



S
I
S
M
A
T

Sistema de Autorización de Vehículos Automatizado

Petteri Simo Raafael Ketola

Desarrollo de Aplicaciones Multiplataforma (DAM)

2º Año CFGS

30-05-2025

Resumen

Este Trabajo de Fin de Grado tiene como propósito el desarrollo de un sistema de reconocimiento automático de matrículas (ANPR) mediante técnicas de visión computacional y aprendizaje profundo, orientado a facilitar el control de acceso vehicular en zonas urbanas, aparcamientos y entornos regulados como las Zonas de Bajas Emisiones (ZBE). El principal reto abordado fue lograr una solución precisa, eficiente y de bajo coste frente a alternativas comerciales, muchas de las cuales requieren licencias propietarias y una inversión significativa.

El estudio se diseñó mediante una arquitectura modular basada en software libre: OpenCV para el procesamiento de imágenes, YOLOv8 para la detección de matrículas y EasyOCR como motor de reconocimiento óptico de caracteres. Se implementó una base de datos local para la gestión de vehículos reconocidos, así como un formulario de edición sencillo para usuarios sin conocimientos técnicos. El sistema fue probado en condiciones variables de iluminación, ángulo y calidad de imagen, simulando escenarios reales de uso.

Los principales hallazgos muestran que el sistema alcanza niveles de precisión aceptables (por encima del 85%) en detección y lectura de matrículas europeas, manteniendo un tiempo de respuesta realista para aplicaciones en tiempo real. Además, se comprobó su capacidad de adaptación a diferentes entornos sin necesidad de ajustes manuales complejos.

En conclusión, el proyecto demuestra que es viable implementar un sistema ANPR funcional y escalable usando tecnologías de código abierto. Se proponen futuras mejoras, como la integración con plataformas web, el uso de indicadores visuales en accesos físicos o la ampliación del reconocimiento a otros formatos de matrículas. Este trabajo contribuye a la democratización del acceso a soluciones de control vehicular inteligente, con potencial aplicación en ámbitos tanto públicos como privados.

Abstract

This Final Degree Project aims to develop an Automatic Number Plate Recognition (ANPR) system using computer vision and deep learning techniques, designed to facilitate vehicle access control in urban areas, parking facilities, and regulated zones such as Low Emission Zones (LEZ). The main challenge addressed was to achieve a precise, efficient, and cost-effective solution compared to commercial alternatives, many of which require proprietary licenses and significant investment.

The system design is modular and based on open-source software: OpenCV for image processing, YOLOv8 for license plate detection, and EasyOCR as the optical character recognition engine. A local database was implemented to manage recognized vehicles, along with a simple editing form for end users without technical expertise. The system was tested under varying lighting conditions, angles, and image qualities to simulate real-world scenarios.

Key findings show that the system achieves acceptable accuracy levels (above 85%) in the detection and recognition of European license plates, while maintaining real-time response speeds suitable for practical applications. Additionally, it proved adaptable to different environments without the need for complex manual configurations.

In conclusion, the project demonstrates the feasibility of implementing a functional and scalable ANPR system using open-source technologies. Future improvements are suggested, such as integration with web platforms, the use of visual indicators at physical access points, and the extension of recognition capabilities to other license plate formats. This project contributes to the democratization of intelligent vehicle control systems, with potential applications in both public and private sectors.

Contenidos

Resumen.....	2
Abstract.....	3
1. Introducción.....	5
2. Origen del Trabajo.....	6
3. Análisis de mercado.....	7
3.1 Estado del mercado.....	7
3.2 Comparativa con productos similares.....	8
4. Cronografía.....	11
4.1 Idea.....	11
4.2 Análisis del mercado.....	11
4.3 Requisitos.....	12
4.4.1 Requisitos funcionales.....	12
4.4.2 Requisitos no funcionales.....	13
4.5 Definición de MVP.....	13
4.6 Diagrama de flujo.....	14
4.7 Programación.....	15
4.8 Pruebas.....	16
4.9 Análisis de resultados.....	16
4.10 Documentación.....	17
5. Desarrollo.....	18
5.1 Hardware.....	18
5.2 Programa de detección.....	18
5.3 Base de datos.....	19
5.3.1 Tabla 'usuario'.....	20
5.3.2 Tabla 'matricula'.....	21
5.3.3 Tabla 'administrador'.....	21
5.4 Panel de control.....	22
6. Pruebas.....	24
7. Análisis de resultados.....	29
7.1 Puntos positivos.....	29
7.2 Problemas y limitaciones.....	29

7.3 Comparativa con otros sistemas.....	29
7.4 Interpretación de los resultados.....	30
8. Conclusión.....	31
9. Mejoras de cara a paso a producción.....	33
10. Bibliografía.....	36

1. Introducción

En los últimos años, el uso de sistemas de reconocimiento automático de matrículas (ALPR, por sus siglas en inglés) ha crecido notablemente debido a su aplicabilidad en diversos contextos urbanos e industriales. Estos sistemas permiten automatizar procesos de control de acceso, gestión de aparcamientos, supervisión del tráfico o implementación de zonas de bajas emisiones, facilitando así la toma de decisiones en tiempo real y la mejora de la eficiencia operativa.

Este Trabajo de Fin de Grado tiene como objetivo desarrollar un sistema de reconocimiento de matrículas utilizando tecnologías de visión por computador y algoritmos de inteligencia artificial, haciendo especial énfasis en la combinación de soluciones de código abierto. Para ello, se ha diseñado una arquitectura modular compuesta por tres componentes principales: OpenCV para el preprocesamiento y visualización de imágenes, YOLOv8 como modelo de detección de objetos para localizar las matrículas, e CR para la lectura de los caracteres identificados.

La elección de estas herramientas responde tanto a criterios de rendimiento como de accesibilidad tecnológica. Frente a soluciones comerciales, el sistema propuesto busca demostrar que es posible obtener resultados precisos y consistentes con un coste reducido y mayor flexibilidad de adaptación.

A lo largo del trabajo, se abordan los distintos aspectos del desarrollo del sistema, desde el análisis de requisitos hasta las pruebas de funcionamiento en distintos escenarios. Se incluye también un estudio comparativo con otras soluciones disponibles en el mercado, así como una reflexión sobre las posibles mejoras e integraciones futuras, tales como el uso en dispositivos embebidos o su conexión con sistemas de reserva y asignación de plazas.

Con este proyecto se pretende ofrecer una solución funcional y escalable, alineada con las necesidades actuales de automatización y sostenibilidad en el entorno urbano.

2. Origen del Trabajo

La elección de un tema de trabajo de fin de grado no es fácil. Al final, es donde se pone a prueba las cualidades obtenidas durante el grado. La importancia de realizar un buen trabajo no solo trata de demostrar las capacidades como desarrollador sino de la creación de un producto único e innovador.

Entre los temas tratados durante el curso, detección me sobresalió entre todos por ser de las presentaciones más interesantes que he hecho. Además, las tecnologías de detección y visión computacional se pueden aplicar de muchas maneras.

Darle “ojos” a un ordenador es una de las mejores maneras de expandir su funcionalidad fuera del campo de procesamiento computacional. Una entrada visual de datos puede contener mucha información. Los humanos usamos procesamos información óptica en 13 milisegundos, pero un ordenador requiere mucho más procesamiento al no ser tan optimizada para el trabajo. Aun con las limitaciones que tiene la visión computacional, los avances en detección de modelos, optimización de procesos con IA y procesadores más potentes, permiten el procesamiento de imágenes a tiempo real (<1 segundo).

En la primera reunión con el tutor, discutimos posibilidades del uso de la detección para el TFG. Primero, surgió la idea de un sistema de vigilancia que pudiera detectar movimientos por una zona y reconocer objetos como personas o animales y que grabase estos acontecimientos. La idea no era mala, pero salió la idea de un sistema de detección de matrículas. Por sí solo, un sistema de detección de matrículas no da para mucho, pero implementarlos en un programa de autorización de vehículos usando una base de datos de registro parecía una idea brillante.

Tradicionalmente, las zonas privadas usan sistemas de acceso con interacción de un portero. Este portero comprobaba cada uno de los vehículos que entraban o salían del recinto. Esto requiere la interacción humana y, aunque sea bastante confiable por su simplicidad, podría perfectamente ser reemplazada por un sistema de visión computacional.

La idea estaba clara. Un sistema que usaba tecnologías modernas para la automatización de un trabajo de labor manual.

3. Análisis de mercado

3.1 Estado del mercado

El reconocimiento de matrículas ha sido tradicionalmente una tarea manual o semi-automatizada (por ejemplo, mediante tarjetas, barreras, lectores RFID o vigilancia humana). Sin embargo, el avance en visión computacional ha permitido el desarrollo de sistemas ANPR (Automatic Number Plate Recognition), que automatizan el proceso con mayor velocidad y menor intervención humana. Se compara ambos enfoques considerando criterios como eficiencia, coste, escalabilidad, precisión y aplicabilidad:

Criterio	Sistemas tradicionales	Sistemas ANPR
Velocidad de procesamiento	Lenta (depende de intervención humana)	Rápida (procesamiento en tiempo real)
Precisión	Alta si es verificado por humanos, pero sujeto a errores humanos	Alta (90–98%), aunque depende de las condiciones
Escalabilidad	Limitada (más personal = más coste)	Alta (puede controlar cientos de entradas)
Coste inicial	Bajo (barrera, lector, operador)	Medio–alto (cámaras + software + servidores)
Coste a largo plazo	Alto (sueldos, errores, mantenimiento)	Bajo (mantenimiento técnico, sin personal)
Intervención humana	Necesaria	Opcional o nula
Integración con sistemas	Limitada	Alta (APIs, bases de datos, plataformas cloud)
Aplicabilidad en tiempo real	Baja	Alta
Generación de datos	Manual o inexistente	Automática (fecha, hora, imagen, placa, etc.)

Tabla 1: Comparativa entre sistemas tradicionales y ANPR

Empresas como Smart Parking, previo a 2025, han logrado instalar más de 1.500 sistemas ANPR, destacando su experiencia en sistemas de estacionamiento automatizados.

Estudios

Valoración actual y futura: Según MarketsandMarkets, el mercado global de sistemas ANPR fue valorado en **3.100 millones de USD** en **2022** y se proyecta que alcance los **4.800 millones de USD** en **2027**, con una tasa de crecimiento anual compuesta (CAGR) del **9,2%** durante el período 2022-2027. ([*Automatic Number Plate Recognition System Market by Type, Component, Application, and Region - Global Forecast to 2027, MarketsandMarkets, 2023*](#))

En 2014, se informó que existían más de 8.000 cámaras ANPR en las carreteras del Reino Unido, capturando aproximadamente 26 millones de imágenes diariamente ([*CCTV cameras on Britain's roads capture 26 million images every day, TheGuardian, 2014*](#)).

Zonas de bajas emisiones

En España, la Ley de Cambio Climático y Transición Energética de 2021 establece que todos los municipios con más de 50.000 habitantes deben implementar Zonas de Bajas Emisiones (ZBE). Estas zonas restringen el acceso de vehículos más contaminantes y requieren sistemas de control de accesos activos, como el reconocimiento automático de matrículas (ANPR), para garantizar su efectividad.

Actualmente, 153 municipios están obligados a tener en funcionamiento una ZBE. Sin embargo, según informes recientes, solo 53 ciudades han implementado adecuadamente estas zonas, mientras que otras están en proceso o no han iniciado los trámites correspondientes.

El Gobierno planea reforzar la implementación real de las ZBE mediante un nuevo decreto que obligará a los ayuntamientos a aplicar restricciones efectivas y a imponer sanciones a los vehículos contaminantes. Además, se exigirá que las ZBE estén claramente delimitadas, incluyan sistemas de seguimiento activo y publiquen sus normativas en el acceso nacional de movilidad de la DGT.

Teniendo en cuenta estas condiciones. Aún hay varias ciudades que aún no tienen implementados sistemas ANPR para las zonas de bajas emisiones, indicando una posibilidad de implementar productos asociados.

3.2 Comparativa con productos similares

En el mercado hay una gran cantidad de programas de detección de matrículas. De sistemas comerciales, como PlateSmart ARES, Genetec AutoVu

y Hikvision ANPR System, a no comerciales / Open Source, como OpenALPR y Sighthound. Cada una tiene sus ventajas e inconvenientes.

Licencia

SisMat viene de entrada con licencia AGPL-3.0, ya que según la propia licencia de YOLOv8 requiere que se redistribuya con esa licencia. Esto obliga a que el código sea disponible a cualquier usuario (disponible en mi [github](#)), dando crédito al autor original. En comparación, PlateSmart es de licencia propietaria teniendo que pagar para usar aunque OpenALPR se puede usar gratuitamente gracias a su licencia GPL, igual que el mío.

Detección

La precisión de todos los productos es generalmente alta. Para la detección, todos los productos usan OCR (Reconocimiento Óptico de Caracteres) para la lectura de matrículas y reconocimiento de modelos para la detección de objetos.

En cuanto a OCRs, EasyOCR de mi proyecto se diseñó de base para detección de texto. PlateSmart usa un sistema de detección propia que está optimizado para un alto rendimiento. OpenALPR usa Tesseract que, aunque sea más ligero, pierde precisión, especialmente con imágenes borrosas. EasyOCR ofrece una mejor precisión a costo de uso de recursos que Tesseract, siendo necesario la precisión para lectura de matrículas en condiciones poco controladas. Puede que mi sistema no compare con detección contra PlateSmart por la falta de optimización, pero con entrenamiento avanzado de modelos se podría conseguir un resultado similar.

Para modelos uso YOLOv8, que es mucho más moderno que el sistema de cascadas Haar y YOLOv3 de OpenALPR, mejorando precisión, rendimiento y extensibilidad. Otra vez, no compara con los algoritmos de PlateSmart con precisión de modelos, pero se pueden entrenar modelos propios para casos específicos que requieran más rendimiento.

Interfaz gráfica

PlateSmart ofrece un panel web donde puedes gestionar el sistema completamente. OpenALPR contiene una página web, pero tiene funcionalidades limitadas. SisMat contiene una página web para administrar todo tipo de permisos además de tener un panel de administrador para variables de sistema.

Personalización

PlateSmart ofrece poca personalización del producto final al ser un producto completo. OpenALPR puedes personalizar ciertos parámetros en la página web pero no tanto en la detección. En SisMat, se puede personalizar el panel web a

las necesidades del usuario final. Además, los parámetros de detección se pueden ajustar para una detección más precisa para ciertos entornos.

Característica	PlateSmart ARES (Comercial)	OpenALPR (Libre)	SisMat
Licencia	Propietaria	GPL	AGPL
Precisión	Muy alta (>95%)	Media-alta (80–90%)	Alta (90%)
OCR	Propio	Tesseract	EasyOCR
Reconocimiento de modelos	Propio	Cascadas Haar / YOLOv3	YOLOv8 (moderno, preciso, rápido)
Velocidad	Tiempo real	Decente	Tiempo real (depende de hardware)
Costo	Alto	Gratuito	Gratuito
Personalización	Muy baja	Media	Media - Alta
Hardware necesario	CPU/GPU profesional	Hardware ligero	Hardware medio (GPU opcional)
Interfaz gráfica	Profesional (panel web)	Limitada o inexistente	Personalizable (puede agregarse GUI)
Soporte	Técnico profesional	Comunidad	Propio / Github

Tabla 2: Comparativa de sistemas ANPR disponibles

4. Cronografía

4.1 Idea

Durante las primeras semanas del proyecto, se desarrolla un tema de trabajo pajo ciertas circunstancias. Entre ellos las cualidades que se pretenden cumplir en la selección de un concepto son:

- Tener relación con el campo de la informática y tecnología, preferiblemente aquellas tratadas durante el grado.
- Tener preferiblemente como objetivo cubrir una necesidad real.
- Debe ser factible desarrollarse bajo la experiencia y tiempo disponible. No vamos a empezar un proyecto demasiado difícil.
- El tema se valora si presenta cualidades innovadoras. Debe ser algo más que un trabajo de una semana. Hay que plantearse un tema que requiera tiempo para la búsqueda de información y el desarrollo.

Por ello un sistema de autorización y gestión de matrículas, acaba siendo la mejor opción que cubre la mayoría de puntos. Presenta una cierta complejidad, especialmente al usar una base de datos y una página de gestión además del sistema de detección. El producto ubre una necesidad real. Contiene un sistema de detección con reconocimiento computacional, una tecnología en auge gracias los avances en redes neuronales y procesamiento de modelos con inteligencia artificial.

4.2 Análisis del mercado

Para el análisis del mercado es importante estudiar productos similares. Para ello he comparado tanto productos comerciales, como de código abierto. Comparar productos similares con el diseño propio, permite considerar la relevancia del plan de desarrollo y realizar cambios antes de empezar con el desarrollo.

Los sistemas comerciales, aunque generalmente usen algoritmos precisos de detección, vienen licencias propietarias con costes relativamente elevados. Por otra parte, los programas de código abierto no proporcional gran nivel de precisión, pero es de esperar por software gratuito. Por la principal cualidad que se pretende implementar en SisMat es ofrecer precisión de detección, aun usando componentes gratuitos. SisMat decide usar detección de modelos avanzados e integrarlos para tener un producto que proporcione mayor precisión aún siendo gratuito. Comercializar el producto es viable, aunque licencias de Yolo hay únicamente para versiones actualizadas, que aunque

suponga un coste adicional, ofrecen muchas ventajas a las versiones anteriores.

Otra cualidad que no se ve mucho es la personalización. La mayoría de productos comerciales ofrecen todas las funcionalidades en su panel de control, sin posibilidades de expandir o customizar funcionalidades propias. En cambio, los programas de código abierto no proporcionan muchas funcionalidades en sus propios sistemas, aunque sea posible su desarrollo. SisMat se desarrollará con capacidades de desarrollos adicionales además de ofrecer un panel completo con la capacidad de agregar y eliminar datos y/o funcionalidades.

Los productos de pago también ofrecen soporte a tiempo real para cualquier tipo de problemas o dudas que tenga el cliente con su producto, mientras que los sistemas de código abierto, se enfoca en el apoyo centrado a la comunidad que rodea al producto. SisMat tiene planteado tener apoyo de la comunidad a través de foros de Github.

4.3 Requisitos

Para la obtención de un producto final que tenga unas funcionalidades concretas, se han de definir los requisitos funcionales (lo que ha de realizar) y no funcionales (rendimiento, experiencia de usuario...). Definir funcionalidades permite configurar el sistema bajo ciertas especificaciones, ayudando a elegir cada uno de sus componentes que mejor se adapten a nuestras necesidades, desde el lenguaje de programación hasta el hardware.

4.4.1 Requisitos funcionales

Los requisitos funcionales expresan la capacidad del sistema para realizar ciertas acciones. Todo sistema tiene sus limitaciones, y poder identificar que tareas es capaz de realizar de manera consistente, además de la definición de la estructura individual y conjunta de todos los componentes que componen esta.

En nuestro caso, partimos de los siguientes requisitos:

- El programa debe obtener información a tiempo real de los vehículos a través de cámaras digitales.
- El sistema debe detectar la presencia de vehículos en la imagen.
- El sistema debe localizar y extraer la matrícula de los vehículos detectados.
- El sistema debe indicar si se ha permitido o denegado la entrada al vehículo.

- La información de las matrículas, usuarios y administradores se obtienen dinámicamente de una base de datos.
- La información de la base de datos se debe poder cambiar de una página web de formulario con interfaz de usuario intuitiva.
- La página de formulario debe tener páginas individuales para la creación, modificación y listado de cada tipo de elementos.

4.4.2 Requisitos no funcionales

En cambio, los requisitos no funcionales hacen enfoque a las características que no afectan directamente a la funcionalidad del sistema. Poder definir una base de un rendimiento y calidad, permiten la creación de un producto pulido y enfocado al consumidor. Muchos de estos requisitos van enfocados a la velocidad de carga, compatibilidad e interfaz de usuario con la finalidad de optimizar recursos, reducir errores y proporcionar una mejor experiencia de usuario (UX).

En nuestro caso, partimos de los siguientes requisitos:

- El sistema debe procesar imágenes con una latencia inferior a 2 segundos por imagen.
- El sistema debe ser compatible con sistemas operativos Windows y Linux.
- La página de formulario debe cargar en menos de 4 segundos.
- La interfaz de la página de formulario debe ser sencilla y apta para usuarios sin conocimientos técnicos.

4.5 Definición de MVP

Con los requisitos definidos, ahora podemos definir el Producto Mínimo Viable (MVP en inglés). El producto presenta las funcionalidades reducidas para la demostración de las capacidades de un producto, añadiendo funcionalidades en un futuro. El MVP de SisMat debe completar las siguientes pautas:

- Detectar un **vehículo y su matrícula** a través de la cámara.
- Reconocer los **caracteres de la matrícula**.
- Abrir **dos ventanas** mostrando la **entrada de vídeo** de la cámara de entrada y salida.
- Mostrar al usuario el **contador de vehículos** dentro del recinto.
- Gestionar toda la **información del sistema** de una página web.
- Permitir cambiar las **variables de sistema** a través de una página web.

- Tener un sistema de **sesión** en la página web.

4.6 Diagrama de flujo

Para poder desarrollar correctamente el sistema, se ha de entender cada una de las etapas del comienzo hasta el final del programa. Para ello necesitaremos un diagrama de flujo para ver cada proceso y estado que el programa puede estar.

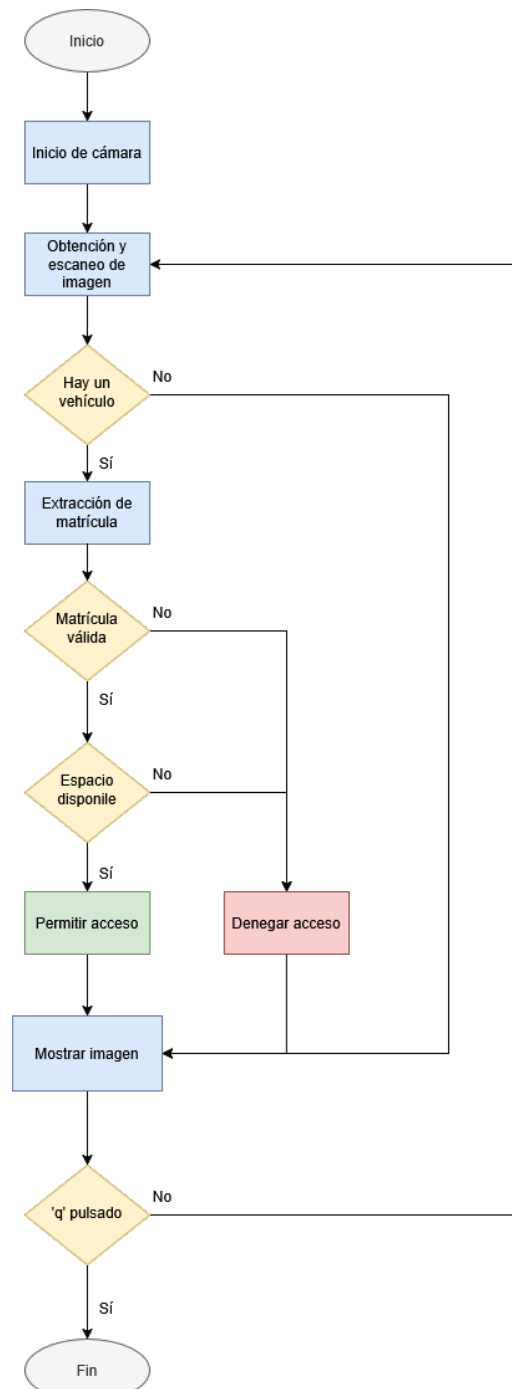


Figura 1: Diagrama de flujo de SisMat

4.7 Programación

La programación supone de las fases más relevantes de la creación del sistema. Es donde se desarrolla toda la lógica según los requisitos indicados. Hay veces donde se ha de rediseñar ciertas funcionalidades a lo largo del desarrollo al haber posibles problemas con ciertos desarrollos a la hora de implementarlos.

Entorno

El programa usa lenguajes de programación universales, por lo tanto, se pueden ejecutar en varias plataformas. En mi caso he usado Windows 10, al ser el más familiar de las disponibles.

Los entornos de desarrollo componen de IntelliJ Idea y PyCharm por su facilidad de uso y herramientas avanzadas de edición, además de ser entornos a los cuales estoy familiarizado.

Programa de detección

El programa de detección se encarga de la detección de vehículos. En un principio se pretende implementar únicamente la entrada, al ser lo más crucial para el resto del sistema. La funcionalidad de salida se implementa al final.

El uso de Python era fundamental para el uso de modelos y sistemas de detección al tener la mayoría de desarrollos de librerías como OpenCV y Ultralytics Yolo.

Base de Datos

La base de datos se compone de toda la información los permisos de matrículas, usuario y administradores además de las funcionalidades requeridas para otorgar permisos y relacionar los datos entre sí. Usar tablas relacionales con SQL, permiten facilidad de desarrollo y rendimiento. Se creó primero la tabla 'usuario' primero para poder relacionarlo a la tabla 'matricula', la cual se generó posteriormente. Por último, tras una revisión, se añadió la tabla 'administrador' y su relación en la tabla 'usuario'.

Usé MySQL junto a PHPMysqlAdmin por la facilidad de administración directa de la propia BBDD y sus tablas.

Página web

La página web permite la administración de los datos guardados en la BBDD de una manera intuitiva al usuario final. Incluir páginas para la inspección de elementos, agregación, edición y borrado (también conocido como CRUD) de elementos.

El desarrollo a través de Springboot permite implementar funcionalidades y objetos con JPA a través de la estructura e información definida de la Base de datos.

Interconexión

Al tener las bases generadas, solo falta hacer que funcione todos los componentes entre sí. Se implementa la parte de comprobación de matrículas en el programa de detección para obtener información a tiempo real del estado de las matrículas.

Otro último desarrollo es la conexión de la página web con la BBDD. Asegurar que los elementos tengan los nombre correctos y que los objetos de JPA se han generado correctamente. Puede haber páginas que tengan problemas de carga por errores como estos.

4.8 Pruebas

Realizar pruebas es crucial para verificar el correcto funcionamiento de un producto procurando que el sistema funcione de manera esperada, sin errores inesperados. Someter el sistema a pruebas reduce la probabilidad de encontrar futuros problemas en el desarrollo que acaben siendo mucho más difíciles de corregir, además de garantizar una experiencia reducida de errores al usuario final.

Por cada componente se hacen las pruebas unitarias acordes con las funcionalidades que se han de realizar. Una vez el sistema esté completo, se realizan las pruebas de integración, garantizando funcionalidad completa y conjunta de todos los componentes del sistema. Por último, se realizan las pruebas funcionales para comprobar si se cumplen los requisitos establecidos para el producto.

La idea no es crear pruebas que tengan probabilidades altas de funcionar con resultados esperados, sino pruebas forzadas que comprueben la robustez del sistema.

4.9 Análisis de resultados

Para comprender las características del producto final, se ha de analizar los resultados obtenidos.

El objetivo principal es comprobar la eficacia del reconocimiento automático de matrículas en distintas condiciones y escenarios de uso. Para ello, se han definido una serie de pruebas controladas que permiten observar el comportamiento del sistema ante diversas variables, como la iluminación, el ángulo de la cámara, la calidad de la imagen y la tipología de las matrículas.

El análisis contempla tanto el funcionamiento de cada componente (detección, reconocimiento, base de datos e interfaz) como su integración en el sistema completo. Se han procesado imágenes estáticas y secuencias de vídeo con diferentes configuraciones para analizar el rendimiento en tiempo real y la estabilidad del sistema. Además, se ha evaluado la capacidad de la herramienta para registrar e interpretar correctamente los datos reconocidos, así como su interacción con el usuario a través del formulario de edición.

Durante este proceso se recogen métricas como el tiempo de respuesta, la cantidad de aciertos y errores en la lectura, y el comportamiento ante casos límite. También se documentan posibles incidencias o limitaciones observadas para establecer un punto de partida objetivo en el que basar las conclusiones y futuras mejoras.

El enfoque de este análisis es tanto funcional como técnico, con la intención de valorar el sistema de forma global y detallada, considerando no solo el reconocimiento de matrículas, sino su viabilidad como solución práctica de control vehicular automatizado.

4.10 Documentación

Documentar los elementos de la programación ayuda a entender el funcionamiento del sistema fuera del usuario inicial. Esto permite a los desarrolladores futuros entender tanto los elementos generales como específicos del sistema.

Para ello he incluido documentación incrustada en el código en el programa de detección de Python en forma de comentarios.

En el programa de Springboot he incluido un javadoc con documentación acerca de los elementos del sistema.

Adicionalmente, este documento también sirve de guía para la estructura del proyecto.

5. Desarrollo

5.1 Hardware

En cuanto a recursos físicos se requiere dos componentes: Un entorno de ejecución y un par de cámaras. Dependiendo de si queremos que el sistema únicamente tenga una cámara y para la salida haya un botón u otro sistema de salida. También, si se desea vigilar otros accesos, se pueden incluir más cámaras y la configuración de software adecuada.

En el entorno se ejecuta el programa de detección y, opcionalmente la base de datos y la página web, aunque estas también se pueden alocar en servicios en la nube o máquinas dedicadas. Por simplicidad y conveniencia, todos los programas se ejecutan en el mismo entorno.

Usar cámaras permite al ordenador tener visión y poder obtener información sobre los vehículos que se aproximen a nuestra puerta. Para ello usaré dos cámaras USB: uno para la entrada y otro para la salida.

El proyecto no dispone de ningún tipo de actuador físico para la puerta de acceso, pero de momento simularé la funcionalidad con una señal por software.

5.2 Programa de detección

La base

Principalmente, se usa Python para los scripts, por su gran variedad y calidad de librerías de visión computacional, además de ser la más usada para esta clase de desarrollos. Las librerías que se pretenden son desarrollados casi exclusivamente para Python, demostrando sus capacidades de desarrollos de librerías que se basan en nuevas tecnologías.

Visión computacional

La base de la entrada de video lo compone las librerías de OpenCV, un conjunto de funcionalidades que sirven de base para el desarrollo de aplicaciones con imágenes o videos. Se usa OpenCV para capturar la entrada de las dos cámaras y mostrar la salida de cada una en una transmisión propia, mostrando el contador de vehículos se encuentran en el establecimiento.

Detección

Para la detección de vehículos uso las librerías de YOLOv8 de Ultralytics, un algoritmo de detección de modelos a tiempo real de licencia AGPL-3.0, la cual es reconocida por su precisión y rendimiento. La versión 8 de YOLO ofrece muy buen rendimiento en comparación con otras versiones aún siendo gratuita para uso no comercial. YOLO es capaz de reconocer una gran variedad de modelos.

Desde animales a objetos, nos sirve para reconocer diferentes vehículos como coches, camiones o motos. Por la configuración disponible, detectar matrículas de motos se hace imposible, ya que no disponen de matrícula por la parte delantera. Para esto se podría poner una cámara que detecte la matrícula por la parte de atrás, lo cual supondría un rediseño del sistema. Una vez finalizada el proyecto, miraré otras alternativas propias que podría implementar, pero de momento usaré este, ya que el enfoque no es tanto en crear un sistema de reconocimiento sino un sistema completo que permita implementar la tecnología de detección para este uso específico.

Una cualidad no aplicada es la aceleración con GPU, siendo este procesada por la CPU. Las tarjetas gráficas pueden conseguir un rendimiento superior para detección, especialmente con una tarjeta gráfica de Nvidia ya que las librerías CUDA permiten el procesamiento rápido de este tipo de cargas.

Lector de caracteres

Para la lectura de los caracteres de la matrícula uso EasyOCR, un lector de texto de código abierto. Esta librería es especialmente buena para matrículas borrosas o que presenten en un ángulo, perfecto para la detección de vehículos. EasyOCR procesa la imagen capturada por Yolo, extrayendo los caracteres.

5.3 Base de datos

La base de datos se basa en un modelo de matrículas, cada matrícula puede pertenecer a un usuario y un usuario puede tener varias matrículas. Como algunos usuarios puede que lleguen con diferentes vehículos, por lo tanto, cada usuario puede acceder con cualquiera de sus vehículos registrados dentro de las fechas registradas. Tanto la instancia de MySQL, como phpMyAdmin se encuentran en contenedores de docker para gestionarlos fácilmente.

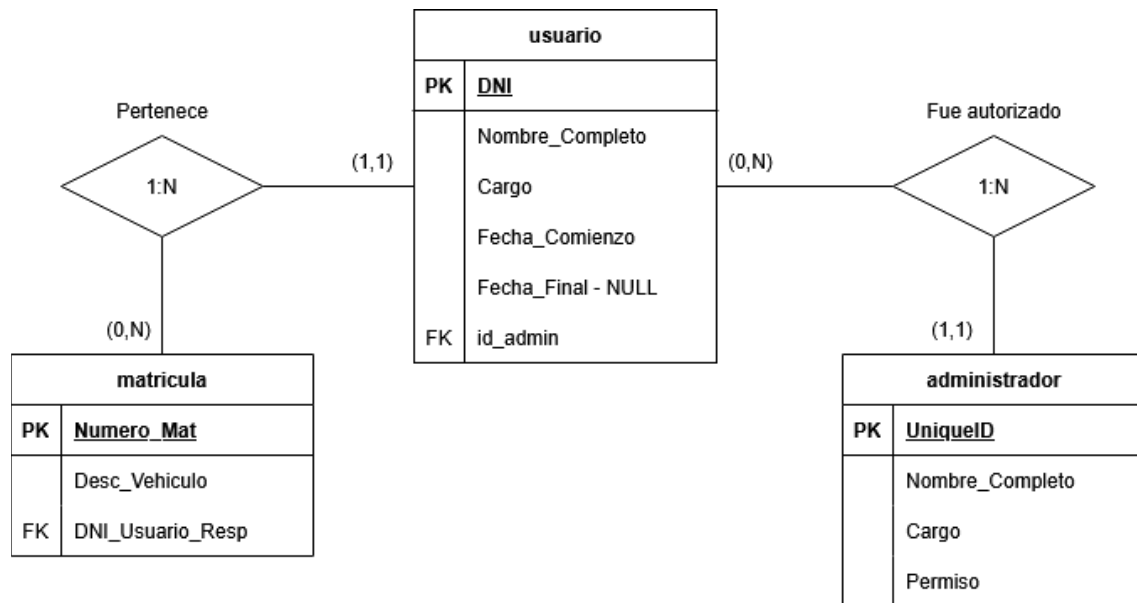


Figura 2: Esquema entidad-relación de la base de datos del sistema

5.3.1 Tabla 'usuario'

Los usuarios tiene un DNI como identificador. Cada usuario tiene su fecha de comienzo de su periodo de permiso de acceso. Se puede indicar una fecha final, aunque no es obligatorio. De cada usuario se incluye información personal con campos indicando el nombre completo y su cargo.

A cada usuario se le otorga permiso por un administrador. Este se manifiesta en un campo de clave foránea con el ID del administrador que ha otorgado este permiso.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/> 1	DNI	varchar(9)	utf8mb4_0900_ai_ci		No	None		
<input type="checkbox"/> 2	Nombre_Completo	varchar(200)	utf8mb4_0900_ai_ci		No	None		
<input type="checkbox"/> 3	Cargo	varchar(50)	utf8mb4_0900_ai_ci		No	None		
<input type="checkbox"/> 4	Fecha_Comienzo	date			No	None		
<input type="checkbox"/> 5	Fecha_Final	date			Yes	NULL		
<input type="checkbox"/> 6	id_admin	int			No	None		

Figura 3: Tabla usuario

5.3.2 Tabla 'matricula'

Teniendo en cuenta que las matrículas europeas usan hasta 8 caracteres, usaremos eso en nuestro campo. No incluimos guiones, punto o comas como caracteres.



Figura 4: Formato de matrículas en la Unión Europea

La estructura de la matrícula es muy simple. Contiene el número de matrícula como identificador y una descripción del vehículo registrado. Se indica el DNI como clave foránea del usuario al que pertenece esa matrícula. No se incluye tilde en los nombres de elementos informáticos para evitar errores de lectura.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1 Numero_Mat 🔑	varchar(8)	utf8mb4_0900_ai_ci		No	None		
<input type="checkbox"/>	2 DNI_Usuario_Resp 🔑	varchar(9)	utf8mb4_0900_ai_ci		No	None		
<input type="checkbox"/>	3 Desc_Vehiculo	varchar(200)	utf8mb4_0900_ai_ci		No	None		

Figura 5: Tabla matricula

5.3.3 Tabla 'administrador'

En cuanto a los administradores, se incluye un ID único autogenerated como identificador. Cada administrador posee de booleano permiso, indicando por si puede otorgar permisos o no. Este se usa en la página para que solo los administradores con permiso puedan autorizar a usuarios.

Adicionalmente, como información personal, existe el nombre completo del usuario y el cargo que ocupa en el local.


#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/>	1 ID 	int			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2 Nombre_Completo	varchar(200)	utf8mb4_0900_ai_ci		No	None		
<input type="checkbox"/>	3 Cargo	varchar(50)	utf8mb4_0900_ai_ci		No	None		
<input type="checkbox"/>	4 Permiso	tinyint(1)			No	None		

Figura 6: Tabla administrador

5.4 Panel de control

El panel de control es el elemento clave para la modificación de registros de las matrículas, sus usuarios y los administradores. Para ello se ha creado un portal web segmentada en diferentes páginas con una interfaz simple y fácil de leer. Este diseño viene con la intención de proporcionar una mejor experiencia de gestión de la información para personas administrativas de un centro u empresa que carezcan de experiencia en informática. Además, una interfaz bien estructurada proporciona una mejor experiencia de usuario y reduce confusión y la creación de errores.

La página se compone de cuatro secciones: **inicio de sesión**, **matrículas**, **usuarios** y **panel de administración**.

Inicio de sesión

Para la página se requiere iniciar sesión para acceder al panel de control. Esto evita a que cualquiera que conecte pueda modificar información. Hay usuarios de sistema corrientes y administradores.

Componentes comunes

Cada página forma de una serie de elementos comunes. Entre ellos hay:

- Listado de elementos. Se muestran los elementos de esa sección, guardadas en la tabla de la BBDD correspondiente.
- Formulario de agregación.
- Formulario de edición
- Borrado de elementos
- Filtrados de elementos por validez en aquel momento.

Matrículas

Se muestra la lista de conjunto de matrículas de la base de datos. Por cada registro información disponible de cada matrícula en línea además de mostrar el propietario del vehículo y la persona que la ha autorizado. Se puede acceder al los detalles del usuario directamente pulsando a su registro en su campo en el registro. En cada registro hay botones para modificar y eliminar ese elemento.

En esta página hay un botón para agregar una matrícula, llevando al formulario correspondiente.

Usuarios

Se muestra el conjunto de usuarios registrados en el sistema. Por cada registro de cada usuario se muestra su información y la persona que ha autorizado a ese usuario. Se puede acceder al los detalles del usuario directamente pulsando en el campo de DNI en línea.

Adicionalmente, en el formulario de modificación, se muestran las matrículas que pertenecen al usuario seleccionado.

Es esta página también existe un botón para acceder al formulario de creación.

Panel de administración

Si se inicia sesión como administrador, habrá disponible una sección dedicada para el panel de admin. Aquí se ve el conjunto de variables de sistema que se pueden configurar según las necesidades. Debajo, se ven todos los administradores que hay según la base de datos y su información. Aquí se pueden agregar, configurar y eliminar los administradores de sistema fuera del acceso del resto de usuarios. Esto evita el acceso no autorizado y garantiza la seguridad del sistema.

Número de matrícula	Descripción del vehículo	Propietario	Autorizado Por		
1122DEF	Peugeot 208, gris	Sergio Navarro Pérez	Carlos Torres Molina		
1234ABC	SEAT León 2020, blanco	Miguel Jimenez	Carlos Torres Molina		
2468STU	Ford Transit, furgoneta blanca	Laura Martínez Sánchez	Laura Sánchez Díaz		
3344GHI	Renault Kangoo, blanco - vehiculo comercial	Pedro Luis Ortega Ramos	Aitana Sánchez Gómez		
5566JKL	Citroën C3 Aircross, rojo	David García Cordero	Aitana Sánchez Gómez		
9876YZA	Honda PCX 125, blanco	David García Cordero	Aitana Sánchez Gómez		
9900PQR	Hyundai i30, azul	Lucía Morales Herrera	Carlos Torres Molina		
EL896CJ	Skoda Elroq, Blanco	David García Cordero	Aitana Sánchez Gómez		
M666YOB	Turismo rojo americano de 5 puertas	Miguel Jimenez	Carlos Torres Molina		
SL593LM	Tesla Model S, Blanco	Sergio Navarro Pérez	Carlos Torres Molina		

Figura 7: Sección de matrículas del panel web

6. Pruebas

Una de las dificultades del sistema es el uso de vehículos para la detección. Yolo es capaz de reconocer mejor objetos claros cuando se encuentran dentro de un contexto que lo abarque. En mi caso, al no poseer de un vehículo para pruebas, he decidido usar imágenes de vehículos para su simulación. Probando con vehículos reales, el sistema es bastante más preciso gracias a la existencia de contexto que acompañe al vehículo.

Para verificar el correcto funcionamiento del sistema realicé las siguientes pruebas: **Unitarias, de integración, funcionales, rendimiento y UX.**

Pruebas unitarias

Es importante comprobar el correcto funcionamiento de cada componente no solo para verificar que funciona sin errores, sino para evitar problemas a la hora de realizar desarrollos próximos y facilitando la resolución de bugs. Según cada componente se realizaron las siguientes pruebas:

(Resultado a completar → Prueba a realizar)

- Sistema de detección:
 - Se detectan correctamente las cámaras
 - Se conecta primero una cámara y después ambas y ver si salen las entradas por pantalla.
 - Correcto funcionamiento de detección de vehículos.
 - Mostrar imágenes de coches a las entradas para ver si genera la caja enmarcada en el sistema de detección.
 - Correcto funcionamiento del lector de matrículas.
 - Mostrar por consola las matrículas que lee el OCR.
 - Comprobar si el contador de vehículos funciona
 - Observar si el contador de vehículos en la cámara de entrada se actualiza al entrar o salir un vehículo.
- Base de datos:
 - Asegurar la estructura correcta de las tablas.
 - Insertar registros y observar si salen errores.
 - Funcionalidades en cascada de las claves foráneas funcionan correctamente
 - Borrar registros relacionados por clave foránea y observando los resultados de las claves correspondientes en tablas relacionadas.

- Página web:
 - Garantizar la estructura uniforme de elementos en todas cada página
→ Inspeccionar visualmente cada página y ver que la interfaz se mantiene igual en todas las páginas y que no falten elementos.
 - La interfaz se mantiene utilizable en todo momento
→ Comprobar el funcionamiento del sistema con ventanas de diferentes formas y tamaños
 - Todos los enlaces de la página tienen referencias correctas → Pulsar cada enlace para comprobar el correcto funcionamiento y que no se encuentran enlaces rotos.

Pruebas de integración

Revisar únicamente los componentes individuales no garantiza el funcionamiento completo del sistema. Por ello se realizan pruebas finales del producto final para asegurar que funciona de manera esperada.

(Resultado a completar → Prueba a realizar)

- El acceso a la BBDD funciona correctamente desde el programa de detección.
→ Agregando un registro y verificando si al detectar la matrícula por la entrada, otorga permiso.
- Los registros de la base de datos se mantienen actualizados a través de todos los componentes del sistema.
→ Agregar un objeto, por ejemplo una matrícula nueva, y observar como el elemento aparece en la página web al momento y se otorgue correctamente acceso en el sistema de detección.
- Los elementos mostrados en la página web corresponden 1:1 a los registros en la base de datos.
→ Comprobar todos los datos mostrados en phpMyAdmin con los mostrados en la página web

Pruebas funcionales

Para comprobar que nuestro sistema esté según lo diseñado, se realizan pruebas en comparación a los requisitos establecidos. La idea es saber si el producto completa los test de calidad definidos.

(Requisito → Resultado del benchmarking)

Requisitos funcionales

- El programa debe obtener información a tiempo real de los vehículos a través de cámaras digitales.
→ Realizado con la cámara de entrada.
- El sistema debe detectar la presencia de vehículos en la imagen.
→ El sistema detecta vehículos con precisión.
- El sistema debe localizar y extraer la matrícula de los vehículos detectados.
→ El sistema extrae y muestra la matrícula con precisión satisfactoria
- El sistema debe indicar si se ha permitido o denegado la entrada al vehículo.
→ Los mensajes de cada lectura viene acompañada de su mensaje correspondiente.
- La información de las matrículas, usuarios y administradores se obtienen dinámicamente de una base de datos.
→ Estructura correcta de base de datos y validez de datos en la detección y en el panel web.
- La información de la base de datos se debe poder cambiar de una página web de formulario con interfaz de usuario intuitiva.
→ La página web consiste de una interfaz simple y clara. Los datos se modifican sin errores.
- La página de formulario debe tener páginas individuales para la creación, modificación y listado de cada tipo de elementos.
→ La página dispone de estos elementos.

Requisitos no funcionales

- El sistema de detección debe procesar imágenes con una latencia inferior a 2 segundos por imagen.
→ La página carga de manera constante a aproximadamente 30 fotogramas por segundo (1 fotograma cada 0.033 segundos)
- El sistema debe ser compatible con sistemas operativos Windows y Linux.
→ Todos los componentes y librerías tienen soporte multiplataforma.
- La página de formulario debe cargar en menos de 4 segundos.

→ La página al ser ligera carga en menos de 1 segundo de manera consistente.

- La interfaz de la página de formulario debe ser sencilla y apta para usuarios sin conocimientos técnicos.

→ La página web consiste de una interfaz entendida y aprobada por un familiar sin experiencia informática.

Pruebas de rendimiento

Con el objetivo de evaluar la eficiencia y precisión del sistema desarrollado, se llevaron a cabo pruebas de rendimiento bajo diferentes condiciones. Estas pruebas se centraron en tres componentes clave: detección de matrículas (YOLOv8), reconocimiento de caracteres (EasyOCR), y velocidad de procesamiento global del sistema. Además, se prueba la velocidad de carga de la página.

Se utilizó un conjunto de datos compuesto por imágenes y secuencias de vídeo capturadas en diversos entornos, incluyendo condiciones de buena iluminación, baja visibilidad, matrículas inclinadas o parcialmente ocultas. El sistema fue ejecutado en un equipo con las siguientes características:

- **Procesador:** Intel Core i7
- **GPU:** NVIDIA RTX 3080
- **RAM:** 16 GB
- **Sistema operativo:** Windows 10

Las métricas analizadas fueron:

- **Precisión de detección:** Porcentaje de matrículas correctamente localizadas.
- **Precisión OCR:** Porcentaje de caracteres reconocidos correctamente.
- **Exactitud completa:** Número de matrícula completamente reconocidas (sin errores).
- **Tiempo medio de procesamiento por imagen o fotograma.**

Escenario	Detección YOLOv8	OCR (EasyOCR)	Tiempo por imagen
Luz natural	98%	93%	0,24 s
Iluminación nocturna	85%	78%	0,27 s
Ángulo inclinado	90%	83%	0,25 s
Matrículas sucias/borrosas	72%	40%	0,29 s

Tabla 3: Análisis de rendimiento de los componentes del sistema

En cuanto a la página web, el sistema hace uso de estilos de bootstrap y se basa en los controladores de springboot para el acceso de elementos de la base de datos y sus páginas. Esto hace que la página cargue en menos de 1 segundo con consistencia, siendo la navegación rápida y satisfactoria.

7. Análisis de resultados

Este apartado presenta una evaluación estructurada del sistema desarrollado, considerando tanto sus fortalezas como sus limitaciones. El análisis se divide en cuatro bloques: aspectos positivos, problemas detectados, comparativa con otras herramientas y una interpretación general de su aplicabilidad.

7.1 Puntos positivos

Durante las pruebas realizadas, se identificaron varios aspectos favorables del sistema. En primer lugar, destaca su alta precisión en condiciones óptimas, especialmente cuando las imágenes presentan buena iluminación y la matrícula se encuentra bien orientada. Además, el tiempo de procesamiento observado resulta adecuado para aplicaciones casi en tiempo real, lo que lo hace viable para entornos como aparcamientos o accesos con flujo continuo de vehículos.

Otro punto fuerte es el diseño modular de la arquitectura, lo que permite reemplazar o actualizar componentes (por ejemplo, el motor OCR o el modelo de detección) de forma sencilla, sin tener que rehacer todo el sistema.

7.2 Problemas y limitaciones

No obstante, también se identificaron algunas limitaciones. En situaciones con bajo contraste, imágenes desenfocadas o matrículas sucias, el motor OCR comete errores de reconocimiento. Asimismo, las matrículas parcialmente visibles o captadas desde ángulos extremos presentan dificultades tanto en la detección como en la lectura completa de los caracteres.

A nivel lingüístico, EasyOCR ofrece su mejor rendimiento con caracteres latinos estándar, por lo que el sistema puede requerir ajustes para trabajar con alfabetos no latinos u otros formatos de matrícula.

7.3 Comparativa con otros sistemas

Se realizó una comparativa con otras soluciones similares. Frente a motores como Tesseract o herramientas como OpenALPR, el sistema implementado con YOLOv8 y EasyOCR muestra una mayor adaptabilidad a imágenes diversas y mejor rendimiento OCR en imágenes con ruido o baja calidad. Además, la detección con YOLOv8 resulta más rápida y precisa que con enfoques anteriores como las Cascadas de Haar o versiones previas de YOLO.

Aunque el sistema no está tan optimizado para vídeo en tiempo real como algunas soluciones comerciales como PlateSmart, se compensa con una

mayor flexibilidad y menor coste, lo que lo hace accesible para desarrollos académicos o implementaciones a pequeña escala.

7.4 Interpretación de los resultados

A partir de los resultados obtenidos, se concluye que el sistema es especialmente adecuado para entornos semi-controlados, como aparcamientos, accesos privados o zonas urbanas con cámaras fijas. En estos contextos, el sistema puede operar de forma efectiva y fiable. Sin embargo, presenta debilidades en escenarios más exigentes, como situaciones con cámaras móviles, desenfoques frecuentes o matrículas en posiciones poco visibles, donde sería necesaria una optimización adicional o el uso de hardware especializado.

8. Conclusión

El desarrollo de este sistema de reconocimiento automático de matrículas (ANPR) ha demostrado que es posible construir una solución funcional, adaptable y económica mediante el uso de tecnologías de código abierto y técnicas actuales de visión por computador. A través de la integración de herramientas como OpenCV, YOLOv8 y EasyOCR, se ha conseguido un sistema capaz de detectar y reconocer matrículas en tiempo real con una precisión aceptable y un rendimiento adecuado para aplicaciones prácticas en contextos urbanos y privados.

Durante el proceso se identificaron y resolvieron diversos retos técnicos, como la variabilidad en las condiciones de iluminación, el ángulo de captura o la calidad de imagen, lo que permitió validar la robustez del sistema en escenarios realistas. Además, la incorporación de una base de datos y una interfaz de usuario accesible ha permitido ampliar el uso del sistema a usuarios no técnicos, facilitando su adopción y mantenimiento.

Este proyecto no solo aporta una solución técnica viable, sino que también pone de manifiesto el potencial del software libre para reducir barreras económicas en la implementación de tecnologías inteligentes. Asimismo, ofrece una base sólida sobre la que construir futuras mejoras, como el reconocimiento de matrículas internacionales, la integración total con la plataforma web o la asignación de plazas.

En definitiva, el trabajo realizado representa una contribución significativa al ámbito del control vehicular automatizado, alineándose con las necesidades actuales de movilidad urbana sostenible, seguridad y eficiencia operativa.

Habilidades implementadas:

- Comprensión del valor de mercado y el auge de sistemas ANPR para uso del gobierno y empresas privadas.
- Uso de cámaras para la detección de modelos.
- Ampliación de conocimientos de librerías de detección de Python. Siendo la primera vez que use EasyOCR o YOLO. También he conseguido crear esta vez un sistema que utilice varias entradas con OpenCV, en vez de la previa que solo usé una.
- Refuerzo de habilidades de programación Java y uso de Springboot para el desarrollo del servicio web.
- Mejora de habilidades de diseño web con HTML y uso de estilos con Bootstrap.

- Refuerzo de capacidades de diseño, creación y gestión de BBDD.
- Puesta en prueba de habilidades de creación y desarrollo de proyectos en el campo de informática y tecnología.

9. Mejoras de cara a paso a producción

El proyecto se basa en un producto mínimo viable o MVP, la cual incluye las funcionalidades fundamentales para la valoración del concepto y dar base a ideas futuras. Por ello, según se vea necesario, el producto podrá ser ampliado con diferentes tipos de mejoras, agregando nuevas funcionalidades e incrementando la versatilidad y el valor del producto. Muchos de estos desarrollos no se han implementado por

A continuación explico algunas de las ideas que me surgieron:

Indicadores de acceso

Con el sistema actual, los conductores de los vehículos que usan el acceso, no tienen manera de saber el estado de autorización. Si la puerta no se abre, no se sabe si sucede por:

- No haya plazas libres.
- Autorización no otorgada.
- Autorización otorgada pero no válida.

O si se abre:

- Si ha de dar paso primero a un vehículo que está saliendo.
- Si el permiso está a punto de caducarse.

Para resolver este problema propongo las siguientes soluciones:

- Un sistema de luces que indique si un vehículo puede acceder o no al local. Para resolver cada problema se usaría:
 - Luz roja fija indica que no se ha detectado un vehículo. O si está abriendo la puerta, que hay un vehículo saliendo y el proceso de autorización no ha comenzado todavía.
 - Luz roja parpadeante indica que el vehículo detectado no tiene autorización o tiene autorización caducada.
 - Luz verde fija indica que se otorga acceso.
 - Luz verde parpadeante indica que se otorga acceso pero que el acceso caduca el próximo día.
- Una pantalla que indique verbalmente el estado del acceso y/o posibilidad de contactar con un responsable en caso de problema.

Este sistema ayuda de manera visual reducir confusión y optimizar el flujo de vehículos e informar a usuarios del estado de su autorización.

Seguimiento del estado de cada vehículo

Como responsables de la zona, nos podría interesar registrar si un vehículo se encuentra en la zona o no. Con esto podemos saber si falta algún vehículo en particular o simplemente vigilar el flujo de vehículos. Este sistema podría tener interés en circunstancias escolares o policiales ya que el seguimiento de patrones de ciertos vehículos puede desvelar comportamientos inusuales.

Por ello incluir en el sistema la detección de matrículas no solo en la entrada sino también en la salida para monitorizar con estados de si se encuentra x vehículo en la zona vigilada con la fecha de último registro para localizar cambios.

Detección y asignación de plazas

El seguimiento también se puede complementar con la gestión de plazas individuales. La idea consiste en tener sensores en cada plaza de aparcamiento para detectar si un vehículo se encuentra en esa plaza. Con esto podemos saber la capacidad a tiempo real de plazas sin tener que usar el contador del sistema de detección.

También se podría reservar ciertas plazas para ciertos usuarios como, por ejemplo, altos cargos o personal de administración. Así aseguramos que ciertas plazas no sean consideradas para el contador de plazas disponibles, ya que tienen ya su propia plaza dedicada. Con esto en mente, se podrán reservar manualmente ciertas plazas para situaciones extraordinarias, por ejemplo, están ocupado 2 plazas para guardar material de mantenimiento.

Usando este modelo conseguimos organizar mejor la asignación de plaza y evitamos problemas de congestionamiento de vehículos.

Aceleración con GPU

Una de las cualidades del MVP es ser modular. Todos los componentes son modulares y multiplataforma, aunque puede que rindan mejor en ciertas configuraciones. En el caso de la librería YOLO, se podría implementar aceleración de GPU, si esta está disponible en la máquina final. Cualquier hardware de aceleración de gráficos suficientemente apta para el trabajo podría sobrepasar el rendimiento de una CPU. Esto se debe a la arquitectura de los procesadores de gráficos, siendo muy eficientes para procesos de computación compleja y procesamiento de modelos grandes gracias a su abundancia de núcleos de computación adaptadas para el trabajo. En el caso de tarjetas gráficas compatibles de Nvidia, se pueden usar librerías CUDA. Estas librerías, con las cuales YOLO es compatible, optimizan el procesamiento de imágenes

complejas a gran velocidad, gracias a su optimización con el hardware dedicado de Nvidia.

El problema es que este hardware no es accesible para todos los sistemas y requiere configuración adicional.

Panel web *Todo en Uno*

Una de las ideas propuestas por el tutor es incluir el sistema totalmente en el panel web. Esto sustituiría el sistema actual de ventanas dedicadas en el script de Python y haría el uso más intuitivo y menos localizado.

Esta implementación supondría tener un host que distribuya la entrada de vídeo a la página web por poderse retransmitir. Elementos como los contenedores de la base de datos se mantendrían también en una red común al requerir del acceso a los datos del sistema para su verificación. Además, la inclusión del feed de vídeo en la página web, requiere una cantidad sustancial de desarrollo que afectaría negativamente al rendimiento de la página web.

10. Bibliografía

Informe de MarketsandMarkets (ANPR)

MarketsandMarkets. (2023). *Automatic Number Plate Recognition System Market by Type, Component, Application, and Region - Global Forecast to 2027*. <https://www.marketsandmarkets.com/Market-Reports/anpr-system-market-140920103.html>

Uso de cámaras CCTV en reino unido

Hopkins, N. (2014, 23 de enero). *CCTV cameras on Britain's roads capture 26 million images every day*. The Guardian. <https://www.theguardian.com/uk-news/2014/jan/23/cctv-cameras-uk-roads-numberplate-recognition>

Página oficial de parking network

Parking Network. (2024, 6 de diciembre). *Resumen de noticias de la semana 49. Parking Industry Blog*. <https://www.parking.net/parking-industry-blog/parking-network/news-summary-week-of-49>

Página oficial de Ultralytics / YOLO

Ultralytics. (s.f.). *YOLO (You Only Look Once)*. <https://ultralytics.com/yolo>

Fuentes de librerías usadas

Ultralytics. (s.f.). *YOLOv8: State-of-the-art computer vision model*. <https://yolov8.com/#what-is>

OpenCV. (s.f.). *Open Source Computer Vision Library*. GitHub. <https://github.com/opencv/opencv>

JaiedAI. (s.f.). *EasyOCR: Reconocimiento óptico de caracteres listo para usar*. Github. <https://github.com/JaiedAI/EasyOCR>

Base de datos

MySQL. (s.f.). *MySQL: The world's most popular open source database*. <https://www.mysql.com/MySQL>

Contenedores

Docker, Inc. (s.f.). *Docker: acelerando el desarrollo de aplicaciones en contenedores*. <https://www.docker.com/>

Entornos de desarrollo

JetBrains. (s.f.). *PyCharm: el único IDE de Python que necesitas.*

<https://www.jetbrains.com/pycharm/>

JetBrains. (s.f.). *IntelliJ IDEA: el IDE para el desarrollo profesional en Java y*

Kotlin. <https://www.jetbrains.com/idea/>

Framework de desarrollo web

Spring. (s.f.). *Spring Boot.* <https://spring.io/projects/spring-boot>

Estilos de página

Bootstrap. (s.f.). *Bootstrap: The most popular HTML, CSS, and JS library in the*

world. <https://getbootstrap.com/>

Repositorio del trabajo

Petteri Ketola. *TFG.* <https://github.com/Pextlordi/TFG>