TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

UNIVERSIDAD DE ALMERIA

ESCUELA SUPERIOR DE INGENIERÍA

"Sistema de reconocimiento de matrículas y monitorización en entornos urbanos"

Curso 2016/2017

Alumno/a: Antonio José Blanco Béjar

Director/es:
Jose Antonio Piedra Fernández
Nicolás Padilla Soriano

Agradecimientos

Primeramente agradecer a mis tutores, por aportarme sus conocimientos para poder desarrollar mi Trabajo de Fin de Grado, y así haber podido aprender un nuevo lenguaje de programación que hasta entonces no había utilizado.

De manera especial quiero agradecerles que se hayan adaptado a mi horario, puesto que trabajando ya era difícil de por sí, ellos supieron comprenderlo y buscar formas alternativas como el *Hangouts* de *GOOGLE*. Por esas ayudas en horarios no lectivos(9 o 10 de la noche), por haberme guiado cuando he tenido problemas, por su profesionalidad y su atención, gracias, muchas gracias.

A mí novia, que cuando la situación era difícil y creía que no podía más, ella siempre me recordaba que el título es lo más importante, que hay que aguantar y seguir luchando.

Por sus ánimos, cariño, comprensión, su enorme ayuda para todo y apoyo incondicional.

Por tantas horas de estudio, por haber estado meses sin pisar la calle y siempre a mi lado.

A mi familia, por haberme ayudado a llegar hasta aquí.

Y también a mi familia política, que me han dejado utilizar esa especie de calle en la que hemos podido realizar las fotos para el estudio, por su apoyo y por esos coches.

Gracias a todos, porque aunque el TFG lo he realizado yo sólo, lo cierto es que sin todas las personas mencionadas anteriormente no habría sido posible.

MUCHAS GRACIAS A TODOS.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1. Introducción

En este primer apartado, vamos a exponer la finalidad de la aplicación a desarrollar, además de justificarla con diferentes ámbitos de aplicación del software desarrollado. Para conseguir el fin propuesto, es necesario marcarnos unos objetivos, junto con una serie de etapas intermedias para cumplirlos. Por último, mostramos la estructura a seguir para cumplir con los objetivos propuestos.

1.1 Preámbulo

Vamos a desarrollar un sistema de detección y reconocimiento placas de matrículas de vehículos. Nuestro objetivo principal es que pueda ser empleado en zonas urbanas para el control de acceso a ciertas zonas restringidas o simplemente para la monitorización del tráfico.

Para ello vamos a trabajar con imágenes obtenidas por una cámara, sin el uso de tecnologías externas que puedan ayudar a la iluminación de la escena. Esto supone una gran dificultad a la hora de localizar la placa, principalmente en situaciones nocturnas donde la iluminación es menor y en situaciones con factores meteorológicos adversos.

Nuestro trabajo consiste en localizar la placa dentro de una imagen tomada por una cámara, además de identificar la matrícula que aparece en ella por medio de un módulo OCR (*Optical Character Recognition*). En la figura 1 se puede observar el resultado de este proceso sobre la imagen que se muestra.

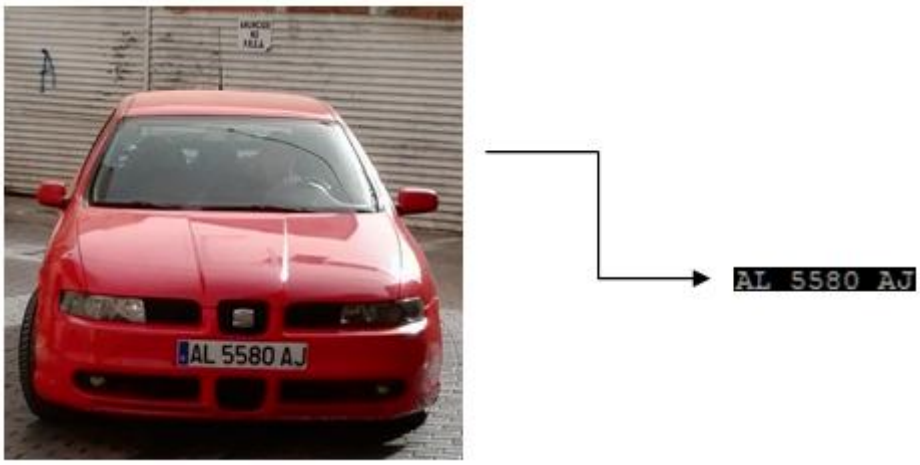

Figura 1. Resultado del detector con módulo OCR

El propósito de la visión artificial es obtener la información visual de la imagen para extraer características relevantes visuales. Aplicada a la industria, abarca la informática, la óptica, la ingeniería mecánica, y la automatización industrial. Por un lado, la visión artificial académica se centra en el procesamiento de imágenes, mientras que la industrial integra sistemas de adquisición de imágenes, dispositivos de entrada/salida y redes de ordenadores para el control de equipos destinados a la fabricación. De forma general, los sistemas de visión artificial están destinados a realizar inspecciones visuales de alta velocidad, funcionamiento continuo o repetitividad de las medidas.

En la industria, la finalidad de un sistema de inspección por visión artificial suele ser comprobar la conformidad de una pieza con ciertos requisitos, tales como las dimensiones, formas, presencia de componentes, etc.

1.2 Justificación

A lo largo de la presente memoria, se harán alusiones a las necesidades de sistemas de inspección automática de matrículas. Debido al creciente uso de vehículos por parte de toda la población, es necesario tener un control. Por ello, se hace un registro de las matrículas de los vehículos ya que es la forma más clara de identificarlos. Éstas son visibles externamente por todo el mundo y cada placa es única para cada vehículo.

La visión artificial se postula como la herramienta más apropiada para hacer el control de vehículos, como el caso que se muestra en la figura 2. Al realizar una comparativa entre una supervisión bajo un ojo humano y un sistema de inspección mediante visión, se obtienen ciertas ventajas en favor de la segunda opción:

- Menos tiempo de ejecución.
- Menor cantidad de recursos humanos necesarios.
- Menor fatiga por parte del usuario o equipo asociado a la tarea.
- Mayor fiabilidad en cuanto al acierto. Debido a la fatiga humana aparece el error en la inspección. Si se tiene una aplicación robusta a fallos, desaparecen los errores por agotamiento.

Existe el factor de que el sistema no tiene el raciocinio humano. Esto supone que la aplicación a implementar debe estar provista de una flexibilidad que permita adaptarse a nuevos cambios que puedan surgir, simulando "dentro de lo posible" el comportamiento racional de un individuo.

Figura 2. Ejemplo de aplicación[1]

Como parte de la justificación de este trabajo, se destaca un amplio campo de aplicación. Algunos de los usos más frecuentes son los siguientes:

- Control de acceso a parkings. El reconocimiento de la matrícula se utiliza para determinar cuánto tiempo ha permanecido un coche estacionado, o para llevar un registro de vehículos.
- Controles de acceso en los que sólo se permite la entrad de ciertos vehículos.

[1] Imagen extraída de: *www.addsp.com*

- En ciertos países, estos sistemas se instalan en las grandes ciudades para detectar y monitorizar el tráfico. Los vehículos son registrados en una base de datos.

- Control de flotas de vehículo en empresas de logística.

- Fuerzas de seguridad, policía de tráfico, etc. para localizar coches robados o sin seguro.

Además, existe la particularidad de que no es necesario ningún equipo acoplado para mejorar aspectos de la situación, como puede ser la iluminación.

1.3 Objetivos

Queda evidenciado que la finalidad del trabajo es la inspección automática de las matrículas que aparecen en la escena. De este modo, lo que se pretende es detectar la placa de la matrícula, además de reconocer los caracteres que en ella aparecen. Este es el objetivo principal, pero también hay otra serie de objetivos:

- La imagen puede ser tomada tanto de día como de noche en entornos iluminados. El objetivo es reducir la influencia de la luminosidad sobre los resultados de la aplicación.

- Las imágenes sobre las que se va a trabajar podrán proceder de cualquier tipo de cámara y podrán tener diferentes tamaños. El objetivo es adaptarse a la calidad de la imagen obtenida que varía en función de las máquinas utilizadas y del tipo de espectro que cubren para capturar la instantánea de la matrícula.

- La aplicación tiene que ser capaz de detectar los caracteres de imágenes capturadas a distancias variables.

- Las imágenes podrán ser tomadas con diferentes ángulos. Se pretende crear un sistema flexible.

Para poder cumplir estos objetivos, ha sido necesario marcar una serie de pautas definidas al inicio del proceso, las cuales son específicas para este proyecto y que han de cumplirse sin lugar a duda. Estas fases se pueden observar en la figura 3.

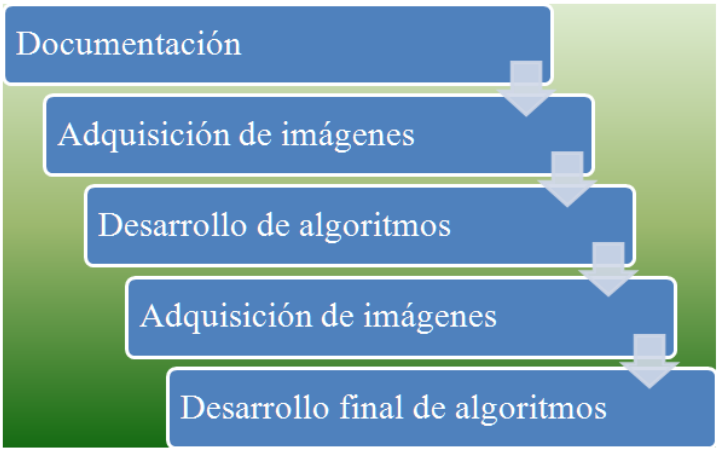

Figura 3. Fases en el desarrollo

En primer lugar realizar la documentación sobre la aplicación a resolver. Crear una estrategia o varias que lleven a la solución final y abrir una alternativa diferente a otras aplicaciones similares. Dentro de este contexto nace el estudio sobre los detectores de características locales que permiten detectar la matrícula en condiciones de luminosidad y en contextos muy variables.

La adquisición de imágenes es la fase más crítica, ya que de ello depende la utilización de unos algoritmos u otros. En esta fase se deben adquirir imágenes en las que la matrícula se presente a distancias, luminosidades y contextos variables. Pero para ello se debe de tomar una distancia máxima y otra mínima. También, el conjunto de imágenes tomado ha de ser amplio, de modo que se tenga la mayor cantidad de situaciones posibles.

La siguiente pauta es realizar la programación de todos esto algoritmos, que se van a desarrollar con la librería *OpenCV*. El lenguaje de programación elegido es *Python*, que se complementa con la librería elegida, *OpenCV*, que está destinada exclusivamente a la visión artificial.

Otra pauta es localizar la placa dentro de la imagen para poder extraer la matrícula del vehículo, aplicando diversos filtros para lograr detectar la zona donde se puede encontrar dicha matrícula.

La última pauta consiste en realizar una serie de operaciones para tratar la placa y que quede de la mejor forma posible. A partir de esto, se realiza el reconocimiento de caracteres a través de un algoritmo OCR.

Una vez se tenga una primera versión del trabajo, se realiza una segunda fase de captura de imágenes con características similares a las obtenidas anteriormente. El objetivo es corroborar que la aplicación funcione de manera correcta, además de pulir todos los aspectos más débiles de la aplicación.

Capítulo 2
Visión artificial

2. Visión artificial

En visión artificial por norma general se distinguen tres procesos, los cuales se solapan en alguna ocasión:

- **Procesamiento**: Conlleva manipular las imágenes vistas como señales digitales para extraer la información.
- **Análisis**: Está pensado para determinar ciertas estructuras elementales tales como bordes o regiones, y las relaciones existentes entre ellas.
- **Aplicaciones**: Tratan de solucionar los problemas relacionados con algunas situaciones del mundo real.

La visión puede ser, probablemente, el mecanismo sensorial de percepción más importante en el ser humano. No obstante, no quiere decir que sea el único, ya que una incapacidad visual no implica que ciertas actividades mentales no se puedan desarrollar.

El interés de los métodos de procesamiento de imágenes digitales se fundamenta en dos áreas de aplicación. La primera trata de mejorar la calidad de la percepción humana. La segunda tiene como fin procesar los datos que hay en la escena para que las máquinas puedan tener una percepción automática.

El concepto de visión artificial surge de intentar dotar a las máquinas de un sistema de visión. En gran cantidad de sistemas, la visión se complementa con otros mecanismos sensoriales como pueden ser detectores de alcance o proximidad. Es necesario integrar a posteriori todos los sensores para obtener el proceso de percepción global por completo. Las técnicas de procesamiento se utilizan hoy en día para resolver una gran diversidad de problemas. Generalmente no están relacionados, pero requieren del mismo modo métodos que sean capaces de realizar y extraer la información que hay en las imágenes para ser analizadas e interpretadas por los seres humanos.

2.1 ¿Qué es la visión artificial?

La visión artificial por computador es la capacidad de la máquina para ver el mundo que la rodea. De forma más precisa, es la capacidad para deducir la estructura y las propiedades del mundo tridimensional a partir de imágenes bidimensionales.

Desde el punto de vista de la ingeniería, un sistema de visión artificial es un sistema autónomo que hace las veces en algunas tareas del sistema de visión humano. El conjunto de tareas o información que un sistema de visión artificial puede llegar a realizar o extraer van desde una detección de objetos en una imagen hasta la interpretación tridimensional de escenas complicadas.

La visión por computador es una disciplina en pleno auge y de continuo interés en el campo científico-técnico, gracias a que existe un alto número de aplicaciones posibles y a la importante función que desempeña. Algunos de estos campos son la robótica, los procesos de inspección automática, navegación de vehículos, análisis de imágenes médicas, etc. La información visual es energía luminosa procedente del entorno. Para poder utilizar esta información es necesario que se transforme a un formato en la que pueda ser procesada. El ojo humano es el encargado de realizar esta misión en la visión humana, mientras que en la visión artificial esta tarea es llevada a cabo por una cámara. Esta

cámara convierte la energía luminosa en impulsos eléctricos, que de esta manera puede ser digitalizada para su procesamiento en un ordenador.

2.2 Similitudes y diferencias entre la visión artificial y la visión humana

A continuación, se va a realizar una comparativa para definir las principales diferencias y similitudes entre la visión artificial y la visión humana.

La visión humana puede definirse como un sistema integrado, de forma genérica, por dos ojos, el nervio óptico y el cerebro. El ojo humano, que se puede apreciar en la figura 4, es una lente que forma una imagen óptica y detecta la imagen con su retina. Desde aquí, a través del nervio óptico, la información es transmitida al cerebro para que se procese y se extraiga la información necesaria.

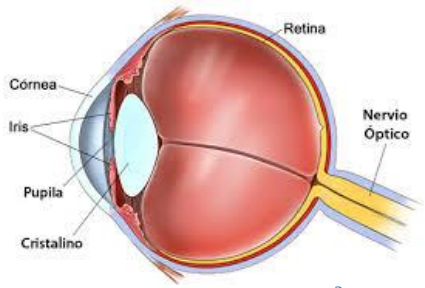

Figura 4. Ojo humano[2]

Respecto a lo anterior, se presentan una serie de similitudes en un sistema de visión artificial: una cámara con una lente recoge la imagen y se transmite la secuencia de video a un ordenador que analiza la información, la cual se puede emplear para el fin que se desee.

La similitud entre un sistema de visión artificial y el sistema de visión humano es muy alta. Por ejemplo, el ojo está identificado con la cámara.

El ojo humano es compuesto por una gran cantidad de órganos y partes bien diferenciadas. Muchas de ellas, salvando la comparativa biológica-mecánica, se corresponden con elementos de un sistema de visión automático (como se muestra en la figura 5):

- El iris de un ojo se puede decir que es el diafragma de una cámara digital. Ambos se encargan del control de cantidad de luz que entra al sensor.
- El cristalino realiza la función de la óptica. Consigue un enfoque correcto y un objeto nítido.
- La retina es similar al sensor. Determina la luminosidad que tiene que tener cada píxel de la imagen, es decir, se encargan de generar la imagen que realmente se ve.
- El nervio óptico hace de bus de conexión, ya que conecta la cámara con el sistema de procesamiento.
- El lóbulo parietal del cerebro es parecido al sistema de procesamiento de datos. Esto quiere decir que en un sistema automático de visión se aplica un algoritmo ya implementado que determina si la imagen cumple o no una serie de requisitos.

[2] Imagen extraída de: *www.fotonostra.com*

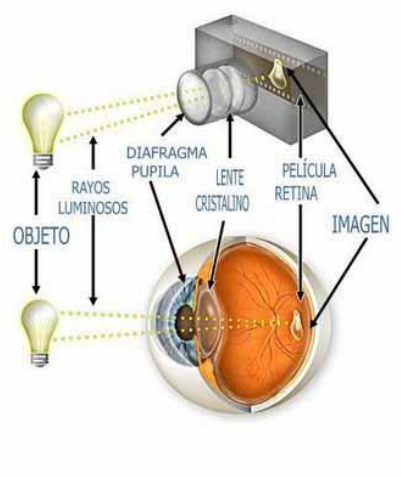

Figura 5. Correspondencia ojo humano - sistema de visión artificial[3]

La lista anterior muestra y resume los dos sistemas de visión, realizando una comparativa general.

Actualmente no es posible proveer a un sistema de una visión parecida a la humana. Para justificarlo, basta con decir que la medicina no es capaz de saber con certeza el funcionamiento del sistema de visión humano.

Tanto la detección de objetos a una cierta velocidad, como diferenciar la superficie de los mismos, son actividades que la visión humana realiza a diario. Es realmente complicado aplicar estos procesos en máquinas, por lo que se concluye que el cerebro humano tiene una gran capacidad de cálculo.

El desarrollo de la visión artificial parte de la visión humana, como punto de inicio. Casi todos los sistemas disponen una arquitectura modelada, hasta cierto punto, como la visión humana. En una persona el ojo es el encargado de detectar una imagen, la información es extraída por una parte del cerebro, otra parte del mismo acepta esta información y ordena las operaciones a realizar. En un sistema artificial, la cámara o el sensor hacen de ojo, un procesador que se ha construido y que está programado para el análisis de la imagen, procesa la imagen obtenida de la cámara y un actuador acepta la información que recibe el procesador y dirige las acciones necesarias.

Es imposible que las capacidades de la visión humana puedan ser imitadas con el mismo grado de calidad por un sistema de visión artificial. El ojo humano puede detectar 100 millones de píxeles en el espectro de la luz, mientras que una cámara normal puede llegar a detectar unos 250.000 elementos. Pero la tecnología avanza y poco a poco se van creando cámaras tan potentes que se van acercando al funcionamiento del ojo humano.

No obstante, para muchas aplicaciones es suficiente con una cantidad justa de píxeles que puede proporcionar cualquier cámara. También es cierto que si se quiere un aumento de píxeles, es necesario un incremento del tiempo de cómputo, el cual es en muchas ocasiones un factor determinante para dar validez a una aplicación.

Aun así, no hay sistemas que sean capaces de imitar con las mismas garantías de visión a la humana. *Por ejemplo, no hay un sistema de visión capaz de conducir un coche con tráfico y eventos no esperados.* Pero para uso en la industria no es necesario llegar a tal extremo, ya que la mayoría de las tareas de visión artificial son mucho más sencillas. El fin último de estos sistemas suele ser hacer trabajos sencillos y repetitivos, los cuales pueden llegar a fatigar la vista humana.

[3] Imagen extraída de: *www.fundamentoscientificos1b.blogspot.com.es*

Que una tarea sea monótona y repetitiva, hace que la efectividad de un humano en la inspección como en la figura 6, no sea superior al 80%. Mientras que un sistema automático de visión implementado de manera correcta puede llegar a una efectividad del 99,7%.

Figura 6. Inspección de productos[4]

Por este motivo surge la visión artificial, quedando bastante claro que no se puede imitar a un sistema de visión humana, pero puede llegar a superarlo cuando las tareas de repetición o peligrosas pueden disminuir la efectividad humana. Como resumen se destacan los siguientes postulados:

- La visión humana puede detectar más detalles.
- La visión humana maneja tareas más complicadas.
- La visión artificial tiene mejor respuesta en tareas de repetición.
- La visión artificial puede trabajar con luz visible o sin ella.
- Como resumen de las anteriores, hay una gran analogía entre visión artificial y visión humana.

2.3 Imágenes digitales

La visión artificial está fundamentada en el tratamiento de imágenes. Las instantáneas pueden ser fotos o ser extraídas de un vídeo obtenido a partir de cualquier tipo de cámara: cámara web, cámara de vídeo digital, etc. Las imágenes han de ser tratadas para llegar a obtener el máximo número de conclusiones posibles para conseguir el objetivo final del proyecto.

2.3.1 Formación de las imágenes digitales

El punto inicial de todo sistema de percepción visual es la imagen. El proceso para captar la imagen implica que se tenga que utilizar algún sensor para obtenerla representación bidimensional de la escena.

En el ser humano los ojos son los encargados de adquirir la imagen. Estos transforman la información visual en impulsos nerviosos que a continuación, va a utilizar el cerebro para interpretar la imagen. La estructura del ojo puede variar de unos humanos a otros.

La visión humana hace que se vean las imágenes como una representación continua. No obstante, para procesar una imagen en un ordenador, es necesaria una transformación previa de la imagen. Esta imagen continua debe ser transformada de alguna forma en un conjunto de números que pueda

[4] Imagen extraída de: *www.lyl.ingenieria.com*

manipular el ordenador. Este proceso es conocido como digitalización de la imagen. Para crear una imagen, hay que considerar una unidad básica: el píxel.

El píxel es la menor unidad homogénea que forma parte de una imagen digital. Si se amplía lo suficiente una imagen digital, se pueden observar los píxeles de la imagen. Aparecen como pequeños cuadros en color o en escala de gris.

La formación de las imágenes es debida a una sucesión de píxeles. El orden marca la coherencia de la información para el uso digital.

Cada píxel queda codificado por un conjunto de bits que tienen una longitud determinada, la cual es llamada profundidad de color. Por ejemplo, un píxel puede codificarse con un byte (8 bits), adquiriendo 256 variaciones: 2^8. En las imágenes reales se utilizan tres bytes para definir un color, es decir, en total se pueden representar 2^{24} colores (16.777.216).

Para transformar el valor de un píxel a un color, es necesario conocer la profundidad, el brillo del color y el modelo de color que se está usando. En el modelo de color RGB (Red-Green-Blue), se forman los colores en función del porcentaje que cada uno de ellos represente. Por ejemplo, para obtener el color violeta es necesario mezclar el rojo y el azul, pudiendo obtener diferentes tonalidades variando las proporciones de los colores primarios. En el modelo RGB se suelen utilizar 8 bits para indicar la proporción de cada uno de los colores primarios. Así, cuando uno de ellos vale 0 no interviene en la mezcla, mientras que cuando vale 255 (máximo valor para 8 bits) interviene con su máxima tonalidad. El modelo RGB es el utilizado por la mayor parte de los dispositivos.

Para expresar el tamaño de la imagen, se utiliza el concepto de Megapíxeles, el cual equivale a un millón de píxeles. Esta unidad es empleada con gran frecuencia para indicar la resolución de las cámaras de fotos digitales. Si una cámara tiene una resolución de 2048 x 1536, tiene una resolución de 3,1 Megapíxeles (2048 x 1536 = 3.141.728).

Al realizar fotos, el tamaño de las mismas queda definido por el número de mega píxeles que tenga la cámara y por el tamaño de las impresiones que se pueden realizar. Pero hay que tener en cuenta que cada Megapíxel se distribuye en un área y, por tanto, la diferencia entre 7 y 8 mega píxeles es menos representativa que entre 3 y 4, debido a que no se trata de una medida exponencial.

La resolución de una imagen queda definida como el número de píxeles que la describen. Se suele medir en píxeles por pulgada (ppi), definiendo la calidad de la imagen, lo que condiciona también la cantidad de memoria que va a ocupar. Por ejemplo, si una imagen tiene una resolución de 72ppi, implica que tiene 5.184 píxeles en una pulgada cuadrada (72 píxeles ancho x 72 píxeles alto).

Se puede deducir que a mayor resolución de una imagen, tiene más cantidad de píxeles que la describen. Es evidente que a mayor resolución, mejor representación se va a obtener utilizando un dispositivo de salida adecuado, ya que puede proporcionar un mayor detalle descriptivo y una transición de color más suave.

El tamaño de una imagen en píxeles expresa su tamaño expresado en píxeles horizontales y verticales. Se puede obtener de manera sencilla si se conoce el tamaño de impresión y la resolución de la imagen. Para ello basta con multiplicar el alto y el ancho por la resolución. Por ejemplo una imagen de 9 x 12 pulgadas escaneada a 300ppi, tiene una dimensión de 2700 x 3600 píxeles.

Estos conceptos están relacionados y dependen mutuamente, aunque se refieren a características diferenciadas y no se deben confundir. El tamaño de la imagen indica sus dimensiones una vez impresa, mientras que el tamaño del archivo indica la cantidad de memoria necesaria para almacenarla.

Lógicamente, la resolución de la imagen condiciona estos dos conceptos. Dado que la resolución de la imagen es fija, al aumentar el tamaño de la misma se reduce la resolución y viceversa.

Si se reduce la resolución manteniendo el tamaño, se eliminan píxeles y se tiene una descripción menos precisa de la misma además de unas transiciones de color más abruptas. El tamaño del archivo que genera la imagen es proporcional a su resolución, de modo que si se modifica este parámetro se modifica el tamaño del archivo.

Es éste, por tanto, un elemento importante para decidir la resolución de una imagen ya que plantea un compromiso a la hora de capturar toda la información que se necesita de la misma y mantener el tamaño del archivo. Todo va a depender del uso final de la Figura.

2.3.2 De números a colores

Para transformar la información numérica de un píxel en un color, hay que saber el modelo de color y su brillo, así como la profundidad del color. En este punto es donde aparece el modelo de color RGB, el cual permite crear un color compuesto por 3 colores básicos, dependiendo de la saturación de cada uno.

En los dispositivos gráficos se define el concepto de profundidad de color: que es la cantidad de bits con la que se codifican los píxeles. De este modo, cada píxel se puede expresar con un byte (8 bits), de modo que puede tener 256 variaciones de color. De forma general las imágenes utilizan 3 bytes para definir un color, es decir, se pueden representar 2^{24} colores. La mayor parte de los dispositivos que se utilizan en un ordenador utilizan el modelo RGB. Está basado en la adición de los 3 colores primarios, donde se puede representar un color por medio de la adición entre ellos.

Además, las escalas de grises y negros tienen sus variaciones, muy semejantes a los demás tonos. También existe el color blanco que es la máxima saturación de los 3 colores primarios, y el negro como la ausencia de todos ellos. En las figuras 7y 8 se pueden ver los dos ejemplos de escalas de grises.

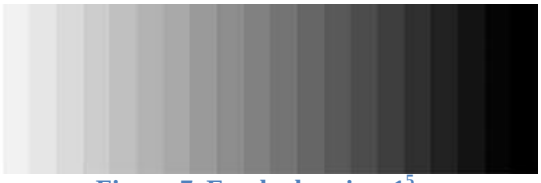

Figura 7. Escala de grises 1[5]

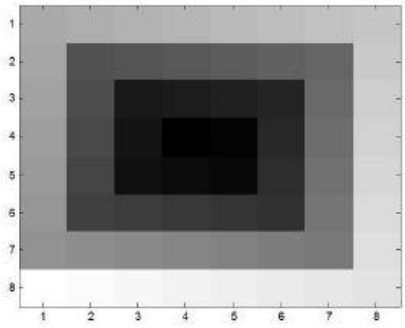

Figura 8. Escala de grises 2[5]

[5] Imagen extraída de: *www.fotonostra.com*

En la figura 9 se puede observar el valor de los píxeles en niveles de gris en una imagen. La saturación de los grises se representa en una escala de 0 a 255, representando cada elemento de la matriz a un píxel.

En el modelo RGB, cada color primario tiene su propia escala para representar un color específico. El problema sería tener 3 elementos de 3 matrices diferentes para cada uno de los posibles colores. Si se sabe que los 3 elementos de las distintas matrices se adicionan, como distinguir al color amarillo (255, 255, 0) del cyan (0, 255, 255).

173	177	181	185	189	193	197	201
169	85	89	93	97	101	105	205
165	81	29	33	37	41	109	209
161	77	25	5	9	45	113	213
157	73	21	17	13	49	117	217
153	69	65	61	57	53	121	221
149	145	141	137	133	129	125	225
255	253	249	245	241	237	233	229

Figura 9. Valor de los píxeles en niveles de gris[6]

2.4 Aplicaciones varias de la visión artificial

En muchas ocasiones la visión por computador no es la mejor solución a los problemas. Pero como ya se dijo, las soluciones humanas tienden a ser inexactas, lentas y sin rigor, por lo que en muchas ocasiones, especialmente en la industria, la visión artificial es la alternativa más y mejor considerada.

También cabe diferenciar aquellas aplicaciones en las que la visión artificial constituye una herramienta por sí sola y aquellas en las que es una parte de un sistema multisensorial. El primer caso se refiere a las aplicaciones donde la visión es una capacidad sensorial más para la percepción del entorno que rodea al robot. Se van a analizar varios campos de aplicación en las siguientes secciones.

2.4.1 Navegación en robótica

La visión es un elemento de un sistema con más sensores. La información que procede de la visión se valida, compara e integra con el resto de la información que proporcionan otros sensores.

Para la navegación en robótica se recurre generalmente a técnicas de visión estereoscópica para tratar de reconstruir la escena en 3-D. El uso del movimiento basado en la visión constituye un gran recurso.

[6] Imagen extraída de: *www.fotonostra.com*

2.4.2 Biología, Geología y Meteorología

En el campo de la biología se puede distinguir entre aplicaciones microscópicas y macroscópicas.

En una imagen microscópica se pueden encontrar una gran cantidad de organismos, que utilizando técnicas de segmentación, pueden ser aislados para identificarlos mediante propiedades tipo excentricidad, tamaño, color, etc. o simplemente para contar el número de microorganismos presentes en una imagen.

En imágenes macroscópicas se pueden utilizar las regiones para identificar diferentes tipos de texturas en vegetales o características de diferentes áreas naturales por su color o el crecimiento de ciertas especies por diferencia de imágenes. También puede ser interesante identificar el grado de floración de un determinado cultivo.

La visión artificial también es usada en geología para, por ejemplo, detectar movimiento de terrenos captando dos imágenes en diferentes momentos de tiempo para comprobar la variación mediante una diferencia de imágenes (bajo condiciones similares de iluminación).

En meteorología se utilizan las técnicas de detección y predicción del movimiento para observar la evolución de ciertas masas nubosas u otros fenómenos meteorológicos a través de imágenes recibidas vía satélite. También puede resultar interesante hallar la cota de nieve.

2.5.3 Medicina

En la rama médica hay muchas aplicaciones en las que está presente el procesamiento de imágenes, como se puede observar en la figura 10, y a menudo está orientado a detectar dolencias o enfermedades. Algunas de las técnicas utilizadas son la radiografías, resonancias magnéticas, etc.

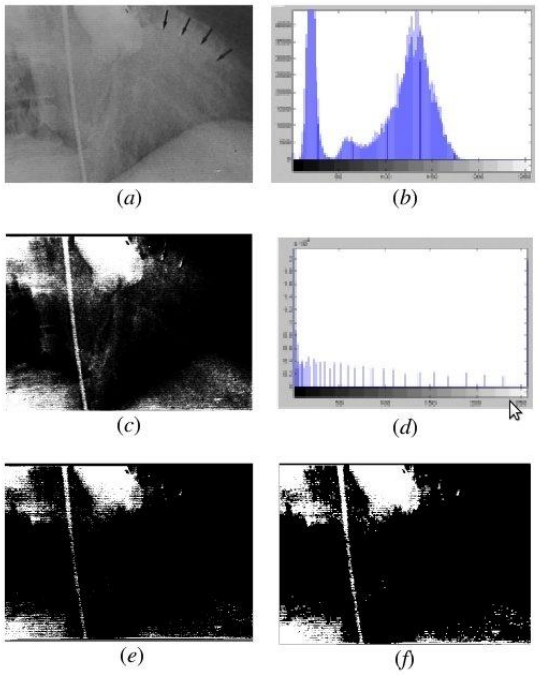

Figura 10. Sistema de inspección en medicina[7]

[7] Imagen extraída de: *imagenologia.robustiana.com*

Un claro ejemplo es el de la radiografía que se muestra anteriormente, en la que la imagen presenta baja calidad (a) y se puede extraer información sobre las manchas blancas que aparecen en la misma. En (b) se muestra su histograma de frecuencias. Se modifica el histograma mediante el aumento del contraste y gamma, obteniendo la imagen en (c) y su histograma en (d). Pero todavía se puede extraer más información, si se convierte a imagen binaria con un umbral a placer se obtiene (e) y mediante un proceso de rellenado de huecos se consigue (f). A partir de esta imagen se extraen las manchas para obtener el área de las diferentes regiones etiquetadas.

En el campo médico se pueden desarrollar otras aplicaciones como el movimiento de las paredes cardiacas a partir de imágenes de resonancia magnética.

2.5.4 Identificación de construcciones, infraestructuras y objetos en escenas de exterior

Es posible detectar la presencia de algunas regiones a través de la segmentación, utilizando imágenes de satélite o tomadas desde el aire. Se pueden utilizar para detectar la presencia de construcciones como edificios, carreteras, etc. a través de técnicas de extracción de bordes o contornos combinados con la segmentación.

Por ejemplo, se puede dar el caso de reconstrucciones de tejados de casas urbanas, que mediante una base de datos se puede partir de ella y elegir modelos para reconstruirlo. Otro ejemplo, consiste en la detección de carreteras usando el método de contornos deformables. Para ello la *transformada de Hough* puede ser muy útil.

2.4.5 Reconocimiento y clasificación

A partir del tamaño y el recuento, se puede desarrollar una aplicación para clasificar objetos. Por ejemplo, para contar monedas en función de su área, perímetro, número de *Euler*, etc. tras haberlas convertido a binario.

Otra aplicación destacada es el reconocimiento de caras de personas utilizando perfiles de intensidad. Otras aplicaciones presentes proponen una lectura automática de datos del DNI, así como reconocer objetos basados en el color.

También es posible el reconocimiento de huellas dactilares, utilizando un conjunto de máscaras. También se pueden mencionar el reconocimiento de caracteres usando aplicaciones OCR.

2.4.6 Inspección y control de calidad

La inspección se puede realizar para verificar ciertas características o verificar dimensiones en objetos manufacturados. La inspección de refiere a la verificación de si un objeto cumple con ciertos criterios. Para ello se compara un objeto con modelos que describan características buscadas. Se puede observar un claro ejemplo de este tipo en la figura 11.

Uno de los fines de los controles de calidad es detener la producción si se detecta que el sistema genera productos que no cumplen con estándares globales. Por ejemplo, en la fabricación de galletas se puede detectar cuando las galletas no son redondas utilizando la transformada de *Hough*.

Otra aplicación es la inspección de placas de circuitos impresos, donde hay numerosas pistas y permite verificar la ubicación de los componentes, chequear las pistas, etc.

Figura 11. Sistema de control de calidad[8]

2.4.7 Cartografía

Utilizando imágenes estereoscópicas, aéreas o de satélite se pueden obtener elevaciones del terreno usando técnicas de correspondencia basadas en el área. Se utiliza para la elaboración de catastros en zonas rurales, utilizando imágenes aéreas que permiten una fácil identificación de las parcelas y sus delimitaciones tras el tratamiento mediante técnicas de extracción de bordes y regiones. Si los sensores están perfectamente calibrados se puede llegar a determinar la superficie real de las parcelas basándose en el área de las imágenes medidas en píxeles. En la figura 12, se puede ver este tipo de aplicaciones.

Figura 12. Cartografía mediante visión artificial[9]

2.4.8 Fotointerpretación

Es la ciencia que trata del análisis de las imágenes por parte de un experto para extraer cierta información. Por ejemplo, para ver las construcciones existentes en una determinada imagen de satélite o una imagen aérea. A mayor calidad de la imagen, mejor resultado del análisis. Por eso las imágenes pueden ser tratadas con todas las técnicas encaminadas a mejorar la calidad de la imagen. Además, puede ocurrir que se quiera obtener una imagen en color por combinación de las bandas roja, verde y azul que proceden de un sensor multiespectral de satélite.

2.5 OpenCV

OpenCV proviene de los términos anglosajones Open *Source Computer Vision Library*. Por lo tanto, *OpenCV* es una librería de tratamiento de imágenes, destinada principalmente a aplicaciones de visión por computador en tiempo real, diseñada por Intel. Es gratis para uso comercial y de investigación, debido a su publicación bajo licencia BSD.

[8] Imagen extraída de: *www.metroinstruments.com*
[9] Imagen extraída de: *www.kumbaya.name*

Esta librería es multiplataforma, existiendo versiones para *GNU/Linux*, *Mac OS* y *Windows*. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos, calibración de cámaras, visión estéreo y visión robótica. La librería pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado, realizando su programación en código C y C++ optimizados, aprovechando además las capacidades que proveen los procesadores multinúcleo. *OpenCV* puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel.

Desde su aparición, se ha utilizado en infinidad de aplicaciones, desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos. Esta librería se compone de 4 módulos:

- *Cv*: contiene las funciones básicas de la biblioteca.
- *Cvaux*: contiene las funciones auxiliares (experimental).
- *Cxcore*: contiene las estructuras de datos y funciones de soporte para álgebra lineal.
- *Highgui*: funciones para el manejo de la GUI.

OpenCV tiene una estructura modular, lo que significa que el conjunto incluye diferentes librerías compartidas o estáticas. Los siguientes módulos están disponibles:

- *Core*: un módulo compacto que define las estructuras de datos básicas, incluyendo el denso *array* multi-dimensional Mat y funciones básicas usadas por otros módulos.
- *Imgproc*: un módulo de procesamiento de imágenes que incluye filtrados lineales y no lineales de imágenes, transformación geométrica de imágenes, conversión de color, histogramas. etc.
- *Video*: un módulo de análisis de vídeo que incluye estimación de movimiento, sustracción de fondos y algoritmos de seguimiento de objetos.
- *Calib3d*: algoritmos geométricos básicos multi-vistas.
- *Features2d*: detección de características más destacadas, descriptores.
- *Objdetect*: detección de objetos y ejemplos de clases predefinidas.
- *Highgui*: un interfaz de uso fácil para la captura de vídeo.
- *Gpu*: algoritmos GPU-acelerados de diferentes módulos *OpenCV*.
- Otros módulos de ayuda.

2.6 Algoritmos utilizados en visión artificial

La visión artificial está destinada a trabajar con imágenes. Las imágenes pueden ser fotos o extraídas de un vídeo obtenido a partir de una cámara de cualquier características (cámara web, cámara de vídeo digital, etc.).Para lograr el objetivo del trabajo hay que tratarlas, para obtener todas las características posibles.

También se hará uso de la tecnología que existe para que el procesamiento sea lo más sencillo posible, como puede ser el uso de ordenadores, programas y entornos de programación. En este trabajo no se han utilizado todas las técnicas destinadas al tratamiento de imágenes, sino que se han utilizado las que más se adaptaban a las necesidades.

Estos algoritmos se han obtenido, en numerosas ocasiones, de la librería *OpenCV* en la que ya estaban implementados, teniendo que modificar ciertos parámetros. También se ha realizado la programación de algoritmos para poder llegar al objetivo principal del trabajo, reconocer la matrícula.

A lo largo de este capítulo se van a enumerar en la medida de lo posible todas las técnicas de procesamiento de imágenes (visión artificial) utilizadas y explicar cada una de ellas.

2.6.1 Representación digital de una imagen. Tipos de imágenes en *OpenCV*

Para obtener una imagen en visión artificial, es necesario el uso de algún tipo de sensor que la proporcione. A partir del sensor se obtiene una representación en dos dimensiones de la escena.

Se podría representar una imagen como una función bidimensional de intensidad luminosa o nivel de gris, f(x, y) donde x e y denotarían las coordenadas espaciales bidimensionales y el valor de f en cada punto (x, y) sería proporcional a la luminosidad de la imagen en ese punto. Sin embargo, un ordenador no puede trabajar con una imagen que tenga una función continua, por lo que hay que muestrearla y digitalizarla. El muestreo aproxima la escena real a una imagen integrada por una matriz de MxN puntos, cuyos elementos representan el nivel de gris en cada punto de la imagen, o los valores en las correspondientes matrices de color RGB.

Cada imagen está compuesta por una matriz de píxeles, donde al brillo de cada píxel se le conoce como nivel de gris del mismo. De forma genérica, los valores de nivel de gris varían entre 0 y 255, donde los extremos son el blanco y el negro. Todos los demás valores representan distintas intensidades de gris.

Por norma general, 0 es el color negro y 255 el blanco, pudiendo diferenciar 256 variaciones de negro a blanco como se observa en la figura 13.

En función de la aplicación y los requerimientos necesarios, es mejor un tamaño de imagen u otro. Debido a ello, hay algunas aplicaciones donde las imágenes tienen muy baja resolución, mientras que otras necesitan una alta resolución.

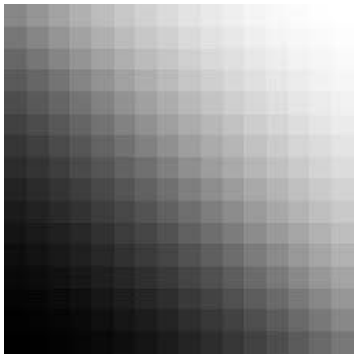

Figura 13. Diferentes tonalidades de gris[10]

[10] Imagen extraída de: *www.fotonostra.com*

En *OpenCV*, una imagen es una matriz de datos, es decir, un conjunto ordenado de elementos reales. Las imágenes se almacenan como matrices, donde cada elemento se corresponde con un píxel.

Para el almacenamiento de imágenes, *OpenCV* trabaja con números en punto flotante con precisión simple de 32 bits. Aunque para reducir la memoria se puede trabajar con matrices de enteros de 8 bits con signo, enteros de 16 bits con signo y enteros con signo de 32 bits.
En cuanto al tipo de imágenes en *OpenCV*, existen 3 tipos básicos de imágenes y entre ellas difieren por la forma en la que son interpretados los elementos de la matriz de datos.

2.6.1.1 Imagen binaria

Cada píxel asume uno de los valores discretos. En esencia estos dos valores se corresponden con blanco y negro. Una imagen binaria se almacena como una matriz bidimensional de 0 (negro) y 1 (blanco). Cada píxel se corresponde con 1 bit. En la Figura 14 se muestra un ejemplo de imagen binaria.

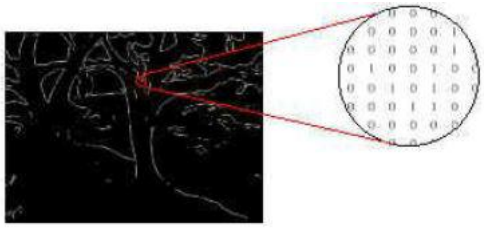

Figura 14. Imagen binaria[11]

Una imagen binaria puede considerarse como un tipo especial de imagen de intensidad, que contiene sólo el blanco y el negro.

Una imagen binaria puede ser almacenada en una matriz *double* o *uint8*. Sin embargo, una matriz *uint8* es preferible, ya que utiliza mucha menos memoria.

2.6.1.2 Imagen de intensidad

También llamada imagen en escala de grises. Consiste en una matriz de datos cuyos valores representan intensidades dentro de un mismo rango. La matriz puede ser *double*, en cuyo caso contiene los valores en el intervalo [0, 1], o de tipo uint8, en cuyo caso el rango es [0, 255]. Los elementos de la matriz representan diferentes intensidades o niveles de gris, donde el 0 representa el negro y el 255 el blanco. Cada píxel se corresponde con 1 byte, de ahí los 255 niveles permitidos. En la figura 15 se muestra un ejemplo de una imagen en escala de grises.

[11] Imagen extraída de: *www.pixabay.com*

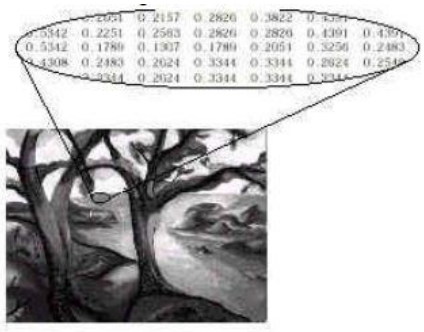

Figura 15. Imagen en escala de grises[12]

2.6.1.3 Imagen de color

También llamada imagen en formato RGB. Representa cada color del píxel como un conjunto de tres valores, que representan las intensidades de rojo, verde y azul que componen el color. Cada píxel se codifican con 3 bytes (uno por color), siendo el número total de colores posibles del orden de 16,7 millones. Esto es lo que se llama una imagen multicanal, en este caso 3 canales.

También existen imágenes RGBA de cuatro canales, donde el cuarto canal indica el nivel de transparencia de un píxel, con aplicaciones en visión nocturna o imágenes por satélite. En la figura 16 se muestra un ejemplo de imagen a color.

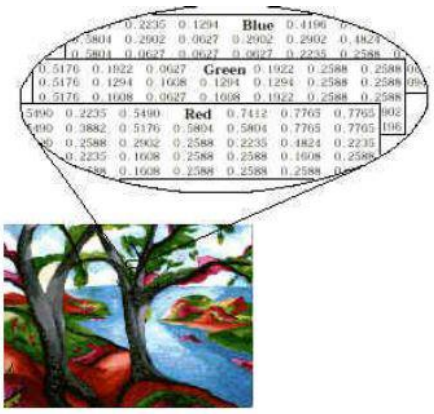

Figura 16. Imagen de color[12]

2.6.2 Etapas en el procesamiento de una imagen

Las etapas en el desarrollo de un proyecto dependen del objetivo del mismo. Aunque se pueden definir una serie de etapas de forma global para el procesamiento de una imagen como se observa en la figura 17.

[12] Imagen extraída de: *www.pixabay.com*

Figura 17. Etapas en el procesamiento de una imagen

Para un sistema de seguimiento visual, las etapas de segmentación, extracción de características y reconocimiento no suelen ser llevadas a cabo, mientras que sí se requiere la implantación de otras técnicas específicas. Analizando cada una de las etapas de manera individual, aparecen las siguientes definiciones:

- Adquisición de la imagen: El objetivo es obtener una imagen digital. Para este trabajo, la imagen es un vehículo en el que se ve la matrícula.

- Procesamiento previo: Se realizan operaciones sobre las imágenes para llevarla a un punto para obtener información de ellas en etapas posteriores.

- Detección de contornos: Está presente prácticamente en todos los procesamientos de imágenes. Una vez se tienen los contornos, se trabaja para obtener la información que se quiere.

- Segmentación: Obtiene todas las regiones donde los píxeles comparten algún atributo. Estas regiones son los objetos de interés en la imagen.

- Extracción de características: Estas características pueden presentarse en forma de líneas, círculos, elipses, formas determinadas, etc. pero también pueden tenerse como características locales de la imagen, es decir, puntos que atesoren multitud de información que pueda relacionarse con patrones externos.

- Localización y/o reconocimiento de objetos: Corresponde a la localización de un objeto. Muchas veces consiste en detectar formas o figuras con cierta forma. Junto con la localización suele aparecer el reconocimiento de estos objetos.

- Interpretación de la escena: Una vez obtenida la información en las etapas anteriores se procede a interpretar la escena, considerando para ello la relación entre los objetos simples previamente reconocidos y localizados, así como un cierto conocimiento sobre restricciones y reglas que rigen el mundo real. Esta etapa está estrechamente ligada a la inteligencia artificial, estando aún en sus primeros pasos y resultados definitivos.

En un sentido más amplio, las etapas se pueden agrupar en dos niveles: visión de bajo nivel y análisis de la escena.

La visión de bajo nivel incluye las primeras etapas de procesamiento. El segundo nivel es el análisis de la escena, cuya misión es coger las características que ha obtenido el primer nivel y hacer una descripción de la escena a un nivel superior.

2.6.3 Adquisición de la imagen

Las imágenes digitales son señales discretas, cuyo origen suele ser una señal continua. Para verlo de manera más clara, una cámara digital obtiene imágenes del mundo real, el cual es continuo. Existen dos etapas en el proceso de obtención de imágenes:

- **Captura**: Se utiliza un dispositivo, que suele ser óptico, con el cual se obtiene la información relativa a una escena.

- **Digitalización**: Se transforma la información obtenida, que es una señal con componentes continua, en la imagen digital, que es una señal con todas sus componentes discretas.

2.6.3.1 Modelos de captura de imágenes

Se distinguen entre dos dispositivos para la captura de una imagen. El primero es el dispositivo pasivo, el cual está basado en el principio de cámara oscura. El segundo se trata de un dispositivo activo, que se basa en el escaneo.

Los dispositivos basados en cámara aventajan a los basados en escaneo en cuanto a la velocidad. También son más simples y se asemejan más al sistema visual humano. Debido al gran avance de la tecnología, los modelos de cámara han acabado igualando, e incluso superando, a los de escaneo en cuanto a calidad de imagen obtenida se refiere.

Esta clasificación no incluye todas las formas posibles para la creación de imágenes, como pueden ser las imágenes sintéticas.

Cámara oscura:

Se puede usar para obtener imágenes de escenas tridimensionales (mundo real) y proyectarlas en un plano bidimensional. Ejemplos de este tipo son las cámaras de fotos y las de vídeo. Este modelo también se usa para obtener una representación en tres dimensiones si se usan dos o más cámaras para obtener diferentes perspectivas.

El escaneo:

En este caso existe un elemento activo (normalmente haz de láser) que recorre la escena que se quiere capturar. De este modo, son necesarios dos dispositivos, el emisor del haz y el receptor. Estos dispositivos se usan con diferentes fines.

2.6.3.2 Digitalización

Se trata del proceso de paso del mundo continuo al mundo discreto (analógico a digital). En la digitalización se distinguen dos procesos: muestreo y cuantificación.

Muestreo:

El muestreo consiste en la medición a intervalos respecto de alguna variable (tiempo o espacio). El parámetro fundamental es la frecuencia de muestreo, que representa el número de veces que se mide un valor por unidad de cambio. El número de muestras por unidad de cambio sobre el objeto original lleva al concepto de resolución espacial de la imagen, que se define como la distancia, sobre el objeto original, entre dos píxeles adyacentes.

Cuantificación:

La cuantificación consiste en la discretización de los posibles valores de cada píxel. Los niveles de cuantificación son potencias de 2 para facilitar el almacenamiento en el computador.

2.6.4 Operaciones generales. Técnicas clásicas

A continuación, se van a exponer una serie de operaciones generales que se pueden realizar sobre la imágenes con el fin de modificarlas.

2.6.4.1 Escalado de una imagen

El escalado es una transformación que se realiza a una imagen cuando se multiplican las coordenadas de cada píxel por un factor de escala. Si el factor está entre 0 y 1 la imagen se reducirá, mientras que si es mayor que 1 la imagen se aumentará.

El factor de escala puede ser diferente para ambas coordenadas. Se puede aplicar un factor sobre la coordenada horizontal y otro diferente sobre la vertical. Este escalado modifica la forma de las figuras y se conoce como escalado anisótropico, figura 18.

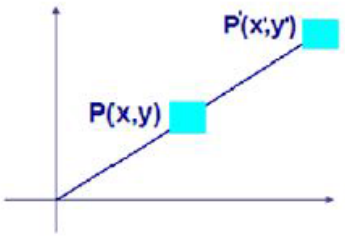

Figura 18. Escalado de una imagen

Las coordenadas de los píxeles de la imagen escalada (x', y') vienen dadas por las siguientes ecuaciones.

$$x' = sx \cdot x$$
$$y' = sy \cdot y$$

Si sx es igual a sy se producirá un escalado isotrópico. Si sx es distinto a sy el escalado es anisótropico.

2.6.4.2 Modificaciones en el color de una imagen. Escala de grises

Una imagen en escala de grises está representada por medio de una matriz bidimensional de m x n elementos, mientras que una imagen de color RGB es representada por una matriz tridimensional m x n x p. m y n denotan las dimensiones de la matriz y p representa el plano, que para RGB puede ser 1 para rojo, 2 para verde y 3 para azul.

La operación que permite transformar una imagen RGB en escala de grises obtiene las tres matrices. La imagen resultante en escala de grises tiene las dos dimensiones de la imagen y a mayores una dimensión más, donde se guarda la matriz unidimensional de los niveles de gris. Por tanto, la imagen inicial en color RGB será igual que la resultante en escala de grises salvo por el color.

En la figura 19, se puede apreciar la transformación de una imagen a color a escala de grises.

Figura 19. Imagen color / escala de grises

2.6.4.3 Conversión a imagen binaria. Conversión automática a imagen binaria.

Las imágenes digitales se componen utilizando un amplio rango de valores de intensidad, a las que se las denomina imágenes de nivel de gris. Normalmente el rango de niveles de gris es 256 (8 bits por píxel), pero ello no quiere decir que tenga que ser siempre así. De modo que puede variar dependiendo de la aplicación.

Por otro lado, hay aplicaciones que no necesitan tantos niveles de gris, ya que las imágenes pueden estar representadas por pocos niveles. En caso de reducir el nivel de gris hasta llegar a tener 2 valores, se tiene una imagen binaria. Este tipo de imagen se muestra en la figura 20.

Figura 20. Imagen escala de grises / binaria

Donde los píxeles en negro equivalen a 0 y los píxeles en blanco a 1 (255), que se almacenan en la memoria del ordenador. Con este tipo de imágenes se consigue reducir la representación y el procesamiento de la imagen al mínimo.

Reducir la representación de la imagen hace que se aproveche la memoria y la potencia computacional. Además, se pueden obtener las propiedades geométricas y topológicas de manera rápida de los objetos que componen la imagen.

En un ordenador el procesamiento de imágenes binarias es mucho más rápido, lo que ha provocado que se empleen estas imágenes para una gran mayoría de las aplicaciones. A día de hoy, que se ha mejorado de forma exponencial la potencia computacional para tratar imágenes en niveles de gris, se siguen utilizando las imágenes binarias para las aplicaciones. Esto se debe a que con las imágenes binarias se consigue una aplicación más simple y robusta.

Es preciso definir qué es la conversión a imagen binaria: es el proceso mediante el cual se convierte una imagen en escala de grises en una imagen binaria. Este proceso es necesario, ya que no existen cámaras que sean capaces de obtener imágenes binarias, lo máximo que consiguen son imágenes en niveles de gris.

Si se tienen imágenes en niveles de gris bien contrastadas, se puede realizar una conversión a imagen binaria con poco procesamiento, además de ser rápido y fiable. En ocasiones se utilizan técnicas de retro-iluminación para obtener imágenes bien contrastadas que se puedan reducir a binarias y quede representada claramente la información deseada.

La forma de obtener una imagen binaria es considerar el bit más significativo del nivel de gris de cada píxel. Significa fijar el umbral en el punto medio de la escala de grises, que sirve como referencia. Si el valor del píxel es menos que el umbral, ese píxel valdrá 0, mientras que si es mayor valdrá 1.

Aunque convertir a imagen binaria siguiendo esta técnica no es la forma más correcta, siendo descartada en muchas ocasiones. El rango dinámico de una imagen no siempre se extiende a todo el rango de niveles de gris posible y aún así, muchas veces es más adecuado fijar el umbral de conversión a binario en otro punto distinto del valor medio de la escala de grises.

Cuando una imagen está bien contrastada, apenas se pierde información. Muchas veces esta operación permite separar los objetos del fondo.

Para realizar una correcta conversión a binario, la etapa clave es la elección del umbral, que es la referencia para separar el fondo de los objetos. La separación se haría de forma ideal si se conociese la distribución de los píxeles oscuros y claros. En general estas distribuciones no son conocidas de antemano pero con el histograma de la imagen se obtiene la información del número de píxeles asociados a cada nivel de gris que resulta muy valiosa.

Si las distribuciones de los niveles de gris asociados a los píxeles claros y oscuros están muy separadas, entonces el histograma será *bimodal*. en este caso, el umbral más óptimo es aquel que divida ambos niveles de gris, estando situado en el valle del histograma.

Por el contrario, cuando las distribuciones comienzan a estar más solapadas la elección del umbral como un valor perteneciente al valle empieza a ser difícil, puesto que el valle ya no aparece tan claro. En este tipo de imágenes la conversión a binario no proporcionará buenos resultados.

Para establecer el umbral de conversión a binario en una aplicación industrial pueden utilizarse dos estrategias: preestablecer un umbral fijo para todas las imágenes capturadas o bien calcular dinámicamente en cada imagen el umbral idóneo.

La opción más sencilla es establecer un umbral determinado y fijo para realizar la conversión a binario. El valor más idóneo del umbral puede obtenerse analizando previamente los histogramas de las imágenes obtenidas en la aplicación. No es un mal método debido a que el coste computacional para determinar el umbral es nulo, pero debe ser implantado únicamente en entornos muy controlados, con una iluminación muy estable. La más mínima variación en la iluminación en la escena origina cambios en los niveles de gris de la imagen que invalidarían la idoneidad del umbral preestablecido.

Cuanto más anchura tenga el valle en el histograma, más fiable va a ser trabajar con un umbral fijo porque existirá un mayor margen de error. La anchura del valle puede absorber las oscilaciones en los niveles de gris que pueda haber por variaciones en la luz y hacer que el sistema funcione correctamente aún cuando existan perturbaciones. Una buena configuración del sistema de iluminación permitirá maximizar la distancia entre los dos picos que aparecen en el histograma correspondiente al objeto y al fondo.

Otra forma más ortodoxa para trabajar con un umbral fijado y mucho más robusta ante cambios en la iluminación, es calcular a partir de la información del histograma de cada imagen cuál es el valor más idóneo como umbral. El objetivo de esta técnica es obtener para cada imagen en escala de grises, la mejor imagen binaria localizando el umbral óptimo. De este modo se pueden evitar problemas que aparecen al tener un umbral fijo como puede ser el desplazamiento del histograma debido a cambios en la iluminación. Este tipo de conversión a binario se conoce con el nombre de conversión binaria adaptativa. Este tipo de conversión analiza los diferentes niveles de gris de la imagen y halla el umbral más óptimo para cada zona de la imagen. En la parte izquierda de la Figura 21 se muestra una imagen binaria con umbral fijo y en la parte derecha se muestra una binaria adaptativa.

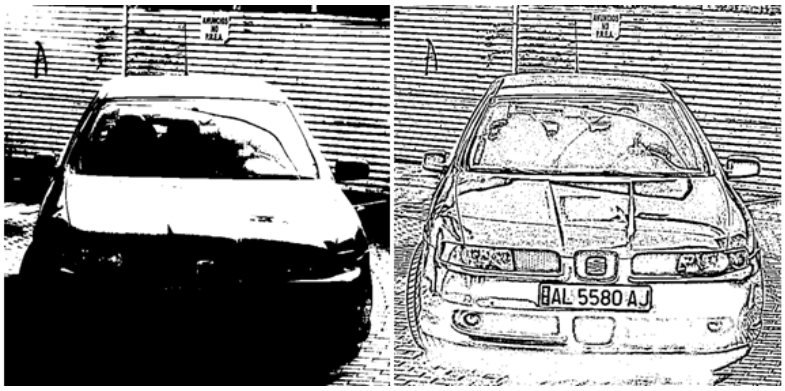

Figura 21. Diferentes conversiones a imagen binaria

Aunque estos algoritmos hallan el umbral más adecuado para cada imagen, hay que recordar que para obtener buenos resultados es necesario trabajar con imágenes que tengan un histograma *bimodal*. En

imágenes con histogramas relativamente planos tampoco se puede obtener una buena representación con dos niveles de gris aunque se emplee una búsqueda automática del umbral.

A continuación se expone un algoritmo sencillo no optimizado de elección de un umbral que funciona bien siempre que el histograma presente dos modos. Se calculan el valor de nivel de gris medio y su varianza para cada uno de los dos modos del histograma, uno entre 0 y k y el otro en k y N. El algoritmo busca iterativamente el valor m para el que la suma ponderada de las varianzas es mínima.

Sea una imagen I de N niveles de gris, el pseudocódigo para la conversión a binario automática es el siguiente:

- Calcular el histograma de I.
- Para cada valor k del nivel de gris, generar dos poblaciones dividiendo el histograma en k.
- Calcular las varianzas de los grupos generados.
- Calcular la suma ponderada de estas dos varianzas (ponderar con el porcentaje de píxeles de su población).
- Guardar este valor de la suma en un vector de N elementos.
- Buscar el índice m correspondiente a la suma mínima de las varianzas.
- Convertir a binario la imagen en el umbral m.

Teniendo un histograma *bimodal* como el de la figura 22, se observa de manera clara como el umbral óptimo es el que separa las dos crestas del gráfico, ya que está separando entre dos intensidades bien contrastadas y diferenciadas.

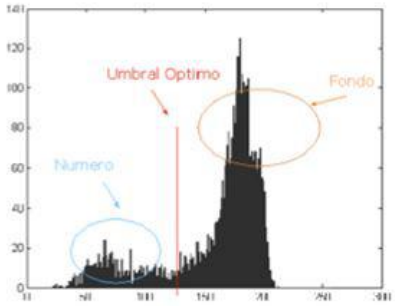

Figura 22. Histograma con el umbral óptimo[13]

2.6.5 Otras operaciones frecuentemente utilizadas

En este apartado se han incluido dos técnicas de tratamiento para imágenes binarias. Estas operaciones son la eliminación de regiones según su área, y el rellenado de figuras. Ambas técnicas son muy útiles, ya que acomodan la imagen y la llevan a una situación muy parecida, pero con más posibilidades de realizar un algoritmo robusto y que logre el objetivo.

[13] Imagen extraída de: *www.iie.fing.edu.uy*

2.6.5.1 Eliminación de regiones según su área

Existen la posibilidad de diferenciar varios objetos o regiones independientes en una imagen binaria. Avanzando un poco más, se puede llegar a eliminar regiones según su área. Así, por ejemplo, si se tiene una imagen binaria formada por varios objetos o etiquetas, se pueden eliminar todas aquellas que no lleguen a un mínimo o umbral que se estipule.

De este modo, se trata de una herramienta muy útil para el filtrado de imágenes, ya que elimina de forma eficiente el ruido, además de objetos menores que aparecen en la conversión a binario y no representan ningún objeto deseado.

En la figura 23 se puede apreciar el resultado de esta transformación.

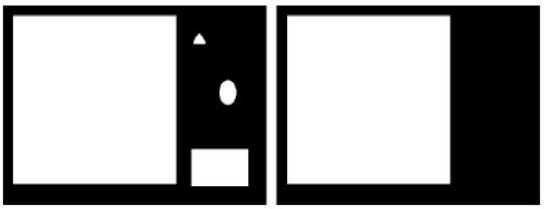

Figura 23. Filtrado de imagen por área[14]

2.6.5.2 Rellenado de figuras

Otra herramienta muy eficaz para procesar imágenes es eliminar los agujeros presentes en ellas. El agujero se puede definir como una región de puntos negros independientes que están completamente rodeados por puntos blancos.

El objetivo principal de esta técnica es que las figuras independientes sean más sólidas. Con ello se puede erosionar y eliminar con más margen, ya que se obtiene el contorno deseado con mayor facilidad.

El resultado se puede observar en la figura 24.

Figura 24. Rellenado de figuras[14]

2.6.6 Detección de contornos

Una de las técnicas más útiles es extraer los bordes en las imágenes. Un borde se puede definir como una frontera entre dos regiones cuyos niveles de gris son significativos.

La extracción de contornos a veces puede ser muy útil si se ejecuta en función de un determinado nivel de gris. La detección de contornos es usada en multitud de aplicaciones de visión artificial. Por

[14] Imagen extraída de: *docs.opencv.org*

ejemplo, a la hora de analizar las formas de ciertos objetos como por ejemplo detectar fallos de imprenta en el etiquetado, analizando las formas que aparezcan en las etiquetas y permitiendo con un simple cambio de umbral la detección de distintos logotipos o formas en una misma etiqueta.

Para determinar los bordes es necesario el uso del gradiente, con el cual se detectan todas las posibles direcciones del píxel en el que está.

$$\overline{gradI} = \left(\frac{dI}{dx}, \frac{dI}{dy}\right)$$

$$|gradI| = \sqrt{\left(\frac{dI}{dx}\right)^2 + \left(\frac{dI}{dy}\right)^2}$$

En la práctica lo que se hace es aplicar máscaras o filtros a la imagen. Estas máscaras son matrices que van recorriendo cada píxel de la imagen y realizando una operación determinada.

Como ya se comentó, un borde es la frontera entre dos regiones con niveles de gris significativamente distintos como se aprecia en la figura 25.

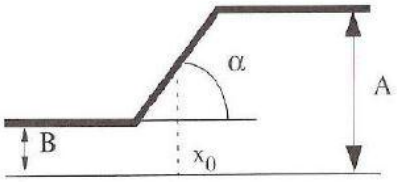

Figura 25. Dos niveles de gris

Donde A denota un nivel de gris alto; mientras que B denota un nivel de gris bajo.

Esta explicación es para una variación continua del nivel de gris. Pero en tratamiento digital de imágenes se tienen variables discretas. Ya que lo que se quiere es detectar cambios en la luminosidad, aparecen una serie de dificultades a la hora de detectar el píxel del contorno y entre las más distinguidas están las siguientes.

- Superficies con distinta orientación.
- Efectos de iluminación: existencia de sombras y reflejos que se pueden detectar como bordes por error.
- Cambios de texturas.

A continuación se van a explicar los operadores de contornos que más se utilizan: algoritmo de segmentación *Canny* y operador *Sobel*.

2.6.6.1 Algoritmo de segmentación Canny

Este algoritmo es usado para detectar todos los bordes existentes en una imagen. Está considerado como uno de los mejores métodos para la detección de contornos mediante el empleo de máscaras de *convolución* y basado en la primera derivada. Los puntos de contorno son zonas de píxeles en las que existe un cambio brusco de nivel de gris.

El algoritmo de detección de bordes y contornos de *Canny* se basa principalmente, en tres criterios:

- **Detección**: Su función es evitar la eliminación de bordes importantes y no obtener bordes falsos.

- **Localización**: Define que la distancia entre la posición real y la que se ha localizado del borde se debe minimizar.

- **Respuesta**: Que integra todas las respuestas que corresponden a un único borde.

El algoritmo de *Canny* se basa en los 3 siguientes grandes pasos:

- **Obtención del gradiente**: Se calcula la magnitud y orientación del vector gradiente en cada píxel.

- **Supresión no máxima**: Se logra el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un píxel de ancho.

- **Histéresis de umbral**: Se aplica una función de histéresis basada en dos umbrales; con ese proceso se pretende reducir la posibilidad de aparición de contornos falsos.

Para obtener el gradiente, el primer paso que se realiza es aplicar un filtro *gaussiano* en la imagen, de modo que se suaviza la imagen y se elimina el ruido. No es recomendable realizar un suavizado excesivo porque se pueden perder detalles de la imagen.

En la figura 26 se muestra un ejemplo del resultado de aplicar esta función.

Figura 26. Contornos por algoritmo de Canny

2.6.6.2 Operador Sobel

Es un operador diferencial discreto que calcula una aproximación al gradiente de la función de intensidad de una imagen. En cada punto de la imagen, obtiene el vector del gradiente que corresponde y la norma de este vector.

Este operador calcula el gradiente de la intensidad en cada píxel de la imagen. De este modo, para cada punto, el operador obtiene la magnitud del mayor cambio posible, su dirección y el sentido de oscuro a claro. El resultado muestra que tan abruptamente o suavemente cambia una imagen en cada punto analizado y, en consecuencia, que tan probable es que éste represente un borde en la imagen y, también, la orientación a la que tiende ese borde.

El gradiente de una función de dos variables para cada punto es un vector bidimensional cuyos componentes están dados por las primeras derivadas de las direcciones verticales y horizontales. Para cada punto de la imagen, el vector gradiente apunta en dirección del incremento máximo posible de la intensidad, y la magnitud del vector gradiente corresponde a la cantidad de cambio de la intensidad en esa dirección.

En la figura 27 se muestra un ejemplo de este detector de contornos.

Figura 27. Contornos por Sobel X / Sobel Y / Sobel

2.6.7 Reconocimiento de caracteres de la imagen

Una vez que la imagen ha sido tratada y se tiene en la perspectiva deseada, hay que realizar la extracción de caracteres. Es decir, reconocer las letras y los números que componen la placa de la matrícula. Para ello se implementa un algoritmo OCR (*Optical Character Recognition*).

El reconocimiento de caracteres se puede realizar de diversas formas, tales como operaciones XOR con unas plantillas, extracción de características (número de *Euler*, número de agujeros, etc.), etc.

Para reconocer los caracteres, se puede crear algoritmos propios o utilizar una librería ya existente.

La extracción de los caracteres va a permitir guardarlos en un archivo para que se puedan utilizar de la forma que el usuario desee.

Para poder aplicar un OCR, es importante que la calidad de la imagen donde se encuentran los caracteres sea alta, es decir, sin ruidos ni elementos extraños. También hay que tener en cuenta que la tipografía de texto a reconocer no debe ser poco común, ya que sería muy complicado detectar los caracteres.

Para su reconocimiento, se parte de la premisa de que los caracteres ya están segmentados, es decir, etiquetados. También es necesario que la imagen contenga sólo dos niveles de gris, es decir que esté convertida a binaria.

Aun así, las imágenes reales no son perfectas, de modo que el OCR se encuentra con varios problemas:

- El dispositivo que toma la imagen puede introducir niveles de gris al fondo que no pertenecen a la imagen original.

- La resolución de los dispositivos puede introducir ruido en la imagen, afectando los píxeles que han de ser procesados.

- La distancia entre caracteres, que no siempre es la misma, puede producir errores de reconocimiento.

- La conexión de dos o más caracteres por píxeles comunes también puede producir errores.

Todos los algoritmos OCR tienen como finalidad diferenciar texto de una imagen cualquiera. Para hacerlo se basan en cuatro etapas, que se muestran en la figura 28.

Figura 28. Etapas de un algoritmo OCR

- Conversión a binario: Los algoritmos OCR parten de una imagen binaria, de modo que es necesario convertir la imagen de entrada (color, escala de grises, etc.) en una imagen en blanco y negro. Al realizar esto, se conservan las propiedades esenciales de la imagen. El resultado de este proceso es una imagen en blanco y negro donde quedan marcados de manera clara los contornos de los caracteres y símbolos de la imagen.

- Segmentación de la imagen. Es el proceso más costoso y necesario para el posterior reconocimiento de caracteres. La segmentación implica la detección mediante el etiquetado de los contornos o regiones de la imagen. Para ello se basa en la información de intensidad o información espacial. No existe un único método genérico para realizar la segmentación que sea lo suficientemente eficaz para el análisis de un texto. Aunque las técnicas más utilizadas son variaciones de los métodos basados en proyecciones lineales.

- Adelgazamiento de componentes. Una vez etiquetados los componentes de la imagen, se les tiene que aplicar un proceso de adelgazamiento para cada uno de ellos. Este procedimiento consiste en ir borrando sucesivamente los puntos de los contornos de cada componente de forma que se conserve su tipología. La eliminación de los puntos ha de seguir un esquema de barridos sucesivos para que la imagen siga teniendo las mismas proporciones que la original y conseguir que no se deforme.

- Comparación de patrones. En esta etapa se comparan los caracteres obtenidos con anterioridad con unos teóricos que están almacenados en una base de datos. Para realizar esta comparación existen diversos procedimientos: métodos estadísticos, métodos estructurales, etc.

Capítulo 3
Sistemas de detección de matrículas

3. Sistemas de detección de matrículas

Para tener una idea clara de la aplicación que se quiere desarrollar así como su alcance en diferentes campos. En este capítulo se va a hablar sobre qué es un sistema de detección de matrículas, se va a proponer una vista general de algunos métodos que existen hasta el momento (estado del arte), se pondrán de manifiesto las dificultades que entraña el trabajo y por último se justificará argumentando casos prácticos en los que se puede usar.

3.1 ¿Qué es un sistema de detección de matrículas?

Un sistema de detección de matrículas es un sistema que se basa en la visión artificial y que es empleado para identificar a los vehículos a través de su placa. Es muy común usarlo en aplicaciones de seguridad, en el control de vehículos que se almacenan en una base de datos o en el control del tráfico.

Todos los vehículos poseen una matrícula, por lo que el mejor método para llevar un control de los mismos es a través del reconocimiento de sus placas. Se pretende que a través de la visión artificial, se pueda automatizar el reconocimiento de vehículos a través de su matrícula.

Hay que hacer una puntualización muy importante: las matrículas de los automóviles varían en función del país en el que estén matriculados. Por ello, hay que tener en cuenta que, normalmente, el software va a ser creado para un país en concreto. Esto va a implicar que las características del sistema va a estar orientadas al tipo de placa a tratar.

3.2 Diferentes modos de actuación.

La implementación de un sistema de detección de matrículas no es única. El abanico de posibilidades se puede ampliar si se tiene en cuenta que cada posibilidad cuenta con una multitud de variantes. En este apartado solo se van a mencionar las principales vías de actuación.

3.2.1 Sistema basado en la detección de límites verticales

Esta técnica utiliza el filtrado de imágenes para realizar un cálculo de las estadísticas verticales de la misma. El modo de actuación se basa en hacer una suma de las características verticales que hay en cada fila, de tal modo que la placa va a estar en el máximo de dicha suma como se observa en la figura 29. Se entiende como características verticales las variaciones más drásticas en el gradiente de la imagen. Para determinar los límites en filas donde está la matrícula, hay que fijar un umbral.

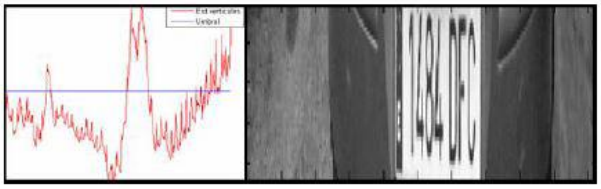

Figura 29. Detección de verticales[15]

Una vez que se tienen estos límites, se hace algo parecido para obtener los límites laterales, como se refleja en la figura 30. Se vuelve a realizar una suma, como en el caso anterior, y se reconoce el límite izquierdo como el primer máximo por la izquierda y el derecho como el primer máximo desde la derecha.

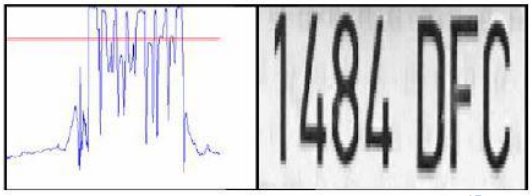

Figura 30. Detección de horizontales[15]

En una fase posterior se aplica un algoritmo OCR cuando la placa está completamente detectada para extraer los caracteres de la misma.

El inconveniente de este sistema es que al introducir un umbral para fijar los límites superior e inferior, se obliga a que la imagen que se obtenga sea siempre a la misma distancia de la matrícula.

3.2.2 Sistema basado en conversión binaria

Este es un método muy usado para todos los formatos de placas en los que el fondo es blanco. Consiste en convertir a binario la imagen con un umbral óptimo, el cual se suele conseguir a partir del histograma de la imagen. Con ello se consigue que la placa será representada por la mayoría de píxeles blancos en la matriz binaria como se observa en la figura 31 .

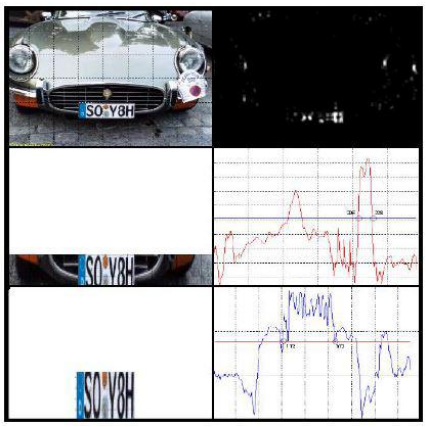

Figura 31. Detección a través de conversión binaria[15]

[15] Imagen extraída de: *docplayer.es*

La limitación que presenta este método es que es muy sensible a la iluminación ambiental, por lo que en un automóvil de color claro la iluminación juega a favor del color del mismo frente a la placa. En este caso, es muy difícil detectar la matrícula.

3.3 Diferentes variantes y problemas derivados de la escena

Debido a la variabilidad de la escena, la visión artificial no se puede considerar como algo exacto, lo que significa que todas las imágenes no tienen las mismas condiciones y características. En la adquisición se puede tener distorsión en cuanto a la perspectiva, ya que siempre hay variaciones y modificaciones. Éste no es el único factor, ya que también hay que tener en cuenta cambios en la iluminación, defectos en el objeto, etc.

Adecuando estos factores al presente trabajo, la traducción es que no se va a conseguir siempre una imagen en la que la matrícula salga en la misma posición ni a la misma distancia, de modo que se van a tener tamaños y orientaciones diferentes. También puede ocurrir que haya defectos en la matrícula (suciedad, fracturas, doblez, etc.). El color del coche se puede considerar un factor añadido, particularizando en el color del parachoques que sustenta la placa.

Desde el punto de vista propio existen dos alternativas para sobreponerse a estas dificultades:

- Adecuar la adquisición de las imágenes para que siempre tengan unas condiciones similares. Puede ser una buena solución siempre que el ámbito de aplicación del sistema sea concreto y bien definido; por ejemplo el acceso a un parking determinado. De este modo se puede comenzar a trabajar con un formato de imágenes que reúnen patrones similares.

- Implementar una aplicación de visión artificial flexible, que pueda adaptarse a un alto rango de variaciones. Para ello, las tecnologías a aplicar han de producir unos resultados invariantes a luminosidad, escala, orientación, etc. Esta solución es la mejor si se quiere un sistema que sea utilizado en varios ámbitos; por ejemplo controles de acceso, localización de vehículos robados, etc.

Hay que tener en cuenta esos factores para implementar un sistema que sea más flexible o menos. Si se adopta la primera solución, el desarrollo del sistema va a ser más sencillo. Pero si no se conoce el emplazamiento final del sistema, hay que adoptar la segunda alternativa.

3.4 Aplicaciones prácticas

Tal como se dijo en el primer capítulo, la alta densidad de vehículos en la sociedad lleva ligado un control necesario. También se justificó el reconocimiento de placas de matrícula a través de visión artificial como la alternativa más considerada.

Los sistemas de reconocimiento de matrículas que están basados en visión artificial tienen una gran versatilidad, lo que conduce a una gran cantidad de aplicaciones. No obstante, a continuación se hace referencia a las utilizadas más frecuentemente.

3.4.1 Localización de vehículos buscados o robados

Una aplicación muy demandada es la de encontrar vehículos que están en busca y captura. Estos vehículos suelen ser robados o con intención de ser localizados. La ubicación de la planta del sistema puede ser tanto algún punto fijo, como el interior de otro vehículo.

3.4.2 Alternancia en el acceso a núcleos urbanos

Estos sistemas pueden ser una solución para gestionar el acceso alterno al centro urbano de los vehículos en función de características de sus matrículas; por ejemplo limitar el acceso para las matrículas pares o impares en función del día de la semana. Este método supondría una alternativa a la descongestión de los núcleos urbanos, proporcionando igualdad de oportunidades a todos los usuarios.

3.4.3 Cobro de peajes urbanos

En algunos países se ha optado por esta alternativa. Con ello se ha conseguido mejorar la movilidad en las zonas del centro de la ciudad. También está la posibilidad de cobrar peajes urbanos a vehículos foráneos que usan las vías de la ciudad, sin contribuir al Impuesto de Circulación de la misma. Un claro ejemplo se puede observar en la figura 32.

Figura 32. Cobro de peajes urbanos[16]

3.4.4 Control de velocidad instantánea

En vez de controlar la velocidad instantánea de un vehículo, se puede detectar la matrícula del mismo, como se muestra en la figura 33, y por medio de una secuencia de vídeo hallar la velocidad a la que el vehículo se aleja. No es el mejor método, pero puede utilizarse como una alternativa en situaciones nocturnas.

Figura 33. Control de velocidad instantánea[16]

[16] Imagen extraída de: *www.laplace.us.es*

3.4.5 Control de velocidad media

Otra alternativa para sustituir a los controles de velocidad instantánea es hallar la velocidad media de los mismos. Para ello se reconoce la placa en el primer punto y cuando llega al segundo punto se calcula la velocidad media a la que ha circulado en ese tramo. Podemos ver este ejemplo en la figura 34.

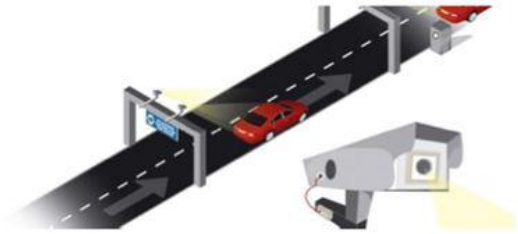

Figura 34. Control de velocidad media[17]

3.4.6 Sistema de parking

Para parkings donde los usuarios tienen plaza en propiedad, es muy útil usar una estrategia para reconocer los caracteres del vehículo que quiere entrar. Otro uso puede ser calcular el tiempo que un vehículo ha estado estacionado en un parking, identificando la placa a la entrada y la salida del mismo. En la figura 35 podemos ver un ejemplo de cómo sería este tipo de sistema.

Figura 35. Sistema de parking[18]

3.4.7 Optimización del tráfico urbano

Cada vez es más necesaria la optimización de la movilidad de los vehículos en las grandes ciudades, por lo que este sistema se presenta como una gran alternativa. Lo que se pretende es identificar los vehículos de forma dinámica, instalando estos sistemas en partes estratégicas y clave del centro urbano. La lectura de las matrículas proporciona velocidades medias y puede informar a los usuarios del tiempo que puede llegar a emplear en diferentes rutas. Se trata de una aplicación para optimizar el tráfico en tiempo real.

[17] Imagen extraída de: *www.laplace.us.es*
[18] Imagen extraída de: *www.addsp.es*

Capítulo 4
Raspberry Pi y Python

4. Raspberry Pi y Python

4.1 Raspberry Pi

Raspberry Pi es un ordenador de placa reducida (SBC) de bajo coste desarrollado en Reino Unido por la Fundación *Raspberry Pi*, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

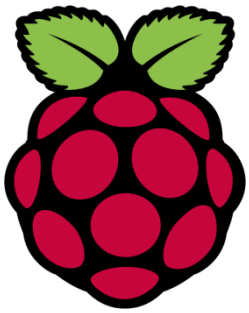

Figura 36. Logo de Raspberry Pi[19]

Aunque no se indica expresamente si es hardware libre o con derechos de marca, en su sección de preguntas y respuestas frecuentes (FAQ´s) explican que disponen de contratos de distribución y venta con dos empresas, pero al mismo tiempo cualquiera puede convertirse en revendedor o redistribuidor de las tarjetas *Raspberry Pi*, por lo que se entiende que es un producto con propiedad registrada pero de uso libre. De esa forma mantienen el control de la plataforma pero permitiendo su uso libre tanto a nivel educativo como particular. Tampoco deja claro si es posible utilizarlo a nivel empresarial u obtener beneficios con su uso, asunto que se debe consultar con la fundación.

En cambio el software si es *open source*, siendo su sistema operativo oficial una versión adaptada de *Debian*, denominada *RaspBian*, aunque permite otros sistemas operativos, incluido una versión de *Windows 10*.

El diseño incluye un *System-on-a-chip Broadcom BCM2835*, que contiene un procesador central (CPU) *ARM1176JZF-S* a 700 MHz (el firmware incluye unos modos "Turbo" para que el usuario pueda hacerle *overclock* de hasta 1 GHz sin perder la garantía), un procesador gráfico (GPU) *VideoCore IV*, y 512 MB de memoria RAM (aunque originalmente al ser lanzado eran 256 MB). El diseño no incluye un disco duro ni unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente; tampoco incluye fuente de alimentación ni carcasa. El 29 de febrero de 2012 la fundación empezó a aceptar órdenes de compra del modelo B, y el 4 de febrero de 2013 del modelo A.

[19] Imagen extraída de: www.raspberrypi.org

4.1.1 Historia

Este proyecto fue ideado en 2006 pero no fue lanzado al mercado febrero de 2012. Ha sido desarrollado por un grupo de la Universidad de Cambridge y su misión es fomentar la enseñanza de las ciencias de la computación los niños. De hecho, en enero de este año Google donó más de 15.000*Raspberry Pi* para colegios en Reino Unido. La *Raspberry Pi*, es una excelente herramienta para aprender electrónica y programación.

Los primeros diseños de *Raspberry Pi* se basaban en el microcontrolador *Atmel ATmega644*. Sus esquemas y el diseño del circuito impreso están disponibles para su descarga pública.

En mayo de 2009, *la Fundación Raspberry Pi* fue fundada en *Caldecote, South Cambridgeshire*, Reino Unido como una asociación caritativa que es regulada por la *Comisión de Caridad de Inglaterra y Gales*.

La fundación *Raspberry Pi* surge con un objetivo en mente: Desarrollar el uso y entendimiento de los ordenadores en los niños. La idea es conseguir ordenadores portables y muy baratos que permitan a los niños usarlos sin miedo, abriendo su mentalidad y educándolos en la ética del "ábrelo y mira cómo funciona". El ideólogo del proyecto, *David Braven*, un antiguo desarrollador de videojuegos, afirma que su objetivo es que los niños puedan llegar a entender el funcionamiento básico del ordenador de forma divertida, y sean ellos mismos los que desarrollen y amplíen sus dispositivos. El cofundador de la fundación es *Eben Upton*, un antiguo trabajador de la empresa *Broadcom*, el cual es el responsable de la arquitectura de software y hardware de la *Raspberry Pi*.

Eben Upton, se puso en contacto con un grupo de profesores, académicos y entusiastas de la informática para crear un ordenador con la intención de animar a los niños a aprender informática como lo hizo en 1981 el ordenador *Acorn BBC Micro*.

La fundación da soporte para las descargas de las distribuciones para la arquitectura ARM, *RaspBian* (derivada de *Debian*), *RISC OS 5*, *Arch Linux ARM* (derivado de *Arch Linux*) y *Pidora* (derivado de *Fedora*); y promueve principalmente el aprendizaje del lenguaje de programación *Python*. Otros lenguajes también soportados son *Tiny BASIC*, *C*, *Perl* y *Ruby*.

El primer prototipo basado en ARM fue montado en un paquete del mismo tamaño que una memoria USB. Tenía un puerto USB en un extremo y un puerto HDMI en el otro.

El primer lote de 10.000 placas se fabricó en Taiwán China, en vez de Reino Unido, con esto se conseguía un abaratamiento en los costes de producción y acortar el plazo de entrega del producto, ya que, los fabricantes chinos ofrecían un plazo de entrega de 4 semanas y en el Reino Unido de 12. Con este ahorro conseguido, la fundación podía invertir más dinero en investigación y desarrollo.

Las primeras ventas comenzaron el 29 de febrero de 2012 (Modelo B, mostrado en la figura 37). Las dos tiendas que vendían las placas, *Premier Farnell y RS Components*, tuvieron una gran carga en sus servidores inmediatamente después del lanzamiento. En los seis meses siguientes llegarían a vender 500.000 unidades.

Figura 37. Raspberry Pi Model B[20]

El 16 de abril de 2012 los primeros compradores empezaron a informar que habían recibido su *Raspberry Pi*. El 22 de mayo de 2012 más de 20.000 unidades habían sido enviadas.

El 6 de septiembre se anunció que se llevaría la producción de placas al Reino Unido, a una fábrica de Sony y que en ella se producirían 30.000 unidades cada mes, y se crearían 30 nuevos puestos de trabajo.

El 4 de febrero de 2013, se lanzó el modelo A, mostrado en la figura 38, pero debido a temas burocráticos los principales proveedores sólo lo pudieron poner a la venta ese día en Europa.

Figura 38. Raspberry Pi Model A[20]

En diciembre de 2015 se pueden comprar modelos con mejores prestaciones; *Raspberry Pi 2 Model B,* mostrado en la figura 39.

- Placa base (*ARM Quad-Core 900MHz, 1 GB RAM, 4x USB, HDMI, RJ-45*) de *Raspberry Pi*.

Figura 39. Raspberry Pi 2 Model B[20]

[20] Imagen extraída de: *www.raspberryshop.es*

En febrero de 2016 sale a la venta un nuevo modelo, la versión 3 con las siguientes características: *ARM Quad-Core 1,2 GHz, 1 GB RAM, 4x USB, HDMI, RJ-45* y una conectividad inalámbrica integrada de *802.11 b/g/n* y *Bluetooth*.

4.1.2 Hardware

Las ventas iniciales fueron del modelo B. El modelo A solo tiene un puerto USB, carece de controlador *Ethernet* y cuesta menos que el modelo B, el cual tiene dos puertos USB y controlador *Ethernet 10/100*. En 2014 se lanzó el modelo *Raspberry Pi 2 B*. El último modelo lanzado en 2015 es el *Raspberry Pi Zero*.

A pesar que el Modelo A no tiene un puerto RJ45, se puede conectar a una red usando un adaptador USB-Ethernet suministrado por el usuario. Por otro lado, a ambos modelos se puede conectar un adaptador Wi-Fi por USB, para tener acceso a redes inalámbricas o internet. El sistema cuenta con 256 MB de memoria RAM en su modelo A, y con 512 de memoria RAM en su modelo B. Como es típico en los ordenadores modernos, se pueden usar teclados y ratones con conexión USB compatible con *Raspberry Pi*.

El *Raspberry Pi* no viene con reloj en tiempo real, por lo que el sistema operativo debe usar un servidor de hora en red, o pedir al usuario la hora en el momento de arrancar el ordenador. Sin embargo se podría añadir un reloj en tiempo real (como el DS1307) con una batería mediante el uso de la interfaz I^2C.

Los esquemas del modelo A y el modelo B fueron lanzados el 20 de abril de 2012 por la fundación.

La aceleración por hardware para la codificación de vídeo (H.264) se hizo disponible el 24 de agosto de 2012, cuando se informó que la licencia permitiría su uso gratuitamente; antes se pensó en anunciarlo cuando se lanzara el módulo de cámara. También se puso a la venta la capacidad para poder usar el codificación-decodificación de MPEG-2 y Microsoft VC-1. Por otro lado, se hizo saber que el ordenador soportaría CEC, permitiendo que pudiera ser controlado mediante un mando a distancia de televisión.

El 5 de septiembre de 2012, se anunció una revisión 2.0 de la placa, que ofrecía un pequeño número de correcciones y mejoras, como unos agujeros de montaje, un circuito para hacer reset, soporte para depuración JTAG, etc.

El 15 de octubre de 2012, la fundación anunció que todos los Raspberry Pi Modelo B serían enviados a partir de ese momento con 512 MB de RAM en vez de 256 MB.

4.1.3 Especificaciones técnicas

	Raspbery Pi 1 Modelo A	**Raspberry Pi 1 Modelo B**	**Raspberry Pi 1 Modelo B+**	**Raspberry Pi 2 Modelo B**	**Raspberry Pi 3 Modelo B**
SoC:	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + puerto USB)			Broadcom BCM2836 (CPU + GPU + DSP + SDRAM + puerto USB)	Broadcom BCM2837 (CPU + GPU + DSP + SDRAM + puerto USB)
CPU:	ARM 1176JZF-S a 700 MHz (familia ARM 11)			900 MHz quad-core ARM Cortex A7	1.2 GHz 64-bit quad-core ARMv8
Juego de instrucciones:	RISC de 32 bits				
GPU:	Broadcom VideoCore IV, OpenGL ES 2.0, MPEG-2 y VC-1 (con licencia), 1080p30 H.264/MPEG-4 AVC				
Memoria (SDRAM):	256 MiB (compartidos con la GPU)	512 MiB (compartidos con la GPU) desde el 15 de octubre de 2012		1 GB (compartido con la GPU)	
Puertos USB 2.0:	1	2 (vía hub USB integrado)	4		
Entradas de vídeo:	Conector MIPI CSI que permite instalar un módulo de cámara desarollado por la RPF				
Salidas de vídeo:	Conector RCA (PAL y NTSC), HDMI (rev1.3 y 1.4), interfaz DSI para panel LCD				
Salidas de audio:	Conector de 3.5 mm, HDMI				
Almacenamiento integrado:	SD / MMC / ranura para SDIO		MicroSD		
Conectividad de red:	Ninguna	10/100 Ethernet (RJ-45) vía hub USB			10/100 Ethernet (RJ-45) vía hub USB, Wifi 802.11n, Bluetooth 4.1
Periféricos de bajo nivel:	8 x GPIO, SPI, I^2C, UART			17 x GPIO y un bus HAT ID	
Reloj en tiempo real:	Ninguno				
Consumo energético:	500 mA, (2.5 W)	700 mA, (3.5 W)	600 mA, (3.0 W)	800 mA, (4.0 W)	
Fuente de alimentación:	5 V vía Micro USB o GPIO header				
Dimensiones:	85.60mm x 53.98mm (3.370 x 2.125 inch)				
Sistemas operativos soportados:	GNU/Linux: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux. RISC OS				

4.1.4 Software

Otra diferencia importante entre la *Raspberry Pi* y el PC de escritorio o portátiles, aparte de su tamaño y su coste, es el sistema operativo (el software que permite controlar el ordenador) que utiliza.

La mayoría de los PCs y portátiles disponibles hoy en día funcionan con alguno de estos dos sistemas operativos: *Microsoft Windows* o *Apple OS X*. Ambas plataformas son de código cerrado, creados en un ambiente reservado utilizando técnicas patentadas. Estos sistemas operativos son conocidos como de código cerrado por la naturaleza de su código fuente, es decir, la receta en lenguaje de computadora que le dice al sistema que hacer. En el software de código cerrado, esta receta es mantenida como un secreto muy bien guardado. Los usuarios pueden obtener el software terminado, pero nunca ver cómo está hecho.

La *Raspberry Pi*, por el contrario, está diseñada para ejecutar el sistema operativo *GNU/Linux*. A diferencia de *Windows* u *OS X*, *Linux* es de código abierto. Esto quiere decir que es posible descargar el código fuente del sistema operativo por completo y hacer los cambios que uno desee. Nada es ocultado, y todos los cambios hechos están a la vista del público. Este espíritu de desarrollo de código abierto ha permitido a Linux rápidamente ser modificado para poder ejecutarse sobre la *Raspberry Pi*, un proceso conocido como portabilidad.

Varias versiones de *Linux* (conocidas como distribuciones) han sido portadas al chip *BCM2835* de la *Raspberry Pi*, incluyendo *Debian, Fedora Remix* y *Arch Linux*. Las distintas distribuciones atienden diferentes necesidades, pero todas ellas tienen algo en común: son de código abierto. Además, por lo general, todas son compatibles entre sí: el software escrito en un sistema *Debian* funcionará perfectamente bien en uno con *Arch Linux* y viceversa.

Igual que con la diferencia entre la arquitectura ARM y la x86, hay un punto clave que hace la diferencia práctica entre *Windows*, *OS X* y *Linux*: el software escrito para *Windows* u *OS X* no funciona en *Linux*. Afortunadamente, hay un montón de alternativas compatibles para la gran mayoría de los productos de software comunes y lo mejor, casi todos son de libre uso y de código abierto como lo es el propio sistema operativo.

4.1.5 Accesorios

En mayo de 2012, la fundación informó acerca de que se estaba experimentando con un módulo de cámara para *Raspberry Pi*, el prototipo usaba un sensor de 14 megapíxeles, y se conectaba al puerto CSI de la placa mediante un cable plano flexible. En noviembre del mismo año, se presentó el prototipo final en la feria Electrónica 2012 en Munich, y se dio a conocer que el sensor sería de 5 mega píxeles y que podría grabar vídeo a 1080p H.264 a 30 fotogramas por segundo. Finalmente el módulo se puso a la venta el 14 de mayo de 2013 en los principales proveedores. Las dimensiones del módulo son 25 x 20 x 9 mm. Para poder hacer uso de él, se tiene que activar en el menú *raspi-config* de *Raspbian*. Más tarde, a finales de octubre de 2013 se puso a la venta un módulo de cámara de infrarrojos. Actualmente podemos encontrar sensores de 8 mega píxeles (figura 40).

Figura 40. Cámara para Raspberry Pi[21]

Periféricos y carcasas son comercializados por empresas ajenas a la fundación. Por ejemplo la *Gertboard*, figura 41, que ha sido creada con propósito educativo, sirve para hacer uso del puerto *GPIO* y poder interaccionar con LEDs, interruptores, señales analógicas, sensores, y otros dispositivos. También incluye un controlador opcional para *Arduino* para poder interaccionar con el *Raspberry Pi*.

Figura 41. Gertboard para Raspberry Pi[21]

4.1.6 Uso

En enero de 2012, encuestas hechas en Reino Unido acerca de la penetración en las aulas de *Raspberry Pi* concluyeron que por cada placa que había en un colegio público, había cinco en colegios privados. Por ello se espera que en un futuro empresas patrocinen la adquisición de placas en colegios públicos.

El CEO de *Premier Farnell* declaró que el gobierno de un país de medio oriente expresó interés en proveer una placa a cada chica estudiante, con el objetivo de mejorar sus expectativas de empleo en el futuro.

A finales de enero de 2013, se dio a conocer que *Google*, junto con la ayuda de otros 6 socios, repartiría 15.000 placas entre estudiantes de Reino Unido que mostraran motivación por las ciencias de la computación.

[21] Imagen extraída de: *www.raspberryshop.es*

4.2 Python

Python es un lenguaje de programación creado por *Guido van Rossum* a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses "*Monty Python*". Es un lenguaje similar a *Perl*, pero con una sintaxis muy limpia y que favorece un código legible.

Figura 42. Logo de Python[22]

Se trata de un lenguaje interpretado o de *script*. Este tipo de lenguaje, es aquel que se ejecuta utilizando un programa intermedio llamado interpreté, en lugar de compilar el código a lenguaje maquina que pueda comprender y ejecutar directamente un ordenador.

La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo los lenguajes interpretados son más flexibles y más portables.

Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi-interpretado. En *Python*, como en Java y muchos otros lenguajes, el código fuente se traduce a un pseudocódigo máquina intermedio llamado *bytecode* la primera vez que se ejecuta, generando archivos **.pyc* o **.pyo* (*bytecode* optimizado), que son los que se ejecutaran en sucesivas ocasiones.

La característica principal de este lenguaje es la siguiente: no es necesario declarar el tipo de dato que va a contener una determinada variable, sino que su tipo se determinara en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.

Otra característica de este lenguaje es que no permite tratar una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Por ejemplo, si tenemos una variable que contiene un texto (variable de tipo cadena o *string*) no podremos tratarla como un número (sumar la cadena "9" y el número 8). En otros lenguajes el tipo de la variable cambiaría para adaptarse al comportamiento esperado, aunque esto es más propenso a errores.

El intérprete de *Python* está disponible en multitud de plataformas (*UNIX*, *Solaris*, *Linux*, *DOS*, *Windows*, *OS/2*, *Mac OS*, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios.

[22] Imagen extraída de: *www.python.org*

La orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos.

Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.

Capítulo 5
Desarrollo de la aplicación

5. Desarrollo de la aplicación

Como ya hemos visto en apartados anteriores, un ANPR (*Automatic Number Plate Recognition*) es un sistema automatizado para el reconocimiento de matrículas de automóviles. Este sistema lo hemos desarrollado utilizando el lenguaje de programación *Python* junto con la librería *OpenCV*, la cual proveerá las funciones para el procesamiento de las imágenes y poder detectar las zonas donde se encuentran las matrículas, detectar los caracteres de dicha matrícula y reconocerlos.

La imagen de partida del estudio para desarrollar nuestro ANPR ha sido la que se muestra en la figura 43. Se realizará un proceso experimental para la validación y búsqueda de los mejores algoritmos que se adapte al estudio.

Figura 43. Imagen principal

Principalmente, se desarrollarán dos grandes partes. La detección de la matrícula y el reconocimiento de esta, mostradas en la figura 44.

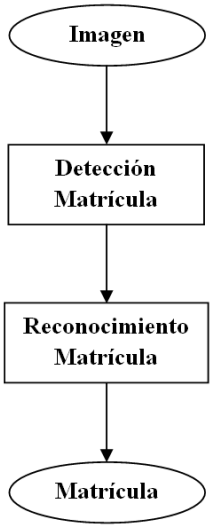

Figura 44. Diagrama de flujo general

5.1 Detección de la matrícula

En esta parte, desarrollamos el proceso para detectar la zona de la imagen que contiene la matrícula aplicando los siguientes algoritmos descritos en apartados anteriores. Este proceso se puede observar en la figura 45.

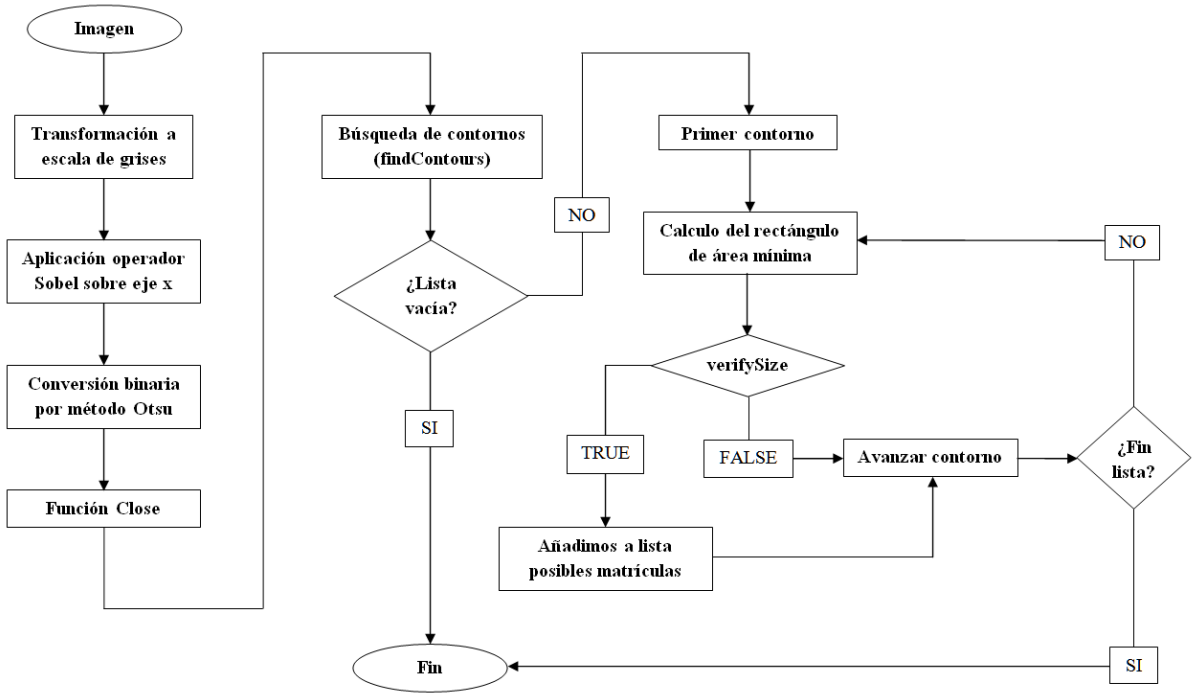

Figura 45. Diagrama de flujo, detección de matrícula

Primero pasamos la imagen a escala de grises, como se puede observar en la figura 46.

Figura 46. Imagen principal (en escala de grises)

Una vez que tenemos la imagen en escala de grises, calculamos el gradiente X con el operador *Sobel* (figura 47).

Figura 47. Imagen principal (sobel x)

Ahora, convertimos a binaria la imagen anterior con el método de *Otsu*, lo que nos da la siguiente imagen (figura 48).

Figura 48. Imagen principal (conversión binaria por método Otsu)

Seguimos con la función *CLOSE* aplicada a la imagen anterior. Su resultado se muestra en la figura 49.

Figura 49. Imagen principal (aplicada función CLOSE)

Llegados a este punto, aplicamos la función de *OpenCV findContours*, lo que nos devuelve una lista de contornos.

Una vez que tenemos la lista de contornos, hacemos lo siguiente para cada contorno:

- Calculamos el rectángulo de área mínima para el contorno.

- Verificamos si el rectángulo de área mínima calculado se puede tratar de una matrícula, para ello hemos definido la función booleana *verifySize(candidate)*, se puede observar en la figura 50, donde definimos la relación de aspecto de la matrícula, un área mínima, un área máxima y un margen de error. Con esto podemos comprobar mediante unas restricciones (por el área del rectángulo o por su inclinación) si el rectángulo que queremos verificar es una posible matrícula.

```
def verifySize(candidate):
    error=0.4
    #Spain car plate size: 52x11 aspect 4,7272
    aspect=4.7272
    #Set a min and max area. All other patches are discarded
    areamin= 16*aspect*16 # minimum area
    areamax= 70*aspect*70 # maximum area
    #Get only patches that match to a respect ratio.
    rmin= aspect-aspect*error
    rmax= aspect+aspect*error

    size = candidate[1]
    angle = candidate[2]
    height = size[0]
    width = size[1]

    if(height!=0 and width!=0):
        area= height * width
        r= width / height
        if(r<1):
            r= 1/r
    else:
        return False

    if(( area < areamin or area > areamax ) or ( r < rmin or r > rmax ) or \
        (height<width and (angle < -90 or angle > -80)) or \
        (height>width and (angle > -0 or angle < -10))):
        return False
    else:
        return True
```

Figura 50. Función verifySize(candidate)

- Si la función *verifySizes(candidate)* nos devuelve *True*, añadimos el rectángulo a una lista para almacenarlo. Si nos devuelve *False* no hacemos nada, por lo que estaríamos descartándolo.

Como podemos ver en la figura 51, en el caso de nuestra imagen principal, solo detecta un rectángulo como posible matrícula que concuerda con lo que estamos buscando. En un principio, podemos decir que nuestro algoritmo es potencialmente fiable en cuanto a la detección de la matrícula, aunque solo para esta imagen de momento, ya que posteriormente lo lanzaremos para otras imágenes para comprobar cuanto de fiable puede llegar a ser.

Figura 51. Imagen principal (posibles matrículas detectadas)

Una vez detectadas las zonas donde existe la posibilidad de encontrar una placa, creamos un recorte de cada zona utilizando una transformación de perspectiva y creando una imagen de 520 x 110 píxeles (al igual que las medidas reales una placa de matrícula 520mm x 110mm), para conseguir que los caracteres de la placa nos aparezcan lo más alineados posibles en el eje horizontal. El resultado de realizar esta función lo podemos ver en la figura 52.

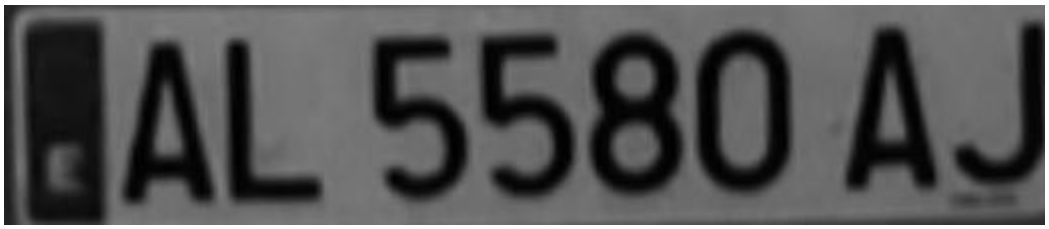

Figura 52. Imagen principal (matrícula recortada con transformada de perspectiva)

5.2 Reconocimiento de la matrícula

Una vez obtenida la imagen donde se encuentra la posible placa, le realizamos un tratamiento similar al anterior para, en este caso, segmentar los caracteres de la placa y su posterior reconocimiento. Lo podemos ver en el diagrama de la figura 53.

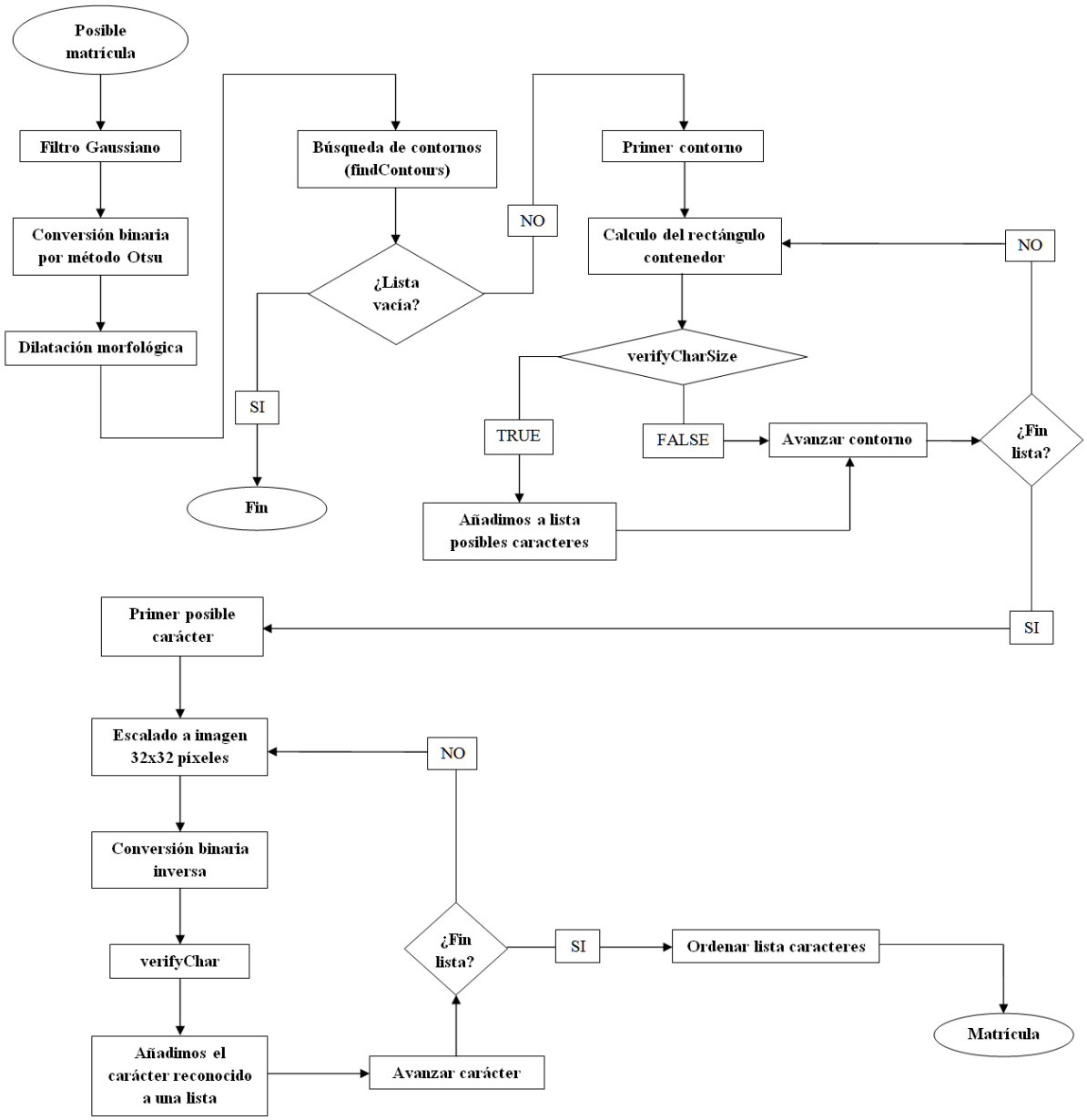

Figura 53. Diagrama de flujo, reconocimiento de la matrícula

5.2.1 Detección de caracteres

En primer lugar, le aplicamos un filtro *Gaussiano* para suavizar bordes (figura 54)

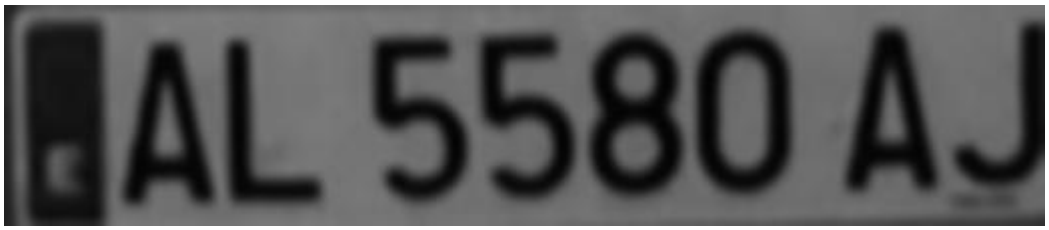

Figura 54.Imagen principal (matrícula con filtro Gaussiano)

A continuación, aplicamos una conversión binaria a la imagen anterior con el método de *Otsu*. El resultado se muestra en la figura 55.

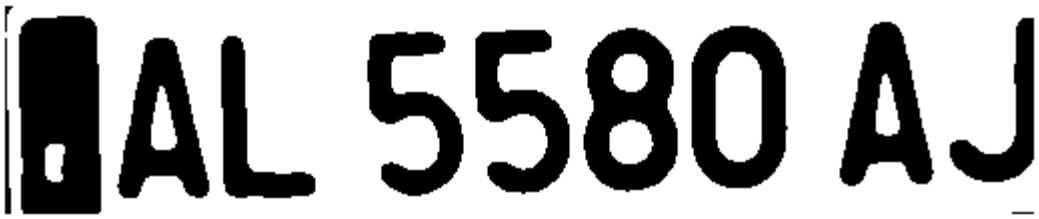

Figura 55. Imagen principal (matrícula con conversión binaria por método Otsu)

Seguidamente, le aplicamos una dilatación morfológica para adelgazar los caracteres.

Llegados a este punto, aplicamos la función de *OpenCV findContours*, lo que nos devuelve una lista de contornos. En la figura 56 se pueden apreciar los contornos detectados.

Figura 56. Imagen principal (matrícula con contornos detectados)

Una vez que tenemos la lista de contornos, hacemos lo siguiente para cada contorno:

- Calculamos los puntos extremos (superior, inferior, izquierda, derecha).
- Creamos dos vértices (superior-izquierda e inferior-derecha) correspondientes al rectángulo que contiene al posible carácter detectado. Con estos dos vértices creamos un candidato con la siguiente información: vértice superior-izquierda, vértice inferior- derecha y el tamaño del rectángulo que contendrá la altura y la anchura.

- Verificamos si el rectángulo obtenido se puede tratar de un carácter, para ello hemos definido la función booleana *verifyCharSize(charCandidate)*, se puede observar en la figura 57, donde definimos la relación de aspecto de un carácter (45 píxeles / 77 píxeles), un margen de error, una altura mínima y una altura máxima. Con esto podemos comprobar mediante unas restricciones (por la relación de aspecto o por su altura) si el rectángulo que queremos verificar es un posible carácter.

```
def verifyCharSize(charCandidate):
    error=0.35;
    #Char sizes 45x77
    aspect=45.0/77.0;
    #Set a min and max height. All other patches are discarded
    minHeight=70; # minimum height
    maxHeight=90; # maximum height
    #We have a different aspect ratio for number 1, and it can be ~0.2
    minAspect=0.2;
    maxAspect=aspect+aspect*error;

    size = candidate[1]
    height = float(size[1])
    width = float(size[0])

    if(height!=0 and width!=0):
        area= height * width
        r= (width / height)
    else:
        return False

    if (( r < minAspect or r > maxAspect ) or \
        (height < minHeight or height > maxHeight) or \
        (height < width)):
        return False;
    else:
        return True;
```

Figura 57. Función verifyCharSize(charCandidate)

- Si la función *verifyCharSize(candidate)* nos devuelve *True*, añadimos el rectángulo a una lista para almacenarlo. Si nos devuelve *False* no hacemos nada, por lo que estaríamos descartándolo.

Según el número de posibles caracteres podemos determinar si una imagen corresponde a una placa de matrícula o no, ya que si no detecta entre 6 y 8 caracteres podemos concluir que no se trata de una placa de matrícula

5.2.2 Reconocimiento de caracteres

En las posibles placas donde se detecte entre 6 y 8 caracteres procederemos a reconocer dichos caracteres de la siguiente manera:

- Para cada posible dígito detectado, creamos un recorte utilizando los datos de sus vértices y tamaño, excepto si tiene una anchura menor a 25 píxeles, puesto que se podría tratar de un uno, por lo que crearíamos el recorte añadiéndole anchura para que se asemeje a la anchura media de los demás caracteres.

- Redimensionamos el recorte para obtener una imagen de 32 x 32 píxeles.

- A continuación le aplicamos una conversión binaria inversa de umbral fijo 127 (los píxeles con un valor por debajo de 127 los pondremos a blanco, 255, y los píxeles con valor por encima de 127 a negro, 0).

- Verificamos el carácter con la función de *verifyChar(imagenChar)* donde hemos definido un diccionario en el cual tenemos como clave una letra o número y asociado a dicha clave, la plantilla correspondiente a la letra o número de la clave. En la figura 58 se pueden ver todas las plantillas utilizadas para realizar el reconocimiento.

Figura 58. Plantillas utilizadas para el reconocimiento de caracteres

Recorremos el diccionario utilizando el método *Template Matching* que nos compara la imagen del carácter detectado con la de la plantilla. Esta comparación nos devuelve un valor entre 0.0 y 1.0 que corresponde al grado de similitud entre las dos imágenes. Determinamos como grado de similitud mínimo un valor de 0.55, es decir, que las imágenes sean semejantes en un 55% y establecemos una variable auxiliar para guardar el grado de similitud del carácter reconocido para, de esta forma, establecer una condición para guardar solo el carácter que más se asemeje a nuestro carácter detectado.

- Esta función *verifyChar(imageChar)*, que se muestra en la figura 59, nos devuelve el carácter más semejante a la imagen que se está analizando en ese momento, *imageChar*.

```
def verifyChar(imageChar):
    #Creamos un diccionario donde la clave sera el caracter correspondiente a la plantilla
    caracteres = {'0':numero0,'1':numero1,'2':numero2,'3':numero3,'4':numero4,
                  '5':numero5,'6':numero6,'7':numero7,'8':numero8,'9':numero9,
                  'A':letraA,'B':letraB,'C':letraC,'D':letraD,'E':letraE,'F':letraF,
                  'G':letraG,'H':letraH,'I':letraI,'J':letraJ,'K':letraK,'L':letraL,
                  'M':letraM,'N':letraN,'O':letraO,'P':letraP,'Q':letraQ,'R':letraR,
                  'S':letraS,'T':letraT,'U':letraU,'V':letraV,'W':letraW,'X':letraX,
                  'Y':letraY,'Z':letraZ}

    caracter = '' #Variable para almacenar el caracter reconocido
    res_ant = 0 #Variable para almacenar el valor de similitud del caracter reconocido. Inicializamos a un valor de 0

    #Recorremos el diccionario para reconocer el caso actual utilizando el metodo Template Matching
    for x, y in caracteres.iteritems():
        template = y
        w, h = template.shape[::-1]

        th_temp, template = cv2.threshold(template,127,255,cv2.THRESH_BINARY)

        res = cv2.matchTemplate(imageChar,template,cv2.TM_CCOEFF_NORMED)

        threshold = 0.55 #Valor minimo de similitud entre el caracter detectado y la plantilla

        if((res >= threshold) and (res >= res_ant)):
            caracter = x #Incluimos el valor de la clave del diccionario a la variable del caracter reconocido
            res_ant = res #Actualizamos el valor de similitud con respecto al caracter reconocido

    return caracter
```

Figura 59.Función verifyChar(imageChar)

Cada carácter reconocido lo añadimos a otro diccionario cuyo clave será el punto x del vértice superior-izquierda y el valor, el carácter obtenido en el reconocimiento (en la figura 60 se muestra este resultado), para posteriormente hacer uno de la función *sort()* y ordenar los caracteres por su posición (en la figura 61 se muestran los caracteres ordenados).

```
[(291, '8'), (237, '5'), (110, 'L'), (419, 'A'), (183, '5'), (345, '0'), (58, 'A'), (474, 'J')]
```

Figura 60. Lista de caracteres reconocidos

```
[(58, 'A'), (110, 'L'), (183, '5'), (237, '5'), (291, '8'), (345, '0'), (419, 'A'), (474, 'J')]
```

Figura 61. Lista de caracteres reconocidos ordenados

Una vez ordenados solo tenemos que recorrerlo e ir añadiendo lo caracteres a una variable final la cual será la matrícula reconocida, como se muestra en la figura 62.

AL5580AJ

Figura 62. Resultado final obtenido

Como en este primer caso el resultado final es bastante favorable, esta primera fase la vamos a dar por finalizada. En el siguiente apartado probaremos este proceso en diferentes tipos de imágenes para refinar los resultado que vayamos obteniendo.

5.3. Estudio y comprobaciones de la aplicación.

Para verificar el correcto funcionamiento de la aplicación hemos tomado imágenes de cuatro vehículos a diferentes distancias y a diferentes alturas de la posición de la cámara. De los cuatro vehículos, las imágenes de tres de ellos (matrículas 3455 GWP, 8808 HPG y AL 5452 AH) han sido tomadas en un día soleado a medio día. Las imágenes del vehículo restante (matrícula 2010 JPK) han sido tomadas en un día nublado al atardecer, por lo que hay menos iluminación que las demás.

La toma de imágenes se ha hecho en tres alturas 0.5, 1.5 y 2.5 metros; y a una distancia del vehículo de aproximadamente 2, 4, 6, 8 y 10 metros.

Para las imágenes tomadas a distancias de 2 y 10 metros no obtenemos resultados positivos, como se puede ver en la tabla 1, por lo que nos centraremos en el estudio de las distancias entre 4 y 8 metros.

Distancia (m)	Altura (m)	2010 JPK	3455 GWP	8808 HPG	AL 5452 AH
2	0.5		3455GWP		AL5452AH
	1.5		-3-55--P	--------	--5-52-
	2.5				
10	0.5				
	1.5				
	2.5				

Tabla 1. Resultados obtenidos en 2 y 10 metros de distancia

En esta tabla y en las que veremos posteriormente, se interpretan de la siguiente manera:

	No detecta matrícula
--5-52-	Detecta matrícula y caracteres, pero el reconocimiento no es correcto.
3455GWP	Detecta y reconoce la matrícula al completo

Como se puede observar en la figura 63 del *Caso A: altura 0.5m / distancia 4m*, y se ha explicado en ocasiones previas, en cada imagen procesada es más que posible que detecte más de un posible candidato de ser matrícula. Estas se descartarán al no detectar caracteres por lo que al final siempre nos quedaremos con el candidato correcto. Como podemos ver en la figura 63, donde si se muestran todos los candidatos que se detectan, reconoce la matrícula completamente. En los demás casos hemos omitido las falsas detecciones.

5.3.1. Caso A: altura 0.5m / distancia 4m.

Para este caso, como se puede observar en las figuras 63, figura 64, figura 65 y figura 66, tenemos 3 reconocimientos correctos de 4 posibles por lo que tenemos una tasa de acierto del 75%, pero en cambio, si realizamos el porcentaje de acierto sobre los caracteres reconocidos en vez de sobre matrícula completamente reconocida la tasa de acierto asciende al 96% (28 caracteres reconocidos correctamente de un total de 29).

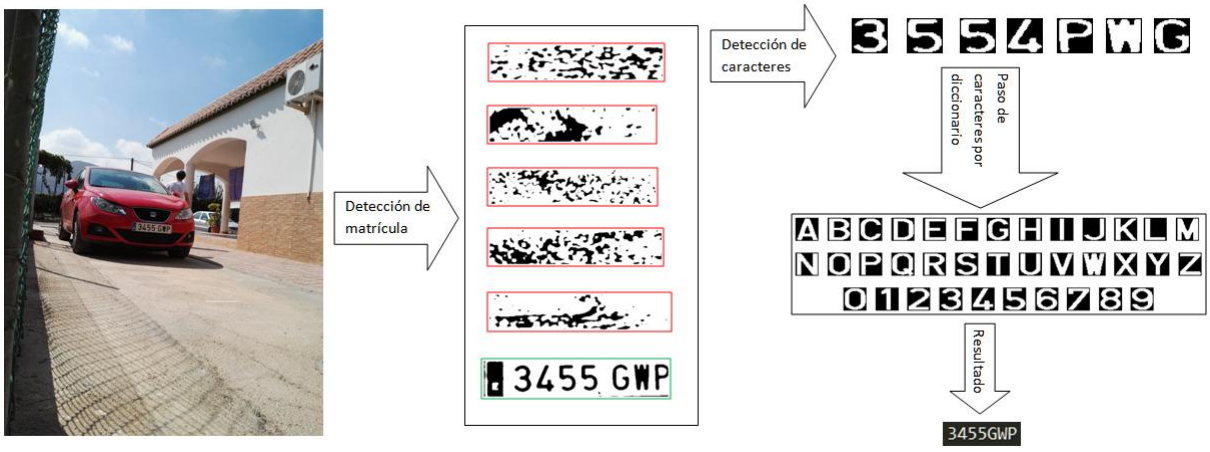

Figura 63. Seguimiento de "3455GWP" a 0.5m de altura y 4m de distancia

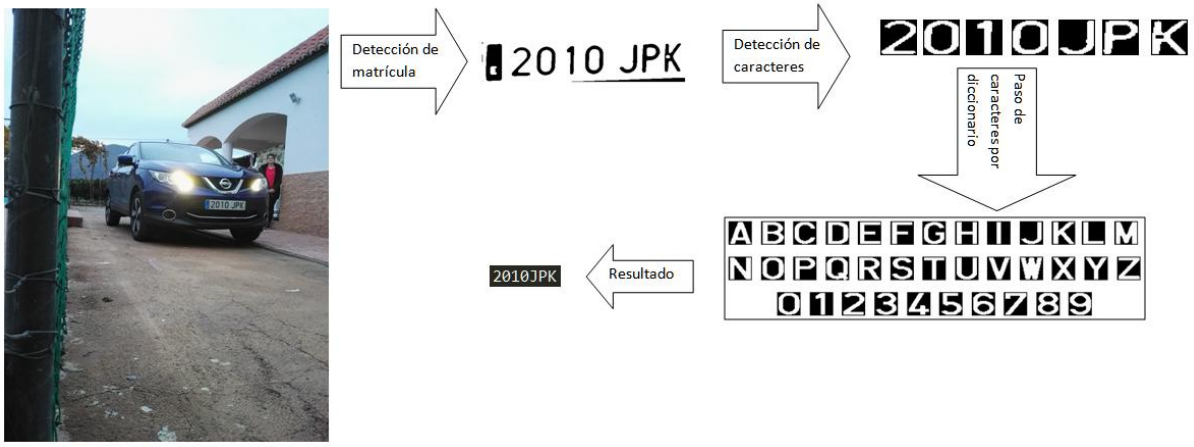

Figura 64. Seguimiento de "2010JPK" a 0.5m de altura y 4m de distancia

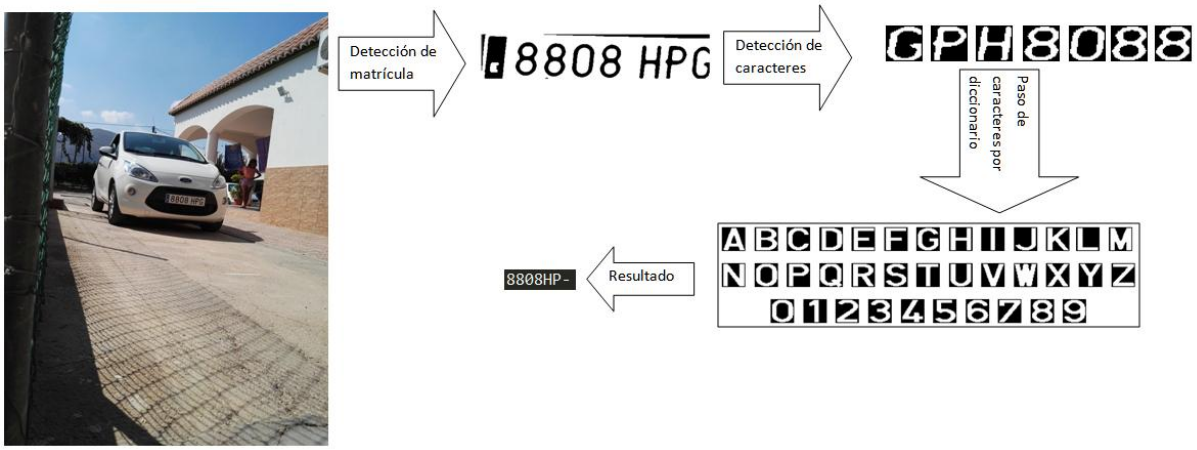

Figura 65. Seguimiento de "8808HPG" a 0.5 m de altura y 4m de distancia

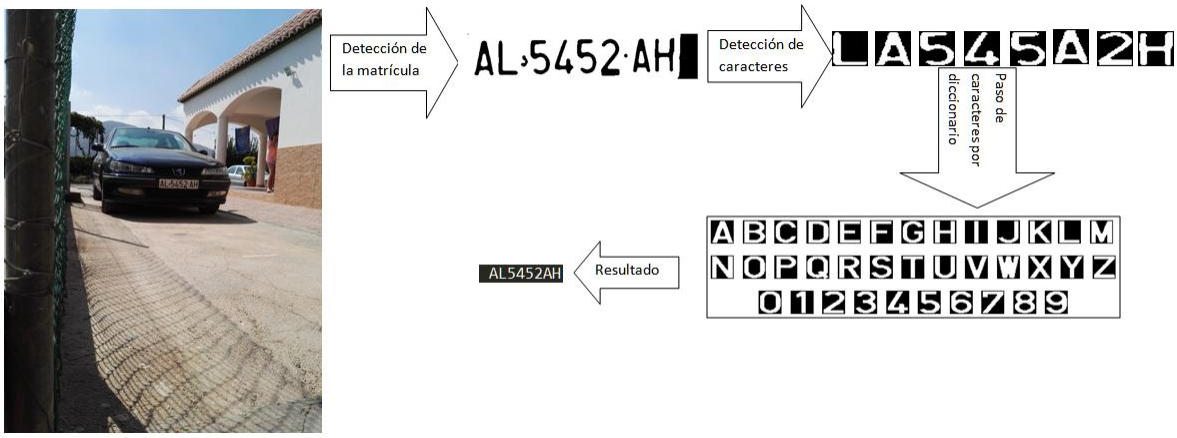

Figura 66. Seguimiento de "AL5452AH"a 0.5m y 4m de altura

5.3.2. Caso B: altura 0.5m / distancia 6m.

Para este caso, figura 67, figura 68, figura 69 y figura 70, tenemos el mismo resultado que el anterior: un 75% de tasa de acierto en cuanto a matrícula completamente reconocida y un 96% de tasa de acierto en cuanto a reconocimiento individual de caracteres.

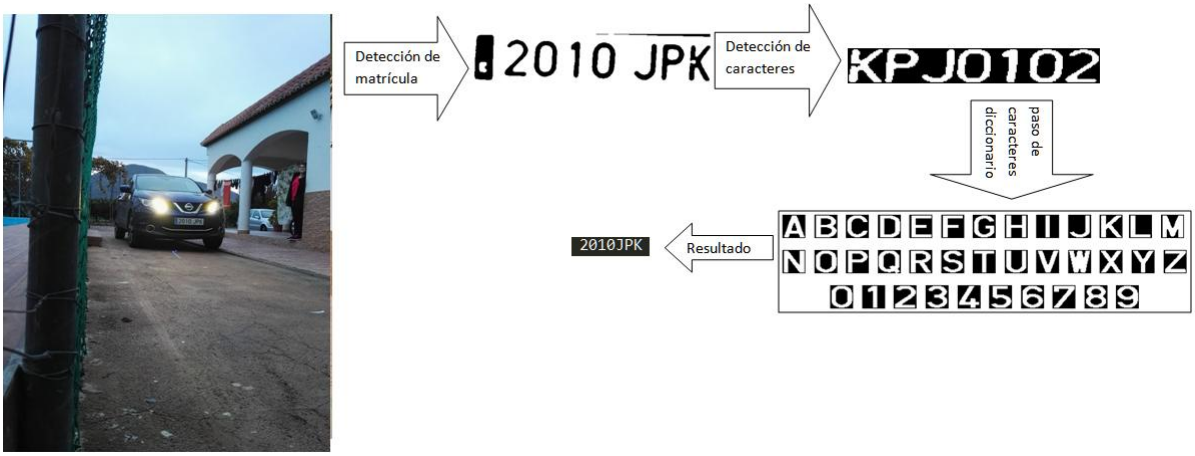

Figura 67. Seguimiento de "2010JPK" a 0.5m de altura y 6m de distancia

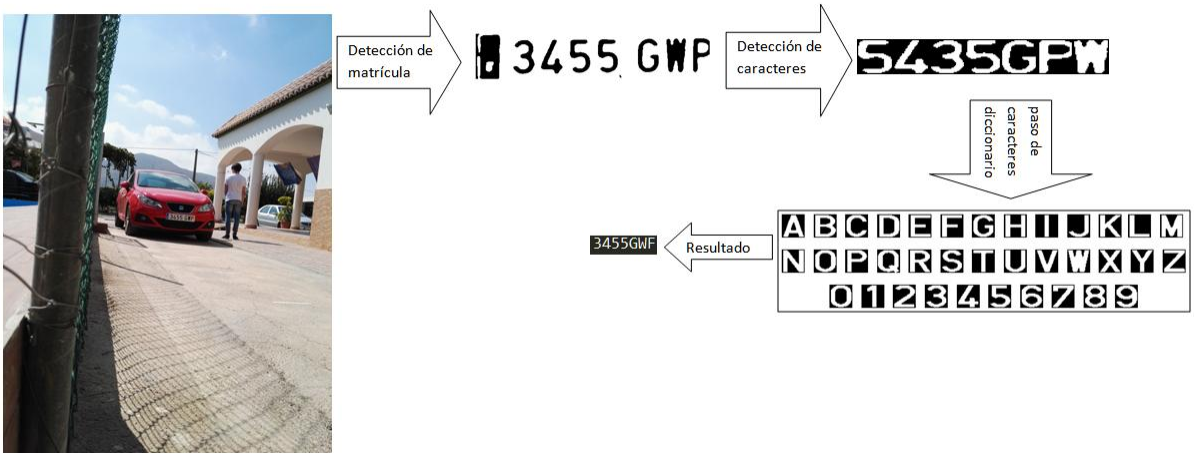

Figura 68. Seguimiento de "3455GWP" a 0.5m de altura y 6m de distancia

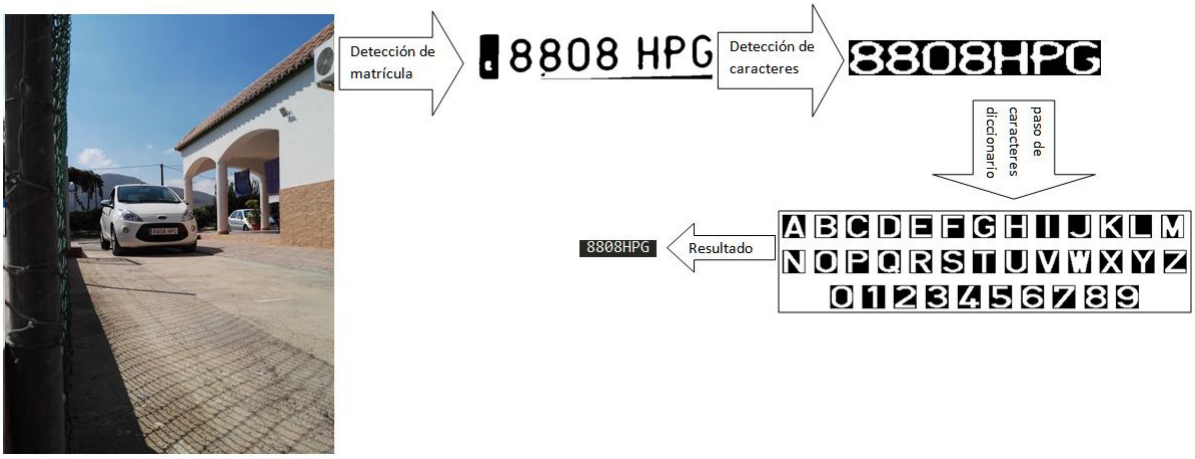

Figura 69. Seguimiento de "8808HPG" a 0.5m de altura y 6m de distancia

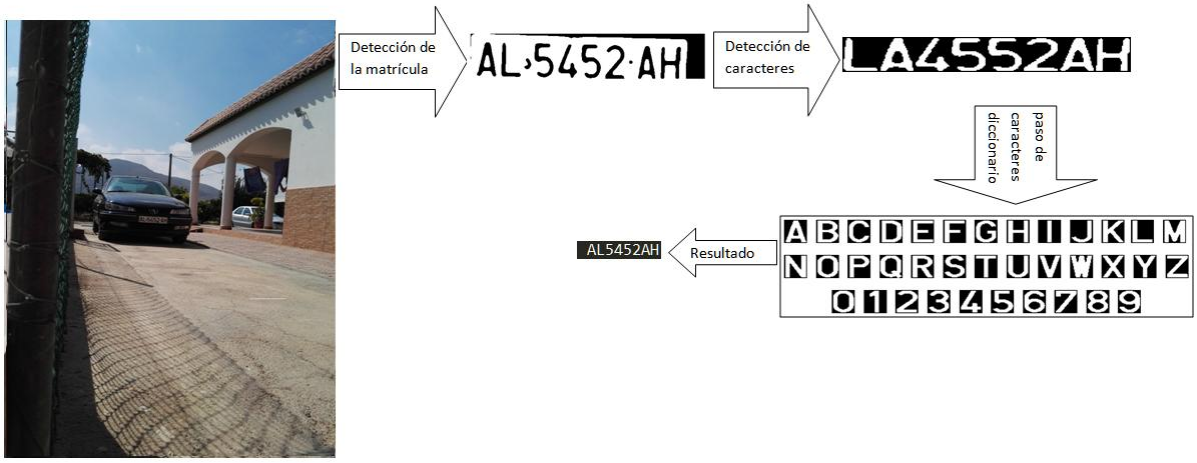

Figura 70. Seguimiento de "AL5452AH" a 0.5m de altura y 6m de distancia

5.3.3. Caso C: altura 0.5m / distancia 8m.

Para este caso, figura 71, figura 72, figura 73 y figura 74, obtenemos una tasa de acierto del 100%.

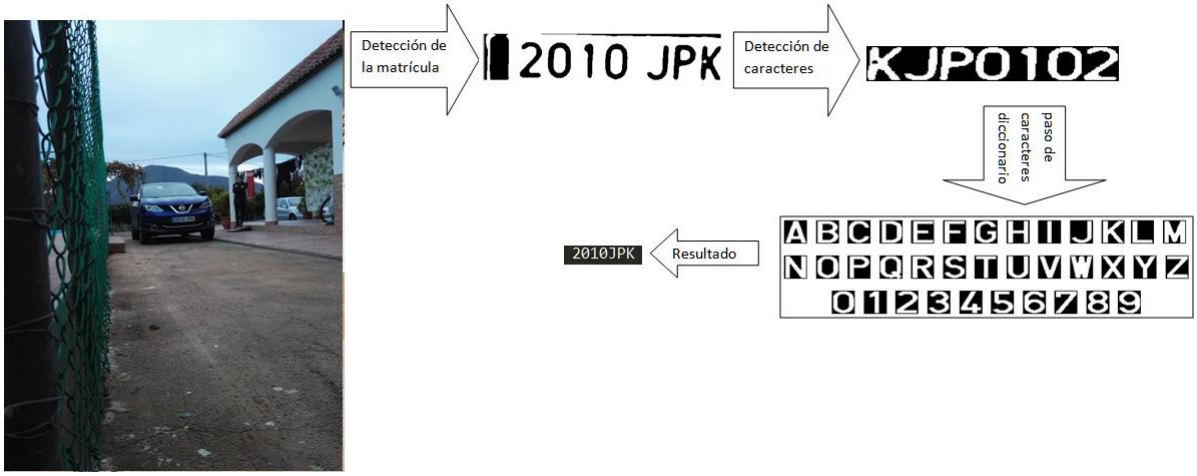

Figura 71. Seguimiento de "2010JPK" a 0.5m de altura y 8m de distancia

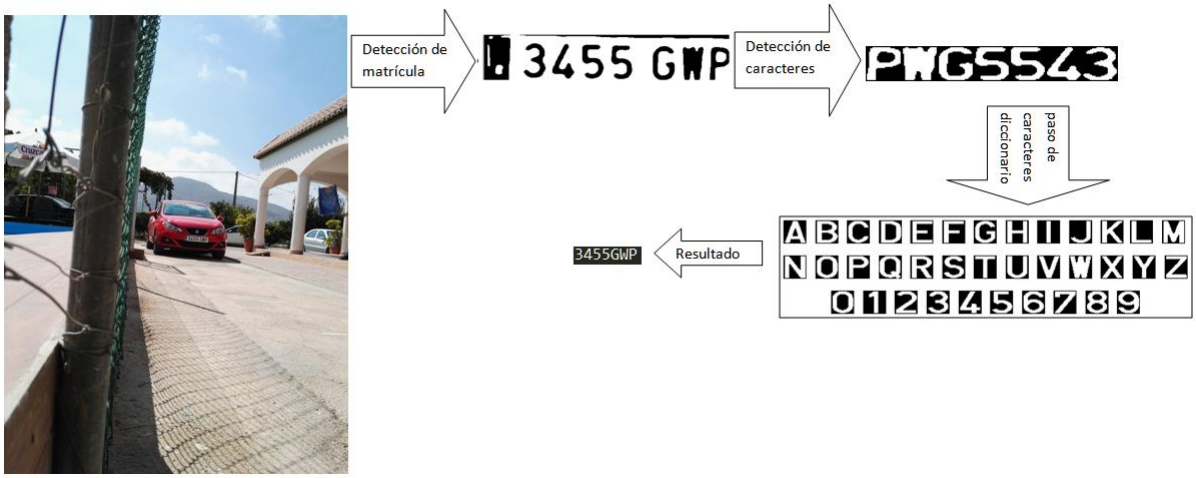

Figura 72. Seguimiento de "3455GWP" a 0.5m de altura y 8m de distancia

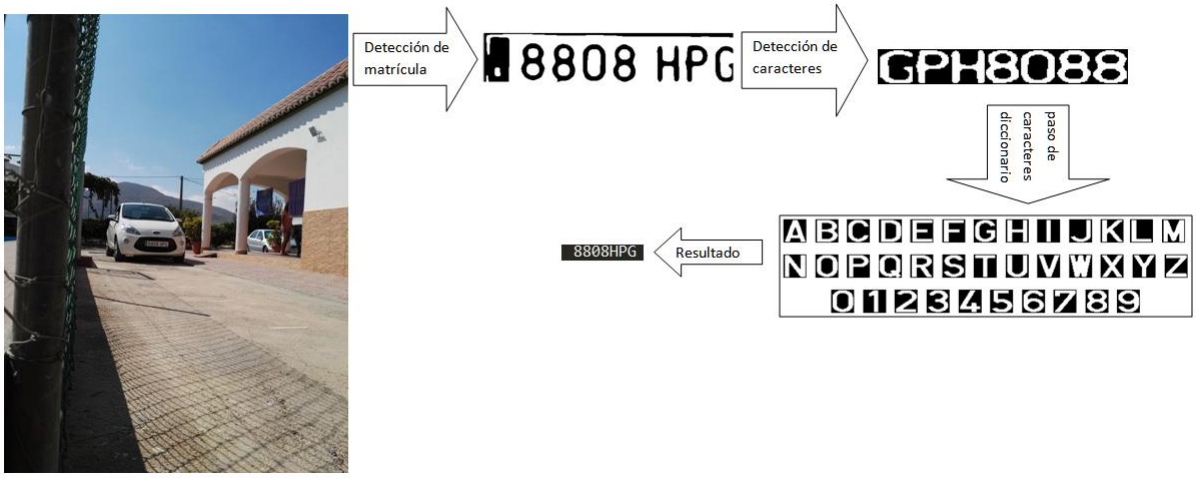

Figura 73. Seguimiento de "8808HPG" a 0.5m de altura y 8m de distancia

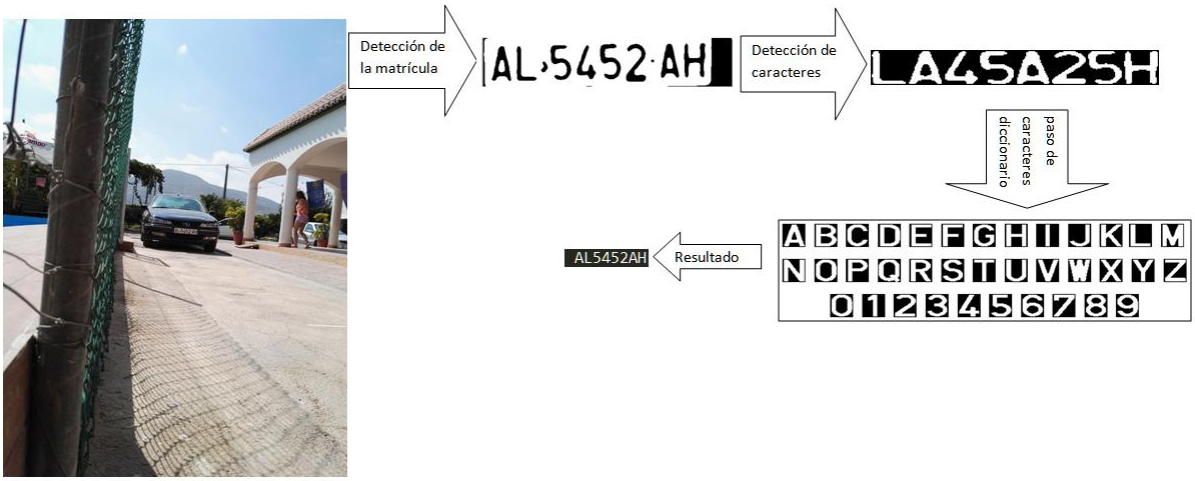

Figura 74. Seguimiento de "AL5452AH" a 0.5m de altura y 8m de distancia

5.3.4. Caso D: altura 1.5m / distancia 4m.

Para este caso, figura 75, figura 76, figura 77 y figura 78, al igual que el anterior, obtenemos una tasa de acierto del 100%.

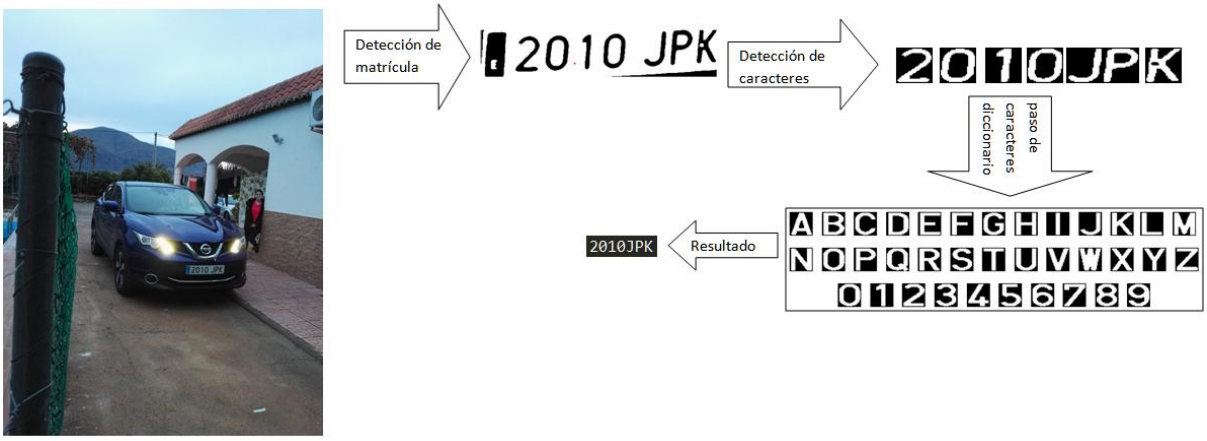

Figura 75. Seguimiento de "2010JPK" a 1.5m de altura y 4m de distancia

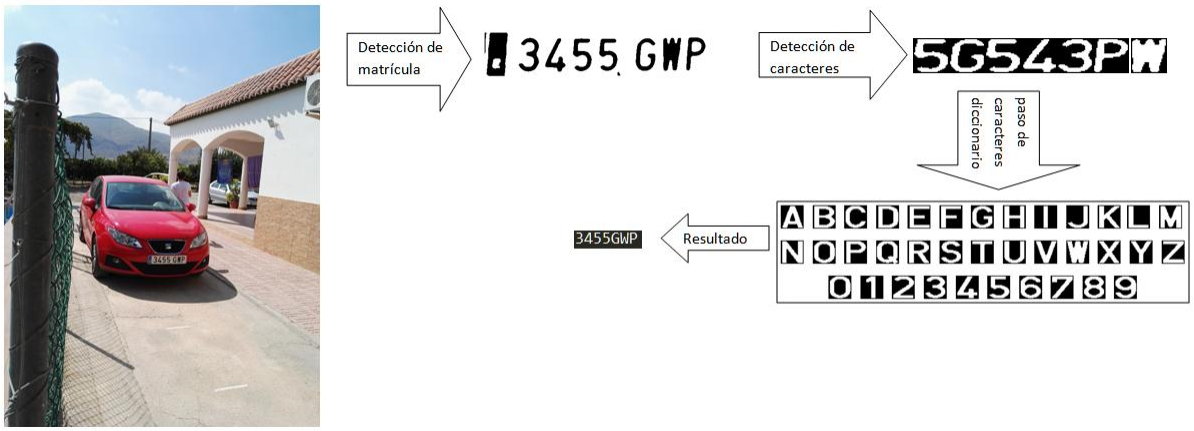

Figura 76. Seguimiento de "3455GWP" a 1.5m de altura y 4m de distancia

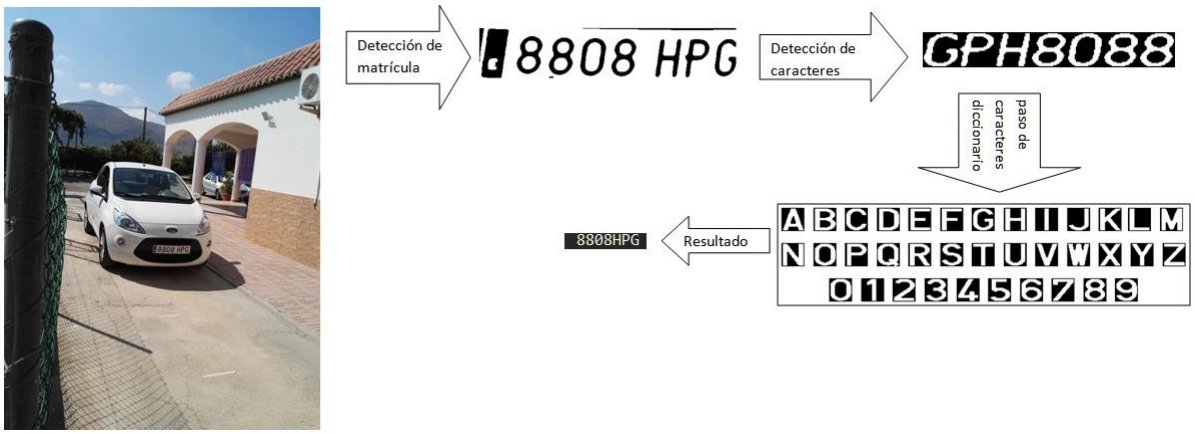

Figura 77. Seguimiento de "8808HPG" a 1.5m de altura y 4m de distancia

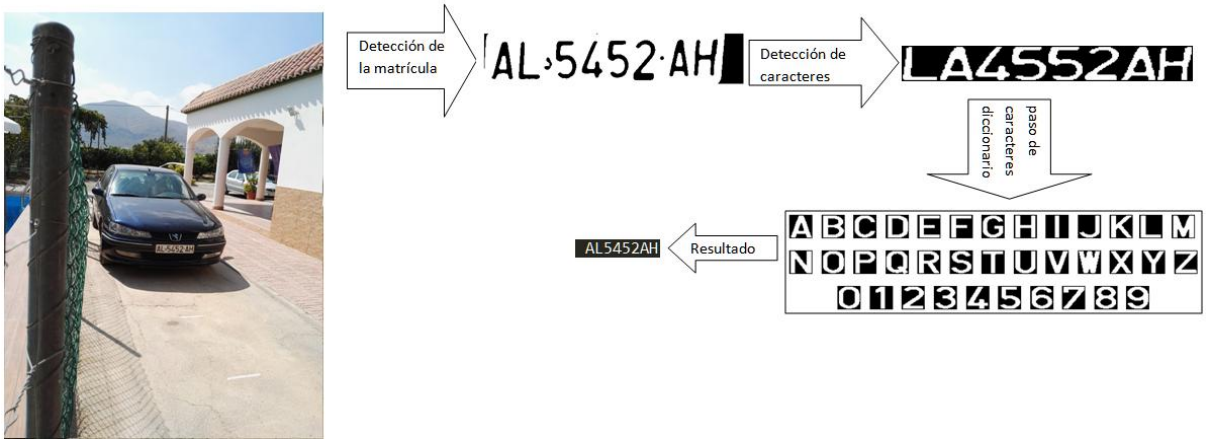

Figura 78. Seguimiento de "AL5452AH" a 1.5m de altura y 4m de distancia

5.3.5. Caso E: altura 1.5m / distancia 6m.

Para este caso, figura 79, figura 80, figura 81 y figura 82, la tasa de acierto es del 100%

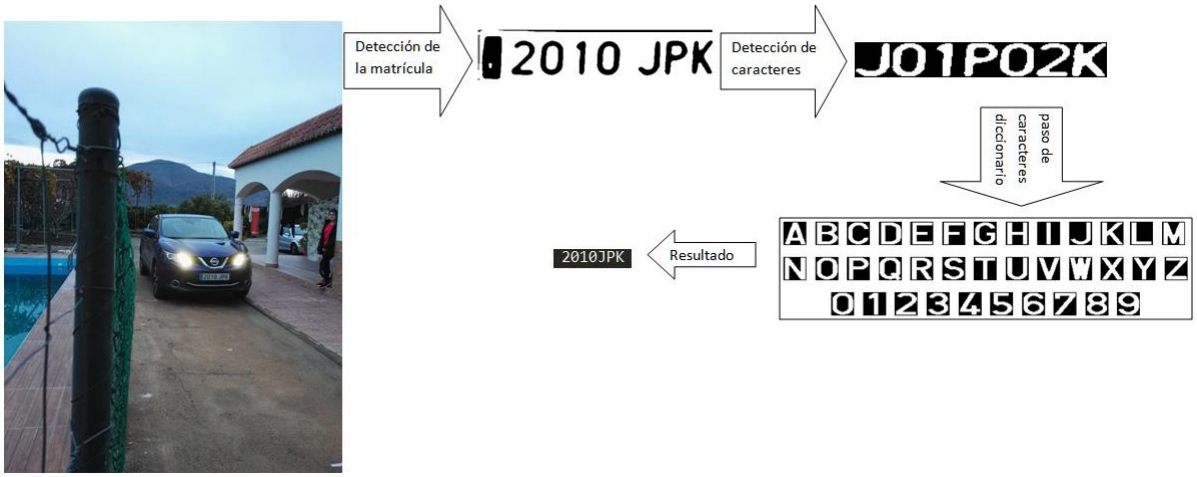

Figura 79. Seguimiento de "2010JPK" a 1.5m de altura y 6m de distancia

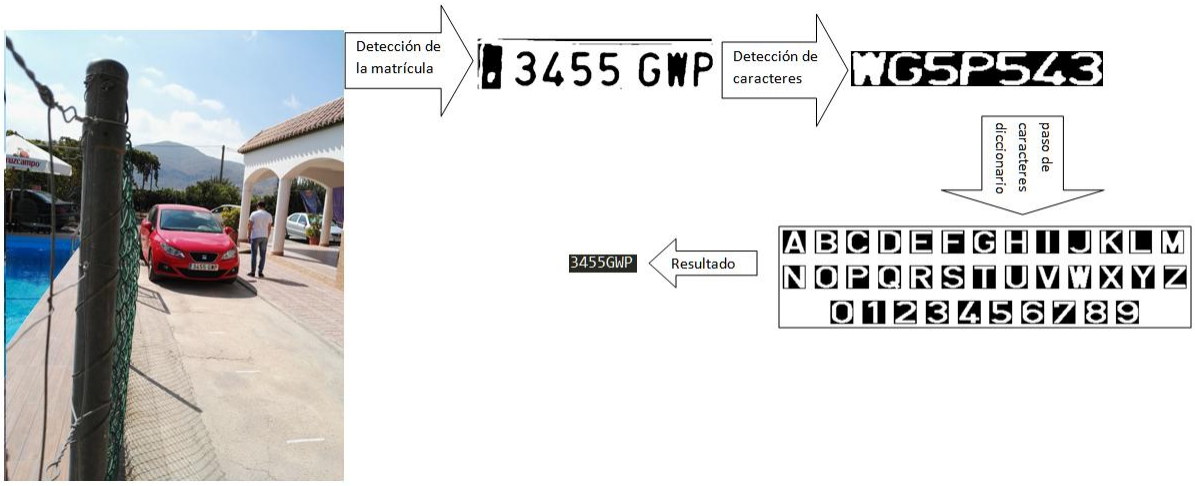

Figura 80. Seguimiento de "3455GWP" a 1.5m de altura y 6m de distancia

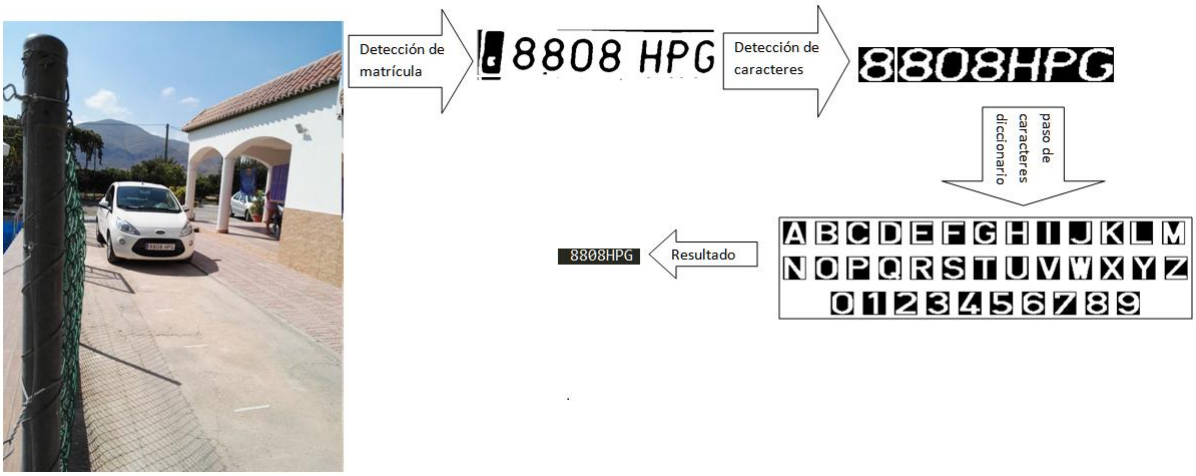

Figura 81. Seguimiento de "8808HPG" a 1.5m de altura y 6m de distancia

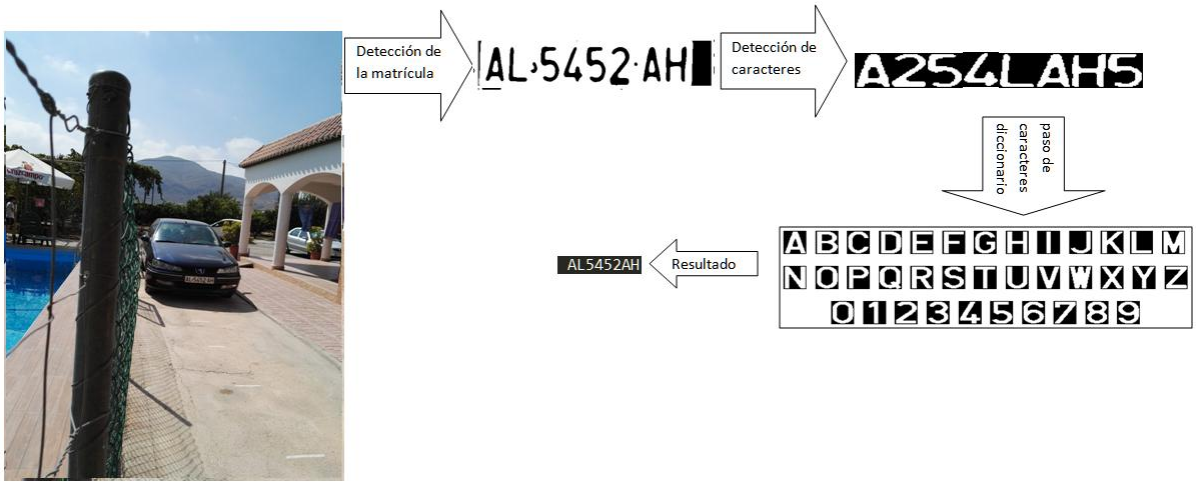

Figura 82. Seguimiento de "AL5452AH" a 1.5m de altura y 6m de distancia

5.3.6. Caso F: altura 1.5m / distancia 8m.

En esta ocasión, figura 83, figura 84, figura 85 y figura 86, se vuelve a obtener los resultados anteriores: 75% de tasa de acierto en cuanto a matrículas completamente reconocidas y un 96% de tasa de acierto en reconocimiento de caracteres individualmente.

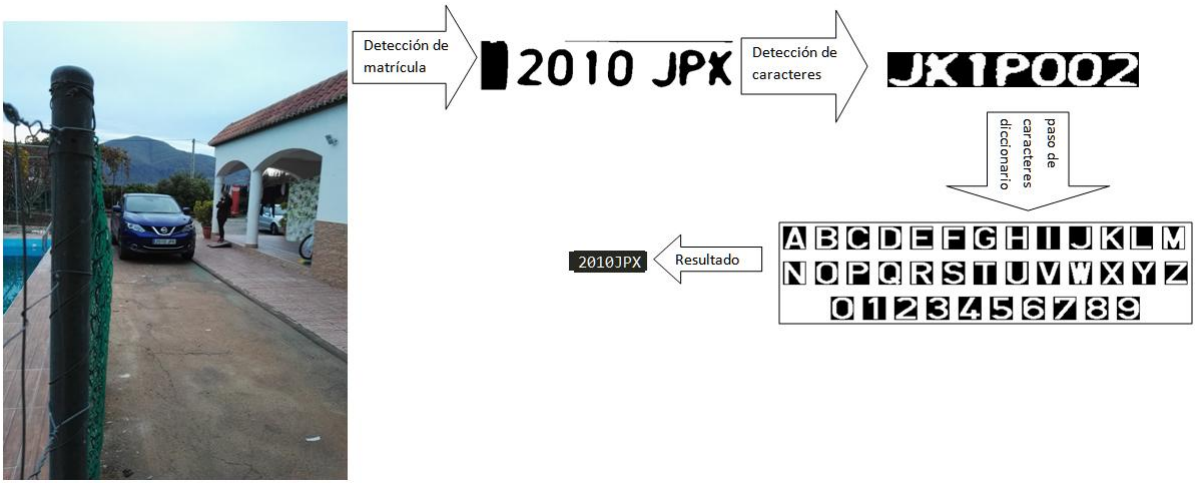

Figura 83. Seguimiento de "2010JPK" a 1.5m de altura y 8m de distancia

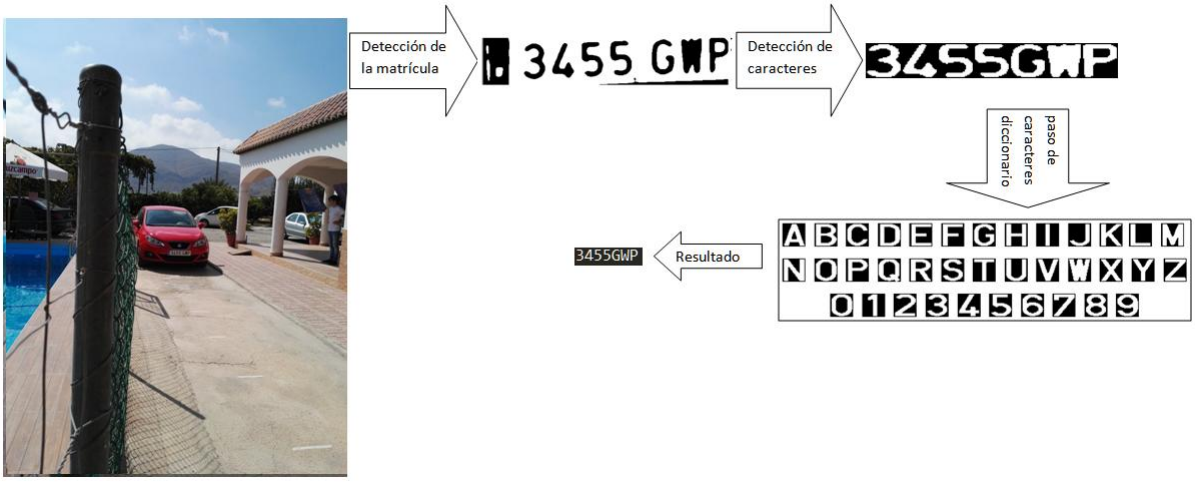

Figura 84. Seguimiento de "3455GWP" a 1.5m de altura y 8m de distancia

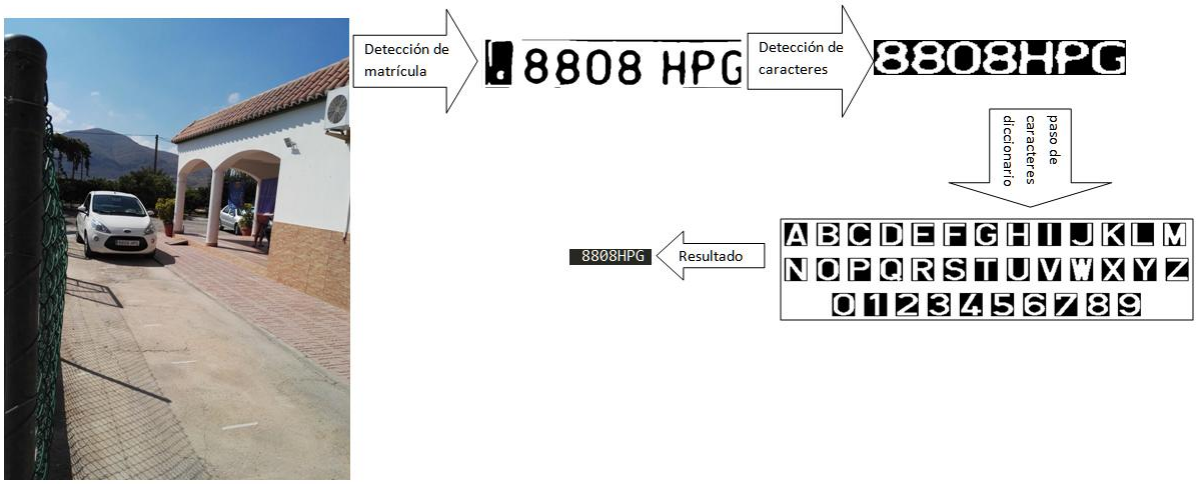

Figura 85. Seguimiento de "8808HPG" a 1.5m de altura y 8m de distancia

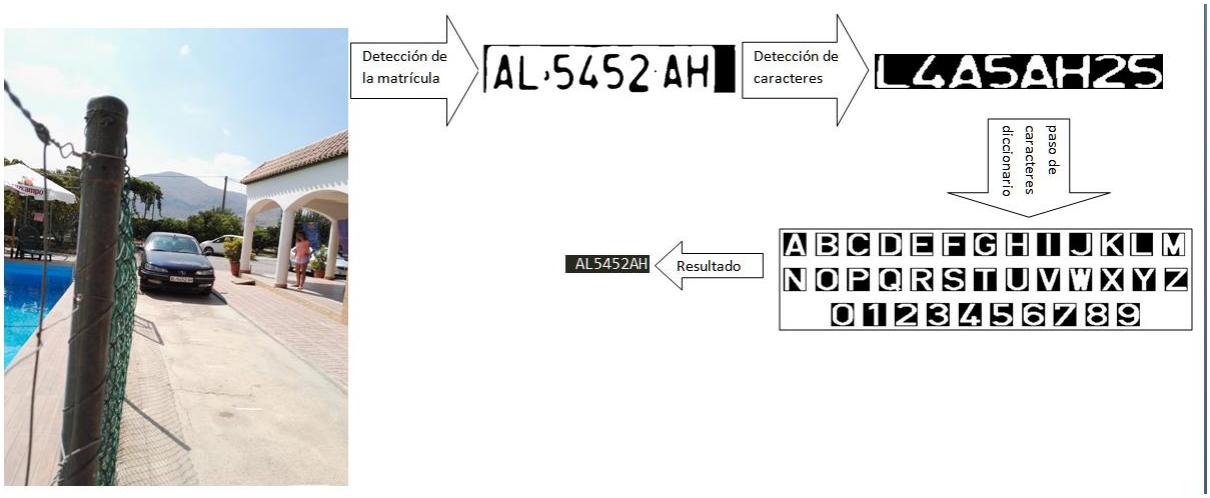

Figura 86. Seguimiento de "AL5452AH" a 1.5m de altura y 8m de distancia

5.3.7. Caso G: altura 2.5m / distancia 4m.

En este caso, figura 87, figura 88, figura 89 y figura 90, la tasa de acierto desciende a un 25% en cuanto a matrícula completamente detectada y a una tasa de acierto del 75% con respecto a caracteres reconocidos individualmente.

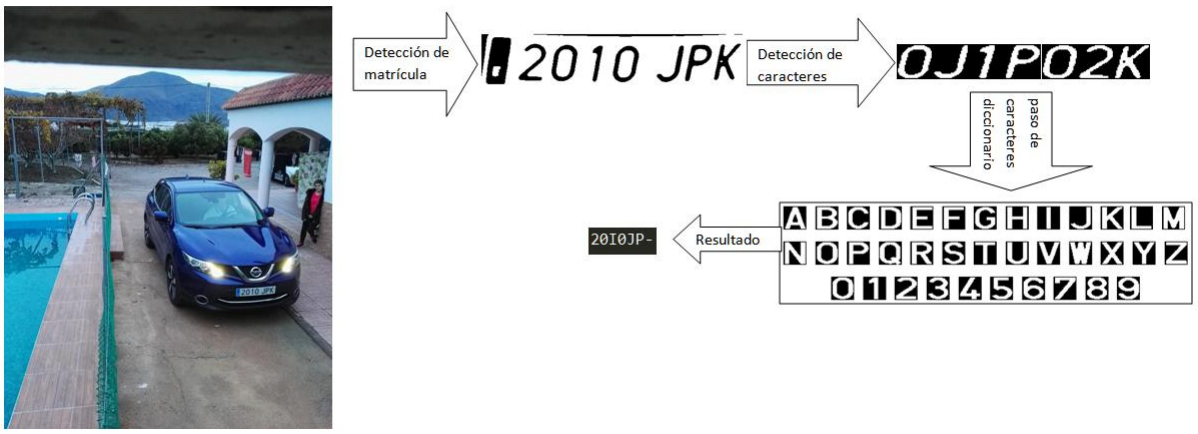

Figura 87. Seguimiento de "2010JPK" a 2.5m de altura y 4m de distancia

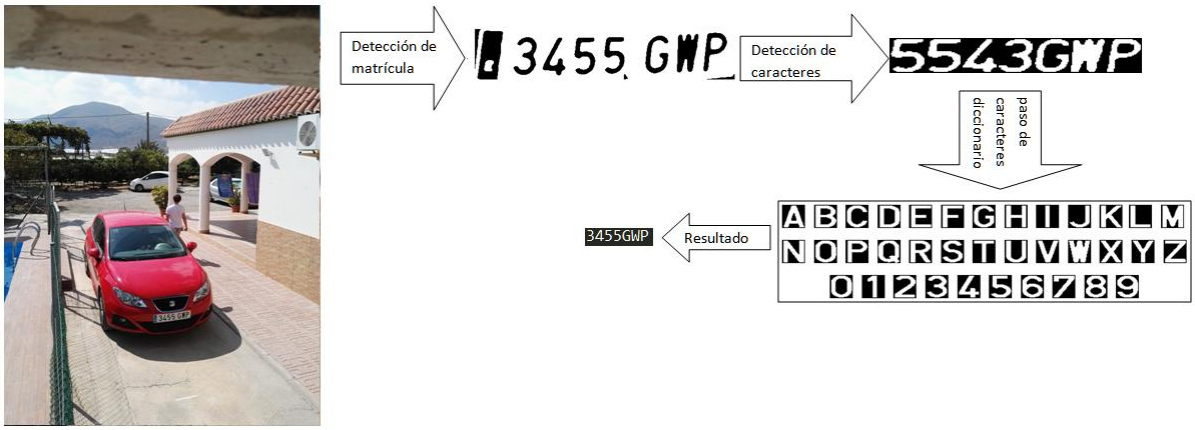

Figura 88. Seguimiento de "3455GWP" a 2.5m de altura y 4m de distancia

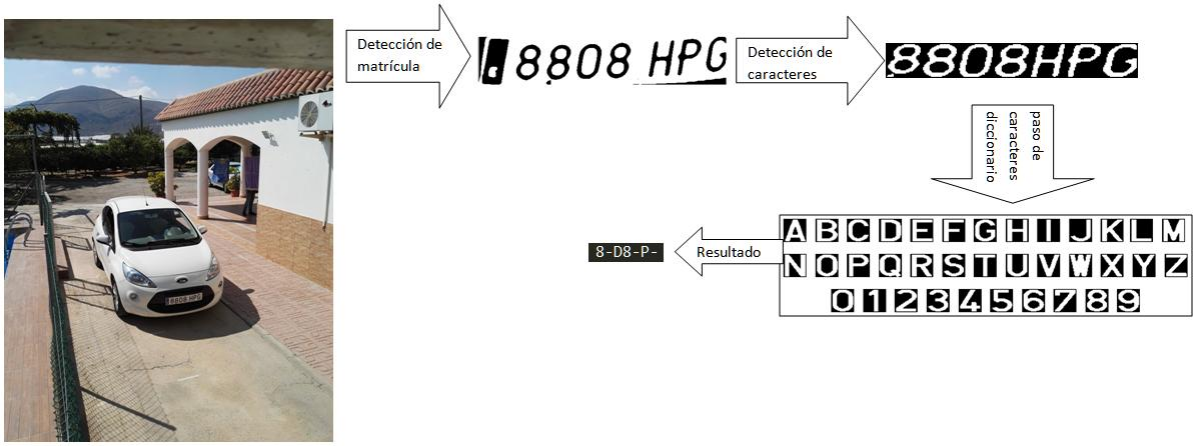

Figura 89. Seguimiento de "8808HPG" a 2.5m de altura y 4m de distancia

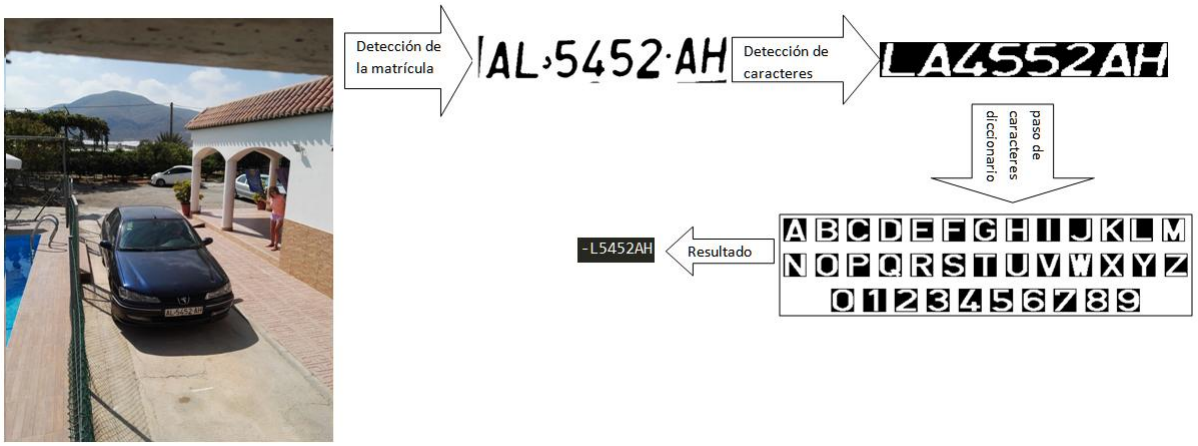

Figura 90. Seguimiento de "AL5452AH" a 2.5m de altura y 4m de distancia

5.3.8. Caso H: altura 2.5m / distancia 6m.

Para este caso, figura 91, figura 92, figura 93 y figura 94, volvemos a tener un 75% de tasa de acierto en cuanto a matrícula completamente reconocida y un 96% de tasa de acierto en cuanto a reconocimiento individual de caracteres.

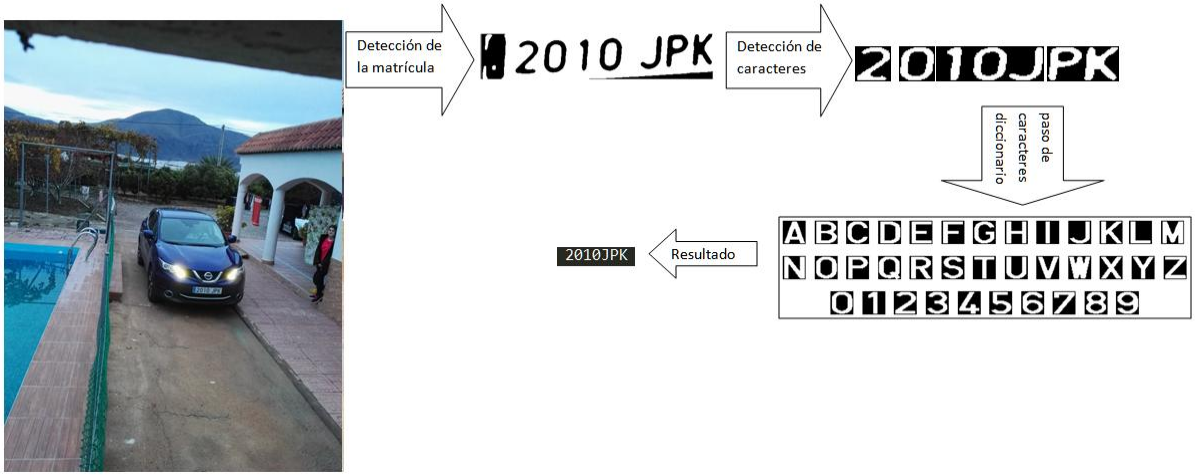

Figura 91. Seguimiento de "2010JPK" a 2.5m de altura y 6m de distancia

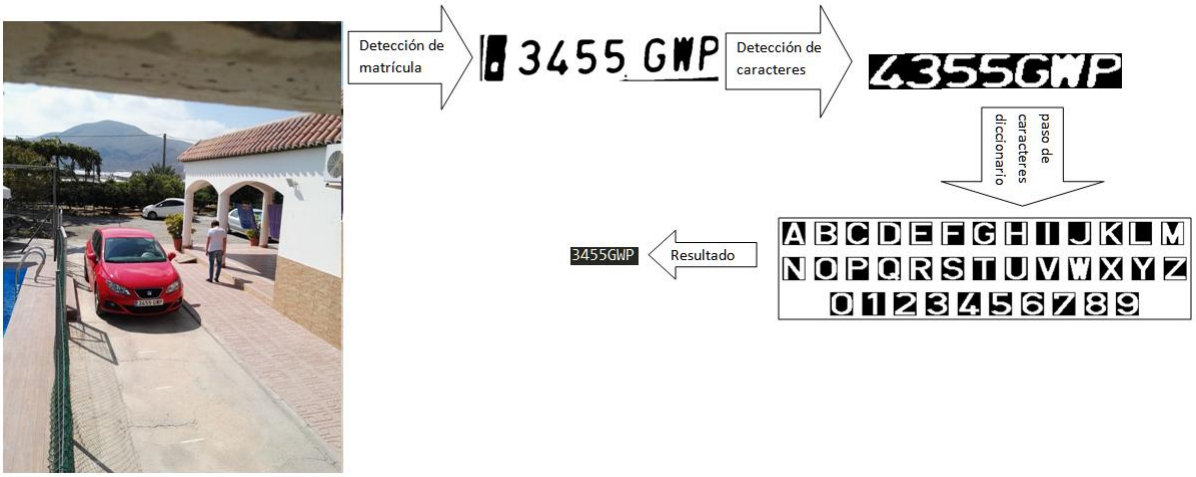

Figura 92. Seguimiento de "3455GWP" a 2.5m de altura y 6m de distancia

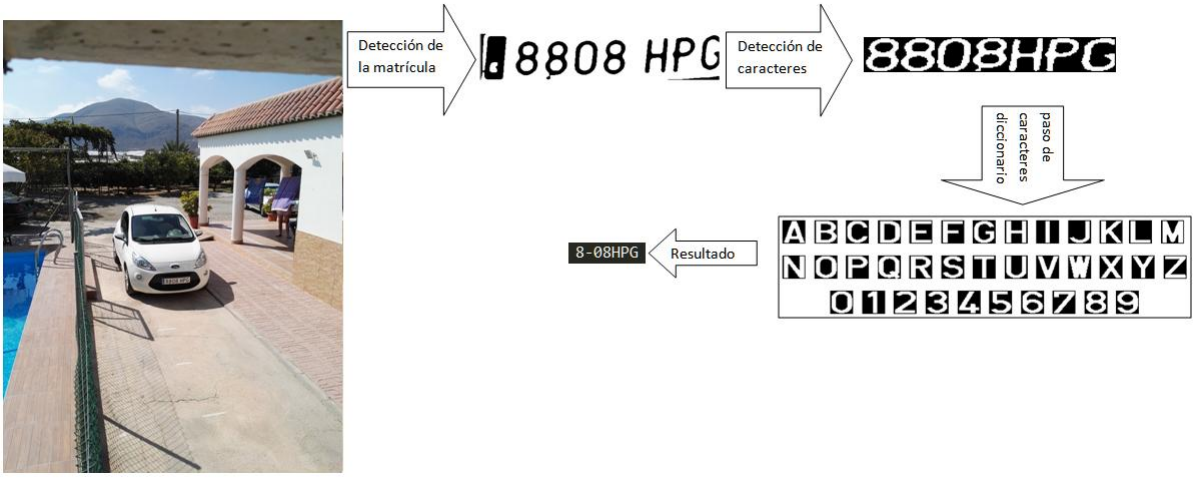

Figura 93. Seguimiento de "8808HPG" a 2.5m de altura y 6m de distancia

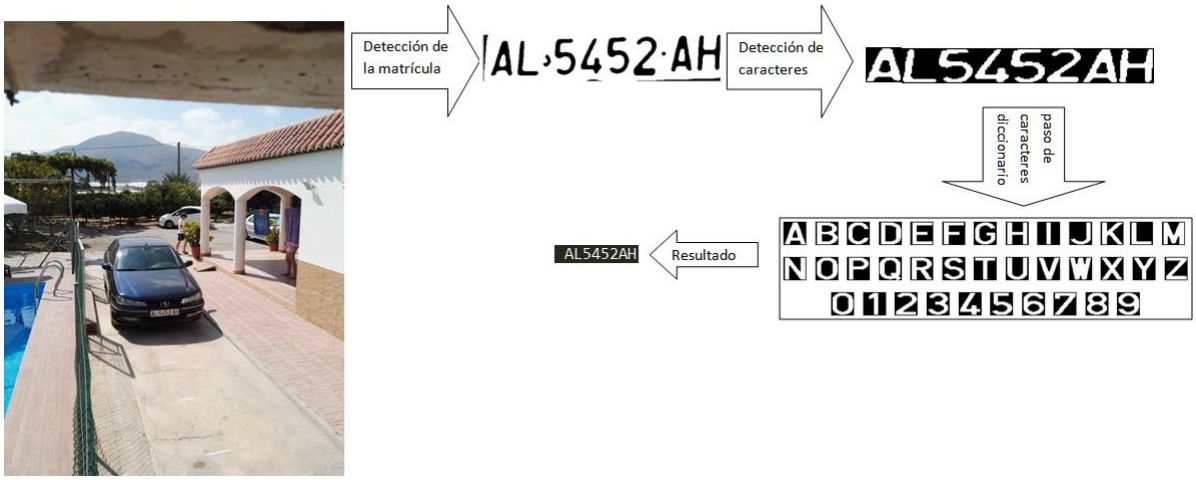

Figura 94. Seguimiento de "AL5452AH" a 2.5m de altura y 6m de distancia

5.3.9. Caso I: altura 2.5m / distancia 8m.

Por último, para este caso, figura 95, figura 96, figura 97 y figura 98, obtenemos las siguientes tasas de acierto: un 25% en cuanto a matrícula completamente reconocida y un 89% en cuanto a reconocimiento de caracteres individualmente.

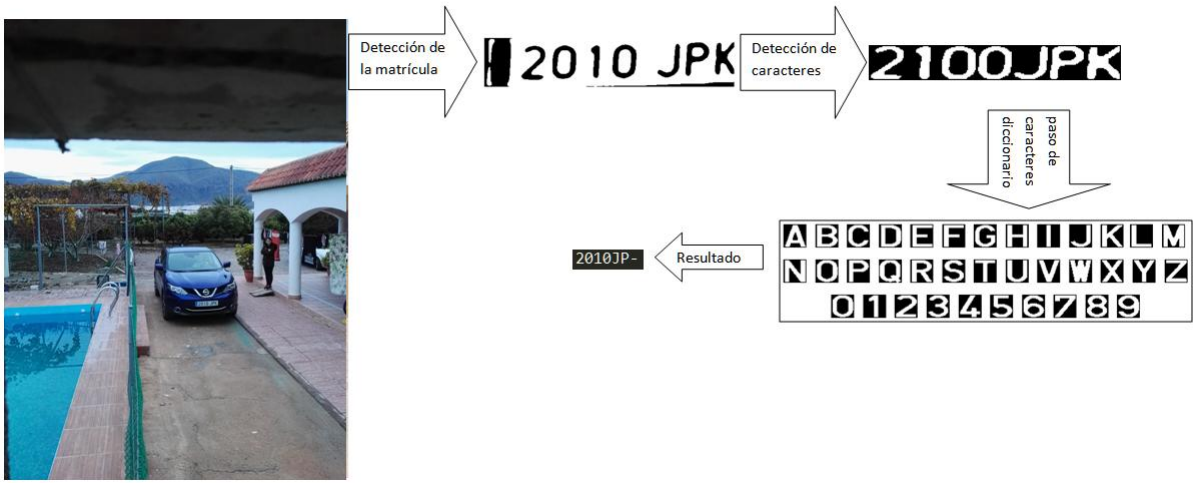

Figura 95. Seguimiento de "2010JPK" a 2.5m de altura y 8m de distancia

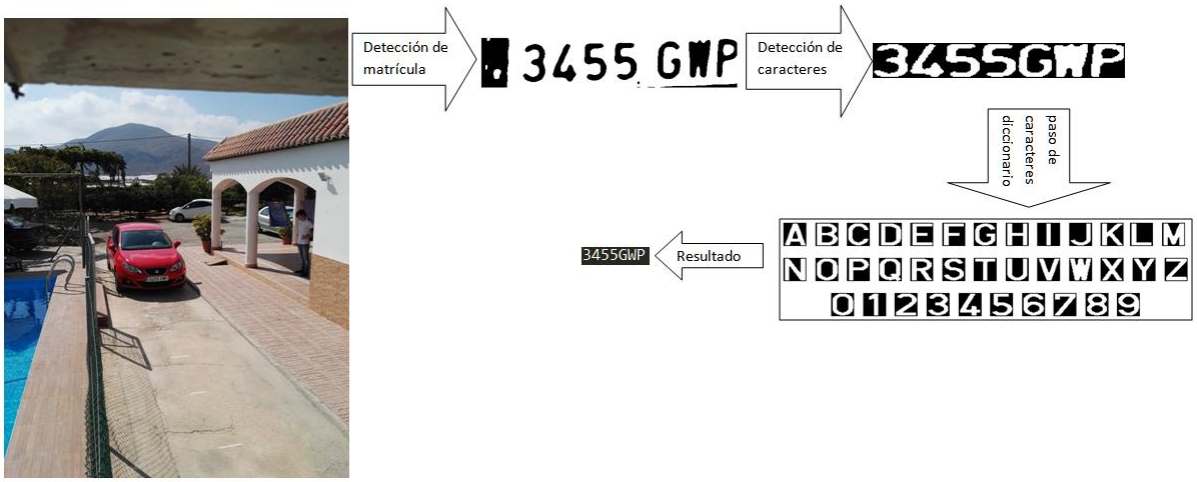

Figura 96. Seguimiento de "3455GWP" a 2.5m de altura y 8m de distancia

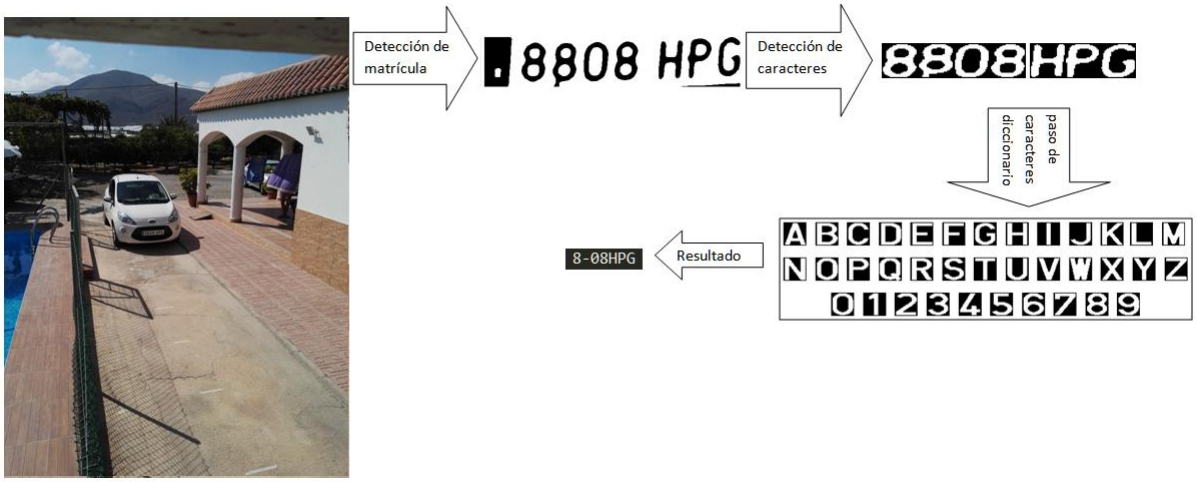

Figura 97. Seguimiento de "8808HPG" a 2.5m de altura y 8m de distancia

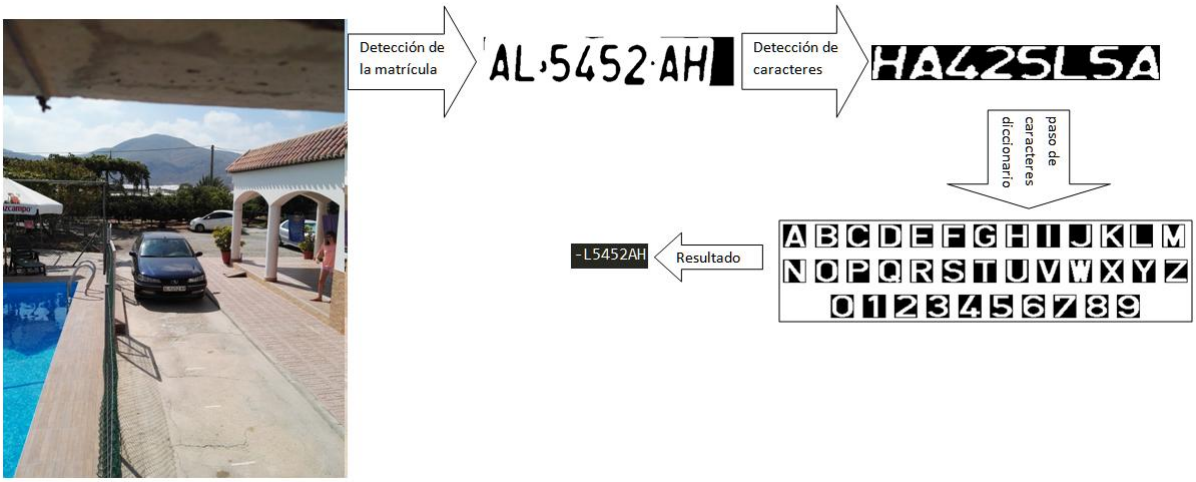

Figura 98. Seguimiento de "AL5455AH" a 2.5m de altura y 8m de distancia

5.4 Resumen de los resultados obtenidos.

En la tabla 2, podemos comparar todos los casos vistos anteriormente y sus resultados:

Altura (m)	Distancia (m)	2010 JPK	3455 GWP	8808 HPG	AL 5452 AH
0.5	4	2010JPK	3455GWP	8808HP-	AL5452AH
	6	2010JPK	3455GWF	8808HPG	AL5452AH
	8	2010JPK	3455GWP	8808HPG	AL5452AH
1.5	4	2010JPK	3455GWP	8808HPG	AL5452AH
	6	2010JPK	3455GWP	8808HPG	AL5452AH
	8	2010JPX	3455GWP	8808HPG	AL5452AH
2.5	4	20I0JP-	3455GWP	8-D8-P-	-L5452AH
	6	2010JPK	3455GWP	8-08HPG	AL5452AH
	8	2010JP-	3455GWP	8-08HPG	-L5452AH

Tabla 2. Resultados obtenidos en 4, 6 y 8 metros

A continuación, se muestra la tabla 3, con las tasas de acierto obtenidas en cada caso, por matrículas completas y por caracteres totales reconocidos.

Altura (m)	Distancia (m)	Tasas de acierto	
		Por matrículas	Por caracteres
0.5	4	75%	96,55%
	6	75%	96,55%
	8	100%	100%
1.5	4	100%	100%
	6	100%	100%
	8	75%	96,55%
2.5	4	25%	75,86%
	6	75%	96,55%
	8	25%	89,65%

Tabla 3. Tasa de acierto

5.5 Tiempos de procesamiento

Otro aspecto importante, además de la tasa de acierto, son los tiempos de procesamiento. Ya que nuestra aplicación está orientada al procesamiento continuo de imágenes, por lo que la velocidad de procesado debe de ser alta para que puede procesar varias imágenes de un mismo vehículo y de esta manera comparar los resultados obtenidos en un corto plazo de tiempo para verificar el reconocimiento.

Las imágenes anteriormente mostradas en el estudio, tiene un tamaño de 8 Megapíxeles (3264 x 2448). Además, nuestra aplicación, previamente al procesamiento principal, realiza un escalado para reducir a la mitad el tamaño de esta, lo que nos quedaría una imagen con el siguiente tamaño: 1632 x 1224. Este proceso de escalado también influye en los tiempos finales de procesamiento.

Aunque nuestro sistema está orientado para que sea ejecutado en una *Raspberry Pi*, hemos lanzado el código en distintos equipos para comparar los tiempos de procesamiento.

5.5.1. Equipo A

Nuestro primer equipo donde hemos realizado pruebas, consta de las siguientes características más relevantes para el procesamiento:

- CPU: Intel Core i5-4460, 3.20 GHz
- Memoria RAM: 8 GB
- SO: Windows 7 Professional

En este equipo obtenemos la siguiente tabla de tiempos de procesamiento, tabla 4, dada en segundos:

Altura (m)	Distancia (m)	2010 JPK	3455 GWP	8808 HPG	AL 5452 AH
0.5	4	0.4159	0.4269	0.3619	0.3919
	6	0.3740	0.4069	0.3660	0.4900
	8	0.4850	0.4149	0.4129	0.4400
1.5	4	0.4670	0.4379	0.3949	0.4320
	6	0.4749	0.4530	0.3649	0.4749
	8	0.4309	0.4579	0.4819	0.3709
2.5	4	0.4170	0.4019	0.3350	0.3939
	6	0.3789	0.3519	0.4250	0.3980
	8	0.5420	0.3739	0.5329	0.3889

Tabla 4. Tiempos en Equipo A dada en segundos

En este equipo obtenemos un tiempo medio de procesamiento de 0.4185 seg.

5.5.2. Equipo B

Nuestro segundo equipo donde hemos realizado pruebas, consta de las siguientes características más relevantes para el procesamiento:

- CPU: Intel Core i7, 2.30 GHz
- Memoria RAM: 8 GB
- SO: MAC OS X 10.11.4

En este equipo obtenemos la siguiente tabla de tiempos de procesamiento, tabla 5, dada en segundos:

Altura (m)	Distancia (m)	2010 JPK	3455 GWP	8808 HPG	AL 5452 AH
0.5	4	0.3920	0.3828	0.3568	0.3093
	6	0.2989	0.3801	0.3405	0.4543
	8	0.4822	0.4051	0.3441	0.3609
1.5	4	0.4925	0.3842	0.3865	0.3636
	6	0.4763	0.4353	0.3437	0.4748
	8	0.3623	0.4314	0.4925	0.2914
2.5	4	0.3683	0.3370	0.2681	0.3096
	6	0.3458	0.3071	0.3726	0.3142
	8	0.5830	0.2654	0.5694	0.3132

Tabla 5. Tiempos en Equipo B dada en segundos

En este equipo obtenemos un tiempo medio de procesamiento de 0.3758 seg.

5.5.3 Equipo C

Nuestro tercer equipo se trata de una *Raspberry Pi 2 Model B* con las siguientes características:

- CPU: ARM Quad-Core Cortex A7, 900MHz
- Memoria RAM: 1 GB
- SO: Raspbian

En este equipo obtenemos la siguiente tablas de tiempos de procesamiento, tabla 6, dada en segundos:

Altura (m)	Distancia (m)	2010 JPK	3455 GWP	8808 HPG	AL 5452 AH
0.5	4	2.8949	3.0531	2.8334	2.9307
	6	2.5177	3.1270	2.8655	3.6391
	8	4.0115	3.2726	2.8075	2.9302
1.5	4	3.9341	3.1285	3.0763	2.9892
	6	3.7253	3.5050	2.6090	3.6898
	8	2.9961	3.5341	3.9823	2.4464
2.5	4	3.0162	2.7595	2.1658	2.6342
	6	2.7934	2.5540	3.0193	2.5494
	8	4.8108	2.1776	4.6640	2.5738

Tabla 6. Tiempos en Equipo C dada en segundos

En este equipo obtenemos un tiempo medio de procesamiento de 3.0619 seg.

Capítulo 6
Prueba con procesamiento sobre vídeo

6. Prueba con procesamiento sobre vídeo

En este punto vamos a comentar una pequeña prueba del sistema desarrollado procesando video en tiempo real. Esta prueba se ha realizado sobre el Equipo B, detallado en el capítulo anterior, a medio día, pero con iluminación algo más baja de lo normal ya que el día se encontraba nublado.

Como se puede observar en la figura 99, estamos simulando una calle donde se encuentra nuestro sistema instalado. En este instante el vehículo se encuentra, aproximadamente, a más de 10 metros de distancia del punto donde se encuentra la cámara.

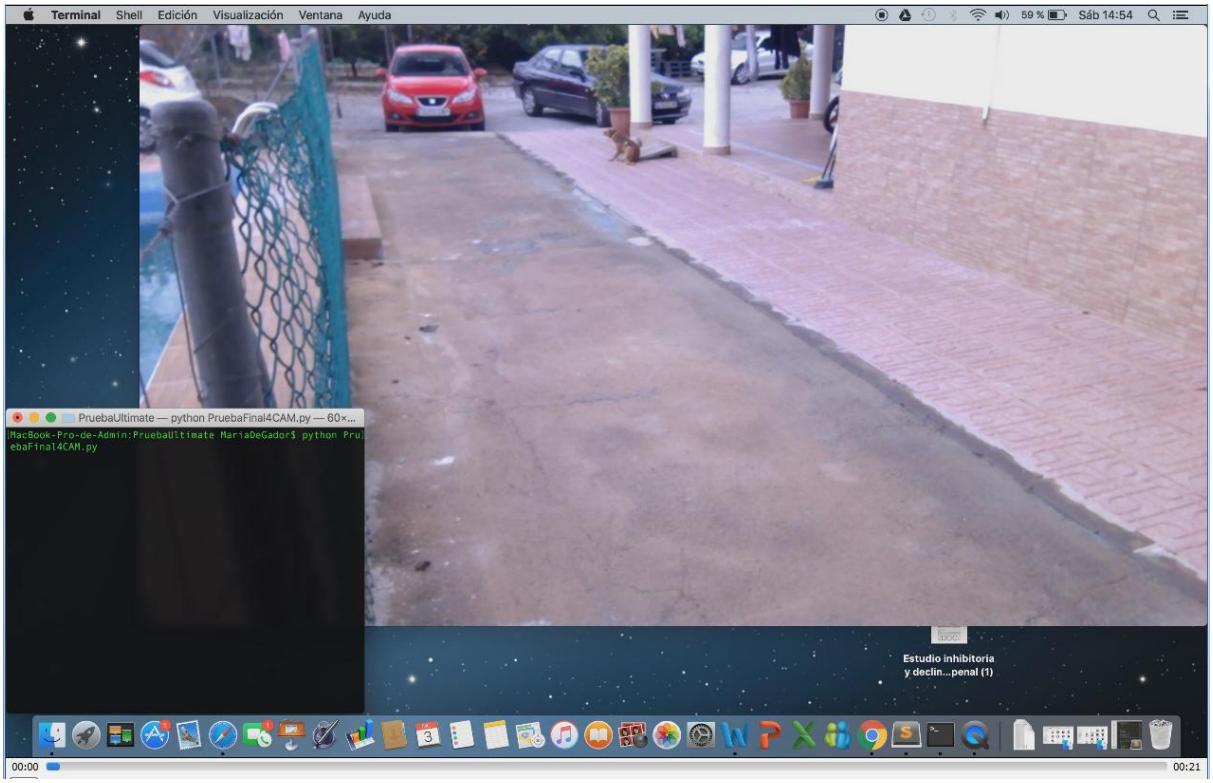

Figura 99. Captura de vídeo, segundo 0

El sistema nos da el primer dato de reconocimiento, como se ve en la figura 100, cuando el vehículo se encuentra, aproximadamente, a unos 8 metros de distancian y nos muestra el siguiente resultado: *355GF*.

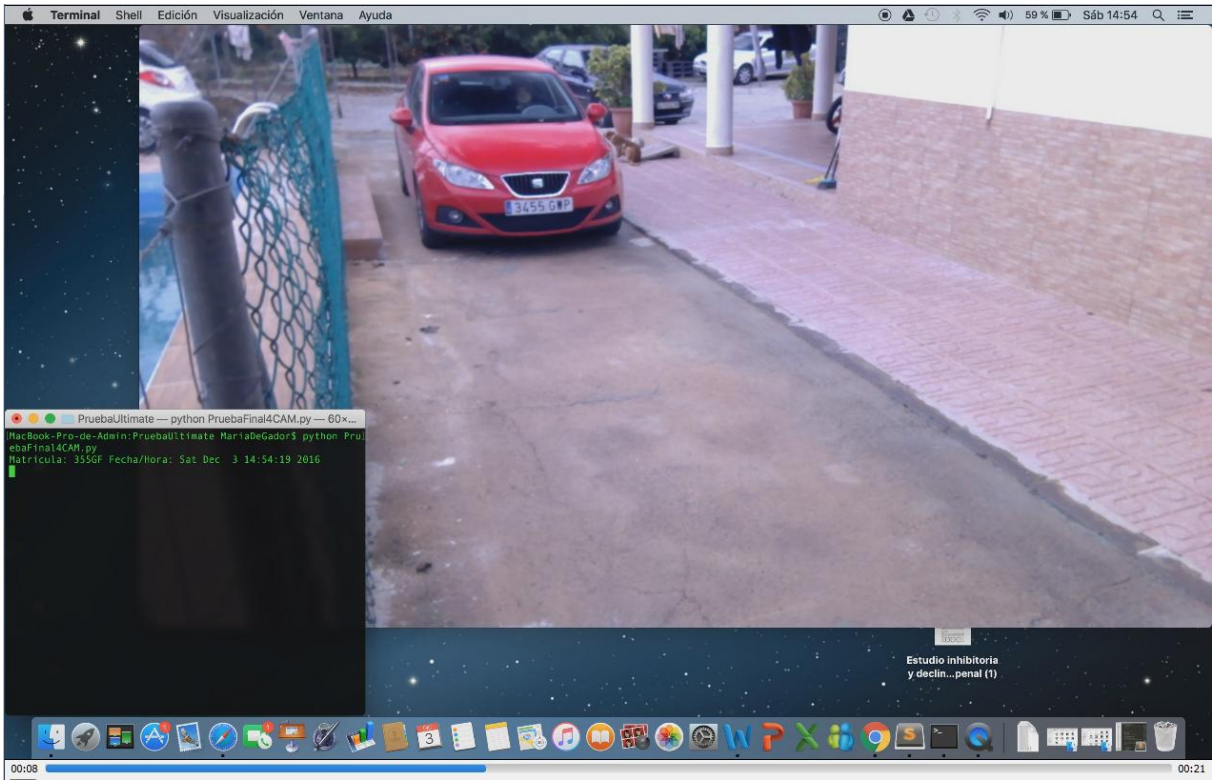

Figura 100. Captura de vídeo, segundo 8

En un principio, puede parecer que el sistema no funciona correctamente y que le falta mejora, pero como se puede ver en la figura 101, en un mismo segundo a medida que avanza el vehículo, obtenemos 8 resultados más:

- 3455G-P
- 3455GWP
- 3455GWP
- 3455GF
- 3455WF
- 3455G
- 3455G-P

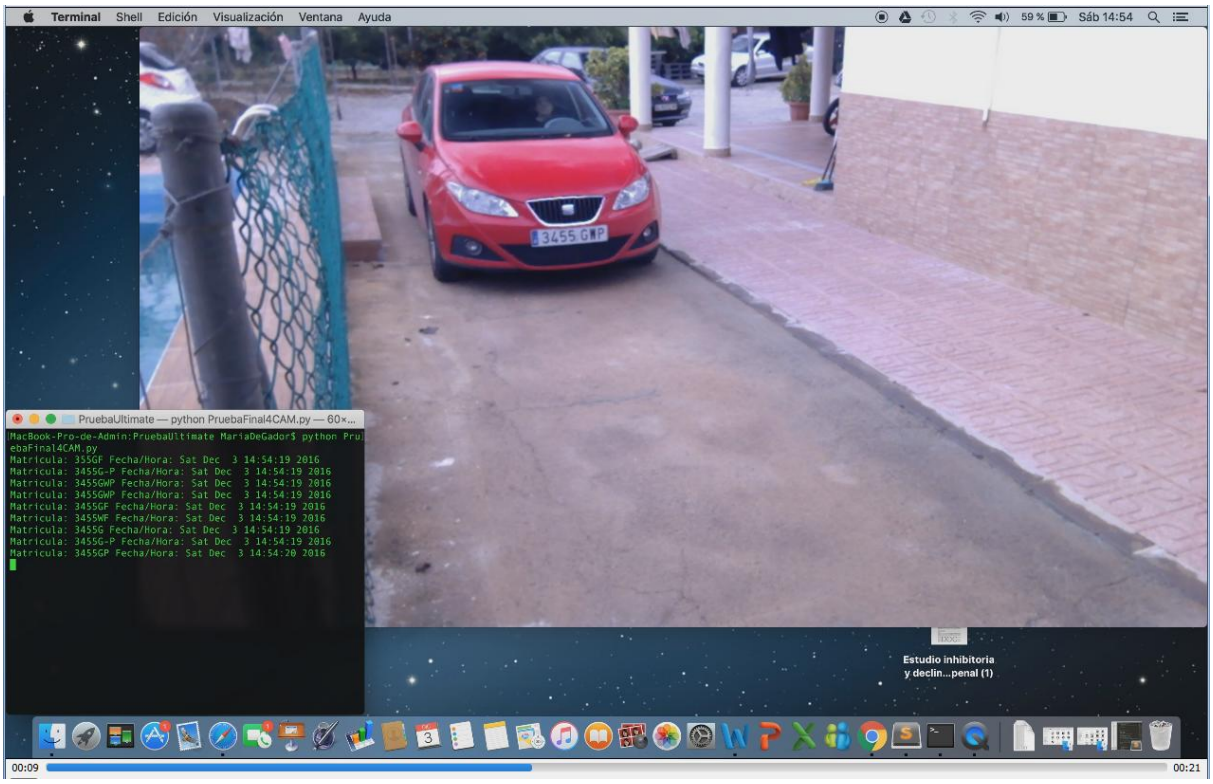

Figura 101. Captura vídeo, segundo 9

Podemos realizar un seguimiento para comprobar que caracteres se repiten más veces en una determinada posición y determinar, de esa manera, cual es la matrícula real del vehículo detectado.

En la figura 102, se muestra el siguiente segundo del vídeo. En este caso llega a reconocer hasta 12 veces en el mismo segundo con los siguientes resultados:

- 3455GP
- 3455GWP
- 3455GWF
- 3455WF
- 3455GWP
- 3455G-F
- 3455GWP
- 3455GWF
- 3455GWF
- 3455GF
- 3455GWF
- 3455GWF

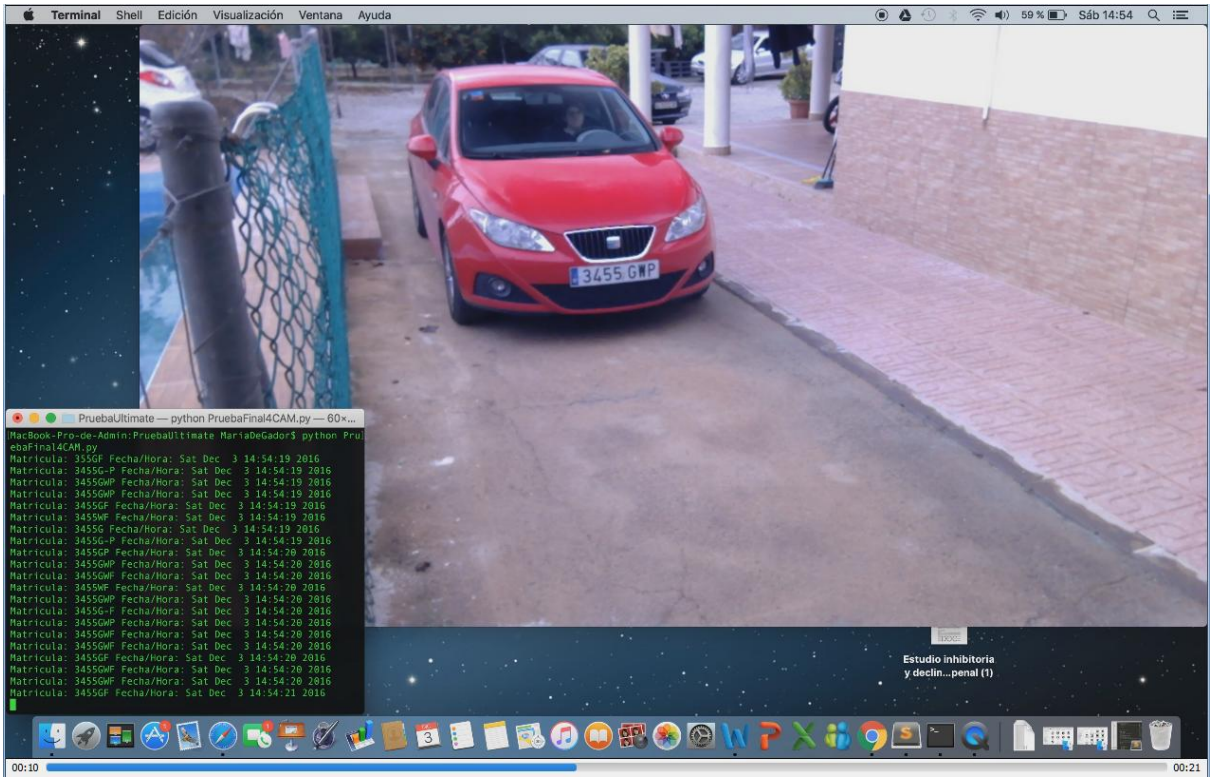

Figura 102. Captura vídeo, segundo 10

En la figura 103, el vehículo ya se encuentra, aproximadamente a unos 4 metros del punto de captura, el sistema nos devuelve los siguiente resultados en este segundo de avance:

- 3455GF
- 3455GWP
- 3455GWP
- 3455GWP
- 3455GWP
- 3455GWP
- 3455GWF
- 3455GWF
- 3455GWF
- 3455GWP
- 3455GWP
- 3455G-F
- 3455GWF
- 3455GW

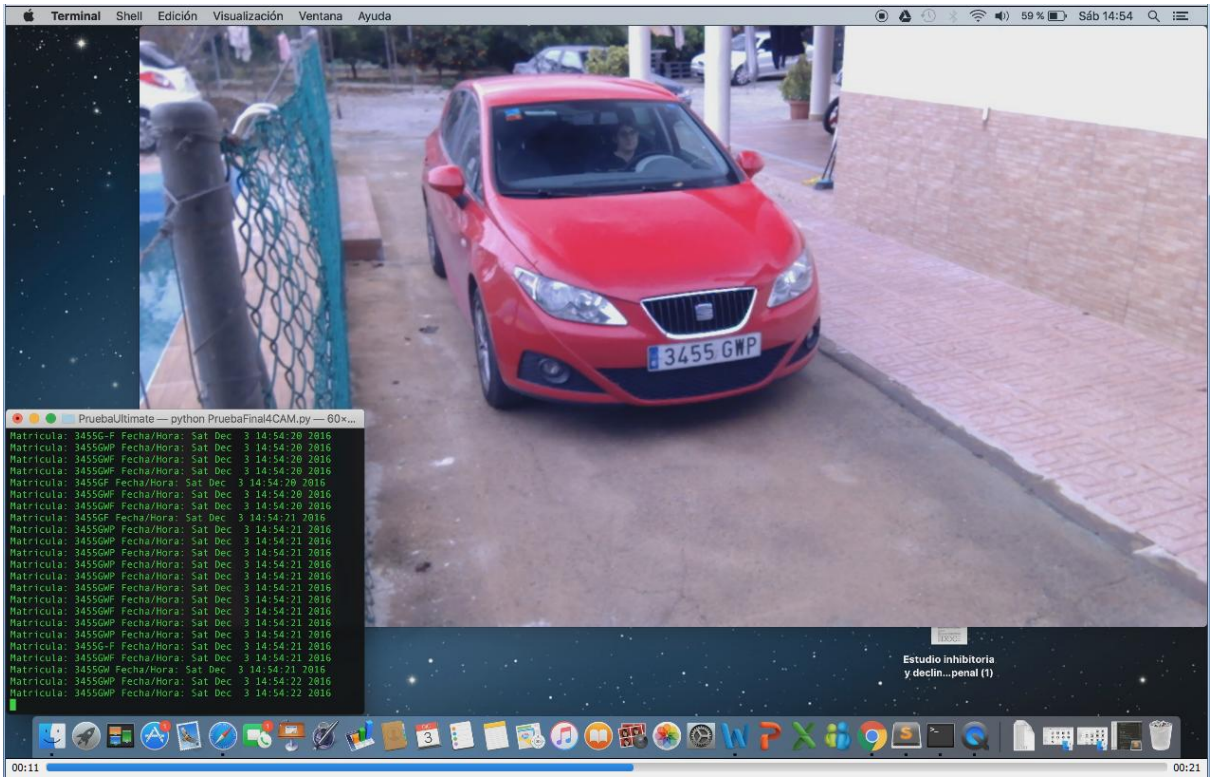

Figura 103. Captura vídeo, segundo 11

En este último segundo, figura 104, el vehículo ya se encuentra a aproximadamente menos de 4 metros, y nos devuelve los siguientes resultados:

- 3455GWP
- 3455GWP
- 3455GWP
- 3455GWP
- 3455GWP
- 3455G-P
- 3455GWP
- 3455GWP
- 3455G-P
- 3455G-P
- 3455G-P
- 3455G-P
- 3-55---

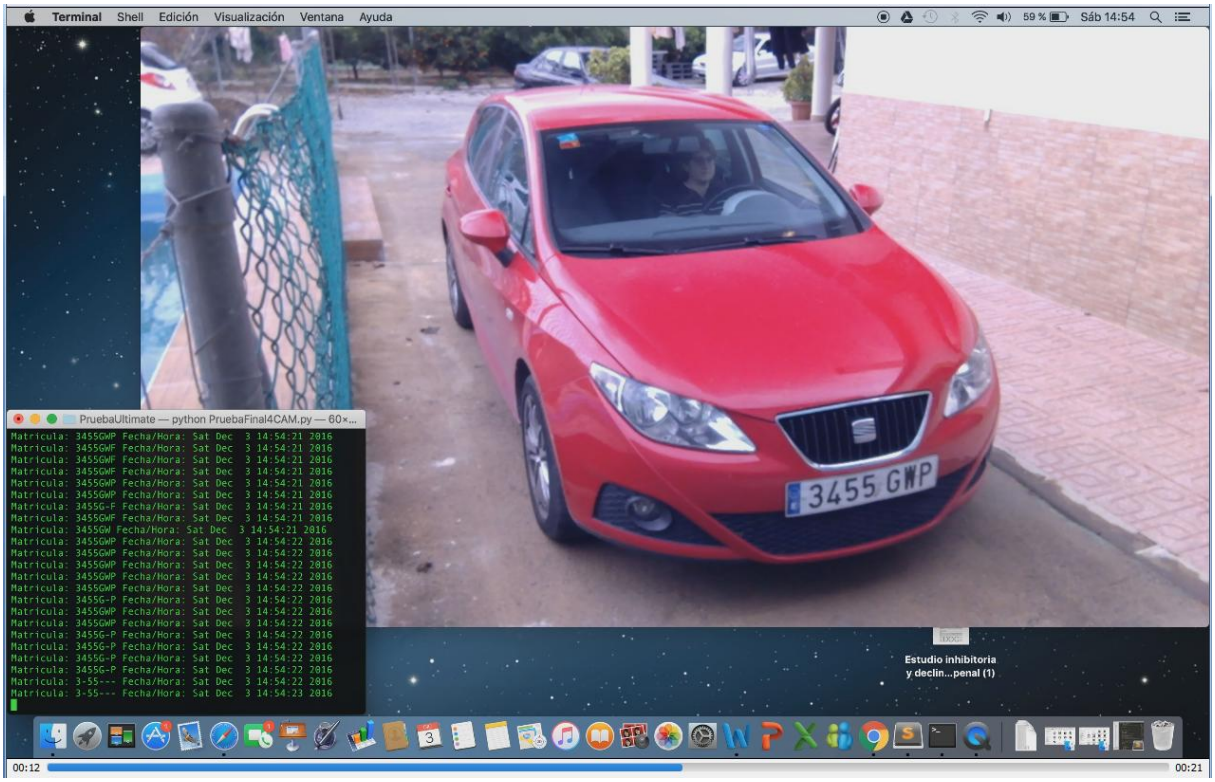

Figura 104. Captura vídeo, segundo 12

Como se puede observar en esta última imagen, el último reconocimiento desciende la tasa de acierto considerablemente, ya que a distancias cortas, como se ha comentado anteriormente, el sistema pierde eficacia.

Por lo tanto, en esta prueba con vídeo, desde que el sistema realiza el primer reconocimiento hasta el último transcurren 5 segundos en los cuales realiza 46 reconocimientos en total, es decir, una media de 9 reconocimientos por segundo. En la tabla 7 se muestran los resultados obtenidos. En esta tabla podemos apreciar cuantas veces se ha obtenido cada resultado, con el correspondiente porcentaje con respecto al número total de resultados.

Reconocimiento	Nº veces	Porcentaje
3455GWP	19	41,30%
3455GWF	9	19,57%
3455G-P	7	15,22%
3455GF	3	6,52%
3455WF	2	4,35%
3455G-F	2	4,35%
3455G	1	2,17%
3455GP	1	2,17%
3455GW	1	2,17%
3-55---	1	2,17%
TOTAL	46	

Tabla 7. Resultados obtenidos en vídeo

Con esto, podemos determinar que la matrícula reconocida sería "3455GWP", que es la que se reconoce en mayor porcentaje de veces. Por lo que este resultado sería correcto ya que se corresponde con la matrícula real.

Conclusiones

7. Conclusiones y trabajos futuros

7.1 Conclusiones

En el presente capítulo se exponen las conclusiones sobre el desarrollo del sistema y las líneas futuras asociadas a la aplicación que se han realizado.

El desarrollo del proyecto ha precisado de una duración de un año aproximadamente desde el planteamiento inicial, pasando por la investigación, desarrollo del sistema y experimentación.

Considero que es un plazo razonable puesto que he tenido que compatibilizarlo con un trabajo de jornada completa con horario de mañana y tarde, además de tener en cuenta que si bien los profesores me han aportado muchos conocimientos, el grueso del desarrollo se ha llevado a cabo de forma autónoma.

Respecto a los problemas encontrados durante el desarrollo del mismo, dejando de un lado los problemas circunstanciales, cabe destacar la constante necesidad de explotar al máximo las capacidades de dispositivos de bajo coste. Esto incluye desde la optimización software de las imágenes adquiridas hasta una cuidada implementación del algoritmo de procesado, intentando conseguir siempre el máximo, en la medida de lo posible, de los parámetros de eficacia y eficiencia.

Se considera que, en gran medida, los objetivos se cumplen de forma razonable, como consecuencia de un alto aprovechamiento de los medios de los cuales se dispone.

Una conclusión más concreta en relación con la aplicación es la siguiente:

Una conclusión más a resaltar es que no se obtiene resultados en corta (2m) y en larga distancia(10m), distancias en las que si se obtienen algún dígito pero para nada nos lleva a un resultado que sea válido.

Podemos concretar que nuestro sistema es eficaz en media distancia (entre 4 y 8 metros). En cuanto a la altura, podemos decir que para las tres alturas comprobadas en el estudio (0.5m, 1.5m y 2.5m) obtenemos buenos resultados.

Tal y como se puede apreciar en tablas anteriormente vistas, los resultados no son los mismos para todas las distancias, ya que concluimos que en el caso de la distancia de 4 y 6 metros, los resultados mejores obtenidos sería a la altura de 1.5 metros, en cambio para la distancia de 8 metros, sería 0.5m.

También podemos concretar que el sistema funciona en distintas condiciones de iluminación, como se ha podido observar en los distintos caso, ya que algunas de las imágenes se han tomado en días soleados a medio día y otras con menos iluminación tomadas al atardecer en un día nublado.

También podemos decir que tenemos buenos resultados en la pequeña prueba realizada con vídeo, ya que en el caso probado nos da el resultado correcto, aunque no es determinante ya que no se ha podido realizar más pruebas de este tipo.

El estudio de las distancias se ha realizado de manera experimental por expertos para el procesamiento de videos en tiempo real en entornos urbanos como calles y avenidas. Este estudio nos permite concretar de una manera mucho más exacta y precisa, a la vez que cómoda para el que realiza el proyecto, cual serían los mejores resultados.

A modo de ejemplo, podemos decir que desde que un coche entra en una calle, a la que le está reservado su acceso excepto a residentes, taxistas y autobuses urbanos, la cámara empieza a captar una seria de imágenes, procesando aquellas que reúnen los requisitos de posible matricula, y llevando a cabo varios resultados, puesto que nos daría buenos resultados a 4, 6 y 8 metros, ello es claramente posible y totalmente viable ya que tiene un procesamiento excelente.

Además tiene como extra que el hecho de que a varias distancias sea capaz de reconocer completamente los caracteres de una matrícula que conlleva que el margen de error sea mínimo, puesto que si lo reconoce a 8, 6 y 4, lo que haría sería verificar los resultados a medida que el coche avanza.

Es un sistema claramente eficaz y eficiente diseñado para componentes *low cost*.

7.2 Trabajos futuros

Como trabajos futuros, se podría ampliar el estudio a diferentes escenarios, incluyendo más variación en cuanto a luminosidad se refiere.

Ampliar el estudio a un mayor número de matrículas e incluir diferentes tipos de estas. Este sistema se ha desarrollado para detectar y reconocer las placas de vehículos estándar hoy en día, con un tamaño de 520mm x 110mm. Se podría ampliar el sistema para que reconociera placas de motocicleta que son de distinto tamaño, o incluso de ciclomotor que tienen distinta tipología.

Además se podría mejorar el sistema optimizando el tiempo de computo. En este sentido, hoy día existe otra *Raspberry Pi* más actual que con la que se han realizado las pruebas y ello aplicado a que la tecnología avanza a niveles exponenciales, podríamos decir que aproximadamente cada año sale al mercado una nueva versión de *Raspberry Pi* con mejores características técnicas, lo que nos llevaría a un descenso del tiempo de procesamiento, con dichas actualizaciones de hardware.

Bibliografía

8. Bibliografía

8.1 Libros y artículos

González, R.C., Wintz, P. ,"Procesamiento digital de imágenes". Addison-Wesley, 2001, 2nd ed.

Lutz, Mark. "Programming Python", 2001

Eben Upton; Gareth Halfacree; Raspberry Pi User Guide, 3rd Edition, 2014

Baggio ; Shervin Emami; David Millán Escrivá; Khvedchenia Ievgen, 2012. "Mastering OpenCV with practical computer vision projects" (Chapter 5. Number Plate Recognition Using SVM and Neural Networks)

Erik Bergenudd, "Low-Cost Real-Time License Plate Recognition for a Vehicle PC", KTH Electrical Engineering, Suecia, 2006.

Carlos Parra Ramos, David Regajo Rodríguez, "Reconocimiento Automático de Matrículas", Universidad Calos III de Madrid, 2006.

8.2 Referencias web

Image Gradients, http://docs.opencv.org/master/d5/d0f/tutorial_py_gradients.html

Canny Edge Detection, http://docs.opencv.org/master/da/d22/tutorial_py_canny.html

Contours: GettingStarted, http://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html

7. Bounding Rectangle, http://docs.opencv.org/master/dd/d49/tutorial_py_contour_features.htm
Template Matching, http://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html

Image Thresholding, http://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

Perspective Transformation,
http://docs.opencv.org/master/da/d6e/tutorial_py_geometric_transformations.html

2.Gaussian Blurring, http://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html

4.Closing, http://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html

9.Extreme Points, http://docs.opencv.org/master/d1/d32/tutorial_py_contour_properties.html

UNIVERSIDAD DE ALMERÍA

GRADO EN INGENIERÍA INFORMÁTICA, 2016/2017