UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Sistema de visión artificial para la detección y lectura de matrículas

Autor:

Muñoz Manso, Roberto

Tutor:

de la Fuente López, Eusebio

Departamento de Ingeniería de Sistemas y Automática

Valladolid, julio de 2014

***A** mis padres*
y mi hermana

Summary and keywords

SUMMARY

Algortihm for dynamic and automatic license plates inspection and extracting characters from it, to be used in diferents areas. It will work on cars that appear in an image, wich is taken by a camera when detects the vehicle, so it works in real time.

It will consist on three blocks: location of the license plate, process it and the recognition of the characters. This algorithm is designed to obtain satisfactory results regardless of the size and the rotation that may have the license plate.

For the development it has been used the open source library OpenCV, wich is based on the programming language C++. OpenCV library is designed for image processing through artificial vision.

KEYWORDS

Artificial Vision, OpenCV, SURF, License plates, OCR

Resumen y Palabras Clave

RESUMEN

Algoritmo para la inspección dinámica y automática de placas de matrículas y la extracción de caracteres de la misma, para poder ser usado en diferentes ámbitos. Se trabajará sobre coches que aparecen en una imagen tomada por una cámara al detectar el vehículo, es decir, en tiempo real.

Constará de tres bloques: localización de la placa de la matrícula, tratamiento de la misma y reconocimiento de los caracteres. Dicho algoritmo estará creado para obtener resultados satisfactorios independientemente del tamaño y rotación que pueda tener la placa.

Para el desarrollo se ha usado la librería de código abierto OpenCV basada en el lenguaje de programación C++, la cual está diseñada para el tratamiento de imágenes por visión artificial.

PALABRAS CLAVE

Visión Artificial, OpenCV, SURF, Matrículas, OCR

Índice general

Índice de figuras

Capítulo 1
Introducción y justificación del trabajo

En este primer capítulo de la memoria, se va a exponer la finalidad de la aplicación a desarrollar, además de justificarla con diferentes ámbitos de aplicación del software desarrollado. Para conseguir el fin último, es necesario marcar unos objetivos, junto con una serie de etapas intermedias para cumplirlos. Por último se realiza un resumen de los diferentes capítulos que componen la memoria.

1.1. Preámbulo

El presente trabajo se ha realizado en el departamento de Ingeniería de Sistemas y Automática de la Escuela de Ingenierías Industriales de la Universidad de Valladolid. Surge de la idea de desarrollar un sistema de detección y reconocimiento de placas de matrículas en automóviles. El objetivo principal es que pueda ser empleado en el control de acceso a parkings.

Para ello se va a tabajar 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: localizar de manera precisa dónde está situada la

placa de la matrícula, cuando además de ella aparece también una parte del vehículo. Para ello se ha elegido basar el desarrollo del proyecto en tecnologías que permiten la extracción de características locales.

La totalidad del trabajo consiste en localizar la placa dentro de una imagen tomada por una cámara, además de identificar el código que aparece en ella por medio de un módulo OCR como se muestra en la figura 1.1.

Figura 1.1. 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, la 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. 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:

- Menor tiempo de ejecución.
- Menor cantidad de recursos humanos necesarios.
- Menor fatiga por parte del usuario o equipo asociado a la tarea.
- Mayor fiablidad 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 1.2. Ejemplo de la 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 cuanto tiempo ha permanecido un coche estacionado, o para llevar un registro de vehículos.
- Controles de acceso en los que sólo se permite la entrada a ciertos vehículos.
- 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 vehículos 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 en 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. 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.

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 siguiente pauta es realizar la programación de todos estos algoritmos, que se van a desarrollar con la librería OpenCV y Tesseract. El lenguaje de

programación elegido es C++, que se complementa con la librería elegida OpenCv que está destinada exclusivamente a la visión artificial y con Tesseract.

Otra pauta es localizar la placa dentro de la imagen para poder extraer el identificador del vehículo, atenuando el efecto de otras variables dentro de la imagen como la iluminación, escala y rotación.

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 ésto, se realiza el reconocimiento de caracteres a través de un algoritmo OCR.

Todas estas pautas quedan reflejadas de manera más general en la figura 1.3.

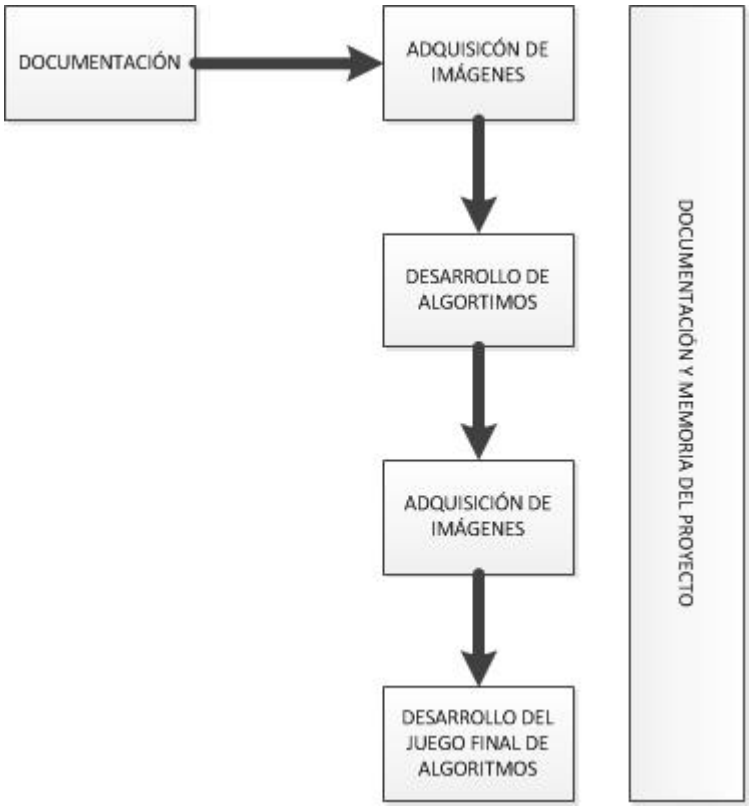

Figura 1.3. Fases en el desarrollo

La primera fase se corresponde con un objetivo parcial del proyecto. Éste abarca toda la **documentación** sobre la visión artificial.

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 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.

Con el juego de imágenes, se propondrán, elaborarán y probarán **algoritmos** que resuelvan esta aplicación.

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.

Al mismo tiempo que se realizan estos puntos expuestos, se realizará de manera paralela esta **memoria** para la documentación del trabajo.

1.4. Estructura de la memoria

En esta parte del primer capítulo se van a analizar los diferentes aspectos que tratarán los capítulos que se incluyen en la memoria de este trabajo trabajo.

- **Capítulo 1:** Incluye una introducción del trabajo, con una justificación y objetivos del mismo.
- **Capítulo 2:** Se realiza un estudio del estado del arte y se hace hincapié en la necesidad de los sistemas de detección de matrículas para el control de vehículos.
- **Capítulo 3:** Introducción a la visión artificial y las ventajas sobre la visión humana en la automatización. También se referencian proyectos y trabajos varios realizados con anterioridad sobre visión artificial. Se justificará OpenCV como lenguaje.
- **Capítulo 4:** Se hará una mención a la formación de imágenes digitales. Posteriormente se mostrarán las operaciones principales utilizadas en este trabajo para tratar y obtener información.
- **Capítulo 5:** Explicación detallada de los detectores de características locales.
- **Capítulo 6:** Desarrollo y explicación de la estrategia seguida para realizar el programa.
- **Capítulo 7:** Funcionamiento de la aplicación. Resultados experimentales obtenidos. Se ejecuta una batería de imágenes sobre la aplicación y se comentan los resultados.

- **Capítulo 8:** Se comentan las conclusiones obtenidas y la revisión de los objetivos del trabajo. También se mostrarán líneas futuras.

Capítulo 2
Sistemas de detección de matrículas

Para tener una idea clara de la aplicación que se quiere desarrollar así como su alcance en diferentes campos, es necesario incluir este capítulo en la memoria. En él 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.

2.1. ¿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 van a estar orientadas al tipo de placas a tratar.

2.2. Diferentes modos de actuación. Estado del arte

La implementación de un sistema de detección de matrículas no es única, ya que el estado del arte muestra que hasta el día de hoy hay varias alternativas. 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.

2.2.1. 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 2.1. 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.

Una vez que se tienen estos límites, se hace algo parecido para obtener los límites laterales, como se refleja en la figura 2.2. 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.

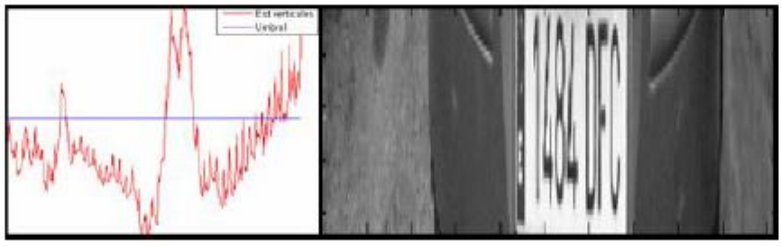

Figura 2.1. Detección de verticales [1]

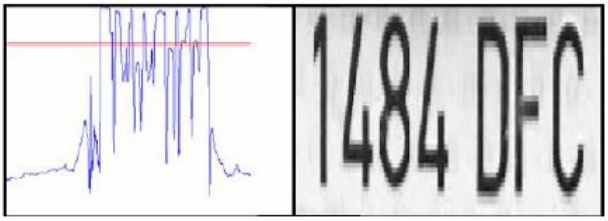

Figura 2.2. Detección de horizontales [1]

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.

2.2.2. Sistema basado en binarización

Este es un método muy usado para todos los formatos de placas en los que el fondo es blanco. Consiste en binarizar 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 pixeles blancos en la matriz binaria como se observa en la figura 2.3.

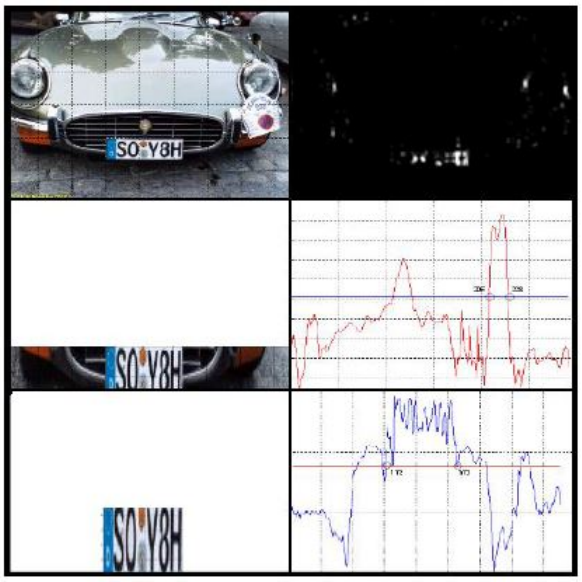

Figura 2.3. Detección a través de binarización [1]

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.

Durante el presente documento se va a reflejar el *modus operandi* de otra técnica, que es la empleada para este trabajo. Dicha técnica está basada en la extracción de características locales mediante SIFT y SURF. Ello permite conocer con alto grado de precisión dónde se encuentra la placa dentro de la imagen.

Aunque se han mencionado estas dos técnicas, no son las únicas que se emplean para ofrecer soluciones de detección de placas. Por ejemplo las basadas en la transformada de Hough, dirección y sentido de los gradientes de contornos, etc.

Cabe destacar la técnica empleada en la mayoría de los parkings, donde se utiliza una luz infrarroja para resaltar la matrícula en la captura de la imagen. Se utiliza una radiación invisible para el ojo humano. De este modo la placa estará completamente definida haciendo una binarización y un etiquetado.

2.3. 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 de defectos en la matrícula (suciedad, fracturas, doblez). 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 pueda ser utilizado en varios ámbitos; por ejemplo controles de acceso, localización de vehículos robados, control de acceso a diferentes parkings, etc.

Hay que tener en cuenta estos 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.

Por esta sencilla razón, este trabajo se va a realizar usando tecnologías invariantes a cambios en la escala, rotación, iluminación, etc, en los que destacan los detectores de características locales SIFT y SURF.

2.4. Aplicaciones prácticas

Tal como se dijo en el capítulo primero, 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.

2.4.1. 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.

Figura 2.4. Localización de vehículos [2]

2.4.2. 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.

2.4.3. 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 viás de la ciudad, sin contribuir al Impuesto de Circulación de la misma.

Figura 2.5. Cobro de peajes urbanos [3]

2.4.4. Control de velocidad instantánea

En vez de controlar la velocidad instantánea de un vehículo, se puede detectar la matrícula del mismo 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 2.6. Control velocidad instantánea [3]

2.4.5. 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.

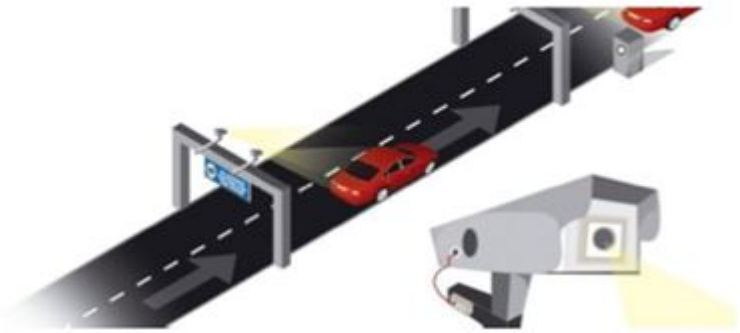

Figura 2.7. Cálculo velocidad media [3]

2.4.6. Sistemas 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.

Figura 2.8. Sistemas de parking [1]

2.4.7. Optimización del tráfico urbano

Cada vez es más necesaria la optimización de la movilidad de los vehículos en las ciudades grandes, por lo que este sistema se presenta como una gran alternativa. Lo que se pretende es identificar los vehículos de forma dinámica, instalando estos sitemas 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 pueda llegar a emplear en diferentes rutas. Se trata de una aplicación para optimizar el tráfico en tiempo real.

Capítulo 3
Introducción a la visión artificial

Este tercer capítulo define qué es la visión artificial y se compara con un sistema de visión humana obteniendo similitudes y diferencias, indicando las ventajas e inconvenientes de la utilización de cada uno de ellos. Se explica qué es una imagen digital, así como el concepto de píxel. Se hace hincapié en la importancia de la visión artificial orientada a diferentes campos de aplicación como la medicina, robótica, etc. Por último, se realiza una explicación de la librería OpenCV.

3.1. Introducción

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 importantes 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 realzar y extraer la información que hay en las imágenes para ser analizadas e interpretadas por los seres humanos.

3.2. ¿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 un punto de vista ingenieril, un sistema de visión artificial es un sistema autónomo que hace las veces en algunas de las 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 muestreada y digitalizada para su procesamiento en un ordenador.

3.3. 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, que se puede apreciar en la figura 3.1, 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.

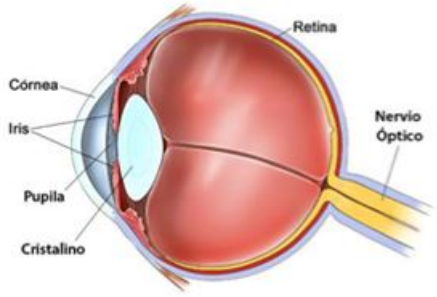

Figura 3.1. Ojo humano [4]

Respecto 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 vídeo a un ordenador que analiza la información, la cual se puede emplear para el fin que se desee.

La simlitud entre un sistema de visión artificial y el sistma 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 3.2.

- 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.

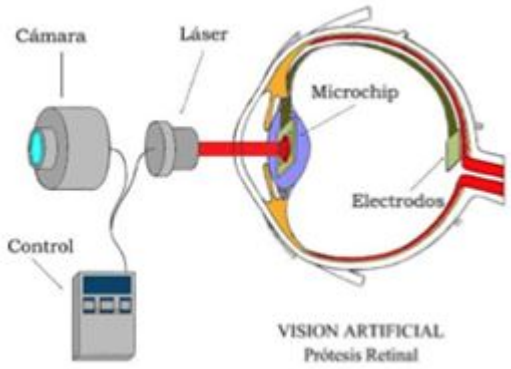

Figura 3.2. Correspondencia ojo humano-sistema de visión [4]

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.

Aún 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.

Figura 3.3. Inspección de productos [5]

Que una tarea sea monótona y repetitiva, hace que la efectividad de un humano en la inspección, como en la figura 3.3, no sea superior al 80 %. Mientras

que un sistema automático de visión implementado de manera correcta puede llegar a una efectividad del 99.7 %.

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.

3.4. 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. La forma de tratar las imágenes dependen de las herramientas de las que dispone el ser humano, desde algoritmos básicos hasta los más complejos que puedan llegar a existir. También puede ayudar la tecnología existente para facilitar el procesamiento como ordenadores, programas, etc.

3.4.1. 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 obtener la 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 transrormación pervia de la imagen. Esta imagen continua debe ser transformada de alguna forma en un conjunto de números que pueda 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 presentada siendo su conjunto una matriz de información para el uso digital como se ve en la figura 3.4.

Figura 3.4. Matriz de píxeles

Cada píxel queda codificado por un conjunto de bits que tienen una longitud determinada, la cual es llamada la profundidad de color. Por ejemplo, un píxel puede codificarse con un byte (8 bits), admitiendo 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 máxi-

ma 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 (en esta medida de computación no se usa la base binaria de 1024). Este 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 píxeles, 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 megapíxeles 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 megapíxeles es menos representativa que entre 3 y 4, debido a que no se trata de una medida exponencial.

La resolución de una imagen queda difinida 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 2.700 x 3.600 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 ilustración.

3.4.2. 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 modificar 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 3.5 y 3.6 se pueden ver dos ejemplos de escalas de grises.

En la figura 3.7 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.

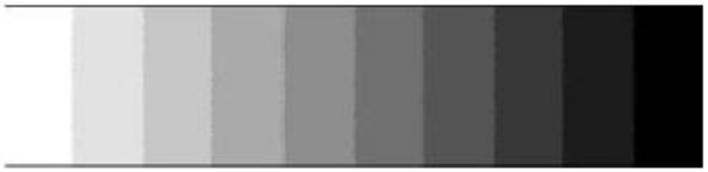

Figura 3.5. Escala de grises 1 [6]

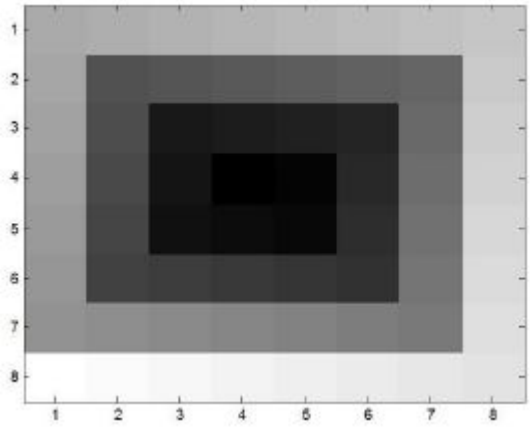

Figura 3.6. Escala de grises 2 [6]

En el modelo RGB, cada color primario tiene su propia escala para representar un color específico. El problema sería tener 3 elementos de 3 diferentes matrices para cada uno de los posibles colores. Si se sabe que los 3 elementos de las distintas matrices se adicionan, cómo 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 3.7. Valor de los píxeles en niveles de gris [6]

3.5. Aplicaciones varias de la visión artificial

En muchas ocasiones la visión por computador no es la mejor solución a los problemas. Por ejemplo, la conducción por carretera de un vehículo, algo que corresponde a una solución humana. 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 el único sensor. El segundo es referido a la navegación en robótica, 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.

3.5.1. 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 estereocópica para tratar de reconstruir la escena en 3-D. El uso del movimiento basado en la visión constituye un gran recurso.

3.5.2. 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

movimientos de terrenos captando dos imágenes en diferentes momentos de tiempo para comprobar la variación mediante una diferencia de imágenes (bajo similares condiciones 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.

3.5.3. Medicina

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

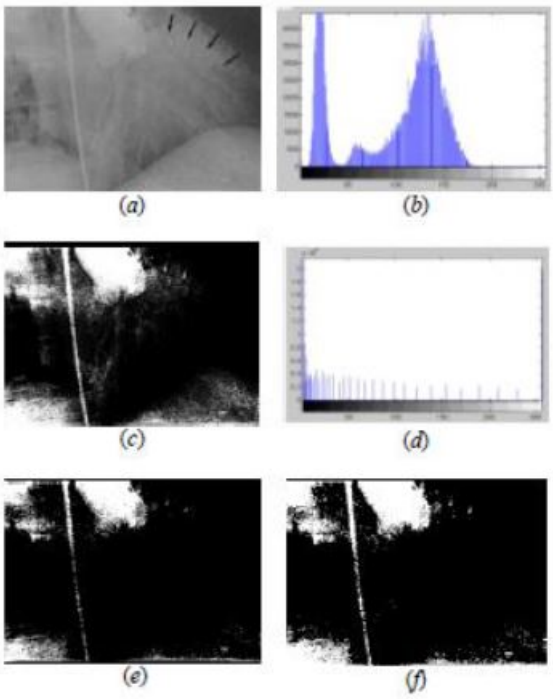

Figura 3.8. Sistema de inspección en medicina [1]

Un claro ejemplo es el de una radiografía que se observa en la figura 3.8, 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 binariza 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.

3.5.4. 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 puede 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.

3.5.5. 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 binarizado.

Otra aplicación destacada es el reconocimiento de caras de personas utilizando perfiles de intensidad. Otras aplicaciones presentes en el estado del arte 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.

3.5.6. Inspección y control de calidad

La inspección se puede realizar para verificar ciertas características o verificar dimensiones en objetos manufacturados.

La inspección se refiere a la verificación de si un objeto cumple con ciertos criterios. Para ello se compara un objeto con modelos que describan características buscadas.

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 3.9. Sistema de control de calidad [1]

3.5.7. 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.

3.5.8. 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.

Figura 3.10. Cartografía mediante visión artificial [1]

3.6. OpenCV

Las siglas *OpenCV* provienen 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 la licencia BSD.

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 estérea 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 multi núcleo. 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 modulo 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.

Capítulo 4
Algoritmos utilizados en visión artificial

En este cuarto capítulo de la memoria se va a exponer una base teórica de las diferentes técnicas aplicadas sobre las imágenes para la realización del sistema. En primer lugar se exponen los conceptos matemáticos sobre las imágenes digitales. A continuación se detallan de manera teórica las operaciones generales: escalar imagen, modificar el color y binarización. También se exponen otros algorimtos utilizados como el etiquetado, rellenado de figuras, filtrado de las mismas, extracción de contornos y detección de rectas. Para finalizar se explica qué es un algoritmo OCR.

4.1. Introducción

La visión artificial está destinada a trabajar con imágenes. Las imágenes pueden ser fotos o extraerlas de un vídeo obtenido a partir de una cámara de cualquier característica (cámara web, cámara de vídeo digital, etc.).

Para lograr el objetivo del trabajo hay que tratarlas, para obtener todas las características posibles. Para tratarlas se utilizan todas las herramientas de las

que se dispone, desde algoritmos básicos hasta los más complejos.

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.

4.2. 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 4.1.

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. Para el presente trabajo es conveniente que las imágenes tengan alta resolución, para poder aplicar la técnica SURF. Tambíen podría funcionar con resoluciones bajas, pero los resultados no serían los esperados.

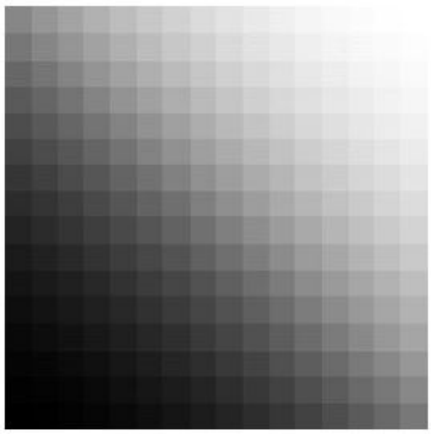

Figura 4.1. Diferentes tonalidades de gris [6]

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.

4.2.1. 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.

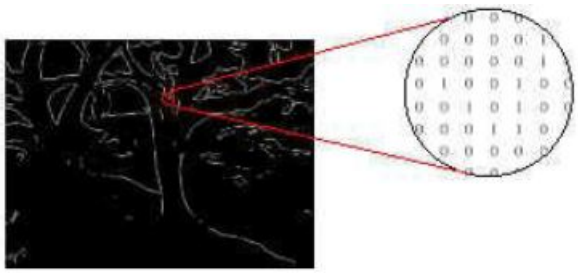

Figura 4.2. Imagen binaria [7]

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. En la figura 4.2 se muestra un ejemplo de imagen binaria.

4.2.2. 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 unit8, 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 4.3 se muestra un ejemplo de una imagen en escala de grises.

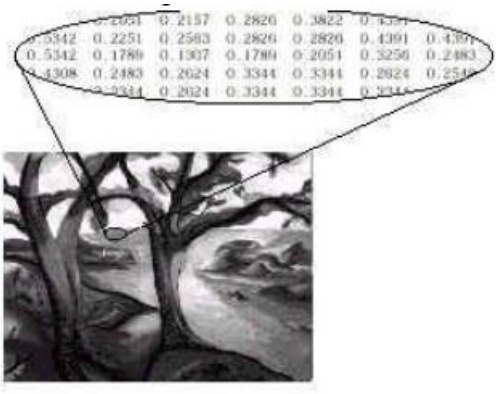

Figura 4.3. Imagen en escala de grises [7]

4.2.3. 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 codifica 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 de 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 4.4 se muestra un ejemplo de imagen a color.

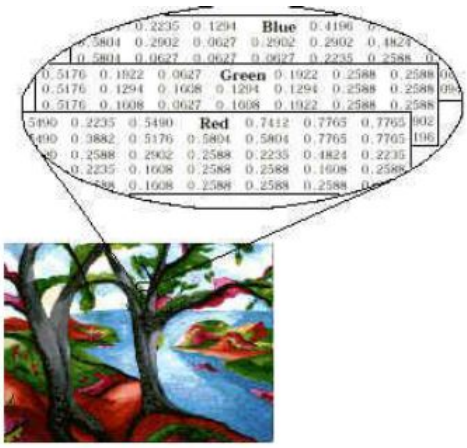

Figura 4.4. Imagen de color [7]

4.3. 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 4.5.

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 venideras.
- **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.

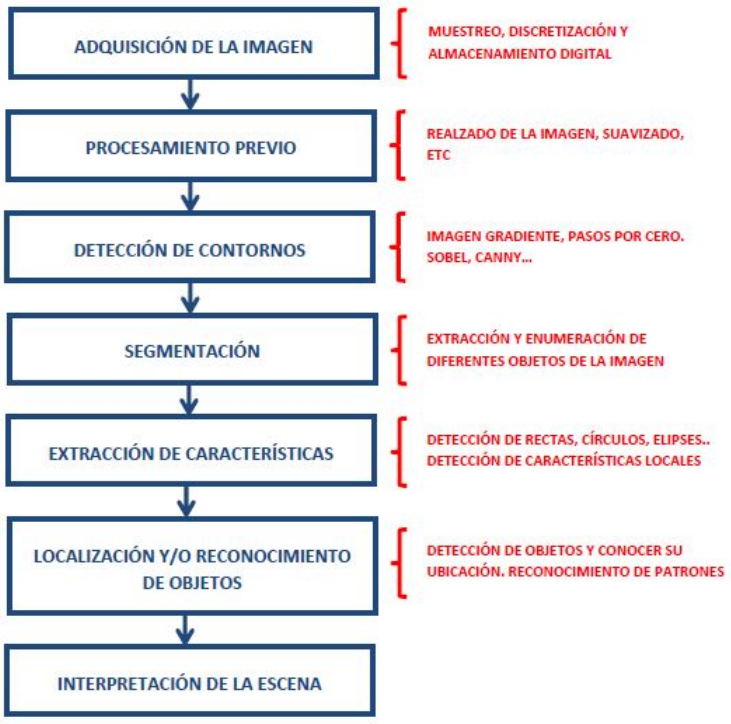

Figura 4.5. Etapas en el procesamiento de una imagen

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.

En el presente trabajo se aplican procedimientos de visión artificial relativamente novedosos que no están incluidos en las técnicas clásicas de visión. Esta tecnología trata de obtener características locales de la imagen, para obtener un modelo dentro de ella.

4.4. 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.

4.4.1. Modelos de captura de imágenes

Se distingue 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 imágenes de elementos bidimensionales. También es posible 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.

4.4.2. 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 espacio 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.

4.5. Operaciones generales. Técnicas clásicas

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

4.5.1. 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 anisotrópico.

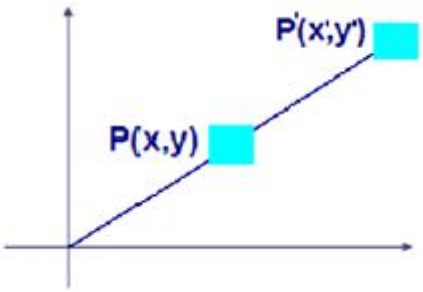

Figura 4.6. Escalado de una imagen [1]

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

$$x' = s_x \cdot x$$
$$y' = s_y \cdot y$$

Si $s_x = s_y$ se producirá un escalado isotrópico. Si $s_x \neq s_y$ el escalado es anisotrópico.

4.5.2. 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.

Figura 4.7. Imagen convertida a escala de grises

4.5.3. Binarización. Binarización automática

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 figura 4.8.

Figura 4.8. Imagen normal/binarizada

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 binarización: 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 binarización con poco procesamiento, además de ser rápido y fiable. En ocasiones se utilizan técnicas de retroiluminación para obtener imágenes bien contrastadas que se puedan reducir a binarias y quede representada claramente la información deseada.

La forma para 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 menor que el umbral, ese píxel valdrá 0, mientras que si es mayor valdrá 1.

Aunque binarizar 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 binarización 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 binarización, 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 pixeles 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 binarización no proporcionará buenos resultados.

Para establecer el umbral de binarización 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 binarización. 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 invalidaría 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 que 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 binarización se conoce con el nombre de binarización adaptativa. Este tipo de binarización analiza los

diferentes niveles de gris de la imagen y halla el umbral más óptimo para cada zona de la imagen. En la parte derecha de la figura 4.9 se muestra una imagen con una binarización con umbral fijo y en la parte izquierda se muestra con la binarización adaptativa.

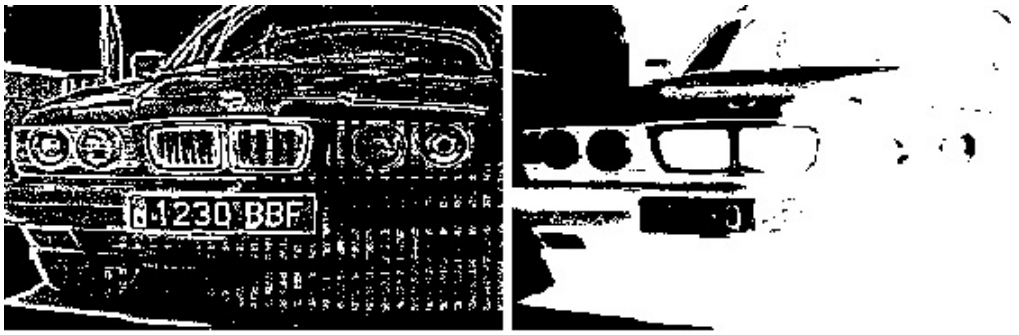

Figura 4.9. Diferentes binarizaciones

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 el mínima.

Sea una imagen I de N niveles de gris, el pseudocódigo para la binarización 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.
- Binarizar la imagen en el umbral m.

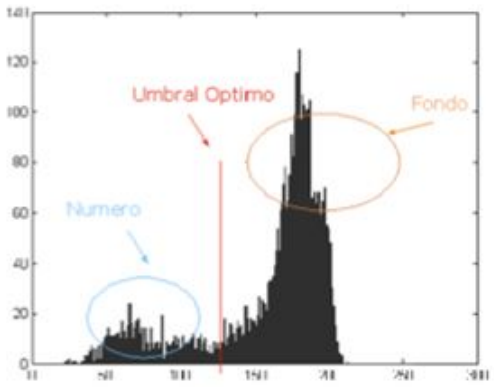

Figura 4.10. Histograma con el umbral óptimo

Teniendo un histograma bimodal como el de la figura 4.10, 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.

4.6. Etiquetado. Extracción de características

En este apartado se va a exponer en qué consiste el etiquetado y cómo se realiza. A partir de las etiquetas creadas, se van a comentar las características que se pueden extraer de las mismas.

4.6.1. Etiquetado

La mayoría de las aplicaciones en visión artificial están destinadas o requieren encontrar y clasificar cada uno de los elementos que encuentran dentro de la imagen. La forma de hacerlo es empleando el etiquetado, el cual asigna un mismo valor a todos los píxeles que forman parte de un mismo objeto en una imagen. Con ello se podrá contar más adelante los elementos, ver si alguno no es apto, obtener características de cada uno de ellos, etc.

El empleo de esta técnica, permitirá en pasos futuros poder aplicar otros algoritmos a la imagen, para poder procesar correctamente sus elementos como encontrar la orientación, ejes de inercia, etc.

Hay que tener claro que no se devuelve ninguna imagen modificada, ya que sólo se pretende obtener información de la imagen de entrada. Con esta información luego se trabajará. Como imagen tal, no se va a trabajar con el valor

de cada pixel sino con la etiqueta que lleve, es decir, en esta función es indiferente el color del píxel, su nivel de gris, etc. Lo que importa es saber a qué objeto pertenece (fondo o alguno de los objetos).

Para un correcto funcionamiento de este proceso, es necesario trabajar con una imagen binaria y los elementos no deben estar conectados entre sí, ya que esos elementos conexionados se tomarán como uno solo y esta operación no daría sus frutos. Para eliminar dichas conexiones se emplean operaciones morfológicas, mas concretamente apertura o cierre. Se puede decir que éste es uno de los casos en los que una imagen binaria aporta información que, ni por asomo, podría suministrar una imagen en color o escala de grises.

La principal precaución a tener en cuenta, es el diferenciar a partir de una imagen binaria cual es el fondo y cuales son los objetos. Unos corresponderán a partes blancas y otros a partes negras, pero hay que identificarlo. Los objetos siempre han de corresponder a partes blancas para poder etiquetarlos. En el caso de que no coincida, se realizará una inversión de la imagen.

Para realizar el algoritmo, en primer lugar hay que enumerar cada píxel que compone la imagen como se muestra en la figura 4.11.

A continuación, el algoritmo agrupa las etiquetas por proximidad, de modo que a cada grupo de píxeles conectados les asigna un valor, el que sea el menor de todos ellos (figura 4.12).

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	58	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Figura 4.11. Enumeración de los píxeles [8]

El etiquetado necesita recorrer la imagen varias veces para clasificar debidamente las etiquetas mínimas. Se necesitará una variable (etiqueta) de valor inicial 1 y que deba ir incrementándose según se encuentre píxeles de valor 0 (negro).

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	45	49	50
51	52	53	54	45	56	58	45	59	60
61	62	63	64	45	45	45	45	69	70
71	72	73	74	45	45	45	45	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Figura 4.12. Enumeración de los píxeles de un objeto [8]

Es necesario que no exista ningún tipo de ruido en el etiquetado, ya que provocaría que hubiese más etiquetas de las que realmente hay que tener, lo cual provocaría errores. Esto es debido a que los píxeles aislados se consideran como objetos. Para que este problema no aparezca, se utiliza alguna de las operaciones morfológicas ya mencionadas.

4.6.2. Extracción de características

La visión humana es capaz de identificar con casi perfecta claridad los objetos, y es fácil comparar tamaños entre varios objetos y asimilar características. En este flanco la visión artificial para conseguir emular a la humana se basa en las características de ésta, y la supera cuando las tareas son repetitivas y monótonas. Debido al cansancio humano se provoca un cúmulo de imprecisiones que favorecen el desarrollo de tecnologías basadas en la visión.

Una vez realizado el etiquetado, se pueden extraer una serie de características de cada objeto. Con estas características se puede decidir si un elemento posee los parámetros buscados, algo que ofrece de manera automática unos resultados similares a los que ofrece el cerebro humano.

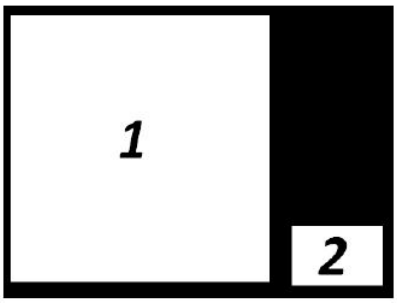

Figura 4.13. Etiquetas de una imagen [8]

Entre todas las características que se pueden medir, se pueden destacar las siguientes: área, centro de gravedad, ejes de inercia, número de Euler, perímetro y número de agujeros.

4.7. 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.

4.7.1. Eliminación de regiones según su área

Existe 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 del tipo sal y pimienta, además de objetos menores que aparecen en la binarización y no representan ningún objeto deseado.

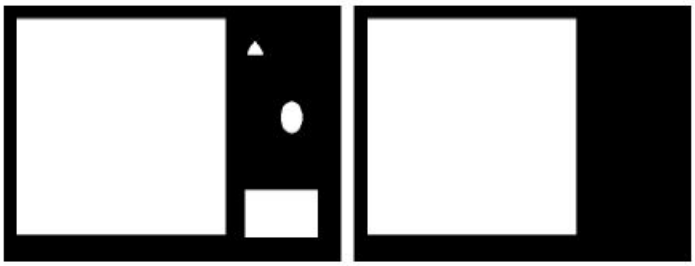

Figura 4.14. Filtrado de imagen por área [9]

4.7.2. 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 solidas. Con ello se puede erosionar y eliminar con más margen, ya que se obtiene el contorno deseado con mayor facilidad.

Figura 4.15. Rellenado de figuras [9]

4.8. 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 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.

$$\overline{gradl} = \left(\frac{dl}{dx}, \frac{dl}{dy}\right)$$

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

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 4.16.

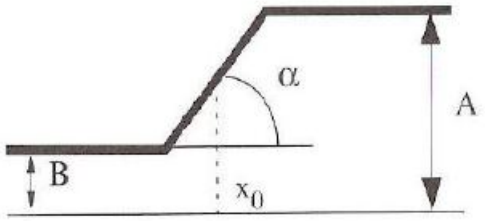

Figura 4.16. Dos niveles de gris [6]

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 textura.

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

4.8.1. 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 pixeles 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 correspondar 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 pixel.
- **Supresión no máxima:** Se logra el adelgazamiento del ancho de los bordes, obtenidos con el gradiente, hasta lograr bordes de un pixel de ancho.
- **Histéresis de umbral:** Se aplica una función de histéresis basada en dos umbrales; con este 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 4.17 se muestra un ejemplo de funcionamiento.

Figura 4.17. Contornos por Canny

4.8.2. 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 4.18 se muestra un ejemplo de ejecución de este detector de contornos.

Figura 4.18. Contornos por Sobel

4.9. Transformada de Hough. Detección de rectas

La transformada de Hough es un algoritmo utilizado para el reconocimiento de patrones en imágenes que permite encontrar ciertas formas dentro de una imagen, como líneas, círculos y cualquier curva que siga una ecuación determinada dentro de una imagen.

Debido a que las imágenes son bidimensionales, para derivar se precisa del gradiente. En cada punto de la imagen el gradiente será diferente, alcanzando valores muy elevados en contornos. Este gradiente apunta en la dirección del cambio de intensidad.

La versión más simple consiste en encontrar líneas. Su modo de operación es principalmente estadístico y consiste en averiguar para cada punto si forma parte de una línea. De este modo se averiguan las posibles líneas de las que puede formar parte el punto. Eso se realiza para todos los puntos de la imagen y al final se determina que líneas fueron las que más puntos posibles tuvieron, siendo éstas las líneas en la imagen. Para aplicar la transformada de Hough, es necesario obtener primero una imagen binaria de los píxeles que forman parte de la frontera del objeto. Normalmente esta imagen es el resultado de un operador de contornos.

Un problema de este método surge cuando se quiere representar una recta que se aproxima a posiciones verticales (singularidad). En este caso, tanto la pendiente como la ordenada en el origen tienden a infinito. Para evitar este problema, lo que se hace es utilizar la representación normal de recta.

Figura 4.19. Líneas detectadas por Hough

Aunque este método está generalizado para la líneas rectas, también se puede aplicar para detectar cualquier forma que tenga la función g(x,c), donde x en un vector de coordenadas y c un vector de coeficientes. Esto permite hallar, entre otros, los puntos geométricos de los puntos de una circunferencia.

4.10. 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 pueden crear algoritmos propios o utilizar, como en este trabajo, una librería ya existente. Para utilizar esta librería se crea un programa, el cual hace llamadas a las API de dicha librería. La librería utilizada es Tesseract, que es un motor OCR, el cual contiene una gran cantidad de funcionalidades.

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; esté binarizada.

Aún 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 pertencen 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 errorres 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 4.20.

- **Binarización.** 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 con 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: método de proyección, métodos estadísticos, métodos estructurales, etc.

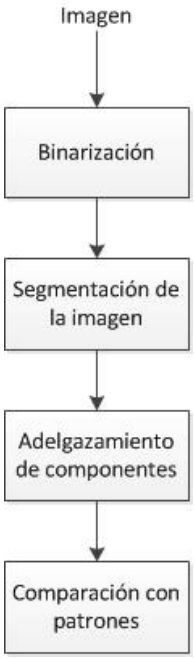

Figura 4.20. Etapas de un algoritmo OCR

Capítulo 5
Extracción de características locales en una imagen

En este capítulo se van a explicar los detectores de características locales debido a su importancia, ya que son la base de la aplicación. Se van a explicar las alternativas tenidas en cuenta SIFT y SURF, justificando la utilización de la técnica SURF.

En la actualidad existen diferentes métodos utilizados en el reconocimiento de imágenes, con mayor o menor precisión. Para este trabajo se han tenido en cuenta dos de estas técnicas, que van a ser descritas en este capítulo.

- SIFT: *Scale Invariant Feature Transform.*
- SURF: *Speeded Up Robust Features.*

SIFT es un algoritmo para detectar y describir las características locales en las imágenes. Fue descubierto en 1999, lo que demuestra que es una técnica bastante reciente, pero no es el único método. Otra técnica, que es la utilizada en el presente trabajo, y que también llama la atención, es SURF, un método inspirado en SIFT que se presentó en 2006, pero que tiene una ventaja, la rapidez.

5.1. SIFT. Scale Invariant Feature Transform

Se trata de una transformación de la información que proporciona una imagen en coordenadas invariantes a la escala en el ámbito local. A partir de las características locales, se busca conseguir invarianza a la escala, a la orientación, parcialmente a cambios de iluminación, etc. También se puede utilizar para buscar correspondencias entre diferentes puntos de vista de una misma escena.

Las características locales se almacenan en los llamados descriptores. Como su nombre indica, tratan de describir localmente zonas importantes de la imagen con determinadas variables, entre ellas el gradiente.

SIFT es un método bastante complejo, por lo que tiene un gran coste computacional. Sin embargo, una correcta implementación del algoritmo puede ser utilizada en una aplicación en tiempo real, siempre y cuando la base de datos de búsqueda no sea muy extensa. Este algoritmo está estructurado en cuatro fases bien diferenciadas, que a continuación se describirán.

5.1.1. Detección de extremos en el espacio de escalas

La primera fase del algoritmo se encarga de buscar un primer conjunto de puntos de interés de la imagen, a los cuales se les llama puntos característicos. Según se vayan llegando a las etapas posteriores, la cantidad de puntos característicos disminuirá. Esto es debido a que no cumplen algunos requisitos y se descartan.

Función en el espacio de escalas

En un primer momento, la búsqueda se realiza sobre todas las localizaciones y todas las escalas de la imagen, ya que son características muy importantes en el momento de ver un objeto desde diferentes vistas. Para detectar localizaciones invariantes a cambios de escala, se utilizará la función conocida como scale-space L(x, y, σ).

Para obtener esta función L(x, y, σ) a partir de la imagen original I(x, y) se utiliza la función gaussiana de la siguiente forma:

$$L\left(x, y, \sigma\right) = G\left(x, y, \sigma\right) * I\left(x, y\right)$$

Donde * denota convolución.

Para hallar todo el espacio L(x, y, σ), es necesario construir una pirámide gaussiana convolucionando con diferentes filtros G(x, y, σ) variando el parámetro σ.

A continuación se definen dos términos que ayudan a comprender la construcción de la pirámide.

- Octava. Se trata del conjunto de imágenes del espacio L que tienen el mismo tamaño, pero que se diferencian en el filtrado σ con el que han sido obtenidas.
- Escala. Son las imágenes del espacio L filtradas con el mismo parámetro σ, pero siendo diferentes sus tamaños.

El número de octavas total dependerá del tamaño de la imagen original y el factor de escalado entre las diferentes octavas es de un medio.

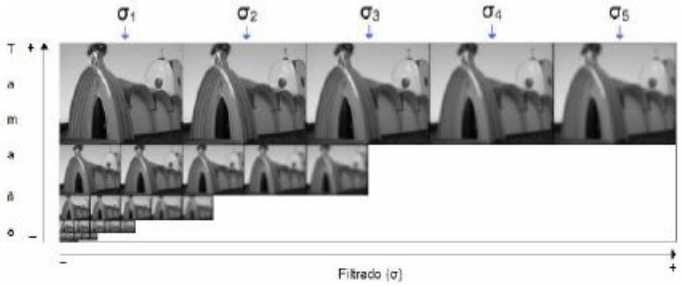

Figura 5.1. Pirámide gaussiana [10]

Función diferencia de gaussianas

En este punto, se utiliza una función que deriva de la función L ya calculada, la diferencia de gaussianas D(x, y, σ). Con ella se pretende detectar puntos característicos estables en la función gaussiana L. Ésta no provoca un gran aumento del coste computacional total, ya que se calcula simplemente restando imágenes vecinas de una misma octava.

$$D\left(x,y,\sigma\right)=\left(G\left(x,y,k\sigma\right)-G\left(x,y,\sigma\right)\right)*I\left(x,y\right)=L\left(x,y,k\sigma\right)-L\left(x,y,\sigma\right)$$

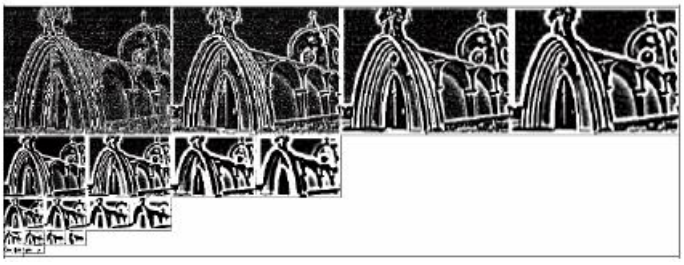

Figura 5.2. Pirámide de diferencias gaussianas [10]

Se utiliza esta función, ya que los máximos y mínimos proporcionan las características más estables comparando con otras funciones utilizadas también en este ámbito.

Detección de extremos locales

Con los cálculos anteriores, ahora se hallan los máximos y los mínimos locales del espacio D(x, y, σ). Todos los pixeles de cada imagen de la pirámide son comparados con sus vecinos de la propia imagen y con los vecinos de la escala anterior y posterior.

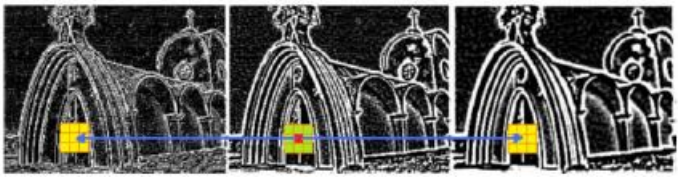

Figura 5.3. Pirámide de diferencias gaussianas [10]

Los puntos sólo se seleccionan como puntos clave si es mayor que vecinos o menor que todos ellos. El coste de realizar todas las comparaciones no es elevado, ya que la mayoría de puntos se van descartando a medida que se explora la imagen. Cabe señalar que solo se podrán detectar puntos de interés en escalas centrales de D(x, y, σ), ya que no existen imágenes vecinas en las escalas laterales.

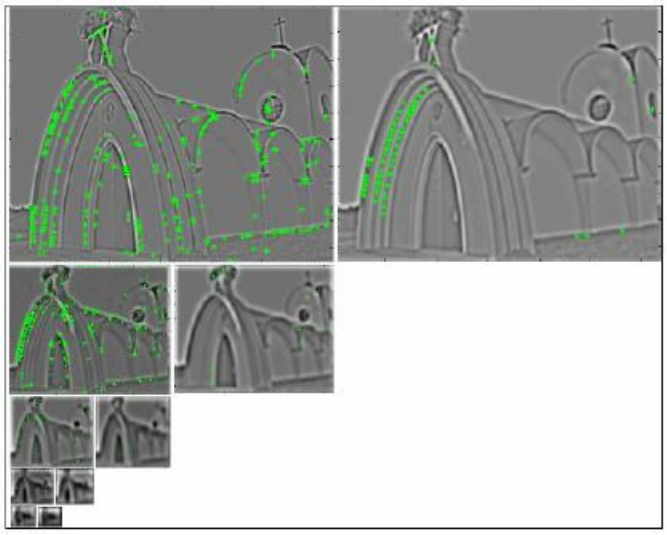

Figura 5.4. Puntos característicos detectados [10]

5.1.2. Localización de puntos característicos

La segunda fase del método SIFT se centra en almacenar toda la información de cada punto característico. Esto es, para cada punto de interés que se encuentra, se guarda a qué escala y octava de la pirámide pertenece y su posición dentro de la imagen correspondiente. Además, estos datos van a permitir descartar algunos a través de dos criterios: bajo contraste y localización a lo largo de bordes.

Los puntos que se descartan debido a estas dos condiciones no interesan porque serían muy sensibles al ruido.

5.1.3. Cálculo de la orientación

Aquí el objetivo será calcular las orientaciones de cada punto característico. Cuando se tengan, se podrán construir los descriptores, ya que éstos son referenciados a sus orientaciones y, por tanto, se conseguirá la invariancia a la rotación. Para ello se define una región alrededor del punto que se va a calcular su orientación, y a cada uno de los pixeles se le calcula su gradiente.

La imagen utilizada para calcular los gradientes, es la imagen de la pirámide L donde se detectó el punto de interés que esté siendo analizado.

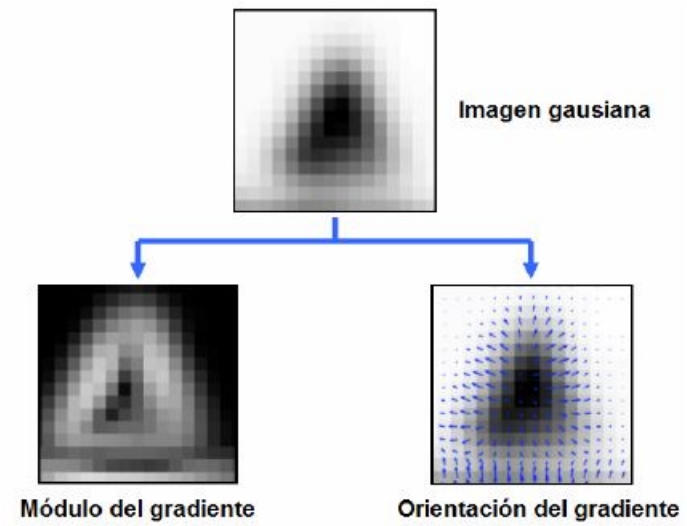

Figura 5.5. [10]

Después de haber obtenido la orientación del gradiente, se agrupa la información en forma de histograma, uno para cada punto característico. Los picos más altos de cada histograma son las direcciones que dominan de los gradientes locales, y por lo tanto, la orientación final del punto de interés. Normalmente va a ser de interés el pico más alto, pero algunas veces no va a ser así. A veces habrá que quedarse con un pico que sea al menos una altura mayor del 80 % del pico principal.

5.1.4. Descriptor de puntos característicos

Los parámetros hallados hasta ahora forman un sistema de coordenadas 2D que describe localmente cada región de la imagen. El siguiente paso va a consistir en obtener un descriptor para cada zona de interés. Cada descriptor se va a

obtener utilizando toda la información obtenida hasta este momento.

En este punto es necesario hacer unas modificaciones para conseguir que tenga mayor robustez ante cambios de iluminación. El objetivo es que sea invariante a tres tipos de variación:

- Luminosidad.
- Contraste.
- No linealidades

5.1.5. Cálculo de correspondencias. Matching

Como se ha visto anteriormente, un descriptor es un conjunto de elementos que indican las orientaciones más importantes alrededor de un punto de interés. Cuando dos imágenes tienen descriptores muy similares, es posible que ambos describan la misma zona y sean, por tanto, correspondientes.

Hay que tener especial cuidado, ya que si no se ponen las restricciones necesarias, el coste computacional puede ser muy elevado llegando a ralentizar la ejecución del programa.

En la figura 5.6 se muestra un ejemplo de fucnionamiento del SIFT. En ella se pueden observar los diferentes puntos característicos hallados, y los descriptores que son los que se unen con el matching.

Figura 5.6. SIFT

5.2. SURF. Speeded Up Robust Features

Es un detector que fue presentado en 2006, basado en Sift. Con este método se pretende ahorrar coste computacional. Guardan mucha similitud, pero presentan notables diferencias. En comparación con el Sift presenta ventajas como velocidad de cálculo superior y mayor robustez ante transformaciones en la imagen.

Para obtener estas mejoras, se reduce la dimensión y complejidad de los descriptores obtenidos. Lo hace de tal manera que éstos continúen siendo suficientemente característicos e igualmente repetitivos.

Detección de puntos de interés

El detector está basado en la matriz Hessiana. Se usa esta matriz debido a la velocidad de cálculo y su precisión. En este caso, debido a la limitación de los filtros gaussianos, se van a usar otro tipo de filtros que aproximan las derivadas parciales de segundo orden de las gaussianas que pueden ser evaluados de manera muy rápida usando imágenes integrales.

Asignación de la orientación

Este paso otorga al descriptor invarianza ante la rotación, dando a cada punto de interés una orientación.

5.2.1. Extracción del descriptor

Para obtener el descriptor, lo primero que se hace es construir una región cuadrada alrededor del punto de interés con una orientación determinada.

En primer lugar se tiene una región de un tamaño determinado. Este tamaño se va reduciendo progresivamente. Para cada nueva sub-región se calculan características en puntos de muestra separados una cierta región que determinan la dirección de la reducción.

5.2.2. Cálculo de correspondencias: Matching

La última etapa del detector va a consistir en unir los puntos de interés hallados en dos imágenes consecutivas. Cada punto de interés de la primera imagen será comparado con los de la segunda imagen.

En la figura 6.2 se pueden ver los puntos característicos y los descriptores unidos con matching.

Figura 5.7. SURF

5.3. Comparativa Sift vs Surf

Después de haber trabajado con ambos métodos se ha podido comprobar el funcionamiento de cada uno de ellos, observando los resultados que proporcionan. A raíz de estos resultados, se puede hacer una comparativa que ha llamado la atención.

La primera es que para la localización y detección de matrículas, Surf aporta mayor precisión y robustez que Sift.

La segunda comparativa que se extrae indica que Surf supone una carga computacional menor, y por tanto menor tiempo de procesamiento que Sift.

Capítulo 6
Desarrollo de la aplicación

En el presente capítulo se van a explicar los pasos seguidos para el desarrollo de la aplicación. Para ello se van a exponer todos los algoritmos que han sido utilizados e implementados.

El proceso seguido tiene un carácter secuencial, de modo que para explicar todo el programa se procederá paso a paso siguiendo la ejecución del programa, donde se pueden separar tres fases principales:

- Algoritmo que busca la ubicación de la matrícula en la imagen. Una vez se tiene su ubicación, se recorta.
- Algoritmo para tratar la imagen, de modo que se pueda aplicar un algoritmo OCR.
- Algoritmo para la extracción de los caracteres de la matrícula.

En la figura 6.1 se muestra el procedimiento seguido para conseguir los caracteres, mostrando paso a paso las etapas para conseguir la detección automática.

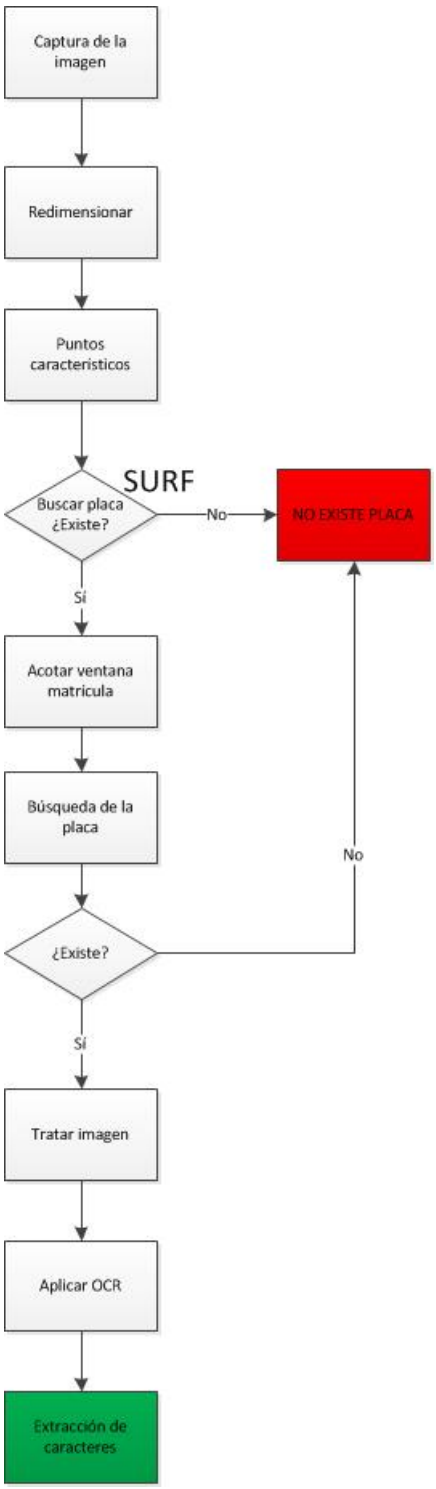

Figura 6.1. Diagrama de flujo de la aplicación

En primer lugar se realiza una captura de la imagen y se la redimensiona. Una vez se tiene la imagen, se le aplica un algoritmo para extraer los puntos característicos (SURF). Si no encuentra ninguno, determina que no existe la placa; si encuentra algún punto, se acota la ubicación de la matrícula en torno a

los puntos encontrados y se recorta.

Una vez acotada la ubicación, se vuelve a buscar la matrícula dentro de la imagen recortada y se comprueba de nuevo si existe. En caso de que no haya coincidencia, determina que no existe. Si la encuentra, comienza a tratar la imagen para poder aplicar a continuación un algoritmo OCR y poder extraer los caracteres de la matrícula.

6.1. Captura de la imagen

El primer paso para la consecución del objetivo final es obtener las imágenes de las que se quieren obtener los caracteres. Debido al empleo de las técnicas de extracción de características locales, la captura se puede hacer en un amplio abanico de posibilidades: diferentes ángulos, iluminaciones, etc. También se pueden tomar a distancias variables, pero dentro de un rango limitado previamente, ya que si se toma a una distancia grande, los resultados obtenidos no tendrían la calidad esperada y la fiablidad del sistema disminuye.

Una cámara fototgráfica digital de 3.1 Megapíxeles ha sido el principal instrumento de trabajo para conseguir las instantáneas, aunque también se ha utilizado un *smartphone*. No obstante, también se han tomado fotos con cámaras de diferentes resoluciones para comprobar si el programa creado funciona correctamente con estas imágenes, de forma que se tendría una aplicación compatible con todo tipo de tamaño de imágenes.

Se han utilizado varios entornos ambientales y con distinta luminosidad para obtener las imágenes. Pensando en una mayor fiabilidad del programa, la mayoría de las instantáneas han sido tomadas en el interior de un garaje donde hay menor luminosidad. No obstante también se ha trabajado con imágenes tomadas en el exterior, ya que ello permite abarcar un rango mucho más amplio de aplicación.

La flexibilidad es una característica imprescindible, por ello se han tomado capturas desde diferentes perspectivas. Gracias a ello el sistema puede trabajar con diferentes ubicaciones, consiguiendo que la ubicación final del sistema pueda ser variable.

Otro aspecto importante a tener en cuenta, es que para trabajar siempre en las mismas condiciones lo primero que se va a hacer nada más obtener la imagen es redimensionarla a un tamaño fijo. Esto va a permitir al sistema que el resultado final de la aplicación sea invariante a cambios en la cámara. El tamaño que se fija

es de 640 x 480 píxeles.

6.2. Programa de la aplicación

Una vez que ya se ha tomado una imagen para trabajar sobre ella, el primer paso es locacalizar la ubicación de la placa de la matrícula dentro de la misma. Para obtener una idea aproximada de su ubicación, se trabaja con los extractores de características locales SURF.

6.2.1. Búsqueda de características locales. SURF

El primer paso para el desarrollo de la aplicación, es saber donde está situada la matrícula dentro de la imagen obtenida. Para ello se utiliza la técnica SURF, la cual extrae las características locales de una imagen. Para la extracción de las características locales se barajaron dos métodos: SIFT y SURF. Al final la elección fue el segundo método, ya que el coste computacional era menor, además de proporcionar mejores resultados.

Para aplicar esta técnica, en primer lugar hay que tener un patrón, a partir del cual buscar puntos característicos que también aparezcan en la imagen capturada. En esta imagen, además de la matrícula, también aparece una parte del vehículo, tal como se muestra en la figura 6.2.

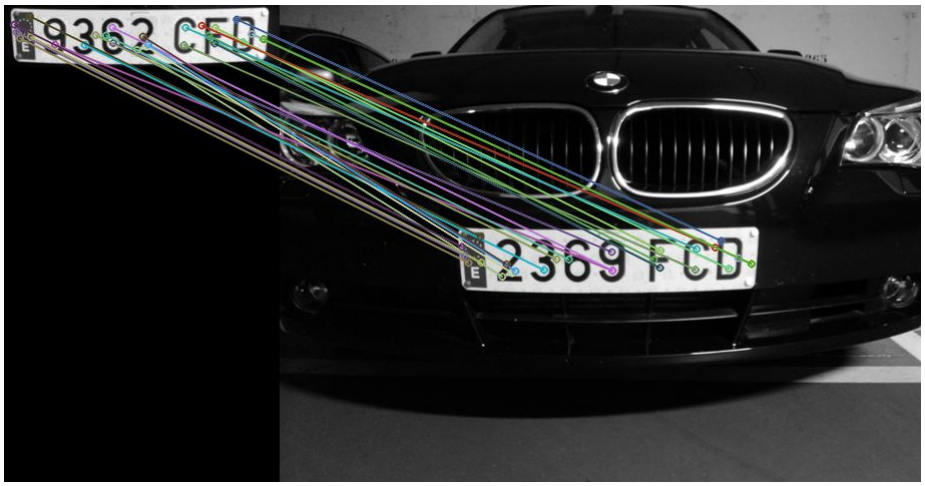

Figura 6.2. Puntos característicos que coinciden

Para poder aplicar el algoritmo SURF, se plantearon diferentes imágenes que pudieran utilizarse como patrón.

En un primer lugar se planteó la utilización del *patrón E* que se muestra en la figura 6.3 y aparece en las matrículas.

Figura 6.3. Patrón E

Pero este patrón fue descartado por tres motivos. El primero y principal fue que no todas las matrículas disponen de esta reseña, las matrículas más antiguas carecen de ella como se muestra en la figura 6.4.

Figura 6.4. Matrícula sin el patrón E

El segundo motivo fue que para poder extraer los puntos característicos de este patrón, era necesario que tanto la imagen capturada como el patrón tuviesen una alta resolución. Esto implica que la aplicación tuviera un tiempo de ejecución mayor del esperado. El tercer y último motivo de su no utilización fue que en una gran cantidad de imágenes no funcionaba de manera correcta como se muestra en la figura 6.5, de modo que la eficiencia de la aplicación era muy baja.

Figura 6.5. Correspondencia mala con patrón E

Debido a que la primera alternativa no ofrecía el funcionamiento deseado, se optó por buscar otra. Para no tener el problema de que no se puedan detectar las matrículas sin el *patrón E*, se decidió utilizar una placa de matrícula como patrón, la cual se muestra en la figura 6.6.

Figura 6.6. Patrón utilizado

Utilizando este patrón se consigue extraer una gran cantidad de puntos característicos, los cuales también aparecen en la inmensa mayoría de las imágenes capturadas. La aparición de dichos puntos, hace que se pueda obtener una idea aproximada de la ubicación de la placa, como se ve en la figura 6.7.

Figura 6.7. Puntos característicos que coinciden

Este patrón almacena una gran cantidad de información, la cual a la hora de hacer coincidir los puntos con la imagen es muy difícil que falle. Además es muy común que se obtenga más de un punto que coincida. Se ha utilizado el patrón con los caracteres 9362CFD porque después de probar diferentes placas de matrículas como patrón, es el que mejores resultados ofrecía. En caso de utilizar otro patrón (figura 6.8), se observa como el punto característico detectado más a la izquierda queda fuera de la matrícula y el área del primer corte es tan grande que se van a detectar muchas rectas fuera de la matrícula no pudiendo realizar bien el segundo corte; mientras que en la figura 6.9 se ve como el resultado es el correcto.

Figura 6.8. Puntos característicos fuera de la placa

Figura 6.9. Puntos característicos dentro de la placa

Una vez que se ha justificado el uso del patrón utilizado en vez de la *reseña E*, hay que explicar el algoritmo que determina los puntos característicos que hay que casar en la imagen.

El algortimo SURF para detectar puntos trabaja con imágenes integrales. Para ello es recomendable que tanto la imagen capturada como el patrón estén en escala de grises, ya que la utilización de imágenes a color no aporta información adicional y se tienen el triple de datos. Esta operación es muy sencilla, ya que OpenCV permite cargar una imagen de color en escala de grises.

Una vez cargadas las imágenes, lo primero que hay que hacer es detectar los puntos característicos de ambas imágenes, los cuales aparecen en la figura 6.10.

Figura 6.10. Puntos característicos

Para ello se emplea la función SURF, la cual es llamada incluyendo la librería correspondiente. Los puntos obtenidos son covariantes al traslado, rotado y reescalado de la imagen. Para extraerlos hay que introducir un umbral llamado *Hessiano*, el cual hace la función de parámetro de la función SURF. Cuanto mayor sea el valor de este umbral, menor será la cantidad de puntos característicos que se extraen, de modo que es aconsejable utilizar un valor que permita obtener puntos característicos en todas las imágenes, pero que no proporcione un exceso de los mismos. Todos los puntos obtenidos son guardados en dos vectores de puntos: uno para el patrón y otro para la matrícula.

En segundo lugar, lo que se hace es calcular los *descriptores* de cada una de las imágenes, los cuales son almacenados en otro vector. Estos descriptores son los puntos que más información contienen de los puntos característicos.

Una vez obtenidos los descriptores de ambas imágenes, se realiza un *matching* entre los mismos como se observa en la figura 6.11. Para ello se define el tipo de descriptor que se quiere utilizar, en este caso el *Flann*. Este descriptor llama a los métodos de búsqueda más cercanas para encontrar las mejores coincidencias. Flann es bastante más rápido que otro métodos de características similares.

Figura 6.11. Matching de los descriptores

Este método en ocasiones genera una cantidad de descriptores que no interesan, ya que se encuentran fuera de la placa de la matrícula como se muestra en la figura 6.12. Para discriminar estos puntos, lo que se hace es calcular la mínima distancia que hay entre los dos puntos más cercanos. Una vez se tiene esta distancia, se eliminan todos aquellos puntos que no tengan ningún otro punto a menos de 1.5 veces la distancia mínima. Después se unen aquellas coincidencias llamadas *buenas*. Estas coincidencias son aquellas cuya ubicación se encuentra en la matrícula de la imagen capturada. Esta operación se realiza de forma iterativa hasta 100 veces con el fin de que en cada iteración se vayan eliminando descriptores que puedan producir resultados indeseados.

Una vez que se obtienen todas las coincidencias *buenas*, se guardan en un vector de puntos. Este vector puede contener tanto un punto, como más de uno formando una nube. En la imagen de la izquierda de la figura 6.13 se muestra un ejemplo del primer caso, mietras que a la derecha se muestra un ejemplo del segundo.

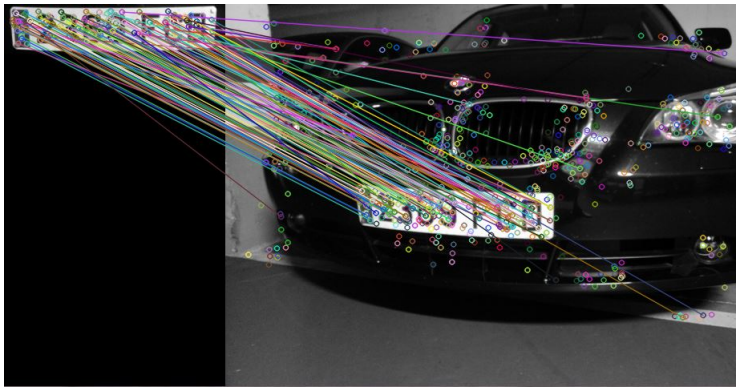

Figura 6.12. Matching con descriptores malos

Figura 6.13. Diferentes cantidades de descriptores

Cuando se han unido todos los descriptores, se vuelve a aplicar un filtro para eliminar todos aquellos puntos que puedan haber quedado fuera de la matrícula. Para ello se crea una función que elimina los puntos que a su alrededor no tengan por lo menos dos puntos que estén a una distancia mínima. En la figura 6.14 se muestran los descriptores obtenidos una vez se han aplicado todos los filtros a la figura 6.12. Además, se crea un vector de puntos pero que sólo contiene aquellos descriptores que están dentro de la imagen tomada. Este vector va a ser utilizado a continuación para obtener una ubicación aproximada de la matrícula y recortarla.

Figura 6.14. Descriptores buenos

6.2.2. Acotar el espacio de búsqueda

Una vez que se tiene el vector de puntos que contiene la nube de puntos de la imagen, se utiliza para cortar la imagen en torno a una ubicación aproximada de la matrícula.

En ocasiones, el vector de puntos puede contener un solo punto. Esto obliga a que el corte que se aplica a la imagen sea diferente en función de la ubicación de los puntos carcterísticos. Por ello, para cortar la imagen se va a trabajar con un solo punto, que en el caso de que haya más de uno va a ser el que esté más a la izquierda. Si sólo hay uno se va a trabajar con él.

Al trabajar con un punto, hay que saber cuál es su ubicación dentro de la imagen. Para ello lo que se hace es dividir la coordenada x de dicho punto entre la anchura de la imagen (número de columnas). Al dividirlo se obtiene un valor que oscila entre 0 (izquierda) y 1 (derecha).

Una vez se sabe su ubicación, se halla la anchura del corte en función de donde esté situado. De modo que si está a la izquierda, se hace el corte desplazando el punto un poco a la izquierda y bastante a la derecha. Mientras que si está a la derecha se hace al revés.

En la figura 6.15 se puede observar como el primer punto está hacia la izquierda, mientras que en la figura 6.16 se puede ver como está más a la derecha, quedando una zona de corte bastante similar.

También existen aquellos casos en los que algún descriptor ha quedado fuera de la matrícula y no se ha eliminado. En estos casos, el área de corte es muy grande (figura 6.17) y no se va a poder llegar a reconocer los caracteres, ya que el siguiente corte no lo va a poder hacer de manera correcta.

Figura 6.15. Corte con punto a la izquierda

Figura 6.16. Corte con punto a la derecha

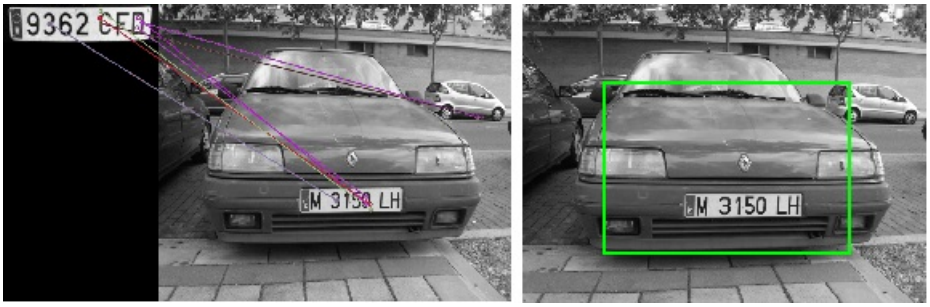

Figura 6.17. Área de corte grande

Con este procedimiento se discrimina la anchura del corte. Para determinar la altura, se crea una tolerancia que se suma al punto con mayor valor en la coordenada y, y se resta al punto con menor valor en la coordenada y. La tolerancia creada es lo suficientemente grande para que sea cual sea el valor en y de los puntos, sea capaz de recortar siempre toda la altura de la matrícula.

Una vez que se obtiene un cuadrado que incluye a la matrícula, hay que recortar la imagen. En la figura 6.18 se ve el corte realizado a la figura 6.15, mientras que en la figura 6.19 se observa el corte de la figura 6.16.

Figura 6.18. Corte 1

Como se puede observar en las figuras 6.18 y 6.19, los resultados obtenidos no son malos, aunque se muestra como alrededor de la matrícula queda algún trozo que corresponde al vehículo.

Figura 6.19. Corte 2

Esos trozos no deseados, hacen que no se pueda tratar la instantánea recortada para aplicarla directamente un algoritmo OCR. Si se quisieran tratar esas imágenes tal cual están, darían muchos problemas ya que aparecería ruido y contornos indeseados como se muestra en la figura 6.20. La existencia de estos contornos implica que al aplicar un algoritmo OCR alguno de ellos pueda ser reconocido como caracter, dando un resultado incorrecto.

Figura 6.20. Imagen con contornos indeseados

Para acotar de una mejor manera el espacio de búsqueda, se vuelve a cortar la imagen. En este caso se trazan unas rectas horizontales para tratar de detectar los límites superior e inferior de la matrícula.

Después de realizar una gran cantidad de pruebas con diferentes imágenes, se llega a la conclusión de que el corte trazando rectas verticales no ofrece buenos resultados ya que al aparecer una gran cantidad de rectas, se hace muy difícil determinar cual debe ser el corte para cada uno de los casos. En la figura 6.21 se muestran dos casos diferentes.

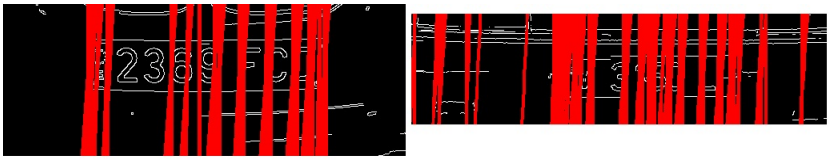

Figura 6.21. Gran cantidad de rectas verticales

En algunos casos, las rectas verticales sí que consiguen delimitar bien la matrícula, no habiendo problema para recortarla. Pero hay casos en los que las

rectas aparecen fuera de la matrícula, haciendo imposible elegir aquellas rectas verticales para delimitar el corte.

Se observa que con el primer corte que se realiza, las matrículas quedan bien acotadas por sus laterales, quedando visible una pequeña parte del parachoques.

La obtención de las rectas horizontales se realiza a través de la transformada de Hough. Para conseguirlo, lo primero es obtener los contornos aplicando Canny. Una vez se tienen los contornos, se detectan todas las rectas y se guardan en un vector. Para detectarlas hay que introducir un parámetro, el cual indica cuál es la longitud mínima de las rectas a obtener. Este parámetro no puede tener un valor elevado, ya que se podría dar el caso de no detectar ninguna recta. Tampoco es conveniente un valor bajo ya que se obtendría una gran cantidad de rectas como se observa en la figura 6.22.

Figura 6.22. Gran cantidad de rectas

Para conseguir sólo las rectas horizontales, se introduce un filtro para dibujar aquellas que están en un rango de grados pequeño, pero suficiente para cubrir la mayoría de los casos. Debido a que algunas imágenes pueden tener cierta inclinación, se buscan todas aquellas rectas cuya inclinación esté entre -5º y +5º.

Figura 6.23. Rectas

En las figuras 6.23 y 6.24 se muestran dos imágenes de las rectas horizontales

que se extraen en diferentes imágenes, una vez que se aplican los filtros de longitud e inclinación.

Figura 6.24. Rectas

Una vez que se obtienen las rectas, hay que elegir aquellas que van a delimitar el espacio a recortar. El primer paso es saber cuál es el punto medio entre las rectas que se encuentran más alejadas entre sí. El método para obtener este valor es crear un algoritmo para saber el valor del máximo y del mínimo punto y de las rectas. Teniendo el valor de estos puntos, se halla el punto medio de las rectas, el cual debería estar situado en la matrícula.

En algunas imágenes, se van a detectar rectas en la parte superior e inferior de la matrícula como se muestra en la figura 6.25, mientras que en otras sólo va a encontrar rectas en uno de los extremos tal como se ve en la figura 6.26.

Figura 6.25. Rectas superiores e inferiores

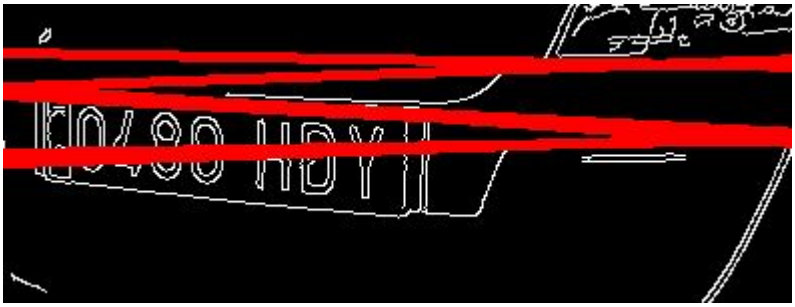

Figura 6.26. Rectas superiores

Hay que discernir entre aquellas imágenes que tienen rectas superiores e inferiores y aquellas que solo las tienen en uno de los extremos. Para conseguir diferenciarlas lo que se hace es trabajar con la media anteriormente obtenida. Si la media es próxima a la mitad de la altura de la imagen, significa que habrá rectas superiores e inferiores. Pero si la media se aleja bastante de la mediana, sólo habrá rectas superiores o inferiores.

Para los casos en los que existen rectas superiores e inferiores, se trabaja de la siguiente manera. Teniendo el punto medio, se buscan aquellas rectas que son inmediatamente superior e inferior al punto medio. Para hallar la recta superior, se crea una variable que contiene un primer valor, el cual se actualiza por el valor de otra recta si ésta tiene un valor menor que el actual y dicho valor es superior al punto medio hallado anteriormente. Para determinar la recta inferior se hace de la misma manera, pero la recta encontrada ha tener un valor inferior al actual, además de ser su valor inferior a la media.

Una vez que se tienen las rectas superior e inferior, hay que saber que en la mayoría de los casos la matrícula va a tener una inclinación. Por ello, las rectas obtenidas no van a tener una pendiente nula, sino que va a tener un pequeño valor. De modo que para elegir un punto de corte se elige la media de los valores extremos de la recta y se suma una pequeña tolerancia. Con esta tolerancia se asegura que la matrícula va a estar completamente contenida en el recorte. Este caso se puede observar en la imagen 6.27.

Figura 6.27. Corte con inclinación

Para el caso en que se detecta que sólo existen rectas superiores o inferiores, se sigue otro procedimiento. Se busca aquella recta que delimita la matrícula superiormente o inferiormente, la cual va a indicar la altura a la que realizar el corte.

Si se determina que se trata de una recta que delimita superiormente o inferiormente a la matrícula, se crea una tolerancia a partir de la posición en y de dicha recta para delimitar el rango inferior o superior. El resultado se puede ver en la figura 6.28.

Figura 6.28. Corte con rectas superiores

Observando las pruebas mostradas en las figuras anteriores, se puede concluir que cuando la matrícula está limitada por rectas superiores e inferiores el corte va a ser más ajustado que en el caso en que sólo se detectan rectas en uno de los extremos. Se debe a que se puede ajustar de manera más precisa la altura a cortar por tener dos puntos exactos, mientras que si sólo existe una de las rectas hay que trabajar con una tolerancia, la cual no se puede ajustar en exceso porque en algunos casos haría un corte malo.

6.2.3. Tratamiento de la imagen

Hasta ahora se ha conseguido recortar la imagen de tal manera que la resultante contiene la matrícula y en algunos casos una pequeña parte del vehículo. El objetivo ahora es tratar la imagen de tal manera que se puedan reconocer los caracteres que en ella aparecen.

El algoritmo empleado para tratar la imagen consiste en una binarización adaptativa, seguido de un recorte para acotar más la matrícula.

La binarización que se realiza es adaptativa porque el algoritmo creado tiene como fin poder reconocer matrículas independientemente de la luminosidad de la escena, contando con que haya un mínimo de luz. En caso de hacer una binarización con umbral fijo, sólo se obtendrían buenos resultados para una iluminación determinada. Mientras que al usar una binarización adaptativa se puede trabajar con diferentes rangos de luminosidad.

En la figura 6.29 se observa como las imágenes obtenidas utilizando la binarización adaptativa pueden presentar ruido. En la figura 6.30 se muestra otro caso en que la matrícula puede quedar muy bien. También se puede observar en la figura 6.31 como en algunos casos los caracteres no quedan rellenos por completo.

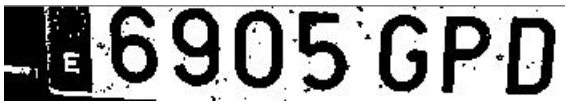

Figura 6.29. Matrícula con ruido

Figura 6.30. Matrícula binarizada limpia

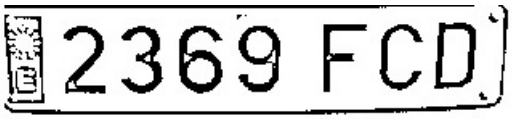

Figura 6.31. Matrícula con huecos en caracteres

Si existe ruido o los caracteres no están rellenos por completo, el algoritmo OCR no se puede aplicar directamente, ya que los resultados obtenidos no serían satisfactorios. Esto implica que hay que limpiar el ruido de las matrículas, además de rellenar aquellos caracteres que no queden rellenos por completo.

Para ello lo primero que se hace es aplicar un algoritmo para detectar todas aquellas zonas de color negro indeseadas (ruido). Este algoritmo lo que hace es aplicar Canny para detectar todos los contornos que aparecen en la imagen, ya sea ruido, caracteres, etc. Una vez se tienen todos estos elementos, lo que se hace es filtrar aquellas zonas cuyo área es menor que uno previamente fijado. Si el área calculado es menor, pinta todo el área que encierran los contornos de color blanco. En la figura 6.32 se puede observar el resultado de aplicar este filtro.

Figura 6.32. Matrícula sin ruido

Figura 6.33. Matrícula con caracteres rellenos

El segundo problema que se ha visto que puede aparecer es que los caracteres no estén rellenos por completo, surgiendo la necesidad de tener que aplicar un algoritmo muy similar al anterior. De nuevo se vuelven a detectar los contornos de los agujeros blancos por Canny. Si el área detectado es menor que un valor determinado, el hueco encerrado por el contorno se pinta

de negro. El resultado de aplicar este algoritmo se puede observar en la figura 6.33.

En este punto surge la cuestión de por qué hacer un tercer corte a la placa. Se debe a que al realizar una binarización adaptitva, se obtiene en algunos casos la matrícula con ruido externo que no se ha podido eliminar o con contornos negros alrededor, como se muestra en la figura 6.34.

Figura 6.34. Matrícula con ruidos externos o contornos

Al obtener estas imágenes, es muy difícil que el algoritmo OCR reconozca de forma correcta los caracteres que hay en la matrícula, teniendo que realizar un último recorte para delimitar de la mejor manera posible la placa.

Para realizar este recorte, lo que se hace es recorrer toda la imagen, y buscar cuatro puntos que delimiten la ubicación de la matrícula. Estos puntos deben pertenecer a una línea de puntos que al menos tenga una cierta longitud, además de producirse un cambio de valor en ellos (0 a 1 ó 1 a 0), lo cual implica que pertenecen a los laterales de la placa. Esto es, se recorre la imagen empezando por el límite izquierdo hacia la derecha y se obtiene la posición del punto que contiene el primer píxel de color blanco (píxel que corresponde a la placa). Esta operación se realiza de la misma manera para cada uno de los laterales de la imagen, obtieniendo un punto por lado como se muestra en la figura 6.35.

Figura 6.35. Puntos detectados

Una vez que se tienen estos puntos, hay que hallar las rectas que van a delimitar el corte. Las rectas por las que se va a cortar son horizontales y verticales, pasando por estos puntos a los que se añade un margen igual a la tolerancia establecida como se muestra en la figura 6.36. Para el punto de la izquierda, la tolerancia es mayor que para el resto de puntos, para tratar de eliminar la *reseña E*, que, como se ha visto, puede ser reconocida por el algoritmo OCR en bastantes casos.

Figura 6.36. Puntos detectados con las rectas de corte

Una vez se realiza el recorte, la imagen queda como se muestra en la figura 6.37.

Figura 6.37. Imagen cortada

La figura 6.38 es un claro ejemplo de ciertos casos en los que la placa tiene tal curvatura que es imposible conseguir eliminar toda la escena negra sin cortar ningún trozo de caracter como se observa en la figura 6.39.

Figura 6.38. Placa rotada

Figura 6.39. Imagen con zonas negras

En este caso no afecta, ya que esa pequeña zona en negro la detectaría como caracter especial y no como letra. No habría problema, porque como se va a comentar en el reconocimiento de caracteres, se ha incluido un filtro en el OCR para que solo detecte letras y números.

Este corte es realmente necesario, porque es habitual que los extremos de las placas los pueda detectar como caracteres. En algunos casos los reconoce como una C, como una J, etc. De modo que aplicando esta técnica se pueden llegar a obtener los resultados deseados con mayor fiabilidad.

6.2.4. Reconocimiento de caracteres. Aplicación de algoritmo OCR

Una vez que se tiene la imagen tratada, se pretende obtener los caracteres que aparecen en la placa de la matrícula. Además, los caracteres reconocidos

serán guardados como texto para que puedan ser utilizados posteriormente. No se ha desarrollado un algoritmo propio para reconocer caracteres, sino que se ha desarrollado una librería propia a partir de un motor OCR (*Optic Character Recognition*) libre.

Tesseract es el motor OCR libre que se ha empleado, el cual fue creado originalmente por Hewlett Packard, que actualmente es desarrollado por Google y distribuido bajo la licencia Apache. Esta licencia permite al usuario que se pueda usar libremente para cualquier fin, distribuirlo, modificarlo y distribuir versiones modificadas de ese software.

Este motor OCR no es el único que existe de manera libre, ya que en la actualidad hay varios. Pero la elección de este motor se debe a que después de haber probado algunos de ellos fue con el que mejores resultados se obtuvieron. Además, este motor OCR es uno de los más utilizados, lo que implica que tanto la documentación existente como la actividad de los usuarios en la red es mayor que para otro software similar, lo que favorece un desarrollo más ágil de la aplicación.

Tesseract está preparado para tratar la imagen y poder extraer los caracteres, ya que es capaz de binarizar, etiquetar los caracteres e identificarlos. Pero para que los resultados sean los esperados, tiene que ser una imagen muy limpia y perfectamente contrastada. Por ello lo mejor es realizar un tratamiento previo y propio a la imagen, para que los caracteres queden lo más resaltados posible.

Se trata de una potente librería, la cual dispone de una gran cantidad de funciones. Pero para este trabajo se ha creado una librería propia, la cual hace llamadas a las API del motor OCR. Una vez creada la librería, basta con hacer llamadas a las funciones que contiene, para obtener los resultados deseados. A continuación se van a exponer las funciones creadas para aprovechar los recursos de esta librería.

En primer lugar, hay que decir que se pueden reconocer números, letras y todo tipo de caracteres. Para este trabajo sólo interesa identificar números y letras, de modo que se ha limitado la búsqueda a obtener dichos caracteres. Se ha creado una función que sólo hace referencia a la extracción de caracteres y números por parte de Tesseract.

Además, puede funcionar en una gran cantidad de modalidades diferentes: texto en columna, texto en fila, texto como bloque, etc. Por lo que ha sido necesario crear otra función para determinar que tipo de texto tiene que reconocer. Para es-

te trabajo se le indica que va a ser una línea de texto con caracteres independientes.

Tesseract dispone de una base de datos muy extensa donde busca los caracteres. Esta base de datos contiene todos los caracteres en una gran diversidad de formatos de letras: Arial, Times new Roman, etc.

Para detectar que caracter es cada uno de los caracteres que aparece en la matrícula, lo que hace es extraer sus características y compararlas con las que tiene alojadas en la base de datos. Para la detección se utilizan varias propiedades, como pueden ser la rectangularidad, el número de Euler, etc.

Todos los caracteres extraídos se guardan en un fichero de texto para que el usuario final los pueda utilizar como quiera. En las figuras 6.40, 6.41, 6.42 y 6.43 se pueden observar ejemplos de los resultados obtenidos con la matrícula al lado.

Figura 6.40. Matrícula 2369 FCD

Figura 6.41. Matrícula M 3150 LH

Figura 6.42. Matrícula 6905 GPD

Figura 6.43. Matrícula 0480 HDY

Existe también algún caso en el ángulo de captura de la imagen es bastante alto o que la imagen sea borrosa, haciendo que los caracteres no queden de forma

correcta al tratar la imagen y no puedan ser reconocidos de la manera esperada (figura 6.44).

Figura 6.44. Reconocimiento de caracteres

Capítulo 7
Resultados experimentales

En este capítulo se van a mostrar una serie de capturas de diferentes vehículos para probar el funcionamiento de la aplicación. Después de haber realizado las pruebas, se incluye una valoración a nivel personal del funcionamiento de la aplicación.

Una vez desarrollada la aplicación, se pone a prueba con el fin de obtener una valoración funcional de la misma. Para ello se van a utilizar diferentes tipos de fotografías, por ejemplo a diferentes distancias, diferentes tamaños (píxeles), diferentes ángulos, matrículas con suciedad, sombras, etc.

De este modo, se podrán mostrar tanto los fallos como los aciertos de la aplicación creada, exponiendo las causas de los fallos.

7.1. Imágenes que varían en la distancia

En primer lugar se van a mostrar una serie de imágenes, las cuales están tomadas a diferentes distancias. Con ello se pretende demostrar que, como ya se dijo, la aplicación funciona para un rango de distancias variable.

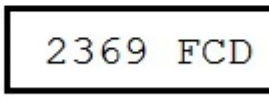

Figura 7.1. Distancia media

Figura 7.2. Distancia cercana

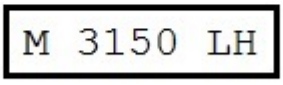

Figura 7.3. Distancia lejana 1

Figura 7.4. Distancia lejana 2

En las figuras 7.1, 7.2, 7.3 y 7.4, se observa como la aplicación funciona correctamente para diferentes distancias, que proporciona los resultados esperados.

En la figura 7.5 se puede apreciar otro vehículo cuya imagen se ha tomado lejana, pero además de ello, también con cierto ángulo produciendo los resultados esperados.

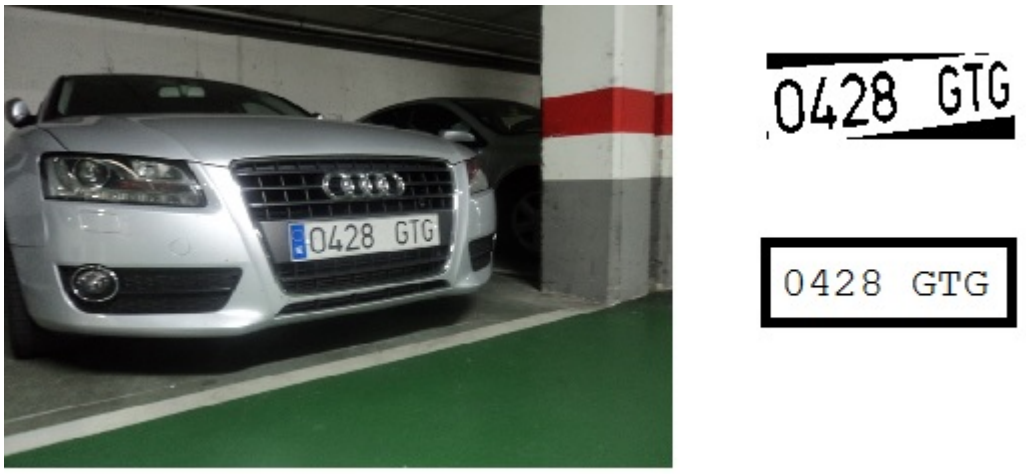

Figura 7.5. Distancia lejana y ángulo

En la figura 7.6 se muestra un ejemplo en el que la aplicación no produce ningún resultado. Esto se debe a que el coche está demasiado cerca de la cámara, no quedando bien enfocada la matrícula y cortando un trozo de la misma. Esto puede ocurrir ya sea por la ubicación del vehículo o por estar demasiado cerca de la cámara.

Figura 7.6. Distancia cercana

7.2. Imágenes con diferentes ángulos

A continuación se van a mostrar imágenes que han sido tomadas con diferentes ángulos, para corroborar que la aplicación funciona situando la cámara en diferentes perspectivas.

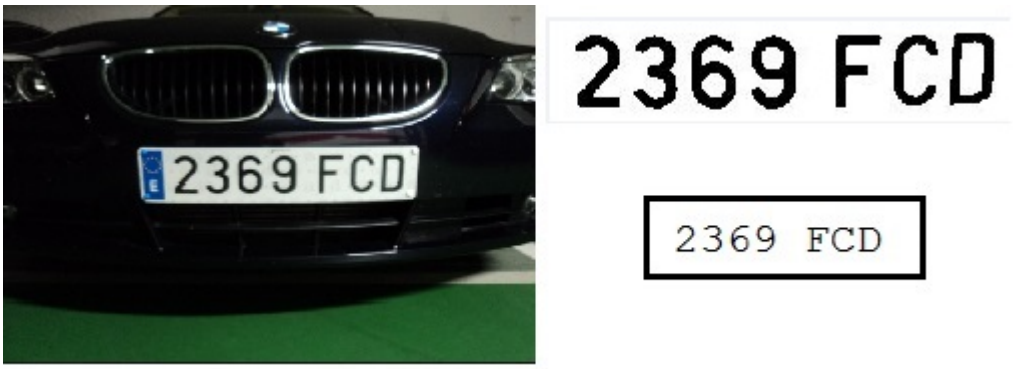

Figura 7.7. Imagen centrada

Figura 7.8. Imagen con cierto ángulo 1

En las figuras 7.7 y 7.8, se muestra que detecta las matrículas para diferentes ángulos.

En las figuras 7.9 y 7.10 se muestran dos capturas realizadas al mismo vehículo, pero con diferentes ángulos produciendo los resultados esperados.

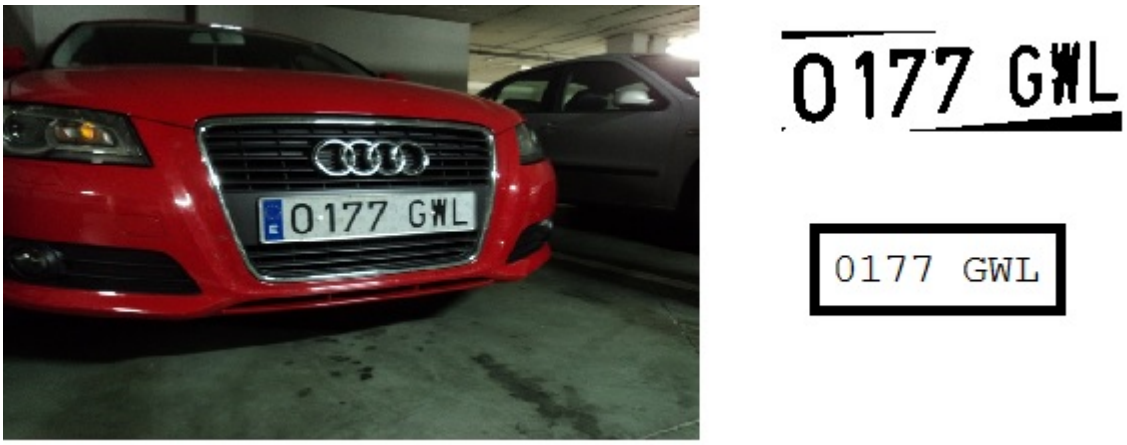

Figura 7.9. Vehículo 1 desde la izquierda

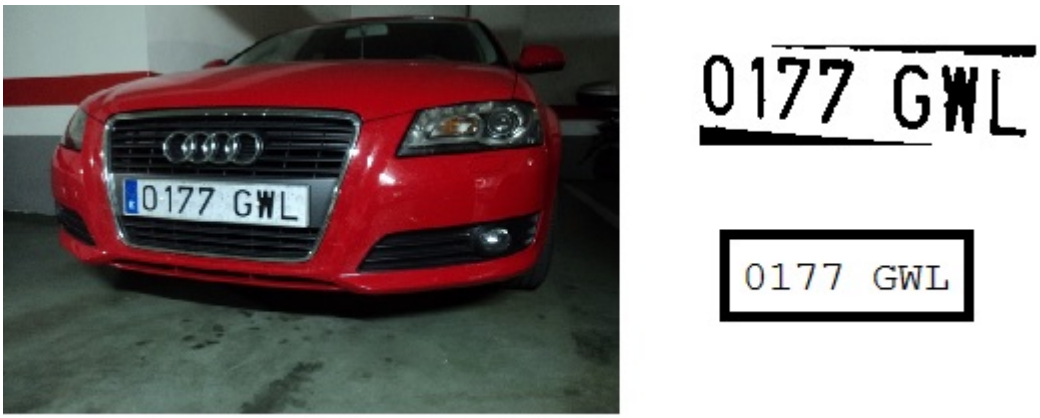

Figura 7.10. Vehículo 1 desde la derecha

Mientras que en en las figuras 7.11 y 7.12 se muestran otras dos imágenes de otro vehículo tomadas desde diferentes ángulos, produciendo un resultado correcto sólo en el primero de los casos. En la segunda imagen no detecta bien los caracteres debido al ángulo, además de que la matrícula queda algo borrosa.

Figura 7.11. Vehículo 2 desde la derecha

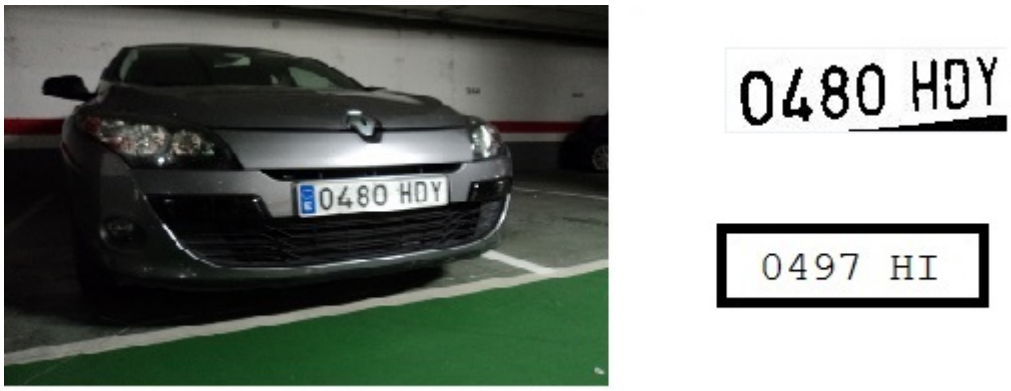

Figura 7.12. Vehículo 2 desde la izquierda

7.3. Imágenes con suciedad o mala visibilidad

En este apartado, se van a incluir imágenes que puedan tener suciedad en la matrícula, o en las que la visibilidad sea reducida. Gracias a estas pruebas se pueden dilucidar los umbrales para los algoritmos empleados.

Figura 7.13. Matrícula con suciedad

Figura 7.14. Matrícula con mala visibilidad 1

En las figuras 7.13 y 7.14 se observa como se detecta perfectamente una matrícula con suciedad.

En las figuras 7.15 y 7.16 se muestran imágenes del mismo vehículo pero con diferente ángulo. En ambos casos la visibilidad es reducida, pero se obtienen buenos resultados.

Figura 7.15. Vehículo 3 mala visibilidad desde izquierda

Figura 7.16. Vehículo 3 mala visibilidad desde derecha

En la figura 7.17 se observa como en la imagen, además de ser lejana, los caracteres quedan borrosos. En este caso la aplicación proporciona buenos resultados.

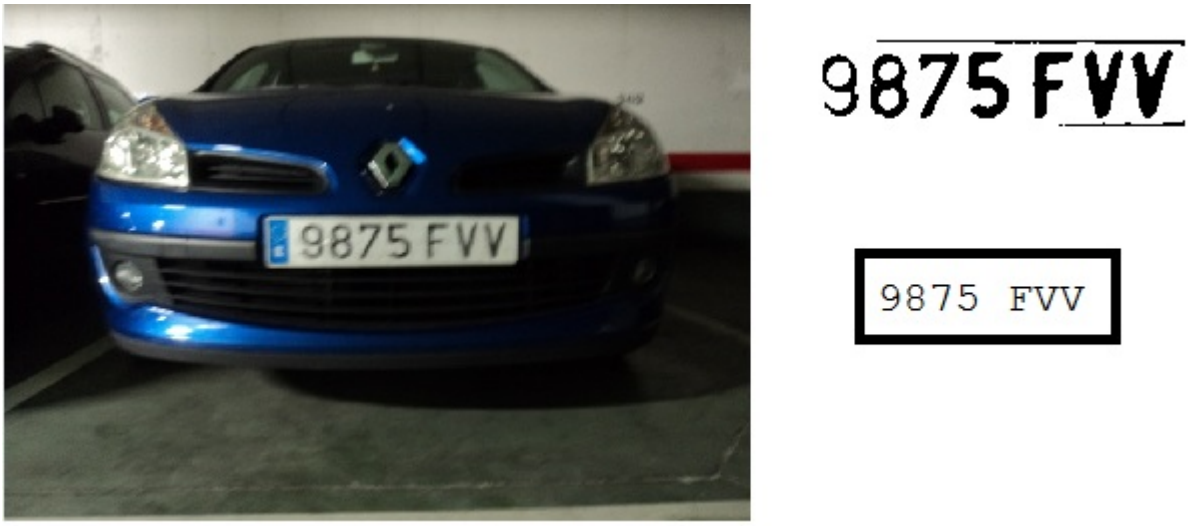

Figura 7.17. Caracteres borrosos

En la figura 7.18 se muestra como otra imagen con mala visibilidad no proporciona buenos resultados. Esta imagen tiene los caracteres borrosos, de modo que al tratarla no quedan como deberían quedar. Ello supone que la aplicación del algoritmo OCR no muestre los resultados deseados.

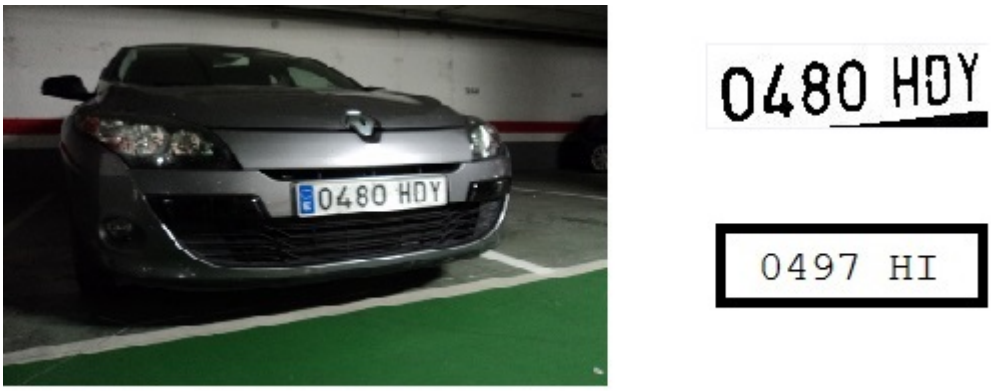

Figura 7.18. Matrícula con mala visibilidad 2

En la imagen 7.19 se observa como la aplicación no ofrece buenos resultados, ya que al realizar la binarización los caracteres quedan bastante dañados, no pudiendo ser mejorados tratando la imagen. Además, se puede observar como la matrícula está doblada en la parte inferior derecha, haciendo que haya un gran contraste de luz entre ese trozo y el resto de la matrícula.

Figura 7.19. Matrícula dañada

7.4. Diferentes tamaños de imágenes

Como ya se comentó en el capítulo anterior, se iba a trabajar con imágenes cyas dimensiones son de 640 x 480 píxeles. De modo que si alguna imagen tiene una resolución mayor que ella, se reduce a este tamaño para trabajar en igualdad de circunstancias.

A continuación se van a mostrar unas imágenes que se han tomado con una cámara de una resolución bastante superior a 640 x 480 píxeles, de modo que han tenido que ser redimensionadas.

Figura 7.20. Imagen con resolución alta

Figura 7.21. Imagen con resolución media

En las figuras 7.20 y 7.21 se observa como al redimensionar las imágenes, se obtiene un resultado correcto.

En las figuras 7.22 y 7.23 se muestran otras dos imágenes tomadas con una cámara superior a 640 x 480 píxeles. Estas imágenes al redimensionarlas proporcionan los resultados correctos.

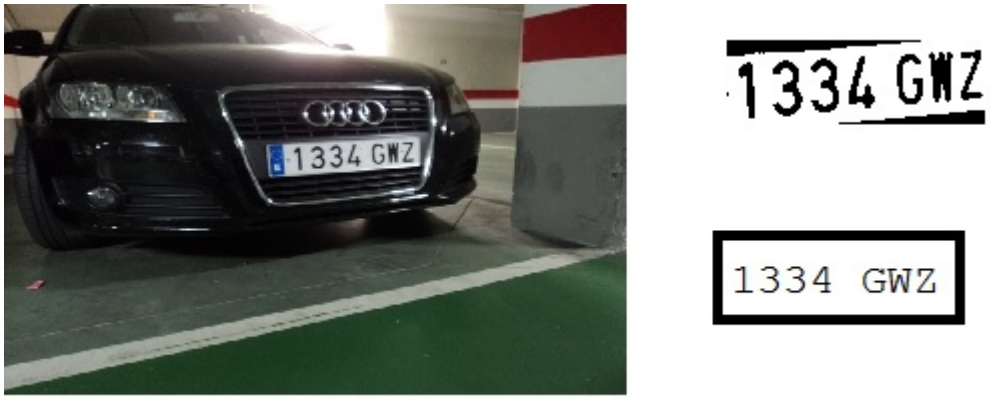

Figura 7.22. Imagen con mayor resolución 1

Figura 7.23. Imagen con menor resolución 2

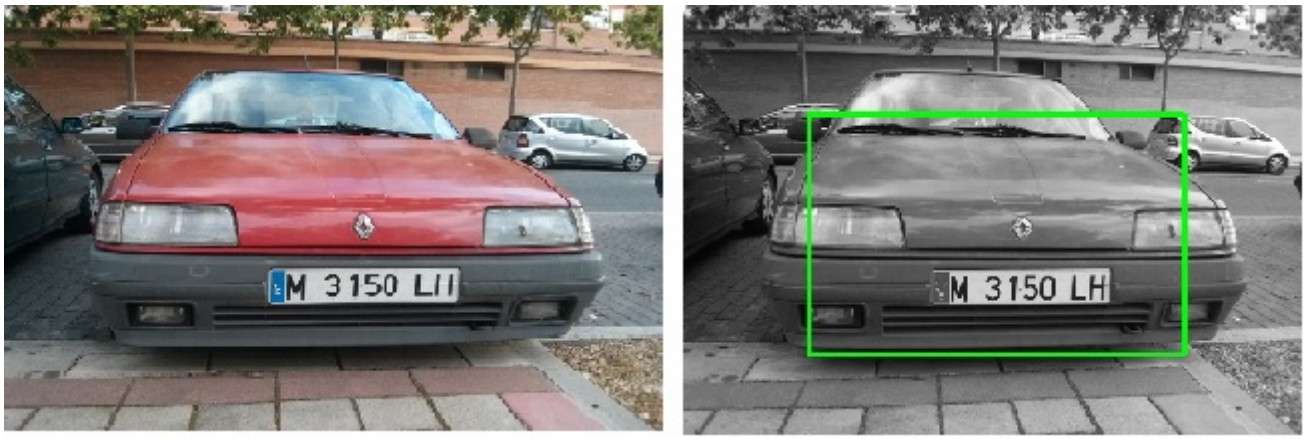

Figura 7.24. Imagen con mayor resolución

En la figura 7.24 no se obtiene el resultado correcto después de redimensionar la imagen. Se puede observar como en la imagen de la derecha el primer rectángulo de corte es muy grande, lo que implica que a la hora de hacer las líneas horizontales para acotar la matrícula aparecen una gran cantidad de rectas, siendo imposible hallar un buen área de corte. La causa principal reside en el redimensionamiento de la imagen, ya que se pierde información y hace que al extraer los puntos descriptores mediante SURF, alguno aparezca fuera de la placa de la matrícula, haciendo que el primer rectángulo de corte (color verde) sea muy grande.

7.5. Valoración de la aplicación

Después de probar la aplicación en una amplia y cuantiosa gama de imágenes, es difícil determinar un porcentaje exacto de funcionamiento correcto. Pero a

partir de las pruebas que se han realizado se puede decir que este porcentaje está en torno al 80 %.

Ante todo hay que destacar que se trata de un porcentaje bastante alto, ya que no se dispone de ninguna ayuda de iluminación externa. Con este tipo de ayuda, se consigue aumentar el porcentaje de detección de los caracteres. También hay que tener en cuenta las diferentes condiciones en las que se realizan las fotografías, lo cual influye en los resultados finales. Si todas las fotografías fueran tomadas de frente y a una distancia justa, con la iluminación ideal, esta aplicación funcionaría en un porcentaje muy elevado.

Por lo tanto, se puede concluir que el algoritmo desarrollado se puede valorar como positivo, ya que se ha demostrado que se pueden extrar los caracteres de las placas en condiciones muy diferentes y variantes.

Capítulo 8
Conclusiones y líneas futuras

Este último capítulo presenta, a modo de conclusión, el logro de los objetivos y las pautas alcanzados en el proyecto, así como líneas futuras de trabajo que se pueden llevar a cabo para mejorar la aplicación.

8.1. Introducción

El presente trabajo nace de la idea de desarrollar un sistema para la detección y el reconocimiento de placas de matrículas a partir de la toma de fotos sobre el vehículo.

Hasta ahora, los métodos comúnmente empleados para la detección se basan en la utilización de un foco infrarrojo que al aplicarse sobre la placa se resaltan los caracteres, permitiendo un reconocimiento sencillo de la matrícula.

En el supuesto en el que se mueve el proyecto, no se dispone de dicho foco, de modo que hay que crear un sistema que sea invariante a condiciones externas como la luminosidad, rotación, etc.

Para este trabajo se pensó en la idea de realizarlo utilizando las técnicas SIFT

y SURF para la extracción de características locales. Con estos algoritmos se detecta la placa de la matrícula, mientras que después de tratarla se la aplica un OCR y se extraen los caracteres de la misma.

8.2. Logros alcanzados

En este apartado se van a repasar los objetivos uno a uno para obtener una valoración final.

- Se han conseguido resultados tanto en **escenas nocturas como diurnas.** Pero hay que decir que en las escenas de día, los resultados obtenidos son más favorables. Esto se debe a que hay mayor diferenciación entre las partes y los caracteres de la matrícula son más visibles.
- Las imágenes para realizar las pruebas y el desarrollo de la aplicación han sido obtenidas de **diferentes tipos de cámaras,** tanto digitales como de teléfono móvil. También se han obtenido resultados fiables para imágenes de **diferentes tamaños.**
- El funcionamiento de la aplicación para imágenes tomadas a **diferentes distancias** es el esperado. Es decir, se obtienen buenos resultados tanto en las escenas lejanas como en las cercanas.
- Los resultados obtenidos para escenas en las que **varía el ángulo** son correctos. Aunque es obvio que para aquellas instantáneas tomadas de frente, la aplicación tiene una efectividad mayor.

Además de los objetivos finales marcados, se han llevado a cabo una serie de etapas intermedias que han sido determinantes para la consecución del resultado final. Todas estas etapas son las que se han desarrollado a lo largo de la presente memoria.

8.3. Valoración

Esta aplicación tiene un porcentaje de acierto elevado. Pero ello no quiere decir que sea infalible, ya que puede fallar cuando el automóvil se encuentra a largas distancias, o que no se puedan reconocer diferentes caracteres por su estado. De todos modos, son casos mínimos, funcionando de la manera esperada en la mayoría de las situaciones.

Haciendo referencia a la tasa de aciertos de la aplicación, se puede decir que está en torno a un 80 %. Se considera una cifra buena, ya que no se dispone de

ningún elemento auxiliar para la toma de las imágenes. Estos elementos pueden ser la iluminación, además de que se ha utilizado una cámara sin flash.

Debido a todos esos factores, la dificultad para desarrollar la aplicación con una tasa de acierto del 100 % resulta imposible.

8.4. Líneas futuras

En este apartado se van a presentar una serie de líneas futuras que pueden mejorar la aplicación.

Se ha visto que la eficacia de esta aplicación es bastante alta. No se puede asegurar una eficiencia del 100 % sea cual sea el algoritmo, ya que hasta el más perfecto no asegura que no vaya a haber una ausencia de fallo. Esto se debe a que las situaciones que se pueden dar son infinitas. De modo que se puede trabajar en aumentar la eficacia del sistema, trabajando sobre aquellos aspectos más conflictivos en la detección.

Cuanto más rápida sea una aplicación, mejor. Por ello se podría seguir trabajando en desarrollar los algoritmos existentes o crear otros nuevos, para disminuir los tiempos de ejecución. La utilización de OpenCV sobre C++ hace que ya sea bastante rápida, mucho más que otros entornos y lenguajes que no están destinados al trabajo con objetos.

Uno de los puntos para mejorar la velocidad de la aplicación es trabajar sobre el algoritmo para la detección de características locales, ya que es la parte que más tiempo tarda en realizar. Esto se debe a que para obtener unos resultados fiables, hay que filtrar todos los puntos que va obteniendo una cantidad elevada de veces.

Para mejorar la robustez de la aplicación, se puede trabajar en desarrollar algoritmos diferentes para el tratamiento de las imágenes en función de su grado de luminosidad, ángulo de captura de la imagen, etc. En definitiva, todos aquellos factores que hacen que esta aplicación tenga una efectividad del 80 %.

Bibliografía

[1] Alberto Alejo, "Proyecto fin de carrera: Desarrollo de un detector de placas de matrículas en imágenes basado en características locales," Universidad de Valladolid, Tech. Rep., Enero 2014.

[2] Javier Enebral González, "Proyecto fin de carrera: Detección y asociación automática de puntos característicos para diferentes aplicaciones," Tech. Rep., Julio 2009.

[3] Alberto Alejo y Aria Isabel San Agustín, "Proyecto fin de carrera: Procesamiento de imágenes para la detección de señales de tráfico verticales," Universidad de Valladolid, Tech. Rep., Julio 2011.

[4] (Última consulta, Junio, 2014) Comparativa visión humana y visión artificial. [Online]. Available: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/VisionArtificial/index.html

[5] (Última consulta, Junio, 2014) Visión artificial en la industria. [Online]. Available: http://www.lyl-ingenieria.com/es/soluciones-industria/vinspec-inspeccion-de-calidad-en-la-gama-m-/c6r44/

[6] (Última consulta, Junio, 2014) Imágenes en escala de grises. [Online]. Available: http://www.aloj.us.es/galba/digital/cuatrimestre_ii/imagen-pagina/2elementos4d.htm

[7] (Última consulta, Junio, 2014) Tutorial opencv. [Online]. Available: docencia-eupt.unizar.es/ctmedra/tutorial_opencv.pdf

[8] (Última consulta, Junio, 2014) Etiquetado y extracción de características. [Online]. Available: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/nieto_b_d/capitulo2.pdf

[9] (Última consulta, Junio, 2014) Tratamiento de imágenes opencv. [Online]. Available: https://moodle2012-13.ua.es/moodle/mod/wiki/view.php?pageid=721

[10] (Última consulta, Junio, 2014) Surf opencv. [Online]. Available: http://docs.opencv.org/modules/nonfree/doc/feature_detection.html

[11] (Última Consulta, Abril, 2014) Página oficial opencv. [Online]. Available: http://opencv.org/

[12] (Última consulta, Junio, 2014) Funciones básicas opencv. [Online]. Available: http://www2.electron.frba.utn.edu.ar/~afurfaro/Info1/Opencv/opencv.pdf

[13] (Última consulta, Septiembre, 2013) Página para descargar el interfaz de programación. [Online]. Available: www.codeblocks.org/

[14] (Última consulta, Mayo, 2014) Instalación opencv. [Online]. Available: http://kevinhughes.ca/tutorials/opencv-install-on-windows-with-codeblocks-and-mingw/

[15] (Última consulta, Junio, 2014) Visual studio. [Online]. Available: http://www.microsoft.com/es-es/download/confirmation.aspx?id=34673

[16] E. de la Fuente y F. M. Trespaderne, "Visión Artificial Industrial," Universidad de Valladolid, Tech. Rep., 2012.

[17] Francisco Trigueros Andrés, "Proyecto fin de carrera: Implementación del algoritmo de cosido de imágenes," Universidad de Valladolid, Tech. Rep., Enero 2011.

[18] (Última consulta, Junio, 2014) Información sobre diferentes aspectos para elaborar los informes y documentos. [Online]. Available: http://es.wikipedia.org/wiki/Wikipedia:Portada

[19] (Última consulta, Junio, 2014) Tratamiento de imágenes opencv. [Online]. Available: http://www.informatica-juridica.com/trabajos/Informe_OpenCV_Tratamiento_Imagenes.pdf

[20] (Última consulta, Junio, 2014) Surf feature detector. [Online]. Available: http://courses.cs.washington.edu/courses/cse576/13sp/projects/project1/artifacts/woodrc/index.htm

[21] (Última consulta, Junio, 2014) Transformada de hough. [Online]. Available: http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html

[22] (Última consulta, Junio, 2014) Creación algoritmo ocr. [Online]. Available: http://stackoverflow.com/questions/9413216/simple-digit-recognition-ocr-in-opencv-python

[23] (Última consulta, Junio, 2014) Diversos algoritmos ocr. [Online]. Available: http://www.dspace.espol.edu.ec/bitstream/123456789/20617/1/D-91508.pdf

[24] (Última consulta, Mayo, 2014) Diferentes aspectos sobre el uso de ocr. [Online]. Available: http://geekland.hol.es/fundamentos-usos-limitaciones-ocr/

[25] (Última consulta, Junio, 2014) Librería tesseract para algoritmo ocr. [Online]. Available: https://code.google.com/p/tesseract-ocr/

[26] (Última consulta, Junio, 2014) Binarización de imágenes en opencv. [Online]. Available: http://es.scribd.com/doc/94351721/OpenCV

[27] (Última consulta, Mayo, 2014) Programa para crear documento en latex. [Online]. Available: http://www.texniccenter.org/

[28] (Última consulta, Junio, 2014) Programación en latex. [Online]. Available: http://www.fceia.unr.edu.ar/lcc/cdrom/Instalaciones/LaTex/latex.html

[29] (Última consulta, Junio, 2014) Aplicaciones de la visión artificial. [Online]. Available: http://dmi.uib.es/~ygonzalez/VI/Material_del_Curso/Teoria/Aplicaciones_VC.PDF

[30] (Última consulta, Junio, 2014) Sistemas de detección de matrículas. [Online]. Available: http://www.tecoyse.net/reconocimiento_matriculas_por_cctv.php

[31] (Última consulta, Junio, 2014) Imágenes digitales. [Online]. Available: http://www.ite.educacion.es/formacion/materiales/86/cd/pdf/m2_caracteristicas_de_la_imagen_digital.pdf

[32] (Última consulta, Junio, 2014) Etapas en el procesamiento de una imagen. [Online]. Available: http://www.sites.upiicsa.ipn.mx/polilibros/portal/Polilibros/P_proceso/Nuevas_tecnologias_Areli_Araos_Pe%C3%B1aloza/procesoimagenes/proc_info.htm

Anexo A

Código desarrollado

En el presente anexo se va a incluir todo el código desarrollado para el funcionamiento del sistema, el cual ha sido desarrollado en el lenguaje de programación C++. Va a estar dividido en dos partes. En la primera se incluye todo el tratamiento sobre la imagen desde que se captura hasta que se obtiene la matrícula utilizando la librería OpenCV. La segunda parte va a contener el algoritmo desarrollado para detectar los caracteres a través de OCR con Tesseract.

A.1. Tratamiento de la imagen

En este apartado se va a incluir todo el código desarrollado en C++ utilizando la librería OpenCV. En él se incluyen todos los algoritmos desarrollados para la obtención de la imagen final a partir de la imagen de entrada.

```
/***------------------------------------------***/
/*                                              */
/*          TRABAJO DE FIN DE GRADO             */
/*                                              */
/* SISTEMA DE VISIÓN ARTIFICIAL PARA LA         */
/* DETECCIÓN Y LECTURA DE MATRÍCULAS            */
/*                                              */
/*     AUTOR: Roberto Muñoz Manso               */
/*                                              */
/* Grado en Ingeniería en Electrónica           */
/* Industrial y Automática                      */
/*                                              */
/***------------------------------------------***/

/*** LIBRERÍAS ***/

#include "opencv2/core/core.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/nonfree/features2d.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/opencv.hpp"
#include "opencv2/calib3d/calib3d.hpp"
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <math.h>
#include <string.h>

/*** DECLARACIÓN DE FUNCIONES ***/
void surf_detection(cv::Mat img_1, cv::Mat img_2);

/*** PROGRAMA PRINCIPAL ***/
int main(int argc, char** argv)
{
```

```
    //Declaración de variables
    cv::Mat img; //Imagen que contiene la fotografía
    cv::Mat img_template; //Imagen que contiene el patrón

    //Carga las imágenes
    img = cvLoadImage(argv[1], 0);
    img_template = cvLoadImage("patron.jpg", 0);

    //Si no se puede cargar alguna de las imágenes
    if( !img.data || !img_template.data )
    {
        printf("No se ha podido cargar una imagen");
        return -1;
    }

    /*Redimensiona la imagen de entrada para trabajar
       siempre
    con las mismas dimensiones*/
    cv::resize(img,img,cv::Size(640,480));

    //Llama a la función SURF
    surf_detection(img_template,img);
}


/** FUNCIÓN QUE ELIMINA PUNTOS EXTERNOS DE LA MATRÍCULA **/
/*Función que elimina todos aquellos puntos característicos
detectados por la función SURF, pero que se encuentran
   fuera
de la matrícula. Para ello determina la mínima distancia
   entre
puntos y elimina todos aquellos que no tienen al menos 5
   puntos
al doble de la distancia mínima hallada.*/

cv::vector<cv::Point2f> nube(cv::vector<cv::Point2f> scene,
    double min_dist,cv::Mat img2)
{
    //Declaración de variables
    double distx,disty;
```

```
    int cont;
    std::vector<cv::Point2f> sal;

    //Compara todos los puntos para hallar la distancia
       mínima
    for (int i = 0; i < scene.size(); i++)
    {
        cont=0;
        for (int j=0; j < scene.size(); j++)
        {
            if (i==j || scene[i].y < (img2.rows/3))
            {

            }
            else
            {
                distx=std::abs(scene[i].x-scene[j].x);
                disty=std::abs(scene[i].y-scene[j].y);
                if (distx<2*min_dist && disty<2*min_dist)
                {
                    cont=cont+1; /*Variable que cuenta
                       cuántos puntos
                    hay más cerca del doble de la mínima
                       distancia*/
                }
            }
        }
        //Si hay más de 5 puntos a menos del doble de la
           mínima distancia
        if (cont >5)
        {
            /*Añade los puntos buenos al vector*/
            sal.push_back(scene[i]);
        }
    }
    return(sal);//Devuelve el vector de puntos buenos
}
```

```
/** FUNCIÓN QUE REALIZA TODO EL TRATAMIENTO DE LA IMAGEN *
   */
/*Función que se encarga de realizar todo el tratamiento de
de la imagen. Detecta los puntos característicos, la
   recorta
y la trata para poder aplicar el algoritmo OCR*/
void surf_detection(cv::Mat img_1,cv::Mat img_2)
{
    /**-------------------------------------------------------**/
    /** APLICAR ALGORITMO SURF **/
    /*Al aplicar el algoritmo SURF se pretende determinar
    todas las coincidencias entre la imagen de entrada y
    el patrón. Sigue varios pasos.*/

    /* Paso 1: Detectar puntos caracteísticos usando
    la función SURF*/
    //Declaración de variables
    int minHessian = 400; //Valor del Hessiano; parámetro
       del SURF
    int j;
    float minx,miny,maxx,maxy;
    //Declaración de la función SURF
    cv::SurfFeatureDetector detector( minHessian );
    std::vector<cv::KeyPoint> keypoints_1, keypoints_2;
    std::vector< cv::DMatch > good_matches;

    //Busca puntos característicos un número determinado de
        veces
    // para descartar los malos
    do
    {
        //Detecta los puntos característicos de las
           imágenes
        detector.detect( img_1, keypoints_1 );
        detector.detect( img_2, keypoints_2 );

        // Muestra los keypoints
        cv::Mat img_keypoints_1;
        cv::Mat img_keypoints_2;
```

```
drawKeypoints( img_1, keypoints_1, img_keypoints_1,
    cv::Scalar::all(-1), cv::DrawMatchesFlags::
   DEFAULT );
drawKeypoints( img_2, keypoints_2, img_keypoints_2,
    cv::Scalar::all(-1), cv::DrawMatchesFlags::
   DEFAULT );

/* Paso 2: Calcular descriptores (vector
   característico). Los
descriptores son los puntos característicos que más
    información contienen */
cv::SurfDescriptorExtractor extractor;
cv::Mat descriptores_1, descriptores_2;
extractor.compute( img_1, keypoints_1,
   descriptores_1 );
extractor.compute( img_2, keypoints_2,
   descriptores_2 );

/* Paso 3: Enlazar coincidencias de los vectores
   descriptores con FLANN */
cv::FlannBasedMatcher matcher;
std::vector< cv::DMatch > matches;
matcher.match( descriptores_1, descriptores_2,
   matches );

//Variables para discriminar puntos característicos
    por distancia
double max_dist = 0;
double min_dist = 100;

//-- Calculo de la maxima y minima distancia entre
   puntos característicos
for( int i = 0; i < descriptores_1.rows; i++ )
{
    double dist = matches[i].distance;
    if( dist < min_dist )
        min_dist = dist;
    if( dist > max_dist )
        max_dist = dist;
}
```

```
    /* Dibujar solo las coincidencias "buenas".
       Aquellas cuya
    distancia es menor que 1.5 veces la mínima
       distancia entre
    puntos característicos*/
    for( int i = 0; i < descriptores_1.rows; i++ )
    {
        if( matches[i].distance < 1.5*min_dist )
        {
            //Los añade al final del vector
            good_matches.push_back( matches[i]);
        }
    }
}
//Condición para iterar
while(good_matches.size()<100);

//Dibujar solo las coincidencias "Buenas"
cv::Mat img_matches;
drawMatches( img_1, keypoints_1, img_2, keypoints_2,
   good_matches, img_matches, cv::Scalar::all(-1), cv::
   Scalar::all(-1), std::vector<char>(), cv::
   DrawMatchesFlags::NOT_DRAW_SINGLE_POINTS );

// Localizar puntos característicos "buenos"
//y los añade al vector de su imagen
std::vector<cv::Point2f> obj;
std::vector<cv::Point2f> scene;
for( int i = 0; i < good_matches.size(); i++ )
{
    // Obtención de los keypoints de las coincidencias
       buenas
    obj.push_back( keypoints_1[ good_matches[i].
       queryIdx ].pt );
    scene.push_back( keypoints_2[ good_matches[i].
       trainIdx ].pt );
}
```

```
/*Se llama a la función nube para eliminar aquellos
   puntos
que se encuentran fuera de la matrícula*/
std::vector<cv::Point2f> obj2;
obj2=nube(scene,0.20,img_2);

/*Obtener posiciones de los puntos extremos. Aquellos
   que
tienen los mayores y los menores valores en x e y*/
for ( j = 0; j < obj2.size()-1; j++)
{
    if (j==0)
    {
        minx= obj2[j].x;
        miny= obj2[j].y;
        maxx= obj2[j].x;
        maxy= obj2[j].y;
    }
    if(obj2[j+1].x < minx)
    {
        minx=obj2[j+1].x;
    }
    if(obj2[j+1].y < miny)
    {
        miny=obj2[j+1].y;
    }
    if(obj2[j+1].x > maxx)
    {
        maxx=obj2[j+1].x;
    }
    if(obj2[j+1].y > maxy)
    {
        maxy=obj2[j+1].y;
    }
}

/*Hallar el rectángulo de corte. Para ello determina la
posición del punto más a la izquierda. En función de su
ubicación crea diferentes rectángulos de corte.*/
float cols = img_2.cols;;
```

```
float div = minx/cols;
double tol=60.0;
float minxx;
float maxyy=maxy+tol;
float minyy=miny-tol;

if(div>0 && div<0.25)
{
    minxx=minx;
    maxx=minx+(minx*3);
}
if(div>0.25 && div<0.35)
{
    minxx=minx-(minx*0.2);
    maxx=minx+(minx*1.6);
}
if(div>0.35 && div<0.45)
{
    minxx=minx-(minx*0.35);
    maxx=minx+(minx*1.2);
}
if(div>0.45 && div<0.55)
{
    minxx=minx-(minx*0.6);
    maxx=minx+(minx*0.6);
}
if(div>0.55 && div<0.65)
{
    minxx=minx-(minx*0.65);
    maxx=minx+(minx*0.4);
}
if(div>0.65 && div<0.75)
{
    minxx=minx-(minx*0.7);
    maxx=minx;
}
if(div>0.75 && div<1)
{
    minxx=minx-(minx*0.8);
    maxx=minx+(minx);
```

```
}

//Vector para guardar los puntos de corte de la imagen
std::vector<cv::Point2f> obj_corners1(4);
obj_corners1[0] = cv::Point2f(minxx,minyy);
obj_corners1[1] = cv::Point2f(maxx,minyy);
obj_corners1[2] = cv::Point2f(maxx,maxyy);
obj_corners1[3] = cv::Point2f(minxx,maxyy);
//Se realiza un corte en forma de rectángulo
cv::Rect myRec2(obj_corners1[0].x,obj_corners1[0].y,
   obj_corners1[1].x-obj_corners1[0].x,obj_corners1[2].
   y-obj_corners1[1].y);
cv::Mat crop = img_2(myRec2);

//-- Dibujar las lineas del rectángulo de corte
cvtColor(img_2,img_2,CV_GRAY2BGR);
line( img_2, obj_corners1[0], obj_corners1[1], cv::
   Scalar( 0, 255, 0), 4 );
line( img_2, obj_corners1[1], obj_corners1[2], cv::
   Scalar( 0, 255, 0), 4 );
line( img_2, obj_corners1[2], obj_corners1[3], cv::
   Scalar( 0, 255, 0), 4 );
line( img_2, obj_corners1[3], obj_corners1[0], cv::
   Scalar( 0, 255, 0), 4 );


/** ALGORITMO PARA ACOTAR EL ESPACIO DE BÚSQUEDA **/
//Declarar variables
cv::Point p1, p2;
cv::Mat rectash;
cv::Mat rec_col;
//Vector que contiene las rectas detectadas
std::vector<cv::Vec2f> lineash;

//Se aplica Canny a la imagen para extraer los bordes
cv::Canny(crop,rectash,100,250,3);

//Imagen para que las rectas salgan a color
cvtColor(rectash,rec_col,CV_GRAY2BGR);
```

```
//Se obtienen las líneas por la transformada de Hough
cv::HoughLines(rectash,lineash,1,CV_PI/180,90,0,0);

float maximo = 0, minimo = 500;
//Se dibujan las lineas horizontales
for(size_t i=0; i<lineash.size(); i++)
{
    float rho = lineash[i][0];
    float theta = lineash[i][1];
    double a = cos(theta);
    double b = sin(theta);
    double x0 = a*rho;
    double y0 = b*rho;

    //Se discrimina para dibujar las horizontales
    float disc = (theta*180)/CV_PI;
    //Solo se dibujan las rectas en un rango de 10°
    if((disc<95) && (disc>85))
    {
        p1.x = cvRound(x0+1000*(-b));
        p1.y = cvRound(y0+1000*(a));

        p2.x = cvRound(x0-1000*(-b));
        p2.y = cvRound(y0-1000*(a));

        /*Se hallan el máximo y el mínimo valor en y
        de las rectas detectadas*/
        if(maximo < (p1.y+p2.y)/2)
        {
            maximo = (p1.y+p2.y)/2;
        }
        if(minimo > (p1.y+p2.y)/2)
        {
            minimo = (p1.y+p2.y)/2;
        }
    }
    //Se dibujan las rectas detectadas
    cv::line(rec_col,p1,p2,cv::Scalar(0,0,255),3);
}
```

```
//Se halla el valor medio del máximo y mínimo valor en
//y de las rectas
float media = 0;
media = (maximo+minimo)/2;
float cormaxy = 500, corminy = 0;
for(size_t i=0; i<lineash.size(); i++)
{
    float rho = lineash[i][0];
    float theta = lineash[i][1];
    double a = cos(theta);
    double b = sin(theta);
    double x0 = a*rho;
    double y0 = b*rho;

    float disc = (theta*180)/CV_PI;
    if((disc<95) && (disc>85))
    {
        p1.x = cvRound(x0+1000*(-b));
        p1.y = cvRound(y0+1000*(a));

        p2.x = cvRound(x0-1000*(-b));
        p2.y = cvRound(y0-1000*(a));

        /*Se hallan los puntos para realizar el corte*/
        float put = (p1.y+p2.y)/2;

        if((cormaxy>put) && (put>media))
        {
            cormaxy=put;
        }
        if((corminy<put) && (put<media))
        {
            corminy=put;
        }
    }
}

//Se corta la imagen previamente cortada
std::vector<cv::Point2f> corte2(4);
int ancho = crop.cols;
```

```
corte2[0] = cv::Point2f(0,corminy-15);
corte2[1] = cv::Point2f(ancho,corminy-15);
corte2[2] = cv::Point2f(ancho,cormaxy+10);
corte2[3] = cv::Point2f(0,cormaxy+10);
cv::Mat crop2;
/*Realiza diferentes cortes en función de si hay rectas
    superiores
e inferiores, o si sólo hay alguna de ellas*/
if(cormaxy+corminy<rec_col.rows)
{
    //Determina los puntos de corte y realiza el corte
    float corte = corte2[1].y+(corte2[1].y)*1.2;
    cv::Rect myRec3(corte2[0].x,corte2[0].y,corte2[1].x
       -corte2[0].x,corte);
    crop2 = crop(myRec3);
}
else
{
    //Realiza el corte
    cv::Rect myRec3(corte2[0].x,corte2[0].y,corte2[1].x
       -corte2[0].x,corte2[2].y-corte2[1].y);
    crop2 = crop(myRec3);
}

/** TRATAR LA IMAGEN **/
cv::Mat imag;

//Se realiza una binarización adaptativa
cv::adaptiveThreshold(crop2,imag,255,
   CV_ADAPTIVE_THRESH_GAUSSIAN_C,CV_THRESH_BINARY,7,9);

/*Se eliminan aquellas regiones detectadas como ruido*/
//Halla los contornos por Canny
cv::Mat drawing;
Canny(imag, drawing, 50, 700, 3);
std::vector<std::vector<cv::Point>> contours;
//Detecta los contornos y los guarda en un vector
cv::findContours(drawing, contours, CV_RETR_EXTERNAL,
   CV_CHAIN_APPROX_NONE);
for (int i = 0; i < contours.size(); i++)
```

```
{
    /*Si el área de las regiones detectadas es menor
       que
    cierto valor, las elimina (los dibuja de color
       blanco).
    El resultado van a ser los caracteres y zonas
       externas
    de gran tamaño.*/
    if (cv::contourArea(contours[i]) < 150)
    {
        cv::drawContours(imag, contours, i, cv::Scalar
           (255), -1);
    }
}

/*Se rellenan las zonas eliminadas de los caracteres
   con la binarización*/
cv::Mat drawing1;
Canny(imag, drawing1, 50, 700, 3);
std::vector<std::vector<cv::Point> > contours1;
//Detecta los contornos y los guarda en un vector
cv::findContours(drawing1, contours1, CV_RETR_EXTERNAL,
    CV_CHAIN_APPROX_NONE);
for (int i = 0; i < contours1.size(); i++)
{
    /*Si el área de las regiones detectadas es
    menor que cierto valor,
    las pinta de color negro.*/
    if (cv::contourArea(contours1[i]) < 200)
    {
        cv::drawContours(imag, contours1, i, cv::Scalar
           (0), -1);
    }
}

/*Detección de los puntos de corte*/
/*Busca los puntos extremos de todos los lados
de la matrícula y los guarda para cortar*/
uchar xmi=0,xma=imag.cols,yma=imag.rows,ymi=0;
for(int j=0; j<imag.cols; j++)
```

```
{
    if (imag.at<uchar>(imag.rows/2,j) == 255)
    {
        xmi=j;
        break;
    }
}
for(int j=imag.cols; j>0; j--)
{
    if (imag.at<uchar>(imag.rows/2,j) == 255)
    {
        xma=j;
        break;
    }
}
for(int j=0; j<imag.rows; j++)
{
    if (imag.at<uchar>(j,imag.cols/2) == 255)
    {
        ymi=j;
        break;
    }
}
for(int j=imag.rows; j>0; j--)
{
    if (imag.at<uchar>(j,imag.cols/2) == 255)
    {
        yma=j;
        break;
    }
}

/*Realiza un corte rectangular pasando por
los puntos detectados, además de una pequeña tolerancia
   */
cv::Rect myRec4(xmi+10,ymi+5,xma-5,yma-5);
cv::Mat cropa = imag(myRec4);

//Guarda la imagen cortada para aplicarla un
//algoritmo OCR
```

```
    cv::imwrite("crop.jpg",cropa);
}
```

A.2. Algoritmo OCR

En este apartado se va a incluir el código desarrollado en C++ utilizando la librería Tesseract para el reconocimiento de los caracteres. En él se incluyen todas las funciones creadas para llamar a las API de la librería Tesseract.

```
#include <baseapi.h>
#include <allheaders.h>
#include <iostream>
using namespace std;

int main(void){

FILE *mat;
tesseract::TessBaseAPI api;
api.Init("", "eng", tesseract::OEM_DEFAULT);
//Definición de caracteres a buscar
api.SetVariable("tessedit_char_whitelist", "
   ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789");
//Establece el método de búsqueda
api.SetPageSegMode(static_cast<tesseract::PageSegMode>(7));
api.SetOutputName("out");

//Recibe imagen tratada
char image[]="crop.jpg";
PIX    *pixs = pixRead(image);

//Crea un string de salida
STRING text_out;
api.ProcessPages(image, NULL, 0, &text_out);

//Guarda los caracteres detectados en un fichero .txt
cout<<text_out.string();
mat = fopen("matricula.txt","w");
fprintf(mat,text_out.string());
fclose(mat);
```

```
cin>>image;
```