UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

DISEÑO DE UN SISTEMA DE BAJO COSTE PARA LA DETECCIÓN DE LA SOMNOLENCIA EN LA CONDUCCIÓN BASADO EN RECONOCIMIENTO DE EXPRESIONES FACIALES

AUTOR: JULIO CANO BERNET

TUTOR: EDUARDO QUILES CUCARELLA

COTUTOR: CARLOS ROLDÁN PORTA

Curso Académico: 2020-21

AGRADECIMIENTOS

Quiero agradecer con unas pocas palabras a la gente que ha colaborado en el desarrollo de este proyecto.

En primer lugar, gracias Eduardo y Carlos por ofrecerme la posibilidad de realizar un proyecto con una temática tan original para un ingeniero industrial y por estar siempre disponibles con extrema rapidez cuando os he necesitado. Eduardo, gracias por introducirme en el apasionante campo de la monitorización de señales humanas, que tras alguna lluvia de ideas derivó en la detección de somnolencia mediante reconocimiento facial. Carlos, gracias por atenderme y por tu disposición para echar una tarde ayudándome en los inconvenientes encontrados durante el montaje. En general, gracias a los dos por los momentos en los que hemos visto las infinitas posibilidades que ofrece un proyecto de este tipo. Han quedado muchas cosas en el tintero, espero que se continúe el proyecto con algunas de nuestras ideas.

En segundo lugar, tengo que agradecer a mis padres su implicación en el proyecto. Gracias por vuestro interés y apoyo y por lo ánimos que me habéis dado. Gracias por prestaros a hacer pruebas. Especial mención al ingeniero de la casa, mi padre, que si bien no es ingeniero de profesión, es quien tiene el ingenio para idear todo tipo de montajes y apaños.

Por último, gracias Rocío por participar en las pruebas y por ser mi compañera de batalla durante la ejecución del trabajo.

RESUMEN

Se estima que entre un 20 y un 30 % de los accidentes de tráfico están relacionados con la fatiga, siendo los accidentes causados por somnolencia hasta dos veces más mortales que los originados por otras causas. Para reducir estas cifras, se han llevado estrategias como la realización de campañas publicitarias, la implementación de tacógrafos en vehículos destinados al transporte por carretera de mercancías y viajeros o el uso de sistemas de detección de somnolencia en automóviles. En el ámbito de esta última, las tecnologías usadas son de diversa índole y pueden basarse en la medición de señales como el movimiento del volante o la posición del vehículo en la carretera o en la monitorización del conductor. El control del conductor es una tecnología poco explotada hasta ahora y que puede llevarse a cabo con enfoques muy distintos. En este proyecto se abordará el registro de expresiones faciales que muestren indicios de somnolencia y se estudiará la viabilidad de su implementación en un paquete electrónico de bajo coste.

En concreto, se pretende determinar el estado del conductor mediante la monitorización de sus expresiones faciales, como pueden ser frecuencia de parpadeos, bostezos, cabeceo o apertura de los ojos. Para ello, se va a desarrollar un algoritmo capaz de detectar la somnolencia y se va a programar en Python. La implementación se llevará a cabo sobre una Raspberry Pi, un dispositivo de bajo coste que permitirá realizar un prototipo que pueda detectar somnolencia e interactuar con periféricos como cámaras o altavoces.

PALABRAS CLAVE: somnolencia, expresiones faciales, bajo coste, prototipo

RESUM

S'estima que entre un 20 i un 30% dels accidents de trànsit estan relacionats amb la fatiga, sent els accidents causats per somnolència fins a dues vegades més mortals que els originats per altres causes. Per a reduir aquestes xifres, s'han dut a terme estratègies com la realització de campanyes publicitàries, la implementació de tacògrafs en vehicles destinats al transport per carretera de mercaderies i viatgers o l'ús de sistemes de detecció de somnolència en automòbils. En l'àmbit d'aquesta última, les tecnologies emprades són de diversa índole i poden basar-se en la mesura de senyals com el moviment del volant o la posició del vehicle en la carretera o en la monitorització del conductor. El control del conductor és una tecnologia poc explotada fins ara i que pot dur-se a terme amb enfocaments molt diferents. En aquest projecte s'abordarà el registre d'expressions facials que mostren indicis de somnolència i s'estudiarà la viabilitat de la seva implementació en un paquet electrònic de baix cost.

En concret, es pretén determinar l'estat del conductor mitjançant la monitorització de les seues expressions facials, com poden ser freqüència de pestanyeigs, badalls, capcinades o obertura dels ulls. Per a això, es desenvoluparà un algoritme capaç de detectar la somnolència i es programarà en Python. La implementació es durà a terme sobre una Raspberry Pi, un dispositiu de baix cost que permetrà realitzar un prototip que puga detectar somnolència i interactuar amb perifèrics com càmares o altaveus.

PARAULES CLAU: somnolència, expressions facials, baix cost, prototip

ABSTRACT

It is estimated that 20-30% of road accidents are related to fatigue, being accidents caused by drowsiness up to twice as deadly as those caused by other causes. In order to reduce these numbers, strategies such as advertising campaigns, the implementation of tachographs in vehicles used for road transport of goods and passengers or the use of drowsiness detection systems in cars have been implemented. Within the scope of the latter area, the technologies used are diverse and can be based on measurement of signals such as steering wheel movement or vehicle position on the road, or on driver monitoring. Driver monitoring is a technology that has been little exploited so far and can be implemented in many different approaches. This project will address the recording of facial expressions that show signs of drowsiness and study the feasibility of its implementation in a low-cost electronic package.

Specifically, the aim is to determine the state of the driver by monitoring their facial expressions, such as frequency of blinking, yawning, nodding or eye opening. For this purpose, an algorithm capable of detecting drowsiness will be developed and programmed in Python. The implementation will be carried out on a Raspberry Pi, a low-cost device that will allow to create a prototype that can detect drowsiness and interact with peripherals such as cameras or speakers.

KEY WORDS: drowsiness, facial expressions, low-cost, prototype

ÍNDICE

DOCUMENTOS CONTENIDOS EN EL TFM

- Memoria
- Presupuesto

ÍNDICE DE LA MEMORIA

ÍNDICE DEL PRESUPUESTO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

DISEÑO DE UN SISTEMA DE BAJO COSTE PARA LA DETECCIÓN DE LA SOMNOLENCIA EN LA CONDUCCIÓN BASADO EN RECONOCIMIENTO DE EXPRESIONES FACIALES

DOCUMENTO I: MEMORIA

1. INTRODUCCIÓN

1.1. MOTIVACIÓN

El transporte por carretera es un medio de desplazamiento muy usado actualmente, tanto para largas distancias donde se encuentran el sector transportes o viajes puntuales no relacionados con este sector como para pequeños desplazamientos habituales como pueden ser dirigirse al trabajo.

En el ámbito de desplazamientos habituales hasta el lugar de trabajo, una encuesta elaborada por Michael Page [1] revela que en España la duración media del trayecto es de 36 minutos, siendo este un tiempo inferior al de la mayoría de los países de Europa. Alrededor de un 50% de trabajadores en España trabaja en un municipio distinto a dónde reside, y dos tercios de los profesionales usan el transporte privado para ir a trabajar.

Los accidentes acaecidos durante estos trayectos se conocen como accidentes *in itinere* y constituyen una fuente de riesgo específica al volante. En 2018 se produjeron 136 muertes en accidentes *in itinere* [2]. En este sentido, la DGT identifica como factor de riesgo los trayectos rutinarios y ha realizado campañas específicas para reducir este tipo de incidentes, pues en 2014 hubo 38.010 accidentes *in itinere* [3] (8% de los accidentes laborales en 2014).

La fatiga juega un papel muy importante en este tipo de accidentes, siendo esta la responsable de hasta un 30 % de los accidentes de tráfico [4]. La fatiga puede aparecer por muchos motivos, entre los cuáles se encuentra los siguientes, que se pueden dar fácilmente durante la tipología de accidentes descrita:

- El alto nivel de tráfico en horas punta de acceso al trabajo, lo que supone aumentar la atención y la concentración necesaria y propicia la aparición de fatiga.
- Una iluminación deficiente, como por ejemplo circular por la noche o de madrugada, requiere también un mayor nivel de atención y potencia el cansancio
- La monotonía del entorno (autopistas o autovías)
- La prisa exige una mayor concentración y puede alterar el estado psicofísico

A estos factores que potencian la fatiga hay que sumarle la posibilidad de conducir estando ya fatigado o con somnolencia.

Los accidentes causados por somnolencia son hasta dos veces más mortales que los originados por otras causas. Para reducir estas cifras, se han llevado estrategias como la realización de campañas publicitarias, la implementación de tacógrafos en vehículos destinados al transporte por carretera de mercancías y viajeros o el uso de sistemas de detección de somnolencia en automóviles. En el ámbito de esta última, las tecnologías usadas son de diversa índole y pueden basarse en la medición de señales como el movimiento del volante o la posición del vehículo en la carretera o en la monitorización del conductor. El control del conductor es una tecnología poco explotada hasta ahora y que puede llevarse a cabo con enfoques muy distintos. En este proyecto se abordará el registro de expresiones faciales que muestren indicios de somnolencia y se estudiará la viabilidad de su implementación en un paquete electrónico de bajo coste.

1.2. ESTADO DEL ARTE/ANTECEDENTES

En la industria automovilística se han estado implementando diversos sistemas que alertan de la aparición de fatiga. A estos sistemas se los conoce por sus siglas en inglés como DDD (*Driver Drowsiness Detection*), y su presencia hasta ahora se limitaba a paquetes de extras que no vienen incluidos en las versiones básicas de los vehículos, si es que el modelo de automóvil en cuestión lo ofrece. Sin embargo, la Unión Europea quiere que, a partir de 2022, todos los coches nuevos monten asistentes electrónicos para aumentar la seguridad como equipamiento de serie obligatorio [5].

Entre las propuestas que plantea, se encuentra la monitorización del volante (presión de las manos, correcciones, cambios de dirección, etc.), la monitorización de las facciones de la cara, movimientos de ojos y cabeza, etc. con cámara y reconocimiento de imágenes y análisis del tiempo que lleva el motor en marcha. No obstante, las opciones para realizar la tarea de detectar el cansancio no acaban aquí. El abanico de ideas es muy amplio y con planteamientos muy distintos. A continuación, se van a analizar las tecnologías que usan algunos de los fabricantes con más presencia en el mercado.

1.2.1. Sistemas basados en señales biomédicas

Estos sistemas se fundamentan en la toma de señales cerebrales, musculares o cardiovasculares. Si bien es cierto que se han desarrollado sistemas capaces de captar señales biomédicas de forma no invasiva, en la mayoría de los casos la monitorización de estas señales requiere de la implementación de electrodos en el conductor, por lo que puede causar incomodidad. Por esto, por el momento no se considera viable su implementación comercial a no ser que se encuentren otras formas de registrar las señales corporales sin elementos invasivos.

Para el análisis mediante señales de electroencefalograma, se requiere de cintas o cascos con electrodos para la cabeza:

Figura 1. Sistemas de monitorización EEG. Izquierda: Enobio® (Neuroelectrics), Derecha: Muse™

En la imagen superior se pueden ver dos sistemas de monitorización en tiempo real de señales EEG. En la izquierda, un sistema de medición de alta precisión integrado junto con almacenamiento en tarjeta SD y transmisión *Wireless*. Este equipo tiene cuenta con un mayor número de electrodos y está

enfocado a el uso médico y en gran parte a investigación. A la derecha, una diadema más discreta y pensada para uso doméstico que proporciona un registro de la actividad cerebral por *Bluetooth.*

Aunque la medición de EEG está bien desarrollada y aporta datos muy valiosos, este tipo de señales son muy sensibles al movimiento por lo que su precisión durante la conducción podría no ser buena. Si a esto se suma la necesidad de implementar electrodos en el conductor, la viabilidad de este sistema en el mercado queda muy limitada.

Si bien el mercado no estaría preparado para la entrada de estos productos, pueden ser muy útiles a nivel de laboratorio. El desarrollo de equipos para detección de la somnolencia requiere de procesos de validación con sistemas robustos como es el registro de señales EEG, con el que se analiza el sueño en el sector sanitario. De esta forma, las señales EEG se usan a menudo como referencia para validar otros sistemas de detección de somnolencia [6].

En cuanto al análisis de señales ECG, la llegada del *smartwatch* hace que esta categoría de sistemas recobre cierto potencial. Hasta ahora, los sistemas presentes en automóviles comerciales para monitorizar las señales del corazón se basan en medir las pulsaciones con equipos integrados en el volante o en el asiento del conductor. Esto tiene la desventaja de que la posición del conductor o donde tenga los brazos puede impedir la medida.

El llevar incorporado en la muñeca un dispositivo no invasivo con función de electrocardiograma, acelerómetro y giroscopio puede brindar oportunidades muy valiosas para la detección de somnolencia analizando señales HRV (*Heart Rate Variablity*) y combinándolas con la monitorización del comportamiento del conductor mediante el resto de sensores disponibles en el *smartwatch*.

1.2.2. Sistemas basados en el comportamiento del conductor

El comportamiento del conductor entendido en el sentido en que la trayectoria o la velocidad del coche puede verse perturbada por una actitud desentendida puede ser una gran fuente de información. De esta forma, se puede obtener fácilmente una idea del nivel de atención evaluando señales como la velocidad o el ángulo de giro del volante.

Cuando los conductores se encuentran fatigados, tienden a salirse de la carretera y realizar correcciones bruscas con el volante. Algunos fabricantes como Bosch [7] detectan estas correcciones monitorizando, como se ha dicho más arriba, el ángulo del volante. En esta línea, Audi va más allá: Creando un perfil de conducción para cada piloto del coche basado en las acciones sobre el volante, pedales y botones y analizando las desviaciones respecto a este perfil durante los trayectos es capaz de determinar si hay riesgo [8].

La adquisición de estas señales es sencilla, aunque puede verse limitada por las condiciones de la carretera: Una carretera bacheada o con muchas curvas dificulta este tipo de análisis al introducir variaciones en la marcha que no son en realidad perturbaciones.

Además, analizar los datos y llegar a una conclusión requiere un elevado tiempo de muestreo: Hay que tomar muestras durante un periodo de conducción que se considere correcto para establecer un estándar con el que comparar el resto del trayecto. De esta forma, se determinan unas características en forma de desviaciones sobre el patrón establecido que se identifican con comportamiento errático. Esta metodología podría no ser eficaz en situaciones en las que el conductor presenta somnolencia

desde el inicio de la conducción, pues no sería posible determinar el patrón de conducción de referencia.

Además, averías o imperfecciones en el vehículo (holguras en pedal y volante, vibraciones, mal estado de rótulas, rodamientos o cremallera de dirección…) también limitaría este tipo de análisis.

Por último, bajo condiciones climatológicas adversas como viento lateral muy fuerte, el sistema podría caracterizar el patrón de giro del volante erróneamente como somnolencia.

Por todos estos inconvenientes, hay fabricantes que optan por monitorizar la posición del vehículo respecto a la vía con tecnologías que no impliquen elementos mecánicos del coche. Ford opta por reconocer conducción errática mediante la identificación de la posición del vehículo respecto a las líneas de la carretera con cámaras frontales [9]. PSA hace lo propio con el sistema AFIL mediante sensores infrarrojos justificando que son más baratos y más robustos en condiciones climatológicas adversas [10].

1.2.3. Sistemas basados en el análisis visual del conductor

El término visión artificial o *Computer Vision* hace referencia al análisis visual mediante técnicas de procesamiento de imagen. La ventaja fundamental de la visión artificial es, al igual que en los sistemas basados en análisis del comportamiento del conductor, la nula intrusión sobre el cuerpo del conductor. Por medio de cámaras situadas en el interior del habitáculo del vehículo se pueden monitorizar las expresiones faciales, la actividad ocular, cabeceos…

Las propuestas técnicas para llevar a cabo la detección de somnolencia bajo el principio de visión artificial son muy variadas y en la literatura se plantean distintos tipos de cámaras (infrarrojas, no infrarrojas, estereoscópicas), diferentes algoritmos y distintos motores de clasificación y reconocimiento de imagen (más adelante se hablará de esto).

El principal hándicap de este tipo de tecnología son las condiciones en las que se toma la imagen: la luminosidad o la presencia de objetos que puedan dificultar el análisis de la imagen (complementos de ropa como gafas o sombreros) puede impedir el procesamiento de la imagen, que ya de por sí presenta un elevado coste computacional.

La tecnología desarrollada para llevar a cabo este proyecto pertenece a este tipo de sistemas. Algunos trabajos previos en este campo se basan en la monitorización del parpadeo, del bostezo o de la posición de la cabeza y lo hacen con distintos programas (MATLAB, Python…), distintos algoritmos de reconocimiento facial (redes neuronales, clasificadores…), distinto hardware (PC, ordenadores de placa simple…). En el ámbito del hardware, este proyecto pretende aportar información sobre la implementación del sistema de detección de somnolencia basado en visión artificial en un ordenador de placa simple comercial, ámbito sobre el cuál no hay demasiada información. Además, se pretende aumentar el valor montando el sistema conjuntamente con un equipo de infrarrojos que permita captar imagen en condiciones de baja luminosidad.

En general, los sistemas homologados para controlar la somnolencia se limitan a generar alertas de sueño dirigidas a recomendar descansos al conductor. En algunos casos, hay procedimientos que el

conductor puede activar mediante los cuales se da al vehículo cierta autonomía para realizar acciones correctivas, como regular el volante para mantener el coche dentro del carril.

Recientemente han aparecido tecnologías paralelas bajo el nombre de *Automatic Emergency Braking* que sortean al conductor para actuar directamente sobre el vehículo en caso de extremo peligro de accidente. Es el caso del *City Safety* de Volvo o el *Toyota Safety Sense,* que cuentan con una funcionalidad pre-colisión que detecta peatones, vehículos que circulen delante, etc. Ante peligro inminente y falta de reacción por el piloto previo aviso por el sistema, este aplica automáticamente los frenos para evitar un choque potencial. Sin embargo, estas tecnologías no comparten fundamento con los sistemas de detección de somnolencia más allá de ser un sistema de seguridad.

1.3. OBJETIVOS

El objetivo de este proyecto es desarrollar un sistema basado en el análisis visual del conductor, de bajo coste e instalable en un vehículo para realizar detección de somnolencia en la conducción. Los objetivos que se plantean para el sistema son los siguientes:

- Funcional: Debe ser capaz de detectar la somnolencia. El reconocimiento de imagen, debido a su naturaleza, puede ser desarrollado para una situación concreta o por el contrario extender la validez de su funcionamiento para distintas situaciones, lo cual implica algoritmos más complejos, programas más pesados y hardware más potente. Como elementos que afectan a la imagen del rostro que se capta se encuentran la luz ambiental, complementos como gafas o sombreros, etc. Además, hay que tener en consideración otros factores como los tipos de rasgos de la persona o la distancia entre la cámara y el rosto, pues también limitan la funcionalidad del sistema. Para evaluar la funcionalidad del prototipo habrá que definir un criterio de validez. En este proyecto se pretende desarrollar un equipo razonable partiendo de la base de que los recursos de los que se dispone son limitados, por lo que el objetivo no es desarrollar un sistema que funcione bajo cualquier situación. Este criterio se definirá más adelante (capítulo 3.1.1) una vez se tenga una idea clara sobre las capacidades del método planteado

- Bajo coste: Debe servir para justificar que este tipo de sistemas no encarece en exceso el precio de un vehículo. Esto permitirá sembrar un precedente en la industria automovilística para que cualquier fabricante considere implementar esta funcionalidad en sus vehículos, ya sea como extra o como equipamiento básico. Para ello, los equipos que formen parte del prototipo deberán ser productos genéricos del mercado, no especializados en las tareas a desarrollar pues esto encarecería demasiado el precio. Sin embargo, si el sistema planteado funciona correctamente, nótese que realizar un sistema con equipamiento específico para la función para la producción en masa podría incluso abaratar costes

- Análisis en tiempo real: El procesamiento de imagen es en general una tarea que requiere no poca capacidad de cálculo. El dispositivo final sobre el que se instale el programa debe ser capaz de ejecutarlo en tiempo real. Esto será un pilar fundamental para la definición un criterio de eficacia

- Instalable en cualquier vehículo: Los equipos seleccionados para el prototipo del sistema deben ser compatibles con cualquier vehículo. Esto permitirá que se estudie el desarrollo en serie del producto diseñado para instalar en vehículos de forma *aftermarket*

Para tal fin, se plantean una serie de objetivos parciales y una hoja de ruta para llevar a cabo el proyecto:

- En primer lugar, hay que realizar una revisión de los algoritmos ya desarrollados para la detección de somnolencia mediante reconocimiento facial. El objetivo de esta fase es encontrar un proceso con un coste computacional bajo que permita reconocer expresiones faciales en tiempo real con dispositivos de bajo poder de cálculo. El resultado de esta primera fase debe ser una decisión sobre qué expresiones faciales se van a monitorizar (2.2), cómo se van a analizar (capítulo 2.3) y qué recursos se van a utilizar para el reconocimiento de imagen (capítulo 2.4)

- En segundo lugar, hay que definir unos criterios que permitan evaluar el éxito o fracaso del proyecto. En el capítulo 3.1 se desarrollarán los ya comentados criterios de validez, eficacia y bajo coste

- En tercer lugar, hay que elegir el hardware y software con el que se va a desarrollar el programa, así como definir el dispositivo portable de bajo coste sobre el que se volcará el código desarrollado (capítulo 3). Para esto, el propósito fundamental será mantener un coste bajo, por lo que se deberá estudiar la oferta de dispositivos y la compatibilidad entre el entorno de desarrollo y el producto final

- El siguiente paso es desarrollar el programa, evaluar el cumplimiento de los criterios de validez y eficacia establecidos, implementarlo en el dispositivo portable y de nuevo determinar el grado de satisfacción de los requisitos planteados. Para ello, se definirá un método de validación que permita evaluar si la detección de somnolencia se está haciendo correctamente y se considerará que es eficiente si permite realizar la tarea en tiempo real

- Finalmente, se procederá a montar el sistema final sobre un vehículo. Esto incluirá la adquisición de hardware necesario, la calibración de parámetros y equipo para optimizar el proceso y el diseño de un procedimiento de ensayo que permita evaluar la validez y utilidad del prototipo final

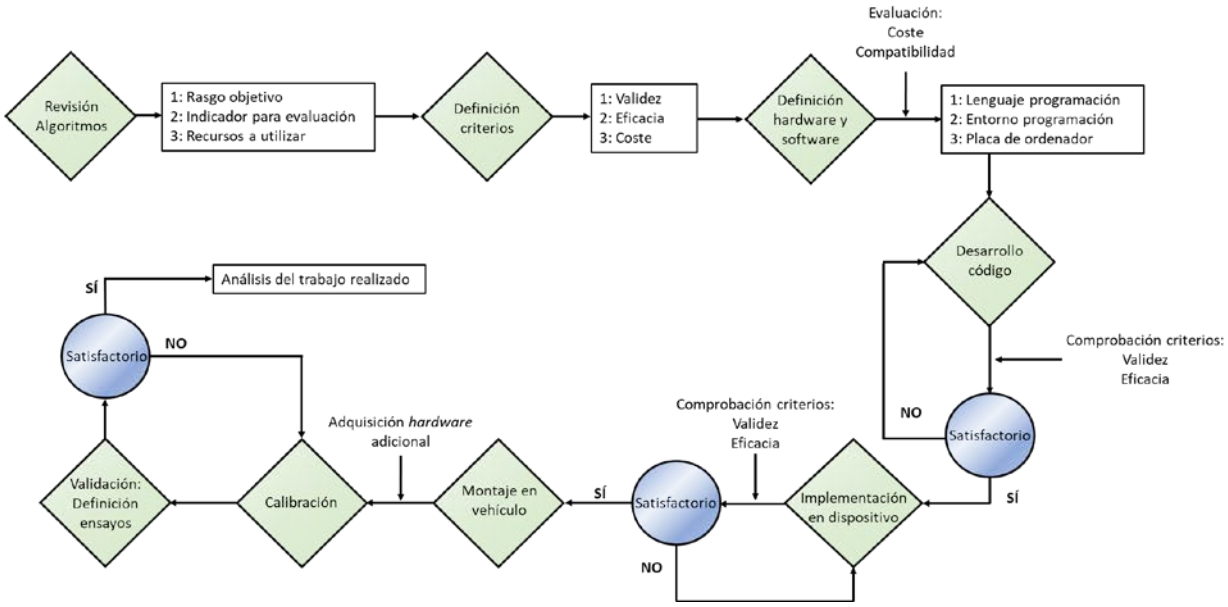

Figura 2. Diagrama de proceso del proyecto

1.4. ALCANCE

El alcance de este trabajo queda limitado a hacer un prototipo de un sistema para detectar la somnolencia en la conducción basado en reconocimiento facial que cumpla con los objetivos planteados.

Es importante hace hincapié en el término “prototipo”. Si bien los resultados de este trabajo pueden constituir una base muy válida para el desarrollo de un producto comercial, el objetivo en cuanto al equipo final no es desarrollar un producto que contemple la integración en el vehículo.

El montaje en el vehículo va a quedar limitado a la elección de los equipos necesarios (en base a la compatibilidad entre ellos y con el vehículo, el coste y el rendimiento) y a su presentación sobre en el automóvil. No se van a contemplar esfuerzos para una integración en la que se consideran cuestiones estéticas como que los equipos estén encapsulados u ocultos, se encuentren todos recogidos en una caja o el cableado quede recogido por los paneles, techo o salpicadero del vehículo: El alcance queda limitado a una presentación funcional.

En cuanto al objetivo de funcionalidad, se pretende desarrollar un equipo razonable partiendo de la base de que un objetivo fundamental es mantener un coste bajo. De esta forma el objetivo no es desarrollar un sistema que funcione bajo cualquier situación, sino que su funcionamiento quedará limitado a unas condiciones concretas las cuales se definirán más adelante una vez se tenga una noción clara sobre las posibilidades de la tecnología planteada.

2. MARCO TEÓRICO

De acuerdo con la hoja de ruta planteada, el primer paso es hacer una revisión de los mecanismos que permiten detectar somnolencia mediante la evaluación de los rasgos faciales. Para ello, primeramente hay que determinar cuáles son los rasgos objetivo y cómo manifiestan la fatiga. Seguidamente se deben definir unos indicadores sobre estos rasgos que permitan identificar un estado somnoliento. Finalmente, hay que hacer una investigación documental sobre las diversas técnicas que se emplean para realizar la tarea del reconocimiento facial.

Una vez concluidas estas fases de revisión bibliográfica, el siguiente paso es establecer unos criterios que delimiten las necesidades del proyecto y aplicarlos a la información recopilada. El resultado de esta operación debe ser la elección de un algoritmo para la detección de somnolencia.

2.1. SIGNOS Y SÍNTOMAS DE FATIGA

En su publicación “Otros factores de riesgo: La fatiga” [4] , la DGT indica de qué manera afecta la fatiga al conductor. A nivel psicológico describe efectos que se relacionan con las capacidades implicadas en la conducción como la alteración de la percepción y de los movimientos, un cambio en el comportamiento y el empeoramiento del procesamiento. Sin embargo, el objetivo en este estadio de la investigación es encontrar la forma en la que estos síntomas se exteriorizan, de forma que se puedan detectar visualmente.

2.1.1. Fatiga y visión

La visión puede volverse borrosa bajo los efectos de la fatiga, provocando una disminución de la agudeza visual. En casos extremos de cansancio, se dan incluso ilusiones ópticas como la percepción de brillos, sombras o deformaciones del entorno. Estos efectos, evidentemente, limitan las capacidades implicadas en la conducción, pero en lo que atañe a este trabajo, hay que centrarse en los signos de los ojos que revelan somnolencia.

La frecuencia de los parpadeos aumenta considerablemente con la somnolencia, así como el tiempo que permanecen cerrados en cada parpadeo. Esto hace que el periodo de parpadeo disminuya y que el porcentaje de este periodo durante el cual los ojos están abiertos se reduzca también.

Además, los ojos suelen quedar parcialmente cerrados en condiciones de cansancio. Cualquiera que haya experimentado somnolencia, no necesariamente en la conducción, ha sentido la pesadez de los párpados, como los ojos se le cierran y como mantenerlos abiertos supone un esfuerzo.

2.1.2. Fatiga y audición

Los sonidos fuertes y repentinos producen reacciones bruscas en un sujeto si se encuentra bajo los efectos de la somnolencia. Esto puede deberse en parte a la pérdida de sensibilidad auditiva: Ignorar parte de la información que se recibe del exterior hace que se sobre reaccione a los sucesos que ser perciben, que no tienen por qué ser necesariamente exagerados.

En este caso no hay signos físicos que indiquen somnolencia. Las orejas no presentan movimientos involuntarios que puedan monitorizarse, por lo que se cierra esta línea de investigación.

2.1.3. Fatiga y movimientos corporales

Pesadez en el cuerpo, migrañas, presión en las sienes, hormigueos, picores, calambres y/o dolores de nuca y de espalda pueden aparecer como consecuencia de la fatiga. Esto puede externalizarse en una serie de acciones algunas de las cuáles pueden monitorizarse de forma relativamente sencilla:

- El bostezo es el indicador de sueño por excelencia
- Cambios de postura y acomodos con frecuencia
- Estiramientos
- Cabeceos

2.2. DEFINICIÓN DE EXPRESIONES OBJETIVO

De la revisión bibliográfica anterior se han obtenido una serie de manifestaciones físicas producidas por el cansancio susceptibles de ser identificadas mediante reconocimiento de imagen:

- Frecuencia de parpadeo
- Tiempo durante el cual los ojos están cerrados
- Grado de apertura de los ojos
- Bostezos
- Cambios de postura y acomodos
- Estiramientos
- Cabeceos

De estas acciones, la gran mayoría son relativas al rostro por lo que bastaría con captar la imagen de la zona de la cara. Solo los cambios posturales y acomodos y los estiramientos implican otras partes del cuerpo, por lo que la estrategia a seguir para detectar estos actos implicaría tener un área de captación mayor y/o un mayor número de cámaras. Dado que este trabajo se ha enfocado en el reconocimiento de imagen enfocado a rasgos faciales, se van a proponer indicadores para manifestaciones del cansancio que se den en el rostro. De esta forma, cambios de postura y acomodos quedan fuera del alcance de este trabajo y también los cabeceos, que si bien bastaría una cámara que recogiera el área facial para captarlos, no son una expresión facial.

De esta manera, se plantean dos áreas faciales clave para detectar la somnolencia:

2.2.1. Boca

El bostezo es un acto involuntario que consiste en la apertura de la boca alcanzando una separación muy amplia de los maxilares. La apertura excesiva y una duración media de 7 segundos lo convierten en una expresión fácilmente reconocible. Además, va acompañado de gestos complementarios: Al bostezar, es corriente inclinar la cabeza hacia atrás y entornar los ojos. También se dan muchos otros sucesos, pero no son reconocibles mediante análisis visual.

Si bien ya se ha comentado que el cabeceo no entra dentro del ámbito de estudio de este proyecto, el análisis de los ojos sí que se va a llevar a cabo y su relación con el bostezo hace que pueda obtenerse más información y de mayor fiabilidad.

Una metodología de estudio del bostezo podría basarse en el grado y tiempo de apertura de la boca. Pese a que no todos los humanos presentan los mismos rasgos, es sencillo diferenciar un bostezo (boca abierta al máximo) de una boca cerrada, por lo que analizando morfológicamente la región de la boca mediante reconocimiento de imagen, puede extraerse la geometría en cada momento y detectar un bostezo.

2.2.2. Ojos

De la revisión anterior se han obtenido tres parámetros de estudio relativos a los ojos que pueden indicar somnolencia: El grado de apertura, la frecuencia de parpadeo y el tiempo que permanecen cerrados.

Cómo se ha dicho, uno de los objetivos del proyecto es que el programa sea eficiente. El reconocimiento de imagen implica procesos computacionalmente costosos, y si de un mismo análisis se pueden obtener 3 variables distintas parece que el ratio entre la información aportada y el tiempo de cálculo invertido es bastante beneficioso.

2.3. DEFINICIÓN DE INDICADORES

Una vez definidas las áreas faciales de interés que pueden delatar la somnolencia, hay que definir una serie de indicadores que nos permitan mediante el análisis de sus expresiones detectar la presencia de cansancio.

Tanto para los ojos como para la boca, se ha nombrado el grado de apertura. Un análisis visual que arroje la geometría de estas zonas podría servir para definir un indicador. En internet se encuentran recursos capaces de extraer las coordenadas faciales de una cara en una imagen, como por ejemplo Dlib, una potente herramienta que se podría considerar la base técnica de este proyecto y de la cual se hablará más adelante.

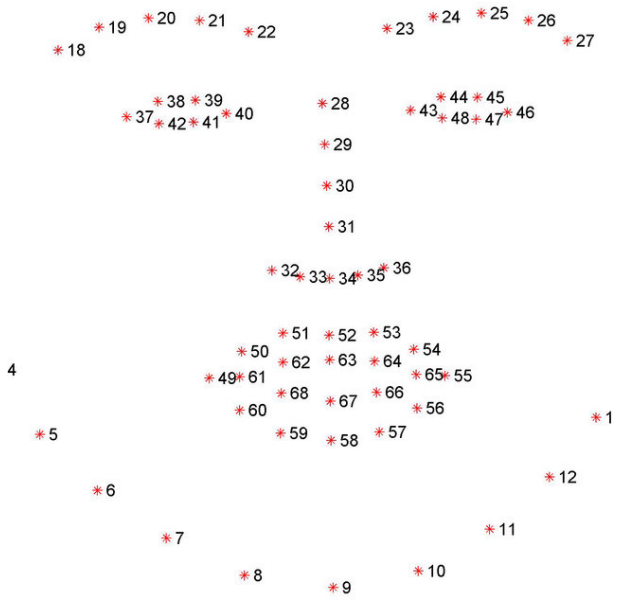

Figura 3. Visualización de las coordenadas faciales que extrae Dlib [11]

Como se puede ver en el mapa de coordenadas con el que trabaja Dlib, cada ojo queda definido por un conjunto de 6 coordenadas y la boca se representa con un set 20 puntos.

2.3.1. Indicador de bostezo

Haciendo una revisión en línea sobre indicadores que permitan discernir entre una boca abierta y una cerrada se puede encontrar el parámetro MAR (*Mouth Aspect Ratio*). Este parámetro es una relación entre el largo y el ancho de la boca:

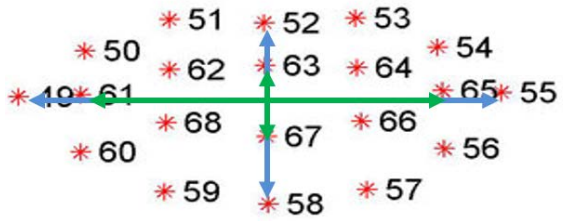

Figura 4. Extracto de las coordenadas de la boca de la Figura 3. Visualización de las coordenadas faciales que extrae Dlib

No todos los autores usan la misma fórmula para evaluar el grado de apertura de la boca. Tal y como muestra la Figura 4, el largo y el ancho de la boca se pueden definir con las coordenadas interiores o exteriores de los labios. Además, hay varios pares de coordenadas enfrentadas verticalmente y no hay una combinación de puntos que sea indiscutiblemente más significativa.

Así pues, en la literatura se encuentra el parámetro MAR calculado tanto con coordenadas interiores de los labios como exteriores y con más o menos pares de coordenadas según el criterio del autor. Para este proyecto se van a emplear los puntos que definen el contorno interior de los labios y se ha escogido este criterio arbitrariamente. Tal vez la única razón objetiva sin haber hecho un estudio con el que comparar la precisión del MAR según las coordenadas escogidas es que con esta configuración se coge un contorno completo de menos puntos que el contorno exterior.

La siguiente figura muestra los puntos escogidos:

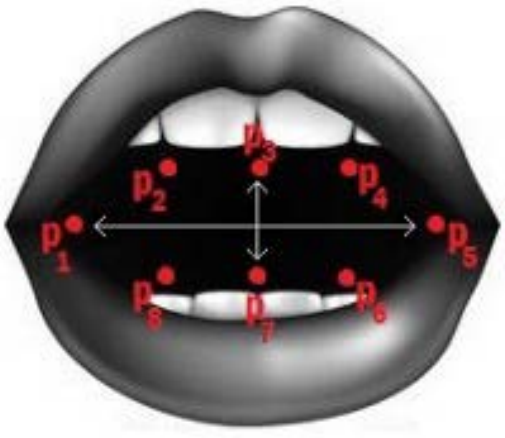

Figura 5. Información del grado de apertura de la boca [11]

Si se consigue detectar la boca en una imagen y extraer las coordenadas que se indican, se puede evaluar su apertura con la siguiente fórmula:

$$MAR = \frac{|p_2 - p_8| + |p_3 - p_7| + |p_4 - p_5|}{2 \cdot |p_1 - p_5|} \qquad (1)$$

2.3.2. Indicador del grado de apertura de los ojos

Para los ojos, el indicador de estudio más extendido es el EAR (*Eye Aspect Ratio*). Como se puede imaginar, su fundamento es el mismo que el del MAR y por ello el procedimiento de análisis es semejante. Extrayendo las coordenadas de los ojos con Dlib se obtiene un contorno de 6 puntos de cada ojo tal y como se muestra en la siguiente figura:

Figura 6. Extracto de las coordenadas de los ojos de la Figura 3

A diferencia que con el MAR, las publicaciones que tratan del parámetro EAR convergen hacia una misma solución. Al haber un contorno único formado por tan solo 6 puntos, este indicador se calcula generalmente de la siguiente manera:

$$EAR = \frac{|p_2 - p_6| + |p_3 - p_5|}{2 \cdot |p_1 - p_4|} \qquad (2)$$

Empleando este mapa de puntos:

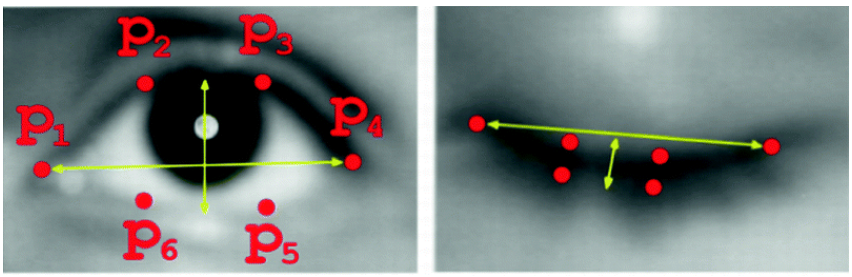

Figura 7. Representación de las coordenadas para el EAR con los ojos cerrados y abiertos [12]

2.4. RECONOCIMIENTO DE IMAGEN

El prototipo que se va a diseñar en este proyecto, a diferencia de los sistemas de detección de somnolencia presentes en el mercado, va a emplear técnicas modernas de reconocimiento de imagen para caracterizar el estado del conductor.

Para examinar las expresiones de ojos y boca descritas en apartado anterior se requiere de técnicas de procesamiento de imagen que arrojen información sobre facciones y expresiones del rostro. Por un lado, hay que abordar la detección facial, que en este caso no consiste en identificar a un sujeto concreto analizando sus facciones sino simplemente en detectar un rostro sea cual sea en la imagen. Esta tarea es llevada a cabo por un clasificador. Por otro lado, hay que realizar la extracción de características que permitan llevar a cabo el análisis planteado en el capítulo anterior.

Los indicadores propuestos se basan en una serie de coordenadas faciales que hay que se extraen de una imagen de un rostro. La forma en la que se extraen estas coordenadas se verá más adelante cuando se expongan las herramientas empleadas. Previamente, para comprender la metodología que se va a seguir y los recursos que se van a emplear, es conveniente realizar una breve explicación del fundamento de la visión artificial.

El análisis digital de imágenes consiste en extraer información contenida en una imagen. Es una rama de la inteligencia artificial que permite a partir entradas como imágenes, vídeos u otros obtener una salida como una acción, decisión, recomendación… Desde un punto de vista ingenieril, se trata de dotar a una máquina de la capacidad del sentido de la vista de los humanos, así como de la capacidad de comprensión de la imagen, de forma que se puedan realizar tareas de análisis visual automáticamente. En una industria, el reconocimiento de imagen se podría emplear para reconocer piezas defectuosas, verificar montajes, tareas de metrología, etc. En este proyecto, el reconocimiento va a ir dirigido a evaluar rasgos faciales y el *output* del análisis va a ser una recomendación (en ningún momento se va a actuar sobre el vehículo, ya sea moviendo el volante, activando los frenos, etc.).

Como se ha explicado, para extraer las coordenadas faciales presentadas en el apartado anterior es necesario haber procesado previamente la imagen con un clasificador de caras de forma que se identifique un rostro. Esta detección puede afrontarse de distintas formas y a continuación se van a presentar las más extendidas que podrían plantearse para el desarrollo de este proyecto.

2.4.1. Detección facial con cascadas de Haar

La detección de objetos mediante clasificadores en cascada basados en funciones de Haar es un método ampliamente usado para tareas de reconocimiento facial debido a su eficacia. El algoritmo descrito por Paul Viola y Michael Jones en [13] presenta un bajo coste computacional que permite que sea empleado en tiempo real.

La base de este algoritmo son las características de Haar, funciones rectangulares planas simples con distintos tamaños y combinaciones de cuadriláteros blancos y negros. La siguiente imagen muestra algunos ejemplos de características o funciones de Haar:

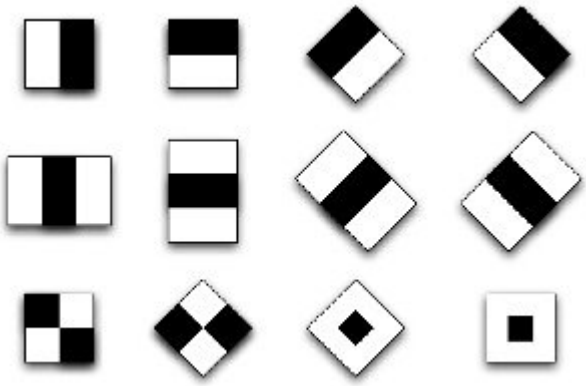

Figura 8. Ejemplo de características de Haar

Estas funciones o características están definidas y pueden encontrarse en internet en formato XML. Hay cuatro tipologías básicas que pueden observarse en la imagen: En la primera línea, vemos características de borde; en la segunda, características lineales; en la tercera, las dos primeras características son de tipo central diagonal y las dos últimas de tipo central. Cada característica tiene un valor, que es el resultado de restar los píxeles del área negra a los píxeles del área blanca. Las funciones también pueden clasificarse por el número de sub-cuadrados por los que están compuestas. Hay funciones formadas por dos, tres y hasta cuatro cuadrados.

La composición en blanco y negro de las características permite compararlas con imágenes en blanco y negro donde el nivel de gris se traduce en nivel de luminosidad. Por lo tanto, el algoritmo trabaja con imágenes en escala de grises en las que busca diferencias de intensidades luminosas entre regiones rectangulares adyacentes. Puesto que el propósito de esta primera fase del reconocimiento de imagen se limita a la detección de una cara en una imagen y no a la identificación de la persona, basta con contemplar propiedades que comparten los rostros humanos. Algunas características comunes pueden ser:

- La ubicación y tamaño de la boca, el puente de la nariz y los ojos
- La región de los ojos es más oscura (menos luminosa) que la parte superior de las mejillas
- El puente de la nariz es más luminoso que los ojos
- Las cejas son oscuras
- Los dientes son claros y los labios oscuros

Sabiendo que estas funciones reflejan cambios de gris en la imagen y que los rostros por lo general comparten ciertas características, se pueden buscar ciertos patrones en la imagen que revelen la presencia de un rostro. Por ejemplo, considerando las características que presentan los rostros expuestas anteriormente y funciones del tipo lineal para detectar la nariz y de tipo borde para detectar la cejas y la boca, se puede recorrer una imagen en busca de un alto nivel de coincidencia.

Figura 9. Ejemplo de aplicación de características de Haar a un rostro. Modificado de [14]

Nótese que hay muchas otras características de Haar aplicables. También se podría haber usado una característica lineal para identificar una sonrisa (dientes blancos, labios negros) o para la boca en su conjunto (los labios son más oscuros que el bozo y que el hueco del mentón), características de borde para la nariz (comparando solo la nariz que presenta más luminosidad que una mejilla), etc.

El algoritmo toma imágenes en escala de grises y la analiza comparando características de Haar de 24x24 píxeles o menores con cada región de 24x24 píxeles que se pueda encontrar en la imagen. Como se ha comentado, el algoritmo trabaja con imágenes en escala de grises por lo que la conversión de RGB a blanco y negro sería un paso de preprocesamiento necesario. Una vez hecho esto, se debe tener en cuenta que, en una imagen en blanco y negro, a diferencia de lo que sucede en una función de Haar, los píxeles no presentan estrictamente un valor binario (blanco o negro) sino que presentan valores intermedios. Sea una imagen de 4x6 píxeles codificada con n = 8 bits ($2^8 - 1 = 255$ posibles valores, 0 representa el negro o la ausencia de color y 255 el blanco o la máxima luminosidad):

255	255	0	0
255	255	0	0
255	255	0	0
255	255	0	0
255	255	0	0
255	255	0	0

224	255	160	96
224	224	128	64
192	192	96	32
255	224	128	64
224	255	160	32
192	192	96	0

Figura 10. Caso ideal para la comparación con características de Haar (izquierda), caso real (derecha)

Si la imagen izquierda de la figura anterior se comparara con la primera característica mostrada en la Figura 8 (esquina superior izquierda) daría como resultado un 100% de coincidencia. Ahora bien, el trabajar con imágenes no ideales complica las cosas. La comparación de una región de una imagen en

escala de grises (como la imagen derecha de la figura anterior) con una característica de Haar se realiza de la siguiente forma:

1. El primer paso es establecer un umbral de luminosidad para sortear el problema de que la escala de grises presenta muchos más valores que blanco o negro absoluto. Así, ajustando el límite por ejemplo en 160, se tendría que todos los valores inferiores serían considerados como negro y los superiores como blanco, obteniendo una imagen con valores de píxel binarios:

Figura 11. Ejemplo de umbral para la discriminación entre blanco y negro

Esto aplicado a una imagen real en la que los valores de píxeles adyacentes pueden diferir en simplemente una unidad, podría resultar en esta situación:

224	192	160	96
224	224	192	64
160	192	96	96
160	224	128	64
224	192	160	64
192	128	192	32

255	255	0	0
255	255	255	0
0	255	0	0
0	255	0	0
255	255	0	0
255	0	255	0

Figura 12. Categorización de píxeles en blanco y negro según su valor y el valor umbral definido

2. El segundo paso consiste en asignar un valor binario a cada píxel. Por convención, se asignan los valores de 1 para la ausencia de luminosidad (píxel negro) y 0 para los píxeles blancos:

255	255	0	0
255	255	255	0
0	255	0	0
0	255	0	0
255	255	0	0
255	0	255	0

0	0	1	1
0	0	0	1
1	0	1	1
1	0	1	1
0	0	1	1
0	1	0	1

Figura 13. Asignación de valores binarios a los píxeles

3. Una vez se han categorizado los píxeles en blanco o negro y se les ha asignado un valor binario, comienza la comparación con las características de Haar. Para ello, hay que comparar el valor asignado de una característica de Haar con el valor de la imagen en estudio. Las características de Haar tienen un valor de 1, resultado de restarle a la media de los valores de los píxeles en el área negra el valor de la media de los píxeles en el área blanca:

$$\Delta = \frac{1}{n} \cdot \sum_{negro}^{n} f(n) - \frac{1}{b} \cdot \sum_{blanco}^{b} f(b) \qquad (3)$$

Según la fórmula anterior y la Figura 13, asignando a *"f"* de un píxel negro el valor de 1, dándole el valor de 0 a un píxel blanco y habiendo "*n*" píxeles negros y "*b*" píxeles blancos en una función de Haar, se puede comprobar que el valor asignado a una característica de Haar es siempre 1.

En una imagen real, esto difiere. En una zona mayormente oscura, habrá píxeles que sean más claros y por ello hayan sido clasificados como blancos según el paso 1. Los mismo sucede a la inversa. Esto es lo que se ha intentado mostrar en la Figura 12. Por ello, aplicando la fórmula 3 a la Figura 13 en la que los píxeles luminosos o blancos valen 0 y los píxeles con ausencia de color o negros valen 1 se obtiene el siguiente valor:

$$\Delta = \frac{1}{12} \cdot \sum_{negro}^{12} f(n) - \frac{1}{12} \cdot \sum_{blanco}^{12} f(b) = \frac{1 \cdot 10 + 0 \cdot 2}{12} - \frac{1 \cdot 3 + 0 \cdot 9}{12} = 0{,}5833$$

Esto supone una coincidencia del 58,33 %. A partir de este valor y del umbral de coincidencia que se haya fijado previamente, se determinará o no la presencia de una característica de Haar.

Este método por si solo presenta algunos problemas. Para tamaños de 24x24 píxeles, hay cerca de 160.000 características de Haar (menos de 1x1 píxel, las funciones de Haar pueden ser de muchos tamaños). Buscar en una imagen 160.000 características en cada área posible de 24x24 píxeles conlleva un coste computacional muy elevado. Por ello, el algoritmo Viola-Jones se sirve de una ayuda fundamental: la imagen integral. La imagen integral es un preprocesamiento tal y cómo sería el paso de la imagen a escala de grises. Consiste en asignar a cada píxel la suma de los valores de los píxeles que se encuentren por encima de él y a su izquierda de la siguiente forma:

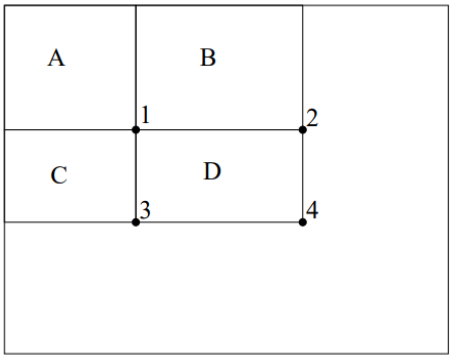

Figura 14. Conjuntos de píxeles A, B, C y D en una imagen (obtenido de [13])

En la imagen integral de la imagen anterior, el píxel 1 contendría la suma de los valores de los píxeles contenidos en el rectángulo A de la imagen original; el píxel de la imagen integral 2 tendría el valor de los píxeles contenidos en los rectángulos A y B de la imagen original; el píxel 4, contendría los valores

contenidos en A, B, C y D de la imagen original; el píxel 3 contendría los valores contenidos en los recuadros A y C de la imagen original.

La imagen integral permite un cálculo más rápido de los valores de una región de la imagen según la ecuación 3 al no tener que realizar una suma de los valores que contiene cada píxel de la región para obtener su valor. De esta forma, se realizan una serie de sumas una vez para obtener la imagen integral, pero luego cualquier región se puede obtener como resultado de 4 operaciones. Sea una imagen en blanco y negro de 24 píxeles como la de la imagen derecha de la Figura 12. A continuación, se van a exponer unos ejemplos de cómo obtener con tan solo 4 operaciones el valor de cualquier sub-imagen:

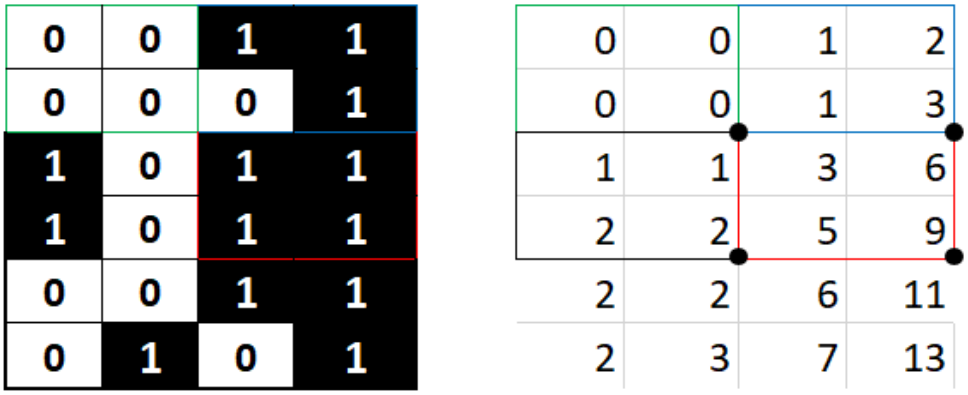

Figura 15. Ejemplo 1: Imagen original e imagen integral con 4 regiones arbitrarias indicadas

Pongamos que se quiere conocer el valor del área encerrada en el rectángulo rojo. Basta con hacer operaciones con los píxeles inferiores derechos de la propia región y de las regiones adyacentes en la imagen integral: Rojo + verde – azul – negro = 9 + 0 – 3 – 2 = 4, que es el valor que tiene el área roja en la imagen original (1 · 4). Para obtener el valor del área azul, habría que restarle el píxel de la esquina inferior derecha del área verde al de la azul, y para obtener el valor del área negra habría que restarle el valor del píxel inferior derecho del área verde al del área negra. Finalmente, el valor del área verde no requiere de operaciones. Esto puede hacerse para cualquier región, siempre y cuando las 4 regiones formen un rectángulo cerrado que incluya el píxel superior izquierdo:

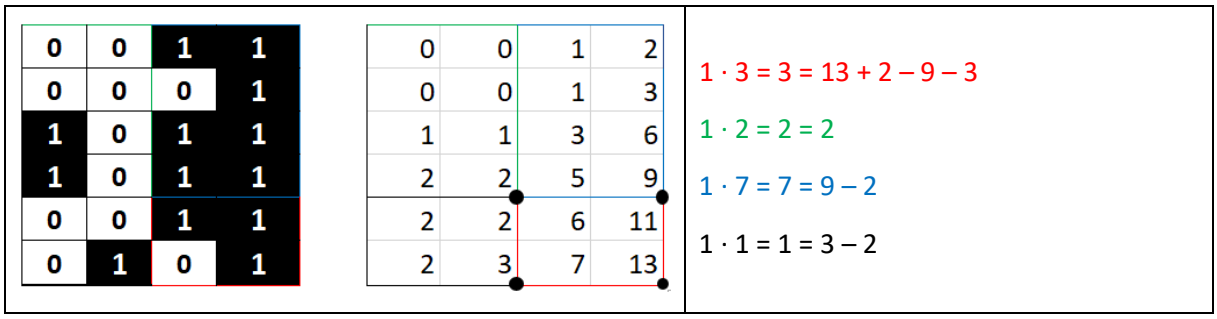

Figura 16. Ejemplo 2 de cálculo de valores con imagen integral

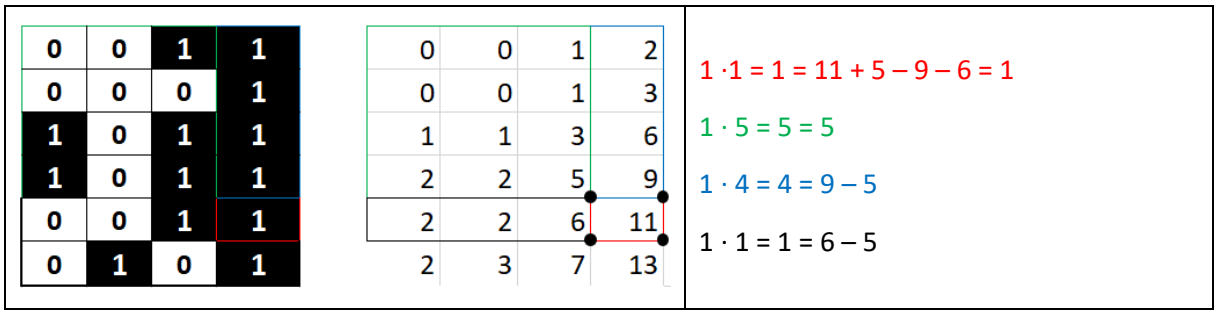

Figura 17. Ejemplo 3 de cálculo de valores con imagen integral

A modo de resumen de los conceptos de valor de una región de píxeles, extracción de una característica de Haar e imagen integral, a continuación, se va a plantear un ejemplo completo. Imaginemos que se quiere realizar la detección del rostro de la Gioconda mediante un clasificador de Haar. El primer paso, sería hacer un preprocesamiento de la imagen que la convirtiera a escala de grises, para posteriormente aplicar mediante un umbral de luminosidad previamente definido un filtro que de un valor de color binario a los píxeles: o blanco o negro.

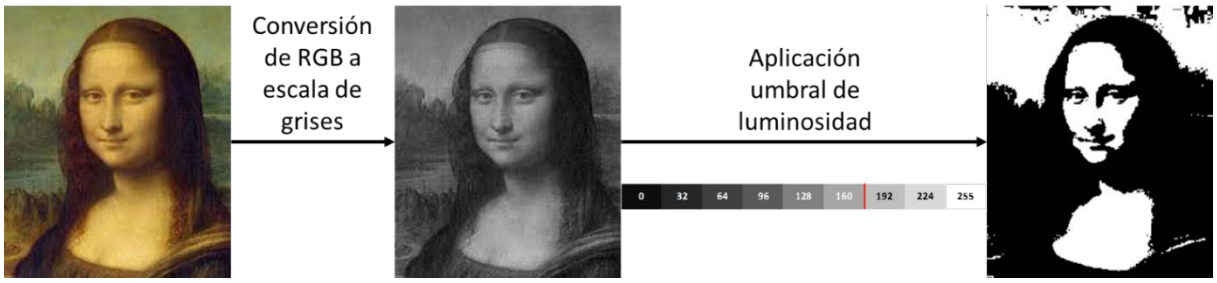

Figura 18. Preprocesamiento de la imagen para extraer características de Haar

A continuación, se deben extraer una serie de características de Haar en una serie de regiones posicionadas relativamente entre ellas de una forma más o menos definida (la estructura facial siempre es la misma: ojos arriba, nariz en medio, boca bajo…). La siguiente imagen ejemplifica como sería el proceso de detectar una función de Haar de borde horizontal con un clasificador que trabaje solamente con esta característica:

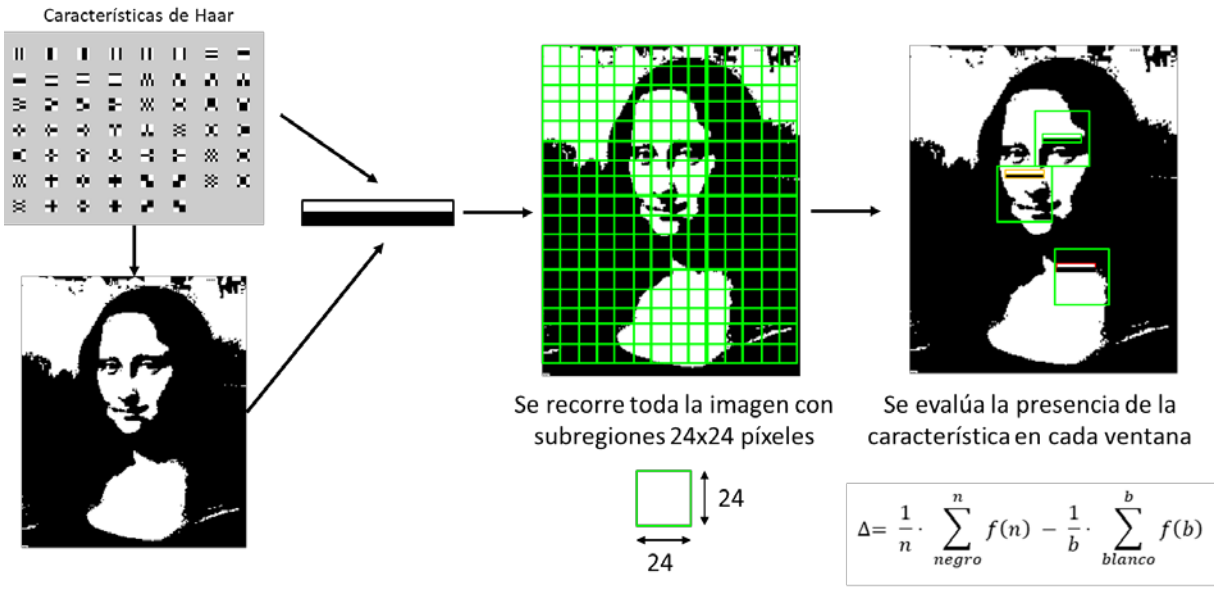

Figura 19. Extracción de características de Haar

Es deseable extraer esta característica en zonas como los labios y la ceja. En la imagen superior se ha representado como se han cogido todas las ventanas de 24x24 píxeles presentes en la imagen y se ha buscado en cada una las caracteríticas de Haar que contempla el clasificador en cuestión (en este caso, solamente contempla una característica de borde). El proceso arroja una coincidencia del 100% con la ceja, del 0% en la mejilla (es una zona totalmente blanca, por lo que coincide la mitad de la característica de Haar) y del -50% en la zona del cuello (hay 0 píxeles negros en el sub-cuadrado negro de la característica y 0 píxeles blancos en el sub-cuadrado blando de la característica).

Para evaluar la coincidencia, se hace uso de la imagen integral. De esta forma, se acelera el proceso tal y cómo se ha explicado anteriormente. Por ejemplo, para conocer el valor de la siguiente región roja en la imagen en blanco y negro:

Figura 20. Extracción del valor de la región roja (ceja) mediante la imagen integral

Habría que sumar los píxeles rojo y verde en la imagen integral relativa a la imagen superior y restarle los valores de los píxeles azul y negro también de la imagen integral. De esta forma, se obtiene con 4

operaciones el valor de esta región, que requeriría de más sumas para obtenerse a través de los píxeles de la imagen de partida. Además, se podría obtener el valor de cualquier región con 4 o menos operaciones. Teniendo en cuenta que se van a analizar todas las regiones posibles de la imagen del tamaño de la característica de Haar, la imagen integral supone un importante ahorro computacional. Nótese que no se ha representado la imagen integral puesto que visualmente carece de sentido (aunque computacionalmente es muy útil).

La imagen integral acelera entonces el proceso de calcular el valor de cada región de la imagen. Sin embargo, para cada región de 24x24 píxeles sigue habiendo unas 160.000 características posibles, por lo que el coste computacional de evaluarlas todas sigue siendo muy elevado. En este punto es donde aparecen los clasificadores. En su artículo sobre la detección de objetos, Paul Viola y Michael Jones explican que entrenan un total de 38 clasificadores para luego usarlos en cascada. Cada clasificador se entrena con 4916 imágenes positivas de rostros (9832 incluyendo la imagen espejo) y 9544 imágenes negativas. El primer clasificador de la cascada emplea simplemente una característica de Haar; el segundo, diez; el tercero, 25 y los clasificadores sucesivos cuentan con cada vez más características. Evidentemente, los clasificadores con más características alcanzarán mayores ratios de detección y menores ratios de falsos positivos, requiriendo también más tiempo de computación.

La estrategia de usar clasificadores en cascada es muy útil ya que, en caso de no haber un rostro, la imagen no suele superar el segundo clasificador, de forma que solamente se llegan a buscar unas 10 características en la imagen. Cada fase de la cascada añade más características de Haar, reduciendo el ratio de falsos positivos así como el ratio de detección. Si una imagen supera los 38 clasificadores, se considera que hay un rostro en ella.

2.4.2. Detección facial con HOG

La detección mediante Histograma de Gradientes Orientados se basa en la idea de que la apariencia y forma de una cara puede ser caracterizada a menudo por una distribución de gradientes de intensidades y direcciones de los bordes. Para detectar objetos, la mayoría de las veces es más importante tener en cuenta las formas que el color. Por eso, al igual que la detección con clasificadores de Haar, este método trabaja con imágenes en escala de grises, por lo que los gradientes de intensidades se refieren a luminosidad.

Como su propio nombre indica, este método se vale del histograma de gradientes de una imagen, entendiendo histograma como la representación gráfica de la distribución tonal de la imagen. La siguiente imagen representa el histograma de la Gioconda en escala de grises y en blanco y negro, es decir, el número de píxeles para cada grado de luminosidad comprendido entre 0 (negro) y 255 (blanco) para imágenes codificadas en 8 bits.

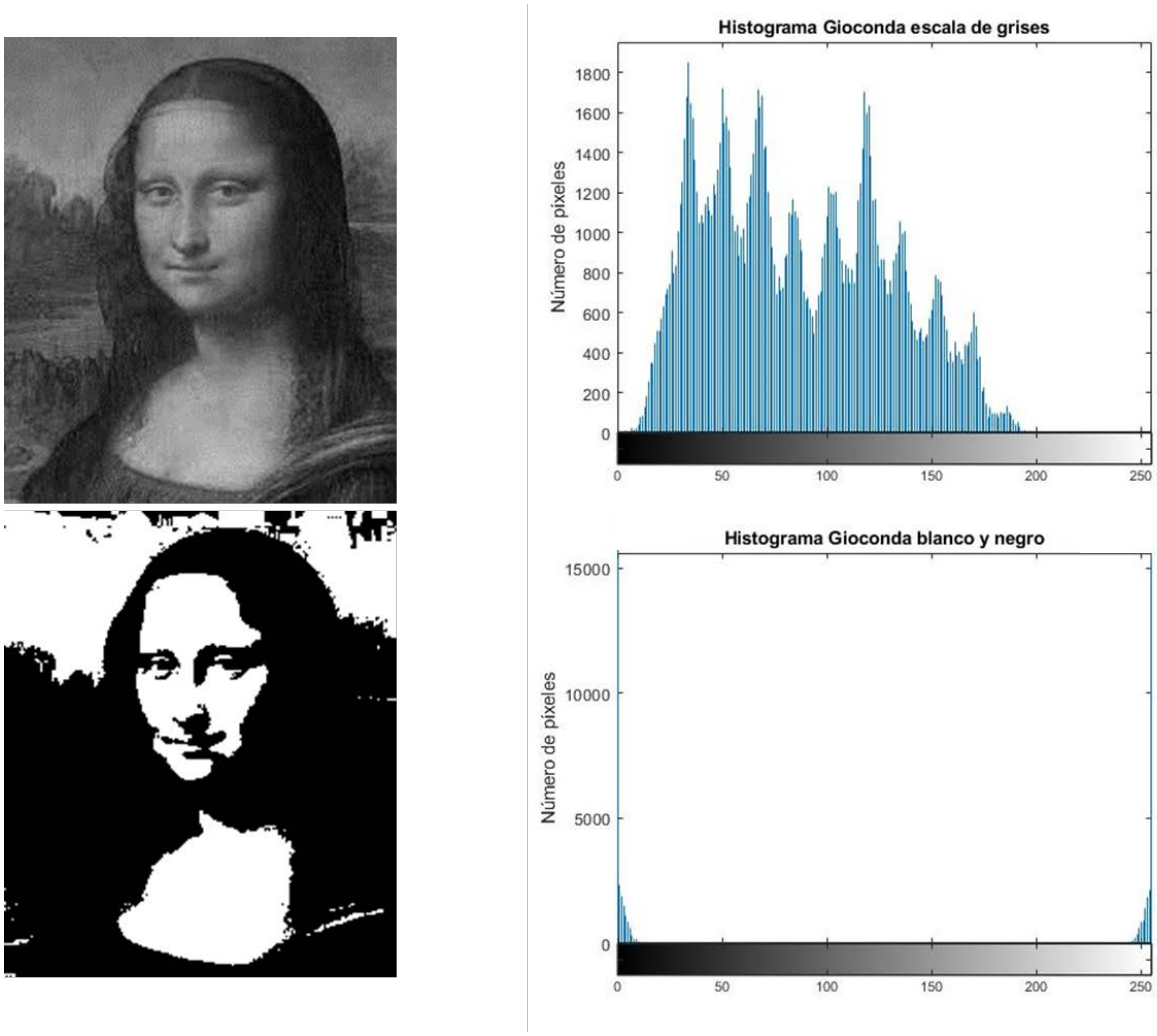

Figura 21. Histogramas de una imagen de la Gioconda en escala de grises y en blanco y negro

Como se puede observar, la distribución de luminosidad en el histograma de la imagen en blanco y negro es prácticamente binaria mientras que el histograma de la imagen en escala de grises presenta una distribución mucho más repartida a lo largo de todo el rango de luminosidad. El que la imagen en blanco y negro no sea totalmente binaria se debe a haber realizado una transformación a blanco y negro con un programa no especializado, que ha dado a algunos píxeles valores distintos de 0 o 255. Los histogramas se han obtenido con MATLAB.

El histograma de gradientes orientados va un paso más allá. Siendo el gradiente un valor vectorial (por lo tanto se trabaja con dos valores: módulo y dirección) que representa la variación de una magnitud (luminosidad) en función de la distancia (en píxeles), en zonas con un alto cambio de luminosidad como son los bordes habrá gradientes elevados. La siguiente imagen muestra una representación del histograma de gradientes orientados de la imagen la Gioconda hecha con MATLAB. Se ha representado el histograma para la imagen en escala de grises y para la imagen en blanco y negro:

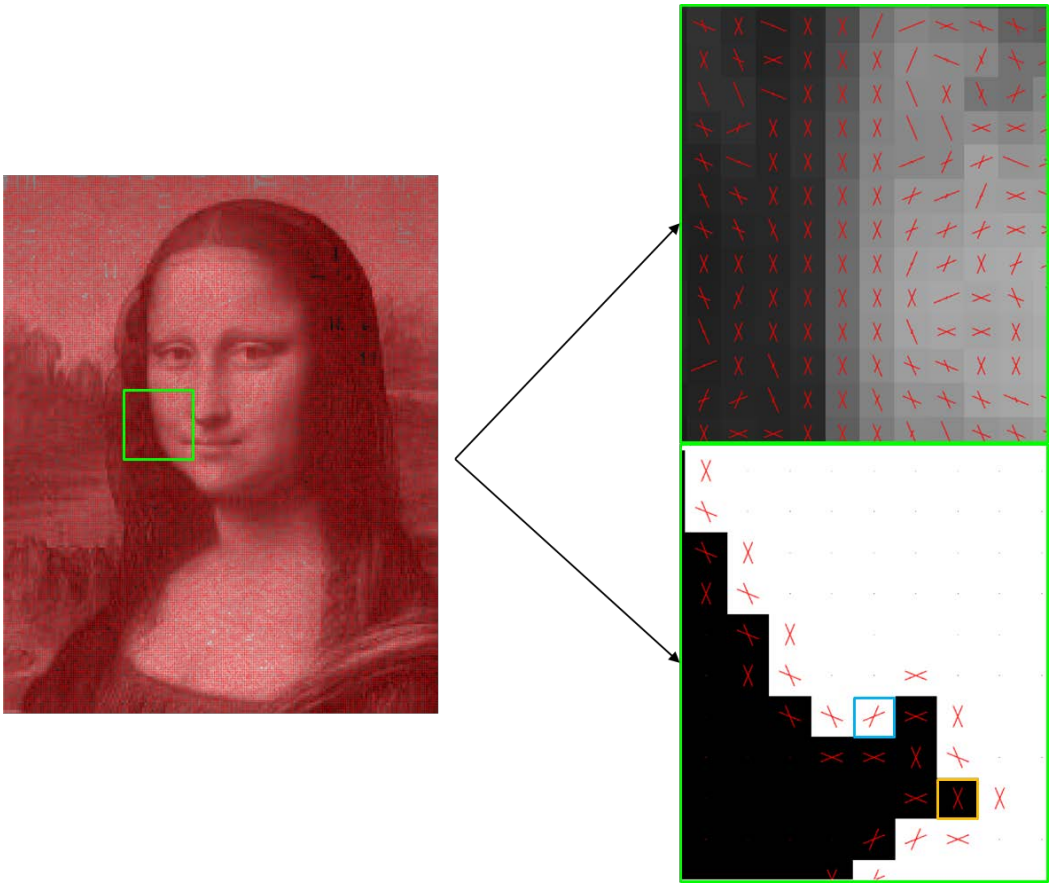

Figura 22. Histograma de gradientes orientados de la Gioconda. Detalle en escala de grises y en blanco y negro

Para obtener la figura anterior se ha realizado el gradiente de cada píxel considerando 4 píxeles vecinos en forma de cruz, es decir, uno arriba, uno bajo, uno a la izquierda y uno a la derecha. En el detalle en escala de grises se puede ver que cada cruz toma una dirección en función de la diferencia de luminosidad con los píxeles vecinos. En el detalle en blanco y negro se puede ver que, al ser las diferencias de luminosidad un "todo o nada", por trigonometría básica la dirección de los gradientes toma valores entre 0° y 360° espaciados 45°: 0°, 45°, 90°, 135°, etc.

Para obtener los gradientes representados en la imagen anterior se hace el siguiente cálculo:

$$\nabla f = \begin{pmatrix} f(x+1,y) - f(x-1,y) \\ f(x,y+1) - f(x,y-1) \end{pmatrix} \qquad (4)$$

A modo de ejemplo, considerando los valores de 1 (negro) y 0 (blanco) por coherencia con los valores asignados en el capítulo 2.4, para el píxel indicado con el borde naranja el gradiente es

$$\nabla f = \begin{pmatrix} f(x+1,y) - f(x-1,y) \\ f(x,y+1) - f(x,y-1) \end{pmatrix} = \begin{pmatrix} 0-1 \\ 0-0 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} = 1_{|180°}$$

Y para el píxel indicado con el borde azul el gradiente sería

$$\nabla f = \begin{pmatrix} f(x+1,y) - f(x-1,y) \\ f(x,y+1) - f(x,y-1) \end{pmatrix} = \begin{pmatrix} 1-0 \\ 0-1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \sqrt{2}_{|-45°}$$

Los cálculos que se han mostrado son un ejemplo básico de cómo calcular el histograma. Se han empleado 4 píxeles vecinos y se ha calculado el gradiente para cada píxel de la imagen en blanco y negro (binario). Sin embargo, realizar este cálculo para cada píxel es lento. Es por ello por lo que este método de detección de caras hace uso de unas máscaras conocidas como *kernels* o matrices de convolución. Estas máscaras son matrices que aplicadas sobre la imagen hacen el efecto de un filtro. En función de lo que se quiera detectar se suelen emplear unos *kernels* u otros. En la detección facial, tiene gran importancia la detección de bordes, por lo que se usan *kernels* preparados para esto. Por ejemplo, para calcular los gradientes tal y como se ha hecho en la Figura 22, los *kernels* a aplicar serían los siguientes:

$$G_x = [-1 \quad 0 \quad 1] \;\; ; \;\; G_y = [-1 \quad 0 \quad 1]^T$$

Como se ha comentado, este es un ejemplo muy simple de obtención de HOG en el que se han usado 4 píxeles en forma de cruz alrededor del píxel sobre el cual se quiere calcular el gradiente. Si se desea mayor precisión, se pueden emplear más píxeles vecinos, por ejemplo, un conjunto de 3x3 píxeles alrededor del píxel sobre el cual se quiere calcular el gradiente. En este caso, algunos *kernels* muy usados en algoritmos de detección de bordes son el operador de Prewitt y el operador de Sobel:

$$\mathbf{G_x} = \begin{bmatrix} +1 & 0 & -1 \\ +1 & 0 & -1 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * \mathbf{A}$$

Figura 23. Operador de Prewitt y aplicación a una imagen para la detección de bordes [15]

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Figura 24. Operador de Sobel y aplicación a una imagen para la detección de bordes [16]

Habiendo explicado los conceptos de histograma de una imagen, histograma de gradientes de una imagen y *kernel*, a continuación se va a detallar la metodología del algoritmo que realiza la detección facial mediante HOG.

En primer lugar, partiendo ya de la imagen en escala de grises, se obtiene tal y como se ha explicado la matriz de módulos y la de ángulos que contiene la información relativa a cada píxel de la imagen. Para ello, se hace uso de los *kernels* y se obtiene una matriz de módulos y otra de ángulos del tamaño de la imagen.

Seguidamente hay que definir una serie de parámetros con los que el algoritmo trabajará:

- Se define un tamaño de "célula", una sub-ventana que contendrá *n* x *n* píxeles de forma similar a lo que se muestra en la Figura 19
- Se define un tamaño de "bloque" a partir de un número de células, es decir, el tamaño del bloque es de *i* x *i* células
- Se define un factor de solapamiento que determina cuál es el solapamiento que puede haber entre bloques en la imagen
- Se define "*j*", un número posible de ángulos para discretizar los valores de orientación. En algunas aplicaciones es habitual trabajar con ángulos sin signo (de 0° a 180°), pues se reduce el coste computacional y la detección puede ser incluso mejor que con ángulos con signo en la detección facial [17]

Con el tamaño de bloque, se computa el vector de histograma de cada célula que contiene. Se presenta un ejemplo a continuación:

> Supongamos que se configura el algoritmo para que recorra la imagen con sub-ventanas de 8x8 píxeles y clasifique los ángulos sin signo en intervalos de 20° (*j* = 9 orientaciones posibles). El proceso de cálculo de dicho vector a partir del módulo y ángulo obtenido para cada píxel de una imagen cualquiera sería el siguiente:

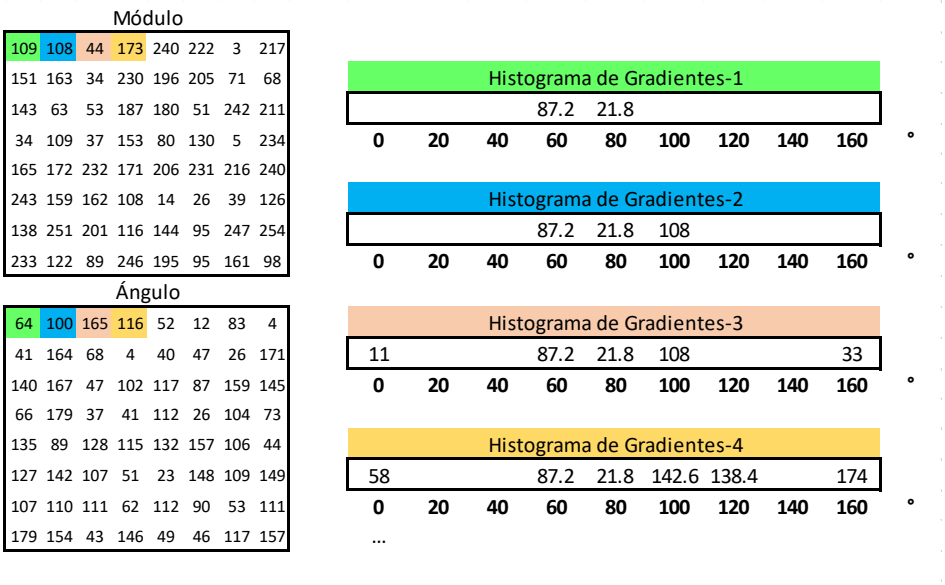

Figura 25. Obtención del vector de gradientes de una sub-ventana 8 x 8 de una imagen a partir del histograma de gradientes orientados

Cómo se ve, para cada ángulo (posición) del vector se van sumando las contribuciones de cada píxel, de forma que, si la orientación del gradiente en un píxel queda entre dos de los ángulos definidos del vector, su módulo se reparte proporcionalmente. Gradientes más fuertes contribuyen más a una determinada orientación, mientras que gradientes pequeños tienen un peso menor.

Una vez se tiene el vector de la célula, se repite la operación para todas las células dentro de un mismo bloque y se concatenan los vectores obtenidos, obteniendo un vector de 1 x ($i^2 \cdot j$). Se procede a efectuar una etapa de normalización para reducir el impacto de las sombras y variaciones de luz, eliminando los efectos de las diferencias locales de luminosidad.

Seguidamente, se concatenan los vectores normalizados correspondientes a todos los bloques de la imagen para obtener un único vector con toda la información. Para poder visualizar el correcto funcionamiento de la detección de bordes, a continuación se va a mostrar la representación del histograma de gradientes orientados de la Gioconda igual que se hacía en la Figura 22 pero configurando el algoritmo con 9 orientaciones células de 8 x 8 píxeles:

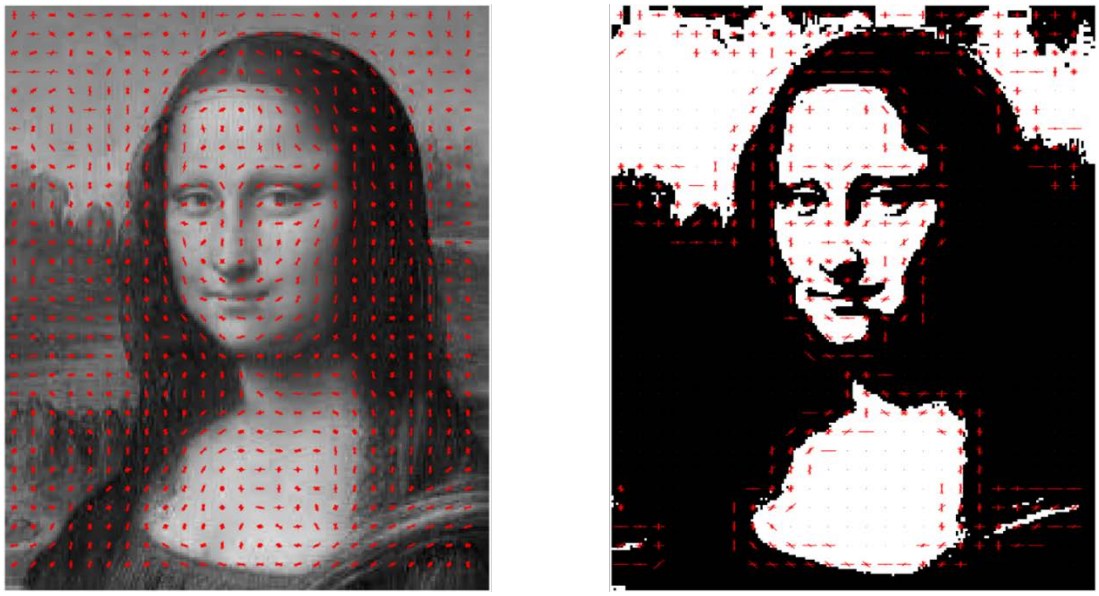

Figura 26. Representación HOG realizda con MATLAB, definiendo 9 posibles orienteaciones, células de 8 x 8 píxeles, bloques de 2 x 2 células y solapamiento del 50%

Como se puede observar, los bordes quedan bien capturados por el algoritmo.

La idea de emplear HOG es transformar la cantidad de información disponible en la imagen en un volumen más pequeño y que aporte los datos importantes. El vector obtenido no aporta información visual con la que podamos ver la imagen a través de sus datos, pero estos son muy valiosos al combinarlos con un algoritmo de clasificación de imágenes SVM (*Support vector Machine*). Al entrenar un clasificador con algoritmo SVM se emplea un conjunto de imágenes positivas (de rostros) y un conjunto de imágenes negativas (no contienen rostros), y lo que se hace es pasarle los vectores de características extraídos según el método expuesto. El clasificador puede entrenarse con rostros de frente, de perfil, desde arriba… Para el caso de la detección facial en la conducción según la estrategia que se está planteando basta con poder reconocer un rostro frontalmente.

2.4.3. Comparativa

El primer método expuesto para la detección de caras (clasificadores de Haar en cascada) se puede considerar robusto, aunque hay otras opciones con menor tasa de fallo. Su punto fuerte es la velocidad, pudiendo detectar caras a razón de 15 *frames* por segundo [13], trabajando casi en tiempo real. Este punto fuerte tiene un gran peso en este proyecto, pues el objetivo es crear un programa capaz de detectar la somnolencia e instalarlo en un dispositivo portable y barato (consecuentemente, de baja potencia). Por lo tanto, el uso de este método puede ser una ventaja para cumplir el objetivo planteado de eficacia.

Su punto débil son los falsos positivos. Para otros proyectos, el hecho de tener unos ratios de falsos positivos más elevados que otros métodos podría constituir una gran debilidad que supusiera su descarte. Sin embargo, en este proyecto esta debilidad apenas supone un problema, pues al obtener una imagen frontal, limpia y estable de un rostro la cascada de clasificadores parte de una situación muy favorable. Además, esto sortea otro de los problemas que presenta este método, y es que el clasificador se entrena para reconocer una cara vista desde el frente, por lo que captar la imagen con ángulo dificultaría la tarea de detectar un rosto.

Una ventaja para nada desdeñable de la detección con cascadas de Haar es que permite detectar caras a diferentes escalas. Si bien el tamaño del rostro en la imagen capturada no variará mucho, sí que podría estar situado más cerca o más lejos dependiendo de la altura de la persona o la configuración del asiento del piloto.

Otro punto negativo de este método es que aunque se ve beneficiado por la facilidad para detectar una cara que da una imagen con un enfoque claro, directo, frontal y constante se deben de tunear los parámetros del clasificador. Como se ha visto en el capítulo 2.4, en algunos pasos intermedios el algoritmo requiere de la definición de una serie de umbrales y límites. En una situación en la que las imágenes con las que trabaje el clasificador son muy cambiantes, dar con unos parámetros que sirvan para detectar la cara en cualquier situación podría ser complicado y se alcanzarían tasas elevadas de fallo. En este proyecto, por un lado, el encuadre va a ser constante: la idea es tomar una imagen y que el rostro aparezca centrado y de forma frontal. Por otro lado, las condiciones de luminosidad pueden variar fuertemente, por lo que la podría ser difícil ajustar el parámetro que convierta de escala de grises a blanco y negro.

De todas maneras, estos parámetros están fuera del alcance de este trabajo pues pertenecen al ámbito del clasificador, herramienta que se ha conseguido de internet y sobre la que no se ha trabajado. Los parámetros que se deben ajustar son parámetros estadísticos como el número mínimo de vecinos que se han de considerar para determinar la detección facial o parámetros relativos a la imagen como factor de escala o tamaños máximos y mínimos del objeto a detectar que serán sencillos de establecer ya que como se ha dicho las condiciones de encuadre serán constantes.

Finalmente, otra desventaja es que la detección deja de funcionar correctamente si el objeto a detectar se tapa aunque sea parcialmente. Por ejemplo, pasar la mano por el rostro podría detener el reconocimiento. Esto podría ser un gran problema, pero considerando que el rostro no suele taparse durante la conducción y que si se tapa es momentáneamente, esto no tendría por qué afectar a la detección de somnolencia.

En cuanto a la detección facial mediante HOG, al igual que los clasificadores de Haar, presenta también una elevada velocidad de computación aunque inferior.

Una ventaja respecto a los clasificadores de Haar es que los clasificadores entrenados con HOG que se plantean para este proyecto, además de estar entrenados para captar rostros con una imagen frontal, también se entrenan con imágenes desde otras perspectivas. Esto no quiere decir que el rostro se vaya a detectar sea cual sea su orientación, el funcionamiento óptimo del clasificador se da cuando capta una imagen frontal, sin embargo, el programa sería ligeramente más robusto. Como se ha comentado, durante la conducción se espera que el rostro se encuentre mirando hacia el frente la mayoría del tiempo, así que esta ventaja que podría ser decisiva en otro proyecto no lo es en este.

Otra ventaja de los clasificadores que usan HOG es que permiten que la cara se tape parcialmente. A diferencia de lo que ocurriría con los clasificadores de Haar, si durante la conducción el sujeto pasa su mano por la cara el programa podría seguir ejecutándose con normalidad y no perder información durante este lapso.

La gran desventaja de este método es la detección de caras a distintas escalas. Dependiendo de la distancia de la cámara al conductor, el rostro podría no estar lo suficientemente cerca para que el

programa pudiera detectarlo. Sin embargo, esto es poco probable, ya que la variación de la posición del rostro en el encuadre es limitada y se trabaja con distancias cortas.

Finalmente, cabe comentar que hay otros métodos para llevar a cabo la detección facial como pueden ser redes neuronales que presentan muchas ventajas (entrenamiento fácil, trabajan con distintas orientaciones del rostro, robustos ante un rostro semi tapado, gran precisión incluso a distintas escalas, etc.). Sin embargo, las redes neuronales no se han considerado para este proyecto, pues la desventaja que presentan es que requieren una mayor potencia computacional, factor limitante en este proyecto.

3. HERRAMIENTAS

A continuación, se van a explicar las herramientas que se han escogido para desarrollar el proyecto, tanto a nivel hardware como software. Previamente, se expondrán los criterios mencionados anteriormente que formarán parte de la decisión sobre que herramientas escoger.

3.1. DEFINICIÓN DE CRITERIOS PARA EVALUAR EL ÉXITO DEL PROYECTO

Como se ha comentado, se deben plantear unos criterios que permitan valorar el desempeño del sistema diseñado para de estar forma evaluar el grado de cumplimiento de los objetivos planteados.

3.1.1. Criterio de validez

En primer lugar, es fundamental definir el criterio de validez. Este sistema está concebido para detectar la somnolencia durante la conducción, por lo tanto, evidentemente deberá ser capaz de hacerlo. Sin embargo, hay que afinar acerca de las condiciones en las que será capaz de detectar la somnolencia. En el capítulo 1.3 se ha hablado de situaciones que podrían afectar a la imagen que se capta del rostro: Luminosidad, oclusión del rostro (por complementos como gafas o sombreros) y escala de la imagen. Esto, tal y cómo se ha visto al estudiar las herramientas para detección de rostros de las que se dispone (capítulo 2.4), efectivamente puede afectar a la detección de rostros.

Los cambios en las condiciones de luminosidad se van a abordar mediante una lámpara de infrarrojos que permita captar imagen en condiciones de poca luz. Con buena luminosidad, el sistema no debería tener problema en funcionar correctamente. Por lo tanto, esta situación queda, a priori, resuelta. El montaje de esta lámpara pretende añadir bastante valor al proyecto, pues tal y cómo se ha comentado en el capítulo 1.1, una iluminación deficiente como por ejemplo al circular por la noche o de madrugada, potencia el cansancio, por lo que es conveniente poder captar la somnolencia en estas condiciones y es una tarea sobre la cual la comunidad no ha trabajado.

La oclusión del rostro es difícilmente tratable. Como se ha explicado al exponer los clasificadores con los que se va a contar para llevar a cabo la detección facial, la oclusión entorpece la tarea llegando incluso a impedirla en el caso de Clasificadores de Haar. Si bien en el capítulo 2.4 se hablaba de tapar el rostro con movimientos como por ejemplo pasar la mano por delante (podrían afectar momentáneamente a la ejecución del programa, pero la mayoría del tiempo el rostro va a estar

descubierto), la oclusión del rostro con gafas o sombreros podría impedir el funcionamiento del programa de forma permanente.

Si a esto le sumamos que el desarrollo del proyecto se ha enfocado a detectar la somnolencia evaluando los ojos, es prácticamente imposible que el sistema planteado sea útil si el sujeto lleva gafas de sol, y el funcionamiento con gafas de ver tendría que estudiarse. Con esto, no se va a tener en cuenta el funcionamiento del sistema bajo condiciones de oclusión facial para definir el criterio de validez.

Finalmente, la escala a la que aparece el rostro en la imagen captada podría afectar a la detección según las características que presentan los clasificadores planteados. Sin embargo, en posición de conducción, se espera que el rostro se encuentre en una posición lo suficientemente cercana a la cámara que permita su detección independientemente de la regulación del asiento, altura del conductor… Es por ello, que el programa debe ser capaz de detectar el rostro independientemente de la distancia a la que se encuentre.

Con esto, se considerará que el sistema es válido si es capaz de llevar a cabo la detección de somnolencia independientemente de las condiciones de luminosidad y posición del conductor, pero se aceptará que no funcione correctamente en caso de que el rostro del conductor no se encuentre completamente al descubierto.

3.1.2. Criterio de eficacia

El segundo criterio a definir es el de eficacia. Este criterio hace referencia exclusivamente a que el programa pueda ejecutarse en tiempo real, lo cual es fundamental para poder hacer una detección de la somnolencia adecuada. Es especialmente importante debido a que la intención es implementar el programa desarrollado en una placa ordenador de bajo coste comercial, no especializada para esta tarea, por lo que la potencia de cálculo del hardware no es demasiado alta y su rendimiento es mejorable.

Si el sistema funciona y es viable, una mejora sustancial sería el desarrollo de un sistema especializado para tal tarea, con lo cual la velocidad de computación podría aumentar y los costes podrían mantenerse al producirse en serie (no habría que desarrollar una placa específica para un solo uso). Sin embargo, de lo que se trata en este momento es de evaluar la eficacia con los medios de los que se dispone. Se determinará que el sistema es eficiente si realiza la detección de la somnolencia en tiempo real y se comprobará mediante ensayos.

3.1.3. Criterio de coste

Uno de los puntos más importantes de este proyecto es mantener el coste bajo, de forma que se demuestre que no hace falta una gran inversión para realizar esta función que supondría un importante aumento de la seguridad en las carreteras. Se considerará que se ha cumplido el criterio de bajo coste si los costes implicados directamente en el sistema (dado que las herramientas de software usadas son gratuitas se considerarán únicamente los costes de los equipos hardware) son menores que el coste de los sistemas que se ofrecen actualmente en el mercado y que oscilan entre 300 € y 800 € [18]. No se van a considerar los costes no recurrentes de desarrollo (horas de trabajo), pues siendo realistas haría que la cifra se alejara en exceso del valor de mercado que tendría el producto desarrollado.

Además, habría que tener en cuenta que el coste de un prototipo es evidentemente más caro que el coste de un producto final de producción en serie.

Los siguientes capítulos detallan las herramientas que se han usado en el proyecto. Lo primero en decidirse ha sido el lenguaje de programación con el que se iba a trabajar (Python) y la plataforma sobre la que se va a implementar el programa (Raspberry Pi).

3.2. PYTHON

En este proyecto tiene un gran peso la parte informática, por lo que el lenguaje de programación no se puede escoger a la ligera. Python es un lenguaje de programación de código abierto, por lo que es gratuito y cuenta con una gran comunidad en internet. Se puede encontrar mucha información, librerías de código y ayuda en la red, lo que supone un gran respaldo al iniciar un proyecto de este tipo.

Lo que hace a Python el candidato perfecto para este trabajo es por un lado su simplicidad y la libertad que permite durante la programación, siendo fácil de escribir y de depurar el código. Por otro lado, las librerías ya desarrolladas de reconocimiento facial y su disponibilidad de forma gratuita en internet son un punto que hay que agradecer a la comunidad. Finalmente, considerando la Raspberry Pi como potencial candidato a ordenador que ejecute el código, la buena relación entre estos dos sistemas también se ha tenido en cuenta.

3.3. RASPBERRY PI

La Raspberry Pi es un ordenador de placa simple de bajo coste conocido por su versatilidad. Sirve como base para gran variedad de proyectos informáticos, de robótica, automatismos, etc. y es un producto superventas.

Emplea un sistema operativo basado en Linux y optimizado para las características de la placa, el Raspberry Pi OS.

La principal ventaja de esta placa es que, al ser un producto muy vendido y mundialmente conocido, dispone de mucho soporte en línea. Además, la filosofía de su sistema operativo (de tipo Unix) es similar a la filosofía de Python y la mayoría de distribuciones Linux vienen con Python instalado por defecto, como ocurre en Raspberry Pi OS.

En concreto, en este proyecto se ha usado la Raspberry Pi 4 Model B, que es la versión del computador más reciente en el mercado. Cuenta con un procesador de cuatro núcleos de 64 bits que funciona a 1,5 GHz, sin embargo, en el momento de la realización del proyecto el sistema operativo Raspberry Pi OS de 64 bits se encuentra en versión beta.

Durante el trabajo, se ha priorizado la sencillez y evitar posibles fuentes de problemas dado que se trabaja con distintas librerías y versiones de Python cuya interacción de por sí ya pueden suponer la aparición de fallos, por lo que se ha desestimado trabajar con la versión del SO en 64 bits por no ser definitiva y se ha usado la de 32 bits. Esto supone que, por un lado, el procesador procesará las

instrucciones a 32 bits en vez de a 64, viéndose reducida su capacidad. Por otro lado, un sistema operativo de 32 bits en una CPU de 64 bits limitará la gestión de RAM a 4 GB.

Con esto, de las versiones de la Raspberry Pi 4 que hay la de 4 GB de RAM podría haber servido. A pesar de ello, se ha adquirido la versión de mayor memoria RAM disponible (8 GB), pues dadas las necesidades del proyecto y la incerteza de si cumplirá el criterio de eficacia, esto permitiría una mayor holgura en el momento en el que el SO de 64 bits esté disponible.

A nivel de desarrollo, la Raspberry Pi 4 ofrece muchas facilidades que simplifican el trabajo. Cuenta con salidas de imagen HDMI, perfectas para conectar la placa a cualquier monitor. También dispone de conectividad Wi-Fi, muy útil para descargar todas las herramientas que se requieren para este proyecto. Además, presenta un puerto específico para conectar una cámara. Todo esto facilita las labores de desarrollo y montaje, lo que permite invertir más tiempo en la elaboración del programa en sí.

En cuanto a la alimentación, funciona a 5,1 V, por lo que para instalarla en el coche habrá que emplear un cargador que se conecte a la toma de 12 V y los convierta a 5,1 V. Según datos oficiales de Raspberry, la corriente que absorbe se encuentra alrededor de 600 mA, pudiendo llegar a 1,25 A en condiciones de máximo estrés. Esto supone un consumo máximo de 6,25 W, pero debido al uso de periféricos la organización recomienda una fuente de alimentación que pueda entregar hasta 3 A (15 W a 5 V).

3.4. VSCODE

El Visual Studio Code es un editor de código gratuito que presenta ayudas para muchos lenguajes de programación, entre ellos Python. Facilita las tareas de programación ofreciendo un entorno de trabajo agradable y personalizable, con la funcionalidad de autocompletado y una buena interfaz de depuración (puntos de paro, ejecución paso por paso, examinación de variables…).

Un editor de código es un programa bastante ligero que permite construir y ejecutar código sin consumir demasiados recursos. También existen los entornos de desarrollo integrados (IDE por sus sigla en inglés), programas con muchas más funcionalidades y por ende, más pesados y lentos. Un conocido IDE es Visual Studio, del cual se desprende el editor que se ha usado en este proyecto, el Visual Studio Code.

Este editor de código se ha usado en Windows para desarrollar el programa que posteriormente sería trasladado a la Raspberry Pi con las modificaciones pertinentes.

3.5. THONNY

Como se ha comentado anteriormente, la mayoría de las distribuciones de Linux vienen con Python instalado por defecto. El Raspberry Pi OS trae además instalado Thonny de forma nativa, un entorno de programación específico para Python.

En este proyecto, el desarrollo de código directamente sobre la Raspberry ha sido limitado, más bien lo que se ha realizado es adaptar el código para compatibilizarlo con este pequeño ordenador. Es por

ello por lo que no se han exprimido las funcionalidades de Thonny, pero al igual que el VS Code, es capaz de depurar e inspeccionar variables.

3.6. CÁMARA

La captura de imagen consiste en la obtención de información de una escena generalmente mediante un dispositivo óptico como puede ser una cámara, si bien se pueden obtener imágenes con otras tecnologías como la termografía infrarroja, los ultrasonidos… Sin embargo, estas últimas no aplican a este campo de investigación.

Para el sistema que se está diseñando, la cámara es un elemento fundamental. Se requiere una cámara que permita obtener imagen en distintas condiciones de luminosidad, tanto de día como de noche, tanto durante la circulación en el exterior como durante un túnel. Cualquier cámara toma imágenes en condiciones de buena luminosidad, pero si se quiere captar el entorno cuando no hay luz hay que recurrir a cámaras especializadas.

De nuevo, trabajar con la Raspberry Pi trae la ventaja de que tiene su propia cámara oficial para tomar fotografía infrarroja. De esta forma, acompañada de una lámpara IR, se puede capturar el rostro cuando no hay luz visible. Esta cámara es la Raspberry Pi NoIR Camera v2, y consiste básicamente en una cámara a la cual se le ha quitado el filtro de infrarrojos que suelen traer por defecto en el objetivo. Las cámaras incorporan este filtro para que solo lleguen al sensor las ondas pertenecientes al espectro visible por los humanos, que son las que se encuentran entre longitudes de onda de unos 380 y 750 nm. Así, la imagen capturada es fiel a lo que ven nuestros ojos. De 100 a 400 nm encontramos la radiación ultravioleta, y de 700 nm a 1 mm encontramos la radiación infrarroja. Si no se montara este filtro, la imagen capturada tendría tonos rojizos, pues el sensor está captando multitud de ondas de radiación infrarroja.

Con la cámara seleccionada, cuando hay luz la imagen tomada tiene efectivamente tonos rojizos. Esto no es un problema ya que como se ha visto en el capítulo 2.4, el color no es importante para el reconocimiento facial y la imagen se pasa a escala de grises. Cuando no hay luz, la cámara no logra capturar nada. Es por ello que se ha adquirido también una lámpara de LEDs infrarrojos para instalar en el habitáculo, de forma que cuando no haya luz visible, la cámara pueda tomar imágenes gracias a la presencia de luz IR y a no tener un filtro que bloquea estas ondas.

3.7. LÁMPARA IR

La lámpara de LEDs IR que se va a usar para poder captar imagen en la oscuridad es una placa de bajo coste con 24 leds que promete iluminar hasta 20 m de distancia. Se ha escogido un modelo con longitud de onda de 850 nm por una razón: Las lámparas IR con una longitud de onda cercana al espectro visible emiten un pequeño resplandor rojo al funcionar, que permitirá saber si la lámpara está funcionando correctamente de forma rápida y sin tener que usar la cámara. Al pretender hacer un montaje del prototipo que es más bien una presentación y no es profesional, es importante poder conocer el estado de la lámpara a simple vista para tareas de diagnóstico de fallos.

No obstante, si el sistema funciona y es válido, sería conveniente hacer el montaje con una lámpara de 940 nm, pues con esta longitud de onda no se emite ningún resplandor perceptible por el ojo humano.

Otra característica de la lámpara que ha influido en su adquisición es su tensión de alimentación, que es de 12 Vdc al igual que muchos de los consumos DC que se encuentran en cualquier coche. Esto facilitará su montaje, pues se puede conectar a la toma de 12 V del coche o a cualquier equipo que se alimente a esta tensión (bombillas, radio…).

Una funcionalidad muy interesante que incluye la lámpara adquirida es la presencia de una LDR (*Light Dependent Resistor*) que regula el funcionamiento de los LEDs, apagándolos cuando hay luz y encendiéndolos cuando no la hay.

Debido a que se carece de información por parte del fabricante, se ha llevado a cabo un montaje para conocer el consumo de potencia de la lámpara y tratar de conocer mejor sus especificaciones:

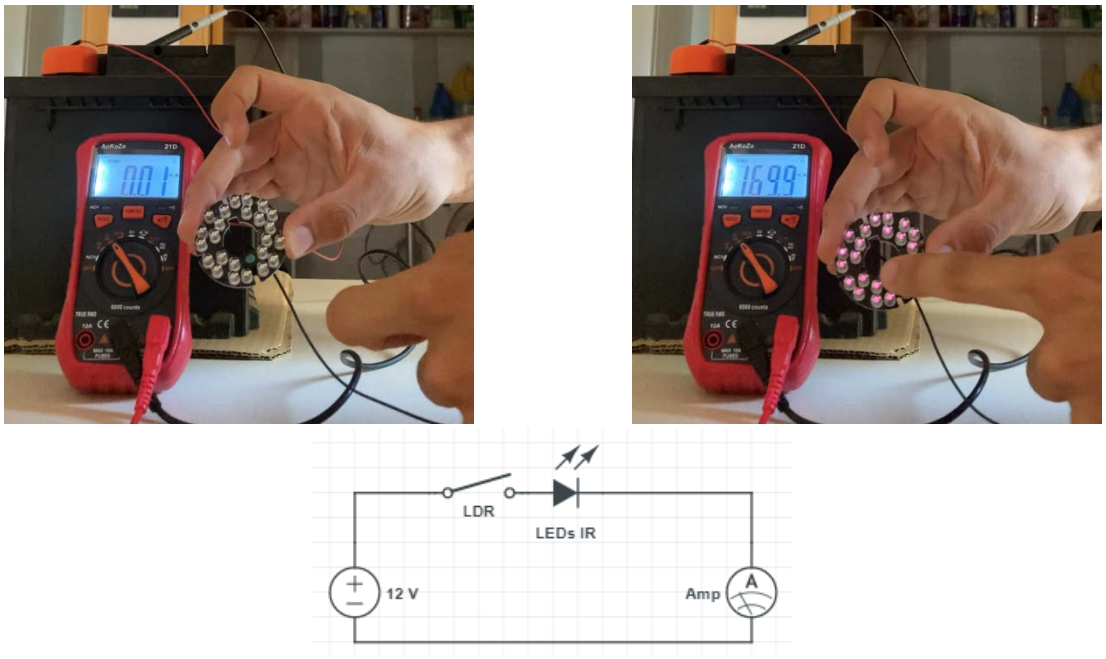

Figura 27. Determinación consumo de potencia de la lámpara IR

Se ha representado la LDR como un interruptor. Cuando la lámpara detecta que hay luz no se activa, por lo que no circula corriente (imagen izquierda). Al tapar la LDR, la lámpara detecta oscuridad por lo que los LEDs se encienden (se puede apreciar el ligero brillo en la imagen derecha). Entonces, el consumo es de 169,99 mA, lo que a 12,05 V de tensión medida en bornes de la batería resulta en una potencia de 2,05 W (85 mW por LED). Haciendo una revisión de modelos de LEDs IR de 850 nm por internet, se encuentra el que forma parte de la lámpara: Consumo de 90 mW, corriente entre 50 y 80 mA y tensión entre 1,4 y 1,6 V, por lo que se deduce que la lámpara se compone de 3 hileras en paralelo de 8 LEDs en serie cada una.

3.8. CONVERTIDOR REDUCTOR

Como se ha explicado, la Raspberry Pi requiere de una alimentación de 5,1 V. Considerando un consumo de 6,25 W, obtener esta alimentación no es inmediato en un vehículo de carretera. Si bien muchos de ellos van equipados con tomas USB de 5V, estas suelen estar limitadas a 500 mA, por lo que no son capaces de entregar la potencia necesaria para alimentar la Raspberry. Por lo tanto, hay que recurrir a la alimentación directa de batería, que se puede hacer a través de cualquier consumo no esencial del vehículo o de una toma de 12 V dedicada.

Por lo general, las cargas del vehículo se alimentan a 12 V (tensión de la batería). De esta forma, hay que añadir una etapa intermedia desde la alimentación hasta la Raspberry, un convertidor reductor de tensión. En este caso se ha optado por emplear un cargador convencional 12/5 V para la conexión a la toma del mechero del vehículo y con salidas USB.

3.9. TARJETA DE MEMORIA

El formato de tarjeta de memoria que emplea la Raspberry Pi es el microSD. En la elección de la tarjeta influyen varios factores:

- Capacidad: La Raspberry Pi puede funcionar con tarjetas de memoria de más de 32 GB, sin embargo el *bootloader* no está preparado para ello, pues solo soporta lectura de datos desde sistemas de archivos con formato FAT16 y FAT32 y las tarjetas de memoria de capacidad mayor a 32 GB presentan formato NTFS. Por lo tanto, siendo 32 GB una capacidad más que suficiente para este proyecto, la tarjeta adquirida es una microSDHC de esta capacidad para evitar realizar tareas de formateo.
- Clase: Este parámetro hace referencia a la velocidad mínima a la que trabaja la tarjeta y está entre 2 y 10. A día de hoy, la mayoría de fabricantes conocidos produce tarjetas de clase 10 (velocidad mínima de 10 MB/s). Pensando en que en este proyecto se trabaja con vídeo y para los ensayos se quiere poder guardar en vivo y siendo la clase 10 casi un estándar de las marcas reconocidas, se adquiere una tarjeta de esta clase
- UHS: De *Ultra High Speed*, este parámetro es una ampliación moderna del anterior que especifica la mejora respecto a la velocidad mínima de las tarjetas de clase 10. La tarjeta adquirida es UHS-I Clase 3, por lo que la velocidad mínima es de 30 MB/s y la máxima de 104 MB/s
- Uso: La tarjeta se va a usar en un proyecto que le va a demandar un buen comportamiento en la gestión de vídeo y datos. Para no tener problemas en la satisfacción de estos requisitos, se adquiere una tarjeta de clase de vídeo V30 (apta para resolución 4K a la velocidad mínima de 30 MB/s) y de clase de rendimiento A1

3.10. LIBRERÍAS DE PYTHON PARA RECONOCIMIENTO FACIAL

Conviene incluir este apartado debido a que los clasificadores empleados son herramientas que se han descargado directamente de internet y están fuera del alcance de este trabajo. Hay que dar crédito a la comunidad que publica recursos de forma desinteresada.

Los clasificadores empleados trabajan con los algoritmos que se han explicado en el capítulo 2.4. Se han realizado dos versiones del programa, una con detección mediante cascadas de Haar y otra con detección mediante HOG. Debido a la dificultad para elegir uno de los dos métodos, se deberán ensayar ambos programas para ver cuál es el que ofrece mejores resultados.

El reconocimiento mediante cascadas de Haar se hace con la librería OpenCV (*Open Computer Vision*), una biblioteca de visión artificial desarrollada por Intel. Es la librería de visión artificial más popular y permite llevar a cabo tareas como detectar movimiento y reconocer formas, siendo el reconocimiento facial una de sus aplicaciones esenciales. Es multiplataforma y funciona tanto en Windows como en Raspberry SO, es libre (bajo una licencia que permite emplearla libremente para propósitos comerciales y de investigación) y dispone de una documentación muy completa en su web, por lo que encaja perfectamente en este proyecto. Está desarrollada en C++, pero incluye conectores para varios lenguajes, Python entre ellos.

El reconocimiento mediante HOG se hace con Dlib, una librería que también es libre y también está desarrollada en C++. Además de la detección facial mediante HOG, contiene una función esencial para el buen funcionamiento del programa que es la que permite extraer las coordenadas faciales (Figura 3. Visualización de las coordenadas faciales que extrae Dlib Figura 3).

4. DESARROLLO DEL PROYECTO

4.1. DESARROLLO DEL ALGORITMO

El proceso de detección de somnolencia es sencillo y se muestra en el siguiente diagrama:

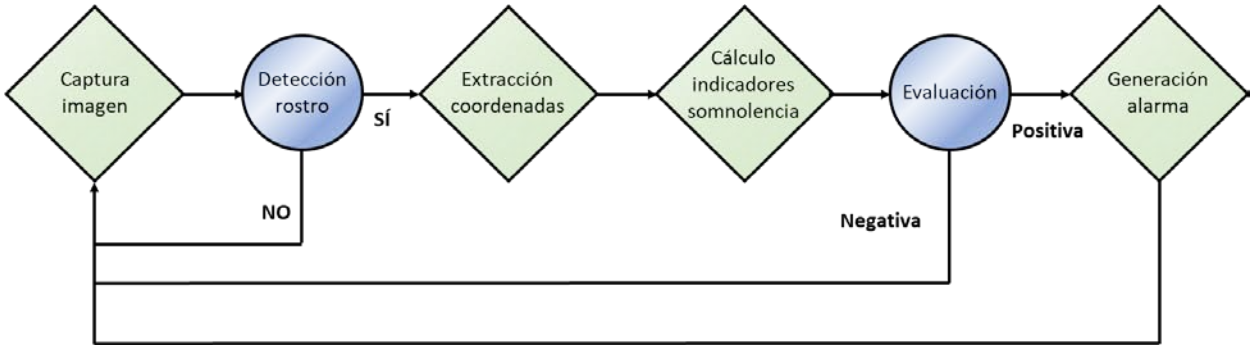

Figura 28. Algoritmo

Para hacer funcionar este programa se requiere una serie de funciones específicas para las siguientes tareas:

- Captura de vídeo
- Detección de rostro
- Extracción de coordenadas

- Para el cálculo de los indicadores se hace uso de una función que convierte el formato de las coordenadas extraídas a un vector y de otra función que calcula las distancias entre los puntos faciales extraídos
- Para la evaluación de los indicadores de somnolencia se requiere de una función de control del tiempo de ejecución del programa
- En cuanto a la generación de alarma, como se ha explicado anteriormente, en este proyecto una vez detectada la somnolencia no va a realizarse ninguna actuación sobre el vehículo. Se ha programado una alarma que se muestra por pantalla

4.2. DESARROLLO EN WINDOWS

El desarrollo del algoritmo se ha realizado en Windows en primera instancia por ser más sencilla la instalación de programas y librerías y haber más recursos en internet. A pesar de haber versiones más recientes, el programa se ha desarrollado en Python 3.6.8, pues en la fecha en la que se ha programado en Windows, Python 3.6 es la última versión de este lenguaje de programación que funciona correctamente con OpenCV. Por lo tanto, el primer paso es descargar e instalar Python, esto se puede hacer directamente desde la página de la organización. Seguidamente, hay que instalar las librerías que no vienen por defecto y que se van a emplear.

4.2.1. Instalación de librerías

El método de instalación que se va a usar es "pip", un sistema de gestión de paquetes para Python que incluye Python por defecto en la versión que se ha empleado. La ventaja de emplear este método es que se pueden instalar librerías que no se distribuyen como parte de la instalación estándar de Python de forma sencilla. Basta con indicar el nombre del módulo que se quiere instalar y pip se encarga de buscarlo, descargarlo e instalarlo. Antes de ejecutar pip, conviene actualizarlo por si hay alguna versión más reciente. Esto se hace mediante el comando `python -m pip install --upgrade pip`:

Figura 29. Actualizar herramienta "pip" desde la consola de comandos

Como se puede ver, en este caso la versión de pip se ha actualizado de la 18.1 a la 21.0.1. Esto previene que no se encuentren los paquetes que se buscan, que se instalen versiones desfasadas, que la instalación no funcione correctamente…

Con esta herramienta instalada ya se pueden instalar las librerías necesarias. En Windows es sencillo, pues se pueden instalar todas desde pip:

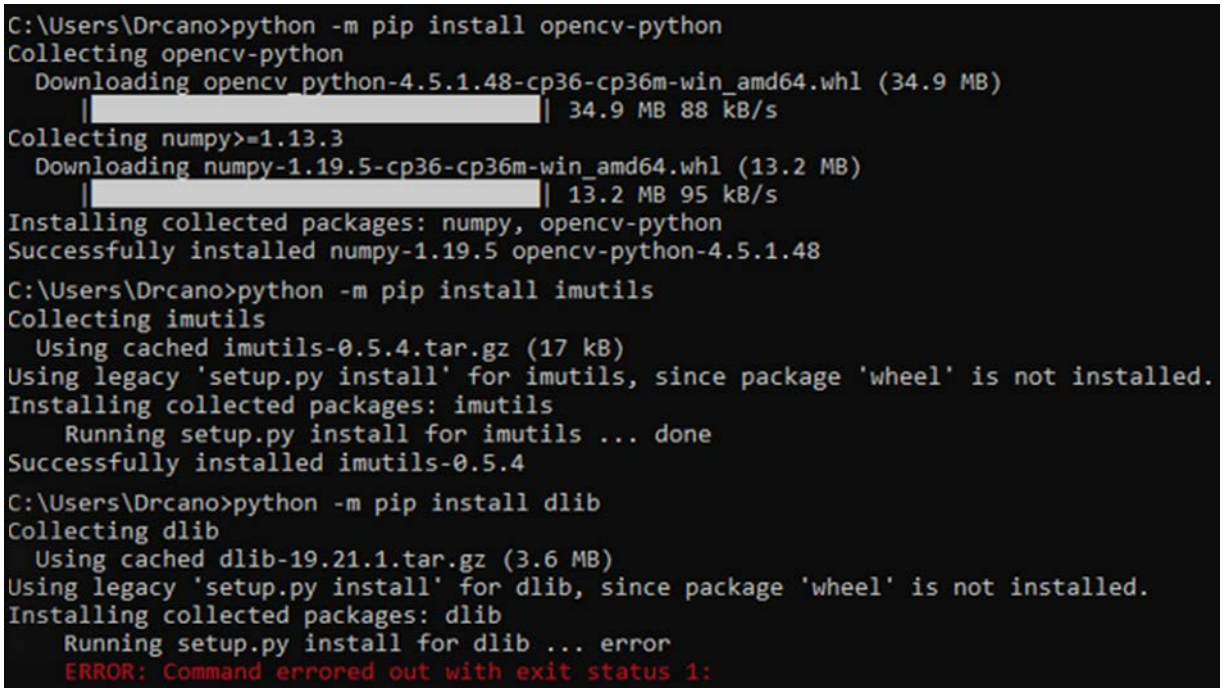

Figura 30. Instalación de librerías requeridas mediante pip. Error en la instalación de Dlib

En la imagen superior se pueden ver varios sucesos. En primer lugar, no todas las librerías se buscan por su nombre: Para instalar OpenCV se debe buscar como "opencv-python". En segundo lugar, si alguna librería necesita de otra para su funcionamiento, pip la descarga automáticamente (es lo que ha sucedido con el paquete NumPy, usado para procesamiento numérico). En tercer lugar, al instalar con pip, al igual que mediante cualquier otro método, pueden surgir problemas. En este caso, la instalación de Dlib no ha sido exitosa debido a que pip busca la última versión, y en este caso no es compatible con Python 3.6.8. Por lo tanto, hay que buscar cuál es la última versión de Dlib compatible con la versión de Python instalada y especificárselo a pip. En este caso, la versión de Dlib adecuada es la 19.8.1, descargable desde PyPi, el repositorio de software de Python online. Así, buscando en https://pypi.org/simple/dlib/ la versión deseada y copiando el enlace, se le puede indicar a pip de la siguiente forma:

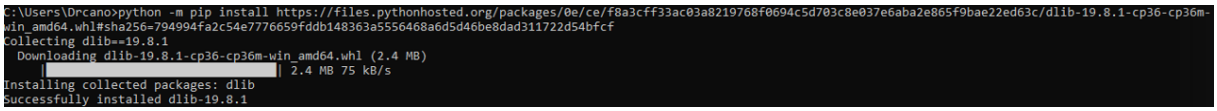

Figura 31. Instalación de una versión en específico de Dlib (19.8.1) mediante pip

En este caso, se ha copiado el enlace del archivo dlib-19.8.1-cp36-cp36m-win_amd64.whl de Dlib 19.8.1, pues es compatible con la versión de Python 3.6 (ver dígitos cpXX del archivo) y para sistemas operativos de 64 bits como el que se está usando. Los archivos .whl son un tipo de formato de paquete de Python, mientras que el otro tipo de extensión común para los paquetes es .tar.gz. Ambos pueden instalarse mediante pip.

Finalmente, se puede comprobar que las librerías se han instalado correctamente y que se pueden importar en Python para su uso:

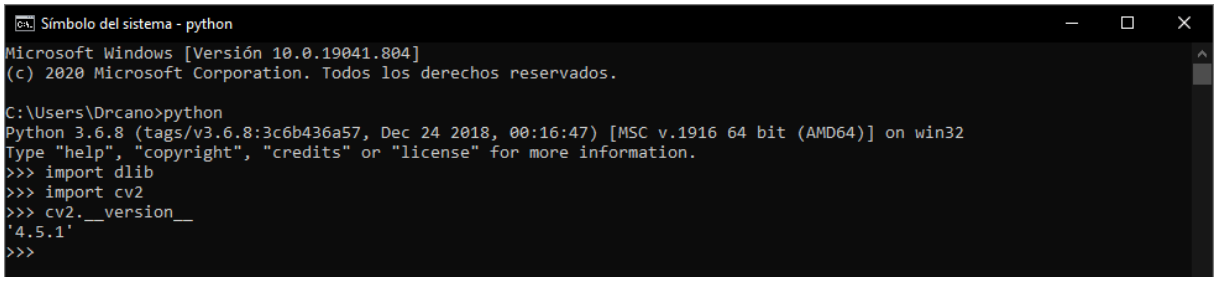

Figura 32. Comprobación instalación librerías

4.2.2. Programación del algoritmo

Al principio del código hay que importar las librerías que se van a usar. Se hace mediante el comando "*import*" seguido del nombre de la librería.

```
import cv2
import dlib
from imutils import face_utils
from scipy.spatial import distance as calculo_distancia
import time
```

OpenCV se importa bajo el nombre "cv2" en Python, y se emplea para realizar el control de la imagen, tanto en la captura como en la salida de vídeo. Además, en el caso de realizar la detección facial con cascadas de Haar, esta librería cuenta con clasificadores. Como se ha comentado anteriormente, Dlib es necesaria para realizar la extracción de las coordenadas faciales, y en caso de realizar la detección facial mediante HOG, cuenta con funciones específicas para ello.

La librería "imutils" presenta una serie de funciones para el procesamiento de imágenes. En este proyecto simplemente se emplea una función que convierte los puntos faciales extraídos con Dlib que tienen un formato de estructura en un vector. La estructura que contiene la información facial que entrega Dlib tiene una subestructura para cada punto, y cada punto es una estructura que contiene dos campos: el valor X y el valor Y. Con la función "*face_utils*" se transforma esto a un vector de 68 filas en la que cada fila contiene el valor X e Y que define el punto.

De la librería "SciPy" se ha importado la función "*distance*", que se usará para calcular la distancia euclídea entre dos puntos contenidos en el vector generado por *face_utils*.

Finalmente, la librería "time" se usa para contar el tiempo que transcurre con alguno de los indicadores de somnolencia en rango de alerta para así generar una alerta cuando se considere. También se emplea para controlar el tiempo de ejecución del programa y tener una referencia en los ensayos realizados, lo que permite analizar la velocidad del programa y comprobar su buen funcionamiento.

Tras importar las librerías, se programan las funciones que calcularán los parámetros EAR y MAR definidos en el capítulo 2.3. Estas funciones se han declarado al inicio del programa, fuera del bucle de cálculo. Esto simplemente se ha hecho para mantener el bucle lo más sencillo y legible posible.

```
def eye_aspect_ratio(ojos):
# Calcula la distancia entre los puntos p2-p6, p3-p5 y p1-p4
# que definen el contorno del ojo según Dlib68
    A = calculo_distancia.euclidean(ojos[1], ojos[5])
    B = calculo_distancia.euclidean(ojos[2], ojos[4])
    C = calculo_distancia.euclidean(ojos[0], ojos[3])
# Cálculo EAR
    EAR = (A+B)/2.0/C
    return EAR

def mouth_aspect_ratio(boca):
# Calcula la distancia entre los puntos que definen
# el contorno interior de la boca según Dlib68
    A = calculo_distancia.euclidean(boca[1], boca[7])
    B = calculo_distancia.euclidean(boca[2], boca[6])
    C = calculo_distancia.euclidean(boca[3], boca[5])
    D = calculo_distancia.euclidean(boca[0], boca[4])
# Cálculo MAR
    MAR = (A+B+C)/2.0/D
    return MAR
```

Estas funciones reciben un vector de la longitud de los puntos que definen el contorno (del ojo o de la boca), y en cada posición el vector contiene el valor de las coordenadas X e Y de ese punto.

Tras esto, comienza el proceso de inicialización de variables y procesos. Se declaran las variables relativas a la detección de somnolencia:

VARIABLE	DEFINICIÓN	VALOR INICIALIZACIÓN
threshold_pestanyeo	Límite inferior del EAR a partir del cual se considera que el grado de apertura de los ojos se corresponde con la presencia de somnolencia	Dependerá de la persona
threshold_cerrados	Tiempo necesario considerado para que un EAR inferior a "threshold_pestanyeo" produzca una alarma de somnolencia. Este tiempo evita que salte la alarma por pestañeos y requiere de un grado de apertura de los ojos bajo sostenido en el tiempo	En general, 5 segundos con los ojos entornados son suficientes para determinar la presencia de somnolencia
tiempo_cerrados	Esta variable recuenta con la librería "*time*" el tiempo durante el cual el EAR es inferior a "threshold_pestanyeo", de forma que cuando "tiempo_cerrados"	0

VARIABLE	DEFINICIÓN	VALOR INICIALIZACIÓN
	sea superior a “threshold_cerrados” salte la alarma de somnolencia	
tiempo_cerrados_ini	Es una referencia de tiempo para el cálculo del tiempo durante el cual el EAR es inferior a “threshold_pestanyeo”	0
threshold_bostezo	Límite inferior del MAR a partir del cual se considera que el grado de apertura de la boca se corresponde con un bostezo	Dependerá de la persona
threshold_bostezo_tiempo	Tiempo necesario considerado para que un MAR inferior a “threshold_bostezo” produzca una alarma de bostezo. Este tiempo evita que salte la alarma por movimientos aleatorios de la boca y requiere de un grado de apertura de boca elevado sostenido en el tiempo	La media de la duración de los bostezos es de 7 segundos, así que se ha inicializado a 5 segundos, un valor algo inferior que permita registrar más bostezos
tiempo_bostezo	Esta variable recuenta con la librería “*time*” el tiempo durante el cual el MAR es superior a “threshold_bostezo”, de forma que cuando “tiempo_bostezo” sea superior a “threshold_bostezo_tiempo” salte la alarma de somnolencia	0
tiempo_bostezo_ini	Es una referencia de tiempo para el cálculo del tiempo durante el cual el MAR es superior a “threshold_bostezo”	0

Tabla 1. Declaración de variables implicadas en el proceso de detección de somnolencia

Como puede verse en la tabla anterior, hay variables cuyo valor depende del sujeto y variables cuyo valor debe ajustarse mediante datos experimentales obtenidos de ensayos que aporten información sobre el comportamiento humano. Por ejemplo, el grado de apertura de los ojos es algo que depende de la persona. Poniendo el ejemplo más sencillo, la monitorización de los ojos de una persona asiática requerirán un ajuste distinto al de los ojos de otra persona (caucásica, africana…). Incluso dos personas caucásicas pueden presentar una forma de los ojos muy distintos. En cuanto a la boca, a pesar de que se ha indicado que la variable “threshold_bostezo” depende de la persona, se podría considerar que esta variable es más robusta que “threshold_pestanyeo”.

El siguiente paso es inicializar las herramientas clave del programa: el detector, la extracción de características faciales y la captura de imagen. Los detectores con los que se ha trabajado son, tal y como se ha explicado, las cascadas de Haar de OpenCV y el clasificador HOG de Dlib. En caso de emplear los clasificadores de Haar de OpenCV, el detector se carga de la siguiente forma:

```
detector = cv2.CascadeClassifier(r'C:\Users\Drcano\pythonProject1\haarca
scade_frontalface_default.xml')
```

Como se puede intuir, se ha empleado un clasificador especial para la detección de caras con un enfoque frontal, situación que se dará normalmente en el puesto de conducción. Existen otros clasificadores de OpenCV para detectar ojos, ojos con gafas, sonrisas… Sin embargo, no hay un clasificador especializado para rostros de perfil o mirando hacia arriba o hacia abajo.

En caso de emplear la clasificación HOG para caras de Dilb, el detector se carga a través de la siguiente función, que como se puede ver, también está especializada para imágenes de rostros de frene:

```
detector = dlib.get_frontal_face_detector()
```

La extracción de coordenadas mediante Dlib se carga llamando a la función de predicción de formas de esta librería con la ruta en la que se encuentra el predictor entrenado para lo que se quiera detectar, en nuestro caso, 68 puntos significativos en el rostro:

```
predictor = dlib.shape_predictor(r'C:\Users\Drcano\pythonProject1\shape_
predictor_68_face_landmarks.dat')
```

La "r" delante de la cadena de la ruta se coloca para evitar que Python interprete el carácter "\" como un carácter de escape. También se podría emplear una barra diagonal "/" en vez de una barra diagonal inversa para evitar problemas, o dos barras diagonales inversas. Finalmente, la captura de imagen en un PC es sencilla y se puede realizar con la propia librería de OpenCV, especificando la cámara 0 de forma que Python controle la cámara que viene por defecto en el ordenador:

```
# Iniciar captura de imagen (0 = webcam)
captura = cv2.VideoCapture(0)
```

Tras la inicialización de variables y de procesos, se da inicio al bucle de análisis de somnolencia. Dentro de este, lo primero que se hace es determinar el tiempo de ejecución del programa para más adelante poder contabilizar el tiempo que los indicadores de somnolencia llevan propasándose de los límites planteados. Seguidamente, se captura un *frame* del vídeo que se está grabando, digitalizándose y guardándose en la variable "*frame*". Acto seguido, se convierte a escala de grises con OpenCV, de forma que se cumplan los requisitos de los detectores faciales:

```
while True:
    tiempo = time.time()
    _, frame = captura.read()
    frame_gris = cv2.cvtColor(src=frame, code=cv2.COLOR_BGR2GRAY)
```

Que los detectores faciales tengan el requisito de trabajar con imágenes en escala de grises no solo es una ventaja sino que en caso de que necesitaran información cromática el programa no podría funcionar en condiciones de baja luminosidad. En la oscuridad, al trabajar con una lámpara infrarroja y una cámara sin filtro IR, la imagen capturada está prácticamente en escala de grises, y esto no permitiría ejecutar un algoritmo que necesite imágenes a color. Además, es una ventaja a nivel

computacional, pues en una imagen en escala de grises la información almacenada se limita al nivel de brillo, mientras que en una imagen a color cada píxel contiene información relativa a la luminancia (nivel de brillo) y color (componentes cromáticas roja, verde y azul). En cambio, en una imagen en escala de grises solo se guarda información relativa al nivel de brillo.

Con la imagen capturada y guardada, es el momento de aplicar el detector facial. En función del detector que se está empleando, se dan unas instrucciones u otras. En el caso de la detección con cascadas de Haar, el clasificador se aplica a la imagen "frame_gris" y retorna un una variable que almacena vectores 1x4 conteniendo información sobre la dimensión del rectángulo que contiene cada cara detectada:

```
caras = detector.detectMultiScale(frame_gris,scaleFactor = 1.1,
minNeighbors = 5, minSize = (30,30))
```

La configuración de los parámetros que se ha establecido es la que aparece en la documentación de la librería por defecto ya que su funcionamiento en los ensayos realizados con el PC ha sido satisfactorio:

- *scaleFactor*: El modelo ha sido entrenado para un tamaño concreto de cara. Con este factor se construye una pirámide de imágenes a distintas escalas a partir de la original, permitiendo al detector que haga un barrido de la imagen a distintas escalas y aumentando así las probabilidades de éxito. Dando el valor de 1.1 a este parámetro se añade un paso intermedio en la función que se encarga de reducir el tamaño de la imagen en un 10 % en cada escalado, teniendo así una pirámide en la que cada imagen es 10 veces más pequeña que la anterior. Cuanto más pequeño sea el valor que se le da a este parámetro, más veces se realizará el escalado de la imagen y más tardará el programa en ejecutarse
- *minNeighbors*: Este parámetro especifica cuantos vecinos debe tener cada rectángulo que representa la detección de una cara para considerar una detección positiva. Como se ha comentado, una de las desventajas de las cascadas de Haar son los falsos positivos. Este parámetro ayuda a combatir esto. Un valor mayor resulta en menos detecciones
- *minSize*: Tamaño mínimo que se considera para el rostro, las detecciones de rostros de menor tamaño serán ignoradas. Este parámetro va ligado a la distancia a la que se encuentre el conductor de la cámara, pues cuanto más lejos esté menor tamaño tendrá el rostro en la imagen. Sin embargo, la variabilidad es pequeña por lo que no es determinante. Tiene su parámetro complementario *maxSize* que no se ha configurado, por lo que no hay límite superior al tamaño del rostro.

En el caso de extraer el rostro con Dlib, no hay parámetros que se deban tunear. La instrucción es simple:

```
caras = detector(frame_gris)
```

Esto devuelve un objeto que representa un rectángulo que encierra cada rostro detectado.

Como se ha visto, tanto Dlib como OpenCV pueden extraer más de un rostro de la misma imagen. Pese a ello, el programa está concebido para extraer solamente uno: el del conductor. Así y todo, para evitar problemas que lleven a un paro del programa en caso de que se detecte otro rostro, se realiza la

detección para tantos rostros como se hayan encontrado en el siguiente bucle, que se encuentra anidado dentro del bucle *while* anterior:

```
for cara in caras:
# Crear mapa facial con el predictor de 68 puntos de Dlib
      cara = dlib.rectangle(int(cara[0]), int(cara[1]),
      int(cara[0]+cara[2]), int(cara[1]+cara[3]))
      puntos = predictor(image=frame_gris, box=cara)
      puntos_coordenadas = face_utils.shape_to_np(puntos)
# Se extraen las coordenadas de cada ojo y se calcula el EAR
      ojo_izq = puntos_coordenadas[42:48]
      ojo_der = puntos_coordenadas[36:42]
      EAR_izq = eye_aspect_ratio(ojo_izq)
      EAR_der = eye_aspect_ratio(ojo_der)
# Cálculo EAR medio
      EAR = (EAR_izq + EAR_der)/2.0
# Se extraen las coordenadas del contorno interior de la boca y se
calcula el MAR
      boca = puntos_coordenadas[60:68]
      MAR = mouth_aspect_ratio(boca)
```

En el caso de haber detectado el rostro con cascadas de Haar, antes de ejecutar el predictor de 68 coordenadas faciales de Dlib hay que transformar el vector con los 4 vértices del rectángulo que encierra el rostro en una variable válida para el predictor de Dlib. Esto consiste fundamentalmente en crear una variable de clase “rectangle” y en hacer una simple operación para transformar el formato del rectángulo de OpenCV en el formato de Dlib. OpenCV considera el vértice inferior izquierdo como origen de coordenadas para el resto del rectángulo y ocupa las dos primeras posiciones del vector 1x4 que devuelve en guardar la coordenada X e Y respecto al origen de la imagen. Las siguientes dos posiciones del vector 1x4 contienen el ancho y el alto del rectángulo. En cambio, Dlib requiere las coordenadas de los dos puntos de la diagonal del rectángulo, por lo que se realiza la operación que se ve en la línea 4 del anterior fragmento de código. Esta línea no se incorpora en el caso de haber realizado la detección con Dlib, pues el objeto que devuelve la detección HOG ya es válido.

Una vez ya se ha detectado la cara y se tiene un rectángulo que indica el área de la imagen en la que se encuentra el rostro, se aplica el predictor de Dlib a esta área para extraer las coordenadas. Las entradas para esta función son la imagen al completo en escala de grises y el rectángulo en el formato pertinente. La salida se debe convertir a un vector de coordenadas tal y como se ha explicado anteriormente, y esto se realiza con una función de la librería *face_utils*.

El cálculo de los indicadores se realiza seleccionando los puntos del mapa facial que representan cada ojo y la boca de acuerdo con la Figura 3 y llamando a las funciones declaradas al inicio del programa. Para el caso de los ojos, se calcula el EAR medio de los dos ojos.

A continuación, se evalúan estos indicadores en la siguiente estructura condicional:

```
# Analizar datos
  if EAR < threshold_pestanyeo: # Ojos entornados
      if tiempo_cerrados_ini==0: # Inicio somnolencia
          tiempo_cerrados_ini = time.time()
      tiempo_cerrados = time.time()-tiempo_cerrados_ini
      if tiempo_cerrados >= threshold_cerrados:
          # Mostrar alarma en el frame
          cv2.putText(frame,"ALERTA SOMNOLENCIA", (10, 30),
          cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
  else:
      tiempo_cerrados_ini = 0
      tiempo_cerrados = 0

  if MAR > threshold_bostezo: # Bostezando
      if tiempo_bostezo_ini==0: # Inicio del bostezo
          tiempo_bostezo_ini = time.time()
      tiempo_bostezo = time.time()-tiempo_bostezo_ini
      if tiempo_bostezo >= threshold_bostezo_tiempo:
          # Mostrar alarma en el frame
          cv2.putText(frame, "ALERTA BOSTEZO", (450, 30),
          cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
  else:
      tiempo_bostezo_ini = 0
      tiempo_bostezo = 0
```

Sí el EAR es menor que el límite especificado al inicio del programa se debe contabilizar el tiempo para ver si es necesario generar una alarma. Siendo esto un bucle, pueden ocurrir dos cosas. La primera, es que el EAR hasta este momento tenía un valor que no indicaba somnolencia. La segunda, es que en ejecuciones anteriores del bucle el EAR ya estaba por debajo del límite. En el caso de que sea la primera vez que el EAR es inferior al límite, se cumple que la variable "tiempo_cerrados_ini" es 0 (valor de inicialización). En ese caso, esta variable registra el tiempo (referido a tiempo de ejecución del programa) en el que se ha comenzado a detectar somnolencia.

A continuación, la variable "tiempo_cerrados" registra la diferencia de tiempo entre el instante en el que se encuentra el programa y el instante en el que se ha comenzado a detectar somnolencia. Si es el caso que se ha entrado al condicional por primera vez, la diferencia de tiempo será 0. Esta variable se evalúa seguidamente comparándola con el límite de tiempo que se da a los ojos cerrados para considerar que la causa es otra que no sea somnolencia. Si el tiempo acumulado es mayor, se genera la alarma. En este caso, tal y como se ha repetido a lo largo de la memoria, no se pretende actuar de ninguna manera sobre el conductor o el vehículo, pues esto es un dispositivo que simplemente detecta la somnolencia. Sin embargo, para comprobar que el funcionamiento es correcto, se ha programado una alarma visual que se muestra por pantalla durante los ensayos.

Si en cualquier momento el EAR deja de estar por debajo del límite, se reinicializan las variables a 0 para ejecutar esta estructura condicional de la misma forma en las siguientes ocasiones.

Lo que se ha explicado para el control del parámetro EAR se aplica de forma análoga para el parámetro MAR, considerando sus propias variables. Con esto se llega al final del bucle y solo queda finalizar el proceso de captura de vídeo mediante la función "*release*" de la librería OpenCV:

```
captura.release()
```

4.2.3. Prueba de funcionamiento

Para analizar el funcionamiento del programa y determinar su viabilidad antes de implementarlo en la Raspberry, se realizan una serie de pruebas para determinar cuál es la velocidad de funcionamiento y como de exitosa es la detección facial y la extracción de características faciales.

Para ello, se diseña una prueba en la que estando delante del monitor, con el rostro en posición frontal, se simula padecer fatiga con pesadez de párpados e incorporando algún bostezo. Se reproduce el vídeo en vivo y se dibuja encima el contorno de los ojos y de la boca uniendo las coordenadas faciales que extrae Dlib. Esto permitirá comprobar el buen funcionamiento de esta función. Además, se escriben sobre la imagen los indicadores y, en el momento en que salta una alarma, también se indica.

El programa se ejecuta siguiendo el siguiente esquema:

- 0-5s: Preparación
- 5-10s: Pesadez de párpados. Si funciona correctamente, debería saltar la alarma de somnolencia al ser un tiempo de 5s
- 10-15s: Bostezo. Si funciona correctamente, debería saltar la alarma de bostezo al ser un tiempo de 5s
- 15-25s: Bostezo combinado con pesadez de párpados
- 25s-final: Probar detección facial moviendo el rostro, inclinándolo, tapándolo para ver los límites

El resultado se puede ver en las siguientes imágenes capturadas:

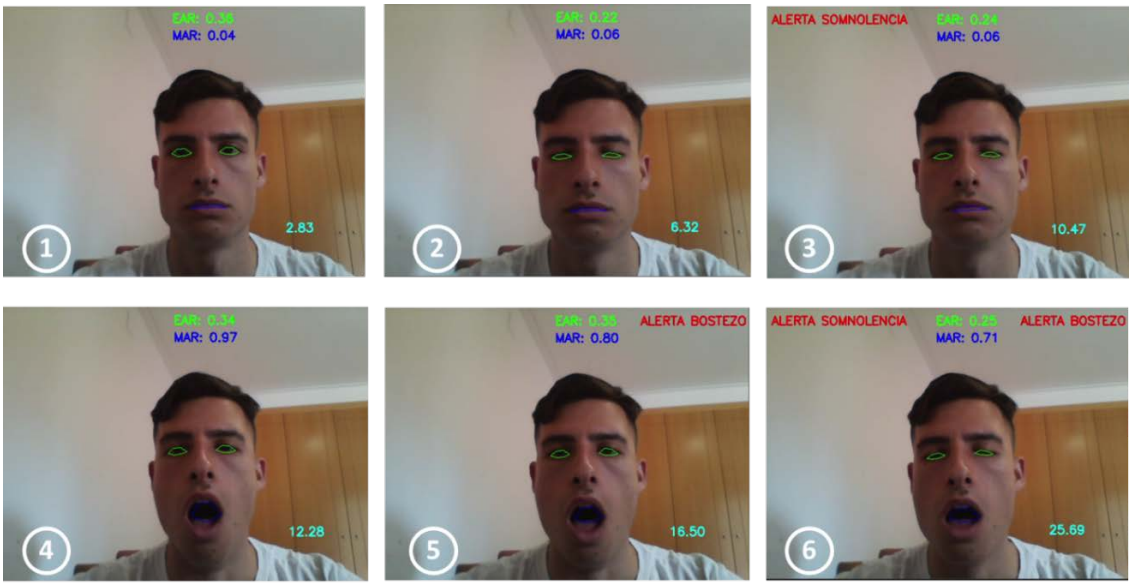

Figura 33. Capturas relevantes de la prueba del programa en Windows (método HOG)

Como se puede ver con los contornos dibujados, la detección de los ojos y la boca es exitosa y el contorno dibujado es correcto. El valor de los parámetros es coherente con la secuencia de tiempos establecida para el ensayo, así como las alarmas.

Otro tipo de información relevante a obtener es la velocidad de ejecución del programa y el éxito en la detección facial. Para ello, durante el experimento también se extraen los resultados en un fichero de texto que incorpora una columna de tiempo. Posteriormente, los datos se importan en Excel para poder procesarlos y se obtiene la siguiente información relevante:

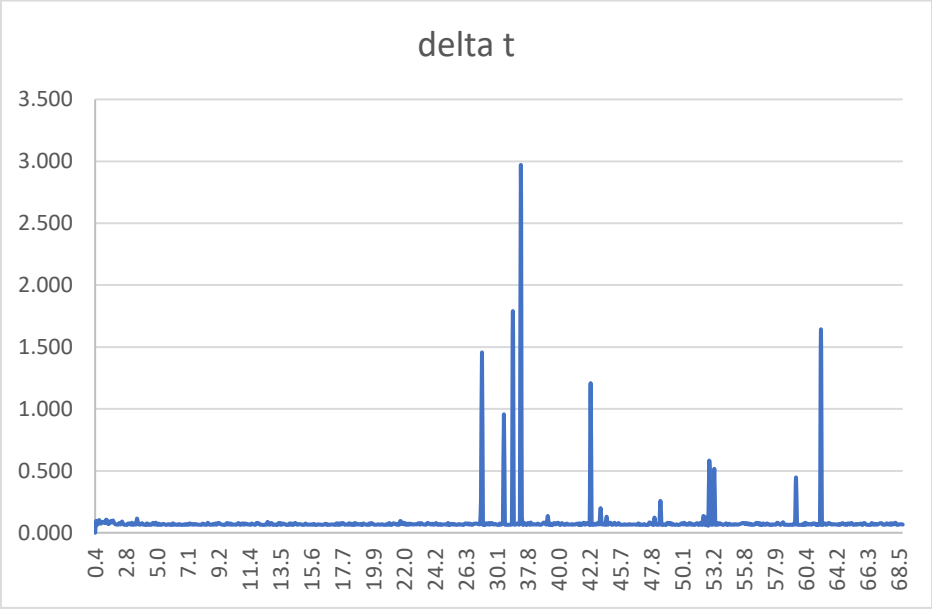

Figura 34. Información de tiempo del ensayo

El promedio del incremento de tiempo es de 0,084s, es decir, el programa ejecutándose en un PC es capaz de analizar un *frame* de vídeo cada 0,084s y no sólo de eso, sino también de mostrar el vídeo con los parámetros y extraer un fichero de texto. Además, este valor medio incluye los picos que se ven en el gráfico, valores que no representan la velocidad de ejecución del programa pues lo que representan es el tiempo que ha transcurrido sin que se haya detectado ningún rostro (probado en la última fase de la prueba).

El fichero de texto está programado de tal forma que si no se detecta una cara, no se guarda ningún dato. Por lo tanto, durante una ejecución con detecciones exitosas, el valor de incremento de tiempo (tiempo que tarda el bucle en ejecutarse) se mantiene bajo y constante alrededor de 0,07 segundos en todas las iteraciones. Si no se detecta ningún rostro el programa se sigue ejecutando sin guardar ningún dato hasta que se detecte de nuevo, lo que provoca que se vuelvan a guardar datos habiendo pasado un lapso de tiempo desde el último guardado. De esta forma, prestando atención a los incrementos de tiempo fuera de la media, es posible saber cuándo la detección facial ha fallado.

Con pruebas de este tipo se puede considerar que el programa funcionando en un PC es eficaz, pues realiza la detección de somnolencia en tiempo real. Respecto al criterio de funcionalidad, en el PC no se puede probar la detección en condiciones de baja luminosidad por no estar la webcam preparada para ello, pero con la prueba realizada en la que se ha intentado respetar las condiciones que se

encontrarían dentro del habitáculo del vehículo durante la conducción se ve que la detección de somnolencia funciona correctamente. Este análisis no es una verificación del cumplimiento de criterios, los criterios se deben verificar una vez el sistema está acabado con pruebas más completas, pero es útil para valorar el desarrollo del proyecto y obtener una idea de su viabilidad.

4.3. IMPLEMENTACIÓN EN RASPBERRY

4.3.1. Instalación de librerías

Igual que en el PC, en la Raspeberry Pi se debe empezar instalando las librerías de Python necesarias para la ejecución del programa. El sistema operativo Raspberry Pi OS viene con la instalación por defecto de Python 2.7.16 y de Python 3.7.13 en el momento de la ejecución del proyecto, y el programa de detección de somnolencia se va a desarrollar en Python 3.7.13 por ser una versión más cercana a la usada en Windows. Dado que la versión de Python que emplea la Raspberry por defecto es la 2.7.13, durante la instalación de las librerías las órdenes deberán ir precedidas del comando "python3" para que la acción se ejecute para la versión de Python 3.7.13.

Al igual que se hizo durante la instalación de librerías en Windows, lo primero que se debe hacer en la Raspberry es actualizar la versión del administrador de paquetes de Python "pip" mediante el comando `python3 -m pip install --upgrade pip`.

Seguidamente, se procede a instalar las librerías. La instalación de OpenCV en la Raspberry Pi es algo más costosa que en Windows debido a los prerrequisitos que tiene para su instalación y las dependencias de la librería. En la instalación mediante pip, el control que tiene el usuario es menor, y aunque la instalación de OpenCV puede ser exitosa, su funcionamiento da lugar a errores debido a que requiere de muchas otras librerías que no se han instalado. Es por ello que la instalación se ha realizado con la ayuda de la librería CMake, diseñada para controlar el proceso de compilación de software y que va a permitir instalar todas las dependencias de OpenCV.

El proceso de instalar OpenCV mediante CMake se resume en los siguientes pasos:

- Actualizar y mejorar todos los paquetes del sistema operativo de la Raspberry con los siguientes comandos:
    - `apt-get update`: Actualiza la lista de paquetes disponibles para Raspberry Pi OS, así como sus versiones
    - `apt-get upgrade`: Instala las nuevas versiones encontradas mediante el comando anterior de los paquetes instalados en el sistema operativo

- Instalar todas las dependencias de OpenCV, es decir, todos los paquetes que hacen falta para que funcione. De los paquetes mostrados a continuación, los esenciales se indican en la página web de OpenCV. Además, se han instalado algunos otros.
    - `apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev`: Instala paquetes necesarios para la gestión de imágenes

- `apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev libx264-dev`: Paquetes de códecs para vídeos, paquetes para el tratamiento de vídeos, etc.
- `apt-get install libgtk2.0-dev libgtk-3-dev`: Instala paquetes GTK (versiones 2 y 3) necesarios para compilar el módulo de OpenCV que permite mostrar imágenes por pantalla, da soporte a la cámara, etc.
- `apt-get install libatlas-base-dev gfortran`: Para optimizar las operaciones de OpenCV
- `apt-get install python3-dev`: Como se ha comentado, OpenCV está desarrollada en C++. Este paquete provee el código necesario para que Python use librerías desarrolladas en otro lenguaje de programación

- Descargar el código fuente de OpenCV 4.5.1 (la misma versión empleada en Windows) desde la página web de OpenCV (https://opencv.org/releases). Además, se recomienda descargar el paquete "opencv-contrib" con módulos extra también disponible en el repositorio oficial

- Instalar el paquete NumPy mediante pip. OpenCV requiere una versión de NumPy superior a la 1.14.5, y la librería SciPy requiere una versión menor que la 1.23.0 y mayor que la 1.16.5. En este caso, pip ha instalado la versión 1.20.2 que satisface ambos requisitos

- Instalar CMake: `apt-get install build-essential cmake pkg-config`

- Compilar OpenCV con CMake desde los archivos descargados. Si hay alguna dependencia que no se haya satisfecho, es probable que se descargue durante la ejecución de este comando. Los comandos a ejecutar son los siguientes:

  - `cd ~/opencv-4.5.1`
  - `mkdir build`
  - `cd build`
  - `cmake -D CMAKE_BUILD_TYPE=RELEASE\`
  `-D CMAKE_INSTALL_PREFIX=/usr/local\`
  `-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-4.5.1/modules\`
  `-D ENABLE_NEON=ON\`
  `-D ENABLE_VFPV3=ON\`
  `-D BUILD_TESTS=OFF\`
  `-D WITH_TBB=OFF\`
  `-D INSTALL_PYTHON_EXAMPLES=OFF\`
  `-D BUILD_EXAMPLES=OFF`
  - `make -j4`

- Finalmente, se instala OpenCV escribiendo `make install`

Cabe decir que todos los comandos escritos en la consola de Raspberry deben ir precedidos de la palabra "sudo", que otorga permisos de *superuser* necesarios para realizar cambios en el sistema.

Una vez superada la instalación de OpenCV, el resto de paquetes se instalan de forma sencilla mediante pip. Dlib se instala en su versión 19.22.0, SciPy en la 1.6.3 , imutils en la 0.5.4 y NumPy se instala automáticamente en una versión adecuada al instalar SciPy.

4.3.2. Habilitación cámara Raspberry

Hacer uso de la cámara en un PC portátil es inmediato pues la webcam viene integrada y simplemente hay que indicar en el programa que se haga uso de la cámara por defecto. En la Raspberry, al ser la cámara un accesorio, hay que configurarla. Primeramente, hay que habilitar la comunicación entre la Raspberry y la cámara accediendo a la configuración de la Raspberry con el comando `sudo raspi-config`. Una vez en el menú, se acceden a las opciones de interfaz, se selecciona la cámara, se habilita y se reinicia la Raspberry para que los cambios surjan efecto.

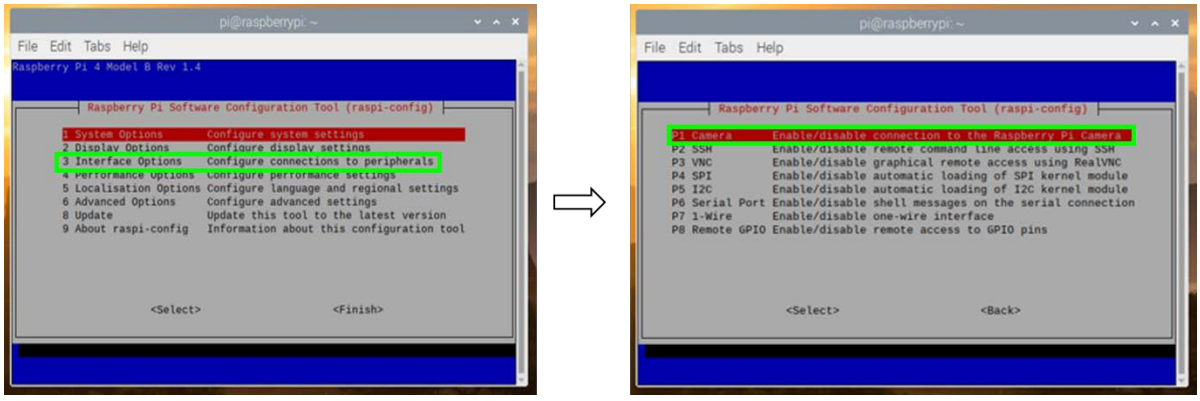

Figura 35. Habilitación cámara en Raspberry desde el menú de configuración

Para comprobar que la cámara funciona correctamente, se pueden tomar imágenes o capturar vídeo desde la consola de comandos “raspistill” o “raspivid”. A continuación, una buena práctica es ejecutar los comandos “apt-get” que se han explicado anteriormente, por si hubiese paquetes nuevos o actualizados. Como se ha comentado, estas órdenes se deben ejecutar con permisos de *superuser*. Finalmente, para poder controlar la cámara con Python, es necesario instalar el paquete “*picamera*” que provee la interfaz necesaria.

4.3.3. Programación en Raspberry

La programación del algoritmo en la Raspberry se ha realizado en el entorno de THONNY y se han tenido que realizar algunas modificaciones respecto al código desarrollado en Windows, fundamentalmente debidas al control de la cámara. A continuación, se van a exponer las diferencias:

- Referente a las librerías, se debe importar la ya mencionada “*PiCamera*” para poder controlar la Raspberry Pi NoIR Camera v2 desde Python
- La inicialización de la captura se hace de la siguiente forma:

```
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 32
captura = PiRGBArray(camera, size=(640, 480))
time.sleep(2)
```

El "*framerate*" no es importante en este caso. Se ha configurado la cámara para que capture 32 *frames* por segundo, pero el programa no puede procesar más de 5 por segundo. La función "*PiRGBArray*" permite a partir de la señal de la cámara construir un vector que contiene la información de la imagen.

- El primer bucle del programa (en Windows era un *while True*), en la Raspberry debe escribirse de la siguiente forma:

```
for frames in camera.capture_continuous(captura, format="bgr",
use_video_port = True):
      frame = frames.array
```

La función "*capture_continuous*" de la librería *PiCamera* se comporta como un iterador infinito de imágenes capturadas continuamente por la cámara. El formato se ha establecido en BGR, por ser el mismo tipo que emplea OpenCV. El parámetro "*use_video_port*" se define como *True* para que la cámara use el puerto de vídeo y no el de imagen. Usar el puerto de imagen resulta en imágenes de mayor calidad, pero esto ralentiza el programa tal aumentando el tiempo del bucle en unos 0,2 segundos.

El resto del código es igual que el desarrollado en Windows

4.3.4. Pruebas de funcionamiento

De forma análoga a lo expuesto en el capítulo 4.2.3, en este apartado se exponen las pruebas estáticas llevadas a cabo con el sistema fuera del vehículo para comprobar que está listo para ser montado. En este caso, al ser pruebas sobre el sistema que se va a montar en el vehículo, las pruebas realizadas son más exhaustivas. Por un lado, debe analizarse el comportamiento del programa usando detección Haar y usando detección HOG, pues siendo la Raspberry un dispositivo de menor potencia que el PC, cobra más importancia el coste computacional de cada método. Además, conviene explorar de forma más completa los límites de la detección facial, pues se debe tener una idea clara previo instalación de la cámara en un vehículo.

A continuación, se van a exponer todas las facetas que se han analizado en las pruebas y el resultado al realizar la detección con cascadas de Haar y con histograma de gradientes orientados:

- Velocidad de procesamiento: Al igual que en las pruebas ejecutadas con el programa en Windows, durante estas pruebas se ha extraído una variable de tiempo para poder monitorizar el avance del proyecto. El resultado es bien distinto según el método de detección que se use. Hay que tener en cuenta que, en los ensayos realizados, se han programado tareas que consumen tiempo como por ejemplo guardar un fichero de texto con resultados, mostrar por pantalla la imagen de lo que está viendo el programa y guardar en vídeo esta imagen. Por lo tanto, las diferencias entre el coste computacional de ambos mecanismos de detección facial se ven atenuados por haber tareas que engrosan la duración del bucle, si esta misma comparación se llevara en funcionamiento real, las diferencias serían mayores.

Los resultados de duración de la ejecución del bucle son, en promedio, de 0,218 s para la detección con cascadas de Haar y de 0,375 s para la detección con HOG. Por lo tanto, tal y como se desprende de la revisión bibliográfica realizada, la detección con clasificadores de Haar es más rápida, en concreto para este proyecto es un 42 % más rápida. Si bien la velocidad es una ventaja, esto no descarta al clasificador de HOG como método de detección, pues el tiempo de cálculo sigue permitiendo hacer una detección en tiempo real

- Éxito en la detección facial: El sistema se diseña para captar el rostro desde una posición frontal o cercana a ella. Así y todo, esto no evita que los detectores faciales fallen en alguna iteración del bucle. Además, con la intención de ser más fiel al montaje final, en los ensayos llevados a cabo fuera del coche la cámara se ha montado con un cierto ángulo hacia el rostro, por lo que la captura no es del todo frontal.
  Realizar una comparación de las tasas de fallo de ambos detectores permite obtener una idea de cuál es más ventajoso para la robustez del sistema. De los datos extraídos se desprende que estando el rostro orientado como se espera, se detecta con ambos detectores casi la totalidad de las veces. Cuando el rostro cambia la orientación, el clasificador de HOG de Dlib permite más margen antes de fallar en la detección.
  Sin embargo, la extracción de coordenadas de los ojos con el rostro girado no es buena, quedando únicamente los datos de bostezo como fuente de información fiable cuando el rostro está girado. Por lo tanto, se considera que el programa no es robusto para detectar somnolencia cuando el rostro está girado, lo cual se conocía de antemano debido a la investigación documental desarrollada.
  Ante la oclusión del rostro, no hay demasiada diferencia entre ambos clasificadores. Ambos son capaces de detectar una cara dentro de ciertos límites. El usar complemento como gafas o gorra no impide que se detecte la cara, pero afecta al predictor de coordenadas de Dlib. El uso de gorra puede provocar en alguna ocasión que la extracción de coordenadas se vea perjudicada. El sistema de detección de somnolencia es totalmente compatible con el uso de gafas de ver. En cuanto a gafas de sol, evidentemente impiden el análisis del ojo. A continuación, se muestran unas imágenes que muestran que la detección es compatible con el uso de complementos:

Figura 36. Ejecución del programa basado en la detección facial mediante HOG durante el uso de complementos

- Falsos positivos: De acuerdo con lo visto en el capítulo 2.4, la detección facial mediante cascada de clasificadores de Haar es más propensa a los falsos positivos que la detección mediante histograma de gradientes orientados. Se ha comprobado que esto es cierto, aunque no es alarmante. Además, dado que la detección facial mediante OpenCV tiene parámetros ajustables como por ejemplo el número mínimo de vecinos, se puede mejorar la detección. Sin embargo, es mejor posponer el ajuste de parámetros a tener el sistema instalado en un vehículo. Se adjuntan imágenes de falsos positivos a continuación:

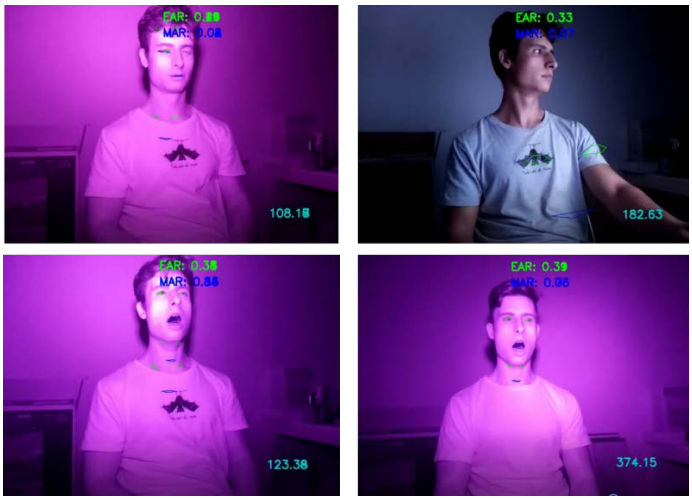

Figura 37. Ejemplo de falsos positivos obtenidos con la detección mediante clasificadores de Haar

4.4. MONTAJE EN VEHÍCULO

Lo primero que se debe hacer antes de montar el dispositivo en el vehículo es asegurarse que el programa se va a ejecutar. Durante las pruebas fuera del vehículo, se montan periféricos en el dispositivo como pantalla o ratón que permiten interactuar con el sistema. Por el contrario, una vez montado en el vehículo, el sistema debe de ser plenamente funcional por sí mismo.

Por ello, se debe configurar la Raspberry Pi para que al encenderse ejecute directamente el programa. De esta forma, estando alimentada a través del coche, cuando el coche se encienda también lo hará la Raspberry y el sistema comenzará a funcionar automáticamente.

Hay varias maneras de hacer que la Raspberry ejecute instrucciones durante su arranque. La más completa que permite ejecutarse en el inicio, considerar dependencias y permanecer ejecutándose en segundo plano una vez ha finalizado el arranque del sistema consiste en crear un "servicio". Esto se hace mediante el proceso "*Systemd* ", una herramienta estándar de Linux que permite inicializar y controlar los servicios (por ejemplo sistema de ficheros, redes, display, etc.) durante el arranque y mientras el sistema está activo.

Para crear el servicio se puede introducir en la consola de comandos de Raspberry Pi OS la orden `sudo nano lib/systemd/system/instrucciones_arranque.service`. Esto crea el archivo "instrucciones_arranque.service" en el directorio indicado para que sea leído por *Systemd* y lo abre mediante "nano", el editor de texto para la consola que incluyen normalmente los sistemas operativos tipo Unix:

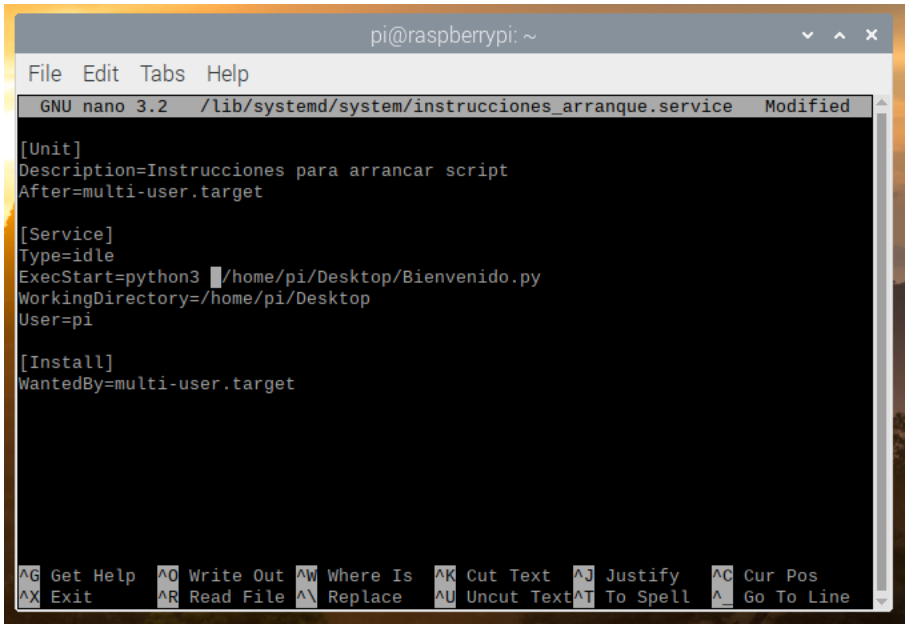

Figura 38. Edición del archivo instrucciones_arranque.service con nano

Se ha configurado de tal forma que el programa empiece a ejecutarse una vez está disponible el entorno multi-usuario. Además, el tipo se ha establecido como "*idle*", lo que hará que la orden en "*ExecStart*" se ejecute cuando el resto de procesos hayan terminado. La orden en sí se escribe en la línea de *ExecStart*, y como se puede ver se ha indicado que el script "Bienvenido.py" se va a ejecutar con Python 3.

A continuación, se da permiso al archivo para ejecutarse durante el arranque del sistema: `sudo systemctl enable instrucciones_arranque.service`

Para alimentar los equipos del sistema se requieren distintas tensiones. Por un lado, la Raspberry Pi funciona a 5,1 V, pero los equipos y cargas de un automóvil suelen funcionar a 12 V por ser esta la tensión de la batería. Por ello, para realizar esta transformación de tensión se requiere un convertidor reductor. La mayoría de coches cuentan con tomas a 12 V con el formato convencional de mechero de automóviles, pero en este caso no se ha empleado esta toma por ser el contacto deficiente, sino que se ha aprovechado un equipo convertidor 12/5 para conectarlo directamente a 12 V empleando la alimentación dedicada a la luz de cortesía.

En cuanto a la lámpara IR necesaria para poder ejecutar el programa en la oscuridad, funciona a 12 V. Por ello, no se ha requerido ningún equipo intermedio y se ha conectado directamente a la alimentación de un equipo no esencial.

Por lo que concierne a la ubicación de los equipos en el vehículo, es un aspecto delicado que debe ser estudiado específicamente para el vehículo en el que se vaya a montar. Tal y como se ha comentado, los mecanismos de detección facial empleados trabajan óptimamente cuando el rostro se encuentra en posición frontal, por lo que hay que instalar la cámara lo más centrada con el asiento del conductor posible. Algunos elementos sobre los que se podría montar la cámara son la visera parasol o el cuadro asegurando que el volante no tapa el rostro. Dependiendo del vehículo, puede ser que el salpicadero ofrezca una buena posición de montaje.

4.5. ENSAYOS

Los ensayos llevados a cabo en carretera pretenden cubrir, dentro de las posibilidades, un conjunto de situaciones a las que se puede enfrentar el programa: variabilidad en la luminosidad, uso de complementos, distintas posturas del asiento, distintos sujetos, etc. Cada variante del programa de detección de somnolencia va a ser estudiada y se van a extraer sus puntos fuertes y débiles.

En primer lugar, se ha realizado un montaje en el que la cámara se sitúa encima del salpicadero y orientada al conductor. Esto da como resultado que el rostro y la cámara no están alineados, por lo que el proceso de detección facial se ve perjudicado fallando en múltiples ocasiones. Se ha comprobado que con este montaje, la regulación del asiento del conductor influye mucho en el funcionamiento del programa. Una postura alejada del volante en combinación con el ángulo entre el rosto y la cámara hacen que la detección sea casi imposible, mientras que con una postura más cercana se consigue un mayor éxito.

Con una postura de conducción cercana a la cámara y usando detección facial mediante histograma de gradientes orientados, se han alcanzado tasas de detección de hasta el 96%. En cambio, con una postura de conducción alejada, la tasa de detección disminuye hasta el 70%. La velocidad de procesamiento se sitúa en torno a las 2,81 imágenes por segundo.

Empleando el detector basado en clasificadores de Haar, los resultados son muy variados. Se han obtenido tasas de detección desde el 27% hasta el 80%, con algunos resultados muy positivos y otros muy negativos. Como punto positivo, se debe destacar la ya mencionada velocidad de este método de clasificación, llegando a procesar hasta 4,76 imágenes por segundo. Además, se ha logrado realizar una buena monitorización del conductor en la oscuridad, cosa que no se ha logrado con el clasificador HOG. Sin embargo, estos resultados obtenidos no se pueden contemplar como válidos en el marco de este trabajo, pues han ido acompañados de otros resultados muy negativos como por ejemplo los abundantes falsos positivos o la caída de la tasa de detección a valores del 27%.

Tras esta primera aproximación al montaje en vehículo en la que los resultados no se pueden considerar admisibles por las altas tasas de fallo presentadas en algunos ensayos realizados con ambos métodos de detección, se ha asignado un mayor peso en el proyecto a la ubicación de la cámara. Estando los clasificadores entrenados para capturar rostros de frente, la cámara debe ir montada enfrente del conductor. Esto puede ser complicado, pues no se debe obstaculizar la visión. En este caso, se ha montado el equipo en la visera parasol. Además, para la detección con cascadas de Haar se ha tratado de reducir la aparición de falsos positivos aumentando el parámetro del mínimo de vecinos para considerar que hay un rostro.

Debido a los requerimientos del programa, los esfuerzos de los ensayos se han centrado mayormente en lograr un sistema capaz de monitorizar al conductor y demostrar que es capaz de detectar somnolencia, pero no se han realizado suficientes ensayos como para validar la aptitud del sistema para esta función. Para obtener un sistema fiable en la detección de somnolencia es necesario invertir tiempo en pruebas del sistema con distintos sujetos y en distintas situaciones y usar un método de validación como puede ser la monitorización paralela con EEG. Sin embargo, por cuestiones de tiempo, esta tarea no puede enmarcarse en el presente proyecto, pues un testeo exhaustivo requiere más tiempo del invertido en desarrollar el prototipo.

A continuación, se presenta el formato de ensayo que se ha llevado a cabo y que se debería repetir de forma repetitiva con varios sujetos y variando las condiciones expuestas en cada ensayo. También se presenta un análisis de los resultados obtenidos para identificar los puntos fuertes y débiles del sistema, un análisis que debería repetirse durante la realización masiva de ensayos para mantener un seguimiento de las variables destacadas del programa. La duración de cada ensayo es de 20 minutos, combinando intervalos de conducción en autopista con intervalos de conducción urbana. La velocidad de computación y la tasa de detección facial se ha calculado considerando intervalos de 5 minutos dentro del propio ensayo, en los que las condiciones de conducción se mantienen uniformes.

<table>
<tr><th>ENSAYO</th><th colspan="6">CONDICIONES</th><th colspan="3">RESULTADOS</th></tr>
<tr><td rowspan="9">ENSAYO 1</td><td rowspan="3">SOMNOLENCIA</td><td colspan="5" rowspan="3">★★★★☆</td><td rowspan="3">Velocidad computación (fps)</td><td>Max</td><td>2,73</td></tr>
<tr><td>Min</td><td>2,67</td></tr>
<tr><td>Med</td><td>2,69</td></tr>
<tr><td rowspan="3">LUMINOSIDAD</td><td colspan="5" rowspan="3">Baja</td><td rowspan="3">Tasa detección facial (% detección positiva)</td><td>Max</td><td>95,51</td></tr>
<tr><td>Min</td><td>83,90</td></tr>
<tr><td>Med</td><td>90,26</td></tr>
<tr><td rowspan="3">DETECCIÓN FACIAL</td><td colspan="5" rowspan="3">Histograma de Gradientes Orientados</td><td rowspan="3">Comentarios</td><td colspan="2" rowspan="3">El ajuste de parámetros EAR es válido. Se han generado 38 alarmas de somnolencia basadas en ojos entornados
El ajuste de parámetros MAR es mejorable. Se han detectado algunos falsos bostezos y no se ha detectado algún bostezo</td></tr>
<tr></tr>
<tr></tr>
</table>

<table>
<tr><th>ENSAYO</th><th colspan="6">CONDICIONES</th><th colspan="3">RESULTADOS</th></tr>
<tr><td rowspan="3">ENSAYO 2</td><td>SOMNOLENCIA</td><td colspan="5">  </td><td rowspan="3">Velocidad computación (fps)</td><td>Max</td><td>3,61</td></tr>
<tr><td colspan="5"></td><td>Min</td><td>3,40</td></tr>
<tr><td colspan="5"></td><td>Med</td><td>3,47</td></tr>
<tr><td>LUMINOSIDAD</td><td colspan="5">Baja</td><td rowspan="3">Tasa detección facial (% detección positiva)</td><td>Max</td><td>95,10</td></tr>
<tr><td colspan="5"></td><td>Min</td><td>55,60</td></tr>
<tr><td colspan="5"></td><td>Med</td><td>76,08</td></tr>
<tr><td>DETECCIÓN FACIAL</td><td colspan="5">Cascada de clasificadores Haar</td><td>Comentarios</td><td colspan="2">La detección del rostro falla con facilidad
Se han dado falsos positivos recurrentemente, lo cual invalida los resultados</td></tr>
<tr><td rowspan="3">ENSAYO 3</td><td>SOMNOLENCIA</td><td colspan="5">  </td><td rowspan="3">Velocidad computación (fps)</td><td>Max</td><td>2,68</td></tr>
<tr><td colspan="5"></td><td>Min</td><td>2,67</td></tr>
<tr><td colspan="5"></td><td>Med</td><td>2,67</td></tr>
<tr><td>LUMINOSIDAD</td><td colspan="5">Alta</td><td rowspan="3">Tasa detección facial</td><td>Max</td><td>99,13</td></tr>
<tr><td colspan="5"></td><td>Min</td><td>84,07</td></tr>
<tr><td colspan="5"></td><td>Med</td><td>93,10</td></tr>
<tr><td>DETECCIÓN FACIAL</td><td colspan="5">Histograma de Gradientes Orientados</td><td>Comentarios</td><td colspan="2">El ajuste de parámetros es válido. No se ha generado ni una sola alarma</td></tr>
</table>

<table>
<tr><th>ENSAYO</th><th colspan="6">CONDICIONES</th><th colspan="3">RESULTADOS</th></tr>
<tr><td rowspan="5">ENSAYO 4</td><td rowspan="2">SOMNOLENCIA</td><td colspan="5" rowspan="2">★★★☆☆</td><td rowspan="2">Velocidad computación (fps)</td><td>Max</td><td>3,81</td></tr>
<tr><td>Min</td><td>3,66</td></tr>
<tr><td rowspan="2">LUMINOSIDAD</td><td colspan="5" rowspan="2">Alta</td><td rowspan="2">Tasa detección facial (% detección positiva)</td><td>Med</td><td>3,71</td></tr>
<tr><td>Max</td><td>98,73</td></tr>
<tr><td>DETECCIÓN FACIAL</td><td colspan="5">Cascada de clasificadores Haar</td><td>Comentarios</td><td>Min</td><td>94,76</td></tr>
<tr><td colspan="3">Med</td><td colspan="3">96,53</td><td colspan="3">El ajuste de parámetros EAR es válido. Se han generado 16 alarmas de somnolencia basadas en ojos entornados
El ajuste de parámetros MAR es correcto, no ha habido bostezos y no se han detectado falsos bostezos
No hay falsos positivos</td></tr>
</table>

Tabla 2. Resumen ensayos durante conducción

Los resultados numéricos mostrados en la tabla superior se han extraído del programa en un fichero de texto con formato de separación por tabulador y se han analizado con Excel. Para conocer ciertamente el desarrollo del ensayo, se ha extraído un vídeo que guarda cada uno de los *frames* captados por la cámara. Nótese que la somnolencia es una apreciación subjetiva del individuo y que, como se ha comentado, estos ensayos se deberían llevar a cabo contrastando los resultados con un sistema fiable como puede ser la medición de señales EEG.

En líneas generales y enmarcando todos los ensayos realizados, durante una conducción por poblado en la que el conductor debe mirar a través de las ventanas y retrovisores frecuentemente, el sistema presenta demasiadas interrupciones por los fallos momentáneos de detección facial.

A continuación, se pretende demostrar el buen o el mal funcionamiento del sistema con extractos de los vídeos recopilados. Se combinan imágenes de los ensayos realizados con distintos clasificadores y con distintas condiciones de luminosidad y somnolencia:

Figura 39. Correcto funcionamiento con el uso de complementos (mascarilla, gorra, gafas)

La imagen izquierda muestra la correcta detección facial aún con una oclusión parcial de los clasificadores de Haar (según la literatura al respecto, menos robustos frente a oclusión que los clasificadores HOG). Además, las coordenadas extraídas de los ojos son correctas. En la imagen central y derecha, de forma similar a lo mostrado en la Figura 36, se da una situación de uso de complementos.

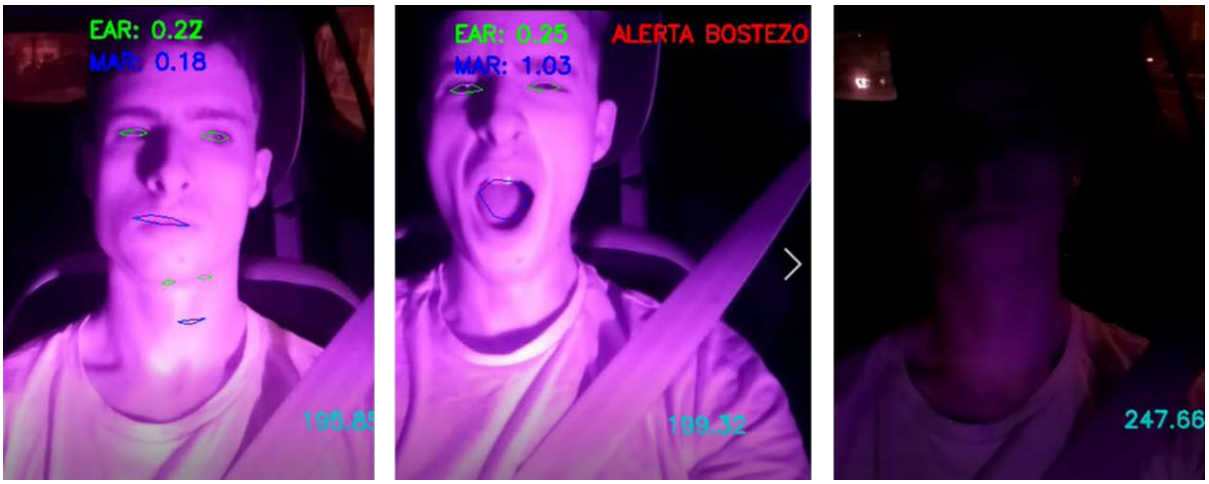

Figura 40. Falso positivo con detección Haar, detección de un bostezo con HOG (en medio), fallo de la lámpara IR (derecha)

En la imagen superior izquierda se puede ver el error recurrente del clasificador con cascadas de Haar. El ajuste del parámetro del mínimo de vecinos del clasificador puede ser válido para condiciones de buena iluminación (ensayo 4 de la Tabla 2) , pero malo para condiciones de oscuridad (ensayo 2 de la Tabla 2). En la imagen central, se muestra un ejemplo del programa funcionando en la oscuridad con el detector HOG. A la derecha, un fallo recurrente de la lámpara de infrarrojos durante la circulación urbana. Si bien el entorno está oscuro, la captación de luz por parte de la LDR de la lámpara provoca que se apague dejando al sistema inoperativo momentáneamente.

5. CONCLUSIONES Y TRABAJOS FUTUROS

A continuación, se van a exponer las conclusiones extraídas de la experiencia acumulada y de los ensayos realizados.

5.1. CONCLUSIONES

Tras los ensayos realizados y llegando al fin del desarrollo de este trabajo, es el momento de hacer una revisión de los criterios de validez, eficacia y coste definidos en el apartado 3.1. Mediante estos, se cuantificará el éxito del proyecto.

5.1.1. Criterio de validez

Se ha demostrado que el sistema desarrollado es muy apto para el funcionamiento en estático, es decir, cuando el rostro permanece relativamente inmóvil. Esto se ha podido comprobar tanto durante la realización de los ensayos fuera del vehículo como con los resultados de los ensayos llevados a cabo en el vehículo, en los cuales la monitorización del conductor es mucho mejor durante una circulación por una vía recta que durante una circulación por poblado, vías con cruces o muchas curvas.

Durante una conducción en la que el conductor debe estar atento al entorno constantemente (y por ende, moviendo el rostro) ya sea porque hay semáforos, pasos de peatones, curvas, cruces, etc. el sistema no es válido, pues no puede llevar a cabo la monitorización del conductor de forma continuada.

El uso de complementos como gorras o gafas de vista no supone un obstáculo para el funcionamiento del programa, si bien no puede funcionar cuando el conductor usa gafas de sol, pues no se pueden monitorizar los ojos. Así mismo, cuando el conductor usa mascarilla, el programa es capaz de detectar el rostro y de monitorizar los ojos.

Las distintas condiciones de luminosidad de sortean de forma efectiva mediante la lámpara de infrarrojos, que permite el buen desempeño del programa en la oscuridad. Sin embargo, en circulación nocturna y por trayectos iluminados con farolas, el sistema no funciona correctamente debido a que la lámpara de infrarrojos se enciende y se apaga a su paso por las farolas.

Con un montaje adecuado, el rostro se captura correctamente independientemente de la distancia del conductor a la cámara, por lo que la postura en el asiento no influye en el buen funcionamiento del sistema.

La detección de somnolencia debe contemplar la variación de rasgos entre personas. Se ha realizado un programa a medida para un conductor y funciona correctamente, sin embargo su funcionamiento podría no ser adecuado con otros conductores.

5.1.2. Criterio de eficacia

El haber alcanzado tasas de análisis de hasta 5 fps durante los ensayos en estático y de 4 fps en ensayos en el coche permite justificar el cumplimiento de este criterio. Independientemente de la variante de programa empleada, se puede considerar que la monitorización del conductor se lleva a cabo en vivo.

5.1.3. Criterio de coste

La siguiente tabla muestra el coste de los equipos implicados en el sistema:

CUADRO DE PRECIOS BÁSICOS: RECURSOS MATERIALES			
Código	Unidad	Descripción	Precio(€)
RM.RP	ud	Coste Raspberry Pi 4 Model B 8GB	85
RM.CAM	ud	Coste Raspberry Pi Camera NoIR v2	27
RM.TM	ud	Coste tarjeta micro SDHC 32 GB 100 MB/s	8
RM.PS2	ud	Coste cable de alimentación USB a USB tipo C	8
RM.PS3	ud	Coste cargador para coche 12 V	10
RM.IR	ud	Coste lámpara de infrarrojos	7
		Total	145.00 €

Tabla 3. Cuadro de precios de los equipos que componen el sistema de detección de somnolencia

El precio de los equipos es inferior al margen de 300-800 € mostrado en el capítulo 3.1.3, por lo que se puede considerar que se cumple el criterio de coste. No se han considerado los costes no recurrentes como puede ser el tiempo de desarrollo del prototipo o el tiempo necesario para a partir del prototipo obtener un sistema sólido debido a que lo que el coste total no sería fiel al coste que tendría este producto en el mercado si se produjese en masa.

Dado el precio de 145 € del sistema, es viable por parte de los fabricantes de automóviles ofrecer un equipamiento de estas características como extra en sus vehículos, así como la comercialización de un equipo de esta tipología como accesorio para el vehículo.

A modo de cierre de las conclusiones, se ha realizado un sistema de detección de somnolencia en la conducción concebido para situarse en frente del conductor, mediante una Raspberry Pi y una cámara que permite monitorizar al conductor la mayoría de las veces, tanto en condiciones de luminosidad como de oscuridad y siendo robusto frente al uso de complementos y frente a la posición del conductor. Es especialmente útil para la conducción en carretera, autovía o autopista, pues el conductor desarrolla somnolencia con mayor facilidad en estas situaciones y su monitorización es más sencilla al estar mirando hacia delante la mayoría del tiempo.

A fecha de entrega de la presente memoria y con las herramientas empleadas, el uso de la detección facial mediante Histograma de Gradientes Orientados proporciona en general mejores resultados que la detección facial mediante cascadas de clasificadores de Haar. Si bien la detección del rostro es más lenta, la monitorización del conductor se puede realizar en vivo y es mucho más robusta frente a condiciones de luminosidad, además de ser más sencilla por no requerir ningún ajuste para sortear un problema de falsos positivos.

En el siguiente capítulo se presenta una propuesta de hoja de ruta que se debería seguir para lograr un sistema de detección de somnolencia en la conducción que cumpliera totalmente con el criterio de validez y que fuese comercializable.

5.2. TRABAJOS FUTUROS

Durante los estadios finales del proyecto, con la experiencia acumulada y una perspectiva más amplia del desarrollo, aparecen ideas para mejorar el proyecto.

En primer lugar, se debe aclarar que los ensayos realizados son escasos y con recursos limitados. La primera tarea a hacer para continuar el proyecto debería ser un ensayado exhaustivo del sistema con varias personas y vehículos. Para ello, se podría seguir la estructura de ensayo plantada en el capítulo 4.5, instalando el sistema en el vehículo de distintos sujetos, ajustando los parámetros pertinentemente y obteniendo una valoración del estado de somnolencia de la persona en el momento de realizar el ensayo. Esto es una tarea que requiere de una población amplia para el ensayo y de una inversión de tiempo considerable.

Además, con el objetivo de validar el sistema de detección de somnolencia, se deberían llevar a cabo pruebas combinadas con casco de EEG como el presentado en 1.2.1. De esta forma, se podría evaluar el estado del sujeto de una forma fiable y compararlo con el resultado del programa realizado en este proyecto.

En segundo lugar, se pueden mejorar las herramientas empleadas en el sistema de forma que aumente la robustez. Con las pruebas realizadas en el capítulo 4.5, ya se pueden ver algunos campos de mejora. Siendo el criterio de validez el que menos se ha satisfecho, se proponen las siguientes medidas para perfeccionar el sistema:

- Uso de una lámpara de infrarrojos cuya activación requiera de menos oscuridad que la lámpara usada. De esta forma, se sortea el problema de las farolas que provocan el apagado de la lámpara de infrarrojos a su paso, no interrumpiéndose así la ejecución del programa

- Uso de una lámpara de infrarrojos con longitud de onda de 940 nm, de forma que pase desapercibida al no emitir ningún brillo

- Incorporación de una rutina al inicio del programa que permita detectar la persona que va a conducir para así aplicar unos parámetros u otros. Esto es factible debido a que las propias herramientas usadas son capaces de reconocer personas, por lo que si al inicio del programa se escanea a la persona es posible aplicar unos límites de EAR o MAR personalizados para cada sujeto. En este caso, sería necesario realizar al inicio de uso del sistema un reconocimiento del conductor para almacenar sus datos en condiciones de no somnolencia

- Entrenar un clasificador específico para la posición del conductor que permita detectar el rostro aunque no esté orientado frontalmente. Para ello, se debería utilizar un conjunto de imágenes de personas conduciendo en las que, además de aparecer rostros de frente, aparezcan rostros girados hacia ambos lados. Si bien no es necesario que estén totalmente

girados (sería inútil pues podría funcionar la detección pero la extracción de coordenadas no sería buena), el disponer de imágenes de entrenamiento con rostros ligeramente de perfil abriría un abanico de posibilidades para la instalación del sistema dentro del coche, pudiendo colocarlo en ángulo respecto al conductor. Además, haría el programa más robusto

6. BIBLIOGRAFÍA

[1] Michael Page, «www.michaelpage.es,» [En línea]. Available: https://www.michaelpage.es/prensa-estudios/estudios/transport-commute/resultados-de-espa%C3%B1a#:~:text=Espa%C3%B1a%20tiene%20una%20tasa%20de,de%20la%20media%20en%20Europa.

[2] DGT, «Día Mundial de la Seguridad y Salud en el trabajo,» 04 Abril 2019. [En línea]. Available: https://revista.dgt.es/es/noticias/nacional/2019/04ABRIL/0426-Dia-Mundial-Seguridad-y-Salud-en-el-trabajo.shtml.

[3] DGT, «Campaña de accidentes in itinere,» 09 Septiembre 2014. [En línea]. Available: https://revista.dgt.es/es/noticias/nacional/2014/09SEPTIEMBRE/0915campana-accidentes-in-itinere.shtml.

[4] DGT. Subdirección general de intervención y políticas viales. Unidad de intervención educativa, «Otros factores de riesgo: La fatiga,» [En línea]. Available: http://publicacionesoficiales.boe.es.

[5] DGT, «Como funciona ADA detector de fatiga,» 03 Marzo 2021. [En línea]. Available: https://revista.dgt.es/es/motor/tecnologia-seguridad/2021/0317-Como-funciona-ADA-detector-fatiga.shtml.

[6] K.Fujiwara, «Heart Rate Variability-Based Driver Drowsiness Detection and Its Validation With EEG,» *IEEE Transactions on Biomedical Engineering,* vol. 66, nº 6, Junio 2019.

[7] Bosch, «www.bosch-mobility-solutions.com,» [En línea]. Available: Bosch https://www.bosch-mobility-solutions.com/en/solutions/assistance-systems/driver-drowsiness-detection/ .

[8] Audi, «www.audi-mediacenter.com,» [En línea]. Available: https://www.audi-mediacenter.com/en/the-sharper-drive-the-new-audi-a5-2956/driver-assistance-systems-3036.

[9] Ford, «media.ford.com,» 13 08 2011. [En línea]. Available: https://web.archive.org/web/20110513232258/http://media.ford.com/article_print.cfm?article_id=34562.

[10] PSA, «uncece.org,» Marzo 2005. [En línea]. Available: https://unece.org/fileadmin/DAM/trans/doc/2005/wp29/ITS-09-05e.pdf..

[11] N. M. K. K. N. A. Relangi S.P.K., «Full Length Driver Drowsiness Detection Model—Utilising Driver Specific Judging Parameters,» *Smart Innovation, Systems and Technologies,* vol. 169, 2020.

[12] A. W. R. C. R. Galindo R., «Landmark Based Eye Ratio Estimation for Driver Fatigue Detection,» *Intelligent Robotics and Applications,* vol. Lecture Notes in Computer Science, nº 11744, 2019.

[13] J. M. Viola P., «Rapid Obejct Detection using a Boosted Cascade of Simple Features,» de *Accepted Conference on Computer Vision and Pattern Recognition*, Cambridge, 2001.

[14] AaronWard, «medium.com,» [En línea]. Available: https://medium.com/@aaronward6210/facial-detection-understanding-viola-jones-algorithm-116d1a9db218. [Último acceso: 23 07 2021].

[15] Wikipedia, «wikipedia.org,» [En línea]. Available: en.wikipedia.org/wiki/Prewitt_operator. [Último acceso: 18 Agosto 2021].

[16] Wikipedia, «wikipedia.org,» [En línea]. Available: en.wikipedia.org/wiki/Sobel_operator. [Último acceso: 18 Agosto 2021].

[17] N. T. B. Dalal, «Histogram of Oriented Gradients for Human Detection,» *lear.inrialpes.fr,* p. 5, 2005.

[18] Fundación Mapfre, «sistemas-adas.org,» [En línea]. Available: https://www.sistemas-adas.org/tipos/sistemas-de-deteccion-de-fatiga. [Último acceso: 20 Agosto 2021].

[19] A. A. R. A. Krishnan A., «Understanding Fairness of Gender Classification Algorithms Across Gender-Race Groups,» *Researchgate,* p. 8, 2020.

[20] A. Coste, «Project 1: Histograms,» sci.utah.edu, University of Utah.

DISEÑO DE UN SISTEMA DE BAJO COSTE PARA LA DETECCIÓN DE LA SOMNOLENCIA EN LA CONDUCCIÓN BASADO EN RECONOCIMIENTO DE EXPRESIONES FACIALES

DOCUMENTO II: PRESUPUESTO

1. PRESUPUESTO GENERAL

A continuación, se va a realizar un desglose de los distintos costes presentes en la ejecución de este proyecto. Esto permitirá valorar económicamente el desarrollo del proyecto.

Los costes a considerar son de recursos humanos por el tiempo dedicado a desarrollar el software y analizar los resultados, de coste computacional por el tiempo dedicado a procesar datos y coste material debido a los equipos y programas necesarios para la consecución de los objetivos. Se van a computar los costes relativos al desarrollo del sistema de detección de somnolencia en la conducción, no incluyendo los costes en los que se ha incurrido al desarrollar el presente documento (horas de trabajo, uso de programas como MATLAB, etc.).

1.1. CUADRO DE PRECIOS

En este capítulo se muestran los precios correspondientes a los distintos actores que han intervenido en la realización del presente TFM. Estos son por un lado los asociados a recursos humanos y, por otro lado, los relativos a recursos materiales.

CUADRO DE PRECIOS BÁSICOS: RECURSOS HUMANOS			
Código	Unidad	Descripción	Precio(€)
RH.TUT1	h	Doctor Ingeniero Industrial (Tutor)	50
RH.TUT2	h	Doctor Ingeniero Industrial (Cotutor)	50
RH.II	h	Ingeniero Industrial	20

Tabla 4. Cuadro de precios básicos: Recursos humanos

- Los doctores ingenieros industriales, ambos tutores de este proyecto, han desempeñado labores de apoyo, supervisión y planificación del trabajo, tareas necesarias para alcanzar los objetivos

- El ingeniero industrial ha realizado las tareas de revisión bibliográfica, selección de los equipos, programación, montaje del sistema y análisis de los resultados, así como la redacción de todo este desarrollo en el presente documento

En cuanto a los recursos materiales, se tienen por un lado los equipos que van específicamente destinados a este proyecto y por otro lado equipos o programas que han participado en el proyecto, pero están destinados a varios usos.

Los programas destinados específicamente a este trabajo son Python, VS Code y Thonny, y no se va a computar ningún coste pues son gratuitos. Para los equipos o programas que no están destinados específicamente a este proyecto, estimar el coste que suponen para el proyecto no es inmediato: Al

adquirir ciertos programas se obtiene licencia por un año y sin embargo el trabajo no se ha prolongado durante tanto tiempo y estos programas han servido para otras labores ajenas a este trabajo. Por lo tanto, se debe calcular el coste de amortización. Lo mismo supone con el PC y el multímetro empleados para el desarrollo del proyecto, cuya vida útil y usos exceden a este trabajo.

- Ordenador portátil HP Pavilion: 600 €
- Licencia Microsoft Office 365: 7 €/mes
- Multímetro Digital: 19 €

El coste de la licencia de Microsoft Office ha sido obtenido directamente de la página web de dicho producto. Considerando que el año 2021 tiene 251 días laborables con jornadas de 8 h, los costes de amortización resultan:

- Licencia Microsoft Office 365:

$$7 {}^{€}/_{mes} \times 1\ mes/_{20\ días\ laborables} \times 1\ día/_{8h\ laborables} = 0{,}044\ {}^{€}/_{h}$$

Para el PC se considera un periodo de amortización de 10 años. En cuanto a su sistema operativo, Windows 10, su coste ya se ha contabilizado en el precio del equipo. Repitiendo los cálculos anteriores:

- HP Pavilion:

$$600 {}^{€}/_{10\ años} \times 1\ año/_{251\ días\ laborables} \times 1\ día/_{8h\ laborables} = 0{,}030\ {}^{€}/_{h}$$

Finalmente, dado que se espera que el multímetro tenga una vida útil muy larga y que su uso en el proyecto se ha limitado a unos minutos para hacer pruebas con la lámpara de infrarrojos, para buscar cables de alimentación de 12 V en el coche y para comprobar la tensión de la Raspberry Pi durante su montaje en el vehículo, no se va a imputar ningún coste.

Otro coste a tener en cuenta es el energético, es decir, la potencia absorbida por el ordenador y la Raspberry Pi durante el desarrollo del trabajo y el combustible invertido en la realización de ensayos. En el capítulo 3.3 se ha comentado que el consumo de la Raspberry Pi es de unos 600 mA a 5 V de media, o sea 3 W. En cuanto al PC, viendo las especificaciones del cargador se ve que la potencia máxima que puede consumir es de 50 W. Sin embargo, las tareas asociadas al PC no han requerido que trabaje a máximo rendimiento, por lo que se va a tomar como valor de consumo 20 W que es el valor de potencia mínima en uso, excluyendo los modos apagado y *stand by*.

Tomando un precio de la electricidad de 0,15 €/kWh, se obtiene que el precio por hora de uso del PC es de 0,003 €/h y para la Raspberry 0,00045 €/h.

El coste relativo a combustible se calcula partiendo del precio del combustible (se tomará como 1,15 €/l), el consumo medio del vehículo (4,2 l/100 km) y el kilometraje recorrido en cada ensayo. Así, se

tienen 4,83 €/100 km. Con todos estos costes y el coste de los equipos que se han adquirido para el proyecto, el cuadro de precios relativo a los materiales queda:

CUADRO DE PRECIOS BÁSICOS: RECURSOS MATERIALES			
Código	Unidad	Descripción	Precio (€)
RM.MO	h	Microsoft Office 365	0,044
RM.PC	h	PC HP Pavilion	0,03
RM.PCEN	h	Coste energético del PC	0,003
RM.RPEN	h	Coste energético de la Raspberry Pi 4 Model B 8 GB	0,00045
RM.GAS	100 km	Coste de combustible	4,83
RM.RP	ud	Coste Raspberry Pi 4 Model B 8GB	85
RM.CAM	ud	Coste Raspberry Pi Camera NoIR v2	27
RM.IM	ud	Coste cable de imagen micro-HDMI a HDMI	9
RM.PS	ud	Coste fuente de alimentación Raspberry 15,3 W	9
RM.TM	ud	Coste tarjeta micro SDHC 32 GB 100 MB/s	8
RM.PS2	ud	Coste cable de alimentación USB a USB tipo C	8
RM.PS3	ud	Coste cargador para coche 12 V	10
RM.IR	ud	Coste lámpara de infrarrojos	7

Tabla 5. Cuadro de precios básicos: Material

Nótese que ninguno de estos precios incluye el IVA, pues este se aplicará en la obtención del presupuesto final.

1.2. UNIDADES DE OBRA

En este apartado se va a determinar el precio unitario de cada unidad de obra realizando a cabo un cuadro de precios descompuesto para cada una. Así, se obtendrán unos presupuestos parciales que, tras actualizarlos con los valores de mediciones del capítulo siguiente, indiquen el presupuesto total de este TFM.

Se ha considerado en concepto de costes directos complementarios un 3% del valor de cada unidad de obra, de modo que se tengan en cuenta gastos difícilmente cuantificables por no ser este trabajo el único beneficiario de ciertas facilidades como la conexión a internet.

Se van a considerar 3 unidades de obra distintas:

- La unidad de obra UO-01 contiene los costes de las tareas de programación requeridas para desarrollar el programa de detección de somnolencia:

Código	Unidad	Descripción	Rendimiento	Precio	Importe (€)
UO-01	h	**Tareas Programación**			
	Coste asociado al desarrollo de las tareas de programación de un código de acuerdo a los requerimientos del proyecto				
RH.II	h	Ingeniero Industrial	1,000	20,000	20,000
RM.MO	h	Microsoft Office 365	0,600	0,044	0,026
RM.PC	h	PC HP Pavilion	1	0,030	0,030
RM.PCEN	h	Coste energético del PC	1,000	0,003	0,003
RM.RPEN	h	Coste energético de la Raspberry Pi	0,800	0,00045	0,00036
%	%	Costes Directos Complementarios	0,030	20,060	0,602
		Costes Directos			20,662
	%	Costes Indirectos	0,010	20,662	0,207
		Coste Total			20,868

Tabla 6. Unidad de obra 01: Tareas de programación

- La unidad de obra UO-02 engloba los costes relativos a los ensayos realizados:

Código	Unidad	Descripción	Rendimiento	Precio	Importe (€)
UO-02	h	**Realización de Ensayos**			
	Coste derivado de ensayar el sistema diseñado, tanto simulaciones estáticas como circulando. Realización del ensayo, procesado de datos y análisis. Colaboración de los tutores en las funcionalidades del programa y en las interfaces vehículo-sistema				
RH.II	h	Ingeniero Industrial	1,000	20,000	20,000
RH.TUT1	h	Doctor Ingeniero Industrial (Tutor)	0,1	50,000	5,000
RH.TUT2	h	Doctor Ingeniero Industrial (Cotutor)	0,1	50,000	5,000
RM.MO	h	Microsoft Office 365	0,500	0,060	0,030
RM.PCEN	h	Coste energético del PC	0,300	0,003	0,001
RM.RPEN	h	Coste energético de la Raspberry Pi	0,800	0,00045	0,00036

Código	Unidad	Descripción	Rendimiento	Precio	Importe (€)
UO-02	h	**Realización de Ensayos**			
	Coste derivado de ensayar el sistema diseñado, tanto simulaciones estáticas como circulando. Realización del ensayo, procesado de datos y análisis. Colaboración de los tutores en las funcionalidades del programa y en las interfaces vehículo-sistema				
RM.GAS	100 km	Coste de combustible	0,600	4,83	2,898
%	%	Costes Directos Complementarios	0,030	32,929	0,988
		Costes Directos			33,917
	%	Costes Indirectos	0,010	33,917	0,339
		Coste Total			34,256

Tabla 7. Unidad de obra 02: Realización de ensayos

- La unidad de obra UO-03 incluye los costes de los equipos adquiridos para la construcción del sistema

Código	Unidad	Descripción	Rendimiento	Precio	Importe (€)
UO-03	h	**Equipos**			
	Coste de los equipos empleados para el desarrollo del sistema de detección de somnolencia en la conducción				
RM.RP	ud	Coste Raspberry Pi 4 Model B 8GB	1,000	85,000	85,000
RM.CAM	ud	Coste Raspberry Pi Camera NoIR v2	1,000	27,000	27,000
RM.IM	ud	Coste cable de imagen micro-HDMI a HDMI	1,000	9,000	9,000
RM.PS	ud	Coste fuente de alimentación Raspberry 15,3 W	1,000	9,000	9,000
RM.TM	ud	Coste tarjeta micro SDHC 32 GB 100 MB/s	1,000	8,000	8,000
RM.PS2	ud	Coste cable de alimentación USB a USB tipo C	1,000	8,000	8,000
RM.PS3	ud	Coste cargador para coche 12 V	1,000	10,000	10,000
RM.IR	ud	Coste lámpara de infrarrojos	1,000	7,000	7,000
%	%	Costes Directos Complementarios	0,030	163,000	4,890
		Costes Directos			167,890
	%	Costes Indirectos	0,010	167,890	1,679

Código	Unidad	Descripción	Rendimiento	Precio	Importe (€)
UO-03	h	**Equipos**			
	Coste de los equipos empleados para el desarrollo del sistema de detección de somnolencia en la conducción				
		Coste Total			169,569

Tabla 8. Unidad de obra 03: Equipos

1.3. MEDICIONES

La dedicación al desarrollo del presente TFM se estima en 300h, tiempo correspondiente a 12 ECTS. Este tiempo se ha distribuido en las distintas tareas (programación y ensayado). Los trabajos de programación son los que más tiempo han abarcado, suponiendo un 70% del total (210h). A los ensayos realizados se atribuye el 30% del tiempo (90h). En cada una de estas dos mediciones se incluye tanto el desarrollo de la tarea en sí como la redacción correspondiente para conformar la memoria del trabajo. Finalmente, a la unidad de obra UO-03 se le asigna una unidad.

1.4. PRESUPUESTO FINAL

Código	Unidad	Descripción	Rendimiento	Precio	Importe (€)
UO-01	h	Tareas de Programación	210	20.868	4382.32
UO-02	h	Realización de Ensayos	90	34.256	3083.07
UO-03	ud	Equipos	1	169.569	169.57

Presupuesto de Ejecución Material.....	7634.95
Gastos Generales 13%.....	992.54
Beneficio Industrial 6%.....	458.10
Presupuesto de Ejecución por Contrata.....	9085.59
IVA 21%.....	1907.97
Presupuesto base de licitación.....	10993.57

Tabla 9. Presupuesto final

Después de contabilizar los gastos generales y el beneficio industrial y aplicar el IVA que en 2021 se sitúa en un 21%, el presupuesto final de este proyecto asciende a la cifra de DIEZ MIL NOVECIENTOS NOVENTA Y TRES EUROS CON CINCUENTA Y SIETE CÉNTIMOS.