UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE UNA APLICACIÓN DE RECONOCIMIENTO DE SEÑALES DE TRÁFICO EN TIEMPO REAL MEDIANTE PYTHON

Autor: Jorge Lorente Monzó

Ingeniería Electrónica Industrial y Automática

Tutor: Cristian Ariel Olguin Pinatti

Curso académico: 2019-2020

ÍNDICE DE DOCUMENTOS:

ÍNDICE DE LA MEMORIA:

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Resumen:

Para realizar este trabajo nos basaremos en la detección de objetos mediante clasificadores en cascada basados en funciones de Haar.

Para ello haremos uso de dos programas de software disponibles de forma gratuita en Internet y de una colección grande de fotografías de señales de tráfico. Con esto, deberemos entrenar las imágenes para que el sistema pueda detectar de forma automática aquello que queramos identificar.

Con la ayuda de un código programado en lenguaje Python usaremos dichos clasificadores antes creados para la detección mediante imágenes estáticas importadas y posteriormente conseguiremos la detección de las señales en vivo a través de la webcam del ordenador, probando así el resultado final del proyecto.

Resum:

Per a realitzar aquest treball ens basarem en la detecció d'objectes per classificadors en cascada basats en funcions de Haar.

Per a això farem ús de dos programes de software disponibles de forma gratuïta a Internet i d'una col·lecció gran de fotografies de senyals de tràfic. En això, deurem entrenar les imatges per a que el sistema puga detectar de forma automàtica allò que vulgam identificar.

Amb l'ajuda d'un codi programat en llenguatge Python farem servir aquests classificadors abans creats per a la detecció mitjançant imatges estàtiques importades i posteriorment aconseguirem la detecció de les senyals en viu a través de la webcam de l'ordenador, provant així el resultat final del projecte.

Abstract:

To carry out this work we will base ourselves on the detection of objects using cascade classifiers based on Haar functions.

To do this we will use two software programs, available for free on the Internet, and a large collection of photographs of traffic signs. With this, we will have to train the images so that the system can automatically detect what we want to identify.

With the help of a code programmed in Python language, we will use these classifiers previously created for detection by imported static images and later we will achieve the detection of live traffic signs through the computer's webcam, thus testing the final result of the project.

DOCUMENTO I:
MEMORIA

1.- Introducción:

La introducción de la electrónica en los automóviles fue un paso muy importante, no solo para la eficiencia y la fiabilidad de las distintas partes del vehículo, sino también para la seguridad vial.

Hasta la década de los 60 los vehículos convencionales estaban formados básicamente por partes mecánicas. A partir de esta década, los fabricantes de automóviles comenzaron a investigar y añadir partes electrónicas que, en un principio y debido a su alto coste, solo se incluían en automóviles de alta gama.

Uno de los principales sistemas implementados en los coches fue el ABS (Sistema antibloqueo de ruedas). Introducido en 1978 por Bosch, este sistema electrónico fue un gran avance en la seguridad, puesto que impedía el bloqueo de las ruedas y con ello disminuía la distancia de frenado y permitía girar el coche durante todo ese tiempo.

Asimismo, el ABS fue la base para crear posteriormente otros sistemas electrónicos como el control de tracción, también por Bosch en 1986, el cual previene el deslizamiento de los neumáticos por pérdida de adherencia al asfalto.

Este proyecto se va a centrar en uno de esos sistemas implementados ya en el siglo XXI: el sistema de reconocimiento de señales de tráfico. Este sistema fue introducido en el mercado por primera vez por BMW en el modelo serie 7 a finales de 2008. En la actualidad, un gran número de vehículos que salen a la venta disponen de este sistema que ayuda al conductor tanto en comodidad como en prevención por posibles señales de tráfico que podrían haber pasado desapercibidas, disminuyendo así el riesgo de posibles accidentes de tráfico.

2.- Objetivos:

El objetivo de este proyecto es el de diseñar una aplicación capaz de detectar las diferentes señales de tráfico añadidas consiguiendo identificarlas al momento.

Para realizar este proyecto me introduciré en el modo de uso de la visión artificial, muy importante en la tecnología actual, concretándolo en un sistema que además permite ayudar en la seguridad de las personas.

Asimismo, con este trabajo me inicio en el lenguaje Python, un lenguaje que es cada vez más usado en el mundo hasta haber superado en la actualidad a muchos otros lenguajes conocidos internacionalmente. Entender Python y poder darle una utilidad práctica me sirve de referencia para posibles futuros proyectos en los que sea necesario este lenguaje.

Además, el hecho de relacionar mi proyecto con el mundo del automovilismo me ayuda a investigar y aprender sobre un sector muy interesante para mí en lo personal, algo que me puede resultar muy útil en caso de realizar en un futuro cualquier trabajo de este sector.

3.- Electrónica en los automóviles:

En la actualidad, los accidentes de tráfico son la tercera causa de muerte no natural en hombres y la cuarta causa en mujeres. Además, constituye la primera causa de muerte en el grupo de 15 a 24 años y la segunda causa de muerte en el grupo de 25 a 34 años. De hecho, hasta 2007 era la principal causa externa.

Estos preocupantes datos nos hacen ver la necesidad de aumentar la seguridad en los vehículos ya que no solo es suficiente con las normas viales, sino que reducir las posibilidades de lesión tras un accidente es un factor clave para bajar drásticamente el número de fallecidos por esta razón.

Esta seguridad en el automovilismo es uno de los principales aspectos en los que los fabricantes han estado investigando y desarrollando en las últimas décadas. En la siguiente tabla podemos ver los accidentes mortales y no mortales del año 2018:

	Total				
COMUNIDAD AUTONOMA	ACCIDENTES CON VICTIMAS	ACCIDENTES MORTALES	FALLECIDOS	HERIDOS HOSPITALIZADOS	HERIDOS NO HOSPITALIZADOS
Andalucía	16.811	255	274	1.284	22.916
Aragón	2.150	78	85	330	2.585
Asturias, Principado de	2.068	35	43	204	2.669
Balears, Illes	3.450	51	53	310	4.332
Canarias	3.439	67	68	401	4.559
Cantabria	628	19	23	62	909
Castilla-La Mancha	2.494	92	100	339	3.310
Castilla y León	4.264	161	176	603	5.266
Cataluña	26.907	295	326	1.694	33.732
Extremadura	1.455	49	51	187	1.881
Galicia	4.380	139	144	622	5.645
Madrid, Comunidad de	16.614	110	114	1.185	20.100
Murcia, Región de	2.518	62	66	285	3.171
Navarra, Comunidad Foral de	825	30	35	119	948
Rioja, La	602	10	10	68	751
Comunitat Valenciana	8.186	174	183	783	10.177
País Vasco	4.752	46	49	428	5.723
Ceuta y Melilla	756	6	6	31	1.000
Total	102.299	1.679	1.806	8.935	129.674

Tabla 3.1: Datos de accidentes en 2018 por comunidades autónomas

Si comparamos los datos de ese año 2018 con los de 20 años antes, en 1998, en el que hubo 97.570 accidentes con víctimas, 4.319 accidentes mortales, 5.957 fallecidos y 141.377 heridos, podemos ver que aún con prácticamente 4.000 accidentes con víctimas más en total, el número de fallecidos y de heridos

ha disminuido drásticamente. Esto se debe en principal medida al aumento de seguridad en los vehículos ya comentado.

Esta disminución en los fallecidos se puede observar más detalladamente y por años en el siguiente gráfico:

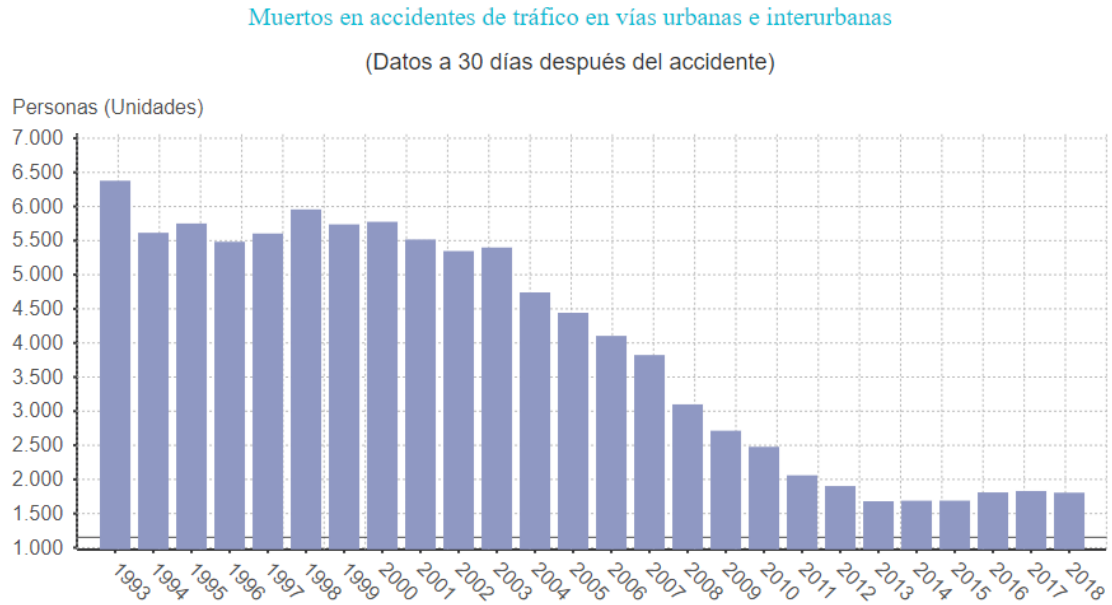

Fig. 3.1: Número de fallecidos en accidentes de tráfico por año. Fuente: DGT, www.epdata.es

Estos avances en seguridad se pueden contrastar también con los datos que aporta la Unión Europea al respecto:

geo \ time	2000	2001	2002	2003	2004	2005	2006
EU (28 countries)	56,990	54,880	53,924	50,989	47,834	45,888	43,663
geo \ time	2007	2008	2009	2010	2011	2012	2013
EU (28 countries)	43,097	39,525	35,315	31,481	30,668	28,231	25,983

geo \ time	2014	2015	2016	2017	2018
EU (28 countries)	25,987	26,162	25,672[d]	25,250[d]	25,178[d]

Tabla 3.2: Fallecidos por año en la Unión Europea

Además podemos observar como países como España tienen números relativamente bajos, al contrario que otros países como pueden ser Rumanía o Bulgaria con unos números de muertes en accidentes de tráfico por habitante mucho más preocupantes:

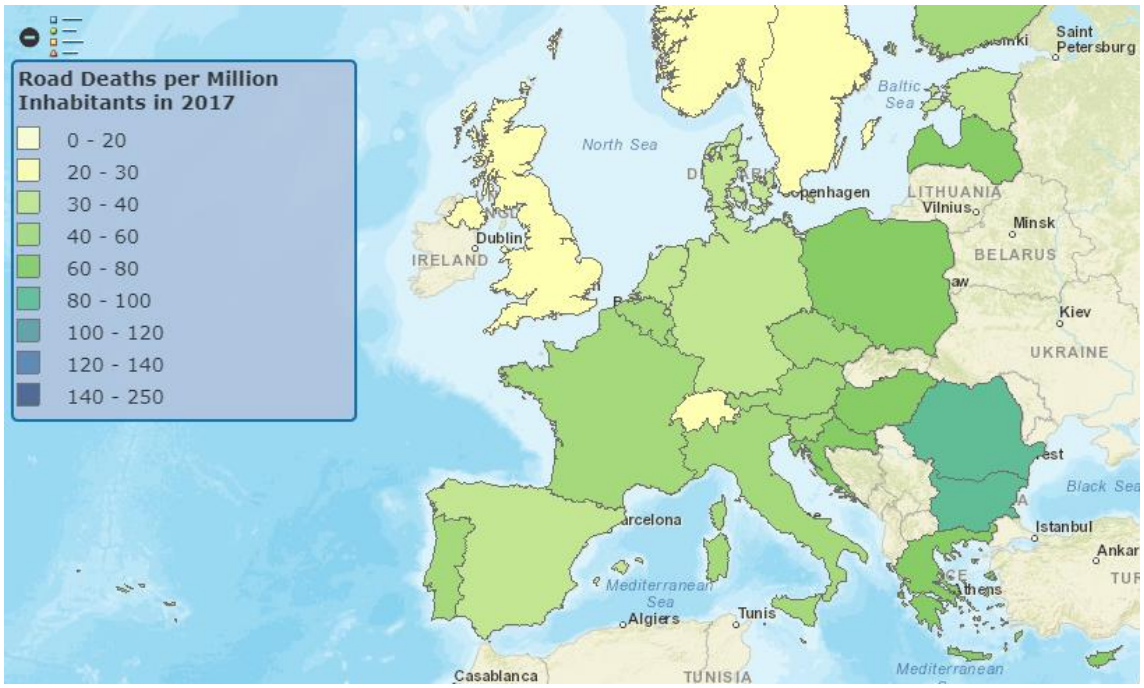

Fig. 3.2: Mapa de fallecidos en carretera en la UE. Fuente: Unión Europea

Con todo esto, la evolución del teléfono móvil y su consecuente aumento de uso mientras se conduce es una razón más para la preocupación de los fabricantes de automóviles en busca de mayor seguridad y más sistemas que ayuden al conductor en caso de que este se despiste. El incremento del uso de redes sociales y de la cámara en los móviles ha contribuido a un mayor número de accidentes.

En el siguiente gráfico podemos observar el elevado número de conductores que admiten utilizar el teléfono móvil al volante y sus principales usos.

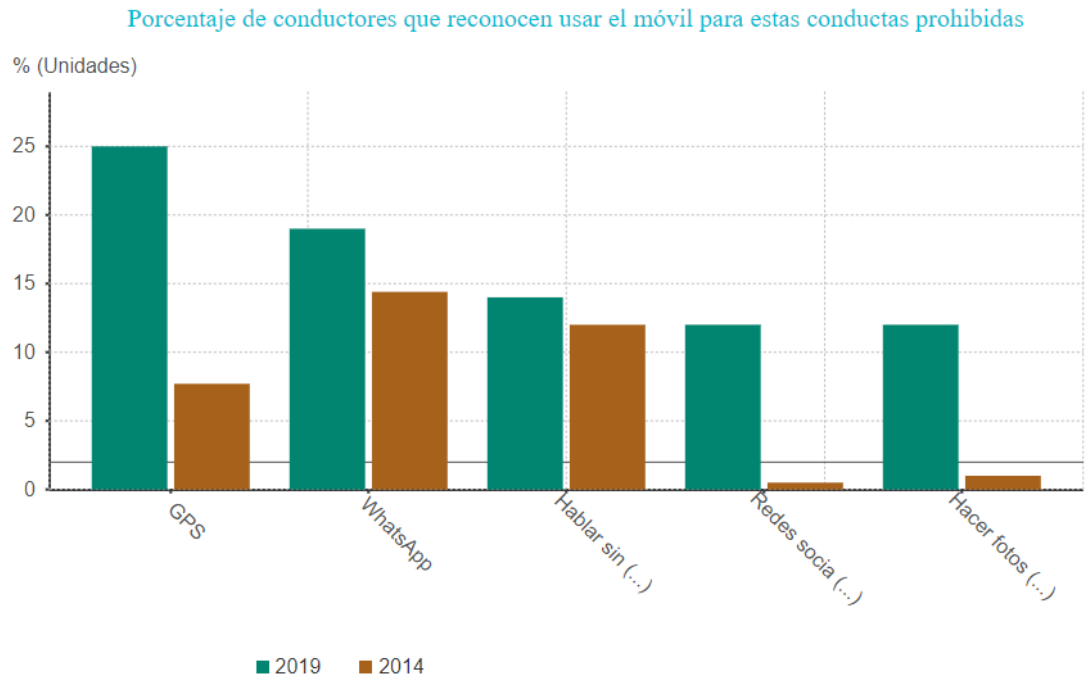

Fig. 3.3: Porcentaje de conductores que reconocen realizar estas actividades con el móvil. Fuente: BP, Castrol, Race, www.epdata.es

Por todo esto, en la actualidad todo coche fabricado contiene una gran parte de electrónica para mejorar tanto el confort del conductor y los pasajeros como la seguridad al volante previniendo estos accidentes o minimizando el daño causado por estos.

Desde el acceso al coche sin llave hasta la climatización automática, todo ha contribuido para finalmente terminar fabricando vehículos completamente autónomos. Estos son algunos de los sistemas electrónicos que se pueden encontrar en un coche en la actualidad:

3.1.- Sistema antibloqueo de ruedas:

Cuando un neumático deja de rotar en la misma dirección que el vehículo se produce un deslizamiento sobre la superficie. Este deslizamiento puede ocurrir bien por un uso excesivo de la fuerza de frenado o bien porque el vehículo esté intentando moverse lateralmente en un giro debido a la inercia.

El sistema antibloqueo de ruedas (ABS) impide que la rueda bloquee usando un sistema de sensores y válvulas que reducen la presión de frenado manteniendo la rueda en el momento donde está a punto de dejar de rotar. Esto

hace que el neumático no deslice cuando el vehículo va hacia adelante o hacia atrás, pero no funciona cuando se desliza lateralmente.

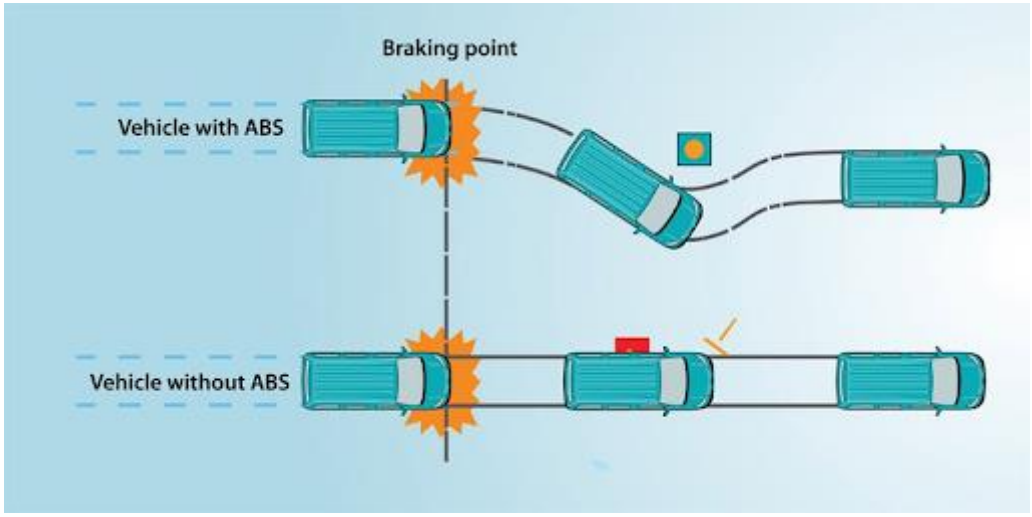

Fig. 3.4: Trayectoria de un coche con y sin ABS

En vehículos sin ABS el conductor ha de intentar ajustar el pedal de freno al punto en el que no se bloqueen las ruedas, pudiendo así frenar de menos y aumentar la distancia de frenado. Gracias a este sistema, la distancia de frenado disminuye considerablemente debido a que el mayor efecto de frenado ocurre en el momento en el que la rueda está casi bloqueada. Asimismo, en caso de emergencia el conductor solo se tiene que preocupar por aplicar la máxima presión en el pedal del freno, dejando que la electrónica se encargue del resto.

El sistema antibloqueo de ruedas funciona con un sensor en cada rueda conectada al módulo de control del sistema ABS, el cual conoce los parámetros normales de frenado, como la rapidez de deceleración según la velocidad momentánea. Si en un momento dado se frena demasiado rápido, el módulo de control activa una serie de válvulas hidráulicas que reducen inmediatamente la presión ejercida en los discos de frenos permitiendo que el neumático continúe girando.

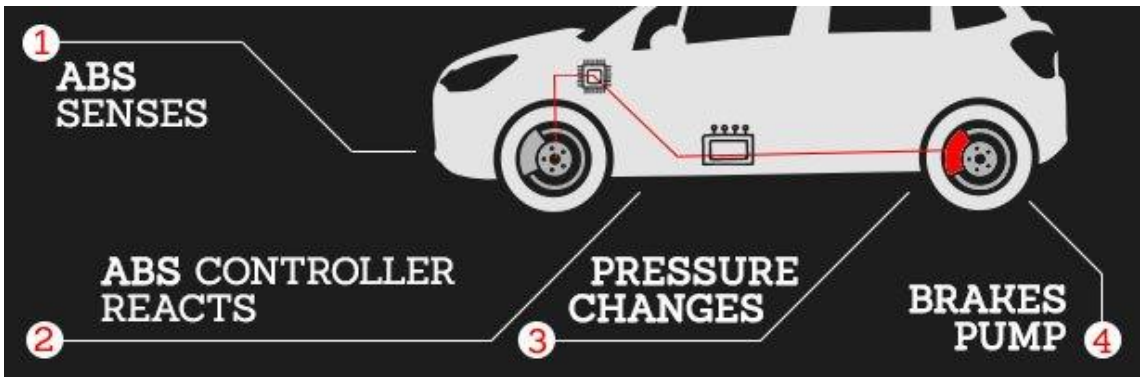

Fig. 3.5: Ubicación de las partes del ABS

3.2.- Motor de arranque:

Hasta 1911 los vehículos se arrancaban girando una palanca que hacía rotar el cigüeñal. Fue en ese año cuando General Motors patentó un sistema de encendido del motor automatizado para el Cadillac Touring Edition.

Hoy en día, el motor de arranque utiliza la energía eléctrica de la batería para hacer girar el cigüeñal, iniciando así la combustión dentro del motor. Tras la primera explosión el propulsor ya es capaz de trabajar por sí solo, dejando de utilizarse el motor de arranque.

El motor de arranque está formado por siete piezas principales que convierten la energía eléctrica de la batería en energía cinética. Esta corriente eléctrica necesaria para producir torsión es inducida por inducción electromagnética del campo magnético.

Al girar la llave de contacto del vehículo, la corriente de la batería pasa a un solenoide, lo que propicia un efecto de palanca sobre el piñón de arrastre del motor de arranque que permite su acoplamiento al engranaje de la corona del volante motor para iniciar el movimiento.

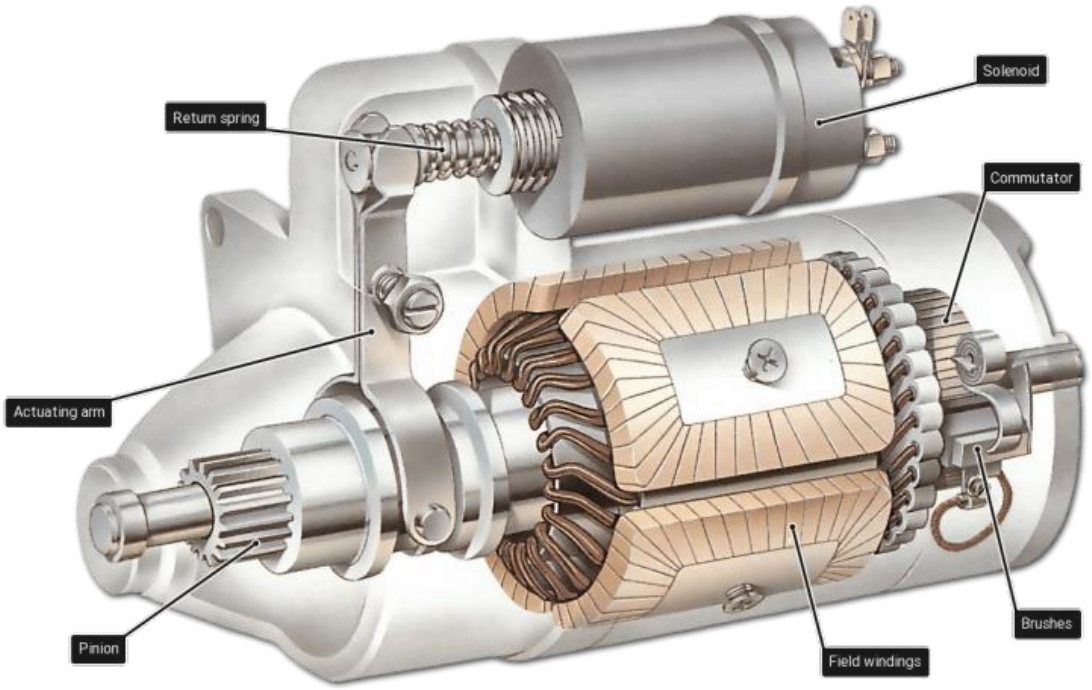

Fig. 3.6: Partes del motor de arranque

3.3.- Dirección asistida:

Fig. 3.7: Dirección asistida

La dirección asistida es el sistema que se encarga de reducir la fuerza necesaria para hacer girar el volante de un automóvil.

Han existido tres formas distintas de dirección asistida hasta el momento:

Dirección asistida hidráulica: Fue la más utilizada durante la década de los 90 y principios del siglo XX. Una bomba de dirección hidráulica conectada al motor permite enviar el fluido necesario a la cremallera de dirección para conseguir que el conductor pueda girar el volante con mucha menos fuerza.

Dirección asistida electro-hidráulica: Funciona igual que la dirección asistida hidráulica pero, en vez de ser el motor el encargado de accionar la bomba de dirección hidráulica, se realiza a través de un motor eléctrico alimentado por la batería. Se activa solo cuando se mueve el volante y reduce el consumo de combustible con respecto a la hidráulica.

Dirección asistida eléctrica (EPS, Electrical Powered Steering): Es la última versión de dirección asistida y funciona mediante un motor eléctrico colocado directamente en la columna de dirección. Al usar únicamente el motor eléctrico, este sistema es, frente a los dos anteriores, más ligero y simple.

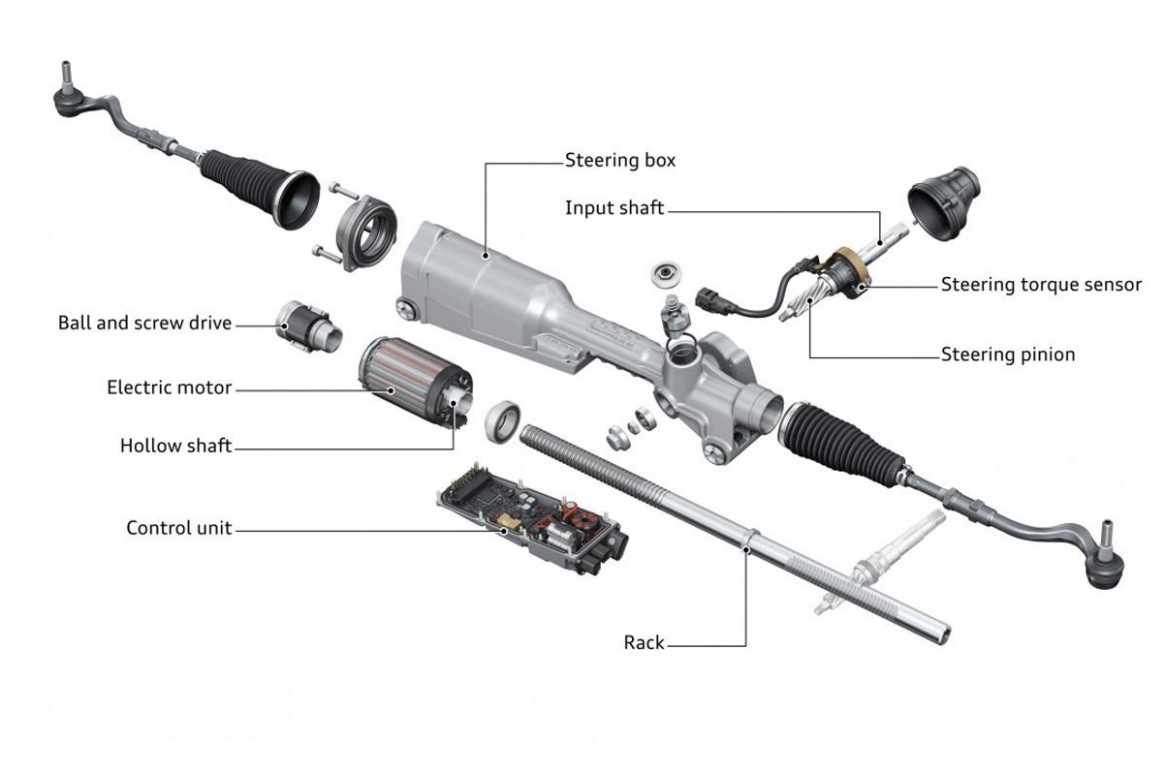

Fig. 3.8: Partes de la dirección asistida

3.4.- Control de acceso sin llave (Keyless):

Introducido por primera vez por Mercedes, este sistema está diseñado para poder abrir y cerrar el coche, y arrancar su motor, sin necesidad de abrirlo manualmente. Con ello permite tener las llaves o el mando a distancia en los bolsillos o en bolsos para que sea suficiente para poder acceder al coche. Al estar próximos al coche, el sistema desbloquea las puertas y desconecta el anti-arranque del motor. Asimismo, cuando el portador de las llaves se aleja del vehículo se bloquean las puertas y se vuelve a conectar el anti-arranque del motor.

Para el funcionamiento de este sistema es necesario la introducción del transmisor de identificación de la llave de contacto y de receptores en el coche. Cuando el conductor tira de la manilla de la puerta se produce un intercambio de códigos encriptados por señal que permite la apertura del vehículo.

3.5.- Regulador de velocidad activo:

Más conocido como control de crucero adaptativo (ACC, Adaptative Cruise Control) permite mantener una velocidad constante y adaptarla al mismo tiempo al tráfico real.

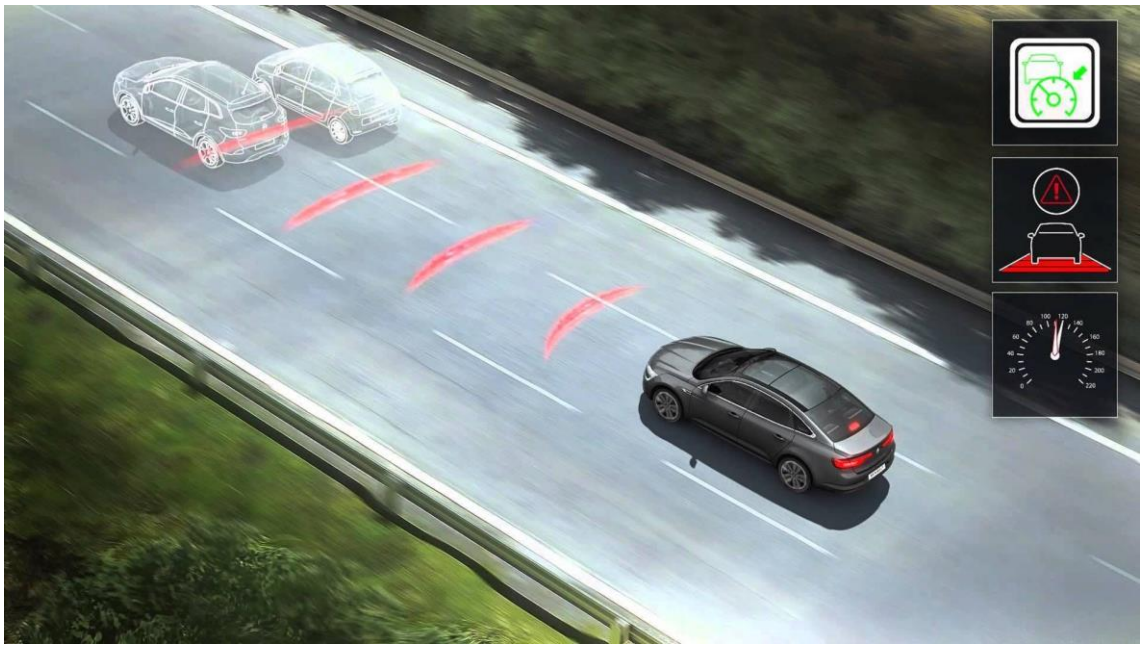

Fig. 3.9: Radar del regulador de velocidad activo

Es una evolución del control de velocidad, cuyo sistema moderno fue inventado por el ingeniero mecánico estadounidense Ralph Teetor, quien se inspiró para su invento yendo en el coche con su abogado, quien reducía la velocidad cuando hablaba y aceleraba cuando escuchaba. Esto molestaba tanto a Ralph Teetor que se centró en inventar un control de velocidad hasta conseguirlo en 1945. El primer vehículo que incluyó este sistema fue el Chrysler Imperial de 1958.

El vehículo emite señales de radar que detectan otros coches u obstáculos que puedan encontrarse en la carretera. Según la detección que realice el sistema aumenta o reduce la velocidad automáticamente. Así, en el caso que deba reducir la velocidad por cualquier tipo de obstáculo, la velocidad volverá al punto de partida indicado por el conductor más tarde.

3.6.- Climatización automática:

El aire acondicionado para coches apareció por primera vez en la década de 1930 en coches fabricados por Packard, aunque era un sistema que se insertaba en el vehículo una vez ya estaba fabricado y ocupaba espacio del maletero. En la década de los 50 fue cuando Chrysler introdujo por primera vez los aires acondicionados de una forma más eficaz.

Aunque en un inicio era en Estados Unidos donde más se fabricaban coches con aire acondicionado, fueron los fabricantes europeos los que evolucionaron el sistema hasta llegar al sistema de control climático o climatización automática de la actualidad.

Fig. 3.10: Controles de la climatización de un coche

Con la climatización automática el conductor solo debe indicar la temperatura a la que quiere el interior del vehículo, y el sistema se encargará de calentar o enfriar el aire y así conseguir llegar a dicha temperatura.

El vehículo utiliza el calor del motor, al igual que un elemento calefactor en el compartimento del motor, para calentar el aire que pasa en caso de necesitar aumentar la temperatura. En caso de querer enfriar el aire, el vehículo está equipado con un compresor adicional bajo el capó que dirige el aire a un condensador, el cual enfría el aire con un refrigerante.

3.7.- Regulación automática del alcance luminoso:

Actualmente con frecuencia nos encontramos coches que utilizan circuitos electrónicos de control en el sistema de iluminación, lo que automatiza el uso de las luces en los vehículos, no solo en carretera, sino también si el conductor se las dejara encendidas al abandonar el coche.

Además de esa función, también nos encontramos con una regulación automática del alcance luminoso, cambiando el ángulo de inclinación de los faros.

Dentro de la regulación automática del alcance luminoso podemos encontrar dos sistemas distintos:

Regulación casi estática del alcance luminoso: Este sistema corrige solamente los cambios en la inclinación debido a los cambios en la carga. Para ello, una unidad de control evalúa los datos del sensor del eje delantero y del sensor del eje trasero, los compara con los datos nominales y acciona consecuentemente los servomotores de los faros.

Regulación dinámica del alcance luminoso: En los vehículos con faros xenón, hoy en día prácticamente solo existen sistemas de regulación dinámica del alcance luminoso, que reaccionan ante los cambios de inclinación en el coche, ya sea por la situación de la carretera o por aceleraciones o frenadas.

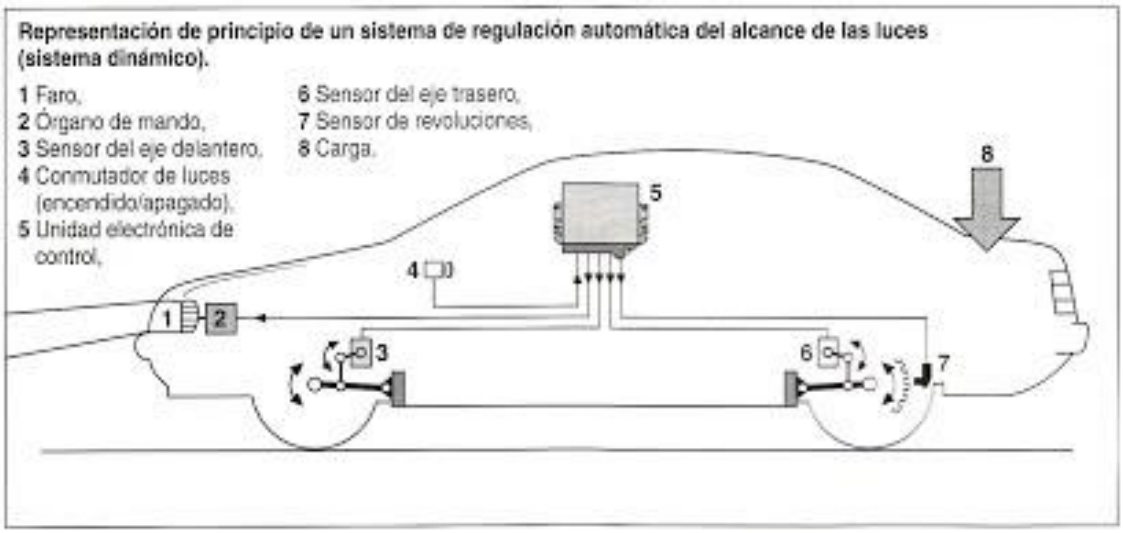

Fig. 3.11: Partes de un sistema de regulación automática del alcance luminoso

3.8.- Iluminación inteligente:

Otro sistema electrónico que podemos encontrar en los vehículos es el sistema de iluminación frontal autoadaptable (AFS, Adaptative Frontlightning System), desarrollado por la compañía Hella. Este sistema se encarga de adaptar la iluminación tanto en anchura como en profundidad, variando el área iluminada en función de la velocidad a la que se conduce.

Cuando se circula por vía urbana y a menos de 50 km/h, la luz que emiten los faros abarca un área de escasa profundidad pero de gran anchura, permitiendo la visibilidad a ambos lados de la carretera, mejorando la visibilidad en cruces e intersecciones y para señales de tráfico. Por el contrario, cuando se circula por carretera con una velocidad mayor a la urbana, la luz emitida alcanza una mayor distancia con una menor anchura, permitiendo localizar posibles obstáculos a una distancia mayor y proporcionando más tiempo para reaccionar a ellos.

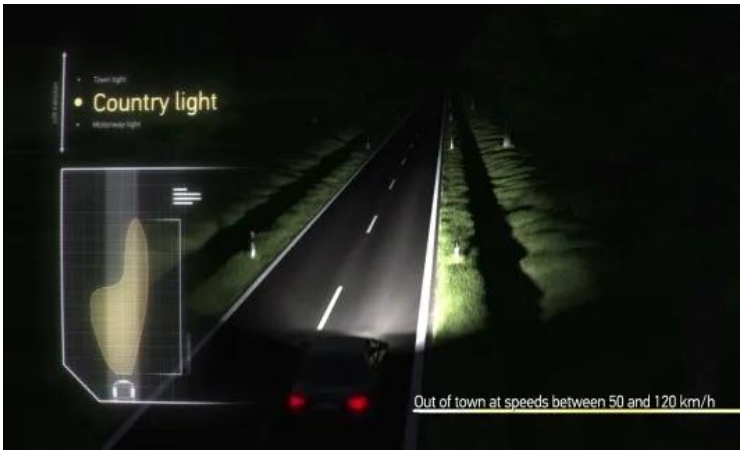

Fig. 3.12: Iluminación inteligente en carretera a más de 50 km/h

Fig. 3.13: Iluminación inteligente en vía urbana a menos de 50 km/h

3.9.- Control de tracción:

El control de tracción es la parte que se encarga de impedir que las ruedas patinen al acelerar. Su uso en los coches comenzó gracias a la implementación del sistema antibloqueo de ruedas, utilizando sus sensores de los frenos antibloqueo para relacionarlos con la gestión del motor. Cuando este sistema detecta que una de las ruedas gira más rápido que las demás puede aplicar el freno en esa rueda y variar la potencia del motor.

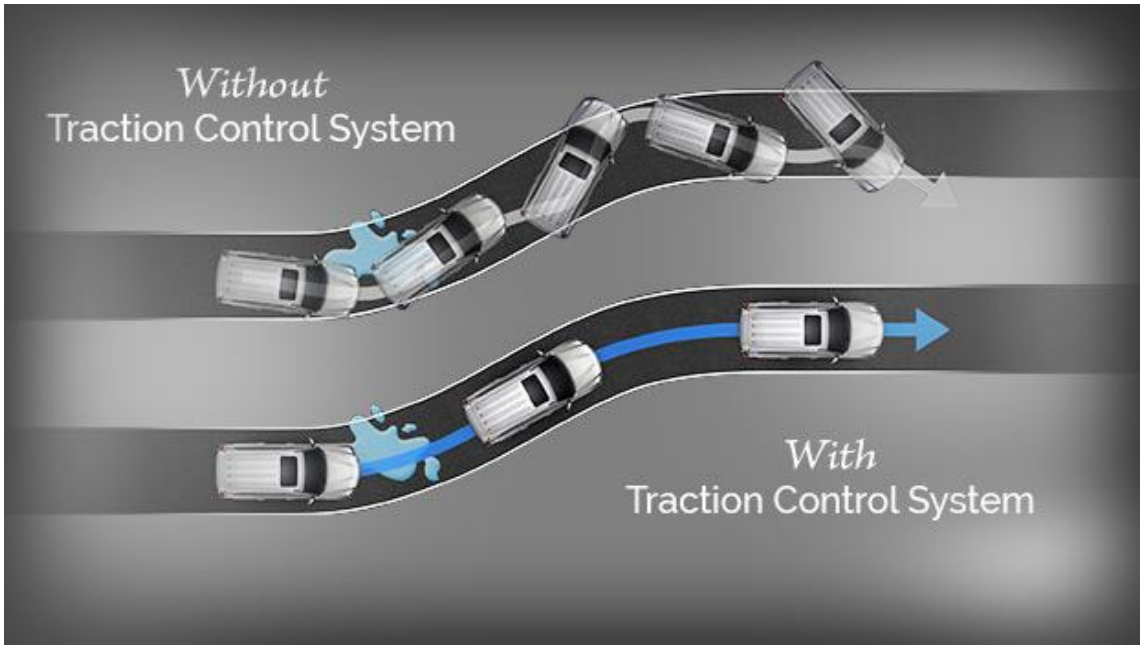

Fig. 3.14: Trayectoria de un coche con y sin control de tracción

Con anterioridad a su uso en los vehículos, el control de tracción ya estaba implementado en ferrocarriles. Siendo de acero tanto la vía como la rueda, la adherencia entre estas no era muy elevada. En los primeros sistemas, una rápida aceleración de las ruedas de tracción hacía saltar una alarma al maquinista para que accionase el arenero, que consistía en un pequeño depósito con una trampilla que dejaba caer arena delante de las ruedas de tracción para aumentar su adherencia.

En la actualidad todo coche fabricado dispone del sistema de control de tracción ya integrado. Existen dos tipos de control de tracción: los que reducen la potencia del motor y los que, además de ello, hacen uso del freno sobre la rueda que está perdiendo adherencia.

Para su funcionamiento se requieren los sensores que emplea el ABS y un sensor de posición del pedal del acelerador. Los sensores del ABS en las ruedas detectan si una rueda está girando más rápidamente que las otras, y el sensor del

acelerador indicará si esto se debe a que se está aplicando demasiada presión en dicho pedal.

Que una rueda comience a patinar trazando una curva (pueden también patinar porque una rueda se encuentre en una zona de baja adherencia como pintura o aceite) se debe a que el neumático exterior aumenta su adherencia (al contrario que el interior) por una transferencia del peso del coche hacia el exterior. Cuando esto ocurre, el control de tracción aplica en primera instancia el freno en la rueda que deslice. En caso de no lograr corregir la situación, es cuando se aplica una disminución de la potencia aplicada por el motor.

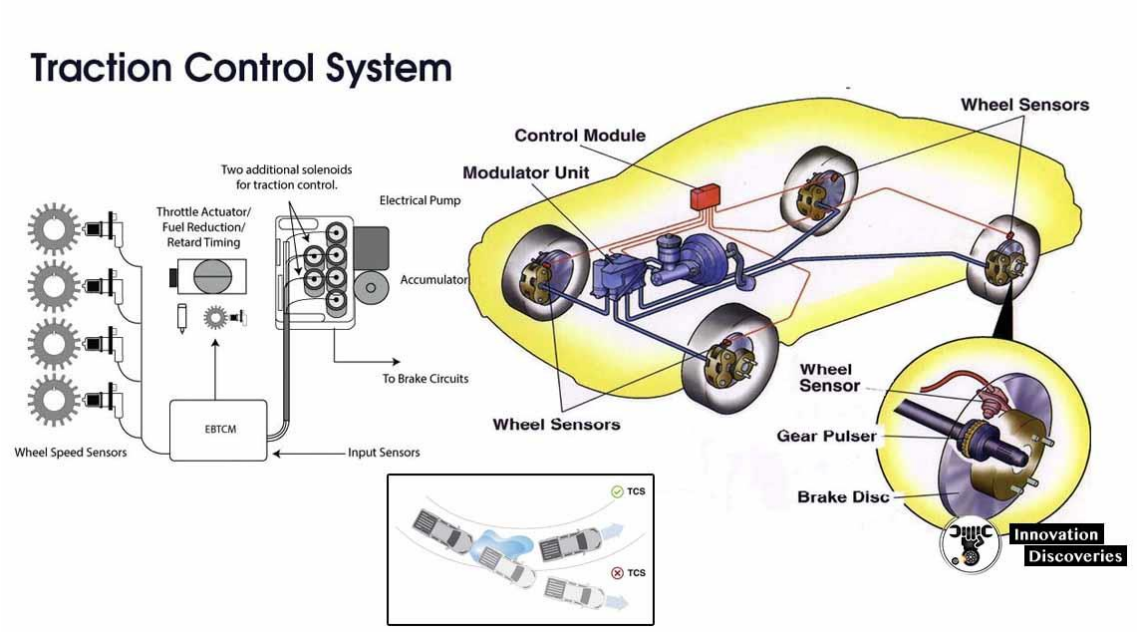

Fig. 3.15: Partes de un sistema de control de tracción

3.10.- El vehículo autónomo:

El vehículo autónomo es aquel que no necesita de conductor, puesto que el conjunto de sistemas electrónicos e informáticos que lo integran son capaces de imitar las capacidades del conductor y así conducir de manera automática y sin ayudas. Todo sistema electrónico en los coches que ayude a automatizar cualquier aspecto es un avance para terminar llegando a este sistema completamente autónomo. Detección en todo momento de cualquier obstáculo y de otros vehículos, control de la velocidad del resto de vehículos, detección de las señales de tráfico, detección de carriles, aparcamiento automático y un largo etcétera que han de disponer estos vehículos.

El coche autónomo es un objetivo real que se tiene desde primera mitad del siglo XX, y en el que muchas empresas y fabricantes no cesan en su intención

de desarrollar. Ya en 1939, el diseñador industrial Norman Bel Geddes presentó en la Exposición Universal el primer vehículo autónomo que se conoce: un vehículo eléctrico impulsado vía campos electromagnéticos proporcionados por un circuito eléctrico embebido en la carretera.

Sería en los años 50 cuando tuvieron lugar los verdaderos ensayos prometedores y en la década de 1980 aparecieron los primeros coches realmente autosuficientes y autónomos.

En los últimos años un gran número de empresas del sector automovilístico y de organizaciones han desarrollado vehículos autónomos. Tanto es así que, desde 2019, 29 estados de Estados Unidos han aprobado leyes que permiten los vehículos autónomos, y en ciertos países y ciudades de Europa también se han llegado a probar estos automóviles en tráfico.

Desde 2016 Uber, empresa estadounidense que proporciona vehículos para el transporte de pasajeros, ha operado con coches autónomos en las ciudades estadounidenses de Pittsburgh y de San Francisco. Sin embargo, un atropello mortal en marzo de 2018 producido por uno de estos coches provocó la suspensión temporal de pruebas de Uber con coches autónomos en estas y otras ciudades.

Fig. 3.16: Coche autónomo de Uber operando en San Francisco

A pesar de los grandes avances electrónicos para la consecución del vehículo autónomo definitivo, hay un problema que preocupa aún a muchas

organizaciones: el marco ético. Este problema reside en la toma de decisiones automáticas de un coche autónomo en relación a un accidente inminente. Por ejemplo, la elección de un peatón como víctima respecto a otro en caso de que uno de los atropellos sea inevitable, o la decisión de si evitar un accidente que puede ser mortal para los pasajeros a cambio de tener que atropellar a un peatón.

Una de las marcas de coches más reconocidas mundialmente por sus características automáticas es Tesla.

Fig. 3.17: Tesla Model S

Los coches Tesla, con su denominado 'Autopilot', son capaces de realizar de forma autónoma muchas de las características antes mencionadas. Introducido en el Model S de 2014, todos sus vehículos comenzaron a ser producidos con esta función, que permite la conducción automática de forma limitada sin necesidad de la actuación del conductor.

Tal y como explica la web oficial de Tesla, para su función Autopilot el coche está equipado con lo siguiente:

Ocho cámaras en el vehículo ofrecen una visión 360 grados alrededor del vehículo con un alcance de hasta 250 metros. Doce sensores ultrasónicos fueron actualizados y complementan esta visión, lo que permite la detección de objetos sólidos y blandos a cerca del doble de distancia y precisión del sistema anterior. Un radar delantero con procesamiento mejorado brinda datos adicionales sobre

el mundo con una longitud de onda redundante que le permite ver a través de lluvia intensa, neblina, polvo e incluso el vehículo que antecede.

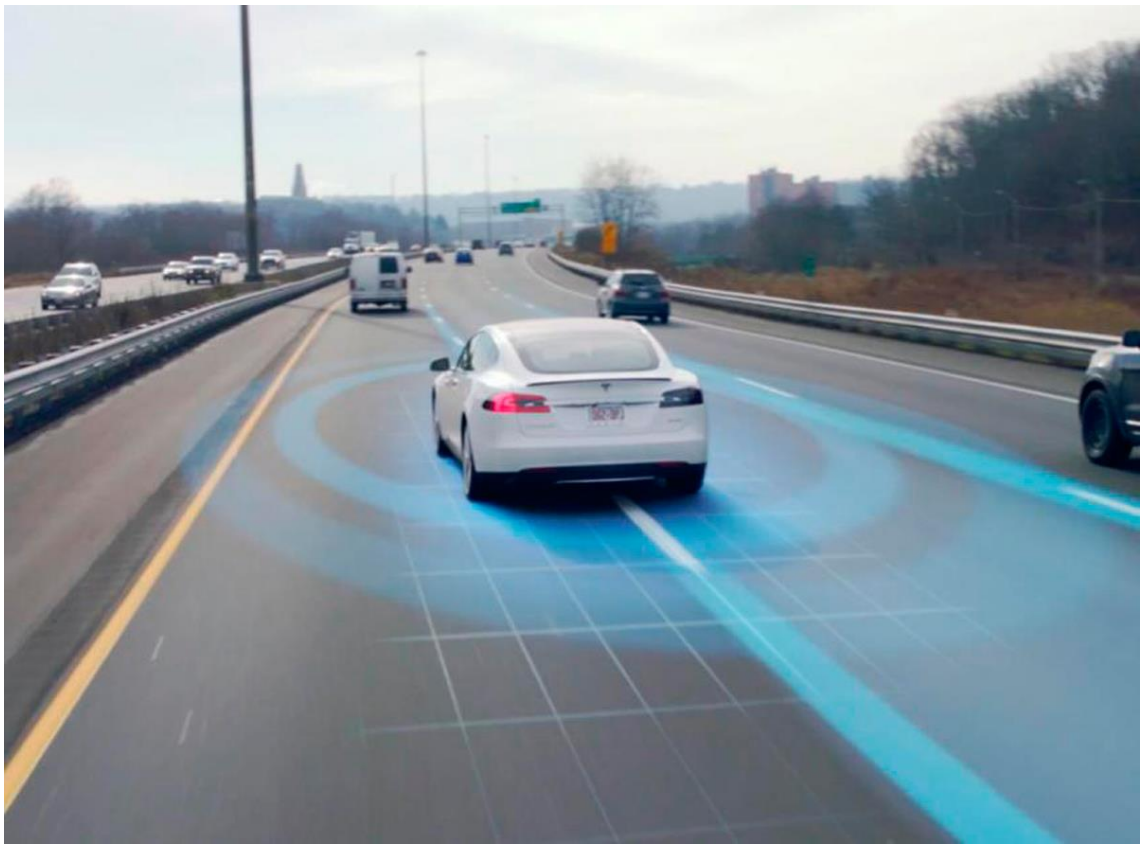

Fig. 3.18: Las cámaras del Tesla proporcionan una visión 360º

Con este sistema es posible, además de establecer una velocidad de crucero adaptativo, fijar una distancia de seguridad con el coche que va delante. Permite el estacionamiento automático y está capacitado para leer GPS y estar conectado a Internet.

Fig. 3.19: Interior y funcionamiento del Autopilot

Sin embargo, cabe recordar que el Autopilot de los coches Tesla es un asistente de ayuda a la conducción, no un sistema de coche autónomo, tal y como se encarga de recordar el propio piloto automático avisando al conductor de la necesidad de mantener las manos en el volante y no desviar su atención de la conducción.

4.- Elementos utilizados en el proyecto:

Para realizar este proyecto se necesitará hacer uso de la visión artificial o visión por computadora. La visión artificial incluye todas las aplicaciones en las que una combinación de software y hardware ejecutan sus funciones para la captura y procesamiento de imágenes.

El origen de la visión artificial industrial se podría atribuir a los años 60 y al origen de un prototipo automatizado basado en cámaras de visión y sistemas de procesamiento de las imágenes captadas. Durante esta década, en las universidades se veía esta disciplina como un escalón más para la mejora de la inteligencia artificial. En muchas de estas universidades y otras empresas se pensaba que se conseguiría una máquina tan inteligente como un humano en el tramo de una generación, lo que hizo que se invirtieran, principalmente en Estados Unidos, millones de dólares en investigación pública y privada. Sin embargo, su punto de inflexión llegó en los años 80 cuando la ingeniería informática y la creación de procesadores más rápidos comenzaron a tener una gran importancia en el mundo.

La visión artificial ha pasado a ser una de las partes más importantes de la electrónica y utilizada en una inmensa variedad de aplicaciones distintas dentro de todo tipo de sectores. En la actualidad se pueden realizar proyectos de este tipo con muy poco presupuesto y de una forma más o menos sencilla.

4.1.- Lenguaje de programación:

El lenguaje de programación elegido para el proyecto es Python. Python es un lenguaje de programación interpretado de código abierto, disponible en diferentes sistemas operativos, siendo un lenguaje sencillo y fácil de entender. Además, el hecho de que sea un lenguaje muy utilizado en todo el mundo nos permitirá encontrar información y recursos de manera más rápida y eficiente.

Fig. 4.1: Logo Python

Python apareció en los años 90 de la mano del ingeniero holandés Guido Van Rossum, quien trabajaba en el Centro de Investigación de Ciencias de la Computación en Amsterdam. Python nación como un proyecto de software libre y de código abierto, algo a lo que se debe parte del éxito conseguido posteriormente.

Python es gestionada actualmente por la Python Software Foundation, con Guido Van Rossum involucrado en el desarrollo y la toma de decisiones de su diseño.

Aunque este lenguaje se puede utilizar para una gran variedad de opciones, el factor que ha extendido en gran medida el uso de Python es su empleo en el Big Data (macrodatos o datos masivos), el Machine Learning (aprendizaje automático), el Deep Learning (aprendizaje profundo) y la Inteligencia Artificial, siendo muchas de las nuevas herramientas creadas por los ingenieros de datos desarrolladas en Python.

El aumento considerable en el uso de este lenguaje de programación se puede observar en gráficos como el porcentaje de visitas que reciben páginas web importantes dependiendo del lenguaje del que se hable:

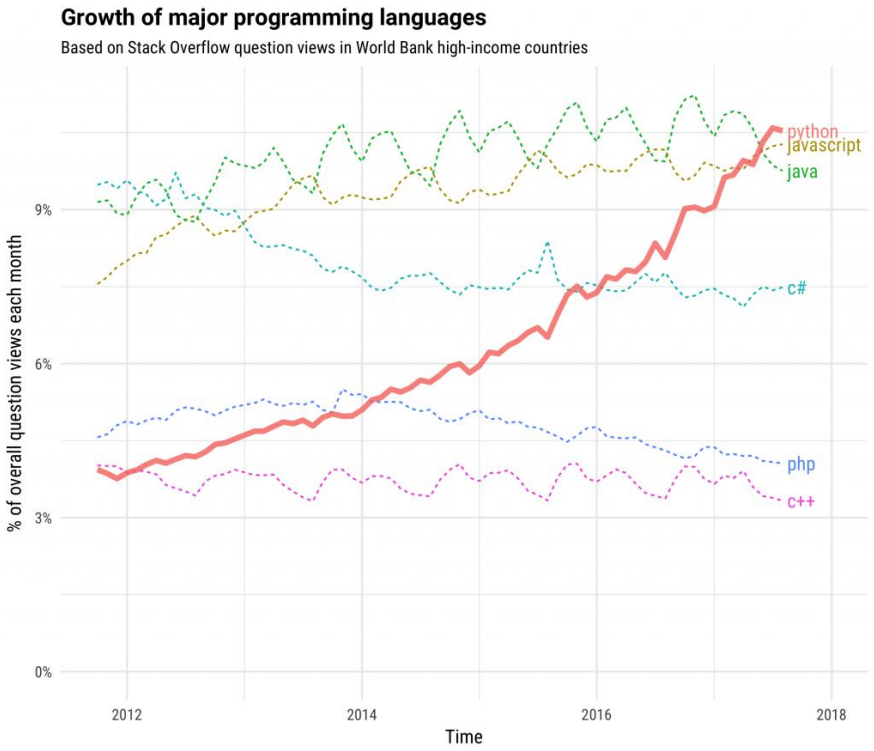

Fig. 4.2: Gráfico de la evolución de distintos lenguajes de programación

Asimismo, también se puede observar cómo está posicionado Python con respecto a otros lenguajes observando la demanda de trabajadores junto con sus respectivos salarios.

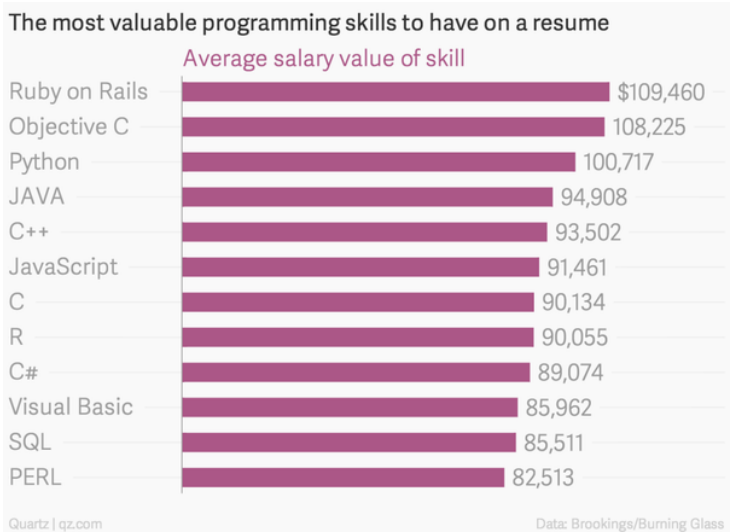

Fig. 4.3: Salario medio según el lenguaje de programación

4.2.- Hardware:

4.2.1.- Cámara:

Para este proyecto se puede utilizar cualquier cámara que se pueda conectar a un ordenador y con el cual se pueda retransmitir en vivo. En este caso, para abaratar costes y por cuestiones de movilidad se ha decidido realizar con la webcam que incorpora el ordenador, aunque añadir una cámara externa resultaría sencillo.

4.3.- Software:

4.3.1.- PyCharm:

El entorno de desarrollo integrado elegido para la realización de este proyecto es PyCharm.

Fig. 4.4: PyCharm

PyCharm es un software multiplataforma, pudiendo ser utilizado en Windows, macOS y Linux, y es utilizado especialmente en lenguaje Python. Desarrollado por la compañía checa JetBrains, lanzó su versión beta en julio de 2010, con su primera versión apareciendo 3 meses más tarde. PyCharm Community Edition, la versión de código libre de PyCharm, fue lanzado el 22 de octubre de 2013.

Este entorno de desarrollo escogido es uno de los más completos para Python, y también uno de los más populares. Una de las principales ventajas que ofrece PyCharm es la de refactorizar el código, es decir, simplificar el código sin cambiar su función para que pueda ser entendido con mayor facilidad. Además cuenta con un editor inteligente en el que se pueden utilizar una gran cantidad de atajos de teclado.

Posee un intérprete en el editor de código que permite conocer en tiempo real los posibles errores del código, lo que hace que sea elegido por muchos usuarios que comienzan a programar. A ello hay que añadir que admite otros lenguajes de programación como javascript, kotlin o CoffeeScript y otras herramientas como html o css.

PyCharm se puede descargar de forma gratuita desde la web oficial www.jetbrains.com/pycharm/ o directamente desde la zona de descargas de la web www.jetbrains.com/pycharm/download/. Su versión de pago ha sido utilizada por grandes compañías como Twitter, Ebay, Amazon o Facebook.

4.3.2.- Cascade Trainer GUI:

Cascade Trainer GUI es un programa que puede ser utilizado para entrenar, probar y mejorar clasificadores de cascada (explicado en el punto "6.- Desarrollo del proyecto").

Fig. 4.5: Logo Cascade Trainer GUI

Con este software seremos capaces de crear modelos de clasificadores con los que poder detectar cualquier objeto que necesitemos, entrenándolo con un conjunto de imágenes que deberemos proporcionar.

Cascade Trainer GUI posee una interfaz gráfica muy sencilla de entender y de utilizar, sin muchas opciones que hagan difícil el uso del programa para cualquier usuario.

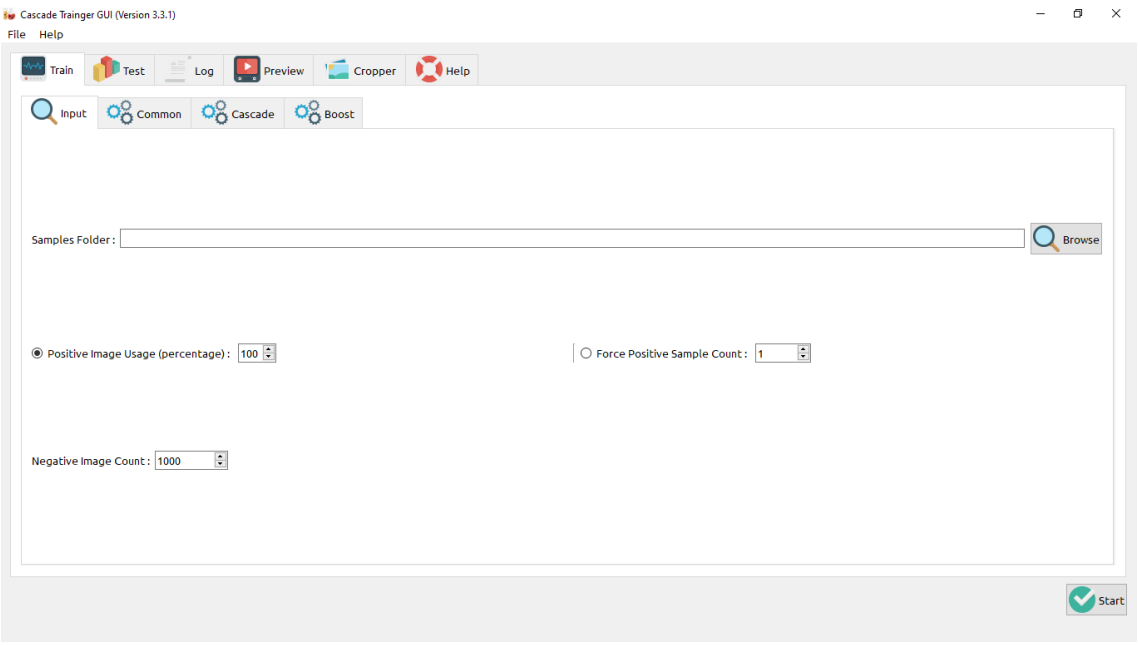

Fig. 4.6: Interfaz Cascade Trainer GUI

Para poder instalar este programa se necesita Windows (7 o superior) y se puede descargar desde la web oficial www.amin-ahmadi.com/cascade-trainer-gui/ de manera gratuita.

4.3.3.- Librerías:

Para instalar las librerías en PyCharm simplemente deberemos ir a File>Settings, desplegaremos la pestaña de nuestro proyecto y haremos click en Project Interpreter. La ventana resultante debería ser esta:

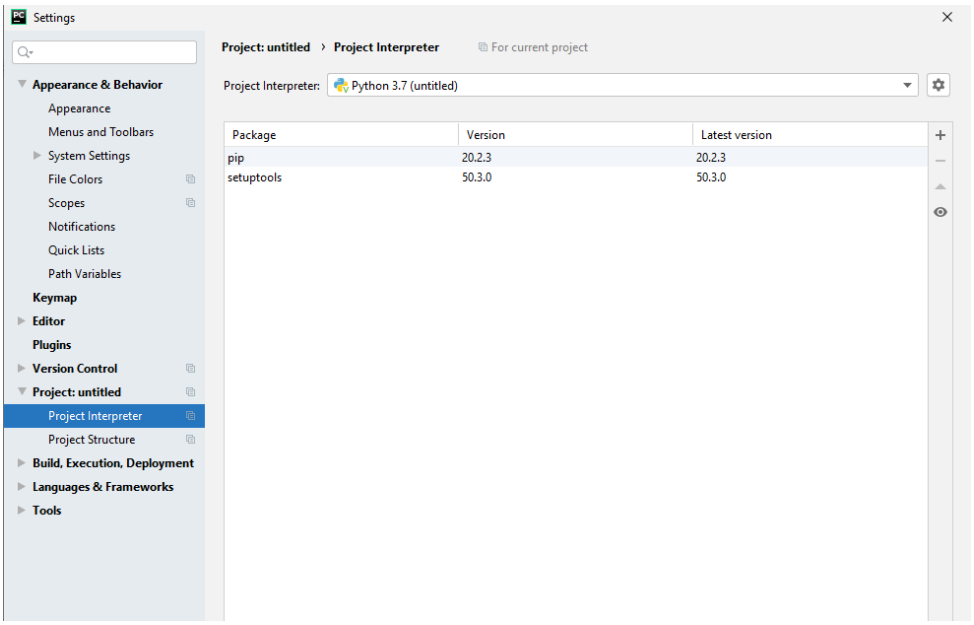

Fig. 4.7: Ventana Project Interpreter en PyCharm

Hacemos click en el símbolo “+” situado a la derecha del todo y entonces buscaremos las librerías que deseemos.

En caso de instalar versiones no compatibles entre ellas trataremos de instalar una versión anterior que sí se pueda utilizar.

4.3.3.1.- OpenCV:

OpenCV (Open Computer Vision) es una de las librerías más utilizadas alrededor del mundo. Fue originalmente desarrollada por Intel y su primera versión alfa apareció en 1999, siendo un año después cuando liberó el código al público. Es una librería multiplataforma que está disponible en Windows, Mac, Linux y Android, además de permitir programarse en C, C++, Python, Java y Matlab.

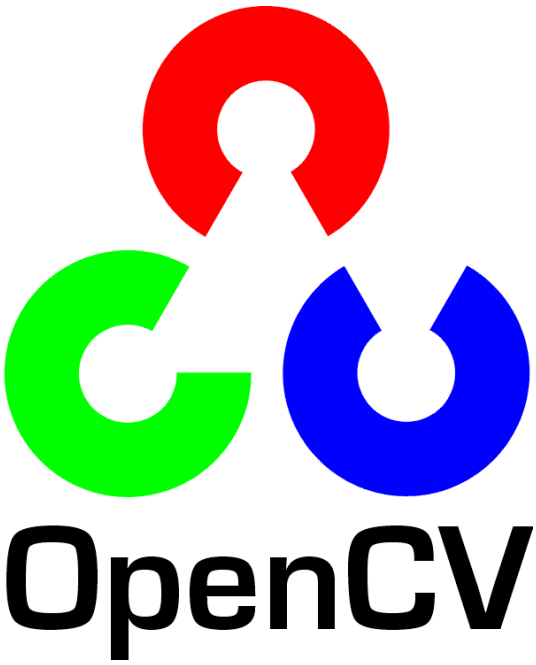

Fig. 4.8: Logo OpenCV

Esta librería está destinada principalmente a la visión artificial, al análisis de imágenes y al aprendizaje automático. El hecho de que sea una de las más famosas del mundo también hace que existan multitud de tutoriales en Internet explicando todo lo que se puede realizar con OpenCV.

La versión instalada en mi proyecto es la versión opencv-python 4.2.0.32.

4.3.3.2.- NumPy:

NumPy es una librería creada exclusivamente para el lenguaje Python. Esta librería agrega funciones matemáticas de alto nivel centradas en las matrices y en los vectores. La librería anterior a NumPy, Numeric, fue creada en 1995 por Jim Hugunin ayudado por varios desarrolladores. En 2005 fue cuando Travis

Oliphant creó NumPy añadiendo características y haciendo grandes modificaciones.

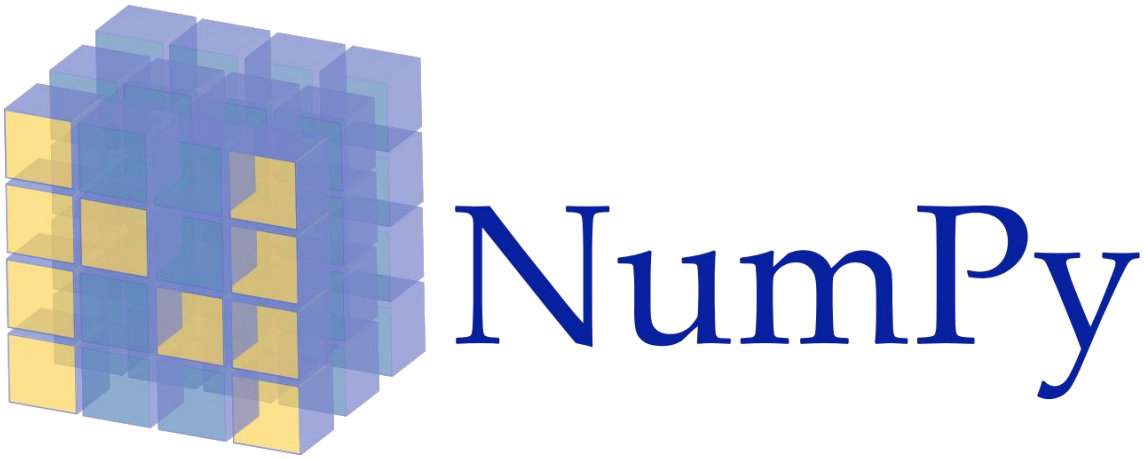

Fig. 4.9: Logo NumPy

La versión añadida a mi proyecto es la versión numpy 1.18.1.

5.- Detección y reconocimiento de señales:

El proyecto que se va a realizar consiste en el reconocimiento de señales de tráfico. En 2019, una ceremonia en Tokio premió un artículo de investigación que revolucionó la manera en la que los coches reconocían las señales de tráfico como la innovación más influyente de la última década.

Fig. 5.1: Señales de tráfico en España

La Convención de Viena sobre Señalización Vial fue un tratado firmado en 1968 que logró estandarizar las señales de tráfico en diferentes países. 31 países de Europa, con un total de 52 países en todo el mundo, han firmado este tratado. Esta estandarización fue el principal impulso para comenzar el desarrollo de los sistemas encargados del reconocimiento de señales de tráfico, pues la existencia de señales de tráfico distintas en cada país habría dificultado en gran medida esta labor.

El primer automóvil con este sistema que salió a la venta fue el BMW Serie 7 en el año 2008, y, ya en 2009, fueron apareciendo más modelos como el Vauxhall Insignia y el Mercedes-Benz Clase S. Sin embargo, por entonces los sistemas solo eran capaces de detectar las señales redondas de límite de velocidad.

La segunda generación de este sistema, que ya permitía el reconocimiento de señales de restricción de adelantamientos fue introducido por el Opel Insignia

y más tarde por el Opel Astra y el Saab 9-5. Posteriormente, marcas como Volkswagen o Volvo también lo introdujeron en algunos de sus modelos.

Los sistemas nuevos que se utilizan hoy en día ya son capaces de reconocer prácticamente la totalidad de señales que te puedes encontrar.

Es tanto la implementación y el uso de estos sistemas en los coches fabricados en la actualidad que se espera que sean obligatorios en todos los automóviles vendidos en la Unión Europea a partir de mayo de 2022.

Estos sistemas son muy útiles, ya no solo por la comodidad de que el coche te indique cual ha sido la última señal que has pasado, sino porque además de la posibilidad de un descuido al no ver una señal, muchas de estas pueden estar situadas en lugares de poca visibilidad o tapadas por algún objeto u obstáculo.

Fig. 5.2: Señal de tráfico de poca visibilidad

6.- Desarrollo del proyecto:

La detección e identificación de señales de tráfico se puede realizar de distintas maneras. Como sabemos existen multitud de señales diferentes, y cada una se diferencia del resto ya sea por los símbolos, por los colores o por la forma de la señal.

Así, se puede detectar la forma de la señal utilizando Python y PyCharm.

Para ello, tenemos una imagen original que contenga formas:

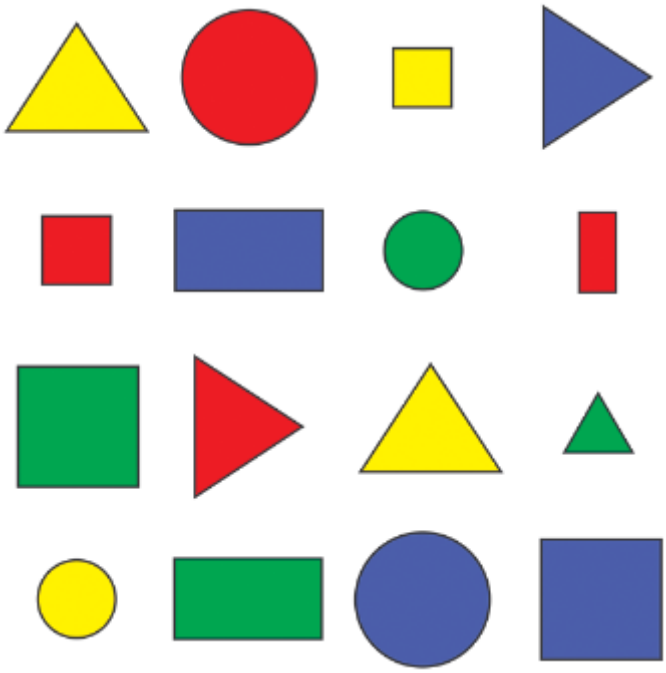

Fig. 6.1: Imagen escogida para las pruebas de detección de formas y colores

Una vez importamos la imagen original, la transformamos a escala de grises para después desenfocarla. Con esto evitaremos problemas por ruidos o problemas por iluminación inadecuada, por ejemplo.

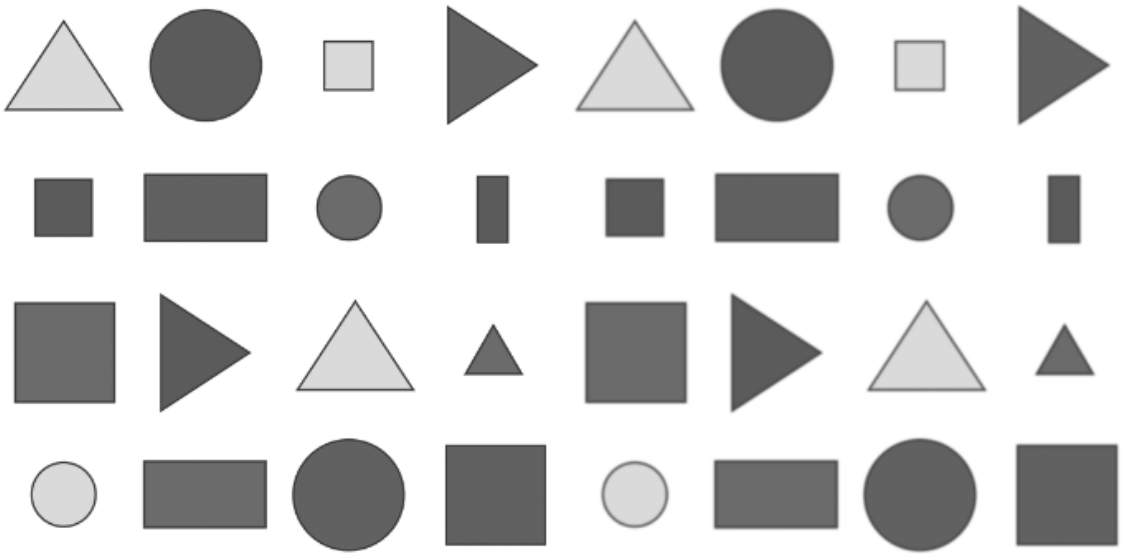

Fig. 6.2: La imagen transformada a escala de grises a la izquierda y desenfocada a la derecha

Una vez tenemos esto, con funciones que trae Python y algunas de sus librerías, podemos pasar la imagen a encontrar los bordes para, con esto, contar el número de esquinas que tiene e identificar qué forma geométrica tiene.

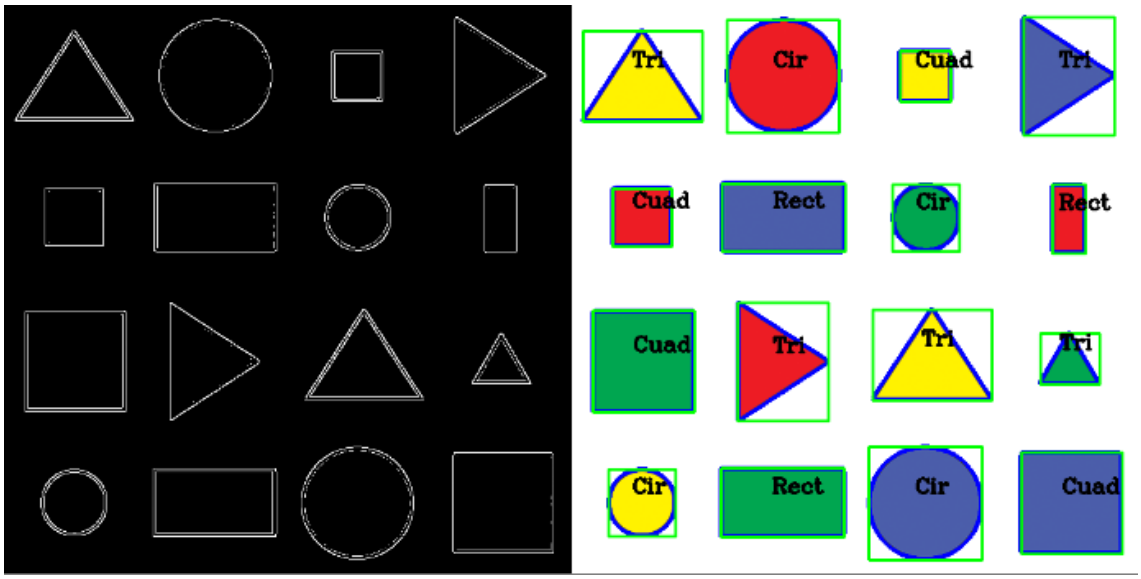

Fig. 6.3: Bordes encontrados a la izquierda y resultado final a la derecha

Asimismo, podríamos detectar las formas geométricas anteriores según el color que tengan. Con funciones encontradas también en Python y en dichas librerías, podríamos separar los colores para identificarlo de esta forma.

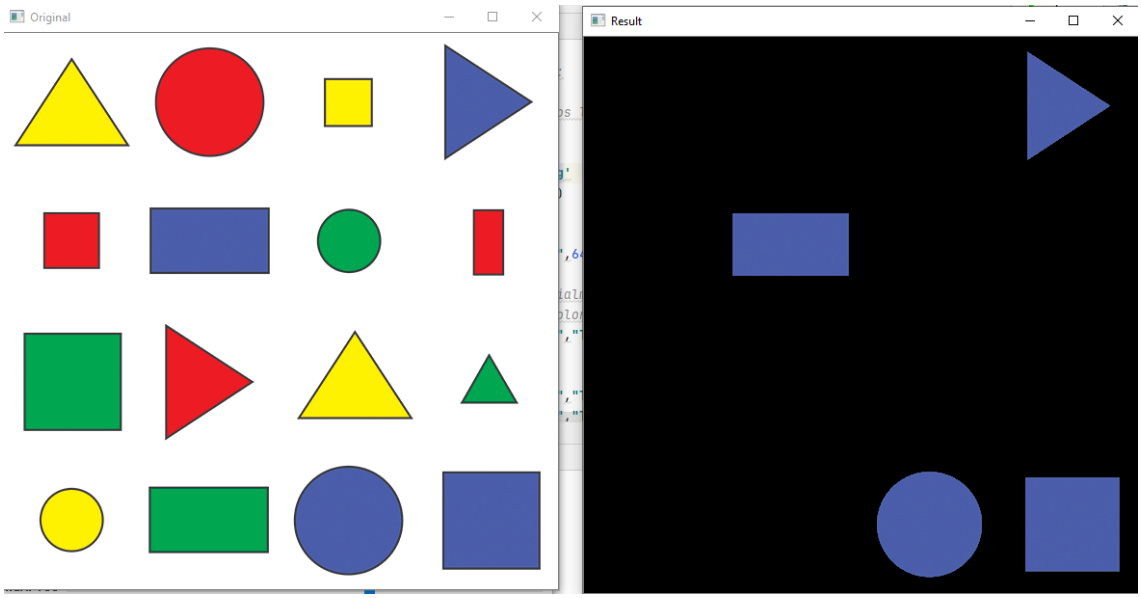

Fig. 6.4: Imagen original a la izquierda y detección del azul a la derecha

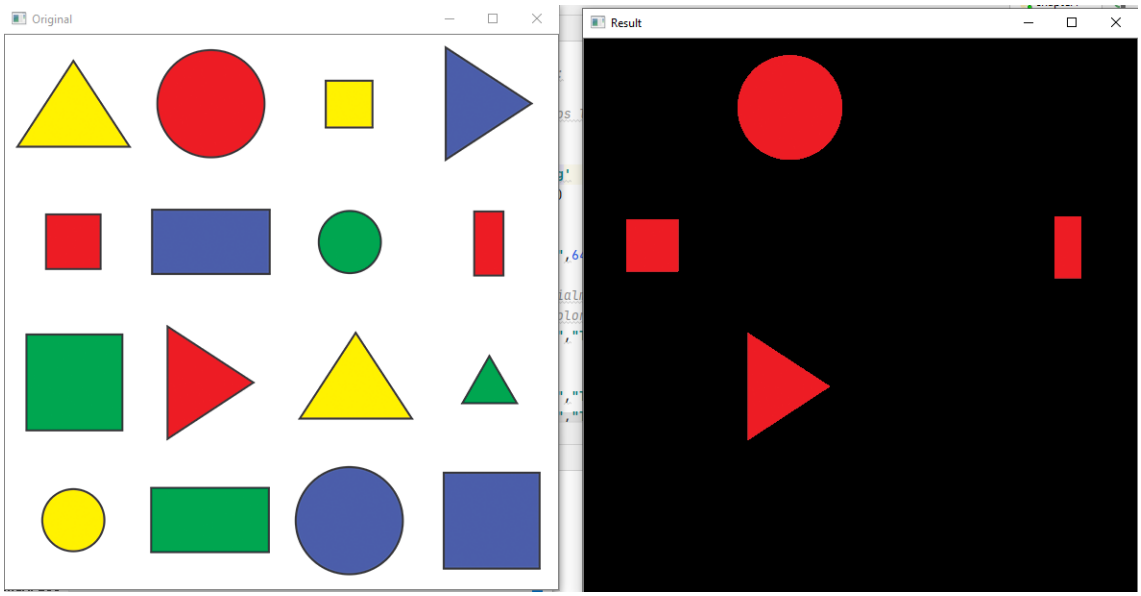

Fig. 6.5: Imagen original a la izquierda y detección del rojo a la derecha

Sin embargo, el método escogido es el de detección de objetos mediante clasificadores en cascada basados en funciones Haar. Este método fue propuesto por Paul Viola y Michael Jones en 2001, y es una forma muy eficaz para detectar objetos.

Se trata de un enfoque basado en el aprendizaje automático, en el que al entregarle un conjunto de imágenes positivas (del objeto a identificar) e

imágenes negativas (de otros objetos) la función se entrena para más tarde detectar los objetos correctamente.

Para realizar esto, se ha descargado una colección de imágenes de señales de tráfico de Internet.

Para crear las cascadas, haremos uso del Cascade Trainer GUI y seguimos los siguientes pasos:

- Crearemos una carpeta en una ubicación cualquiera y dentro de ella crearemos otras dos. Una ha de tener como nombre “p” (positivos) y la otra “n” (negativos).
- En la carpeta “p” introduciremos un conjunto muy grande de la señal de tráfico que queramos identificar. En la carpeta “n” introduciremos por su parte un conjunto más grande aún de imágenes (un número adecuado puede ser del doble al triple que en la otra carpeta) correspondientes a objetos que puedan encontrarse en lugares similares al de las señales de tráfico (como por ejemplo coches, farolas, personas,…), además de otras señales que no sean la que queremos identificar.
- Una vez hecho esto, iniciamos el programa y, en la primera ventana que nos aparece, introducimos la ubicación de la carpeta creada anteriormente en “Samples Folder:”.
- En esta misma ventana, introducimos el número de imágenes que tenemos en la carpeta “n” en “Negative Image Count”.
- Podemos cambiar alguna característica en las siguientes pestañas si lo creemos necesario. Si no, damos al botón Start para iniciar el entrenamiento y solo quedará esperar.

Hay que tener en cuenta algunos aspectos. Cuantas más imágenes hayamos metido en la carpeta “n” y en la carpeta “p” más preciso será el resultado pero, sin embargo, tardará más en finalizar.

Por otra parte, nos puede aparecer al poco tiempo de empezar el siguiente error:

```
OpenCV Error: Bad argument (Can not get new positive sample. The most possible reason is insufficient count of samples in given vec-file.
) in CvCascadeImageReader::PosReader::get, file D:\cv\opencv_3.2.0\sources_withTextModule\apps\traincascade\imagestorage.cpp, line 158
```

Fig. 6.6: Posible error que puede aparecer al ejecutar el programa

Si esto ocurre, bastará con ir a la pestaña inicial “Input” dentro de “Train” y bajar el porcentaje en “Positive Image Usage (percentage):”. En mi caso, al reducirlo al 90% o parecido dejaba de aparecer dicho error.

Una vez terminado el entrenamiento, nos aparecerán en la carpeta seleccionada distintos archivos y una carpeta "classifier" en la que encontraremos esto en su interior:

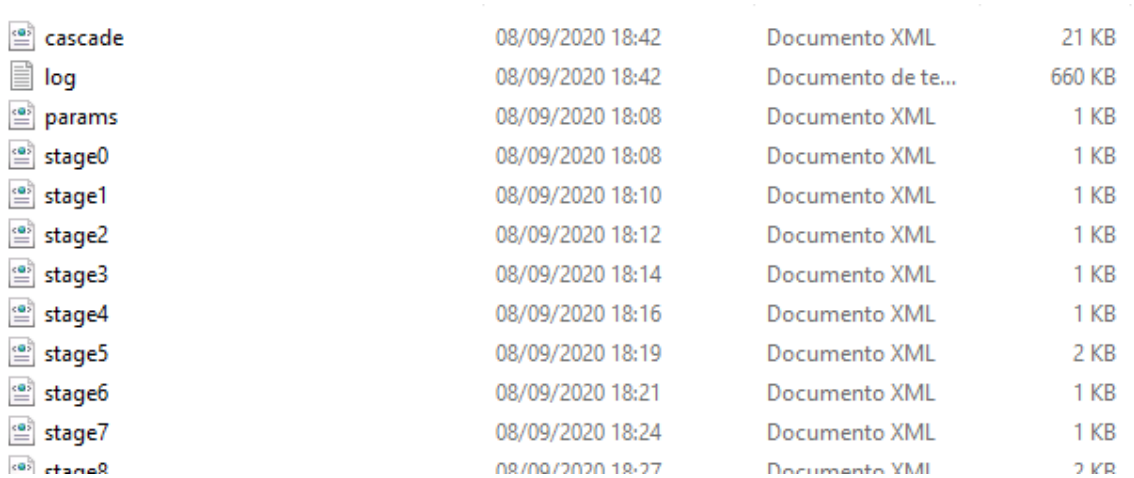

Fig. 6.7: Archivos que deben aparecer al finalizar la ejecución del programa

El archivo que necesitamos es el llamado "cascade", y será el que necesitemos importar en nuestro proyecto. Deberemos hacer esto con todas las señales que queremos identificar. En mi caso, he elegido las de ceda el paso, stop, límite de velocidad de 30 km/h y la de prohibido el paso.

Ejemplos del tiempo que me ha tardado en entrenar son los 48 minutos y medio que duró el entrenamiento de la señal de límite de velocidad de 30 km/h (con 700 imágenes positivas y 1476 imágenes negativas) o los 29 minutos y 44 segundos que tardó el de prohibido el paso (con 578 imágenes positivas y 1626 imágenes negativas).

Una vez tenemos todos los clasificadores en cascada creados pasaremos a la parte del código en PyCharm.

Al crear nuestro proyecto y tener lista nuestra plantilla con las librerías importadas, deberemos crear todas las variables que necesitaremos a lo largo del código:

```
path = 'Resources/haarcascadeceda.xml'  #Ubicación de las cascadas
path2 = 'Resources/haarcascadestop.xml'
path3 = 'Resources/haarcascadelimite30.xml'
path4 = 'Resources/haarcascadenopasar.xml'
numcamara = 0  #Número de la cámara
nombre = 'Ceda el paso'  #Nombres que mostrar
nombre2 = 'STOP'
nombre3 = 'Limite de 30 Km/h'
nombre4 = 'Entrada prohibida'
anchovent = 640  #Anchura de la ventana que queramos crear de la cámara
alturavent = 480  #Altura de la ventana que queramos crear de la cámara
color = (255, 0, 0)  #Color azul
```

Fig. 6.8: Variables del código

Las variables "path" llevan la ubicación de los archivos de las cascadas generados con el anterior programa, mientras que "nombre" llevan los nombres correspondientes. La variable "numcamara" contiene el número de la cámara conectada que queramos utilizar. Si solamente hay una cámara conectada utilizaremos el 0 y en caso de tener más de una cámara conectada habrá que probar número por número para saber cuál es la que buscamos.

Para una mejor detección de las señales por causa de la calidad de vídeo y de otros factores hemos creado una serie de controles deslizables para ajustar ciertos parámetros mientras visualizamos por la cámara.

```
cv2.namedWindow("Resultado")
cv2.resizeWindow("Resultado", anchovent, alturavent + 100)
cv2.createTrackbar("Escala", "Resultado", 130, 1000, empty)
cv2.createTrackbar("Neig", "Resultado", 13, 50, empty)
cv2.createTrackbar("Area Min", "Resultado", 0, 100000, empty)
cv2.createTrackbar("Brillo", "Resultado", 180, 255, empty)
```

Fig. 6.9: Creación de controles deslizables

Con la función cv2.resizeWindow, importada de la librería OpenCV, podemos crear una ventana, escribiendo su nombre (en este caso "Resultado") y el acho y alto que deberá tener.

La función cv2.createTrackbar es una función también importada de OpenCV y nos permite crear un control deslizable de manera sencilla, solamente escribiendo el nombre del control, la ventana donde aparecerá y los valores de inicio y máximo.

Con el control de "Escala" podremos mejorar los resultados. Cuanto más lo disminuyamos mejores serán los resultados. Sin embargo, esto hará que el ordenador necesite más potencia de procesamiento para realizarlo.

Disminuyendo el control de “Neig” (minimum neighbors) detectará más objetos, pero también nos podrá dar más falsas detecciones.

El control de “Area Min” lo utilizaremos para que no detecte objetos más pequeñas que lo que indiquemos.

Por último con el control “Brillo” podremos modificar el brillo de la cámara para ajustarlo a nuestras necesidades.

El siguiente paso consiste en cargar cada uno de los clasificadores con la siguiente función de OpenCV:

```
cascade = cv2.CascadeClassifier(path)
cascade2 = cv2.CascadeClassifier(path2)
cascade3 = cv2.CascadeClassifier(path3)
cascade4 = cv2.CascadeClassifier(path4)
```

Fig. 6.10: Se cargan los clasificadores

Y usamos estas cascadas para detectar en la imagen:

```
#Usa las cascadas creadas para detectar en la imagen:
scaleVal = 1 + (cv2.getTrackbarPos("Escala", "Resultado") / 1000)
neig = cv2.getTrackbarPos("Neig", "Resultado")
objecto = cascade.detectMultiScale(gray, scaleVal, neig)
objecto2 = cascade2.detectMultiScale(gray ,scaleVal, neig)
objecto3 = cascade3.detectMultiScale(gray, scaleVal, neig)
objecto4 = cascade4.detectMultiScale(gray, scaleVal, neig)
```

Fig. 6.11: Uso de las cascadas

Para la función “detectMultiScale” necesitaremos ingresar primero la imagen a revisar (en este caso “gray”, que es la imagen transformada a escala de grises para su mejor detección), el factor de escala (“scaleVal”, creada en la misma imagen teniendo en cuenta el valor del control deslizable “Escala”) y el valor de *minimum neighbors* (“neig”, que adquiere el valor del control deslizable “Neig”)

Posteriormente mostraremos los objetos detectados para cada clasificador en cascada creado (debemos poner el código una vez por cascada):

```
#Muestra los objetos detectados:
for (x, y, w, h) in objecto:
    area = w * h
    minArea = cv2.getTrackbarPos("Area Min", "Resultado")
    if area > minArea:
        cv2.rectangle(img, (x, y), (x + w, y + h), color, 3)
        cv2.putText(img, nombre, (x, y - 5), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, color, 2)
        roi_color = img[y:y + h, x:x + w]
```

Fig. 6.12: Se muestran los objetos detectados

Con el código de esta imagen crearemos un rectángulo alrededor del objeto detectado y añadiremos el nombre de la señal de tráfico recogido en las variables “nombre”, “nombre2”,… Para ello utilizamos más funciones de la librería OpenCV: “cv2.rectangle” y “cv2.putText”.

Y para terminar solo nos queda mostrar la imagen de la cámara en directo por la pantalla, a la vez que mantenerla de forma indefinida. También podemos asignar una tecla para cerrar la ventana, que en mi caso es la ‘q’.

```
cv2.imshow("Resultado", img)
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

Fig. 6.13: Se muestra el resultado por pantalla

La función de OpenCV “cv2.imshow” para mostrar la imagen (“img”, que corresponde a la captura de la webcam) y con “cv2.waitKey(1) & 0xFF == ord(‘q’)” conseguimos que se muestre la imagen de la webcam hasta presionar la letra ‘q’.

La base del funcionamiento del código se puede resumir con la siguiente figura:

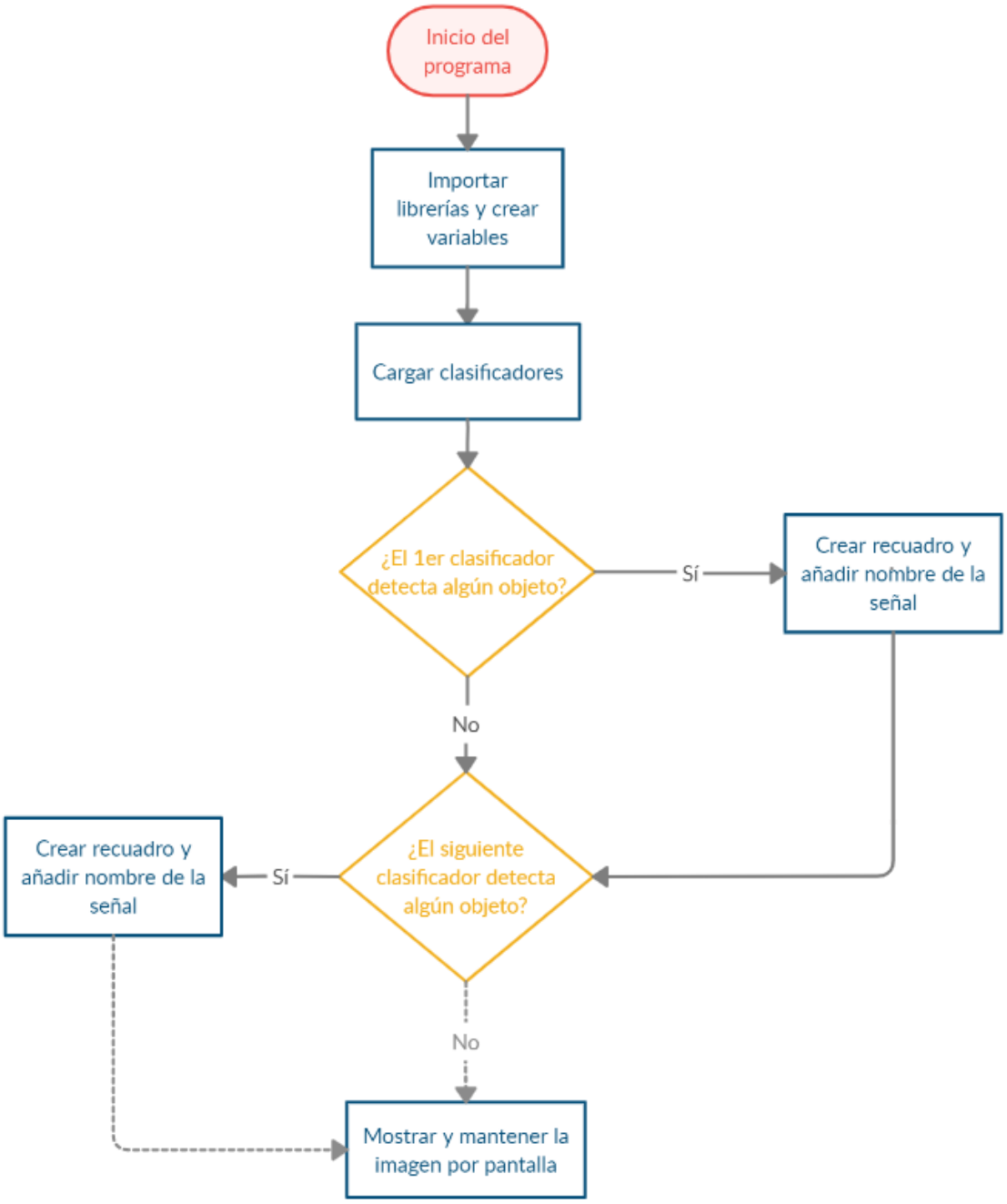

Fig. 6.14. Funcionamiento del código.

7.- Resultados:

Una vez realizado todo el código lo primero que probé fue a importar una imagen de Internet para probar su identificación:

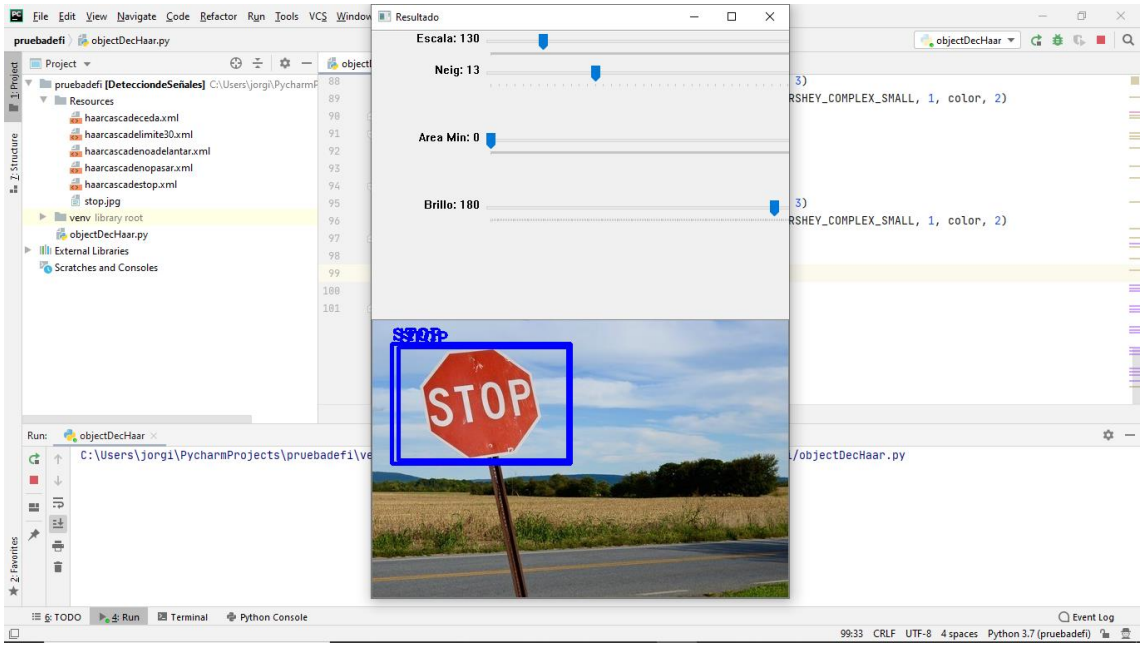

Fig. 7.1: Detección de una señal de STOP de una foto

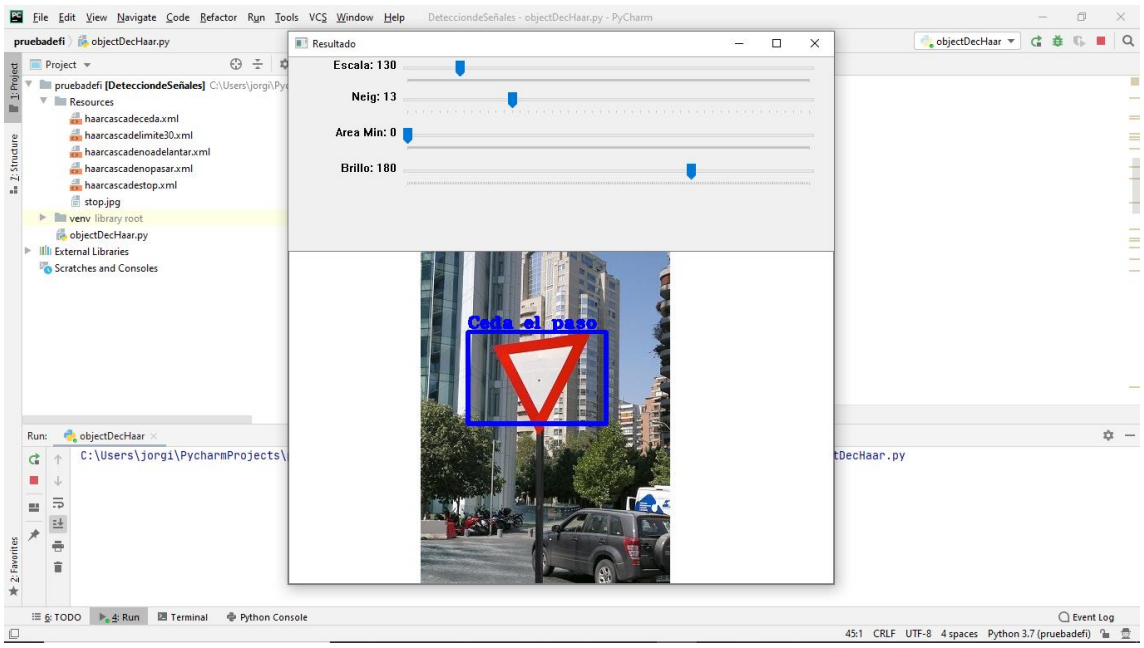

Fig. 7.2: Detección de una señal de ceda el paso de una foto

En las figuras 8.1 y 8.2, podemos observar cómo el programa detecta correctamente las señales, una señal de STOP y una de ceda el paso en estos casos. Encima de dichas señales se crea el recuadro implementado en el código del color indicado y encima escrito el nombre de la señal.

Con ciertas imágenes extraídas de Internet se detectaban muchas más señales de las que realmente había en la imagen, pero el cómputo global fue satisfactorio.

Una vez probado esto, di paso a la cámara en vivo para probar su funcionamiento. Como las primeras pruebas fueron con una sola cascada insertada en el código, en estas solo enseñaba una señal por turno.

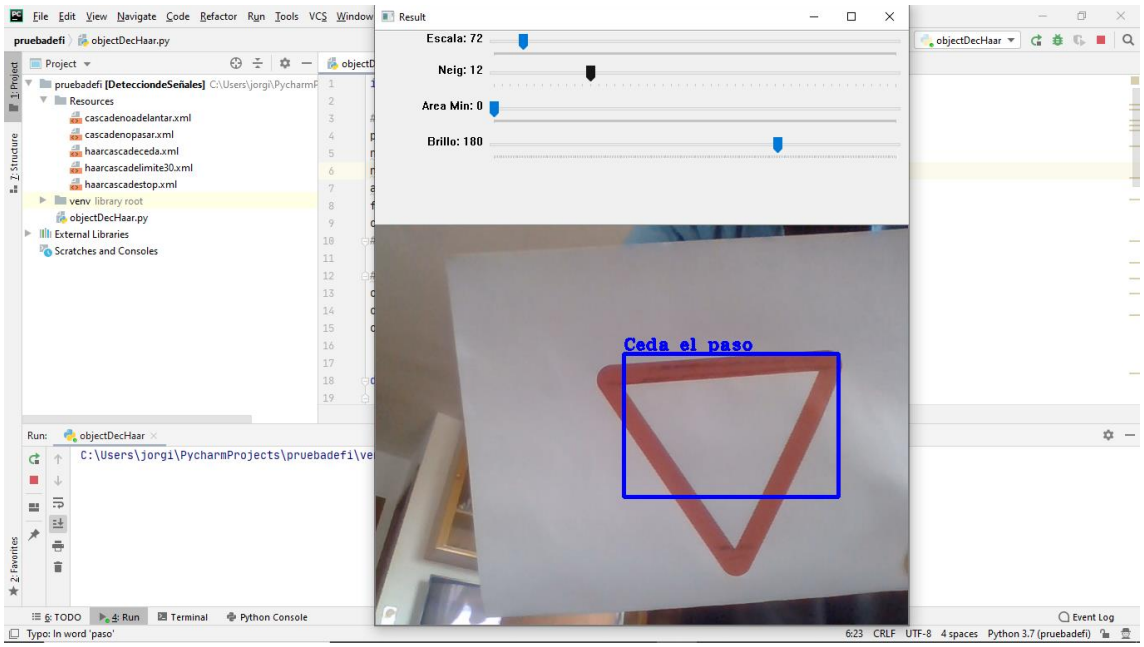

Fig. 7.3: Detección de una señal de ceda el paso a través de la webcam

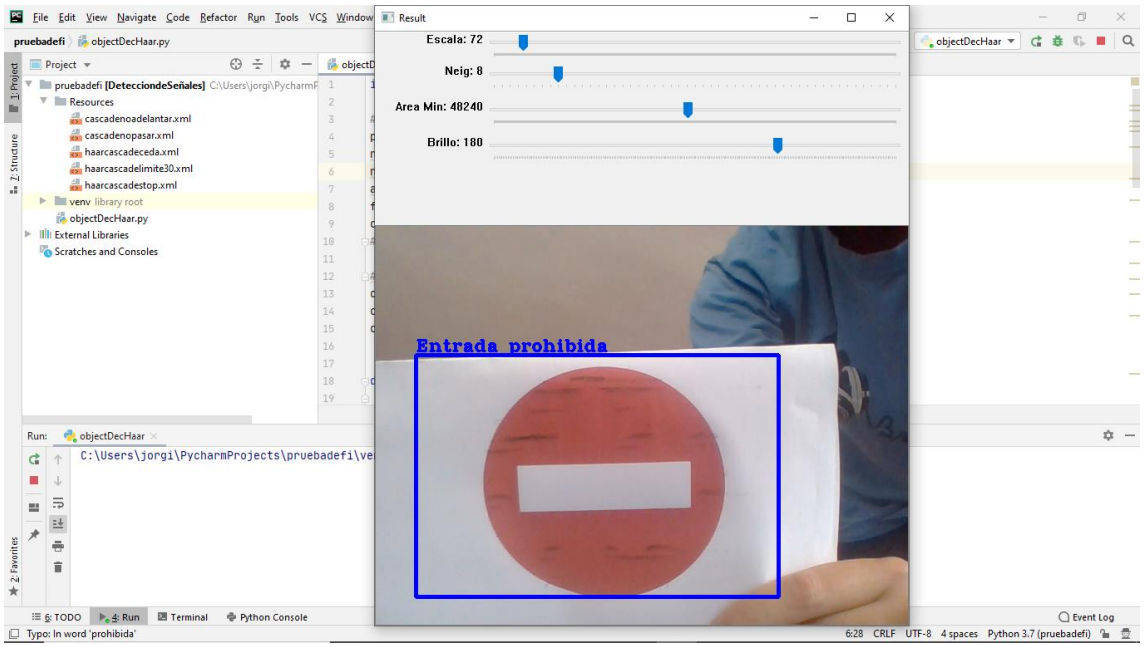

Fig. 7.4: Detección de una señal de entrada prohibida a través de la webcam

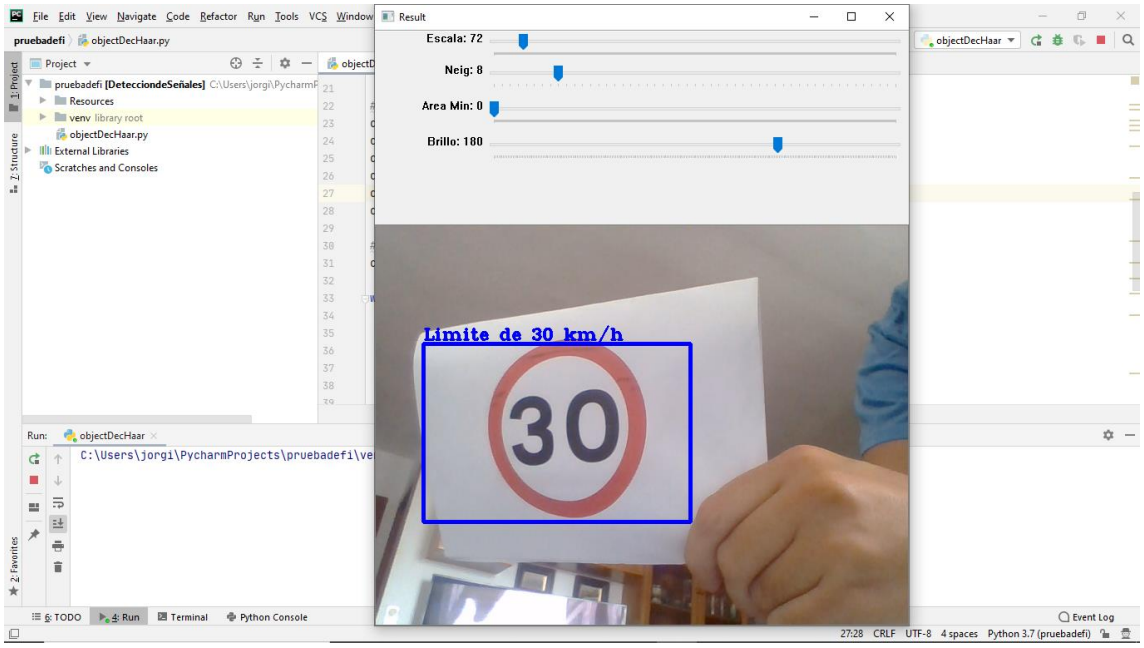

Fig. 7.5: Detección de una señal de límite de velocidad de 30 km/h a través de la webcam

Podemos ver cómo la aplicación funciona correctamente y detecta las señales que aparecen en pantalla. Es posible ir moviendo la señal de lado a lado y se seguirá detectando correctamente, con el recuadro siguiendo la señal por su trayectoria.

La señal que peores resultados me dio fue la de entrada prohibida, que en muchos casos la confunde con la señal de STOP. Cambiando los parámetros pude mejorar un poco los resultados de dicha señal. Sin embargo, para un resultado

más preciso hubiera sido necesario una cantidad de imágenes positivas y negativas mucho más amplio, mejorando así el entrenamiento de la cascada.

Una vez realizado esto, pasé a añadir todos los clasificadores de cascada al mismo tiempo, cosa que no me dio más problemas que las comprendidas anteriormente.

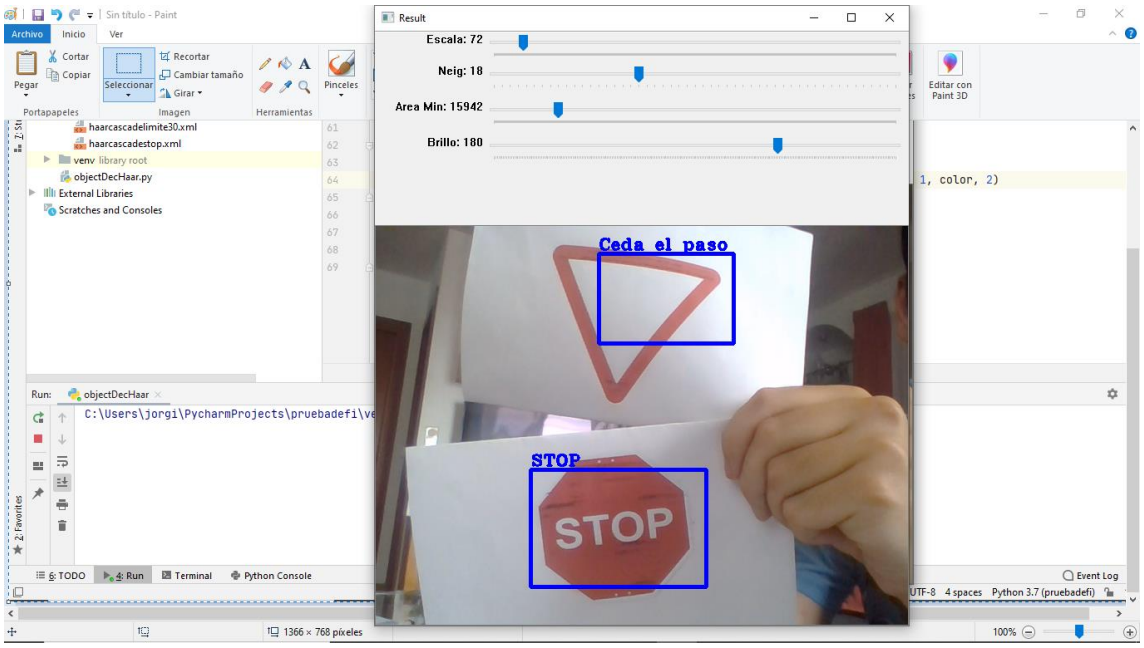

Fig. 7.6: Detección de dos señales simultáneamente a través de la webcam

Tal y como podemos ver en la figura 8.6, la última prueba realizada con la aplicación, el resultado consigue ser bastante preciso. Es importante encontrar el punto preciso de los parámetros probando mover todos los controles deslizables que hemos creado.

Una de las ventajas de usar unos clasificadores en cascada es que en el caso de implementar un sistema en un vehículo será capaz de detectar señales de tráfico que se vean parcialmente. En caso de tener que detectar la forma de una señal, si esta se ocultara parcialmente, el sistema no sería capaz de identificar la forma de la señal correctamente.

8.- Conclusiones:

Finalmente se ha conseguido cumplir con los objetivos indicados a principio de este documento. La realización del proyecto ha supuesto un reto importante para la introducción y la mejora en muchos de los aspectos utilizados en este proyecto. De forma resumida, se puede decir que con este proyecto:

- He aprendido sobre el funcionamiento de muchas de las partes electrónicas de las que consiste un coche.
- He entendido y me he informado sobre la visión artificial, los proyectos que se pueden llegar a realizar con ello y el gran futuro que tiene dentro del mundo de la tecnología.
- Me he introducido en el mundo del lenguaje Python y de todas las posibilidades que da un lenguaje sencillo de comprender y con tanta información disponible en Internet.

Además de esto, el hecho de encontrarse y enfrentarse a errores en un proyecto así te hace aprender más y saber cómo actuar frente a futuros errores que puedan aparecer.

El coste para hacer un sistema de reconocimiento de señales de tráfico es relativamente bajo para incluir en cualquier coche con resultados satisfactorios, incluido su implementación en coches antiguos, y con ello evitar muchos accidentes o, al menos, reducir sus consecuencias. Aun así, en el caso de necesitar un sistema 100% preciso y mucho más efectivo es posible que el coste aumente, aunque no en gran medida. Para ello sería necesario disponer de una grandísima base de imágenes de todas las señales que se quieran identificar y de un sistema de grabación y procesamiento de imágenes más veloz.

Este trabajo me permite tener una gran base para, en un futuro, poder utilizar todo el conocimiento adquirido para cualquier proyecto en el que se necesite aplicar la visión artificial o simplemente programar.

9.- Bibliografía:

CARGLASS. *El sistema de reconocimiento de señales de tráfico*.

<https://www.carglass.es/blog/conduce-seguro/sistema-de-reconocimiento-de-senales-de-trafico/>

GUEMACAR. *La evolución de los sistemas electrónicos de los automóviles.*

<https://www.solucionesguemacar.es/blog/entry/la-evolucion-de-los-sistemas-electronicos-de-los-automoviles>

DARREN COTTINGHAM. *How does ABS (anti-lock braking) work?*

<https://www.drivingtests.co.nz/resources/how-does-abs-anti-lock-braking-work/>

MOTOR.ES. *¿Qué es el motor de arranque? Funcionamiento y averías.*

<https://www.motor.es/que-es/motor-arranque>

HELLA. *Headlamps.* <https://www.hella.com/hella-sg/de/Headlamps-201.html>

HELLA. *Regulación del alcance luminoso.*

<https://www.hella.com/techworld/es/Informacion-Tecnica/Iluminacion/Regulacion-del-alcance-luminoso-838/>

NEUMÁTICOSKM0. *¿Qué es la dirección asistida y cómo funciona?*

<https://www.neumaticoskm0.com/direccion-asistida/>

AUTOBIZ. *Acceso sin llave.* <https://autobiz-ocasion.es/blog/consejos/acceso-sin-llave.php>

ENDADO. *Así funciona el regulador de velocidad activo.*

<https://www.endado.com/blog/asi-funciona-el-regulador-de-velocidad-activo/>

DEAN GIBSON. *What is climate control air-conditioning?*

<https://www.autoexpress.co.uk/car-tech/102614/what-is-climate-control-air-conditioning>

RUBÉN FIDALGO. *Cómo funciona el control de tracción.*

<https://www.autocasion.com/actualidad/reportajes/control-de-traccion-el-acelerador-inteligente>

LA COMUNIDAD DEL TALLER. *¿Qué hay de la llegada del vehículo autónomo?*

<https://www.lacomunidaddeltaller.es/que-hay-de-la-llegada-del-vehiculo-autonomo/>

TESLA. *El futuro de la conducción.* <https://www.tesla.com/es_ES/autopilot>

CARGACAR. *Autopilot de Tesla, cómo funciona y qué ventajas tiene.*

<https://cargacar.com/noticias/autopilot-tesla-como-funciona/>

INFAIMON. *Historia de la visión artificial: así ha evolucionado esta tecnología.*

<https://blog.infaimon.com/historia-vision-artificial/>

MANUEL ZAFORAS. *¿Es Python el lenguaje del futuro?*

<https://www.paradigmadigital.com/dev/es-python-el-lenguaje-del-futuro/>

PYTHON. *The Python tutorial.* <https://docs.python.org/3/tutorial/>

ESCUELA DE PYTHON. *PyCharm: uno de los mejores IDE para Python.*

<https://www.escuelapython.com/pycharm-uno-de-los-mejores-ide-para-python/>

WIKIPEDIA. <https://www.wikipedia.org/>

DOCUMENTO II:
PRESUPUESTO

Para el desarrollo y la implementación de este proyecto será necesario, además de un ingeniero electrónico, el uso de un ordenador, el software utilizado y una cámara para introducir en el coche.

Para el coste del personal estimaremos un sueldo de ingeniero electrónico de 18.500 € brutos al año.

El tiempo de diseño y de desarrollo del proyecto puede haber sido de 6 meses a una media de 2 horas diarias, o lo que podría resultar lo mismo a 3 meses de 4 horas diarias. Teniendo en cuenta el sueldo asignado y los 3 meses de trabajo calculamos:

$$Coste\ del\ personal: \frac{\frac{18500\ €\ al\ año}{2}}{12\ meses} = 771\ €\ al\ mes$$

El resultado del coste del personal sería de unos 771 € al mes por 3 meses de jornada parcial. Esto es:

$$Coste\ total\ del\ personal = 771\ €\ al\ mes \cdot 3\ meses = 2313\ €$$

Para las licencias de software tenemos lo siguiente:

- Por un lado, Cascade Trainer GUI es un programa totalmente gratuito, por lo que su coste es 0 €.
- Por otro tenemos PyCharm. PyCharm Community Edition es la versión gratuita de PyCharm. Sin embargo, tenemos que tener en cuenta PyCharm Proffesional, cuyo precio es de 199 € el primer año.

Para la cámara que introduciremos en el coche (en la parte frontal, muchas veces incorporado en el logo delantero), el precio estimado en una compra de grandes cantidades (cámaras de este tipo se pueden llegar a pedir cientos de miles cada año) puede ser de unos 23 euros la unidad. Si a esto le sumamos el IVA correspondiente:

$$Coste\ cámara = 23 + 23 \cdot 0{,}21 = 27{,}83\ €$$

Por último tenemos que contar el ordenador utilizado. Para ello, la agencia tributaria española, en su tabla de coeficientes de amortización lineal, estima el coeficiente lineal máximo de los equipos electrónicos en un 20% y un período máximo de 10 años. Teniendo en cuenta un ordenador de 700 € (IVA ya incluido):

$$Coste\ de\ un\ año = 700€ \cdot 20\% = \ 140\ €$$

$$Coste\ de\ 6\ meses = 140 \cdot \frac{6\ meses}{12\ meses} = 70\ €$$

Así, tenemos los siguientes resultados:

ELEMENTO	COSTE (€)
Ordenador	70
Personal	2313
Cámara	27,83
Licencias	199
Total	2609,83

Tabla 1: Coste total