a elegir sobre los demás el algoritmo basado en aprendizaje automático para realizar las detecciones de matrículas.

Es necesario recordar que las pruebas de rendimiento se han realizado en un ordenador de alta gama pensado para aplicaciones de alto coste computacional, por lo que los tiempos de procesado en una instalación industrial pueden variar significativamente.

3.4.3. Evaluación de los algoritmos de reconocimiento de caracteres.

A continuación, para cada algoritmo OCR se va a realizar una prueba de funcionamiento sobre 50 imágenes predefinidas, con el fin de obtener una idea general de las características de cada algoritmo.

Estas imágenes contienen matrículas en condiciones de iluminación variadas de Europa, Estados Unidos, Canadá, Australia y Nueva Zelanda.

El objetivo de esta prueba es comprobar de manera objetiva la robustez del algoritmo ante imágenes de matrículas en condiciones tanto favorables como desfavorables.

Evaluación y características del algoritmo OCR basado en reconocimiento de patrones.

El resultado para este algoritmo fue de 39 aciertos sobre 50, un 78% de precisión en la detección.

El resultado del algoritmo es considerablemente bueno, dado el reto que suponen las imágenes de comprobación. Es necesario recalcar que se han tenido que ajustar en gran medida los parámetros de escala y confianza para cada imagen, lo que significa que el algoritmo es extremadamente dependiente de conocer los parámetros de cada entorno.

Otro gran problema que se ha encontrado con este algoritmo han sido los falsos positivos. En especial aquellos en los que se han confundido “I” con “1”, “0” con “O”, “Q” y “D”, etc.

Este problema es muy difícil de solucionar. En Europa y Nueva Zelanda, por reglamento se han eliminado estos caracteres confusos para evitar este tipo de situaciones. En cambio, en EE UU, Canadá y Australia, se permite circular con matrículas personalizadas con cualquier carácter o número.

Por tanto se asume que este algoritmo será mucho más eficiente con matrículas europeas y neozelandesas que con aquellas del resto de países.

Una forma en la que se puede mejorar el algoritmo es alterar la fuente de texto personalizada. Si se conoce el entorno de la instalación ANPR, y se pueden tomar imágenes de referencia en el sitio, se espera que el algoritmo mejore en gran medida su precisión.

El algoritmo toma en promedio 108 ms en detectar e identificar todos los glifos en la imagen. Este tiempo es extremadamente consistente debido a que en todas las imágenes se busca el mismo número de glifos, y el tamaño de las

imágenes se ha escalado para que siempre se tenga que analizar el mismo número de píxeles.

En la tabla 3.5 se recopilan las características determinadas para este algoritmo.

Características del algoritmo OCR basado en Template Matching	
T de procesado:	108 ms
Precisión en test:	78 %
Preprocesado:	Simple, pero particular para cada sistema.
Desventajas:	-Requiere adaptar el algoritmo a las características de cada emplazamiento. -Particularmente malo reconociendo matrículas de EE UU, Canadá y Australia.
Ventajas:	-Simple. -Requiere muy pocas imágenes de prueba para el funcionamiento del algoritmo.

Tabla 3.5: *Características del algoritmo OCR basado en Template Matching. Elaboración propia.*

Evaluación y características del algoritmo basado en modelos de detección.

A continuación se va a realizar la evaluación del algoritmo basada en la prueba sobre 50 imágenes de matrículas. El resultado para este algoritmo fue de 47 aciertos sobre 50. Un 94% de precisión en la detección.

Las imágenes sobre las que se han fallado tenían características muy extrañas y únicas. Por ejemplo, esta matrícula canadiense mostrada en la figura 3.58 tiene una forma muy peculiar, y la parte inferior izquierda de esta figura tan curiosa se confunde con un “2”, denotada por un recuadro rojo.

Figura 3.58: *Matrícula de los Territorios del Noroeste de Canadá. De matriculasdelmundo.com.*

En cuanto al tiempo de ejecución, este segmento toma en promedio 49 ms en ejecutarse y devolver la respuesta de código. Considerando la cantidad de clases que se deben de detectar, este tiempo es extremadamente rápido.

Esto es probablemente debido al reducido tamaño de la imagen de la matrícula en la que se debe realizar la detección. Además, la arquitectura especial de YOLO lee la imagen de entrada una única vez para realizar todas las detecciones, lo que mejora en gran medida los tiempos de procesado.

En la tabla 3.6 se recogen las características evaluadas de este algoritmo.

Características del algoritmo OCR basado en IA	
T de procesado:	49.6 ms
Precisión en test:	96 %
Desventajas:	-Requiere de un gran y variado banco de imágenes para el entrenamiento. -Tiene unos altos requisitos de hardware.
Ventajas:	-Robusto y flexible ante cambios en condiciones de orientación e iluminación. -Extremadamente preciso en la detección. -No requiere de preprocesado de las imágenes.

Tabla 3.6: *Características del algoritmo OCR basado en IA. Elaboración propia.*

Evaluación de los algoritmos basados en soluciones de código libre.

El algoritmo de EasyOCR ha tardado en promedio 131 milisegundos en ejecutarse, y ha leído correctamente 46 de las 50 matrículas, un 92% de precisión.

Por otro lado, el algoritmo de Tesseract ha tardado en promedio 94 milisegundos en ejecutarse, y ha leído correctamente 43 de las 50 matrículas, un 86% de precisión.

Aunque los resultados son muy buenos, se han encontrado varios problemas a la hora de analizar las imágenes. El primer problema que sufren ambos algoritmos es que se detectan partes del fondo de las imágenes y se confunden con letras, números o caracteres especiales “ . : , ; - _ () [] {}”. Aún así, el problema de los caracteres especiales se ha solucionado en gran medida, tal y como se ha visto en el apartado del desarrollo de estos algoritmos.

El problema principal son los falsos positivos en las detecciones de letras y números. Este problema se debe principalmente a que las librerías de OCR buscan encontrar todo el texto de una imagen, cuando en esta aplicación en realidad se necesita encontrar únicamente entre 4 y 9 caracteres.

Esta falta de especialización en el entrenamiento hace que estos algoritmos detecten letras, códigos regionales o palabras presentes en la matrícula ajenas a la información de la misma, ya que tratan de encontrar la mayor cantidad de caracteres posible causando que fallen.

Las características de ambos algoritmos se recogen en la tabla 3.7, y se hará uso de ellas para evaluar los otros algoritmos en el apartado 3.3.4.

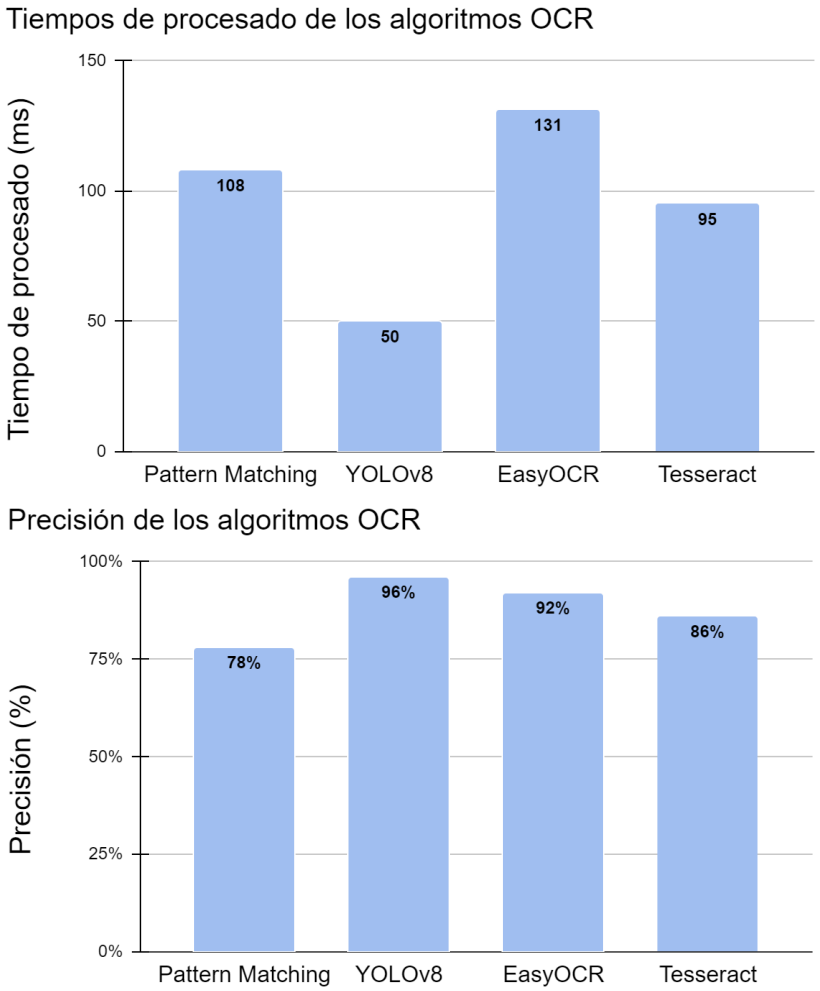
Características de los algoritmos basados en EasyOCR y Tesseract		
Característica	**EasyOCR**	**Tesseract**
T de procesado:	131.1 ms	94.9 ms
Precisión en test:	92%	86%
Desventajas:	-Problemas al no ser modelos entrenados para ANPR.	
Ventajas:	-Previamente entrenados y muy sencillos de implementar. -Preprocesado sencillo y siempre el mismo.	

Tabla 3.7: *Características de los algoritmos OCR de EasyOCR y Tesseract. Elaboración propia.*

3.4.4. Comparación de las características de los algoritmos OCR y elección del algoritmo más óptimo para la aplicación.

Antes de comenzar, se va a aclarar que uno de los objetivos del proyecto es desarrollar personalmente los algoritmos. Los algoritmos basados en EasyOCR y Tesseract son soluciones previamente desarrolladas, entrenadas y extremadamente sencillas de implementar mediante una o dos líneas de código.

Por este motivo, independientemente de su rendimiento, no se va a considerar implementar estas dos soluciones en el algoritmo final. En su lugar, se van a utilizar para comparar el rendimiento de los algoritmos desarrollados manualmente con otros ampliamente empleados.



Figuras 3.59 y 3.60: *Comparación en los tiempos de procesado y precisión de los algoritmos OCR. Elaboración propia.*

Comparación en ventajas y desventajas generales de los algoritmos OCR.		
Algoritmo	**Ventajas**	**Inconvenientes**
Reconocimiento de patrones.	-Simple. -Requiere muy pocas imágenes de prueba.	-Requiere adaptar el algoritmo a las características de cada emplazamiento. -Malo reconociendo matrículas de EE UU, Canadá y Australia.
YOLOv8.	-Robusto y flexible ante cambios en condiciones de orientación e iluminación. -Extremadamente preciso en la detección. -No requiere de ningún preprocesado de las imágenes de entrada.	-Requiere de un grande y variado dataset para el entrenamiento. -Tiene unos altos requisitos de hardware.
EasyOCR.	-Previamente entrenados y sencillos de implementar. -Preprocesado sencillo y siempre el mismo.	-Problemas al no ser modelos entrenados para ANPR (detectan partes del fondo de la imagen)
Tesseract.		

Tabla 3.8: *Comparación en ventajas y desventajas generales de los algoritmos de detección de matrículas. Elaboración propia.*

En las tablas anteriores se muestra una comparación entre las características más relevantes de los algoritmos desarrollados en los apartados anteriores.

El algoritmo más veloz es sin duda alguna el basado en YOLOv8. Como ya se ha explicado anteriormente, la arquitectura de red neuronal de YOLO es conocida por su velocidad de procesamiento, por lo que este resultado no es una sorpresa.

Aunque el algoritmo de YOLOv8 tarda la mitad de tiempo en realizar la lectura de los caracteres en comparación con el resto de los algoritmos, estamos hablando de como mucho dos décimas de segundo para realizar la lectura de los caracteres con el algoritmo OCR más lento.

Si se suma este tiempo a los tiempos de procesado obtenidos en los algoritmos de detección de la matrícula, se obtiene un tiempo máximo de 0.3 segundos por detección.

En cuanto a la precisión en las detecciones, el algoritmo más preciso vuelve a ser el basado en YOLOv8, el cual ha ofrecido resultados muy similares a los algoritmos de EasyOCR y Tesseract. Esto significa que se ha logrado obtener un algoritmo con una eficiencia superior a las soluciones OCR de código abierto más populares.

En cuanto a las ventajas generales, se destaca el contraste entre los dos algoritmos principales. En cuanto al algoritmo de YOLOv8 se destaca el alto coste de entrenamiento y la facilidad de utilización de la red neuronal una vez entrenada.

Por otro lado, se destaca la simplicidad en desarrollo del algoritmo de pattern matching en comparación con la gran cantidad de ajustes específicos necesarios para cada instalación física.

Elección del algoritmo OCR óptimo para la aplicación.

A continuación se va a tomar la decisión sobre qué algoritmo se va a emplear en el segmento OCR del programa. Como ya se ha mencionado anteriormente, se va a discutir si emplear el algoritmo OCR basado en YOLOv8 y el basado en reconocimiento de patrones.

Dado que ambos algoritmos son relativamente viables, se ha de tomar una decisión basándose en los puntos más realistas.

Por un lado, en una aplicación real lo verdaderamente importante es el tiempo de instalación y puesta a punto, donde destaca el algoritmo de inteligencia artificial al ahorrar tiempo y dinero de operarios cualificados que deben configurar el sistema basado en métodos de pattern matching para cada situación específica.

Si se considera además la gran flexibilidad del algoritmo basado en la arquitectura de YOLO y que el algoritmo de pattern matching tiene una precisión inferior al 80%, se ha decidido emplear el algoritmo basado en YOLOv8 en el sistema ANPR final.

Además, se recuerda que las pruebas de rendimiento se han realizado en un ordenador de alta gama pensado para aplicaciones de alto coste computacional, por lo que los tiempos de procesado en una instalación industrial pueden variar significativamente.

3.5. Algoritmo ANPR completo.

Tal y como se ha determinado en los apartados 3.4.2 y 3.4.4, los algoritmos óptimos para la detección y reconocimiento de caracteres han sido aquellos basados en inteligencia artificial, en específico en modelos de predicción de YOLO v8.

El siguiente paso es combinar ambos algoritmos en uno. Esto es tan sencillo como encadenar el resultado del algoritmo de detección con el algoritmo OCR en un mismo programa, ordenando y optimizando el código lo máximo posible. De esta manera, se obtiene el algoritmo ANPR en cascada completo.

Para mostrar un ejemplo con el algoritmo completo, se va a trabajar sobre la imagen de una furgoneta con matrícula europea, mostrada en la figura 3.61.

Figura 3.61: *Imagen de una furgoneta. De O. A. Brekke, 2009.*

En primer lugar se aplica el modelo de detección de matrículas de YOLO v8, con lo que se obtiene la región de la imagen que contiene la matrícula, mostrada en la figura 3.62.

Figura 3.62: *Detección de una matrícula en la imagen de la furgoneta mediante el modelo de predicción de YOLOv8. Elaboración propia.*

El siguiente paso consiste en obtener el recorte de la matrícula de la imagen original, como se muestra en la figura 3.63.

Figura 3.63: *Recorte de matrícula de la furgoneta. Elaboración propia.*

Generalmente, sería necesario aplicar una transformación proyectiva al recorte de la imagen, pero dado que el modelo de predicción ha sido entrenado con imágenes en una gran cantidad de orientaciones, las detecciones son robustas sin necesidad de aplicar esta transformación.

Tras aplicar el modelo de predicción de YOLO v8 para OCR sobre el recorte de la matrícula, se obtiene el resultado mostrado en la figura 3.64.

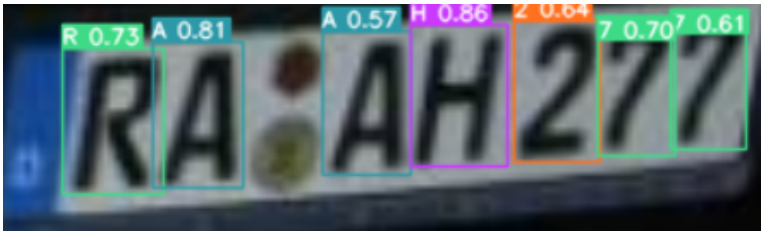


Figura 3.64: *Recorte de matrícula de la furgoneta con las detecciones del modelo OCR de YOLOv8. Elaboración propia.*

Por último se procesan las detecciones para determinar la matrícula en la imagen, obteniéndose en la consola de Python la siguiente cadena de texto con la información de la matrícula leída.

```
Matrícula:  RAAH277
```

Figura 3.65: *Cadena de texto extraída de la matrícula de la furgoneta en la consola de Python. Elaboración propia.*

Por tanto, en el algoritmo completo toma como entrada un fotograma de la cámara de vídeo y devuelve una cadena de texto con la información de la matrícula.

Esto significa que el algoritmo se va a aplicar una vez por cada ciclo de programa para tratar de detectar y leer cualquier matrícula que aparezca en pantalla.

3.6. Evaluación y características del algoritmo ANPR completo.

A continuación se va a realizar una evaluación del algoritmo completo sobre una pequeña base de datos de 200 matrículas, donde se han seleccionado imágenes de matrículas europeas, canadienses, estadounidenses, australianas y neozelandesas.

Estas matrículas pertenecen a turismos, furgonetas y camiones. Además, se encuentran en posiciones similares a aquellas que se tomarían en una instalación ANPR, como los ejemplos mostrados en la figura 3.66.

Figura 3.66: *Collage de imágenes de vehículos del conjunto de imágenes de evaluación. De C. J. Santos, en Roboflow.*

Para facilitar el proceso de la evaluación del algoritmo, se ha desarrollado un breve programa que carga todas las imágenes al mismo tiempo y aplica el algoritmo ANPR final una a una dentro de un bucle, cargando los resultados de en una lista ordenada para posteriormente analizarlos.

Estos datos a analizar son el tiempo promedio de ejecución, si se ha fallado en la detección de la matrícula y la cadena de texto con la información de la matrícula extraída de la imagen.

En cuanto al tiempo de procesado, el algoritmo ha tardado en promedio 141 milisegundos en ejecutarse, reducido hasta 85 milisegundos en caso de no detectar ninguna matrícula en la imagen. Este tiempo es extremadamente rápido, considerando que se trata del algoritmo en cascada final.

Por otro lado, los resultados de precisión se muestran en la gráfica de la figura 3.67.

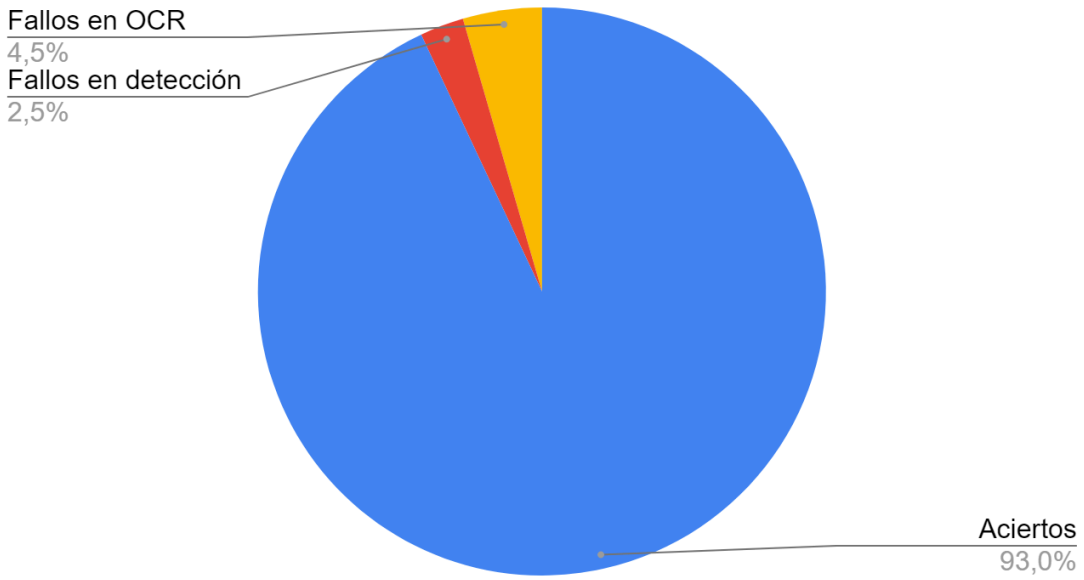


Figura 3.67: *Diagrama con los resultados de precisión en la detección del algoritmo ANPR final. Elaboración propia.*

Como se puede comprobar, el algoritmo final tiene una precisión del 93%.

Los fallos en la detección representan el 2,5% de los casos totales, en los que se ha confundido parte del vehículo con una matrícula.

La mayor parte de los fallos se han dado en la lectura de la información de la matrícula, donde se ha confundido el fondo de la imagen con un glifo en el 4,5% de los casos.

Estos errores son asumibles dados los requerimientos establecidos en el apartado 1.2, donde se ha establecido que el algoritmo debe tener un tiempo de procesado de menos de un segundo y una precisión de más del 80% en la lectura, por lo que se considera que el algoritmo ha superado enormemente las expectativas.

Además, estos errores pueden ser mitigados en gran medida haciendo uso de un banco de datos de imágenes de entrenamiento mucho más grande y variado, lo cual requiere unos tiempos de recopilación de imágenes y entrenamiento que escapan al alcance de este proyecto.

Las características del algoritmo ANPR final se recogen en la tabla 3.9.

Características del algoritmo ANPR final.	
Precisión	93% (4,5% de fallos en detección) (2,5% de fallos en OCR)
T. de procesado	141.36 ms
Ventajas	-Robusto y flexible ante cambios en condiciones de orientación e iluminación. -Extremadamente preciso en la detección. -No requiere de preprocesado en las imágenes.
Desventajas	-Requiere de un gran y variado dataset para el entrenamiento. -Tiene un alto coste computacional, tanto para el entrenamiento como el funcionamiento.

Tabla 3.9: *Características del algoritmo ANPR final. Elaboración propia.*

Además, se recuerda que las pruebas de rendimiento se han realizado en un ordenador de alta gama pensado para aplicaciones de alto coste computacional, por lo que los tiempos de procesado en una instalación de campo pueden ser sustancialmente más elevados.

CAPÍTULO 4. DESARROLLO DE SOLUCIÓN ANPR.

Para emplear el algoritmo sobre un ejemplo realista, en este capítulo se va a explicar detalladamente el desarrollo de una aplicación ANPR pensada para funcionar en una barrera automática doble para un recinto con un único acceso.

Para simplificar algunas decisiones con respecto a la normativa aplicable, se va a plantear la solución ANPR en territorio español.

Este programa debe crear un registro de las matrículas detectadas, que permite dar acceso a los vehículos permitidos y llevar un registro de qué vehículos acceden al recinto y durante cuánto tiempo.

Además, se va a plantear una posible solución de hardware, tanto de software embebido como de los elementos físicos a emplear en la barrera automática.

Los elementos físicos que se van a emplear en esta posible solución ANPR se van a comprar a fabricantes por separado y no se van a diseñar, ya que esto escapa en gran medida del alcance del proyecto.

La solución de software embebido va a ser completamente teórica. Esto se debe a que la migración del algoritmo ANPR a las soluciones de hardware más eficientes puede ser un proyecto en sí mismo y por tanto escapa del alcance de este proyecto.

Se van a explicar todas las decisiones de ingeniería tomadas en cada momento, para tratar de ofrecer claridad y razonamiento detrás de cada idea implementada.

4.1. Desarrollo de aplicación ANPR para un acceso doble.

En este apartado se va a desarrollar una aplicación ANPR para una barrera automática doble, que sirve como entrada y salida de vehículos para un recinto industrial con un único acceso doble.

Para simplificar algunas decisiones con respecto a la normativa aplicable, se va a plantear la solución ANPR en una instalación física de España.

4.1.1. Funcionalidades de la aplicación ANPR.

Las características y funcionalidades más importantes que debe tener la aplicación ANPR se explican a continuación.

La aplicación debe funcionar en una barrera automática doble, que sirve como entrada y salida de vehículos para un recinto industrial con un único acceso.

La aplicación debe identificar correctamente las matrículas tanto en la barrera de entrada como en la de salida. Si la matrícula leída se encuentra en la base de datos, la barrera automática se levanta y se registra la fecha y hora de entrada o salida del vehículo.

Si la matrícula leída no se encuentra en la base de datos, se pregunta al usuario si se desea permitir el acceso o la salida.

En caso afirmativo, se levanta la barrera y se registra el acceso o salida.

En caso negativo, la barrera no se levanta y no se registra ninguna información.

Por otro lado, se debe mostrar en la interfaz de usuario el registro de las entradas y salidas de los vehículos con su fecha y hora entrada o salida, ya sea porque se haya permitido manualmente o porque los vehículos se encuentren registrados en la base de datos.

Por último, se van a mostrar sobre la interfaz las fuentes de vídeo de la entrada y de la salida en tiempo real, así como la información de la matrícula presente en cada cámara, junto con los botones para permitir o rechazar el acceso.

Además, una funcionalidad relevante para seguridad y contraste de información que es interesante añadir, es grabar en un disco duro externo las imágenes obtenidas de las fuentes de vídeo de la entrada y salida.

4.1.2. Desarrollo de la interfaz de usuario.

Una vez determinadas las características de la aplicación, se comienza a desarrollar el programa. Para el desarrollo del programa se va a hacer uso de PyCharm.

El desarrollo de la interfaz de usuario se va a llevar a cabo empleando la librería de Custom Tkinter. Esta librería es una versión extendida de Tkinter, una librería de Python extensamente empleada en el desarrollo de interfaces de usuario.

En un principio se trató de crear la aplicación con Tkinter, pero tras unas pocas horas de trabajo se tomó la decisión de adaptar el desarrollo a la librería de Custom Tkinter para obtener una interfaz de usuario más elegante y funcional.

Tras extensas horas de búsqueda de información y de comprender esta librería mediante ejemplos y documentación, se ha diseñado y desarrollado una interfaz de usuario para esta aplicación, mostrada en la figura 4.1.

Figura 4.1: *Interfaz de usuario de la aplicación ANPR. Elaboración propia.*

En la interfaz de la aplicación se pueden ver las imágenes de las cámaras de vídeo de la entrada y salida en tiempo real, con su respectiva caja de información.

Estas cajas muestran información sobre cada matrícula presente en las entradas y salidas, sobre si dicha matrícula se encuentra en la base de datos o no y sobre el estado de cada barrera.

Todos estos parámetros se van a actualizar en tiempo real para mostrar la mayor cantidad de información posible.

Además, se han implementado dos botones para la entrada y dos para la salida. Estos botones se encuentran desactivados por defecto, y se activarán en el momento que se detecte una matrícula no registrada en alguna de las cámaras.

Con estos botones, el operario puede permitir o denegar la entrada o salida de un vehículo en función de su criterio. Permitir el acceso o salida anotará el evento en un registro.

Este registro de entradas y salidas del recinto se encuentra ordenado con las entradas o salidas más recientes en la parte superior del mismo.

Con esto, se ha desarrollado una estructura general para la interfaz de usuario y se han concretado las distintas funcionalidades que presenta esta aplicación ANPR.

4.1.3. Desarrollo del programa.

Una vez definida la estructura y contenido de la interfaz de usuario, el siguiente paso consiste en desarrollar la lógica para la aplicación e implementar todas estas funciones.

Antes de comenzar el desarrollo, se va a hacer una breve predicción de las características del programa.

Dado que la aplicación procesa la información de dos cámaras al mismo tiempo, los algoritmos de detección y OCR tardan en promedio el doble de tiempo en ejecutarse.

Si se suma esto a que se deben actualizar los elementos de la interfaz gráfica en tiempo real, se prevén unos tiempos de ciclo más altos, especialmente cuando se detecte una matrícula en las cámaras.

Como se explicará más adelante, se han planteado distintas soluciones que mejoran en gran medida este mal rendimiento.

Desarrollo del programa.

El primer paso para el desarrollo del programa consiste en implementar el algoritmo ANPR en el bucle de código de la librería de Custom TKinter.

Este primer paso ha requerido una enorme cantidad de tiempo, ya que han surgido varios problemas.

El primer problema aparece al tratar de ejecutar código en el bucle de código que lee los eventos de la interfaz gráfica y la mantiene activa. El problema consiste en que no se puede insertar código externo en este bucle.

En la propia documentación de la librería se recomienda recurrir a programación multihilo para ejecutar código en paralelo con el bucle principal. Sin embargo, no se pueden actualizar los elementos de la interfaz desde un código ejecutándose en un segundo hilo.

Por ejemplo, esto significa que no se pueden actualizar los contenidos de las etiquetas que muestran la información de la matrícula presente en la pantalla, entre otros muchos elementos.

Por tanto, la única solución a este problema consiste en definir una llamada a una función de CTK. Estas funciones de inicialización se ejecutan una única vez al lanzar el bucle principal.

Sin embargo, si se llama a la función desde el final de la misma, se logra ejecutar código externo en cada iteración del bucle principal.

Una vez resuelto el problema, se procede a desarrollar dos funciones de CTK.

La primera función inicializa los elementos de la interfaz gráfica y define sus propiedades. Estas propiedades son por ejemplo la posición, las variables asociadas a cada elemento y su tipografía, entre muchas otras.

Desde el final de esta primera función, se llama a la función ANPR.

La función ANPR es el núcleo del programa. En ella se ejecutan los dos algoritmos ANPR, uno para la cámara de la entrada y otro para la cámara de la salida.

Además, en esta función se va a implementar la lógica del funcionamiento de la aplicación, que introduce todos los requerimientos adicionales definidos anteriormente.

La función ANPR recibe como entrada un fotograma de vídeo capturado mediante el método de “cv2.VideoCapture(0)”. A continuación se realiza la detección de la matrícula en la imagen.

Como ya se ha explicado anteriormente, los algoritmos ANPR se ejecutan dos veces, una vez por cada cámara. Esto conlleva tiempos de procesado bastante elevados.

En una instalación real no se da ningún problema ya que cada cámara dispone de su propio hardware de procesado de imágenes. Por tanto, aumentar el número de cámaras no influye en el rendimiento del sistema real completo.

En cambio, en el sistema de ejemplo que se está desarrollando en Python se hace uso del mismo hardware para el procesado de las imágenes de ambas cámaras, por lo que los tiempos de procesamiento aumentan en gran medida.

Para solucionar parcialmente este problema se ha recurrido a ejecutar el algoritmo OCR únicamente en el caso de que se realice una detección de matrícula.

Si no se detecta una matrícula, se termina el proceso y se vuelve a ejecutar el algoritmo de detección. Si se detecta una matrícula, entonces se trata de leer la misma ejecutando el algoritmo OCR.

Esta mejora no solo reduce los tiempos de procesado en promedio, sino que además mejora los tiempos de detección al procesar fotogramas nuevos con más frecuencia.

Por desgracia, al implementar esta mejora aparece otro problema distinto. Dado que el algoritmo es extremadamente rápido, existe la posibilidad de que el vehículo del que se ha detectado la matrícula se encuentre lo suficientemente fuera de plano como para que solo se vea una porción de la matrícula, como se muestra en el ejemplo de la figura 4.2.

Figura 4.2: *Detección de una fracción de una matrícula. Elaboración propia.*

Este problema se ha solucionado implementando dos mejoras. La primera consiste en medir la longitud de la cadena de caracteres de la matrícula.

Excluyendo casos muy excepcionales, las matrículas de los países seleccionados para este proyecto tienen entre 4 y 9 caracteres. Por tanto, se comprueba que la longitud mínima de la matrícula sea de 4 caracteres.

La segunda mejora consiste en comprobar si coincide la matrícula con la leída en el fotograma inmediatamente anterior. Si las tres últimas matrículas leídas coinciden se determina que la detección es correcta y se detienen los algoritmos de detección y OCR hasta que se tome una decisión. Aunque esta solución mejora la precisión en la detección, por desgracia ralentiza en gran medida los tiempos de procesado.

Si la matrícula detectada se encuentra en la base de datos, se procede a levantar la barrera y registrar la hora de entrada y la información de la matrícula en el registro, haciendo uso de la librería “os” para leer y escribir el registro.

Si la matrícula no se encuentra en la base de datos, se pregunta al usuario si se desea permitir la entrada. Si la respuesta es negativa, se desactivan los botones, la barrera no se levanta y se rechazan todas las lecturas de matrículas que coincidan con la matrícula rechazada hasta que haya pasado un tiempo predefinido, en este caso de 15 segundos.

Si la respuesta es positiva, se levanta la barrera y se registra la matrícula junto con la hora de acceso o salida.

Junto con las funcionalidades principales del algoritmo se han añadido muchos elementos de mejora de calidad de vida, además de optimizar todo el código empleado.

4.2. Ejemplos de funcionamiento.

Una vez se ha desarrollado la interfaz de usuario y el programa ANPR se han implementado ambas partes sobre una misma aplicación ANPR. En este apartado se va a mostrar un breve ejemplo de funcionamiento sencillo, que abarca la mayoría de funcionalidades del programa.

Tras el arranque del programa se lanza una pantalla de carga, mostrada en la figura 4.3. Esta pantalla de carga se muestra durante los primeros segundos tras iniciar la aplicación, con el fin de realizar todas las configuraciones iniciales.

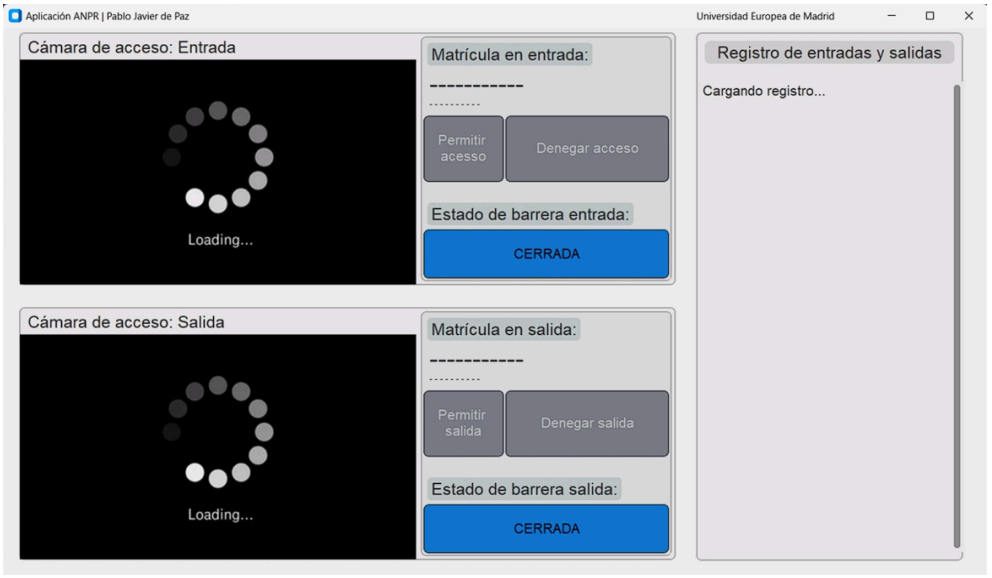


Figura 4.3: *Pantalla de carga de la interfaz de usuario. Elaboración propia.*

Una vez realizada la carga, la interfaz pasa a verse como en la figura 4.4.

Figura 4.4: *Pantalla de operación estándar de la HMI. Elaboración propia.*

Cuando aparece un vehículo en la entrada o en la salida, existen dos posibilidades. La primera es que la matrícula se encuentre registrada en la base de datos y se permita el acceso sin requerir acción alguna del operario. Este evento se muestra en la figura 4.5.

Figura 4.5: *Detección de un vehículo registrado en la base de datos. Elaboración propia.*

La segunda posibilidad es que la información de la matrícula del vehículo detectado no se encuentre registrada en la base de datos. En este caso, los botones de decisión se activan, como se muestra en la figura 4.6.

Figura 4.6: *Detección de un vehículo no registrado en la base de datos. Elaboración propia.*

Cuando se permite un acceso o salida, la barrera se levanta y se registra el evento en el registro de entradas y salidas.

El programa funciona correctamente con dos vehículos a la vez en entrada y salida, ya que se procesan en dos funciones diferentes y completamente separadas.

Es importante destacar que el ejemplo mostrado se ha realizado con un video pregrabado, lo que al emplear la librería de OpenCV implica que se procesan todos los fotogramas.

En una instalación real, el vídeo se actualizará en tiempo real, ignorando todos los fotogramas que han transcurrido durante los instantes de procesamiento de imagen.

4.3. Propuesta de embarcado en hardware.

4.3.1. Propuesta de software embebido.

A lo largo del desarrollo de los algoritmos de detección y OCR visto en el capítulo 3 y el desarrollo de la aplicación ANPR visto en el apartado 4.1, se ha trabajado en el hardware de un ordenador pensado para aplicaciones de ingeniería con un alto uso computacional.

En una instalación de campo se puede trabajar con un hardware mucho más especializado para la aplicación, que permite dedicar toda la capacidad de procesamiento a aplicar los modelos de detección.

Existen tres elementos de hardware viables a la hora de implementar los algoritmos de detección en tiempo real: FPGA, PCL y Raspberry Pi 5.

FPGA (Field Programmable Gate Array).

Sin duda, las soluciones que más poder computacional ofrecen son aquellas basadas en FPGA.

Las FPGA son tecnologías reconfigurables que ofrecen una amplia variedad de configuraciones de arquitecturas, en las que se destaca una gran capacidad de realizar operaciones de tipo pipeline y un gran poder de computación en paralelo. [37].

Esta capacidad computacional en paralelo convierte a las FPGA en la solución industrial más empleada para realizar operaciones convolucionales en el análisis de imágenes. Además, las FPGA son relativamente baratas y fácilmente reprogramables. Se programan en un lenguaje de programación llamado VHDL (Hardware Description Language).

El problema principal con el uso de FPGA para este proyecto surge a la hora de tratar de implementar el código existente de la aplicación desarrollada.

No solo se han empleado librerías específicas de Python que no están disponibles en entornos de desarrollo basados en VHDL como Quartus II o Xilinx System Generator, sino que tener que implementar algoritmos y operaciones complejas a tan bajo nivel supone un gran reto en sí mismo.

Además, el tiempo que llevaría traducir el algoritmo de Python a VHDL es extremadamente alto.

Es por esta serie de motivos que se considera que la opción de embarcar el algoritmo ANPR en una FPGA escapa del alcance del proyecto, ya que requeriría de los mismos recursos o más que todo este proyecto en su conjunto.

PLC (Programmable Logic Computer).

Por otro lado, existen las soluciones de visión computacional basadas en PLC. Las características de flexibilidad, precisión y escalabilidad convierten al PLC en un estándar en la industria de la automatización industrial actual.

A lo largo de los años han surgido diferentes fabricantes que ofrecen sistemas de visión artificial basados en PLC, como por ejemplo Keyence o Siemens con los modelos de Simatic VS120.

Por desgracia, al tratar de implementar el programa desarrollado en este proyecto aparecen los mismos problemas que se han visto con las FPGA.

Puesto que los sistemas de control de PLC trabajan con lenguaje de contactos o lenguaje de texto estructurado, es necesario traducir el programa de Python que se ha desarrollado a uno de estos lenguajes para poder hacerlo funcionar con Keyence o Simatic VS120.

Tal y como se ha explicado con las FPGA, traducir el programa puede llevar varios cientos de horas de ingeniería, lo cual escaparía completamente del alcance del proyecto. [38]

Por suerte, existen algunos modelos de PLC basados en Linux, como el AXC F 2152. Esto permite ejecutar el programa de Python en un entorno PLC sin ningún problema, adaptando el programa a los requisitos del controlador. [39].

Figura 4.7: *PLC AXC F 2152 basado en Linux. De Phoenix Contact.*

Sin embargo, el verdadero factor que ha llevado a rechazar el uso de PLC para el proyecto es su elevado precio de mercado. Los modelos de PLC que pueden ofrecer un poder computacional suficiente para ejecutar el programa ANPR en tiempo real tienen un precio que ronda los 800 €.

Este precio haría inviable el proyecto como producto comercializable, como se verá más adelante en el capítulo 5.

Raspberry Pi 5.

Por último, tenemos las soluciones basadas en Raspberry Pi, un pequeño ordenador con un precio muy reducido y considerablemente potente que funciona con el sistema operativo de Linux.

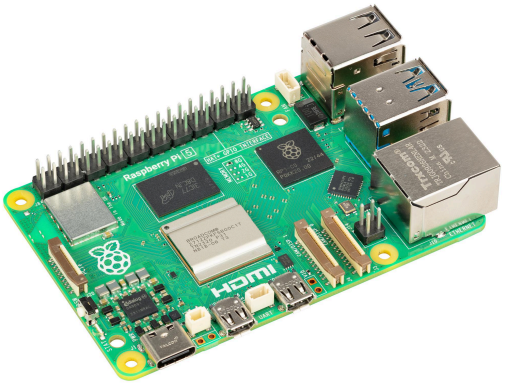


Figura 4.8: *Raspberry Pi 5 con 8 GB de RAM, de raspipc.es.*

Como se ha explicado anteriormente, se puede ejecutar el código de Python del proyecto en Linux sin ningún problema, adaptando un poco algunas características del programa.

Además la versión más reciente, Raspberry Pi 5, tiene un modelo de 8 GB de RAM por un precio de mercado de en torno a 130 €. Este precio extremadamente económico en conjunto con la simplicidad en la implementación de la aplicación ANPR desarrollada convierten a este tipo de solución de hardware en una de las más viables.

Decisión del hardware a emplear.

Puesto que el proyecto tal y como se ha desarrollado en el capítulo 3 no puede funcionar en FPGA, se descarta el uso de esta opción para el proyecto.

Por otro lado, los prohibitivos precios de los PLC industriales, pensados para ofrecer un alto nivel de precisión en la detección y emisión de señales de control, hacen económicamente inviable el uso de esta solución.

Sin duda la solución más económica y que permite una sencilla implementación del sistema ANPR desarrollado es la basada en Raspberry Pi. Por tanto, se va a emplear como hardware de ejecución de la aplicación ANPR.

Aun así, se ha de destacar que la FPGA es sin duda la mejor solución de hardware para la aplicación, pero requiere de cientos de horas de ingeniería para adaptar el código existente a VHDL.

4.3.2. Propuesta de solución física.

Una vez determinado el hardware que va a procesar la aplicación ANPR, es necesario definir brevemente el resto de los elementos físicos del sistema de visión. En este apartado se va a explicar brevemente la propuesta para la solución física del sistema ANPR.

La parte fundamental del sistema ANPR es la cámara. Como se ha visto detalladamente en el apartado 3.1.3, las propiedades de la cámara ya se han determinado con anterioridad.

La cámara debe ser híbrida, por lo que debe ser capaz de captar imágenes en color durante el día e imágenes en grises obtenidas mediante luz infrarroja durante la noche. Debe ser capaz de captar 10 fotogramas por segundo y debe tener un grado de protección IP65 como mínimo.

Existe una gran cantidad de cámaras que cumplen estas características y que se pueden obtener por precios en torno a los 150 €. Estos precios tan bajos se deben en gran medida al bajo requerimiento de fotogramas por segundo.

La cámara de vídeo que se propone utilizar es la IP DAHUA Bullet de 4 MP, la cual se puede obtener por 125 €.

Figura 4.9: *IP DAHUA Bullet de 4 MP de Dahua technology en Amazon.es.*

Para la conexión entre el sistema de captura de imagen y el sistema de procesamiento o interfaz de usuario se hará uso de cables de ethernet para distancias inferiores a 100 metros. En caso de requerir conexión a distancias más elevadas, se propone el uso de módulos Wi-fi Lora, aunque siempre se pueden instalar repetidores de cable ethernet.

Para fijar un modelo físico de barrera se ha determinado que la barrera automática con la que se va a trabajar sea de tipo pluma vehicular, aunque el sistema puede funcionar perfectamente con barreras de tipo puerta, de tipo cadenas y de tipo pivote sin ningún problema.

CAPÍTULO 5. ESTUDIOS ECONÓMICOS

En este capítulo se van a realizar una serie de estudios previos sobre el mercado de los sistemas ANPR para fijar un estándar del producto disponible en el mercado, así como comprender por qué triunfan algunos productos sobre otros.

Por otro lado, se va a especificar la propuesta de los productos comercializables, así como de los diferentes servicios asociados a la venta de los mismos.

Por último, se va a desarrollar un presupuesto generalizado para cada paquete de venta e instalación que se propone vender.

5.1. Estudios previos.

5.1.1. Productos y servicios observados en el mercado.

Tras una investigación previa del mercado de sistemas ANPR en barreras automáticas para recintos privados, se ha determinado que existen tres líneas de producto viables:

- Venta de cámaras ANPR con procesamiento integrado.
- Venta de licencias de software para emplear con cualquier cámara.
- Venta de barreras vehiculares con cámara y sistema ANPR integrados en el armario eléctrico de la barrera.

En realidad, estos tres nichos de mercado se pueden comprender de manera muy sencilla teniendo en cuenta la perspectiva del comprador.

Un cliente que ya dispone de una barrera automática, necesita comprar una cámara ANPR especializada que permita conectarse con la barrera para poder abrirla y cerrarla, además de tener conexión con una base de datos de matrículas permitidas.

Por otro lado, a un cliente que ya dispone de una barrera automática y de una cámara y que ha decidido instalar por primera vez o mejorar el software ANPR le interesa obtener una licencia de software para completar el sistema de accesos.

Por último, a un cliente que no dispone ni de cámara ni de barrera vehicular le interesa comprar un paquete completo con barrera, cámara y software ANPR. Además, por norma general los clientes necesitan almacenamiento local para los vídeos de las cámaras por seguridad o mejora de procesos, por lo que muchos fabricantes venden discos duros junto con las licencias de software.

5.1.2. Precios de venta observados de productos y servicios ANPR.

Aunque muchos fabricantes y distribuidores de sistemas ANPR prefieren no dar un precio fijo, en este apartado se va a estimar el precio de venta de los productos vistos en el apartado 5.1.1.

En muchas ocasiones el coste de un sistema ANPR debe calcularse para cada cliente en particular, puesto que los gastos de instalación o las necesidades varían según el entorno o tipo de instalación que desee el cliente.

Gracias a varios vendedores, como The Electric Shop, Metric Group, Snap Access and Security o Century Secure Group se ha podido estimar un precio de venta promedio para los distintos productos ANPR disponibles en el mercado. Además, se ha empleado un blog de Elliot Blackler (2023) en el que explica detenidamente el razonamiento tras los altos precios.

Los precios de las cámaras industriales con software ANPR integrado se encuentran entre 600 y 1.600 €, incorporando generalmente elementos de visión nocturna y un servicio de instalación.

Una licencia de software ANPR tiene un precio situado entre 400 y 1200 €, en función del número de licencias, de si las licencias incluyen soluciones personalizadas como interfaces de usuario adaptadas y de si se compra la licencia en conjunto con una cámara o no.

El precio de compra e instalación de una barrera automática se sitúa entre los 1.500 y 3.000 €. Si se suma el coste del sistema ANPR, el precio puede alcanzar hasta 5.000 €. [40].

En la tabla 5.1 se recogen los precios promedio observados para cada tipo de producto, sin contar los servicios asociados a la instalación.

Precio de venta de productos ANPR para accesos.	
Elemento	**Rango de precio / ud.**
Cámara	150 - 800 €
Licencia de software ANPR	400 - 1200 €
Cámara ANPR	600 - 1600 €
Barrera vehicular	1800 - 3000 €
Barrera con sistema ANPR	2900 - 5000 €

Tabla 5.1: *Rango de precios de productos ANPR para accesos, de Elliot Blackler (2023), en notechcontrol.com.*

5.2. Propuesta de productos y servicios.

Una vez determinado el mercado de los sistemas ANPR así como los precios y productos de la competencia existente, resta definir una propuesta de producto y mercado objetivo.

Además, se van a redactar una serie de presupuestos para los distintos paquetes de productos que se plantea vender.

Por último, se van a definir unos posibles precios de venta basándose en los presupuestos de cada paquete de productos.

5.2.1. Especificaciones del producto y servicio.

Tal y como se ha definido en los capítulos anteriores, los tipos de cliente que se buscan son aquellos que quieran instalar un sistema ANPR para una barrera vehicular que permita el acceso a un recinto industrial, con el objetivo de aportar seguridad, comodidad y un registro de entradas y salidas.

Observando el mercado de los sistemas ANPR para barreras de accesos, se puede observar que en muchos casos los clientes necesitan una solución particular.

Ya sea por ofrecer diversas soluciones de hardware y software personalizadas, las empresas que más han triunfado en el mercado han sido aquellas que disponen de una gran variedad de paquetes de productos y servicios asociados, como son por ejemplo Metric Group o The Electric Shop.

Por este motivo se va a apostar por esta estrategia. Esto significa que no se va a vender un tipo de producto o servicio en concreto, sino que la propuesta de los productos y servicios que se propone vender se van a agrupar en una serie de paquetes de ventas, con el fin de adaptarse a las necesidades del cliente.

Además, se va a tratar de innovar sobre el mercado ofreciendo un servicio de construcción de una caseta de vigilancia en conjunto con todos los productos que suelen ofrecer los fabricantes.

Los paquetes de productos y servicios que se propone vender son los siguientes:

- Licencia de software ANPR para emplear con hasta 4 cámaras. Incluye una interfaz de usuario personalizada y disco duro de almacenamiento.

- Venta e instalación de cámaras y licencia de software ANPR asociada a las mismas.

- Venta e instalación de barrera automática de tipo pluma vehicular.
- Venta e instalación de barrera automática de tipo pluma vehicular junto con cámaras y licencia ANPR.
- Venta e instalación de una caseta de vigilancia completamente equipada, barrera automática de tipo pluma vehicular, cámaras y licencia ANPR.

5.2.2. Presupuesto.

A continuación se va a estimar un presupuesto de los costes de compra e instalación para cada uno de los productos y servicios que se proponen vender.

Es importante dejar claro que en caso de tratar de crear una empresa que se dedique a vender estos productos, estos costes estimados de operación se pueden ver enormemente afectados.

Esto es debido a que no se han tenido en cuenta costes de gestión más allá de la parte de compra, transporte e instalación, obviando los costes de organización y gestión.

Por otro lado, una empresa especializada y bien organizada puede reducir en gran medida los costes de operación y de compra, por lo que los cambios sobre estos presupuestos teóricos no tienen por qué ser negativos.

Para explicar los costes asociados a cada paquete se va a partir desde el presupuesto de compra e instalación del paquete más completo, la venta e instalación de la caseta de vigilancia completamente equipada, barrera vehicular, cámaras y licencia ANPR, mostrado en la tabla 5.2.

Una vez definido este presupuesto para una instalación completa, se va a emplear para definir los presupuestos de los paquetes reducidos, eliminando aquellos costes que no correspondan a estos paquetes más pequeños.

El presupuesto para la instalación completa se encuentra desglosado en detalle en el anexo 1, considerando además una previsión de contingencias que puedan surgir durante la compra o instalación.

RESUMEN DE PRESUPUESTO PARA INSTALACIÓN COMPLETA	
CATEGORÍAS	**IMPORTE**
CAPÍTULO 1: COMPRA DE PRODUCTOS	
Barrera vehicular	2.723,46 €
Caseta de control	2.696,70 €
Sistema ANPR	870,52 €
TOTAL CAPÍTULO 1:	**6.290,68 €**
CAPÍTULO 2: OBRA CIVIL PARA INSTALACIÓN DE SISTEMA ANPR	
Legislación y permisos de obra	490,00 €
Mano de obra	600,00 €
Costes de ejecución de obra	682,68 €
TOTAL CAPÍTULO 2:	**1.772,68 €**
CAPÍTULO 3: SEGUROS	
Seguros de obra	295,88 €
SUBTOTAL (SIN CONTINGENCIAS):	**8.359,24 €**

Tabla 5.2: *Presupuesto de compra e instalación del paquete de productos más grande. Elaboración propia.*

Partiendo de este presupuesto generalizado, se van a determinar los costes asociados a cada uno de los paquetes reducidos, mostrados en la tabla 5.3.

Presupuesto de compra e instalación de paquetes de productos.	
Paquete de productos y servicios	**Presupuesto**
Licencia ANPR (4 cámaras) + Disco duro	404,90 €
2 Cámaras + Licencia ANPR + Disco duro + Instalación	870,52 €
2 Barreras vehiculares + Instalación	3822,02 €
2 Barreras vehiculares + 2 Cámaras + Licencia ANPR + Instalación	4992,54 €
Caseta de vigilancia + 2 Barreras vehiculares + 2 Cámaras + Licencia ANPR + Instalación	8.359,24 €

Tabla 5.3: *Presupuestos para cada paquete de productos. Elaboración propia.*

5.2.3. Precios de venta.

Una vez determinados los presupuestos para los distintos paquetes de productos, se va a determinar un precio para cada uno de los paquetes basándose en los precios observados en el mercado vistos en la tabla anterior del apartado 5.2.1.

Puesto que el volumen de ventas que se espera en este sector es muy pequeño, se ha de tratar de obtener un gran margen de beneficios en cada venta, siguiendo el estándar de precios del sector industrial.

Donde más márgen de beneficios se puede obtener es en la venta de la licencia de software, ya que realmente es la única parte de todos los paquetes en la que el precio no está determinado por ningún factor.

Precio de venta de venta de paquetes de productos y servicios.	
Paquete de productos y servicios	**Precio de venta**
Licencia ANPR (4 cámaras) + Disco duro	799 €
2 Cámaras + Licencia ANPR + Disco duro + Instalación	1399 €
2 Barreras vehiculares + Instalación	4599 €
2 Barreras vehiculares + 2 Cámaras + Licencia ANPR + Instalación	5999 €
Caseta de vigilancia + 2 Barreras vehiculares + 2 Cámaras + Licencia ANPR + Instalación	9499 €

Tabla 5.4: *Precio de venta de cada paquete de productos. Elaboración propia.*

Si el cliente lo desea, también se ofrece desarrollar un paquete personalizado incluyendo o eliminando servicios y ajustando el precio de manera consecuente, empleando los precios de los paquetes de la tabla anterior como referencia.

CAPÍTULO 6. CONCLUSIONES

Tras finalizar el desarrollo de 7 algoritmos distintos, de una aplicación ANPR, de una propuesta de embarcado físico y de los estudios económicos asociados a la venta de productos ANPR se va a desarrollar el capítulo de conclusiones.

Este capítulo abarca los posibles impactos socioeconómicos de este proyecto, las posibles líneas de mejora y posibles futuras implementaciones para el mismo. Por último, se van a redactar las conclusiones finales seguido de un breve apartado de autoevaluación.

6.1. Impactos medioambientales y socioeconómicos.

Hoy en día la tendencia en las grandes industrias es obtener la mayor cantidad de información de todos los procesos que tienen lugar en la misma, con el fin de optimizar los tiempos y proponer mejoras.

Por este motivo en este proyecto se busca obtener un registro de accesos y salidas de vehículos de transporte que acceden a un recinto industrial, para proveer a la empresa de información sobre la que optimizar los procesos de transporte, o simplemente aportar esta información al conjunto de datos de IoT.

Si se trabaja esta información de manera eficiente se puede llegar a proveer de un impacto económico positivo para la empresa en cuestión. Además, la aplicación ANPR desarrollada en este proyecto requiere de un operario de vigilancia que supervise y permita los accesos o salidas, generando puestos de trabajo.

Además, una optimización de los tiempos que pasa un vehículo en el recinto cargando y descargando material puede reducir perceptiblemente las emisiones de C02.

Entre los posibles impactos sociales atribuibles a este proyecto se destaca sobre todo la seguridad y control aportados en un acceso vehicular para un recinto privado, pudiendo emplearse esta información en futuras denuncias judiciales por robo o daños a la propiedad.

6.2. Trabajos futuros.

Una vez terminado el desarrollo del proyecto, existen muchas líneas de mejora posibles para los algoritmos y unos pasos claros a seguir para continuar con las líneas de desarrollo de este proyecto en caso de disponer de más tiempo o de más recursos.

Mejoras de rendimiento y precisión de los algoritmos.

En caso de disponer de más tiempo y recursos, muchos de los algoritmos se pueden optimizar de varias maneras para ofrecer unos resultados más veloces y precisos.

Como ya se ha visto anteriormente, el algoritmo de detección de matrículas basado en métodos tradicionales tiene problemas con los cambios en las condiciones ambientales.

Una mejora que sería muy interesante aplicar consiste en incorporar un algoritmo iterativo en el que se realice la búsqueda de una matrícula en la imagen alterando el conjunto de parámetros de procesado para cada iteración. Se espera que una mejora de este tipo ofrezca muy buen resultado puesto que los tiempos de ciclo de este algoritmo son extremadamente bajos.

En cuanto al algoritmo de detección de matrículas basado en pattern matching, el cual ha ofrecido los peores resultados, sería interesante ver la mejora en la precisión que puede aportar obtener imágenes en instalaciones físicas reales y emplearlas en el reconocimiento de patrones. De cualquier manera, se prevé que este algoritmo no será viable.

Por otro lado, los algoritmos basados en redes neuronales convolucionales pueden experimentar una mejora notable si se optimizan los bancos de imágenes de entrenamiento. Añadir más imágenes, especialmente una gran cantidad de imágenes donde se han encontrado fallos en las detecciones, puede llevar a optimizar aún más la precisión de estos algoritmos.

Otro punto que se puede mejorar es el método de evaluación de los algoritmos. En este proyecto se ha realizado un análisis sobre todos los algoritmos basándose en un banco de 50 imágenes sobre las que comprobar la eficiencia y velocidad.

Se ha elegido trabajar sobre 50 imágenes porque en los algoritmos muy dependientes de la configuración de los parámetros iniciales como el de métodos tradicionales o los de reconocimiento de patrones se puede llegar a tardar varias horas en evaluar estas 50 imágenes, mientras que en los algoritmos que no requieren preprocesado basados en redes neuronales se puede realizar este test en breves minutos.

Sería muy interesante obtener una gran cantidad de imágenes con las mismas condiciones ambientales, idealmente obtenidas desde el mismo lector de matrículas, para poder realizar un test mucho más amplio y extenso en ejemplos realistas.

De esta manera, se pueden realizar los tests sobre los algoritmos tradicionales y de reconocimiento de patrones sin alterar los parámetros iniciales.

Además, sería muy interesante realizar estos tests sobre el embarcado de software final, para obtener unos resultados de las características de la solución ANPR final.

Mejoras de la aplicación ANPR.

Tanto el algoritmo final como la aplicación ANPR, tal y como se han desarrollado en este proyecto, presentan un problema de seguridad muy serio. Nada impide que alguien aparezca con una impresión de una matrícula sobre papel con la información de una matrícula registrada. El sistema, tal cual está, es susceptible a este problema.

Aunque la aplicación está pensada para funcionar en conjunto con un operario de seguridad para proveer vigilancia constante sobre los accesos, en una instalación con un gran número de accesos muy separados se puede dar este problema de seguridad.

Para solucionar esto, se ha de replantear el proyecto desde el principio. Una posible solución es desarrollar un algoritmo de detección de vehículos, con el fin de determinar si existe un vehículo en la imagen antes de buscar una matrícula.

Otra posible solución es añadir un sensor de presencia para detectar que se encuentre presente un vehículo.

Otra posible mejora para la aplicación sería que el operario pudiese añadir el motivo de la entrada del vehículo al recinto desde la aplicación, para que quede reflejado en el registro de accesos junto a la hora de entrada y salida. También se puede añadir una función para que el operario pueda registrar una matrícula en la base de datos de vehículos permitidos.

Por último, se puede considerar añadir elementos de protección ante los fenómenos ambientales en la barrera automática.

Gracias a estos elementos de protección se puede reducir al mínimo los efectos de la lluvia sobre las cámaras. En un primer lugar no se ha considerado necesario añadir dichos elementos, ya que el coste de compra e instalación de estos elementos de protección puede ser demasiado elevado como para merecer la pena implementar esta mejora.

Futuras líneas de desarrollo.

En este proyecto se tomó la decisión de trabajar con YOLOv8 tras una breve investigación preliminar.

Una de las futuras líneas de desarrollo del proyecto es realizar un estudio comparativo entre los distintos modelos de redes neuronales disponibles en la actualidad, para obtener una perspectiva más clara de los modelos más eficientes para el reconocimiento de matrículas.

El siguiente paso para continuar el desarrollo de este proyecto será tratar de implementar la aplicación ANPR en una FPGA o un PLC.

Esto significa traducir y adaptar todo el código de Python a VHDL o texto estructurado, incorporando la detección mediante las redes neuronales convolucionales.

Por otro lado, otro de los pasos futuros del proyecto en el embarcado de hardware puede consistir en el diseño de un armario eléctrico especializado desde cero, como el mostrado en la figura 6.1.

Figura 6.1: *Unidad ANPR SIRAM-I Totem, de Bcn-Access.*

Este diseño deberá incluir todo el cableado eléctrico de alimentación y de comunicación, la cámara híbrida y las luces infrarrojas para iluminar durante la noche.

El objetivo de este diseño será ahorrar costes a largo plazo, ya que no será necesario comprar el armario especializado a otros fabricantes.

Por último, se pueden fabricar prototipos de este embarcado de software y hardware para probar el producto completo y determinar una viabilidad económica realista de este producto en el mercado, realizando un análisis más detallado de las debilidades y oportunidades reales del producto.

6.3. Conclusiones.

6.3.1. Conclusiones generales.

En este apartado se muestran las conclusiones generales obtenidas en este proyecto.

Se ha determinado mediante un estudio comparativo la viabilidad de los algoritmos basados en métodos tradicionales y en modelos de detección, mientras que el algoritmo basado en reconocimiento de patrones ha demostrado ser inviable para su uso.

En el estudio de los algoritmos de lectura de caracteres se ha determinado que tanto el algoritmo basado en reconocimiento de patrones como el basado en modelos de detección son viables.

Los algoritmos OCR desarrollados en este proyecto presentan una precisión y eficiencia comparable a las soluciones de código libre más populares, llegando a superarlas en eficiencia y precisión en el caso del algoritmo basado en redes neuronales.

Se han empleado en el algoritmo en cascada los algoritmos basados en inteligencia artificial final, puesto que reducen en gran medida los tiempos y costes de puesta en marcha en una instalación física real en comparación con el resto de algoritmos.

El algoritmo en cascada final ha ofrecido un 93% de precisión en la lectura de la matrícula y unos tiempos de ejecución promedio de 141 ms, reducido a 85 ms en caso de no detectar ninguna matrícula en la imagen captada.

La aplicación ANPR permite monitorizar los accesos y salidas en una barrera vehicular doble mediante una HMI. Esta interfaz contiene botones que permiten permitir o denegar un acceso al recinto, además de mostrar un registro de los vehículos que acceden o salen del mismo.

En el breve estudio de mercado se ha determinado que el proyecto es económicamente viable. Ya que es posible desarrollar un sistema ANPR funcional y robusto con componentes de bajo coste, se ofrecen de esta manera precios muy competitivos en comparación con los observados.

6.3.2. Autoevaluación y conclusiones personales.

Por último, se va a concluir este proyecto evaluando mi propio rendimiento de la forma más objetiva posible, además de aportar unas conclusiones desde una perspectiva más personal.

Al comienzo del proyecto, tras plantear los objetivos se desarrolló un cronograma a seguir donde se indicaba de manera arbitraria el tiempo en semanas que se consideraba que debía tomar cada tarea. A lo largo del proyecto, tanto algunos de los objetivos como la distribución de las tareas han cambiado en varias ocasiones.

La versión final del cronograma de las tareas, así como un registro de la cantidad de horas que se ha dedicado a cada una de ellas se muestran en la tabla 6.1.

CRONOGRAMA REAL									
TAREAS	**feb 5**	**feb 12**	**feb 19**	**feb 26**	**mar 4**	**mar 11**	**mar 18**	**mar 25**	**abr 1**
Investigar y estructurar el trabajo	15 h								
Desarrollar el cronograma y la justificación		5 h							
Preparar entrega intermedia			4 h						
Desarrollo del estado del arte				29 h					
Detección de matrículas - IA					34 h				
Detección de matrículas - Tradicional							28 h		
Detección de matrículas - Pattern Matching								23 h	
OCR - IA									23 h

TAREAS	**abr 8**	**abr 15**	**abr 22**	**abr 29**	**my 6**	**my 13**	**my 20**	**my 27**	**jun 3**
OCR - Pattern Matching	16 h								
OCR - IA código libre		14 h							
Algoritmo ANPR final		14 h							
APP ANPR			34 h						
Solución de hardware				13 h					
Estudios económicos				17 h					
Ejemplos de funcionamiento (vídeo)					8 h				
Conclusiones					6 h				
Bibliografía y terminar el documento					17 h				
Preparar la presentación							11 h		

Tabla 6.1: Cronograma final y horas dedicadas al proyecto. Elaboración propia.

Se ha dedicado al proyecto un total de 311 horas a lo largo de cuatro meses, superándose levemente la marca de tiempo de 300 horas recomendada por la universidad para el desarrollo del proyecto.

Tras analizar brevemente el cronograma, se puede ver que se han cumplido todos los objetivos propuestos y las horas que se han dedicado al desarrollo del proyecto se han distribuido de manera bastante uniforme, a excepción de las primeras semanas de abril donde se aprovecharon los días festivos para dedicarle más horas.

También se ha dedicado mucho menos tiempo al proyecto en las últimas semanas de mayo, donde se ha compaginado el proyecto con el comienzo del trabajo a jornada completa.

Es importante destacar que se ha desarrollado el proyecto al mismo tiempo que se han cursado las asignaturas del último año, incluyendo la realización de las prácticas universitarias.

Desarrollar este proyecto ha sido un gran reto tanto técnico como personal, donde he aprendido una enorme cantidad de conocimientos sobre visión artificial, redes neuronales, Python, normativas vehiculares, gestión de bancos de imágenes y el mundo de los sistemas ANPR en general.

Puesto que se ha podido desarrollar un algoritmo funcional con recursos limitados y soluciones de código libre, se ha demostrado que los problemas del lector de matrículas de mi recinto residencial son inexcusables en una solución ANPR comercial desarrollada con muchos más recursos y más tiempo.

En lo personal, he disfrutado cada minuto dedicado a este proyecto. Al adentrarme en profundidad en el mundo de la visión artificial y de la automatización industrial, he conseguido incrementar todavía más mi interés, fascinación y pasión por esta increíble rama de la ingeniería.

BIBLIOGRAFÍA

[1] Shaip-Admin. (2023, 27 abril). Reconocimiento Automático de Matrículas (ANPR) | Shaip. De https://es.shaip.com/blog/automatic-number-plate-recognition-anpr/.

[2] ESSistemas® Spain. (s. f.). Cámaras ANPR / LPR. Recuperado 12 de marzo de 2024, de https://essistemas.com/es/14-camaras-anpr-lpr.

[3] ¿Qué es la visión artificial? - Explicación de la IA y el aprendizaje automático de imágenes - AWS. (s. f.). Amazon Web Services, Inc. Recuperado 12 de marzo de 2024, de https://aws.amazon.com/es/what-is/computer-vision/.

[4] González, Woods (1996). *Tratamiento digital de imágenes*. Maravall, Daniel (1993). *Reconocimiento de formas y visión artificial*. De La Escalera, Arturo (2001). *Visión por Computador*. González Jiménez, Javier (1999). *Visión por Computador*. Por Ortega Sanz, Diego (Ed.), *Sistemas de percepción en robótica*. De la documentación asociada al curso de la Universidad Europea de Madrid (2021).

[5] De La Fuente, E., & Trespaderne, F. M. (2013). VISIÓN ARTIFICIAL INDUSTRIAL. Procesamiento de imágenes para visión automática y robótica. Universidad de Valladolid.

[6] Igual, J. (2016, 17 enero). Todo sobre el Histograma en Fotografía - foto igual. Foto Igual. Recuperado 12 de marzo de 2024, de https://fotoigual.com/tutoriales/todo-sobre-el-histograma-en-fotografia/.

[7] Spurgeon, W. (2021). Introduction to Pattern Matching. Recuperado 25 de marzo de 2024, de https://www.clearview-imaging.com/en/blog/introduction-to-pattern-matching.

[8] IA y machine learning: diferencia entre inteligencia artificial y machine learning. AWS. (s. f.). Amazon Web Services, Inc. Recuperado 25 de marzo de 2024, de https://aws.amazon.com/es/compare/the-difference-between-artificial-intelligence-and-machine-learning/.

[9] Redes neuronales y aprendizaje profundo: diferencia entre campos de la inteligencia artificial - AWS. (s. f.). Amazon Web Services, Inc. Recuperado 25 de marzo de 2024, de https://aws.amazon.com/es/compare/the-difference-between-deep-learning-and-neural-networks/.

[10] Shah, Bhoomi & Bhavsar, Hetal & Bhavsar, Hetal. (2021). Overview of Deep Learning in Food Image Classification for Dietary Assessment System. 10.1007/978-981-16-0730-1_18.

[11] Prakash, J. (2021, 18 octubre). Non Maximum Suppression: Theory and Implementation in PyTorch. LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow With Code, & Tutorials. Recuperado 28 de marzo de 2024, de https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/.

[12] OpenCV. (s. f.). OpenCV - Open Computer Vision Library. Recuperado 25 de marzo de 2024, de https://opencv.org/.

[13] imutils. (2021, 15 enero). PyPI. Recuperado 27 de marzo de 2024, de https://pypi.org/project/imutils/.

[14] tkinter — Python interface to Tcl/Tk. (2024, 15 mayo). Python Documentation. Recuperado 20 de abril de 2024, de https://docs.python.org/3/library/tkinter.html.

[15] Taleti, R. (2021, 13 diciembre). YOLO: You only look once | Object Detection - towards data science. Medium. https://towardsdatascience.com/yolo-you-only-look-once-f05c054a06b4.

[16] Dwyer, B., Nelson, J., Hansen, T., et. al. (2024). Roboflow (Version 1.0) [Software]. Available from https://roboflow.com. computer vision.

[17] Coceurope. (2024, 7 enero). Understanding European license plates - CdS Europe. CdS Europe Complete Datasheet. Recuperado 25 de marzo de 2024, de https://coceurope.eu/blog/understanding-european-car-plates/.

[18] C, C. (2024, 24 marzo). Matrículas de Estados Unidos. Matriculasdelmundo. Recuperado 28 de marzo de 2024, de https://matriculasdelmundo.com/estados-unidos.html.

[19] Kustermann, M. (s. f.). License Plates of the World. Recuperado 28 de marzo de 2024, de http://www.worldlicenseplates.com/.

[20] Number Plate Fonts of Australia and New Zealand. (s. f.). Recuperado 25 de marzo de 2024, de https://www.leewardpro.com/articles/licplatefonts/licplate-fonts-aust.html.

[21] Kusumadewi, I., Sari, C. A., Setiadi, D. R. I. M., & Rachmawanto, E. H. (2019). License Number Plate Recognition using Template Matching and Bounding Box Method. Journal Of Physics: Conference Series, 1201(1), 012067. https://doi.org/10.1088/1742-6596/1201/1/012067.

[22] Kasper-Eulaers, Margrit & Hahn, Nico & Berger, Stian & Sebulonsen, Tom & Myrland, Øystein & Kummervold, Per. (2021). Short Communication: Detecting Heavy Goods Vehicles in Rest Areas in Winter Conditions Using YOLOv5. Algorithms. 14. 114. 10.3390/a14040114.

[23] Huallpa, Elias & Macalupu, Abel & Luque, Jorge & Sánchez-Garces, Jorge. (2023). Determinación del mejor algoritmo de detección de matrículas en ambientes controlados y no controlados. RISTI - Revista Ibérica de Sistemas y Tecnologías de Información. 83-99. 10.17013/risti.49.83-99.

[24] Navacerrada, J. (2017). *Sistema de detección de matrículas con Open CV* [Universidad Politécnica de Madrid]. https://oa.upm.es/51869/1/TFG_JORGE_NAVACERRADA.pdf.

[25] Ley Orgánica 3/2018, de “5 de diciembre”, de Protección de Datos Personales y garantía de los derechos digitales. Boletín Oficial del Estado, *número 294*, de “06/12/2018”. https://www.boe.es/eli/es/lo/2018/12/05/3/con.

[26] Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo, de “27 de abril de 2016”, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos). Diario Oficial de la Unión Europea, *número 119*, de “4/4/2016”. DOUE-L-2016-80807.

[27] Ley Orgánica 15/1999, de “13 de diciembre”, de Protección de Datos de Carácter Personal. Boletín Oficial del Estado, *número 298,* de “14/12/1999”. https://www.boe.es/eli/es/lo/1999/12/13/15/con.

[28] Agencia Española de Protección de Datos | AEPD. (s. f.). Recuperado 25 de marzo de 2024, de https://www.aepd.es/.

[29] Reglamento (UE) 1003/2010 de la Comisión, de “8 de noviembre de 2010”, relativo a los requisitos para la homologación de tipo del emplazamiento y la instalación de las placas de matrícula traseras en los vehículos de motor y sus remolques y por el que se desarrolla el Reglamento (CE) no 661/2009 del Parlamento Europeo y del Consejo, relativo a los requisitos de homologación de tipo referentes a la seguridad general de los vehículos de motor, sus remolques y sistemas, componentes y unidades técnicas independientes a ellos destinados. *Número 291*.
https://www.boe.es/doue/2010/291/L00022-00030.pdf.

[30] Reglamento de Ejecución (UE) 2021/535 de la Comisión, de “31 de marzo de 2021”, por el que se establecen disposiciones de aplicación del Reglamento (UE) 2019/2144 del Parlamento Europeo y del Consejo relativas a los procedimientos uniformes y las especificaciones técnicas para la homologación de tipo de los vehículos y de los sistemas, componentes y unidades técnicas independientes destinados a dichos vehículos, en lo que respecta a sus características generales de construcción y seguridad. *Número 117*, con DOUE - L - 2021 - 80420.

[31] International Organization for Standardization. (2016). *Automatic identification and data capture techniques* (ISO/IEC 30116:2016).

[32] Real Decreto 885/2020, de “6 de octubre”, por el que se establecen los requisitos para la comercialización y puesta en servicio de placas de matrícula para vehículos de motor y remolques, y por el que se modifica el Reglamento General de Vehículos, aprobado por el Real Decreto 2822/1998, de 23 de diciembre. Boletín Oficial del Estado, *número 280*, de “23/08/2020”.
https://www.boe.es/eli/es/rd/2020/10/06/885.

[33] Nicknochnack. (2020). GitHub - nicknochnack/ANPRwithPython. GitHub. Recuperado 26 de marzo de 2024, de https://github.com/nicknochnack/ANPRwithPython.

[34] Akhtar Khan, S. (2023, 25 agosto). How to mask an image in OpenCV Python? Recuperado 26 de marzo de 2024, de https://www.tutorialspoint.com/how-to-mask-an-image-in-opencv-python.

[35] Computervisioning. (2023). GitHub - computervisioning object tracking yolov8 native: Yolov8 object tracking 100% native | Object detection | Computer vision tutorial. GitHub. Recuperado 25 de marzo de 2024, de https://github.com/computervisioneng/object-tracking-yolov8-native.

[36] Free fonts download. (s. f.). Download Free Font - Free Fonts Download. Recuperado 3 de abril de 2024, de https://freefontsdownload.net/free-matricula-espanola-font-124595.htm.

[37] Moctezuma, Juan & Arias-Estrada, Miguel & Parra, R.. (2008). ARQUITECTURA FPGA PARAMETRIZABLE PARA CONVOLUCIÓN DE IMÁGENES.

[38] Rothong, Nuttapon & Chinakunwiphat, Pawanrat & Chainoi, Sudarat & Butsanlee, Borihan. (2023). Design of PLC-Integrated Object Detection for Categorizing Conveyor. 130-134. 10.1109/RI2C60382.2023.10356010.

[39] AXC F 2152 - Controller - 2404267 | Phoenix Contact. (s. f.). Recuperado 6 de mayo de 2024, de https://www.phoenixcontact.com/en-in/products/controller-axc-f-2152-2404267.

[40] Blackler, E. (2023, 12 abril). How Much Does an ANPR System Cost? (2023 Guide). Nortech. Recuperado 8 de mayo de 2024, de https://blog.nortechcontrol.com/anpr-system-cost.

ANEXOS

ANEXO 1: PRESUPUESTO DESGLOSADO.

En este anexo se muestra el presupuesto desglosado de los costes de compra e instalación asociados a la venta del paquete de productos más completo.

Como se ha explicado anteriormente en el apartado 5.2.2 de la memoria, el presupuesto asociado al resto de paquetes de productos se obtiene reduciendo el presupuesto del paquete más completo, eliminando aquellos capítulos que no sean necesarios. Por tanto, únicamente se va a mostrar el desglose del paquete completo.

PRESUPUESTO PARA INSTALACIÓN COMPLETA DESGLOSADO				
CATEGORÍAS	**DESCRIPCIÓN**	**CANT.**	**PRECIO / UD**	**IMPORTE**
CAPÍTULO 1: COMPRA DE PRODUCTOS				
Barrera vehicular				**2.723,46 €**
Barrera automática tipo pluma vehicular	Barrera automática electromecánica PARK 30 APRIMATIC a 24V. Pluma de 3m de longitud.	2	1.230,73 €	2.461,46 €
Iluminación	Farola LED 60W Villa Calle Alumbrado Público IP65	2	131,00 €	262,00 €
Caseta de control				**2.696,70 €**
Caseta de vigilancia (Compra)	Módulo prefabricado de caseta de vigilancia de ABC Modular. Sup. útil: 10,15 m² Dimensiones: 4,16×2,44×2,83 m.	1	2.100,00 €	2.100,00 €
PC con Pantalla Dell 790 I5 | 4GB | 250 HD | W10 | 22"	PC para HMI y actualizar las matrículas permitidas.	1	179,00 €	179,00 €
Cableado ethernet	Transmisión de datos con las cámaras.	50 m	6,46 € / m	64,64 €
Cableado RVK	Cable Eléctrico rvk 3 hilos de 4 mm2, Baja tensión. Apto para exteriores.	100 m	2,13 € / m	213,00 €
Protector de cables para carretera	Protector de cables para la superficie de la carretera. Hasta 8890 kg. 2×0.1×0,032 m.	2	70,03 €	140,06 €
Sistema ANPR				**870,52 €**
Costes de ingeniería	Horas de ingeniería dedicadas al desarrollo de la interfaz de usuario personalizada.	20 h	15 € / h	300,00 €

Cámara de vídeo híbrida	Cámara de vídeo IP DAHUA Bullet 4Mp Ethernet (DH-IPC-HFW2431TP). Con visión nocturna infrarroja.	2	124,21 €	248,42 €
Soporte de acero para cámaras	Soporte de acero para cámaras.	2	93,00 €	186,00 €
Raspberry Pi 5 (8Gb)	Pequeño ordenador basado en Linux. Eficiente para detección de objetos en tiempo real.	1	131.53 €	131.53 €
Disco duro SATA	Disco duro Seagate Barracuda ST6000DM003 6TB para almacenar vídeo e imágenes.	1	104,90 €	104,90 €
Cableado de alimentación	Adaptadores de tensión y corriente.	1	31,20 €	31,20 €
TOTAL COMPRA DE PRODUCTOS				**6.290,68 €**
CAPÍTULO 2: OBRA CIVIL PARA INSTALACIÓN DE SISTEMA ANPR				
Legislación y permisos de obra				**490,00 €**
Solicitud de permiso de obras menores	Permiso de obras menores (< 20.000€) requerido para remodelar un terreno existente.	1	490,00 €	490,00 €
Mano de obra				**600,00 €**
Encargado de obra	Capataz especializado encargado de la obra (2 Días)	1	12 € / h	192,00 €
Operario	Trabajador para la obra (2 Días)	2	9 € / h	288,00 €
Indumentaria de seguridad	Cascos, botas y chalecos para obra.	3	40,00 €	120,00 €
Costes de ejecución de obra				**682,68 €**
Caseta de vigilancia (Transporte e instalación)	Módulo prefabricado de caseta de vigilancia. Sup. útil: 10,15 m². Dimensiones: 4,16×2,44×2,83 m. 2 - 4 horas de instalación.	1	370,00 €	370,00 €
Materiales de construcción	Cementos, tornillería, soportes y extras.	1	100,00 €	200,00 €
Alquiler maquinaria	Martillo neumático, taladros, pistola neumática, etc.	1	11,85 € / día	23,70 €
Alquiler furgoneta	Furgoneta para transporte de bienes a instalar y recogida de escombros.	1	35 € / día	70,00 €
Pintura blanca	Pintura para marcas en el asfalto	2	9,49 €	18,98 €

TOTAL OBRA CIVIL PARA INSTALACIÓN DE SISTEMA ANPR				**1.772,68 €**
CAPÍTULO 3: SEGUROS				
Seguros de obra				**295,88 €**
Seguro de Responsabilidad Civil en Construcción.	Póliza de seguros CESCE.	1	275,00 €	275,00 €
Seguro de Accidentes de Convenio para Construcción.	Seguro ASISA Accidentes - Plus Tipo 1.	1	20,88 €	20,88 €
TOTAL SEGUROS				**295,88 €**
SUBTOTAL (SIN CONTINGENCIAS):				**8.359,24 €**
CAPÍTULO 4: PREVISIÓN DE CONTINGENCIAS				
Contingencias				**1.000,00 €**
Provisión de retrasos en la ejecución de la obra	Incremento presupuesto obra civil	1	500,00 €	500,00 €
Provisión de fallo en productos	Incremento del presupuesto de compra de productos	1	500,00 €	500,00 €
TOTAL PREVISIÓN DE CONTINGENCIAS				**1.000,00 €**
TOTAL (CON CONTINGENCIAS):				**9.359,24 €**

ANEXO 2: CÓDIGO DESARROLLADO.

En este anexo se muestra todo el código en Python 3.9 de cada uno de los algoritmos desarrollados en este proyecto y de la aplicación ANPR final, todo ello comentado y completamente funcional.

Este anexo está dedicado a cualquier futuro programador interesado en comprender, implementar o expandir dichos algoritmos.

En caso de surgir problemas o preguntas a la hora de implementar los algoritmos, invito al lector a contactar conmigo para una consulta directa a través de mi dirección de correo electrónico:

pablodepaz99@gmail.com

En caso de emplear este código en futuros trabajos académicos, sería de agradecer que se citase correctamente este proyecto en la bibliografía. Muchas gracias de antemano y espero que este anexo resulte de gran utilidad.

2.1. Algoritmo de detección basado en métodos tradicionales.

```
######################################################################
# ALGORITMO DE DETECCIÓN DE MATRÍCULAS BASADO EN MÉTODOS TRADICIONALES
# V0.9 #


# PABLO JAVIER DE PAZ
######################################################################
# IMPORTANTE: Esta versión del algoritmo requiere definir MUCHAS condiciones iniciales
# específicas para cada imagen, como los puntos de la transf. afín, la máscara a emplear, etc.
# Estos parámetros se muestran a continuación, para poder ajustarlos rápidamente.
######################################################################
# LIBRERÍAS
######################################################################
import cv2
import imutils
import numpy as np
import time
import math

######################################################################
# VARIABLES
######################################################################
```

```
# Elegir imagen inicial
path = 'imagenesTEST/Coche5.jpg'
#path = 'imagenesTEST/Camion.jpg'
#path = 'imagenesTEST/Furgoneta.jpg'

# Elegir máscara
path2 = 'mascaras/MascaraCircular1.jpg'
#path2 = 'mascaras/MascaraCircular2.jpg'
#path2 = 'mascaras/MascaraCuadrada1.jpg'
#path2 = 'mascaras/MascaraCuadrada2.jpg'

# Asignar puntos afines (específicos para cada instalación física)
P1a, P2a, P3a = [152, 134], [253, 139], [150, 158]
P1b, P2b, P3b = [0, 0], [300, 0], [0, 150]

# Parámetros de Canny
ParamCanny1 = 0
ParamCanny2 = 200

# Iteraciones de dilatación/erosión y potencia de dilatacion/erosión
potDilate = 2  # (Mínimo 2)
itDilate = 2   # Default 1-2
potErode = 2   # (Mínimo 2)
itErode = 1    # Default 1

# Parámetro de epsilon en el aproxPolyDp
paramEpsilon = 0.04

# Parámetros de tamaño de la matrícula. Cambian según el tamaño de la
imagen fuente y cambian
# según el país de procedencia (ej. las matrículas en EEUU tienen dimensiones
distintas que en Europa)
UmbrMaxVer = 10  # Umbral máximo vertical de diferencia entre longitudes de
rectas iguales 2 a 2
UmbrMaxHor = 10  # Umbral máximo horizontal de diferencia entre longitudes
de rectas iguales 2 a 2
DistMinVer = 20  # Longitud mínima en píxeles de los laterales de la matrícula
DistMinHor = 40  # Longitud máxima en píxeles de la parte superior e inferior
de la matrícula

#############################
# MAIN #
#############################

# Se lee el tiempo del comienzo
T1 = time.time()

# Se lee la imagen y sus características
imagen = cv2.imread(path)
```

```
filas, columnas, canal = imagen.shape

# Imagen a grises
grises = cv2.cvtColor(imagen, cv2.COLOR_RGB2GRAY)

# Se definen los puntos para la transformación afin
pts1 = np.float32([P1a, P2a, P3a])
pts2 = np.float32([P1b, P2b, P3b])

# Se mueve el origen de la transformación hasta la esquina superior derecha.
# El motivo: La imagen resultante comienza desde ese primer punto, es para
poder representar la imagen completa.
pts1 = pts1 - np.float32([[152, 134],[152, 134],[152, 134]])

# Se define la matriz de transformación afin
Matriz = cv2.getAffineTransform(pts1, pts2)

# Se aplica la transformacion afin
imagen_afin = cv2.warpAffine(grises, Matriz,(1500, 1500))
imagen_afin = cv2.resize(imagen_afin, (columnas, filas))

# Se elige y se lee la mejor máscara
mascara_preprocesado = cv2.imread(path2, cv2.IMREAD_GRAYSCALE)

# Se ajusta el tamaño de la mascara a la imagen
Xmask = int(imagen_afin.shape[0])
Ymask = int(imagen_afin.shape[1])
mascara_preprocesado = cv2.resize(mascara_preprocesado, (Ymask, Xmask),
interpolation=cv2.INTER_LINEAR)
ret, mascara_preprocesado = cv2.threshold(mascara_preprocesado,
thresh=180, maxval=255, type=cv2.THRESH_BINARY)

# Se aplica la máscara sobre la imagen
imagen_mascara = cv2.bitwise_and(imagen_afin, imagen_afin,
mask=mascara_preprocesado)

# Filtro de reduccion de ruido
grises_filtrada = cv2.bilateralFilter(imagen_mascara, 11, 17, 17)

# Se realiza un leve blur para resaltar bordes
blurred = cv2.GaussianBlur(grises_filtrada, (5, 5), 0)

# Se emplea el buscador de bordes CANNY
bordes = cv2.Canny(blurred, ParamCanny1, ParamCanny2)

# Se realiza un dilate para reforzar las líneas de Canny y homogeneizar bordes
kernel = np.ones((potDilate, potDilate), np.uint8)
dilatada = cv2.dilate(bordes, kernel, iterations=itDilate)
```

```
# Se realiza un erode para reforzar las líneas de Canny y homogeneizar bordes
kernel = np.ones((potErode, potDilate), np.uint8)
erosionada = cv2.erode(dilatada, kernel, iterations=itErode)

# Se buscan los contornos cerrados del rectangulo de la matricula en la imagen
keypoints = cv2.findContours(erosionada.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
contornos = imutils.grab_contours(keypoints)
contornos = sorted(contornos, key=cv2.contourArea, reverse=True)[:10]

ubicacion = None
for i in range(len(contornos)):
  # Se simplifican los contornos cerrados detectados mediante la función
approxPolyDP
  epsilon = paramEpsilon * cv2.arcLength(contornos[i], True)
  approx = cv2.approxPolyDP(contornos[i], epsilon, True)

  # Se comprueba si alguno de los contornos detectados tiene 4 vértices
  if len(approx) == 4:
    P1, P2, P3, P4 = approx[0][0], approx[1][0], approx[2][0], approx[3][0]
    #print(P1, P2, P3, P4)

    # Se calculan las longitudes de cada recta
    dist1, dist2, dist3, dist4 = math.dist(P1, P2), math.dist(P2, P3),
math.dist(P3, P4), math.dist(P4, P1)

    # Se define una variable rectasiguales que da una puntuacion de similitud
entre rectas 2 a 2
    rectasiguales1 = abs(dist1 - dist3)
    rectasiguales2 = abs(dist2 - dist4)
    #print(rectasiguales1, rectasiguales2)

    # Se comprueba si las rectas paralelas son iguales, y si lo son se
comprueba si son de unas dimensiones
    # minimas y arbitrarias, es probable que sea necesario cambiarlas también
para cada sistema.
    if rectasiguales1 < UmbrMaxVer and rectasiguales2 < UmbrMaxHor and
dist1 > DistMinVer and dist2 > DistMinHor:
      ubicacion = approx
      break

# Se visualizan los pasos mediante los siguientes comandos
mascara_matricula = np.zeros(imagen_afin.shape[:2], np.uint8)

# En caso de que no se realice una detección, se evitan las siguientes líneas
de código (que hacen que el programa
# deje de funcionar)
if ubicacion is not None:
```

```
  imagen_mascara_final = cv2.drawContours(mascara_matricula, [ubicacion],
0, 255, -1)
  imagen_mascara_final = cv2.bitwise_and(imagen_afin, imagen_afin,
mask=mascara_matricula)

  # Se recorta la matricula desde la imagen original
  (x, y) = np.where(mascara_matricula == 255)
  (x1, y1) = (np.min(x), np.min(y))
  (x2, y2) = (np.max(x), np.max(y))
  imagen_final = imagen_afin[x1:x2+1, y1:y2+1]

#Se calcula el tiempo total elipsado
T2 = time.time()
print(T2-T1)

cv2.imshow('Deteccion', imagen)
cv2.waitKey(0)
cv2.imshow('Deteccion', grises)
cv2.waitKey(0)
cv2.imshow('Deteccion', imagen_afin)
cv2.waitKey(0)
cv2.imshow('Deteccion', mascara_preprocesado)
cv2.waitKey(0)
cv2.imshow('Deteccion', imagen_mascara)
cv2.waitKey(0)
cv2.imshow('Deteccion', bordes)
cv2.waitKey(0)
cv2.imshow('Deteccion', dilatada)
cv2.waitKey(0)
cv2.imshow('Deteccion', erosionada)
cv2.waitKey(0)
cv2.imshow('Deteccion', mascara_matricula)
cv2.waitKey(0)

if ubicacion is not None:
  cv2.imshow('Deteccion', imagen_mascara_final)
  cv2.waitKey(0)
  cv2.imshow('Deteccion', imagen_final)
  cv2.waitKey(0)
  cv2.imwrite('MatriculaDetectada.jpg', imagen_final)

cv2.destroyAllWindows()
```

2.2. Algoritmo de detección basado en reconocimiento de patrones.

```
#####################################################################
# ALGORITMO DE DETECCIÓN DE MATRÍCULAS BASADO EN RECONOCIMIENTO DE PATRONES
# V0.4 #

# PABLO JAVIER DE PAZ
#####################################################################
# LIBRERÍAS #
#####################################################################
import cv2
import os
import numpy as np
import time
#####################################################################
# VARIABLES
#####################################################################
# Asignar puntos afines (específicos para cada instalación física)
P1a, P2a, P3a = [152, 134], [253, 139], [150, 158]
P1b, P2b, P3b = [0, 0], [300, 0], [0, 150]

# Se establecen las rutas de las imágenes y de las templates
path = 'imagenesTEST/Coche5.jpg'
path_templates = 'templates/recortadas'

# Se establece el threshold de detección
threshold = 0.45

# Tamaño de la matrícula en la imagen
h2, w2 = 30, 100
#####################################################################
# MAIN #
#####################################################################
# Se empieza a contar el tiempo
T1 = time.time()

# Se carga la imagen sobre la que trabajar
imagen = cv2.imread(path)
filas, columnas, canal = imagen.shape

# Se convierte la imagen a grises
grises = cv2.cvtColor(imagen, cv2.COLOR_RGB2GRAY)

# Se definen los puntos para la transformación afin
pts1 = np.float32([P1a, P2a, P3a])
pts2 = np.float32([P1b, P2b, P3b])

# Se mueve el origen de la transformación hasta la esquina superior derecha.
```

```
# El motivo: La imagen resultante comienza desde ese primer punto, es para
poder representar la imagen completa.
pts1 = pts1 - np.float32([[152, 134], [152, 134], [152, 134]])

# Se define la matriz de transformación afín
Matriz = cv2.getAffineTransform(pts1, pts2)

# Se aplica la transformacion afín
imagen_afin = cv2.warpAffine(grises, Matriz, (1500, 1500))
imagen_afin = cv2.resize(imagen_afin, (columnas, filas))

# Se cargan las templates de las matrículas sobre una lista
template_list = os.listdir(path_templates)
print(len(template_list))
templates = []
templates_grises = []
templates_escaladas = []
for i in range(len(template_list)):
  # Se cargan las templates en una lista

templates.append(cv2.imread(os.path.join(path_templates+'/'+template_list[i])))
  # Se convierte a grises todas las matrículas
  templates_grises.append(cv2.cvtColor(templates[i],
cv2.COLOR_RGB2GRAY))
  # Se escalan todas las matrículas hasta el tamaño observado en la
instalación
  templates_escaladas.append(cv2.resize(templates_grises[i], (w2, h2),
interpolation=cv2.INTER_LINEAR))

# Se realiza el template matching y se extraen los puntos coincidentes
localizacion = []
puntos_crudos = []
for i in range(len(template_list)):
  deteccion = cv2.matchTemplate(imagen_afin, templates_escaladas[i],
cv2.TM_CCOEFF_NORMED)
  localizacion = np.where(deteccion >= threshold)
  print(i)
  for pt in zip(*localizacion[::-1]):
    puntos_crudos.append(pt)

print(puntos_crudos)

imagen_final = cv2.cvtColor(imagen_afin, cv2.COLOR_GRAY2RGB)

for i in range(len(puntos_crudos)):
  top_left = puntos_crudos[i]
  bottom_right = (top_left[0] + w2, top_left[1] + h2)
  cv2.rectangle(imagen_final, top_left, bottom_right, (255, 0, 0), 2)
```

```
print(time.time()-T1)

cv2.imshow("imagen", imagen_afin)
cv2.imshow("matricula", templates_grises[59])
cv2.waitKey(0)

cv2.imshow("imagen", imagen_final)
cv2.waitKey(0)
```

2.3. Algoritmo de detección basado en modelos de detección.

```
#####################################################################
# ALGORITMO DE DETECCIÓN DE MATRÍCULAS BASADO EN MODELOS DE DETECCIÓN
# V1.4 #


# PABLO JAVIER DE PAZ
#####################################################################
# LIBRERÍAS
#####################################################################
from ultralytics import YOLO
import cv2
import numpy as np
import time
#####################################################################
# VARIABLES
#####################################################################
# Se carga el modelo entrenado
modelo = YOLO('DetMatriculasV1.pt')

# Apertura de imagen
path = 'imagenesTEST/Coche2.jpg'
# path = 'imagenesTEST/Camion.jpg'
# path = 'imagenesTEST/Furgoneta.jpg'
# path = 'imagenesTEST/MatriculaCerca.jpg'
# path = 'imagenesTEST/OsoCanada.jpg'

#####################################################################
# MAIN
#####################################################################
imagen = cv2.imread(path)

# Se aplica el modelo de la red neuronal
resultados = modelo.predict(imagen, conf=0.2, max_det=1)[0]
T1 = time.time()
# Se dibuja la bounding box sobre la imagen original, para ver lo que se va a recortar
boundingbox = resultados.plot(line_width=1)

# Se extraen de los resultados las coordenadas de la boundingbox de la detección,
# en formato XY,XY (esquinas de la bounding box)
XYXY = resultados.boxes.xyxy
X1bb = int(XYXY[0, 0].item())
Y1bb = int(XYXY[0, 1].item())
X2bb = int(XYXY[0, 2].item())
Y2bb = int(XYXY[0, 3].item())
```

```
# Se crea una máscara vacía donde se cambia a negro todos los pixeles menos los de la boundingbox,
# para ver con mas claridad la imagen
mascara_matricula = np.zeros(resultados.orig_img.shape[:2], np.uint8)
mascara_matricula[Y1bb:Y2bb, X1bb:X2bb] = 255
masked_imagen = cv2.bitwise_and(resultados.orig_img, resultados.orig_img, mask=mascara_matricula)

# Se recortan el resto de pixeles
imagen_final = resultados.orig_img[Y1bb:Y2bb, X1bb:X2bb]

cv2.imwrite("matriculadetectada2.jpg", imagen_final)
# Se mide el tiempo de inferencia
T2 = time.time()
print(T2-T1)
# Se muestran los resultados
cv2.imshow('imshow', boundingbox)
cv2.waitKey(0)
cv2.imshow('imshow', masked_imagen)
cv2.waitKey(0)
cv2.imshow('imshow', imagen_final)
cv2.waitKey(0)
```

2.4. Algoritmo OCR basado en reconocimiento de patrones.

```
##########################################
# ALGORITMO OCR BASADO EN RECONOCIMIENTO DE PATRONES
# V1.0 | 2 LINEAS #


# PABLO JAVIER DE PAZ GARCÍA
##########################################
# LIBRERÍAS
##########################################
import cv2
import numpy as np
import os

##########################################
# VARIABLES
##########################################
w1, h1 = 570, 330
threshold = 0.7
X_Thresh = 50
Y_Thresh = 100

# Se inicializan las variables de alto y ancho del glifo
h, w = 0, 0

##########################################
# MAIN
##########################################

# Se carga el path de la imagen a leer
# path = 'imagenesTEST/matriculadetectada1.jpg'
# path = 'imagenesTEST/matriculadetectada2.jpg'
path = 'imagenesTEST/motocicletaESP.jpg'

# Se cargan todos los glifos de la carpeta
template_list = os.listdir('Glifos')
print("Número de glifos cargados: ", len(template_list))
Glifos = [cv2.imread(os.path.join('Glifos/'+template_list[i])) for i in
range(len(template_list))]

# Se carga un glifo de ejemplo
glifo_2lineas = Glifos[0]
glifo_2lineas_gris = cv2.cvtColor(glifo_2lineas, cv2.COLOR_RGB2GRAY)
ret2, glifo_2lineas_bin = cv2.threshold(glifo_2lineas_gris, 150, 255,
cv2.THRESH_BINARY)

# Se carga además el diccionario de glifos para identificar el valor de cada
detección
```

```
ArchivoGlifos = open("DiccionarioGlifos.txt", "r")
Indice_Glifos = ArchivoGlifos.read().split('\n')

# Se lee la imagen que contiene la matrícula de la que se va a extraer la información
imagen = cv2.imread(path)

# Se escala dicha imagen hasta el tamaño de los glifos, se conviete a grises y se binariza
# En el proceso se guarda una imagen_final para dibujar los resultados de las detecciones
imagen_escalada = cv2.resize(imagen, (w1, h1), interpolation=cv2.INTER_LINEAR)
imagen_final = cv2.resize(imagen, (w1, h1), interpolation=cv2.INTER_LINEAR)
imagen_gris = cv2.cvtColor(imagen_escalada, cv2.COLOR_RGB2GRAY)
ret, imagen_bin = cv2.threshold(imagen_gris, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Se inicializa el array que contendrá los puntos detectados y la clase detectada
puntos_detectados = []

# Se realiza una búsqueda para cada glifo, denotado en "DiccionarioGlifos"
for i in range(len(Indice_Glifos)):
  # Se carga el glifo
  Glifo = Glifos[i]

  # Se convierte dicho glifo a grises y se binariza
  glifo_gris = cv2.cvtColor(Glifo, cv2.COLOR_RGB2GRAY)
  ret2, glifo_bin = cv2.threshold(glifo_gris, 150, 255, cv2.THRESH_BINARY)

  # Se realiza el reconocimiento de patrones
  deteccion = cv2.matchTemplate(imagen_bin, glifo_bin, cv2.TM_CCOEFF_NORMED)
  # min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(deteccion)

  # Se determina el tamaño de un glifo
  h, w = Glifo.shape[0], Glifo.shape[1]

  # Se buscan las detecciones de patrones que superen el umbral de confianza
  localizacion = np.where(deteccion >= threshold)

  # El siguiente paso es quitar puntos redundantes.
  # Como se conoce el tamaño de un glifo, se pueden borrar todos aquellos que se encuentren solapados
  # mediante el uso de un algoritmo Non Maximum Supression casero
  puntos_crudos = []
  for pt in zip(*localizacion[::-1]):
      puntos_crudos.append(pt)
```

```
  # Mientras que la lista de puntos crudos siga conteniendo puntos, se repite el
bucle
  while len(puntos_crudos) != 0:
      # Se define un array con los puntos a borrar
      puntos_a_borrar = []
      # Se define el punto de referencia como el primer elemento de la lista, se
asigna a la lista de detecciones
      # y se añade a la lista de borrar de los crudos
      punto_referencia = puntos_crudos[0]
      puntos_detectados.append((puntos_crudos[0][0], puntos_crudos[0][1], i))
      puntos_a_borrar.append(punto_referencia)

      # Se asignan a la lista de borrar los puntos que no superan el umbral de
distancia
      for j in range(len(puntos_crudos)):
          if j != 0:
              if (abs(puntos_crudos[j][0] - punto_referencia[0]) < X_Thresh and
                      abs(puntos_crudos[j][1] - punto_referencia[1]) < Y_Thresh):
                  puntos_a_borrar.append(puntos_crudos[j])

      # Se borran los puntos asignados a la lista de borrar de la lista cruda
      for j in range(len(puntos_a_borrar)):
          puntos_crudos.remove(puntos_a_borrar[j])

      # Se repite el proceso hasta que la lista de crudos quede vacía

# Se organiza la lista de izquierda a derecha
puntos_detectados.sort()

# # Se traducen los indices de las detecciones a los glifos correspondientes
# Matricula = ''
# for i in range(len(puntos_detectados)):
#     Matricula += str(Indice_Glifos[puntos_detectados[i][2]])
#
# # Se obtiene la matrícula final
# print(" ")
# print("Matrícula: ", Matricula)
# print(" ")

for i in range(len(puntos_detectados)):
  cv2.rectangle(imagen_final, (puntos_detectados[i][0],
puntos_detectados[i][1]),
                (puntos_detectados[i][0] + w, puntos_detectados[i][1] + h), (255, 0,
0), 2)

cv2.imshow("matricula", imagen_bin)
cv2.waitKey(0)
```

```
cv2.imshow("matricula2", glifo_2lineas_bin)
cv2.waitKey(0)

cv2.imshow("matricula", imagen_final)
cv2.waitKey(0)
```

2.5. Algoritmo OCR basado en modelos de predicción.

```
##########################################
# ALGORITMO OCR BASADO EN MODELOS DE DETECCIÓN
# V1.0


# PABLO JAVIER DE PAZ GARCÍA
##########################################
# LIBRERÍAS
##########################################
from ultralytics import YOLO
import cv2

##########################################
# VARIABLES
##########################################
escala = 1
umbral_confianza = 0.35

##########################################
# MAIN
##########################################
# Se utiliza el modelo entrenado
modelo = YOLO('OCRV1.pt')

# Apertura de imagen
# path = 'imagenesTEST/recortadas/northwestterritories.png'
# path = 'imagenesTEST/recortadas/australianorthenterritory.png'
path = 'imagenesTEST/MatriculaDetectada.jpg'
# path = 'imagenesTEST/motocicletaESP.jpg'

# Se lee la imagen y se escala a un tamaño visible
imagen = cv2.imread(path)
w, h = imagen.shape[0], imagen.shape[1]
w, h = w*escala, h*escala
imagen_escalada = cv2.resize(imagen, [h, w], cv2.INTER_CUBIC)

# Se aplica el modelo de la red neuronal
resultados = modelo.predict(imagen_escalada, conf=umbral_confianza)[0]

# Se dibuja la bounding box sobre la imagen original
boundingbox = resultados.plot(line_width=1)
# print('número de detecciones', len(resultados.boxes.cls))

names = resultados.names
clases = resultados.boxes.cls
XYXY = resultados.boxes.xyxy
```

```
# Se crea una lista con las coordX, coordY y Clases de cada detección.
XYC = []
for i in range(len(clases)):
  XYC.append([int((XYXY[i][2]+XYXY[i][0])/2), int((XYXY[i][3]+XYXY[i][1])/2),
int(clases[i])])

# Se ordena la lista
XYC.sort()
# print(XYC)

clases_identificadas = []
for i in range(len(XYC)):
  clases_identificadas.append(XYC[i][2])

# print('clases ordenadas:', clases_concatenadas)

# Se traduce el valor de las clases a la matrícula final
Matricula = ""
for i in range(len(clases_identificadas)):
  Matricula += names[clases_identificadas[i]]

print(' ')
print('Matrícula: ', Matricula)
print(' ')
# cv2.imshow('matrícula', imagen_escalada)
# cv2.waitKey(0)
# cv2.imshow('matrícula', boundingbox)
# cv2.waitKey(0)
```

2.6. Algoritmo OCR basado en EasyOCR.

```
##########################################
# ALGORITMO OCR BASADO EN EASYOCR
# V1.0


# PABLO JAVIER DE PAZ GARCÍA
##########################################
# LIBRERÍAS
##########################################
import cv2
import easyocr
import time
##########################################
# VARIABLES
##########################################
w1, h1 = 840, 200

##########################################
# MAIN
##########################################
T1 = time.time()
imagen = cv2.imread('matriculadetectada1.jpg')
imagen_escalada = cv2.resize(imagen, (w1, h1),
interpolation=cv2.INTER_LINEAR)
imagen_gris = cv2.cvtColor(imagen_escalada, cv2.COLOR_RGB2GRAY)
ret, imagen_bin = cv2.threshold(imagen_gris, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
reader = easyocr.Reader(['en'], gpu=False)

resultado = reader.readtext(imagen_bin)
print("")
print("Lectura OCR:")
print(resultado[0][1])

matricula_cruda = resultado[0][1]

matricula_final = ""
for i in matricula_cruda:
  if i.isalnum():
      matricula_final += i

# print(time.time()-T1)
print("")
print("Matrícula final: ")
print(matricula_final)
print("")
```

2.7. Algoritmo OCR basado en Tesseract.

```
##########################################
# ALGORITMO OCR BASADO EN TESSERACT
# V1.0


# PABLO JAVIER DE PAZ GARCÍA
##########################################
# LIBRERÍAS
##########################################
import cv2
import pytesseract
import time
##########################################
# VARIABLES
##########################################
w1, h1 = 840, 200

##########################################
# MAIN
##########################################
pytesseract.pytesseract.tesseract_cmd =
r'C:\Users\pablo\AppData\Local\Programs\Tesseract-OCR\tesseract.exe'

T1 = time.time()
imagen = cv2.imread('matriculadetectada1.jpg')
imagen_escalada = cv2.resize(imagen, (w1, h1),
interpolation=cv2.INTER_LINEAR)
imagen_gris = cv2.cvtColor(imagen_escalada, cv2.COLOR_RGB2GRAY)
ret, imagen_bin = cv2.threshold(imagen_gris, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)

matricula_cruda = pytesseract.image_to_string(imagen_bin)
print("")
print("Lectura OCR:")
print(matricula_cruda)

# Se filtran los caracteres especiales
matricula_final = ""
for i in matricula_cruda:
   if i.isalnum():
      matricula_final += i

# print(time.time()-T1)
# print("")
print("Matrícula final: ")
print(matricula_final)
print("")
```

```
cv2.imshow("matrícula binarizada", imagen_bin)
cv2.waitKey(0)
```

2.8. Algoritmo ANPR completo.

```
##########################################
# ALGORITMO ANPR COMPLETO
# V1.0


# PABLO JAVIER DE PAZ GARCÍA
##########################################
# LIBRERÍAS
##########################################
from ultralytics import YOLO
import cv2
import numpy as np
import time

##########################################
# VARIABLES #
##########################################
escala = 3
umbral_confianza_det = 0.3
umbral_confianza_ocr = 0.3
y_thresh_valor = 0.6
##########################################


def recortar_mat():
  # Se extraen de los resultados las coordenadas de la boundingbox de la detección,
  # en formato XY,XY (esquinas de la bounding box)
  XYXYDET = resultadosDET.boxes.xyxy
  X1bb = int(XYXYDET[0, 0].item())
  Y1bb = int(XYXYDET[0, 1].item())
  X2bb = int(XYXYDET[0, 2].item())
  Y2bb = int(XYXYDET[0, 3].item())

  # Se crea una máscara vacía donde se cambia a negro todos los pixeles menos los de la boundingbox,
  # para ver con mas claridad la imagen
  mascara_matricula = np.zeros(resultadosDET.orig_img.shape[:2], np.uint8)
  mascara_matricula[Y1bb:Y2bb, X1bb:X2bb] = 255
  masked_imagen = cv2.bitwise_and(resultadosDET.orig_img, resultadosDET.orig_img, mask=mascara_matricula)

  # Se recortan el resto de pixeles
  return resultadosDET.orig_img[Y1bb:Y2bb, X1bb:X2bb]


def leer_mat():
```

```
    # Se determina el diccionario de clases, los indices del diccionario y las
coordenadas
    # de las bounding boxes
    names = resultadosOCR.names
    clases = resultadosOCR.boxes.cls
    XYXYOCR = resultadosOCR.boxes.xyxy

    # Se crea una lista con las coordX, coordY y Clases de cada detección.
    XYC = []
    for i in range(len(clases)):
        XYC.append([int((XYXYOCR[i][2] + XYXYOCR[i][0]) / 2),
int((XYXYOCR[i][3] + XYXYOCR[i][1]) / 2), int(clases[i])])

    # Se ordena la lista
    XYC.sort(key=lambda x: x[1])

    # Se determina el número de filas de la matrícula, y se crean dos listas con el
contenido de cada fila
    fila1, fila2 = [], []
    ythreshold = XYC[0][1] + (y_thresh_valor * XYC[0][1])
    # print('ythreshold: ', ythreshold)
    for i in range(len(clases)):
        if (XYC[i][1]) <= ythreshold:
            fila1.append(XYC[i])
        else:
            fila2.append(XYC[i])

    # Se ordenan las filas de izquierda a derecha
    fila1.sort()
    fila2.sort()

    # Se concatenan las filas detectadas y se extrae el valor final de las clases de
la matricula
    filas_concatenadas = fila1 + fila2
    # print('filas concatenadas: ', filas_concatenadas)

    clases_concatenadas = []
    for i in range(len(filas_concatenadas)):
        clases_concatenadas.append(filas_concatenadas[i][2])

    # Se traduce el valor de las clases a la matrícula final
    Matricula = ""
    for i in range(len(clases_concatenadas)):
        Matricula = Matricula + names[clases_concatenadas[i]]
    return Matricula


# Se cargan los modelos a emplear
modeloDET = YOLO('modelos/DetMatriculasV1.pt')
```

```
modeloOCR = YOLO('modelos/OCRV1.pt')

# Apertura de imagen
path = 'imagenesTEST/Coche2.jpg'
# path = 'imagenesTEST/Camion.jpg'
# path = 'imagenesTEST/Furgoneta.jpg'
# path = 'imagenesTEST/OsoCanada.jpg'

imagen = cv2.imread(path)

# Se aplica el modelo de la red neuronal
resultadosDET = modeloDET.predict(imagen, conf=umbral_confianza_det,
max_det=1)[0]

# Se comprueba que exista una detección de matricula
if resultadosDET.boxes.xyxy.numel():

  # Se recortan el resto de pixeles
  matricula_recortada = recortar_mat()

  # Se escala a un tamaño visible
  w, h = matricula_recortada.shape[0], matricula_recortada.shape[1]
  w, h = w * escala, h * escala
  matricula_escalada = cv2.resize(matricula_recortada, [h, w],
cv2.INTER_CUBIC)

  # Se aplica el modelo de la red neuronal
  resultadosOCR = modeloOCR.predict(matricula_escalada,
conf=umbral_confianza_ocr, max_det=12)[0]
  T1 = time.time()

  # Se comprueba que exista una derección de glifo
  if resultadosOCR.boxes.xyxy.numel():

    # Se llama a la función que lee la matrícula
    Matricula_entrada = leer_mat()

    print(' ')
    print('Matrícula: ', Matricula_entrada)
    print(' ')

    # Se dibuja la bounding box sobre las imagenes
    boundingboxDET = resultadosDET.plot(line_width=1)
    boundingboxOCR = resultadosOCR.plot(line_width=1)

print (time.time()-T1)

# Se muestran los resultados
cv2.imshow('Detecciones', imagen)
```

```
cv2.waitKey(0)
cv2.imshow('Detecciones', boundingboxDET)
cv2.waitKey(0)
cv2.imshow('Detecciones', matricula_recortada)
cv2.waitKey(0)
cv2.imshow('Detecciones', boundingboxOCR)
cv2.waitKey(0)
```

2.9. Aplicación ANPR completa.

```
##########################################
# APLICACIÓN ANPR COMPLETA
# V1.1 #

# PABLO JAVIER DE PAZ GARCÍA
##########################################
# LIBRERÍAS #
##########################################
import customtkinter as ctk
import cv2
from PIL import Image
import numpy as np
from ultralytics import YOLO
import time
from datetime import datetime
import json
##########################################
# VARIABLES #
##########################################
# Globales
global Matricula_entrada
global Matricula_salida
##########################################
umbral_confianza_det = 0.2
umbral_confianza_ocr = 0.25
algoritmoOCR = 1  # 1: 1 linea, 2: 2 lineas
y_thresh_valor = 0.5  # Para el algoritmo de dos lineas
##########################################
path1 = 'imagenesTEST/EntradaV1.0.mp4'
path2 = 'imagenesTEST/SalidaV1.0.mp4'

##########################################
# INICIALIZACIÓN DE VARIABLES SECUNDARIAS
##########################################
# Contadores para las matriculas
Contador_Matricula_Anterior_entrada = 0
Matricula_Anterior_entrada = ""
Contador_Matricula_Anterior_salida = 0
Matricula_Anterior_salida = ""

# Memorias de entrada / salida
matricula_reciente_entrada = ""
matricula_reciente_salida = ""
decision_entrada_pendiente = 0
decision_salida_pendiente = 0
barrera_entrada_arriba = 0
barrera_salida_arriba = 0
```

```
# Variables de los botones
var_permitir_acceso = 0
var_denegar_acceso = 0
var_permitir_salida = 0
var_denegar_salida = 0

# Memorias de tiempo
Timer_barrera_entrada = 0
Timer_barrera_salida = 0
Timer_thresh_barrera = 6  # Tiempo que tardan en cerrarse las barreras

# Una memoria para inicializar el registro
memoria_inicializar_variables = 0
##########################################
# MAIN
##########################################


def cv2topil(imagenaconvetir):
    ######################################################################
    # Esta función convierte una imagen de formato CV2 a PIL #
    ######################################################################

    imagen_pil = cv2.cvtColor(imagenaconvetir, cv2.COLOR_BGR2RGB)
    imagen_pil = Image.fromarray(imagen_pil)
    return imagen_pil


def boton_decision(boton):
    ####################################################
    # Esta función se llama al presionar un botón #
    ####################################################
    # significado de boton:
    # 1: permitir acceso
    # 2: denegar acceso
    # 3: permitir salida
    # 4: denegar salida
    ########################

    global var_permitir_acceso
    global var_denegar_acceso
    global var_permitir_salida
    global var_denegar_salida

    if boton == 1:
        var_permitir_acceso = 1
    elif boton == 2:
        var_denegar_acceso = 1
```

```
    elif boton == 3:
        var_permitir_salida = 1
    elif boton == 4:
        var_denegar_salida = 1


def app():
    ######################
    # CÓDIGO DE LA APP #
    ######################

    global ventana
    global imagen_entrada
    global imagen_salida
    global boton_si_acceso
    global boton_no_acceso
    global boton_si_salida
    global boton_no_salida
    global label_registro
    global label_matricula_entrada
    global label_conocer_matricula_entrada
    global label_matricula_salida
    global label_conocer_matricula_salida
    global boton_estado_barrera_entrada
    global boton_estado_barrera_salida

    ventana = ctk.CTk()
    ventana.title("Aplicación ANPR | Pablo Javier de Paz
"
                  "                                                                    "
                  "                    Universidad Europea de Madrid")
    ventana.geometry("1195x670")  # Imagen dividida en dos verticalmente:
300+70+300
    ventana.minsize(1195, 670)
    ventana.maxsize(1195, 670)
    ctk.set_appearance_mode("Light")
    ctk.set_default_color_theme("dark-blue")

    # Se lee una imagen de ejemplo
    cargando = cv2.imread("imagenes_app/loading.jpg")
    imagen_pil = cv2topil(cargando)
    imagen_entrada_tk = ctk.CTkImage(imagen_pil, size=(480, 270))

    #
**********************************************************************************************
*********
    # WIDGETS #
```

```
  #
*****************************************************************************************
*********
  ##########################
  # Se crean los widgets #
  ##########################
  # Se crean los cuadros, "frame"
  cuadro_entrada = ctk.CTkFrame(master=ventana, width=796, height=305,
border_width=2)
  cuadro_salida = ctk.CTkFrame(master=ventana, width=796, height=305,
border_width=2)
  cuadro_paramentrada = ctk.CTkFrame(master=cuadro_entrada, width=304,
height=295, border_width=2)
  cuadro_paramsalida = ctk.CTkFrame(master=cuadro_salida, width=304,
height=295, border_width=2)
  cuadro_registro = ctk.CTkScrollableFrame(master=ventana, width=300,
height=570, border_width=2,
                                           label_text="Registro de entradas y salidas",
label_font=("ARIAL", 20))

  # Labels de video
  imagen_entrada = ctk.CTkLabel(cuadro_entrada, text=' ',
image=imagen_entrada_tk)
  imagen_salida = ctk.CTkLabel(cuadro_salida, text=' ',
image=imagen_entrada_tk)

  # Labels de texto
  label_entrada = ctk.CTkLabel(cuadro_entrada, text="Cámara de acceso:
Entrada",
                               font=("ARIAL", 20), fg_color="transparent")
  label_salida = ctk.CTkLabel(cuadro_salida, text="Cámara de acceso: Salida",
                              font=("ARIAL", 20), fg_color="transparent")
  label_titulo_paramentrada = ctk.CTkLabel(cuadro_paramentrada,
text="Matrícula en entrada:",
                                           font=("ARIAL", 20), fg_color="#b9c3c4",
corner_radius=5)
  label_titulo_paramsalida = ctk.CTkLabel(cuadro_paramsalida, text="Matrícula
en salida:",
                                          font=("ARIAL", 20), fg_color="#b9c3c4",
corner_radius=5)
  label_registro = ctk.CTkLabel(cuadro_registro, text="Cargando registro...",
font=("ARIAL", 17))
  label_matricula_entrada = ctk.CTkLabel(cuadro_paramentrada,
text="----------",
                                         font=("ARIAL", 30))
  label_conocer_matricula_entrada = ctk.CTkLabel(cuadro_paramentrada,
text="- - - - - - - - - -")
  label_matricula_salida = ctk.CTkLabel(cuadro_paramsalida, text="----------",
                                        font=("ARIAL", 30))
```

```
    label_conocer_matricula_salida = ctk.CTkLabel(cuadro_paramsalida, text="- - - - - - - - -")
    label_estado_barrera_entrada = ctk.CTkLabel(cuadro_paramentrada, text="Estado de barrera entrada:",
                                                font=("ARIAL", 20), fg_color="#b9c3c4", corner_radius=5)
    label_estado_barrera_salida = ctk.CTkLabel(cuadro_paramsalida, text="Estado de barrera salida:",
                                               font=("ARIAL", 20), fg_color="#b9c3c4", corner_radius=5)

    # Botones
    boton_si_acceso = ctk.CTkButton(master=cuadro_paramentrada, width=98, height=80, fg_color="#767b82",
                                    border_width=2, hover_color="#767b82", text="Permitir \nacesso",
                                    text_color="#000000", font=("ARIAL", 17), state="disabled",
                                    command=lambda: boton_decision(1))
    boton_no_acceso = ctk.CTkButton(master=cuadro_paramentrada, width=199, height=80, fg_color="#767b82",
                                    border_width=2, hover_color="#767b82", text="Denegar acceso", text_color="#000000",
                                    font=("ARIAL", 17), state="disabled", command=lambda: boton_decision(2))
    boton_si_salida = ctk.CTkButton(master=cuadro_paramsalida, width=98, height=80, fg_color="#767b82",
                                    border_width=2, hover_color="#767b82", text="Permitir \nsalida",
                                    text_color="#000000", font=("ARIAL", 17), state="disabled",
                                    command=lambda: boton_decision(3))
    boton_no_salida = ctk.CTkButton(master=cuadro_paramsalida, width=199, height=80, fg_color="#767b82",
                                    border_width=2, hover_color="#767b82", text="Denegar salida", text_color="#000000",
                                    font=("ARIAL", 17), state="disabled", command=lambda: boton_decision(4))
    boton_estado_barrera_entrada = ctk.CTkButton(master=cuadro_paramentrada, width=299, height=60,
                                                 text="CERRADA", text_color="#000000",
                                                 font=("ARIAL", 17), fg_color="#0f76d1", hover_color="#0f76d1",
                                                 border_width=2)
    boton_estado_barrera_salida = ctk.CTkButton(master=cuadro_paramsalida, width=299, height=60,
                                                text="CERRADA", text_color="#000000",
                                                font=("ARIAL", 17), fg_color="#0f76d1", hover_color="#0f76d1",
```

```
                                    border_width=2)
    ############################
    # Se colocan los widgets #
    ############################
    # Cam entrada
    cuadro_entrada.place(x=20, y=5)
    label_entrada.place(x=10, y=4)
    imagen_entrada.place(x=1, y=33)
    cuadro_paramentrada.place(x=486, y=5)
    # Cuadro Param entrada
    label_titulo_paramentrada.place(x=8, y=8)
    label_matricula_entrada.place(x=10, y=40)
    label_conocer_matricula_entrada.place(x=10, y=67)
    boton_si_acceso.place(x=3, y=95)
    boton_no_acceso.place(x=102, y=95)
    label_estado_barrera_entrada.place(x=8, y=200)
    boton_estado_barrera_entrada.place(x=3, y=232)
    # Cuadro Param salida
    label_titulo_paramsalida.place(x=8, y=8)
    label_matricula_salida.place(x=10, y=40)
    label_conocer_matricula_salida.place(x=10, y=67)
    boton_si_salida.place(x=3, y=95)
    boton_no_salida.place(x=102, y=95)
    label_estado_barrera_salida.place(x=8, y=200)
    boton_estado_barrera_salida.place(x=3, y=232)
    # Cam salida
    cuadro_salida.place(x=20, y=335)
    label_salida.place(x=10, y=4)
    imagen_salida.place(x=1, y=33)
    cuadro_paramsalida.place(x=486, y=5)
    # Registro
    cuadro_registro.place(x=840, y=5)
    label_registro.pack(side="top", anchor="nw")
    #
*****************************************************************************************
*********
    # Se llama a la función con el programa ANPR
    ventana.after(1500, anpr)

    # Se llama al loop de la función principal
    ventana.mainloop()


def anpr():
    def recortar_mat(resultadosdet_func):

#########################################################################
        # Esta función devuelve un recorte de la imagen original con la región de la
matrícula #
```

```
        ##########################################################################
            # Se extraen de los resultados las coordenadas de la boundingbox de la
detección,
            # en formato XY,XY (esquinas de la bounding box)
            XYXYDET = resultadosdet_func.boxes.xyxy
            if XYXYDET.numel():
                X1bb = int(XYXYDET[0, 0].item())
                Y1bb = int(XYXYDET[0, 1].item())
                X2bb = int(XYXYDET[0, 2].item())
                Y2bb = int(XYXYDET[0, 3].item())

                # Se crea una máscara vacía donde se cambia a negro todos los pixeles
menos los de la boundingbox,
                # para ver con mas claridad la imagen
                mascara_matricula = np.zeros(resultadosdet_func.orig_img.shape[:2],
np.uint8)
                mascara_matricula[Y1bb:Y2bb, X1bb:X2bb] = 255

                # Se recortan el resto de pixeles
                return resultadosdet_func.orig_img[Y1bb:Y2bb, X1bb:X2bb]

        def leer_mat(resultadosocr_func):

        ##########################################################################
            # Esta funcion devuelve una cadena de texto con la matricula detectada #

        ##########################################################################
            # Versión del algoritmo: una linea #
            ########################################

            # Se determina el diccionario de clases, los indices del diccionario y las
coordenadas
            # de las bounding boxes
            names = resultadosocr_func.names
            clases = resultadosocr_func.boxes.cls
            XYXYOCR = resultadosocr_func.boxes.xyxy

            # Se crea una lista con las coordX, coordY y Clases de cada detección.
            XYC = []
            for i in range(len(clases)):
                XYC.append([int((XYXYOCR[i][2] + XYXYOCR[i][0]) / 2),
int((XYXYOCR[i][3] + XYXYOCR[i][1]) / 2),
                            int(clases[i])])

            if algoritmoOCR == 1:
                # Se ordena la lista, de izquierda a derecha
                XYC.sort()
```

```
            clases_concatenadas = []
            for i in range(len(XYC)):
                clases_concatenadas.append(XYC[i][2])

        else:
            # Se ordena la lista
            XYC.sort(key=lambda x: x[1])

            # Se determina el número de filas de la matrícula, y se crean dos listas con el contenido de cada fila
            fila1, fila2 = [], []
            ythreshold = XYC[0][1] + (y_thresh_valor * XYC[0][1])
            print('ythreshold: ', ythreshold)
            for i in range(len(clases)):
                if (XYC[i][1]) <= ythreshold:
                    fila1.append(XYC[i])
                else:
                    fila2.append(XYC[i])

            # Se ordenan las filas de izquierda a derecha
            fila1.sort()
            fila2.sort()

            # Se concatenan las filas detectadas y se extrae el valor final de las clases de la matricula
            filas_concatenadas = fila1 + fila2
            # print('filas concatenadas: ', filas_concatenadas)

            clases_concatenadas = []
            for i in range(len(filas_concatenadas)):
                clases_concatenadas.append(filas_concatenadas[i][2])

        # Se traduce el valor de las clases a la matrícula final
        Matricula_func = ""
        for i in range(len(clases_concatenadas)):
            Matricula_func = Matricula_func + names[clases_concatenadas[i]]

        if 5 < len(Matricula_func) < 10:
            return Matricula_func
        else:
            return 0

    def mostrarvideos(fotograma_func, video):

##########################################################################
        # Esta función coloca la imagen recibida en el cuadro de video de salida o entrada #

##########################################################################
```

```
        # Video de Entrada: 1
        # Video de Salida: 2
        pil_fotograma_func = cv2topil(fotograma_func)
        ctk_fotograma_func = ctk.CTkImage(pil_fotograma_func, size=(480, 270))

        if video == 1:
            imagen_entrada.configure(image=ctk_fotograma_func)
            imagen_entrada.image = ctk_fotograma_func
        elif video == 2:
            imagen_salida.configure(image=ctk_fotograma_func)
            imagen_salida.image = ctk_fotograma_func

    def leerregistroaccesos():

##########################################################################
        # Devuelve una lista con el contenido del txt llamado registro, lee json y
convierte a list #

##########################################################################
        registro_func = open("RegistroAccesos.txt", "r")
        registrojson_func = registro_func.read()
        if registrojson_func != "":
            registrolist_func = json.loads(registrojson_func)
            registro_func.close()
            return registrolist_func
        else:
            registrolist_vacio = []
            return registrolist_vacio

    def escribirregistroaccesos(registro):

##########################################################################
        # Esta función sobreescribe en el registro accesos la lista "registro" #

##########################################################################
        global matricula_reciente_entrada
        registro_func = open("RegistroAccesos.txt", "w")
        registronuevo_json = json.dumps(registro)
        registro_func.write(registronuevo_json)
        registro_func.close()
        matricula_reciente_entrada = Matricula_entrada

    def registrarentrada(matricula_func):

##########################################################################
        # Esta función registra una entrada en formato MAT/HORA_ENTRADA #

##########################################################################
        registro = leerregistroaccesos()
```

```
        ahora = datetime.now()
        ahora_str = ahora.strftime("%d/%m/%Y %H:%M:%S")
        estructura_mat = [matricula_func, ahora_str]
        registro.append(estructura_mat)
        escribirregistroaccesos(registro)
        toggle_barreras(1, -1)

    def registrarsalida(matricula_func):

#####################################################################
        # Esta función registra una salida en formato MAT/HORA_SALIDA #

#####################################################################
        registro = leerregistroaccesos()
        ahora = datetime.now()
        ahora_str = ahora.strftime("%d/%m/%Y %H:%M:%S")
        for i in range(len(registro)):
            if (matricula_func in registro[i]) and len(registro[i]) == 2:
                registro[i].append(ahora_str)
                break
        escribirregistroaccesos(registro)
        toggle_barreras(-1, 1)

    def toggle_preguntas(entrada, salida):
        ###################################################################
        # Esta función enciende o apaga los botones en la interfaz #
        ###################################################################
        # Encender pregunta Entrada: 1
        # Apagar pregunta Entrada: 0
        # Encender pregunta Salida: 1
        # Apagar pregunta Salida: 0
        ###################################
        if entrada == 1:
            boton_si_acceso.configure(fg_color="#5db82c", hover_color="#3e7a1d",
state="normal")
            boton_no_acceso.configure(fg_color="#d12f1d", hover_color="#962215",
state="normal")
        elif entrada == 0:
            boton_si_acceso.configure(fg_color="#767b82", hover_color="#767b82",
state="disabled")
            boton_no_acceso.configure(fg_color="#767b82",
hover_color="#767b82", state="disabled")
        elif salida == 1:
            boton_si_salida.configure(fg_color="#5db82c", hover_color="#3e7a1d",
state="normal")
            boton_no_salida.configure(fg_color="#d12f1d", hover_color="#962215",
state="normal")
        elif salida == 0:
```

```
        boton_si_salida.configure(fg_color="#767b82", hover_color="#767b82",
state="disabled")
        boton_no_salida.configure(fg_color="#767b82", hover_color="#767b82",
state="disabled")

  def toggle_barreras(entrada, salida):

#########################################################################
#####################################
    # Esta función abre o cierra las barreras, y borra la matrícula de la interfaz
cuando las cierra #

#########################################################################
#####################################
    global Timer_barrera_entrada
    global Timer_barrera_salida
    global Timer_thresh_barrera
    global barrera_entrada_arriba
    global barrera_salida_arriba

    if entrada == 1:
        boton_estado_barrera_entrada.configure(text="ABIERTA",
fg_color="#ebe834", hover_color="#ebe834")
        Timer_barrera_entrada = time.time()
        barrera_entrada_arriba = 1

    # Cerrar barrera entrada si el timer expira, además, quitar la matricula de la
interfaz
    elif entrada == 0 and (time.time() - Timer_barrera_entrada) >
Timer_thresh_barrera:
        boton_estado_barrera_entrada.configure(text="CERRADA",
fg_color="#0f76d1", hover_color="#0f76d1")
        Timer_barrera_entrada = 0
        label_matricula_entrada.configure(text="-----------")
        label_conocer_matricula_entrada.configure(text="- - - - - - - - - -")
        barrera_entrada_arriba = 0

    elif salida == 1:
        boton_estado_barrera_salida.configure(text="ABIERTA",
fg_color="#ebe834", hover_color="#ebe834")
        Timer_barrera_salida = time.time()
        barrera_salida_arriba = 1

    # Cerrar barrera salida si el timer expira, además, quitar la matricula de la
interfaz
    elif salida == 0 and (time.time() - Timer_barrera_salida) >
Timer_thresh_barrera:
        boton_estado_barrera_salida.configure(text="CERRADA",
fg_color="#0f76d1", hover_color="#0f76d1")
```

```
            Timer_barrera_salida = 0
            label_matricula_salida.configure(text="-----------")
            label_conocer_matricula_salida.configure(text="- - - - - - - - - -")
            barrera_salida_arriba = 0

    def mostrarregistroaccesos():

#####################################################################
        # Esta función muestra el registro de accesos en la interfaz #

#####################################################################

        registro_func = leerregistroaccesos()
        registro_func.reverse()
        registro_a_mostrar = ""
        for i in range(len(registro_func)):
            registro_a_mostrar += "Matrícula: " + registro_func[i][0] + "\n"
            registro_a_mostrar += "Entrada: " + registro_func[i][1] + "\n"
            if len(registro_func[i]) == 3:
                registro_a_mostrar += "Salida: " + registro_func[i][2] + "\n"
            registro_a_mostrar += "\n"

        label_registro.configure(text=registro_a_mostrar)

    def anpr_entrada(fotograma_entrada):
        ##################################################
        # FUNCIÓN ANPR PARA LA CÁMARA DE LA ENTRADA #
        ##################################################

        global Matricula_entrada
        global Matricula_Anterior_entrada
        global Contador_Matricula_Anterior_entrada
        global matricula_reciente_entrada
        global decision_entrada_pendiente
        global var_permitir_acceso
        global var_denegar_acceso
        global Timer_barrera_entrada
        global barrera_entrada_arriba

        if decision_entrada_pendiente == 0 and barrera_entrada_arriba == 0:
            # Se aplica el modelo de la red neuronal
            resultadosDET = modeloDET.predict(fotograma_entrada,
conf=umbral_confianza_det, max_det=1)[0]

            # Se comprueba si existe una detección de matricula
            if resultadosDET.boxes.xyxy.numel():
                # Se llama a la funcion recortar_mat() para recortar el resto de pixeles
                matricula_recortada = recortar_mat(resultadosDET)
```

```
            # Se aplica el modelo de la red neuronal
            resultadosOCR = modeloOCR.predict(matricula_recortada,
conf=umbral_confianza_ocr, max_det=9)[0]

            # Se comprueba si existe al menos una detección de caracter
            if resultadosOCR.boxes.xyxy.numel():
                # Se llama a la función que lee la matrícula
                Matricula_entrada = leer_mat(resultadosOCR)

                archivo_permitidas = open("Permitidas.txt", "r")
                permitidas_list = archivo_permitidas.read().split('\n')
                archivo_permitidas.close()

                if Matricula_entrada != matricula_reciente_entrada:
                    if Matricula_entrada in permitidas_list:
                        print(' ')
                        print('Matrícula Permitida: ', Matricula_entrada)
                        print(' ')
                        # Se escribe en el registro la entrada permitida
                        registrarentrada(Matricula_entrada)
                        mostrarregistroaccesos()
                        matricula_reciente_entrada = Matricula_entrada
                        label_matricula_entrada.configure(text=Matricula_entrada)
                        label_conocer_matricula_entrada.configure(text="Matrícula
registrada.  Permitiendo "
                                                                  "acceso...")

                    else:
                        # Si la matricula es la misma, se aumenta el contador
                        if (Matricula_Anterior_entrada == Matricula_entrada and
Contador_Matricula_Anterior_entrada < 3
                                and Matricula_entrada != 0 and Matricula_entrada !=
matricula_reciente_entrada):
                            Contador_Matricula_Anterior_entrada =
Contador_Matricula_Anterior_entrada + 1

                        # Si la matrícula no es la misma que la anterior
                        elif Matricula_Anterior_entrada != Matricula_entrada:
                            Contador_Matricula_Anterior_entrada = 0

                        # Si la matrícula es la misma que la anterior y el contador es
correcto
                        # se confirma que la matricula leida es desconocida
                        elif (Matricula_Anterior_entrada == Matricula_entrada and
Contador_Matricula_Anterior_entrada >= 3
                                and Matricula_entrada != matricula_reciente_entrada):
                            print(' ')
                            print('Matrícula Desconocida: ', Matricula_entrada)
                            print(' ')
```

```
                    label_matricula_entrada.configure(text=Matricula_entrada)
                    label_conocer_matricula_entrada.configure(text="Matrícula
desconocida.")
                    # Se activa la función que permite elegir si dejar pasar
                    decision_entrada_pendiente = 1
                    # 1 significa encender los botones de pregunta entrada
                    toggle_preguntas(1, -1)

                # Matricula anterior
                Matricula_Anterior_entrada = Matricula_entrada

            # Se dibuja la bounding box sobre las imagenes
            boundingboxDET = resultadosDET.plot(line_width=1)

            # Se convierte la imagen CV2 a PIL
            mostrarvideos(boundingboxDET, 1)

        else:
            # Si no hay una detección en pantalla
            # se convierte el fotograma a formato PIL y se carga la imagen en la
label de video
            mostrarvideos(fotograma_entrada, 1)

    elif decision_entrada_pendiente == 1:
        # Se muestra la imagen
        mostrarvideos(fotograma_entrada, 1)
        print("decidiendo entrada...")
        # Si se hace click en permitir acceso
        if var_permitir_acceso == 1:
            registrarentrada(Matricula_entrada)
            matricula_reciente_entrada = Matricula_entrada
            mostrarregistroaccesos()
            toggle_preguntas(0, -1)
            var_permitir_acceso = 0
            decision_entrada_pendiente = 0

        # Si se hace click en denegar acceso
        elif var_denegar_acceso == 1:
            toggle_preguntas(0, -1)
            var_denegar_acceso = 0
            decision_entrada_pendiente = 0
            label_matricula_entrada.configure(text="----------")
            label_conocer_matricula_entrada.configure(text="- - - - - - - - - -")

    elif barrera_entrada_arriba == 1:
        # Se muestra la imagen
        mostrarvideos(fotograma_entrada, 1)
        print("permitiendo entrada...")
```

```
  def anpr_salida(fotograma_salida):
      ################################################
      # FUNCIÓN ANPR PARA LA CÁMARA DE LA SALIDA #
      ################################################

      global Matricula_salida
      global Matricula_Anterior_salida
      global Contador_Matricula_Anterior_salida
      global matricula_reciente_salida
      global decision_salida_pendiente
      global var_permitir_salida
      global var_denegar_salida
      global barrera_salida_arriba

      if decision_salida_pendiente == 0 and barrera_salida_arriba == 0:
          # Se aplica el modelo de la red neuronal
          resultadosDET = modeloDET.predict(fotograma_salida,
conf=umbral_confianza_det, max_det=1)[0]

          # Se comprueba si existe una detección de matricula
          if resultadosDET.boxes.xyxy.numel():
              # Se llama a la funcion recortar_mat() para recortar el resto de pixeles
              matricula_recortada = recortar_mat(resultadosDET)

              # Se aplica el modelo de la red neuronal
              resultadosOCR = modeloOCR.predict(matricula_recortada,
conf=umbral_confianza_ocr, max_det=9)[0]

              # Se comprueba si existe al menos una detección de caracter
              if resultadosOCR.boxes.xyxy.numel():
                  # Se llama a la función que lee la matrícula
                  Matricula_salida = leer_mat(resultadosOCR)

                  archivo_permitidas = open("Permitidas.txt", "r")
                  permitidas_list = archivo_permitidas.read().split('\n')
                  archivo_permitidas.close()

                  if Matricula_salida != matricula_reciente_salida:
                      if Matricula_salida in permitidas_list:
                          print(' ')
                          print('Matrícula Permitida: ', Matricula_salida)
                          print(' ')
                          # Se escribe en el registro la entrada permitida
                          registrarsalida(Matricula_salida)
                          mostrarregistroaccesos()
                          matricula_reciente_salida = Matricula_salida
                          label_matricula_salida.configure(text=Matricula_salida)
                          label_conocer_matricula_salida.configure(text="Matrícula
registrada.  Permitiendo "
```

```
                                                    "salida...")

                    else:
                        # Si la matricula es la misma, se aumenta el contador
                        if (Matricula_Anterior_salida == Matricula_salida and
Contador_Matricula_Anterior_salida < 3
                                and Matricula_salida != 0 and Matricula_salida !=
matricula_reciente_salida):
                            Contador_Matricula_Anterior_salida =
Contador_Matricula_Anterior_salida + 1

                        # Si la matrícula no es la misma que la anterior
                        elif Matricula_Anterior_salida != Matricula_salida:
                            Contador_Matricula_Anterior_salida = 0

                        # Si la matrícula es la misma que la anterior y el contador es
correcto
                        # se confirma que la matricula leida es desconocida
                        elif (Matricula_Anterior_salida == Matricula_salida and
Contador_Matricula_Anterior_salida >= 3
                                and Matricula_salida != matricula_reciente_salida):
                            print(' ')
                            print('Matrícula Desconocida: ', Matricula_salida)
                            print(' ')
                            label_matricula_salida.configure(text=Matricula_salida)
                            label_conocer_matricula_salida.configure(text="Matrícula
desconocida.")
                            # Se activa la función que permite elegir si dejar pasar
                            decision_salida_pendiente = 1
                            # 1 significa encender los botones de pregunta entrada
                            toggle_preguntas(-1, 1)

                        # Matricula anterior
                        Matricula_Anterior_salida = Matricula_salida

                    # Se dibuja la bounding box sobre las imagenes
                    boundingboxDET = resultadosDET.plot(line_width=1)

                    # Se convierte la imagen CV2 a PIL
                    mostrarvideos(boundingboxDET, 2)
                else:
                    # Si no hay una detección en pantalla
                    # se convierte el fotograma a formato PIL y se carga la imagen en la
label de video
                    mostrarvideos(fotograma_salida, 2)

            elif decision_salida_pendiente == 1:
                # Se muestra la imagen
                mostrarvideos(fotograma_salida, 2)
```

```
        print("decidiendo salida...")
        # Si se hace click en permitir acceso
        if var_permitir_salida == 1:
            registrarsalida(Matricula_salida)
            matricula_reciente_salida = Matricula_salida
            mostrarregistroaccesos()
            toggle_preguntas(-1, 0)
            var_permitir_salida = 0
            decision_salida_pendiente = 0

        # Si se hace click en denegar acceso
        elif var_denegar_salida == 1:
            toggle_preguntas(-1, 0)
            var_denegar_salida = 0
            decision_salida_pendiente = 0
            label_matricula_salida.configure(text="----------")
            label_conocer_matricula_salida.configure(text="- - - - - - - - - -")

    elif barrera_salida_arriba == 1:
        # Se muestra la imagen
        mostrarvideos(fotograma_salida, 2)
        print("permitiendo salida...")

########################
# MAIN DE AMBOS ANPR #
########################
global memoria_inicializar_variables
global Timer_barrera_entrada
global Timer_barrera_salida

# Se llama a las funciones que muestran por primera vez las variables
if memoria_inicializar_variables == 0:
    mostrarregistroaccesos()
    memoria_inicializar_variables = 1

# Se llama a las funciones de cerrar barreras
if Timer_barrera_entrada != 0:
    toggle_barreras(0, -1)

if Timer_barrera_salida != 0:
    toggle_barreras(-1, 0)

# Se obtiene el fotograma actual del video
ret, fotograma1 = video_entrada.read()
if ret:
    anpr_entrada(fotograma1)

ret, fotograma2 = video_salida.read()
if ret:
```

```
        anpr_salida(fotograma2)
    ventana.after(33, anpr)


########################################
# INICIALIZACIÓN DE MODELOS/ VÍDEOS #
########################################

# Se cargan los modelos a emplear
modeloDET = YOLO('modelos/DetMatriculasV1.pt')
modeloOCR = YOLO('modelos/OCRV1.pt')

# Apertura de video
video_entrada = cv2.VideoCapture(path1)
video_salida = cv2.VideoCapture(path2)

# Lanzamiento del programa
app()
```

ANEXO 3: NORMATIVA APLICABLE (RESUMEN).

En este anexo se muestran los fragmentos más relevantes de las dos normativas principales aplicables a este proyecto, es decir, aquellos fragmentos que hacen referencia a la normativa de protección de datos y a la normativa que regula la posición de las matrículas.

Las normativas de Estados Unidos, Canadá, Australia y Nueva Zelanda son muy extensas y pueden variar a lo largo de sus propios territorios, por lo que no se ha considerado mostrarlas en este anexo.

Únicamente se van a mostrar en este anexo las normativas europeas y españolas, puesto que algunos algoritmos se han planteado de tal forma que únicamente son viables en este territorio, y son argumentativamente las normativas más restrictivas a nivel global.

Las normativas mostradas son fragmentos del informe 297/2012 (respecto a la ley de protección de datos de carácter personal en instalaciones con lectura de matrículas) y del reglamento UE 1003/2010: Instalación y homologación de matrículas de vehículos.

3.1. Informe 297/2012 de Ley Orgánica 15/1999: Protección de datos de Carácter Personal en instalaciones de lectura de matrículas.

Informe 297/2012.

I

Se plantean en la presente consulta diversas cuestiones relacionadas con la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de datos de Carácter Personal, en adelante LOPD, y su normativa de desarrollo respecto de la posible instauración de un sistema de captura y tratamiento del dato relativo a la matrícula de los vehículos que acceden a las estaciones de servicio, a fin de identificar los vehículos que abandonan las estaciones de servicio sin pagar el carburante que les ha sido suministrado. En particular, se plantea si dicho sistema podría quedar incardinado en el fichero ya declarado sobre videovigilancia; si es necesaria la obtención del consentimiento y los principios que serían aplicables, así como el plazo máximo de conservación de tales datos.

En primer lugar, estudiaremos si nos encontramos ante un dato de carácter personal que determinará la aplicación de la LOPD. Esta Agencia ha reiterado en diversos informes que **el dato de la matrícula de un automóvil es un dato de carácter personal**, en la medida en que esté incorporado en un fichero, puesto que la existencia del Registro de Vehículos permite conocer los datos del titular del vehículo sin esfuerzos desproporcionados. En particular, en informe de 8 de febrero de 2007 - reiterado en el de 12 de junio de 2012 - se llegaba a dicha conclusión por los siguientes argumentos:

"Ello exige, en primer lugar, analizar si los datos de la placa de matrícula de un vehículo han de ser considerados como datos de carácter personal, a tenor de lo dispuesto en la Ley Orgánica 15/1999, de 13 de diciembre, de Protección de datos de Carácter Personal.

En este sentido, el artículo 2.1 de la Ley Orgánica dispone establece que "La presente Ley Orgánica será de aplicación a los datos de carácter personal registrados en soporte físico, que los haga susceptibles de tratamiento, y a toda modalidad de uso posterior de estos datos por los sectores público y privado".

Por su parte, el artículo 3 a) de dicha Ley añade que se entenderá por datos de carácter personal "cualquier información concerniente a personas físicas identificadas o identificables". En este mismo sentido se pronuncia el artículo 2 a) de la Directiva 95/46/CE del Parlamento y del Consejo, de 24 de octubre de 1995, relativa a la Protección de las Personas Físicas en lo que respecta al tratamiento de datos profesionales y a la libre circulación de estos datos, que dispone "toda información sobre una persona física identificada o identificable (el «interesado»); se considerará identificable toda persona cuya identidad pueda determinarse, directa o indirectamente, en particular mediante un número de identificación o uno o varios elementos específicos, característicos de su identidad física, fisiológica, psíquica, económica, cultural o social".

Para interpretar cuándo ha de considerarse que nos encontramos ante un dato de carácter personal esta Agencia ha venido siguiendo el criterio sustentado por las distintas Recomendaciones emitidas por el Comité de Ministros del Consejo de Europa, en las que se indica que la persona deberá considerarse identificable cuando su identificación no requiere plazos o actividades desproporcionados. En este sentido se pronuncia el artículo 5 o) del Proyecto de Reglamento de desarrollo de la Ley Orgánica 15/1999, que fue sometido a informe de esta Agencia, habiendo el mismo sido emitido en fecha 17 de enero de 2007.

En consecuencia, el tratamiento de los datos correspondientes a las placas de matrícula de los vehículos se encontrará sometido a lo dispuesto en la Ley Orgánica 15/1999 en caso de que se considere a los datos contenidos en dichas placas datos de carácter personal, para lo que sería preciso que dichos datos pudieran permitir la identificación de un individuo sin que ello exija plazos o esfuerzos desproporcionados.

El artículo 5 h) del Texto articulado de la Ley sobre Tráfico, Circulación de Vehículos a Motor y Seguridad Vial, aprobado por Real Decreto Legislativo 339/1990, de 2 de marzo, establece que "se atribuyen al Ministerio del Interior las siguientes competencias en el ámbito de esta Ley, sin perjuicio de las que tengan asumidas las Comunidades Autónomas en sus propios Estatutos (...) los registros de vehículos, de conductores e infractores, de profesionales de la enseñanza de la conducción, de centros de formación de conductores, de los centros de reconocimiento para conductores de vehículos a motor y de manipulación de placas de matrícula, en la forma que reglamentariamente se determine".

En consecuencia, el citado precepto reconoce la subsistencia del Registro de Vehículos, creado por el artículo 244 del Código de la Circulación, aprobado por Decreto de 25 de septiembre de 1934, habilitando expresamente al desarrollo reglamentario del Texto Refundido para establecer el régimen del citado Registro.

Dicho desarrollo se produjo a través de la aprobación del Reglamento General de Vehículos, en virtud de Real Decreto 2822/1998, de 23 de diciembre, cuyo artículo segundo establece en su párrafo primero que "la Jefatura Central de Tráfico llevará un Registro de todos los vehículos matriculados, que adoptará para su funcionamiento medios informáticos y en el que figurarán, al menos, los datos que deben ser consignados obligatoriamente en el permiso o licencia de circulación, así como cuantas vicisitudes sufran posteriormente aquéllos o su titularidad".

En cuanto a su finalidad, el párrafo segundo del precepto previene que "estará encaminado preferentemente a la identificación del titular del vehículo, al conocimiento de las características técnicas del mismo y de su aptitud para circular, a la comprobación de las inspecciones realizadas, de tener concertado el seguro obligatorio de automóviles y del cumplimiento de otras obligaciones legales, a la constatación del Parque de Vehículos y su distribución, y a otros fines estadísticos".

Por último, y en lo atinente a la publicidad de sus datos, el párrafo tercero del citado artículo 2 añade que "el Registro de Vehículos (...) será público para los interesados y terceros que tengan interés legítimo y directo, mediante simples notas informativas o certificaciones". En consecuencia, se establece el carácter público del Registro, bastando para la consulta de sus datos la alegación de la existencia de un interés legítimo y directo en la consulta.

De lo que se ha venido indicando cabe desprender que la identificación del titular de los vehículos cuya matrícula sea conocida únicamente exigirá la consulta del Registro de Vehículos, cuida finalidad esencial es la identificación del titular, para lo cual únicamente será necesaria la invocación del interés legítimo del solicitante.

En consecuencia, cabe considerar que la identificación del titular del vehículo no exige esfuerzos o plazos desproporcionados, por lo que el tratamiento del dato de la matrícula habrá de ser considerado como tratamiento de un dato de carácter personal".

Si el dato de matrícula puede ser considerado un dato de carácter personal, siempre que esté incorporado en un fichero que lo haga susceptible de tratamiento, habrá de estarse a la normativa de protección de datos para estudiar el fichero que pretende crearse.

II

Siendo aplicable a un fichero de matrículas de coches la LOPD y su normativa de desarrollo, queremos comenzar indicando que el mismo no puede quedar incardinado dentro del fichero de videovigilancia que la consultante dice tener creado e inscrito, por alterarse la finalidad del fichero. Así, se afirma que se dispone ya de un sistema de videovigilancia con el consiguiente fichero y correspondientes avisos informativos que cumple lo revisto en la Instrucción 1/2006 de esta Agencia. Por tanto, tales ficheros tendrán una finalidad eminente de vigilancia, con unos claros fines de seguridad.

Sin embargo, lo que ahora se pretende, según el propio escrito de consulta, es "la captura y tratamiento del dato relativo a la matrícula de los vehículos". Según el Anexo I.1, "cuando un vehículo acceda a la calle de surtidores de alguna de las EESS de ..., un lector capturará la matrícula, de forma automática, y un OCR interpretará y extraerá los caracteres de la matrícula, volcándolos en una base de datos". Se trata, por tanto, de la **creación de un nuevo fichero** según la definición del art. 3.b) LOPD, con datos diferentes de los derivados de la captación y grabación de imágenes. Se constituiría un nuevo fichero, por cuanto no se almacenarían en el mismo imágenes captadas en las estaciones de servicio, sino el dato de matrícula de los vehículos que acceden a las estaciones de servicio. Además, se trataría de la creación de un nuevo fichero cuya finalidad, según se expone, no es la vigilancia y seguridad, sino que su finalidad primordial sería "la necesidad de erradicar, o al menos mitigar, dichos comportamientos fraudulentos", esto es, "personas que suministran carburante a sus vehículos en EESS de …. Abandonan sus instalaciones sin proceder a realizar el correspondiente pago".

No obstante, nuestra legislación sólo permite el tratamiento de datos personales cuando son obtenidos para una finalidad determinada, explícita y legítima. Debe aquí tomarse en consideración la jurisprudencia del Tribunal Constitucional que indica que el derecho a la protección de datos personales es un derecho fundamental autónomo, diferenciado del derecho fundamental a la intimidad, (STC 292/2000) por lo que ha de ajustarse al **principio de finalidad y proporcionalidad del tratamiento**.

El artículo 4.1 de la Ley Orgánica 15/1999 consagra el principio de proporcionalidad en el tratamiento, estableciendo que "los datos de carácter personal sólo se podrán recoger para su tratamiento, así como someterlos a dicho tratamiento, cuando sean adecuados, pertinentes y no excesivos en relación con el ámbito y las finalidades determinadas, explícitas y legítimas para las que se hayan obtenido". De ello se desprende la necesidad de que el tratamiento de un determinado dato de carácter personal, como sería, en este caso, la huella dactilar, deba ser proporcionada a la finalidad que lo motiva.

Respecto a la proporcionalidad ha señalado el **Tribunal Constitucional en la Sentencia 207/1996** que se trata de una exigencia común y constante para la constitucionalidad de cualquier medida restrictiva de derechos fundamentales, entre ellas las que supongan una injerencia en los derechos a la integridad

física y a la intimidad, y más en particular de las medidas restrictivas de derechos fundamentales adoptadas en el curso de un proceso penal viene determinada por la estricta observancia del principio de proporcionalidad.

En este sentido, hemos destacado que, para comprobar si una medida restrictiva de un derecho fundamental supera el juicio de proporcionalidad, es necesario constatar si cumple los tres siguientes requisitos o condiciones: *"si tal medida es susceptible de conseguir el objetivo propuesto (juicio de idoneidad); si, además, es necesaria, en el sentido de que no exista otra medida más moderada para la consecución de tal propósito con igual eficacia (juicio de necesidad); y, finalmente, si la misma es ponderada o equilibrada, por derivarse de ella más beneficios o ventajas para el interés general que perjuicios sobre otros bienes o valores en conflicto (juicio de proporcionalidad en sentido estricto)".*

De este modo, si dicha finalidad pudiera ser conseguida por la realización de una actividad distinta al citado tratamiento, sin que dicha finalidad sea alterada o perjudicada, debería optarse por esa última actividad, dado que el tratamiento de los datos de carácter personal supone, tal y como consagra nuestro Tribunal Constitucional, en Sentencia 292/2000, de 30 de noviembre, una limitación del derecho de la persona a disponer de la información referida a la misma. En este sentido, podemos recordar el análisis efectuado por el Grupo de trabajo creado por el artículo 29 de la Directiva 95/46/CE, en el Documento de Trabajo sobre biometría, de fecha 1 agosto de 2003.

Por tanto, deberán realizarse los tres juicios propuestos por el TC. Entendemos que concurre el juicio de idoneidad, cumpliendo la creación y utilización del fichero la finalidad perseguida. Ahora bien, habrá de determinarse si no cabe el uso de otra medida más moderada que consiguiera el mismo objetivo. En este sentido, podrían existir medios que no afectando al tratamiento de datos personales permitieran la consecución del mismo objetivo, como son la implantación de un sistema prepago del carburante – incluso con tarjetas de crédito - , o bien la previa autorización del suministro por parte de los empleados de la estación de servicio, o incluso que el suministro sólo se realice por empleados directamente que no permitan abandonar la estación sin el pago del suministro. En este sentido, entendemos que a priori podrían existir otros medios de consecución de las finalidades perseguidas que fueran menos invasivos en la esfera de los datos personales.

Ahora bien, este juicio de necesidad y de ponderación en sentido estricto es necesariamente casuístico. Y puesto que no conocemos en profundidad la situación de la consultante, ni otras posibles soluciones de seguridad que se estén planteando, expondremos las obligaciones que, en su caso, y con carácter general, se derivarían de la implantación de la base de datos de matrículas prevista.

Si, efectivamente, procediera la creación del fichero propuesto por superar el juicio de proporcionalidad con los tres requisitos indicados, habrían de cumplirse las obligaciones de la LOPD. Por supuesto, todo tratamiento de

datos personales debe estar legitimado, en los términos del artículo 6 LOPD. En el supuesto planteado, nos encontraríamos ante el requerimiento de consentimiento inequívoco del afectado, que se entenderá prestado por el hecho de que, estando debida y suficientemente informado del tratamiento de sus datos para los fines previstos, manifieste – no necesariamente mediante una declaración de voluntad, sino mediante sus actos – su consentimiento al tratamiento; en el caso de la consulta, si habiendo sido informado con carácter previo y de manera suficiente, decide acceder a la estación de servicio y repostar.

En primer lugar, la creación de un fichero para la finalidad prevista debe resultar "necesario para el logro de la actividad u objeto legítimos de la persona, empresa o entidad titular y se respeten las garantías que esta Ley establece para la protección de las persona ", como prevé el artículo 25 LOPD.

En este sentido, la creación de ficheros exige la notificación previa a esta Agencia de Protección de Datos, para la práctica de la consiguiente inscripción de conformidad con los artículos 26 y 27 LOPD y 55 y ss. del Reglamento de desarrollo de dicha Ley Orgánica aprobado por Real Decreto 1720/2007 de 21 de diciembre, en adelante RDLOPD.

El tratamiento de los datos personales en cuestión exigirá el cumplimiento del deber de información a que se refiere el art. 5 LOPD, debiendo en consecuencia incluirse la información a que se refiere dicho precepto en un lugar visible con carácter previo al tratamiento de los datos. Si el sistema se implantara de forma que un lector capturará la matrícula de forma automática cuando un vehículo acceda a la calle de surtidores, en alguno o algunos lugares absolutamente visibles de dicha calle de surtidores deberá informarse a todos los usuarios de los extremos del artículo 5 LOPD. En segundo lugar, deberá permitirse el ejercicio de los derechos de acceso, rectificación, cancelación y oposición en la forma indicada en los artículos 15 y ss. LOPD y Título III RDLOPD.

En tercer lugar, la existencia de un fichero que incorpore datos de carácter personal exige el cumplimiento del deber de seguridad de los datos a que se refiere el artículo 9 LOPD, adoptándose "las medidas de índole técnica y organizativas necesarias que garantice la seguridad de los datos de carácter personal y evite su alteración, pérdida, tratamiento o acceso no autorizado, habida cuenta del estado de la tecnología, la naturaleza de los datos almacenados y los riesgos a que están expuestos, ya provengan de la acción humana del medio físico o natural". Medidas de seguridad que son desarrolladas en el Título VIII RDLOPD, a adoptar por el responsable del fichero. En el supuesto de hecho planteado, si únicamente se trataran los datos de la matrícula del coche así como la existencia, en su caso, de uno o varios impagos, sólo sería exigible según el artículo 81 RDLOPD un nivel de seguridad básico, sin perjuicio "de las disposiciones legales o reglamentarias específicas vigentes que pudieran resultar de aplicación en cada caso o las que por propia iniciativa adoptase el responsable del fichero". Junto con la

implantación de las medidas de seguridad de nivel básico, deberá elaborarse el documento de seguridad en los términos del art. 88 RDLOPD.

En cualquier caso, todo tratamiento de datos personales ha de cumplir los principios enumerados en el artículo 4 LOPD, de forma que los datos sean "adecuados, pertinentes y no excesivos en relación con el ámbito y las finalidades determinadas, explícitas y legítimas para las que se hayan obtenido". Si la finalidad del fichero está relacionada con el control de los impagos por parte del acreedor, tendría sentido que se trataran y conservaran los datos de aquellas matrículas de los vehículos que pudieran estar asociados a uno o varios impagos, a los efectos de cursar la oportuna denuncia a las Fuerzas y Cuerpos de Seguridad y del uso de este dato por parte del responsable del fichero. Ahora bien, pudiera resultar excesivo el tratamiento del dato de la matrícula cuando la misma no ha quedado asociada a ningún impago. Es decir, si bien la recogida del dato de todas las matrículas de los coches cuando acceden a las calles de surtidores pudiera estar justificado por la finalidad del fichero, no así el tratamiento posterior, mediante su conservación, de aquellas matrículas que no hayan quedado asociadas a un impago. Serían datos excesivos en relación con la finalidad para la que se han obtenido, por lo que, una vez que un vehículo abandone la estación de servicio sin haber incurrido en impago, el sistema deberá instaurar un procedimiento para cancelar tales datos; todo ello de conformidad con el artículo 4.5 LOPD que indica que "Los datos de carácter personal serán cancelados cuando hayan dejado de ser necesarios o pertinentes para la finalidad para la cual hubieran sido recabados o registrados. No serán conservados en forma que permita la identificación del interesado durante un período superior al necesario para los fines en base a los cuales hubieran sido recabados o registrados".

Esta misma norma es aplicable a los datos que, aun asociados a un impago, sean conservados. Tal conservación no puede ser permanente, sino que sólo estará permitida en la medida en que siga cumpliendo con la finalidad legítima para la que se establece el sistema. Mientras los datos sean necesarios para cursar la oportuna denuncia a las Fuerzas y Cuerpos de Seguridad, podrán conservarse. Igualmente podría mantenerse mientras el propio acreedor, la empresa de suministro de carburante, no suministre carburante a todos aquellos vehículos asociados a un impago, durante un periodo de tiempo razonable y siempre dentro del plazo de prescripción de la acción para exigir el pago del suministro en cuestión.

En cuanto a la posible cesión de los datos en cuestión y el tratamiento por otras empresas, debemos en primer lugar determinar quién es el responsable del fichero en cuestión. Para ello deberá estarse a la definición del artículo 3.d) LOPD como "persona física o jurídica, de naturaleza pública o privada, u órgano administrativo, que decida sobre la finalidad, contenido y uso del tratamiento". La actividad de distribución al por menor de carburante y combustibles petrolíferos, podrá ser ejercida libremente por cualquier persona física o jurídica de conformidad con el artículo 43.2 de la Ley 34/1988 de 7 de octubre de Hidrocarburos. Ahora bien, tal y como indica ese mismo precepto, "Las instalaciones utilizadas para el ejercicio de esta actividad deberán contar

con las autorizaciones administrativas preceptivas para cada tipo de instalación así como cumplir con el resto de la normativa vigente que en cada caso sea de aplicación". El artículo 44 de la misma norma prevé la creación de un registro de instalaciones de distribución al por menor, teniendo en cuenta a estos efectos el Reglamento de Instalaciones Petrolíferas, aprobado por el Real Decreto 2085/1994, de 20 de octubre, que prevé la autorización administrativa previa para las labores de refino, transporte y almacenamiento a que se refieren los artículos 39 y 40 de la Ley, estando el resto sometidas a la mera inscripción en los correspondientes registros, sin perjuicio del deber de cumplimiento de todas las medidas de seguridad y de las instrucciones técnicas a que se refiere el mencionado reglamento.

En este sentido, de la consulta no se desprenden datos suficientes que pudieran determinar si es la empresa consultante la que decide sobre la finalidad, uso y contenido del tratamiento; si así lo fuera, ésta sería considerada responsable del fichero, debiendo en consecuencia asegurar el cumplimiento de todas las obligaciones derivadas de la LOPD.

El empleado de la estación de servicio, o en su caso el responsable de cada una de las instalaciones, que pudieran tener acceso al fichero podrían ser considerados como usuarios, concepto definido en el art. 5.2.p) RDLOPD como "sujeto o proceso autorizado para acceder a datos o recursos".

Si la empresa consultante pudiera ser considerada como responsable del fichero, pudiendo permitir el acceso de los datos a diversos usuarios, nos encontraríamos ante un tratamiento de datos personales por parte del propio titular del fichero, que podría, en su caso, consultarse por todas las estaciones de servicio.

Ahora bien, una cosa es el tratamiento que realice el propio responsable del fichero, y otra diferente el hecho de permitir la consulta a otras personas o entidades diferentes.. Tanto si cada una de las estaciones de servicio debiera ser considerada como responsable del fichero, existiendo una verdadera cesión para poder comunicar los datos a otras estaciones de servicio; como si la cesión se produce "a Estaciones de Servicio de otras entidades del Grupo empresarial", podríamos encontrarnos ante una cesión de datos personales, definida en el artículo 3.i) LOPD como "toda revelación de datos realizada a una persona distinta del interesado". Y no sería entonces de aplicación el artículo 6 LOPD, sino el artículo 11 LOPD. Puesto que no consideramos que la cesión en cuestión implique necesariamente la conexión del tratamiento con ficheros de terceros, y no teniendo el supuesto de hecho cabida en ninguno de los restantes apartados del art. 11.2 LOPD, la cesión exigiría el previo consentimiento del interesado en los términos del artículo 11.1. La cesión, que como decimos habría de estar expresamente consentida, sólo sería posible "para el cumplimiento de fines directamente relacionados con las funciones legítimas del cedente y del cesionario". Puesto que la consulta sólo enuncia esta posibilidad en sentido muy amplio, sin identificar las funciones ni fines del cesionario, no podemos estudiar en sentido más concreto esta cesión. Baste indicar que el acceso y tratamiento de los datos por parte de otras empresas

del grupo podría constituir una cesión y debería cumplir las obligaciones de la LOPD.

3.2. Reglamento UE 1003/2010: Instalación y homologación de matrículas de vehículos.

REGLAMENTOS

REGLAMENTO (UE) No 1003/2010 DE LA COMISIÓN

de 8 de noviembre de 2010

relativo a los requisitos para la homologación de tipo del emplazamiento y la instalación de las placas de matrícula traseras en los vehículos de motor y sus remolques y por el que se desarrolla el Reglamento (CE) no 661/2009 del Parlamento Europeo y del Consejo, relativo a los requisitos de homologación de tipo referentes a la seguridad general de los vehículos de motor, sus remolques y sistemas, componentes y unidades técnicas independientes a ellos destinados

(Texto pertinente a efectos del EEE)

LA COMISIÓN EUROPEA,

Visto el Tratado de Funcionamiento de la Unión Europea,

Visto el Reglamento (CE) no 661/2009 del Parlamento Europeo y del Consejo, de 13 de julio de 2009, relativo a los requisitos de homologación de tipo referentes a la seguridad general de los vehículos de motor, sus remolques y sistemas, componentes y unidades técnicas independientes a ellos destinados, y, en particular, su artículo 14, apartado 1, letra a),

Considerando lo siguiente:

(1) El Reglamento (CE) no 661/2009 es un Reglamento particular a los efectos del procedimiento de homologación de tipo establecido en la Directiva 2007/46/CE del Parlamento Europeo y del Consejo, de 5 de septiembre de 2007, por la que se crea un marco para la homologación de los vehículos de motor y de los remolques, sistemas, componentes y unidades técnicas independientes destinados a dichos vehículos (Directiva marco).

(2) En virtud del Reglamento (CE) no 661/2009 se deroga la Directiva 70/222/CEE del Consejo, de 20 de marzo de 1970, relativa a la aproximación de las legislaciones de los Estados miembros sobre el emplazamiento e instalación de las placas traseras de matrícula de los vehículos a motor y de sus remolques. Los requisitos establecidos en dicha Directiva deben trasladarse al presente Reglamento y, en su caso, modificarse a fin de adaptarlos al desarrollo de los conocimientos científicos y técnicos.

(3) En el Reglamento (CE) no 661/2009 se establecen disposiciones fundamentales sobre los requisitos para la homologación de tipo de los vehículos de motor y sus remolques con respecto al emplazamiento y la

instalación de las placas de matrícula traseras. Por consiguiente, es necesario establecer también los procedimientos, ensayos y requisitos específicos para esta homologación de tipo.

(4) Las medidas previstas en el presente Reglamento se ajustan al dictamen del Comité Técnico sobre Vehículos de Motor.

HA ADOPTADO EL PRESENTE REGLAMENTO:

Artículo 1

Definiciones.

A efectos del presente Reglamento, se entenderá por:
1) «tipo de vehículo con respecto al emplazamiento y la instalación de las placas de matrícula traseras», los vehículos que no presenten diferencias esenciales entre sí en los aspectos siguientes:

— las dimensiones del espacio destinado a la colocación y fijación de la placa de matrícula trasera,

— la ubicación del espacio destinado a la colocación y fijación de la placa de matrícula trasera,

— la forma de la superficie destinada a la colocación y fijación de la placa de matrícula trasera;

2) «superficie prácticamente plana», la superficie de material sólido —que también puede consistir en una malla de figuras geométricas o una rejilla— que presente un radio de curvatura de 5 000 mm como mínimo;

3) «superficie de malla de figuras geométricas», la superficie consistente en un conjunto de orificios circulares, ovalados, rombales, rectangulares o cuadrados distribuidos de manera uniforme a intervalos que no superen los 15 mm;

4) «superficie de rejilla», la superficie formada por barras paralelas distribuidas de manera uniforme con una distancia entre sí que no supere los 15 mm;

5) «superficie nominal», la superficie teórica geométricamente perfecta sin tener en cuenta las irregularidades, como pueden ser las protuberancias o hendiduras;

6) «plano longitudinal medio del vehículo», el plano de simetría del vehículo o, si el vehículo no es simétrico, el plano vertical longitudinal que atraviesa el centro de los ejes del vehículo

7) «inclinación», el grado de desviación angular con respecto a un plano vertical.

Artículo 2

Disposiciones para la homologación de tipo CE de un vehículo de motor o un remolque con respecto al emplazamiento y la instalación de las placas de matrícula traseras.

1. El fabricante o su representante presentará ante la autoridad de homologación la solicitud de homologación de tipo CE de un vehículo con respecto al emplazamiento y la instalación de las placas de matrícula traseras en los vehículos de motor y sus remolques.

2. La solicitud se redactará de conformidad con el modelo de ficha de características que figura en el anexo I, parte 1.

3. Si se cumplen los requisitos pertinentes del anexo II del presente Reglamento, la autoridad de homologación concederá una homologación de tipo CE y asignará un número de homologación de tipo de conformidad con el sistema de numeración que figura en el anexo VII de la Directiva 2007/46/CE. Un Estado miembro no podrá asignar el mismo número a otro tipo de vehículo.

4. A efectos del apartado 3, la autoridad de homologación de tipo expedirá un certificado de homologación de tipo CE de conformidad con el modelo que figura en el anexo I, parte 2.

Artículo 3

Validez y extensión de las homologaciones concedidas con arreglo a la Directiva 70/222/CEE.

Las autoridades nacionales permitirán la venta y la puesta en servicio de vehículos a los que se haya concedido la homologación de tipo antes de la fecha mencionada en el artículo 13, apartado 2, del Reglamento (CE) no 661/2009 y seguirán concediendo a estos vehículos la extensión de sus homologaciones con arreglo a lo dispuesto en la Directiva 70/222/CEE.

Artículo 4

Entrada en vigor.

El presente Reglamento entrará en vigor el vigésimo día siguiente al de su publicación en el Diario Oficial de la Unión
Europea. El presente Reglamento será obligatorio en todos sus elementos y directamente aplicable en cada Estado miembro.

Hecho en Bruselas, el 8 de noviembre de 2010.

ANEXO II

Requisitos para el emplazamiento y la instalación de las placas de matrícula traseras

1. REQUISITOS

1.1. Forma y dimensiones del emplazamiento de una placa de matrícula trasera.

1.1.1. El emplazamiento consistirá en una superficie rectangular plana o prácticamente plana con las dimensiones mínimas siguientes:

anchura: 520 mm
altura: 120 mm
o
anchura: 340 mm
altura: 240 mm

1.1.2. La superficie que va a quedar cubierta por la placa de matrícula podrá contener orificios o huecos.

1.1.2.1. En el caso de los vehículos de la categoría M1, si el orificio o hueco no supera los 40 mm de ancho, habrá que tener en cuenta su longitud.

1.1.3. La superficie que va a quedar cubierta por la placa de matrícula podrá tener protuberancias, siempre y cuando estas no sobresalgan más de 5,0 mm con respecto a la superficie nominal. No se tendrán en cuenta los parches de materiales muy suaves, como la espuma o el fieltro, utilizados para eliminar las vibraciones de la placa de matrícula.

1.2. Emplazamiento e instalación de una placa de matrícula trasera.

1.2.1. El emplazamiento estará diseñado de manera que la placa de matrícula, una vez instalada con arreglo a las instrucciones del fabricante, tenga las características siguientes:

1.2.1.1. Posición de la placa con respecto al plano longitudinal medio del vehículo:

1.2.1.1.1. El punto central de la placa no estará situado a la derecha del plano longitudinal medio del vehículo.

1.2.1.2. Posición de la placa con respecto al plano longitudinal vertical del vehículo:

1.2.1.2.1. la placa estará perpendicular al plano longitudinal del vehículo;

1.2.1.2.2. El borde izquierdo de la placa no podrá estar situado a la izquierda del plano vertical que sea paralelo al plano longitudinal medio del vehículo y toque el borde más exterior del vehículo.

1.2.1.3. Posición de la placa con respecto al plano transversal vertical:

1.2.1.3.1. la placa podrá estar inclinada respecto a la vertical:

1.2.1.3.1.1. con una inclinación mínima de – 5° y máxima de 30°, siempre y cuando el borde superior de la placa no esté a más de 1,20 m del suelo;

1.2.1.3.1.2. con una inclinación mínima de – 15° y máxima de 5°, siempre y cuando el borde superior de la placa esté a más de 1,20 m del suelo.

1.2.1.4. Altura de la placa con respecto al suelo:

1.2.1.4.1. la altura del borde inferior de la placa con respecto al suelo será de 0,30 m como mínimo;

1.2.1.4.2. la altura del borde superior de la placa con respecto al suelo será de 1,20 m como máximo; no obstante, cuando no sea posible cumplir el requisito relativo a la altura debido al diseño del vehículo, la altura máxima podrá exceder de 1,20 m, siempre y cuando se mantenga tan cerca de ese límite como lo permitan las características de fabricación del vehículo, y, en cualquier caso, no superará los 2,00 m.

1.2.1.5. Visibilidad geométrica:

1.2.1.5.1. Cuando la altura del borde superior de la placa con respecto al suelo no exceda de 1,20 m, la placa será visible en todo su emplazamiento, incluso en los planos siguientes:

— los dos planos verticales que tocan los bordes laterales de la placa y forman con el plano longitudinal medio del vehículo un ángulo de 30° medido hacia el exterior,

— el plano que toca el borde superior de la placa y forma con la horizontal un ángulo de 15° medido hacia
arriba y

— el plano horizontal que atraviesa el borde inferior de la placa.

1.2.1.5.2. Cuando la altura del borde superior de la placa con respecto al suelo exceda de 1,20 m, la placa será visible en todo su emplazamiento, incluso en los planos siguientes:

— los dos planos verticales que tocan los bordes laterales de la placa y forman con el plano longitudinal medio del vehículo un ángulo de 30° medido hacia el exterior,

— el plano que toca el borde superior de la placa y forma con la horizontal un ángulo de 15° medido hacia
arriba y

— el plano que toca el borde inferior de la placa y forma con la horizontal un ángulo de 15° medido hacia
abajo.

1.2.1.6. El espacio entre los bordes de la placa de matrícula colocada y fijada y la superficie real del emplazamiento de la placa de matrícula no excederá de 5,0 mm a lo largo de todo el contorno de la placa de matrícula.

1.2.1.6.1. El espacio máximo establecido podrá superarse localmente cuando se mida en un orificio o hueco en el interior de la superficie de malla de figuras geométricas o entre las barras paralelas de la superficie de rejilla.

1.2.2. A efectos de los requisitos relativos al dispositivo de alumbrado de la placa de matrícula trasera, se tendrán en cuenta la posición y la forma reales de la placa de matrícula colocada y fijada con arreglo a lo establecido en el punto 1.2, en particular el radio de curvatura resultante.

1.2.3. Cuando el emplazamiento de la placa de matrícula trasera quede oculto en el interior de los planos de visibilidad geométrica debido a la instalación de cualquier dispositivo mecánico de acoplamiento, se indicará en el informe de ensayo y se declarará en el certificado de homologación de tipo CE.

2. PROCEDIMIENTO DE ENSAYO

2.1. Determinación de la inclinación vertical y la altura de la placa de matrícula con respecto al suelo.

2.1.1. Antes de proceder a la medición, se colocará el vehículo sobre una superficie lisa, con la masa ajustada a la declarada por el fabricante en orden de marcha, pero sin conductor.

2.1.2. Los vehículos equipados con una suspensión hidroneumática, hidráulica o neumática o un dispositivo de corrección automática de la altura en función de la carga se someterán a ensayo con la suspensión o el dispositivo en las condiciones de marcha normal especificadas por el fabricante.

2.1.3. Si la placa de matrícula está orientada hacia abajo, el resultado de las mediciones relativas a la inclinación se expresará en negativo.

2.2. La medición de las protuberancias se realizará perpendicularmente y directamente con respecto a la superficie nominal que va a quedar cubierta por la placa de matrícula.

2.3. La medición del espacio entre el borde de la placa de matrícula colocada y fijada y la superficie real se realizará perpendicularmente y directamente con respecto a la superficie real que va a quedar cubierta por la placa de matrícula.

2.4. La placa de matrícula utilizada para verificar la conformidad tendrá uno de los dos tamaños indicados en el punto 1.1.1.