UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

DETECCIÓN Y RECONOCIMIENTO DE SEÑALES VIALES

José Manuel Esteve de Prada
Tutor: Álvaro García Martín
Ponente : José María Martínez Sánchez

Julio 2017

DETECCIÓN Y RECONOCIMIENTO DE SEÑALES VIALES

AUTOR: José Manuel Esteve de Prada
TUTOR: Álvaro García Martín

Video Processing and Understanding Lab
Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2017

Trabajo parcialmente financiado por el gobierno español bajo el proyecto TEC2014-53176-R (HA-Video)

Resumen

En los últimos años, el mundo automovilístico ha sufrido una evolución enorme en forma de innumerables avances, a destacar uno por encima de todos ellos: la búsqueda de la conducción autónoma. Y aunque aún quedan años para que la conducción autónoma sea una realidad en las calles, cada vez mas coches vienen equipados con sistemas que ofrecen información de utilidad al conductor. Dicha información se recoge de la propia vía por la que se circula.

En persecución de este objetivo tan general han surgido nuevas empresas y departamentos específicos dentro de las marcas de coches para crear sistemas que al funcionar comúnmente logren el objetivo de la conducción autónoma. Además, dichos sistemas se han empezado a comercializar como ADAS (Advanced Driver Assistance Systems), permitiendo su prueba y mejora en entornos reales de conducción. Uno de esos sistemas, denominado TSR (Traffic Sign Recognition), consiste en reconocer la señal detectada en la carretera.

Este Trabajo Fin de Grado tiene como objetivo una primera implementación de un algoritmo propio de detección y reconocimiento de señales verticales de tráfico. Dicha detección no es sencilla debido a las condiciones cambiantes que se pueden presentar en la carretera y en circulación, desde cambios climatológicos o cambios de luz hasta la dificultad de captar una señal desde un coche en marcha.

Aunque otros sistemas de detección y reconocimiento hacen uso de varias cámaras y sensores, en esta aproximación se utilizará una única cámara con el objetivo de crear un sistema cuya mayor cualidad sea su capacidad de adaptación al medio, pudiendo integrarse en el software de un coche moderno o en un Smartphone para su uso desde el salpicadero de un coche o la cúpula de una moto. Debido a mi escasa experiencia en el campo del procesamiento de imagen, este sistema no busca destacar por encima de los mencionados en el estado del arte, si no plantear una solución sencilla como primera aproximación a un problema que se ha vuelto muy ocurrente en la actualidad.

Palabras clave

ADAS, TSR, Detección, Clasificación, Reconocimiento de señales viales, OpenCV

Abstract

Lately, a huge evolution has taken place within the automotive world. We can find many improvements in this area, being the search of the autonomous driving the most important one. Despite the fact that there still remain a lot of years of progress to encounter autonomous driving in the street, more and more vehicules are now equipped with systems providing some useful information to the driver. These pieces of information are taken from the road itself.

Some new companies and specific departments within car brands have arisen in search of this very general target, creating systems that, in the ordinary usage, are able to perform autonomous driving. Besides, these systems are starting to be sold as ADAS (Advanced Driver Assistance Systems), allowing their testing and upgrading in a real driving environment. One of the abovesaid systems, called TSR (Traffic Sign Recognition), consists in the recognition of the detected traffic sign.

This Bachelor Thesis aims to obtain a first implementation of an own algorithm of vertical traffic signs detection and recognition. This detection is not easy due to the changing conditions of the road and the circulation, such as weather or light changes or the difficulty to receive traffic signs from a moving car.

Although other detection and recognition systems use more than one camera or sensor, we will be making use of just one camera, trying to create a system whose best quality be its adaptive capacity to the environment. This way, it could be included in a modern car software or in a Smartphone and then used from a car dashboard or a motorbike wind deflector. As I am aware of my limited experience concerning image processing, the following system aspires, not to stand out among the ones described in the State of the art section, but to propose a simple solution to this existent question.

Keywords

ADAS, Advanced Driver Assistance System, TSR, Traffic Sign Recognition, Detection, Classification, OpenCV,

Agradecimientos

A todas y cada una de las persona que me he cruzado en el camino de este TFG y me han preguntado sobre él, se han interesado por su progresión, me han animado o simplemente me han escuchado quejarme, algunas especialmente.

A mi familia, en concreto a mis padres, por las ganas puestas en mi formación que se han convertido en oportunidades.

A Álvaro García, tutor de este trabajo, sobre todo por aceptar una propuesta en pañales de un alumno que ni siquiera conocía y guiarla de forma tal que se ha transformado en este proyecto.

A amigos, conocidos y profesores de la universidad, este es el trabajo de fin de período, pero no hay que olvidar el desarrollo personal que se produce en la universidad. Cada día de estos años ha formado parte de la persona que soy ahora haciendo este trabajo fin de grado.

Todos habéis aportado vuestro granito de arena y significa mucho más de lo que pensáis. Muchas gracias a todos por la confianza y el apoyo

José Manuel Esteve de Prada,
Julio 2017

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Hace años, cuando se introdujeron los navegadores GPS, como el TomTom, además del avance en cuanto a rutas guiadas, se produjo un avance en la información que se le ofrecía al conductor en cuanto al estado del tráfico, carril por el que circular, velocidad máxima permitida en la vía, etc. Tras estos sistemas llegaron los avisos cuando se superaba dicha velocidad máxima. Pero, ¿cómo sabía el navegador cual era la velocidad máxima de la vía con tal precisión? La respuesta es sencilla, por geoposicionamiento.

Sin embargo con la búsqueda del coche autónomo, el geoposicionamiento ya no es suficiente, el coche ha de ser capaz de entender y adaptarse a las condiciones cambiantes de la vía y así surgieron los sistemas TSR de ayudas a la conducción. En la actualidad multitud de marcas de coches luchan por ser punteras en la introducción de sistemas TSR y para ello utilizan procesamiento de imagen sobre la vía por la que se circula. Un gran número de estos sistemas ya están comercializados como software interno del coche o como software externo o adicional y que se vende por separado, al igual que lo hacían aquellos primeros navegadores GPS.

La motivación de este trabajo no es competir con un estado del arte ya desarrollado y afianzado en el mercado, si no lograr una primera aproximación a un algoritmo capaz de detectar y reconocer un gran abanico de señales verticales de tráfico, centrándose, más concretamente, en las señales de España.

Dicha aproximación no busca centrarse en un modelo de cámara concreto, una posición fija dentro del coche o sólo señales de velocidad. Si no que busca una solución general a un problema que ya ha sido resuelto por algunos fabricantes de automóviles de forma concreta, la detección y posterior reconocimiento de las señales viales verticales que pueden encontrarse en la carretera.

1.2 Objetivos

Una vez establecida la motivación, los objetivos parciales para llegar a alcanzar dicha solución, son los siguientes:

1. Estudio del estado del arte:

   Analizar en profundidad a qué se ha llegado y cómo funcionan los sistemas que ya se encuentran en el mercado. Proporcionar una visión general de las etapas en las que se dividen dichos algoritmos y en qué consiste cada una de esas etapas.

2. Diseño del algoritmo propio a desarrollar:

   Partiendo de lo aprendido en el estado del arte, diseñar el algoritmo de forma que pueda ser aplicado lo más genéricamente posible, esto es: variando las condiciones de luz, posición, velocidad, modelo de cámara, etc.

3. División y desarrollo del algoritmo en diferentes etapas:

   Implementación del algoritmo diseñado en el objetivo anterior.
   El algoritmo será desarrollado por fases y cada fase se irá probando y revisando de nuevo a medida que se vayan completando nuevas fases.

4. Evaluación de los resultados del algoritmo:

   Análisis de resultados tras realización de pruebas con el objetivo de averiguar el rendimiento real del algoritmo.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capitulo 1. Introducción:** motivación y objetivos principales del proyecto.
- **Capitulo 2. Estado del arte:** descripción de los algoritmos existentes hasta la fecha, .
- **Capitulo 3. Diseño:** descripción de la estructura del algoritmo a implementar.
- **Capitulo 4. Desarrollo:** desarrollo del algoritmo y problemas encontrados durante su realización.
- **Capitulo 5. Integración, pruebas y resultados:** pruebas del algoritmo y consecución de objetivos.
- **Capitulo 6. Conclusiones y trabajo futuro:** conclusiones del trabajo realizado y trabajo futuro.

2 Estado del arte

Debido a que la detección y reconocimiento de objetos y, en concreto para el tema que atañe a este trabajo, de señales viales ha sido un tema candente en los últimos años, hay numerosos ensayos hablando de aproximaciones al entendimiento de una señal de tráfico por parte de programas software.

De hecho, las primeras investigaciones datan de los años 80. Sin embargo por aquel entonces el coste computacional era aún muy alto y esto limitó durante muchos años los avances de estas tecnologías [1].

Una vez la capacidad computacional aumentó exponencialmente, surgieron diferentes aproximaciones al problema hasta llegar a las más populares hoy en dia, en la siguiente sección se dividirá el sistema en etapas comunes entre todos los algoritmos para poder estructurar la exposición de este capítulo.

2.1 Planteamiento del sistema

El punto en el que coinciden los diferentes TSRs existentes es la división del trabajo efectuado en etapas. Así, las etapas que forman parte de los algoritmos más famosos son:

- Detección
- Refinamiento
- Reconocimiento o clasificación
- Seguimiento

Dependiendo de cada algoritmo, puede haber etapas que estén presentes y otras que no. Así, a modo de ejemplo, en la figura 2-1 encontramos todas las etapas.

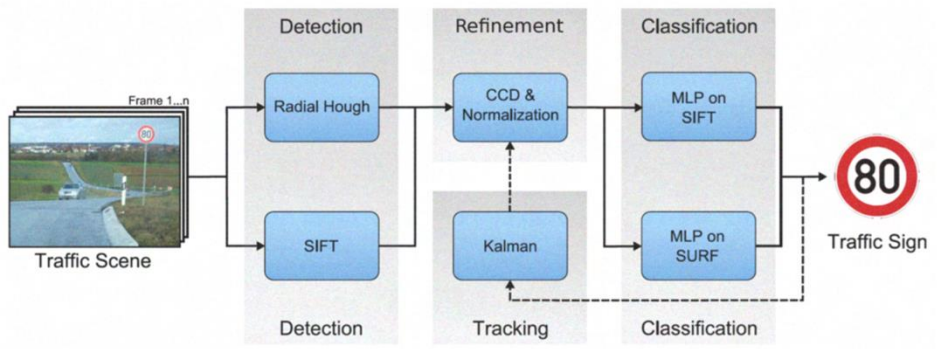

Figura 2-1: Esquema del diseño del algoritmo de [1]

Y sin embargo en la figura 2-2, las etapas presentes son menos, faltando en este caso la etapa de refinamiento.

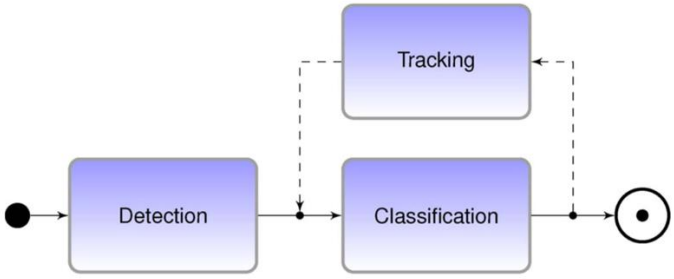

Figura 2-2: Esquema del diseño del algoritmo de [1]

En las siguientes secciones se expondrá cada una de estas etapas, exponiendo los diferentes algoritmos que se utilizan dentro de cada una de ellas.

2.2 Detección

La etapa de detección es la primera, en la que la región de la imagen en la que se encuentra la señal es extraída para posteriores etapas. Es, por tanto, crucial el trabajo de esta etapa, ya que las regiones no extraídas no serán evaluadas posteriormente.

2.2.1 Color

Una de las características más clara de las señales verticales de carretera, al menos en España, es que se los colores que utilizan son colores muy bien definidos, estando entre ellos el rojo, el azul y el blanco. Por ello numerosos sistemas, como los encontrados en [1], [1] y [1] realizan su primera clasificación en base a la detección de umbrales de color, como puede apreciarse en la figura 2-3.

Figura 2-3: Ejemplo de uso de detección de color en [1]

Como se puede apreciar, en la figura 2-3(b), en blanco está marcada la región en la que se ubica la señal de cara a siguientes fases.

2.2.2 Forma

Debido a la forma artificial de las señales, esta técnica es muy útil ya que gracias a la transformada de Hough [1] es posible encontrar tanto círculos como líneas rectas de un tamaño mayor a un mínimo y menor a un máximo. Manteniendo el número de círculos buscado en valores muy pequeños, se produce una gran eficiencia. [1]

Finalmente para evitar la aparición de varios círculos para una única forma redonda, se aplica un una supresión de tamaño 2r + 1 en la zona en la que ya se ha encontrado un círculo. [1]

2.2.3 Rasgos locales

Es también posible realizar una búsqueda basada en el contenido de la señal. Se utiliza en varios ocasiones como segundo algoritmo de detección para filtrar las regiones detectadas en un primer paso.

Ejemplos de algoritmos que utilizan esta técnica son algoritmos como SIFT (Scale-Invariant feature transform) [1], que localiza buenos rasgos a evaluar y adapta el modelo Gaussiano a dichos ejemplos.

2.3 Refinamiento

La etapa de refinamiento consiste simplemente en la mejora de la zona extraída, en la mayoría de los algoritmos no está presente, sin embargo en los que está supone un extra que beneficia la eficacia del algoritmo, como visto en [1].

2.3.1 Algoritmo CCD (Contracting Curve Density)

Este algoritmo reduce los problemas causados por la observación desde una perspectiva diferente a la frontal mediante la rotación de la región detectada hasta hacerla cumplir con la forma deseada.

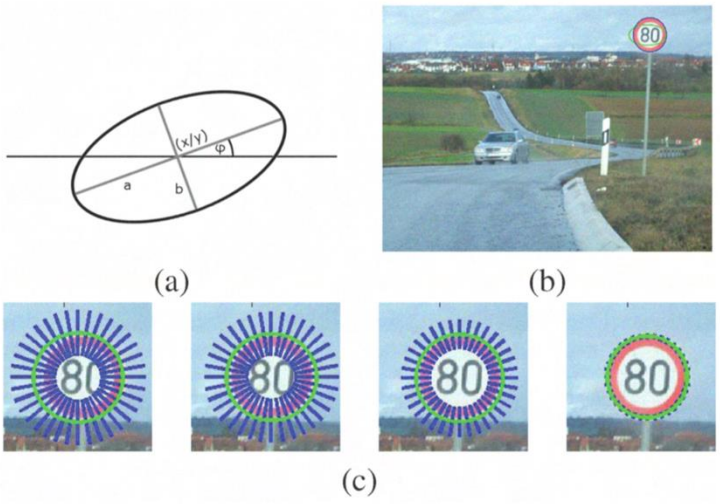

Figura 2-4: Ejemplo de uso del algoritmo CCD para el refinamiento en [1]

Como puede observarse en la figura 2-4, al detectar la señal desde una perspectiva lateral o estar la señal doblada, la región de la señal está ligeramente girada. Tras la aplicación del proceso ilustrado en la figura 2-4(c) la imagen final mejorada carece de una inclinación tan notable.

2.4 Reconocimiento

En la fase de reconocimiento encontramos desde algoritmos que aplican las técnicas de comparación directa (template matching) o comparación cruzada (cross correlation) [1] hasta algoritmos que implementan un pequeño clasificador en sus etapas previas y finalmente para las señales numéricas emplean pequeños programas OCR (Optical Character Recognition) [1].

Además, el empleo del algoritmo Viola-Jones junto con Haar Cascade [1] es bastante usual y los resultados que proporciona son muy buenos y cabe mencionarlo aunque se sale de este trabajo por utilizar aprendizaje automático.

2.5 Seguimiento (Tracking)

Por último, la etapa de seguimiento, que al igual que la de refinamiento no es necesaria, pero sí es conveniente usarla en algunos casos en los que aparecen falsos positivos en la fase de reconocimiento. El seguimiento de una señal reconocida se emplea para asegurar que dicho reconocimiento es robusto, filtrando las señales que en fotogramas muy cercanos en tiempo se repiten, o dentro de un mismo fotograma aparecen a un lado y otro de la imagen (y por tanto de la carretera) y separándolas de los falsos positivos que hayan llegado hasta la fase de reconocimiento del algoritmo [1] y [1].

Gracias a este algoritmo no solo la información de forma o color es evaluada, si no que también existe una evaluación referente al tiempo de aparición de las formas o colores detectados.

2.5.1 Algoritmo de Kalman

El filtro de Kalman es el algoritmo de seguimiento más utilizado [1] y [1] y permite al sistema predecir si dicha señal debería aparecer en el siguiente fotograma, así como evaluar la posición en la que aparece para valorar si hay consistencia con su aparición anterior. [1]

3 Diseño

3.1 Introducción

En este capítulo se describe el diseño del que ha partido el algoritmo de detección y reconocimiento fruto de este trabajo. Tras entender y valorar las diferentes aproximaciones existentes hasta el momento, se ha optado por separar las fases de detección de las señales y reconocimiento de las mismas.

De cara a la fase de detección de las señales y habiendo observado que las señales se pueden clasificar principalmente por su forma y color, se ha optado por diferenciar dichos rasgos significativos con el objetivo de optimizar la detección. La división de las señales que busca detectar este algoritmo es la siguiente:

- Señales triangulares de borde rojo. (43 Señales de advertencia de peligro y 1 señal de reglamentación.
- Señales circulares de borde rojo. (39 Señales de prohibición o restricción)
- Señales circulares azules. (25 señales de obligación)

Si bien es cierto que en el reglamento de tráfico hay más grupos de señales [1], se ha considerado, por el reducido espacio de tiempo, que sería mejor centrarse en la detección de estos tres grandes grupos de señales.

El algoritmo ha sido diseñado por etapas. En primer lugar buscará los colores de interés (en este caso rojo y azul) y únicamente sobre las regiones con dichos colores llevara a cabo la segunda etapa, búsqueda de formas. La búsqueda de regiones con colores de interés presenta una eficiencia computacional mucho mayor a la búsqueda de formas, por ello se ha optado por restringir la búsqueda de formas a regiones más pequeñas.

Por tanto el esquema final que seguirá el algoritmo es el incluido en la figura 3-1.

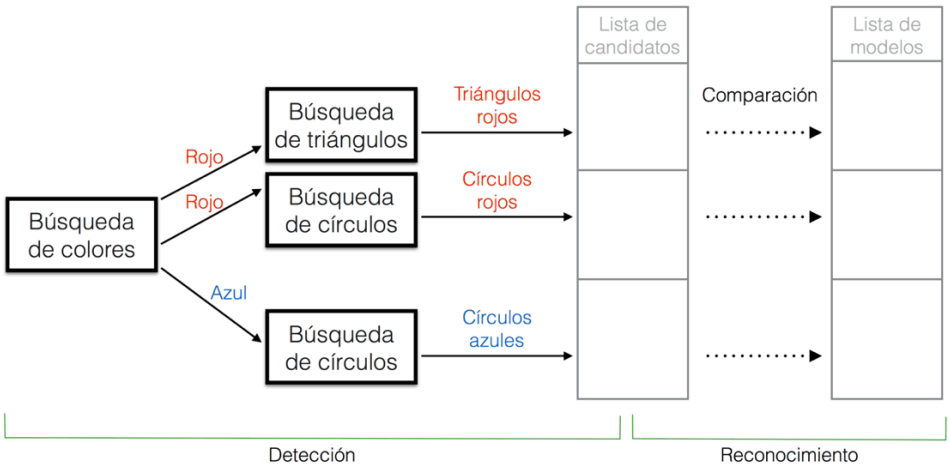

Figura 3-1: Esquema del diseño del algoritmo

A continuación se expone de forma detallada cada etapa del algoritmo.

3.2 Detección de colores

La primera etapa del sistema es la detección de color, como ya se ha comentado, para la búsqueda de las señales elegidas los colores interesantes son el rojo y el azul, el algoritmo los detectará y guardará la posición de los píxeles de dicho color mediante el uso de una máscara binaria donde el "0" representado como negro es ausencia de color buscado y el "1" representado como blanco significa que el color buscado está en ese pixel.

Cabe decir que una máscara binaria representará los colores rojos detectados y otra máscara los azules, así se podrán buscar las formas específicas que estén presentes en cada una de ellas, conociendo en todo momento si la forma tiene color azul y rojo. Esto es interesante ya que, por ejemplo, no es necesario buscar triángulos en la máscara azul porque no existe ninguna señal triangular azul.

De cara a la creación de la máscara de color, se seleccionarán un rango mínimo y un rango máximo utilizando el estándar de color HSV, que aunque no es tan conocido como otros estándares como son RGB o BGR implica una ventaja para este caso concreto. HSV permite abarcar de forma más fácil un mismo tono sin tener en cuenta su brillo o saturación.
El estándar de color HSV permite describir el color en función de: el matiz de color, la cantidad de saturación y el nivel de brillo.

Trabajar con este sistema permitirá abarcar de forma más sencilla un color real captado en distintas situaciones de luminosidad o mediante diferentes cámaras.

3.3 Detección de formas

La segunda etapa del sistema corresponde a la detección de formas, una vez que las regiones del color indicado han sido delimitadas, se realiza una extracción de contornos sobre las máscaras de color. Sobre dichos contornos el algoritmo buscará las formas mencionadas anteriormente y entonces ya se podrán considerar dichos resultados como ROI (Region Of Interest) que serán posteriormente candidatos para la etapa de reconocimiento del algoritmo. Como se ha comentado anteriormente, en este trabajo la atención está centrada únicamente sobre las formas triangulares y circulares.

3.3.1 Detección de triángulos (y cuadrados)

Para la detección de triángulos en primer lugar se detectarán polígonos y se filtrarán aquellos que tengan tres vértices, que por tanto tendrán también tres lados y consideraremos triángulo.

Aun así, un polígono de tres lados detectado como un triángulo, no tiene porqué ser un triángulo de nuestro interés. Por ejemplo, en la figura 3-2 se aprecia un triángulo detectado por la función, que carece de interés para nuestro cometido. Sin embargo en la figura 3-3 observamos un triángulo que debe atraer el interés del algoritmo por ser similar en forma a una señal triangular de tráfico.

Figura 3-2: Ejemplo de triángulo que carece de interés (no equilátero)

Figura 3-3: Ejemplo de triángulo de interés para el algoritmo (equilátero)

Por ello el algoritmo filtrará los triángulos encontrados según la longitud de sus lados, buscando triángulos equiláteros y aplicando un margen debido a que las señales no tienen por qué ser vistas completamente de frente. Con esta aproximación evitaremos falsos positivos.

Sirva como ejemplo de las señales que pertenecerían a esta rama del algoritmo la siguiente figura:

Figura 3-4: Ejemplo de señal triangular de advertencia de peligro

El mismo algoritmo de detección de triángulos podría ser usado, tras aplicar los cambios oportunos, para identificar señales cuadradas con bastante precisión en una fase futura del trabajo en la que se quisiera ampliar el abanico de señales reconocidas. De nuevo con un refinamiento basado en el cálculo de la longitud de sus lados.

Figura 3-5: Ejemplo de señal rectangular de indicaciones generales

3.3.2 Detección de círculos

En el mundo del procesamiento de imagen es muy conocida y muy empleada la función de la transformada de Hough [1] para el reconocimiento de circunferencias. Es por ello que será la empleada para el reconocimiento de círculos en este algoritmo, al igual que en [1].

Dicha transformada ha de ser empleada tanto para los contornos azules como para los rojos en búsqueda de señales de los tipos siguientes:

Figura 3-6: Ejemplo de señal circular de prohibición

Figura 3-7: Ejemplo de señal circular de obligación

3.4 Reconocimiento

La tercera y última etapa consiste en el reconocimiento de la región extraída anteriormente. En el momento en el que una ROI ha llegado hasta este punto será considerada como señal candidata o candidato a partir de ahora.
El procedimiento será el mismo sea cual sea el color o la forma del candidato, lo único que variará será las plantillas o modelos con los que sea comparado. Así un candidato del que se sepa que es triangular y rojo, será comparado únicamente con modelos de ese tipo de señal.

La comparación será hecha por correlación directa tras ajustar las dimensiones del candidato a las del modelo.

4 Desarrollo

4.1 Introducción

En este capítulo se expondrán cada una de las distintas etapas en las que se ha dividido el desarrollo del algoritmo. Desde una primera etapa centrada en la detección sobre imágenes estáticas, hasta cada una de las etapas expuestas en el capitulo anterior: detección de color y contorno, detección de formas y reconocimiento.

4.2 Primeras etapas

A modo de primera aproximación al algoritmo, se ha desarrollado un programa similar al final que actúa sobre una única imagen en lugar de sobre los fotogramas de un video.
Esto ha permitido analizar los parámetros para la detección de color, de forma y apreciar con más detalle los resultados obtenidos para cada una de las imágenes con el objetivo de mejorar los valores de detección para su posterior aplicación sobre el video.

En primer lugar, las imágenes utilizadas han sido de una única señal por imagen, captada de frente y no intentando imitar el ángulo, la situación o la velocidad con la que se captaría desde un coche en movimiento por la carretera. Por ello podríamos considerar esta etapa como un trabajo con imágenes en una situación ideal. Esta etapa nos ofrece unos valores generales sobre los que seguir trabajando en las posteriores etapas. Las figuras siguientes son ejemplos que se han utilizado en esta etapa del algoritmo:

Figura 4-1: Imagen de señal utilizada en primera etapa.

Figura 4-2: Imagen de señales utilizada en primera etapa.

La figura 4-1 representa una prueba sencilla, el color de la señal se diferencia notablemente del resto de colores de la imagen y el tono de rojo es bastante uniforme.

En la figura 4-2 se aprecian varias señales y aunque en un principio parecen distinguirse claramente del fondo, en este caso podemos evaluar distintos tonos de rojo, además de valorar cómo se comporta el reconocimiento de formas dependiendo de la distancia a la que se encuentra la señal.

En la segunda etapa se han evaluado los parámetros prefijados anteriormente sobre un fotograma de un video real. En esta etapa se está trabajando ya con fotogramas ejemplo que se puede encontrar el algoritmo en una situación real. En este caso se ha evaluado de igual manera que en la anterior etapa, tratando de perfeccionar los valores. A continuación, una imagen ejemplo para esta segunda etapa:

Figura 4-3: Imagen de señales utilizada en segunda etapa.

En la figura 4-3 lo primordial era estudiar los valores de la detección de color de la que se hablará a continuación.

4.3 Detección de color y contorno

En primer lugar mencionar que funcionalmente se han separado la detección de color y la detección de contorno aunque están íntimamente relacionadas en este algoritmo y por eso se explicarán en la misma subsección.

Para la detección de color se ha utilizado la función de OpenCV *cv2.inRange()*, en este caso, los valores utilizados han sido (formato HSV):

- Valores rojo:
  - Rango: H: 0 – 10, S: 0 – 255, V: 50 – 255.
  - Rango: H: 150 – 200, S: 0 – 255, V: 50 – 255.

- Valores azules:
  - Rango: H: 100 – 120, S: 130 – 255, V: 50 – 255.

La razón principal de que el valor rojo tenga dos rangos es porque en el cono de color de HSV el color rojo se ubica en las posiciones correspondientes al principio de los valores y al final (al ser un cono los extremos están conectados, por tanto son valores consecutivos). De hecho se puede observar que los valores para saturación y brillo son idénticos para ambos rangos.

Aplicando estos valores para la detección de color, los resultados obtenidos se pueden apreciar en las figuras 4-4, 4-5 y 4-6.

Figura 4-4: Detección de color rojo sobre figura 4-1.

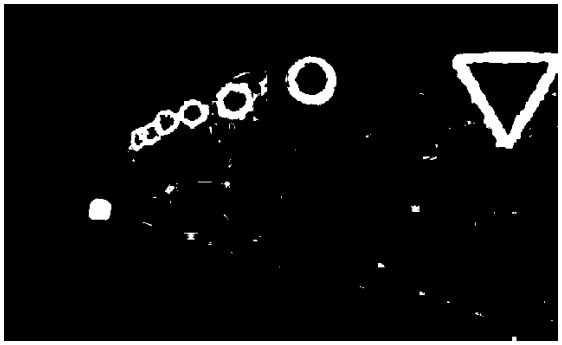

Figura 4-5: Detección de color rojo sobre figura 4-2.

Figura 4-6: Detección de colores azul y rojo sobre figura 4-3.

Buscando además contornos sobre las máscaras de color anteriores conseguimos eliminar píxeles sueltos que pudieran causar falsos positivos, limitando, aún más, con un simple paso la imagen de entrada para la posterior detección de formas.
Para encontrar los contornos se ha usado la función de OpenCV *cv2.findContours()*, añadiéndole un filtro a su salida para filtrar los contornos por tamaños y mantener únicamente los contornos de nuestro interés.

A continuación los contornos vienen marcados en color rojo sobre las imágenes originales.

Figura 4-7: Detección de contornos rojos sobre figura 4-1.

Figura 4-8: Detección de contornos rojos sobre figura 4-2.

Figura 4-9: Detección de contornos azules y rojos sobre figura 4-3.

En la figura 4-9 se han incluido también los contornos azules (al igual que en la figura 4-6 se ha incluido la detección de color azul) para demostrar que el proceso seguido es el mismo independientemente del color que se busque.

En estos ejemplos se aprecia que mediante la detección de color y la posterior búsqueda de contornos se simplifica la imagen de entrada manteniendo las ROI, con lo cual será más sencillo y más eficiente detectar formas directamente sobre los contornos, ya que se evalúan máscaras binarias (donde 0 corresponde a la ausencia del color buscado y 1 a su presencia) en lugar de matrices de píxeles que representarían una imagen.

Además conviene recordar que aunque en las figuras 4-6 y 4-9 se hayan representado la detección y contornos de ambos colores sobre la misma imagen, cada color tiene su propia máscara y su propia matriz de píxeles de contorno. Por tanto la diferenciación entre tipos de señales según su color se puede seguir haciendo.

4.3.1 Problemas encontrados y soluciones adoptadas

En esta parte del algoritmo no se han encontrado apenas problemas de importancia más que los esperados, que son:

- La dificultad de encontrar los valores HSV para detectar los colores buscados con exactitud.
  Ante este problema se ha optado por ampliar el rango para no dejar fuera posibles candidatos en condiciones pobres de iluminación, esto ha desembocado en que la máscara de color no sólo incluía las ROI si no también otras regiones de la imagen que tenían colores marrones o rosas. Sin embargo gracias a la búsqueda de contornos y a los siguientes pasos, la inclusión de regiones extra no ha sido un problema.

- La inclusión de pequeños contornos insignificantes para la solución final, que incrementan el coste computacional del algoritmo.
  Cuanto mayor número de contornos se encuentren, mayor tiempo llevará el posterior análisis en busca de formas en esos contornos. Es por ello que se ha optado por eliminar los contornos demasiado pequeños como para suponer una ROI. Esto se ha llevado a cabo dentro de la función propia *encuentraContornos()*, eliminando los contornos menos de 100 puntos.

4.4 Detección de formas

Partiendo de los colores y contornos extraídos en el paso anterior, y de acuerdo al esquema de diseño presentado en la figura 3-1, han de plantearse dos vías en función de las formas a detectar.

4.4.1 Formas triangulares

La función *cv2.approxPolyDP()* busca polígonos en una lista de puntos (en este caso la lista de puntos es la lista de puntos que forman el contorno) y retorna una lista de polígonos encontrados dados sus puntos. Con un simple filtro sobre el número de puntos de cada polígono se pueden filtrar polígonos triangulares y rectangulares (que son los de interés para el algoritmo). En este caso, nos interesaremos únicamente por los triangulares ya que son el primer objetivo de este trabajo.

Es por todo lo anterior, por lo que la función propia *buscarTriangulos()* filtra el resultado de *approxPolyDP()* a sólo los triángulos y después calcula la longitud de cada lado del triángulo para comprobar su equilateralidad. Este cálculo se realiza mediante la fórmula:

$$\text{Distancia 1-2} = (x_2-x_1)^2 + (y_2-y_1)^2$$

Que calcula la distancia entre dos puntos cualesquiera en un espacio bidimensional. Aplicando dicha formula para cada uno de los 3 lados del triángulo, obtenemos 3 resultados:

- Distancia 1-2
- Distancia 2-3
- Distancia 3-1

Si estas distancias fueran iguales se trataría de un triángulo equilátero perfecto. Debido a que estamos tratando formas provenientes de una situación real, conviene ponerle un margen de tolerancia. Dicho margen se ha fijado en un 20% para que elimine falsos positivos claros.

Cuando un triángulo ha pasado los filtros comentados anteriormente, se consideraría una ROI preparada para la fase de reconocimiento.

En las siguientes figuras se puede apreciar mediante una caja verde los triángulos reconocidos mediante este algoritmo y que han superado los filtros mencionados anteriormente por tanto candidatos para la siguiente fase, el reconocimiento.

Figura 4-10: Detección de triángulo (interior) preparado para ser candidato en fase de reconocimiento.

Figura 4-11: Detección de triángulo (exterior) preparado para ser candidato en fase de reconocimiento.

Figura 4-12: Detección de triángulo (interior y exterior) preparados para ser candidatos en fase de reconocimiento.

Como puede apreciarse, en la figura 4-10 el algoritmo ha detectado el triángulo interior de la señal, en la figura 4-11 el triángulo exterior y en la figura 4-12 ambos triángulos, interior y exterior. Lo normal es que dependiendo del fotograma de video que procese (una misma señal por norma general aparecerá en varios fotogramas) el algoritmo capte el triángulo interior, exterior o ambos para una misma señal, por tanto esto no debería de preocuparnos. Lo que sí debería hacerlo seria que una señal no fuera captada por el algoritmo de detección.

4.4.1.1 Problemas encontrados y soluciones adoptadas

En esta etapa han surgido algunos problemas de mayor envergadura que en la anterior. Sin embargo todos han sido finalmente resueltos de una forma eficaz:

- Manejo de los tipos de datos.
  Probablemente el problema menos importante aunque ha entorpecido el desarrollo del algoritmo. Al trabajar con las listas de puntos para los contornos y las listas de puntos de los polígonos, en ocasiones han aparecido dificultades para calcular la distancia de los lados partiendo de los puntos en formato *numpy.ndarray*. Además de para dibujar después el rectángulo contenedor de la ROI.
  La solución no ha sido otra que trabajar más a conciencia con los arrays y las tuplas que provee Python y en la medida de lo posible convertir los *numpy.ndarray* a arrays de Python.

- Regiones marcadas como triángulos cuando no lo eran.
  Este ha sido un problema surgido en el primer momento de desarrollo de esta parte del algoritmo. Ha sido solucionado de forma muy eficaz introduciendo el filtro de triángulos equiláteros.

- Señales no captadas por el algoritmo de detección.

En varias ocasiones al procesar un video ha habido señales triangulares no captadas por el detector de formas, siguiendo el hilo del programa desde el principio se ha descubierto que el problema estaba en la detección de color y no en la detección de forma. Para solucionarlo se ha extraído el fotograma en el que aparecía dicha señal para extraerle el color y se han modificado los rangos de color para que detectaran esa señal. Los rangos finales son los expuestos en la sección 4.2

4.4.2 Formas circulares

Esta vez partiendo de la máscara de color y no de los contornos, y mediante la función *cv2.HoughCircles()* incluida en OpenCV se ha aplicado la conocida transformada de Hough para detectar círculos. Dicha función requiere que la entrada sea una imagen en escala de grises. Es por ello que lo primero que hará la función *buscarCirculos()* del algoritmo será unir el fotograma real y la máscara de color y después convertir el resultado a escala de grises para que *cv2.HoughCircles()* pueda trabajar.

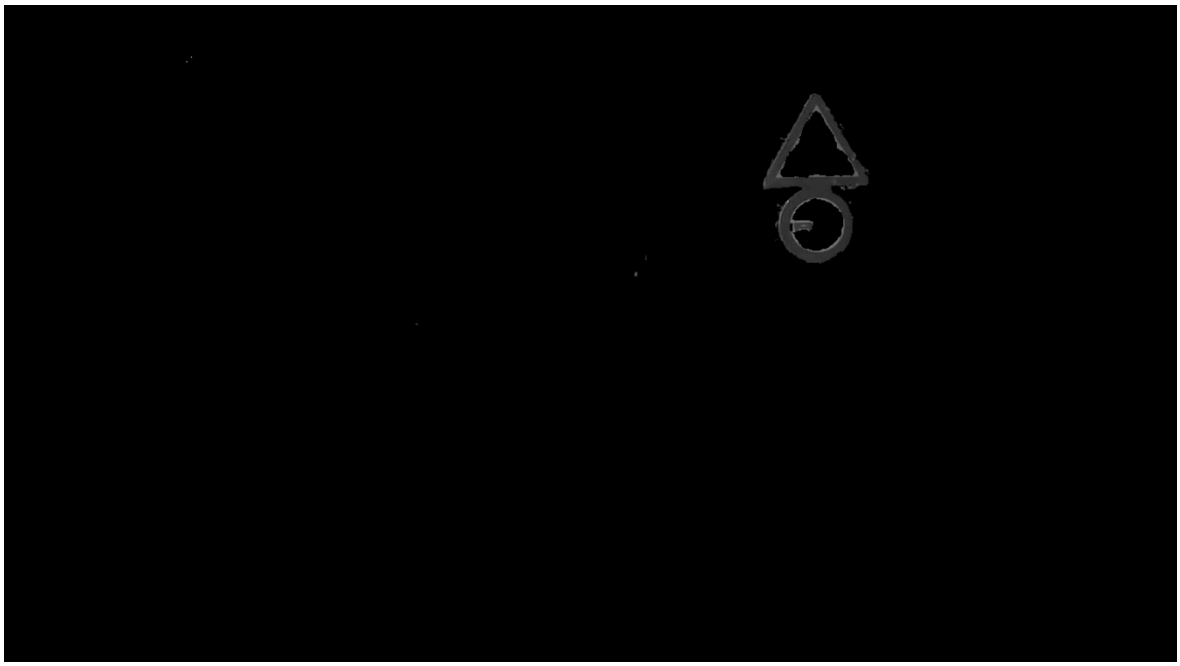

Figura 4-13: Escala de grises para transformada de Hough sobre figura 4-10.

Además, mediante parámetros de la función de OpenCV se puede filtrar que los círculos encontrados respondan a un determinado rigor, esto será importante para eliminar falsos positivos y reducir considerablemente el tiempo de procesamiento.

En las siguientes figuras se aprecian círculos encontrados por el algoritmo, delimitados por un rectángulo gris.

Figura 4-14: Detección de círculo (interior) preparado para ser candidato en fase de reconocimiento.

Figura 4-15: Detección de círculo (exterior) preparado para ser candidato en fase de reconocimiento.

Como se mencionaba anteriormente, lo normal es que, para una misma señal haya varios fotogramas que la contengan y, en esos fotogramas la ROI varíe entre la forma interior, la forma exterior o ambas, las figuras 4-14 y 4-15 son un claro ejemplo de esto.

Figura 4-16: Detección de triángulo (interior y exterior) preparados para ser candidatos en fase de reconocimiento.

Gracias a la abstracción de la función *buscarCirculos()* es viable encontrar círculos sea cual sea su color, simplemente proporcionándole como parámetro a la función la máscara del color sobre el que buscar. En la figura 4-16 con el mismo procedimiento se ha hallado una señal circular azul.

4.4.2.1 Problemas encontrados y soluciones adoptadas

A continuación se exponen dos grandes problemas encontrados en el uso de la transformada de Hough para la detección de círculos.

- Demasiados círculos encontrados por la función *cv2.HoughCircles()*.
  Al hacer pruebas con el objetivo de detectar círculos que no se estaban detectando, se ha bajado el rigor pasado como argumento a la función de la transformada. Sin embargo esto ocasionaba problemas mayores relacionados con el rendimiento de la función, ya que dentro de una misma región encontraba numerosos círculos sin sentido como puede apreciarse en la figura 4-17.

Figura 4-17: Ejemplo de problema encontrado en detección de círculos

La solución por la que se optó fue por trabajar algo más sobre los rangos de color para que no se perdieran zonas del círculo, hacer un suave difuminado antes de aplicar la transformada de Hough y ajustar el rigor de la función lo máximo posible para evitar falsos positivos.

- Rendimiento de la función *cv2.HoughCircles()*.
  El mayor problema encontrado ha sido que en el momento en el que la imagen en la que buscar era muy grande o los círculos encontrados eran muchos, la transformada de Hough ralentizaba notablemente el algoritmo. En principio se pensó en evitar el uso de la transformada en favor del uso de la función *cv2.approxPolyDP()* usada anteriormente para formas poligonales, ya que un círculo no es más que un polígono de infinitos lados. Sin embargo se desestimo dicho cambio ya que al ampliar los rangos de color, difuminar y ajustar el rigor de la función (solución del problema anterior), no solo se consiguieron mejores resultados si no que se consiguieron tiempos de ejecución mucho menores debido a que la cantidad de círculos encontrados disminuyó considerablemente.

4.5 Reconocimiento

La fase de reconocimiento es la última del algoritmo, cuando una región ha llegado hasta esta fase se tratará como ROI y se asumirá que se trata de una señal. Por ello las regiones que lleguen hasta este paso se llamarán candidatas y serán comparadas con cada una de las señales que forman parte del grupo, que serán los modelos.

En el primer momento en el que el programa es lanzado se cargan los modelos en el sistema, ya que es mucho más eficiente frente a cargarlos cada vez que aparezca un nuevo candidato. Para ello, la función propia *cargarModelos()* generaliza la carga de cada grupo de señales (triangulares rojas, circulares rojas, circulares azules) en una lista de modelos.

El candidato será entonces redimensionado para cumplir las dimensiones del modelo (todos los modelos de un mismo grupo de señales tienen las mismas dimensiones, aunque a efectos prácticos se extrae las dimensiones de cada modelo y se redimensiona el candidato con esos datos ya que el coste de hacer esto no es apenas significante).

La comparación entre modelo y candidato se realiza mediante comparación de plantillas (template matching). Al tener ambas las mismas dimensiones el algoritmo sólo llevará a cabo una comparación para cada par, indicando el índice de similitud entre dicho par. Se entenderá por tanto que la ROI corresponde al modelo con el que tiene mayor índice de similitud.

Llevar a cabo la comparación directamente sobre las imágenes a color no es eficaz, ya que los colores de los píxeles raramente van a coincidir y además el coste computacional es más elevado. Por ello desde un primer momento se pensó en llevar a cabo un preprocesamiento de las imágenes antes de la citada comparación. En este caso la primera solución fue aplicar una detección de bordes (Canny) [1] y comparar los bordes extraídos del candidato con los extraídos del modelo. En las siguientes figuras se pueden observar ejemplos de la detección de bordes sobre candidatos.

Figura 4-18: Ejemplo de detección de bordes sobre candidato

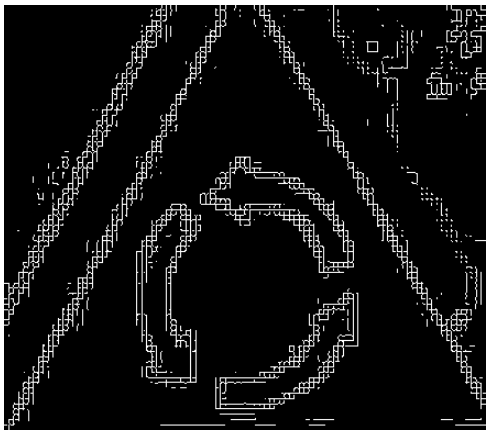

Figura 4-19: Ejemplo de detección de bordes sobre candidato

4.5.1 Problemas encontrados y soluciones adoptadas

- Eficacia del algoritmo de reconocimiento.
  Mediante la técnica de detección de bordes, el rendimiento del algoritmo de reconocimiento es significativamente mayor, sin embargo no es todo lo eficaz que se esperaría de él.
  Por ello se ha pensado en otra solución alternativa que se basa en la utilización de un umbral (pudiendo ser de carácter adaptativo) para separar el símbolo o dibujo dentro de la señal del resto de señal. Aplicando esta técnica de umbralización se conseguiría tener el fondo en blanco y el símbolo en negro, al igual que en la imagen modelo. Y de nuevo la comparación sería eficiente en rendimiento ya que se siguen comparando imágenes que tienen dos vías (blanco y negro)

- Tratamiento del ruido que rodea a la forma de la señal exteriormente.
  Como puede observarse en la esquina superior izquierda de la figura 4-18, se detectan bordes fuera de lo que es el cuerpo de la señal. Esto puede producirse porque detrás de la señal hubiera un árbol o cualquier otro elemento. Dicho ruido externo altera el índice de similitud entre candidato y modelo, por ello la solución propuesta es eliminarlo. Para su eliminación, ya que conocemos los vértices del triángulo del candidato, lo mejor sería convertir a negro todos los píxeles exteriores a dicho triángulo detectado anteriormente.

- Tratamiento del candidato cuando la forma detectada es la interior o la exterior
  Hay una diferencia muy clara que se ha ido mencionando en este documento al exponer algunas figuras. Como por ejemplo entre las figuras 4-18 y 4-19 o entre las figuras 4-14 y 4-15 o, el ejemplo más claro, en la figura 4-12. Una señal vertical, sea de la forma que sea, tiene un borde, y al buscar la forma de ese borde, el algoritmo se encuentra con una forma exterior y una interior. Al comparar frente a un modelo, se está asumiendo que el borde del candidato es el exterior. Por ello la solución pasa por la búsqueda del borde interno del candidato, con unos valores más laxos y su comparación únicamente con el interior del modelo.

Las soluciones mencionadas anteriormente son complementarias y podrían ponerse en funcionamiento todas a la vez, lo cual haría aumentar la eficacia del reconocimiento.

5 Integración, pruebas y resultados

5.1 Integración

Como se ha mencionado anteriormente, el objetivo de este trabajo no consistía en implantar un sistema concreto, si no en el diseño e implementación de un algoritmo que pudiera ser usado para diferentes fines y sobre todo que fuera adaptativo a diferentes tipos de cámara. Por ello la integración no ha sido basada sobre una cámara en concreto, si no que se han barajado varios tipos para las pruebas. Así, el uso final podría variar desde una aplicación móvil que utilizara la cámara del Smartphone hasta una implementación basada en el sistema de un coche que se aprovechara de cada cámara instalada por el fabricante.

Por ello se dan dos premisas para estos casos concretos:

- En caso de su utilización incorporada al sistema de seguridad activa de un vehículo, es recomendable situar la cámara lo más alta posible y en posición horizontal.

- En caso de su empleo en una aplicación móvil, es básico trabajar en el soporte que unirá el Smartphone con el vehículo, así se evitarán fotogramas borrosos.

En base a esto, las pruebas recogidas en este capítulo han tratado de variar la cámara para hacer el algoritmo lo más adaptativo posible. Se han utilizando las siguientes cámaras:
- Cámara de un iPhone 5s
- Cámara de un iPhone 4
- GoPro Hero 3 Silver.

Los videos grabados han sido procesados por el algoritmo en un Mac con la librería de OpenCV instalada y el algoritmo ha sido desarrollado íntegramente en lenguaje de programación Python.

5.2 Pruebas realizadas

Ante la dificultad de probar el algoritmo sobre vídeo en tiempo real, la forma de llevar a cabo las pruebas ha consistido en grabar recorridos tanto por carreteras que atraviesan poblados, como por carreteras secundarias (único carril en un sentido) como por autovías (varios carriles en un mismo sentido) y después procesar el video con el algoritmo midiendo tiempos, para tratar, en la medida de lo posible, de lograr que el algoritmo funcione en tiempo real.

Con el objetivo de poner a prueba el algoritmo, los videos empleados para las pruebas han sido grabados en diferentes días, diferentes tiempos dentro del día y diferentes condiciones de luminosidad, que pueden venir otorgadas por la posición del sol (de cara, de espaldas o desde un lado). Siempre manteniendo como premisa que la escena debía ser diurna, de cara

a centrar la detección y el reconocimiento en escenas en las que aún con poca luz o con luminosidad dispar, la luz fuera de carácter natural.

Al haber trabajado con el video en un formato manipulable, se han podido hacer pruebas variando la resolución del video de entrada. Dicha resolución se ha variado entre 720p y 1080p en búsqueda de la mayor capacidad de detección y reconocimiento y el menor tiempo invertido en procesamiento, los datos apreciados se comentarán en la siguiente sección.

Las imágenes captadas por ambas cámaras tienen una cantidad similar de FPS (Fotogramas Por Segundo), entorno a 30 FPS. En la tabla de resultados de la siguiente sección también se observa que el mismo video ha sido evaluado para diferente cantidad de fotogramas a procesar. La cantidad de fotogramas a procesar varía entre todos los fotogramas y un fotograma procesado de cada tres de video. Al procesar mayor número de fotogramas, las posibilidades de no detectar o no reconocer una señal disminuyen ya que una misma señal aparece en más fotogramas procesados, sin embargo cuantos más fotogramas se evalúen, más costoso computacionalmente será el procesado de dicho video. Aplicando esto a la búsqueda de un procesamiento en tiempo real, es necesario que la evaluación de un fotograma emplee menos tiempo que el que supone dicho fotograma en el video.

5.3 Resultados

En la siguiente tabla se presentan los resultados obtenidos en las pruebas realizadas sobre algunos de los videos grabados. En ella se indica la resolución empleada, la cantidad de fotogramas analizados, el porcentaje de tiempo que supone el procesamiento del algoritmo sobre el tiempo total del video y los porcentajes de señales detectadas y reconocidas sobre el total de las señales que aparecen en los videos.

Tabla 5-1: Tabla de resultados.

Video	Calidad	Frames procesados	Longitud del video (s)	Tiempo empleado (s)	% tiempo	Señales totales	Señales detectadas	% señales detectadas	Señales reconocidas	% señales reconocidas
7398-med	720	1 de 1	11	18,43	167,5%	3	2	66,7%	2	66,7%
7398-med	720	1 de 2	11	11,43	103,9%	3	2	66,7%	2	66,7%
7398-med	720	1 de 3	11	9,14	83,1%	3	2	66,7%	1	33,3%
7398	1080	1 de 1	11	40	363,6%	3	2	66,7%	1	33,3%
7398	1080	1 de 2	11	22,2	201,8%	3	2	66,7%	1	33,3%
7398	1080	1 de 3	11	16,5	150,0%	3	1	33,3%	1	33,3%
7401-med	720	1 de 1	8	13,8	172,5%	2	1	50,0%	1	50,0%
7401-med	720	1 de 2	8	8,42	105,3%	2	1	50,0%	1	50,0%
7401-med	720	1 de 3	8	6,7	83,8%	2	1	50,0%	1	50,0%
7401	1080	1 de 1	8	31,7	396,3%	2	1	50,0%	1	50,0%
7401	1080	1 de 2	8	18,2	227,5%	2	1	50,0%	1	50,0%
7401	1080	1 de 3	8	12,75	159,4%	2	1	50,0%	1	50,0%
7402-med	720	1 de 1	37	68,5	185,1%	6	4	66,7%	3	50,0%
7402-med	720	1 de 2	37	42,98	116,2%	6	4	66,7%	3	50,0%
7402-med	720	1 de 3	37	35,2	95,1%	6	4	66,7%	3	50,0%
7402	1080	1 de 1	37	151,1	408,4%	6	4	66,7%	2	33,3%
7402	1080	1 de 2	37	100	270,3%	6	4	66,7%	2	33,3%
7402	1080	1 de 3	37	76,3	206,2%	6	4	66,7%	2	33,3%

Video	Calidad	Frames procesados	Longitud del video (s)	Tiempo empleado (s)	% tiempo	Señales totales	Señales detectadas	% señales detectadas	Señales reconocidas	% señales reconocidas
7403-med	720	1 de 1	35	67,7	193,4%	9	8	88,9%	5	55,6%
7403-med	720	1 de 2	35	41,6	118,9%	9	8	88,9%	4	44,4%
7403-med	720	1 de 3	35	33,3	95,1%	9	8	88,9%	4	44,4%
7404-med	720	1 de 1	21	49,4	235,2%	11	9	81,8%	6	54,5%
7404-med	720	1 de 2	21	30,5	145,2%	11	9	81,8%	5	45,5%
7404-med	720	1 de 3	21	22,5	107,1%	11	9	81,8%	4	36,4%
7427-med	720	1 de 1	11	23,56	214,2%	2	2	100,0%	2	100,0%
7427-med	720	1 de 2	11	14,3	130,0%	2	2	100,0%	2	100,0%
7427-med	720	1 de 3	11	11,6	105,5%	2	2	100,0%	2	100,0%

Como puede apreciarse en la tabla, el algoritmo es mejor en la etapa de detección frente a la etapa de reconocimiento. Situándose por encima del 80% en la mayoría de los casos y siempre por encima del 50% de eficacia en la detección de las señales que se presentan en el video. Sin embargo, el porcentaje de acierto en el reconocimiento es menor, variando entre el 33% y el 100%, situándose el caso medio entorno al 50%.

Además, observando los resultados en función de la cantidad de fotogramas analizados del video, no se aprecia diferencia en la cantidad de señales detectadas. Sin embargo si se advierte una diferencia en la cantidad de señales reconocidas. Esto, unido a la observación anterior, hace pensar que la etapa de reconocimiento puede ser mejorada en un futuro, incrementando así la capacidad de éxito del algoritmo.

Otro dato interesante es el tiempo empleado según la resolución escogida (720p frente a 1080p). Obviamente cuanto mayor es la resolución, mayor es el tiempo empleado por el algoritmo para dicho video. A 1080p el coste computacional se acerca al doble del coste a 720p.

En búsqueda de la adecuación del algoritmo a su funcionamiento en tiempo real, y observando en los resultados de la tabla que únicamente en los análisis a 720p y 1 de cada 3 fotogramas analizados el tiempo ha estado por debajo del tiempo real del video, se podría concluir que la mejor aproximación para un sistema en tiempo real es 720p y evaluar 1 de cada 3 fotogramas. En la mayoría de las ocasiones esta configuración ha estado por debajo del tiempo del video, estableciendo una media de tiempo empleado de un 90% del tiempo del video real. De todas maneras en la sección 6.2 se abordará este tema de nuevo.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Este trabajo tenía como propósito la implementación de una primera aproximación a un algoritmo de detección y reconocimiento de señales viales dejando a un lado la IA (Inteligencia Artificial) y el aprendizaje automático. Además, un objetivo secundario de este trabajo era tratar de que el tiempo computacional del algoritmo le permitiera funcionar en tiempo real.

Para conseguir dicho propósito se ha estudiado el estado del arte (capítulo 2) y se ha diseñado un algoritmo en dos fases, detección y reconocimiento, según lo explicado en el capítulo 3 y esquematizado en la figura 3-1. A partir del diseño del algoritmo, se ha desarrollado cada una de las etapas de un primer algoritmo básico que se ha ido completando y mejorando según lo expuesto en el capítulo 4.

A la vista de los resultados, podemos concluir que se han cumplido los objetivos del trabajo y se ha conseguido un algoritmo robusto en su etapa de detección y genérico aunque eficaz en su etapa de reconocimiento. Además, el diseño del algoritmo permite, con una pequeña modificación, abarcar aún más señales, ya que únicamente han de añadirse colores a la búsqueda relatada en las secciones 3.2 y 4.3 o nuevas formas a buscar a las relatadas en 3.3 y 4.4. Además es un algoritmo adaptativo, en el caso en el que se crearan nuevas señales de tráfico a las ya existentes [1] , sólo haría falta añadirlas al conjunto de modelos que se trata en la sección 4.5 de este trabajo.

6.2 Trabajo futuro

Partiendo del trabajo ya realizado se proponen varias mejoras para lograr que el algoritmo sea lo más eficaz posible.

En primer lugar, la implementación y prueba de las mejoras propuestas en la sección 4.5.1:

- La implementación de una comparación basada en umbralización del candidato y el modelo, creando por tanto imágenes con pixeles únicamente blancos o negros y las siluetas rellenas, lo cual elevaría el índice de similitud de las comparaciones, beneficiando la eficacia de la etapa de reconocimiento.
- La búsqueda de la forma interior de la señal, asegurando así que la comparación se esta llevando en todos los casos sobre la forma interior de la señal (el triángulo o círculo interior detectado).
- La eliminación del ruido externo a la ROI extraída. En caso de una señal triangular por ejemplo, la importancia ha de tenerla la zona interior del triángulo, no la zona exterior al triángulo e interior al cuadrado que lo circunscribe.

En cuanto al tiempo y la forma de asegurar que el algoritmo es capaz de trabajar en tiempo real, el trabajo futuro se basaría en limitar el tiempo no controlado por el algoritmo. Dicho

tiempo es el que proviene de las funciones propias de OpenCV. En concreto el mayor problema reside en el tiempo empleado por la transformada de Hough. Para limitar este tiempo se podría emplear un hilo única y exclusivamente para dicha transformada, que una vez superara un cierto tiempo límite, fuera destruido para que el algoritmo pudiera seguir ejecutándose en tiempo real.

Además, una vez se hubiera decidido el objetivo final del algoritmo, conviene estudiar la cámara concreta empleada y adecuar los valores a sus niveles de calidez, luminosidad, etc. Pudiendo llegar incluso a aplicar un preprocesado del fotograma antes de evaluarlo mediante el algoritmo.

Por último, se propone el estudio de algoritmos de seguimiento o *tracking* como los aplicados en [1] y [1].

Referencias

[1] Benjamin Höferlin, Klaus Zimmermann, "Towards Reliable Traffic Sign Recognition", pp. 324-329, 2009.

[2] Meng-Yin Fu, Yuan-Shui Huang, "A Survey of Traffic Sign Recognition", *Proceedings of the 2010 International Conference on Wavelet Analysis and Pattern Recognition, Quingdao*, pp. 119-124, 11-14 Julio 2010.

[3] Andreas Møgelmose, Mohan Manubhai Trivedi, and Thomas B. Moeslund, "Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey", *IEEE Transactions on Intelligent Transportation Systems*, VOL. 13, Nº 4, pp. 1484-1497, Diciembre 2012.

[4] Jack Greenhalgh, Majid Mirmehdi, "Real-Time Detection and Recognition of Road Traffic Signs", *IEEE Transactions on Intelligent Transportation Systems*, VOL. 13, Nº 4, pp. 1498-1506, Diciembre 2012.

[5] Zhiyong Liu, "A Survey of Intelligence Methods in Urban Traffic Signal Control", *IJCSNS International Journal of Computer Science and Network Security*, VOL.7 Nº 7, pp 105-112, Julio 2007.

[6] Zhe Zhu, Dun Liang, Songhai Zhang, "Traffic-Sign Detection and Classification in the Wild", pp 2110-2118, 2016.

[7] Igor Bonaci, "Addressing false alarms and localization inaccuracy in traffic sign detection and recognition", *16th Computer Vision Winter Workshop*, 2-4 Febrero 2011.

[8] Karla Brkic, Axel Pinz, Sinisa Segvic, "Traffic sign detection as a component of an automated traffic infrastructure inventory system", 2009.

[9] Ministerio de Obras Públicas y Transportes, "Catálogo y Significado de las señales", *Señales verticales de circulación*, Tomo 2, Junio 1992.

[10] Wikipedia, "Anexo: Señales de tráfico de reglamentación de España", *Señales de tráfico verticales de España*, Anexo, Mayo 2017.

[11] OpenCV, "http://www.opencv.org/".

[12] “Shape Detection & Tracking using Contours”, *OpenCV-srf*, Septiembre 2009, “http://opencv-srf.blogspot.com.es/2011/09/”.

[13] John Canny, “A Computational Approach to Edge Detection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL. PAMI-8, Nº 6, pp 679-698, Noviembre 1986.

[14] N. Barnes and A. Zelinsky, "Real-time radial symmetry for speed sign detection," Intelligent Vehicles Symposium, 2004 IEEE, pp. 566-571, 2004.

Glosario

GPS	Global Positioning System
SIFT	Scale-Invariant Feature Transform
OCR	Optical Character Recognition
API	Application Programming Interface
ADAS	Advanced Driver Assistance System
TSR	Traffic Sign Recognition
ROI	Region Of Interest
HSV	Hue Saturation Value
RGB	Red Green Blue
BGR	Blue Green Red
OpenCV	Open Source Computer Vision
FPS	Fotogramas Por Segundo
IA	Inteligencia Artificial