UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Aeroespacial y Diseño Industrial

DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA DETECTAR Y ALERTAR DE LA SOMNOLENCIA DURANTE LA CONDUCCIÓN MEDIANTE LA MONITORIZACIÓN DE LA CABEZA Y EL ROSTRO

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial y Automática

AUTOR/A: Santos Fernández, Lucas

Tutor/a: Quiles Cucarella, Eduardo

CURSO ACADÉMICO: 2022/2023

RESUMEN

La somnolencia al volante ha sido identificada como una de las principales causas de accidentes de tráfico en el mundo. La falta de atención y la disminución del tiempo de reacción debido a la somnolencia pueden ocasionar consecuencias fatales. Resulta por tanto de vital importancia desarrollar sistemas ADAS (*Advanced Driver Assistance Systems*) capaces de detectar el estado del conductor para velar por la seguridad vial.

En este trabajo se presenta el desarrollo de un sistema de detección de somnolencia en la conducción basado en la monitorización del rostro utilizando técnicas de visión artificial. Se emplea una Raspberry Pi y una cámara ubicada en el interior del vehículo para capturar imágenes faciales en tiempo real. Estas imágenes son procesadas utilizando algoritmos de visión artificial implementados en Python, que extraen características relevantes del rostro. Luego se calculan parámetros clave asociados con la somnolencia como son la apertura de los ojos y de la boca, el porcentaje de tiempo en que los ojos están cerrados (índice PERCLOS) y la posición e inclinación de la cabeza. Se establecen unos umbrales de alarma que se comparan con los parámetros mencionados para determinar si el conductor se encuentra en un nivel peligroso de somnolencia. En caso de superar dichos umbrales, el sistema activa una alarma para alertar al conductor.

Además, se ha incorporado la escala de somnolencia de Karolinska (KSS), una herramienta reconocida en el campo de la somnolencia en la que el propio sujeto se autoevalúa, lo que ha permitido realizar una evaluación más completa y precisa del estado de somnolencia del conductor. El sistema propuesto se ha sometido a diversos ensayos, incluyendo en condiciones reales de conducción, con resultados interesantes que contribuyen a mejorar la seguridad vial alertando a los conductores sobre su nivel de somnolencia y fomentando la adopción de medidas preventivas.

Palabras Clave:

Somnolencia, conducción, visión artificial, detección facial, monitorización del rostro, seguridad vial, PERCLOS

RESUM

La somnolència al volant ha estat identificada com una de les principals causes d'accidents de trànsit al món. La manca d'atenció i la disminució del temps de reacció a causa de la somnolència poden ocasionar conseqüències fatals. Resulta per tant de vital importància desenvolupar sistemes ADAS (Advanced Driver Assistance Systems) capaços de detectar l'estat del conductor per vetllar per la seguretat viària.

En aquest treball es presenta el desenvolupament d'un sistema de detecció de somnolència en la conducció basat en la monitorització del rostre utilitzant tècniques de visió artificial. Es fa servir una Raspberry Pi i una càmera ubicada a l'interior del vehicle per capturar imatges facials en temps real. Aquestes imatges són processades utilitzant algoritmes de visió artificial implementats en Python, que extreuen característiques rellevants del rostre. A continuació, es calculen paràmetres clau associats amb la somnolència com són l'obertura dels ulls i de la boca, el percentatge de temps en què els ulls es mantenen tancats (índex PERCLOS) i la posició i inclinació del cap. S'hi estableixen uns llindars d'alarma que es comparen amb els paràmetres esmentats per determinar si el conductor es troba en un nivell perillós de somnolència. En cas de superar aquests llindars, el sistema activa una alarma per alertar el conductor.

A més, s'ha incorporat l'escala de somnolència de Karolinska (KSS), una eina reconeguda en el camp de la somnolència en què el propi individu s'autoavalua, el que permet realitzar una avaluació més completa i precisa de l'estat de somnolència del conductor. El sistema proposat s'ha sotmés a diversos assaigs, incloent-hi condicions reals de conducció, amb resultats interessants que contribueixen a millorar la seguretat viària alertant els conductors sobre el seu nivell de somnolència i fomentant l'adopció de mesures preventives.

Paraules Clau:

Somnolència, conducció, visió artificial, detecció facial, monitorització del rostre, seguretat viària, PERCLOS

ABSTRACT

Somnolence behind the wheel has been identified as one of the main causes of traffic accidents throughout the world. Lack of attention and decreased reaction time due to sleepiness can result in fatal consequences. It is therefore of vital importance to develop ADAS systems (Advanced Driver Assistance Systems) capable of detecting the driver's condition to ensure road safety.

This paper presents the development of a somnolence detection system based on face monitoring using artificial vision techniques. A Raspberry Pi and a camera located inside the vehicle are used to capture facial images in real time. These images are processed using artificial vision algorithms implemented in Python, which extract relevant features of the face. Key parameters associated with sleepiness are then calculated, such as eye and mouth opening, the percentage of time the eyes remain closed (PERCLOS index), and head position and tilt. Alarm thresholds compared to the mentioned parameters are established in order to determine if the driver is at a dangerous level of drowsiness. If these thresholds are exceeded, the system activates an alarm to alert the driver.

In addition, the Karlolinska Sleepiness Scale (KSS) has also been incorporated, a recognized tool in the field of sleepiness in which the subjects self-assess themselves, which has allowed a more complete and accurate assessment of the driver's sleepiness state. The proposed system has been subjected to various tests, including in real driving conditions, with interesting results that contribute to improving road safety by alerting drivers about their level of drowsiness and encouraging the adoption of preventive measures.

Key words:

Somnolence, driving, artificial vision, facial detection, face monitoring, road safety, PERCLOS

ÍNDICE DE DOCUMENTOS

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA DETECTAR Y ALERTAR DE LA SOMNOLENCIA DURANTE LA CONDUCCIÓN MEDIANTE LA MONITORIZACIÓN DE LA CABEZA Y EL ROSTRO.

DOCUMENTO I: MEMORIA

ÍNDICE DE LA MEMORIA

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. MOTIVACIÓN

Se considera la somnolencia como un estado intermedio entre la vigilia y el sueño, en el que se siente más cansancio del habitual a lo largo del día. Esta situación implica que las personas que se encuentran en dicho estado puedan quedarse dormidas en momentos indeseados, ocasionando en algunos de ellos graves problemas de seguridad. Por ejemplo, durante la conducción.

La fatiga, por otra parte, se define como la sensación de falta de energía, de agotamiento o de cansancio. Es común que la fatiga derive en somnolencia, momento en el que se reduce considerablemente la capacidad de prestar atención al entorno y que se expresa mediante bostezos frecuentes, pesadez en los párpados, lentitud de movimientos, visión borrosa, etc. Como los conceptos de somnolencia y fatiga están íntimamente relacionados, se utilizarán indistintamente a lo largo de este trabajo.

Existe una amplia cultura de la conducción en las sociedades actuales debido a la constante necesidad de desplazamiento. A pesar de que en las grandes ciudades cada vez se requiere menos el uso de vehículos particulares, promocionándose el transporte público y medios alternativos de transporte como las bicicletas o los patinetes eléctricos, para la mayoría de las personas resulta necesario utilizar el coche. Sobre todo, para aquellos que residen en la periferia de las ciudades o en pequeñas poblaciones. De hecho, en España el promedio de tiempo que se utiliza para llegar al lugar de trabajo es de 36 minutos [1], cuyos trayectos suelen ser monótonos y generalmente en horas punta, características que propician la aparición de somnolencia. Según las estadísticas de la DGT, el censo de conductores con permiso B en el año 2021 en España fue de más de 24 millones. [2]

A pesar de las campañas de concienciación y las medidas preventivas, los accidentes de tráfico relacionados con la fatiga siguen siendo una preocupación mundial. Según la DGT, la somnolencia interviene en entre el 15 y el 30% de los accidentes de tráfico que se producen en España. Este tipo de accidentes son hasta el doble de mortales que otros tipos de siniestros en la carretera. Existen trabajos con mucha importancia en este ámbito como [3] en el que se demuestra que la somnolencia durante la conducción incrementa el riesgo de accidente entre cuatro a seis veces más que si el conductor se encontrase en un estado de vigilia.

Para prevenir estos accidentes, además de las campañas publicitarias de la DGT, cada vez es más común el uso de la tecnología para alertar a los conductores cuando se encuentran en un estado de somnolencia. Cada vez más y sobre todo alentados por la normativa europea, se están incorporando sistemas más complejos y fiables de detección de somnolencia en los nuevos vehículos que se están produciendo y comercializando.

1.2. OBJETIVO

El punto de partida de este trabajo es el TFM de Julio Cano Bernet [4], donde se diseña un sistema de bajo coste para detectar la somnolencia en la conducción en base a expresiones faciales. En dicho TFM se contrastan dos métodos diferentes para la tarea de detección de rostros y se utilizan dos indicadores para evaluar la somnolencia del conductor, que son la

apertura de los ojos y de la boca. Se diseña también el hardware que se utilizará para los ensayos en este trabajo.

El objetivo de este trabajo final de grado es introducir mejoras en el desarrollo del TFM mencionado y llevar a cabo un apartado de ensayos más profundo. Se va a replantear el algoritmo, introduciendo nuevos indicadores de somnolencia y utilizando una combinación de los métodos de detección de rostros propuestos. El sistema ofrecerá como salida una clasificación no excluyente del estado del conductor en 5 tipos diferentes: dormido, distraído, con la mirada distraída, bostezando y somnoliento. Esto permitirá advertir al conductor de su estado con relación a la somnolencia.

Tal y como se realiza en el TFM de Cano, se hará uso de una cámara instalada en el interior del vehículo, frente al conductor, cuyas imágenes tomadas serán procesadas para realizar la detección y caracterización del rostro.

2. ESTADO DEL ARTE

A raíz del análisis de la peligrosidad de encontrarse ante un volante durante un estado de somnolencia, surge la necesidad de implementar sistemas que detecten y actúen en pro de la seguridad vial.

A los sistemas avanzados que ayudan a que la conducción sea más segura se les conoce como sistemas ADAS (Advanced Driver-Assistant Systems), y algunos de los más conocidos son el detector de ángulo muerto, el control de crucero adaptativo, el aviso de colisión o el detector de peatones. En la actualidad, existen algunos sistemas incorporados en vehículos que tratan de detectar la fatiga y la somnolencia al volante. Estos generalmente se encuentran en vehículos de media y alta gama, y se basan en sensores que detectan el giro del volante, la desviación de los carriles, o la cantidad de tiempo de conducción ininterrumpida. En estos casos, es frecuente que se muestre en el cuadro de mandos del vehículo un símbolo de una taza de café acompañado de una señal acústica para aconsejar al conductor que debe detener el vehículo y descansar.

Según un informe realizado por la fundación MAPFRE [5] sobre sistemas avanzados de conducción, la oferta de estos tipos de sistemas en el mercado español se ha disparado.

Por otra parte, gran cantidad de artículos académicos realizan propuestas de sistemas de detección de somnolencia basándose en un análisis de las características faciales, haciendo especial inciso en la apertura de los ojos y la boca. Sin embargo, solamente en algunos vehículos de alta gama se pueden encontrar sistemas que hagan uso de una cámara enfocada al conductor cuya función sea analizar las características faciales.

Así mismo, existen métodos basados en medidas fisiológicas como el ritmo cardíaco y las ondas cerebrales, pero son métodos intrusivos pues requieren colocar electrodos en el sujeto a estudiar.

2.1. FACTORES INFLUYENTES EN LA FATIGA

Aunque la fatiga es un estado complejo cuyo origen se da en diferentes procesos fisiológicos y psicológicos, existe suficiente consenso como para identificar los principales factores de que

alguien la sufra: edad de la persona, pocas horas de sueño y sueño acumulado, experiencia en la conducción, desórdenes de sueño y la hora del día.

Según la DGT [6], la fatiga viene dada por los siguientes cuatro factores principales:

- El momento del día: la madrugada y las primeras horas de la tarde son los momentos más críticos para conducir, pues el sueño aparece con más facilidad.
- La estimulación ambiental y el nivel de actividad del conductor: los entornos monótonos y el ir sin acompañante y en silencio favorecen su aparición.
- Las diferencias individuales de los conductores: cada persona tiene su propio ciclo de actividad, el autoconocimiento ayuda a detectar en qué momentos es más probable que aparezca la somnolencia.
- Las horas de vigilia continuada: a mayor tiempo despierto, más probable es que se tenga sueño.

Pero también existen otros factores relevantes en su aparición, como son una mala ventilación del vehículo, el mal estado del vehículo, la prisa y el estrés, las comidas copiosas, las enfermedades, etc.

2.2. SÍNTOMAS DE FATIGA Y SOMNOLENCIA

El reconocimiento de los síntomas de la fatiga es la base de los sistemas dedicados a detectarla. Existen diferentes tipos de síntomas, aunque los más notorios son los síntomas fisiológicos, como por ejemplo la actividad de los ojos y la boca, las expresiones faciales, los movimientos de la cabeza, la deceleración del pulso cardíaco, etc. Esta clase de síntomas son los más relevantes para este proyecto ya que se centra en la detección de la somnolencia mediante la monitorización del rostro del conductor.

- **Síntomas fisiológicos**

  En un sentido físico, la fatiga aparece de manera evidente en tres ámbitos clave: la visión, la audición y los movimientos corporales.

  Con fatiga la visión puede tornarse borrosa, lo que genera dificultades a la hora de discernir y enfocar los objetos que se encuentran en el campo visual. Con ello, suele aumentar la cantidad y la duración de los parpadeos. El índice PERCLOS, anteriormente expuesto, se encarga de contener información sobre la cantidad de tiempo en que permanecen los ojos cerrados.

  La sensibilidad auditiva de quien padece fatiga disminuye, razón por la que el conductor puede no darse cuenta de cierta información sonora relativa a la conducción. Se pueden experimentar también reacciones fuertes ante sonidos bruscos y repentinos como puede ser una sirena o un claxon.

  Además, la temperatura corporal y el ritmo cardíaco son también parámetros que se relacionan con la somnolencia, y general se miden utilizando electrocardiogramas (ECG) y electroencefalogramas (EEG).

- **Síntomas en el comportamiento y la manera de conducir**

  Tener fatiga implica que los movimientos de quien la padece se vuelven más lentos y menos precisos, por lo que el tiempo de reacción al volante incrementa. Ligado a esto, se tienen distracciones con más facilidad y se altera la percepción del espacio y del tiempo. Por lo tanto, en un estado de fatiga, se tiene menos capacidad para tomar decisiones que requiere la conducción.

  Hay patrones de comportamiento que se relacionan intensamente con la fatiga, como son: inclinar la cabeza hacia un lado, episodios de cabeceo más frecuentes y el incremento de la actividad de los ojos. Además, existen otras variables utilizadas para relacionar con la somnolencia como son los cambios de velocidad del vehículo, los cambios involuntarios entre carriles o la distancia al vehículo que se encuentra delante.

2.3. TIPOS DE MEDIDAS DE LA SOMNOLENCIA

Existen en la actualidad diferentes estrategias para abordar el tema de la detección de la somnolencia en el ámbito de la conducción, pues es un aspecto complejo del ser humano. Se presentan a continuación.

2.3.1. Medidas subjetivas

Se han inventado y desarrollado diferentes escalas para medir la somnolencia de manera subjetiva, es decir, implican que el propio sujeto sea quien evalúe su nivel de somnolencia. Estas escalas se completan mediante cuestionarios o entrevistas en los que el conductor describe su estado en relación con la somnolencia. [7]

La más conocida y considerada como referente es la escala de somnolencia de Karolinska (Karolinska Sleepiness Scale, KSS), desarrollada en el Instituto de Medicina del Sueño de Karolinska en Estocolmo, Suecia. La KSS es una escala de 9 puntos que comprende desde “extremadamente alerta” hasta “muy somnoliento, gran esfuerzo para mantenerse despierto, luchando contra el sueño”. Se considera la calificación de 7 hacia arriba como el punto de partida de la somnolencia.

Calificación	Descripción verbal
1	Extremadamente alerta
2	Muy alerta
3	Alerta
4	Más bien alerta
5	Ni alerta ni somnoliento
6	Algunos signos de somnolencia
7	Somnoliento, sin esfuerzo por mantenerse despierto
8	Somnoliento, algún esfuerzo por mantenerse despierto
9	Muy somnoliento, gran esfuerzo por mantenerse despierto, luchando contra el sueño

Figura 1. Escala de somnolencia de Karolinska.

Otra escala conocida y empleada es la escala de somnolencia de Epworth (Epworth Sleepiness Scale, ESS), que mide la propensión que tiene un sujeto a quedarse dormido en situación de la cotidianeidad. Consta de 8 cuestiones en las que el sujeto debe evaluar su probabilidad de quedarse dormido en una escala del 0 al 3.

Estos métodos subjetivos son ampliamente utilizados y presentan ventajas como que son simples y accesibles para cualquier sujeto, además de tener bajo costo, pero también presentan importantes desventajas:

- Sesgo de la percepción individual: dependen de la propia percepción del individuo que se autoevalúa, lo que puede conllevar a una conclusión inexacta sobre el estado de somnolencia en que se encuentra realmente.
- Influencia de factores externos: el ambiente o el estado emocional del sujeto pueden influir en el estado de somnolencia real de la persona.
- Falta de especificidad: estos métodos pueden no ser demasiado específicos. Pueden confundirse estados como la fatiga y el aburrimiento.
- Sensibilidad limitada: en función de la escala, pueden encontrarse grandes saltos en cuanto a contenido en las opciones a elegir.

En conclusión, este tipo de métodos son útiles en muchos contextos, pero es preciso tener en cuenta que tienen limitaciones y que lo conveniente es complementarlos con métodos objetivos para obtener una evaluación más completa y precisa.

2.3.2. Medidas en relación con la conducción

Se encuentra referido al comportamiento del conductor durante el trayecto, estando relacionado un estado de somnolencia con determinados patrones en elementos como la trayectoria o la velocidad del vehículo. También se pueden emplear señales basadas en la fuerza que se ejerce sobre los pedales o el volante, además de monitorizar el ángulo de giro del volante y la distancia al vehículo que se encuentra delante. Cuando el conductor se encuentra fatigado, tiende a tener comportamientos irregulares en los elementos mencionados, y el grado de desviación del vehículo aumenta. [8]

La ventaja principal de estas técnicas es que son alcanzables desde un punto de vista de adquisición de las señales. Además, son bastante representativas en cuanto a evaluar la somnolencia. Sin embargo, presenta limitaciones como la experiencia que tenga el conductor, las condiciones de la carretera o el tipo de vehículo.

2.3.3. Medidas en relación con el conductor

Es el tipo en que se basa este trabajado final de grado, fundamentándose en técnicas de procesamiento de imágenes para obtener información sobre los cambios faciales del conductor. Su mayor ventaja es que son no invasivos y resultan potencialmente útiles porque se basan en los inevitables síntomas físicos cuando existe fatiga y somnolencia.

Uno de los métodos más estudiados y aplicados para extraer conclusiones acerca de la somnolencia del conductor según [9] es el índice PERCLOS.

2.3.4. Medidas fisiológicas

Esta clase de sistemas tienen su fundamento en las señales de tipo muscular, cerebral y cardiovascular. La mayoría suponen una implementación invasiva pues requieren electrodos para monitorizar las señales cerebrales o cardiovasculares.

Se ha demostrado que realizando un análisis de las señales del electroencefalograma (EEG), la transición de despierto a dormido puede detectarse en las bajas frecuencias porque se produce un cambio de actividad. En la misma línea existen relaciones entre las señales de electrocardiograma (ECG) y el ritmo cardíaco con la somnolencia.

2.3.5. Métodos mixtos

Con la combinación de diferentes medidas subjetivas y objetivas, pueden obtenerse resultados de mayor calidad a la hora de detectar somnolencia.

En la tesis [10] se propone un sistema basado en el procesamiento de imágenes enfocado en detectar y caracterizar el grado de apertura de los ojos, además de obtener otras señales sobre la conducción como es el ángulo de guiñada, que se corresponde con el giro del vehículo sobre su eje vertical. Así, se estima un nivel de somnolencia del conductor. Concretamente, en esta tesis se utiliza el índice PERCLOS. Como señal de referencia para evaluar la somnolencia utiliza la medida subjetiva de la escala de somnolencia de Karolinska.

Tabla 1. Tabla comparativa de las medidas de somnolencia.

Tipo	Descripción	Ejemplos	Ventajas	Desventajas
Subjetivas	El propio sujeto evalúa su nivel de somnolencia mediante un cuestionario	Escala de Karolinska (KSS), escala de Epworth (ESS)	Sencillo y de bajo costo	Sesgo de percepción individual
Conducción	Análisis de los patrones de comportamiento del conductor	Ángulo de giro del volante, cambio de carril frecuente, velocidad del vehículo	Son señales alcanzables en muchos vehículos	Las condiciones de la carretera, el tipo de vehículo
Conductor	Análisis de los cambios faciales del conductor	Grado de apertura de los ojos y de la boca, posición y movimiento de la cabeza	Son no invasivos y se basan en inevitables síntomas físicos que aparecen con la somnolencia	Condiciones como la iluminación o la posición de la cámara pueden alterar resultados

Fisiológicas	Señales de tipo muscular, cerebral y cardiovascular	Señales EEG, señales ECG, actividad cardíaca	Síntomas que representan la somnolencia de manera no visible pero presente	Son invasivos
Mixtas	Combinación de varias medidas	Combinar la escala KSS, el análisis del ángulo de giro del volante y el grado de apertura de los ojos	Pueden tener resultados de mayor calidad al no depender de solamente una medida	Son más complejos de desarrollar

2.4. SISTEMAS COMERCIALES

Las empresas con relación al sector del automóvil llevan años diseñando e incorporando sistemas ADAS en los vehículos. Los sistemas más destacables que tienen el objetivo de detectar el estado de somnolencia en el conductor se encuentran distribuidos en los siguientes tres grupos de diferentes estrategias, que se han descrito anteriormente:

- Mediante análisis de la conducción
- Mediante análisis facial del conductor
- Mediante obtención y análisis de señales fisiológicas

2.4.1. Análisis de la conducción

La marca Volkswagen hace tiempo que tomó la decisión de incorporar el detector de fatiga de Bosch en algunos de sus modelos. Este sistema analiza la conducción y detecta comportamientos anormales, por lo que si concluye que el conductor se encuentra fatigado lo alerta a través de señales acústicas y visuales. Se basa en parámetros relacionados con el sensor de ángulo de giro del volante, el uso de los intermitentes, la hora del día y la duración del trayecto.

La marca Ford también incorpora en algunos de sus vehículos un sistema que, utilizando una cámara frontal, detecta las líneas de los carriles y monitoriza si el vehículo tiende a desviarse. De igual manera, otras marcas como Volvo, BMW y ŠKODA llevan varios años con tecnología Drive Alert para evitar distracciones a causa del cansancio.

2.4.2. Análisis facial del conductor

Ejemplo de este tipo de análisis son empresas como Datik con su producto Magic Eye, dispositivo que supervisa los parpadeos y los movimientos de ojos y cabeza para activar una alarma de aviso al conductor cuando detecta somnolencia. Dispone de una cámara y un

dispositivo para visualizar y/o emitir sonido en forma de aviso. Incluye además un servicio de registro de alertas o eventos.

Smart Eye Pro es otro ejemplo comercial dedicado a realizar un seguimiento de la cabeza y los ojos de 360 grados mediante un sistema multicámara.

2.4.3. Análisis de señales fisiológicas

Un prototipo que se encuentra en estudio y que está basado en una señal fisiológica obtenida a través de sensores en el volante para detectar el ritmo cardíaco es de la marca Toyota. E

La marca Ford se encuentra en la actualidad estudiando la actividad cerebral de sujetos en pruebas de simulación de conducción para en un futuro, aplicarlo a sus vehículos y dar un paso más en la prevención de riesgos debido a la fatiga. [11]

También Ford, concretamente en Brasil, presentó durante el año 2022 una gorra dedicada principalmente a transportistas profesionales en camiones para advertirles ante los síntomas iniciales de somnolencia al volante. Se llama SafeCap y contiene sensores capaces de captar los movimientos de la cabeza gracias a un acelerómetro y un giroscopio.

Cabe destacar que la finalidad de estos sistemas es alertar al conductor y hacerle consciente de su estado, pero la única persona que puede hacer algo al respecto es él mismo. Queda en manos de cada persona ser responsable en la conducción y no conducir sin estar seguro de encontrarse en buenas condiciones para hacerlo.

2.5. TÉCNICAS DE VISIÓN ARTIFICIAL PARA DETECCIÓN DE ROSTROS

Detectar un rostro consiste en distinguirlo de otros objetos en una imagen. A continuación, se presenta un resumen de las diferentes técnicas que existen para llevar a cabo la detección del rostro.

2.5.1. Métodos basados en características

Estos métodos emplean características del ojo para realizar una identificación de aquellas que lo definan, como son la pupila, la intensidad del iris o las esquinas de los ojos. Es común utilizar filtros para realzar determinadas características y atenuar otras.

Estos métodos tienden a ser robustos ante condiciones de cambios de iluminación, sin embargo, encuentran muchos problemas cuando los ojos se encuentran cerrados o parcialmente cerrados.

2.5.2. Métodos basados en plantillas

En estos métodos se utiliza un modelo genérico de la forma geométrica del ojo, a lo que se le considera una plantilla. Esta plantilla se basa en componentes básicas como son los círculos para modelar el iris y las elipses para modelar los límites de la esclerótica, que es el recubrimiento exterior blanco del ojo. Estos métodos comparan una imagen de entrada con la plantilla predefinida y determinan si hay coincidencia entre ellas.

Estos modelos se dividen en dos grupos: los de forma fija y los de forma variable, estos últimos conocidos como modelos deformables. Los modelos deformables son más complejos que los de forma fija porque tienen en cuenta las deformaciones posibles de la plantilla, pero justamente por eso permiten mejorar la detección cuando se producen cambios de escala o rotaciones en los objetos.

Las principales desventajas de estos métodos son que solo funcionan con imágenes frontales de cara, necesita una gran cantidad de imágenes con mucho contraste y requieren mucho tiempo de cómputo.

2.5.3. Métodos basados en apariencia

Estos métodos se basan en la apariencia fotométrica de los ojos, como es la distribución de color, en lugar de basarse en detectar la forma del ojo. Esto significa que son independientes del objeto que se pretende detectar, es decir, no se necesita el conocimiento de las características de un rostro. Estos modelos utilizan técnicas estadísticas que analizan la distribución de intensidad de la apariencia de los objetos de entrada.

Son los métodos más utilizados en la actualidad porque si estos algoritmos se entrenan con gran variabilidad de imágenes, pueden obtenerse altas tasas de detección y bajas tasas de fallos. Las grandes ventajas de estos métodos es que presentan gran robustez, eficiencia y un bajo coste computacional.

2.5.3.1. Cascadas de Haar

Las cascadas de Haar fueron propuestas por Paul Viola y Michael Jones [12] y se han convertido en un método muy popular por su eficiencia en el ámbito de la detección de rostros.

El proceso de detección sigue los siguientes pasos:

1. Creación del clasificador en cascada: se entrena un clasificador en cascada utilizando un conjunto de datos de entrenamiento con ejemplos positivos (imágenes con rostros) y ejemplos negativos (imágenes sin rostros). Está compuesto por una serie de etapas en las que cada una tiene múltiples clasificadores débiles. Estos clasificadores débiles se construyen mediante algoritmos de aprendizaje automático, como el algoritmo AdaBoost.

2. Extracción de características de Haar: estas características son patrones rectangulares que representan diferencias de intensidad en la imagen, como bordes, esquinas y cambios de textura.

3. Etapas de clasificación: durante la detección, la imagen de entrada se recorre a diferentes escalas y se aplican las características de Haar en cada región de la imagen. En cada etapa, se aplica un conjunto de clasificadores débiles secuencialmente para determinar si la región de la imagen es un posible rostro o no. Cada clasificador débil compara las características de Haar con umbrales predefinidos y devuelve un resultado binario: rostro o no rostro.

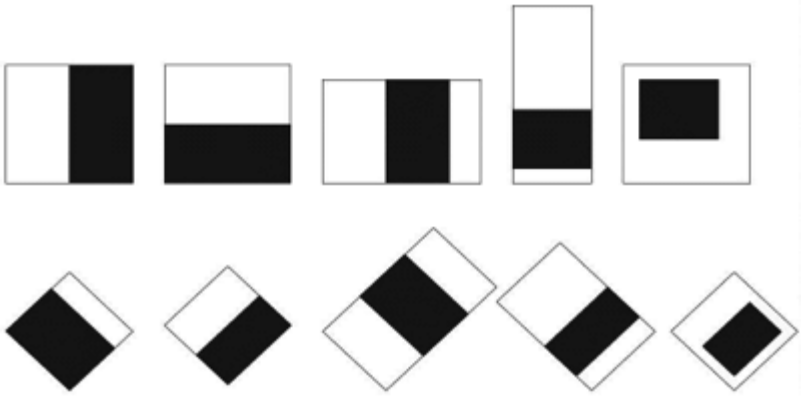


Figura 2. Ejemplo de características de Haar.

4. Clasificación en cascada: cada etapa filtra las regiones de la imagen que probablemente no contienen un rostro. Si una región supera este filtrado, pasa a la siguiente etapa para una evaluación más detallada con más cantidad de características de Haar. Si una región no supera el filtrado, se concluye que no hay rostro en ella y se pasa a siguiente región de la imagen.

5. Eliminación de falsos positivos: son aquellas regiones que se han clasificado erróneamente como rostros sin serlo. Para reducirlos, se aplican técnicas de eliminación de regiones solapadas o la verificación con características adicionales.

2.5.3.2. HOG

HOG (Histogram of Oriented Gradients) es un descriptor de características. Los descriptores de características realizan descripciones de las características visuales del contenido en imágenes, como son la forma, el color o la textura.

Este descriptor divide la imagen en celdas pequeñas y calcula el histograma de las orientaciones de los gradientes dentro de cada celda. La concatenación de estos histogramas de gradientes da lugar a un vector de características que representa el objeto.

Normalmente, a continuación del algoritmo HOG se utiliza un clasificador, como por ejemplo el SVM (Support Vector Machine), para aprender y detectar objetos basados en los vectores de características generados por HOG.

2.5.3.3. CNN

Las redes neuronales convolucionales son un tipo de redes convolucionales profundas que suelen utilizarse para el procesamiento de imágenes, debido a su gran capacidad para considerar las dependencias espaciales de los datos.

Se recopila un conjunto de datos de entrenamiento que contiene imágenes etiquetadas con la presencia o ausencia de rostros. En este proceso de aprendizaje automático, el algoritmo desarrolla la capacidad de discernir rostros. Esto lo consigue extrayendo características discriminativas de las imágenes de entrada, a medida que avanza en las capas de la red neuronal, estas características se extraen con más nivel de abstracción.

Para comprobar el resultado del aprendizaje, se utilizan más imágenes, denominado conjunto de validación, con rostros y sin ellos, para confirmar que ha adquirido esa capacidad con determinada precisión. Cuando esto se cumple, se convierte entonces en un modelo, un algoritmo entrenado que se puede utilizar para aplicarlo a otras imágenes y reconocer rostros en ellas.

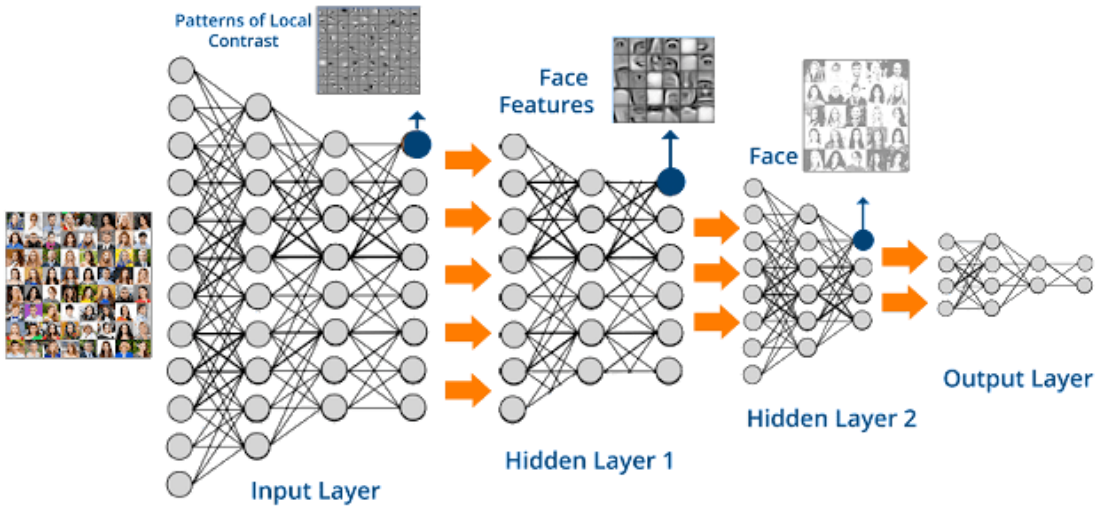


Figura 3. Ejemplo de red neuronal convolucional para imágenes.

3. NORMATIVA

Tal es el impacto de la somnolencia durante la conducción que, recientemente la Comisión Europea ha impuesto regulaciones más críticas en los requisitos técnicos de los nuevos vehículos.

Según el Reglamento Delegado (UE) 2021/1341 de la Comisión de 23 de abril de 2021 por el que se completa el Reglamento (UE) 2019/2144 del Parlamento Europeo, referido a los requisitos de homologación de los vehículos de motor y de los sistemas destinados a estos, los vehículos que se homologuen a partir del 6 de julio de 2022 deben equipar un detector de fatiga de serie.

Desde el citado reglamento, se hace referencia a este tipo de sistemas como DDAW (Driver Drowsinees and Attention Warning Systems) y son obligatorios en los vehículos de las categorías M y N, y que pueda superar por construcción los 70 km/h, debido a que la disminución de la atención del conductor se da mayormente en los trayectos largos a velocidad alta y constante.

En este se establece también como referencia la escala de somnolencia de Karolinska (ESK), fijando notificar al conductor de su estado cuando el mismo alcance un nivel de somnolencia de 8 puntos, aunque también se podrá partir de 7 puntos. La descripción de esta escala se encuentra en el apartado 2.1.1.

El método principal propuesto está basado en el análisis de la conducción, en los dos siguientes aspectos:

- Reducción del número de microcorrecciones en la dirección, acompañada de un incremento de correcciones bruscas.
- Un incremento de cambio en la posición lateral en el carril.

Sin embargo, se contemplan mediciones alternativas para utilizar como referencia en la medición de la somnolencia como la monitorización de los conductores mediante PERCLOS.

4. PROPUESTA DE DESARROLLO

El propósito de este apartado es exponer la metodología seguida para desarrollar el sistema de detección de somnolencia. Como punto de partida, se ha realizado una revisión de los factores que influyen en la fatiga y la somnolencia, y posteriormente los síntomas y procedimientos que permiten su detección a partir de los rasgos faciales.

4.1. DETERMINACIÓN DE EXPRESIONES OBJETIVO

De acuerdo con la información expuesta anteriormente sobre los signos que pueden relacionarse directamente con la somnolencia, y que se expresan mediante el rostro y se pueden detectar utilizando reconocimiento y análisis de imagen, se expone a continuación un resumen de estos:

- *Bostezos*: son indicadores de la presencia de somnolencia en conductores.
- *Apertura de los ojos*: a medida que una persona experimenta somnolencia, se le entrecierran más los ojos. Se mide la apertura de los ojos para detectar si se encuentran cerrados durante determinado tiempo continuado.
- *Porcentaje de tiempo con los ojos cerrados*: se mide la cantidad de tiempo que la apertura de los ojos se encuentra por debajo de un valor umbral, con el indicador del porcentaje PERCLOS.
- *Posición e inclinación de la cabeza*.
- *Dirección de la mirada*.

4.2. DETECCIÓN DEL ROSTRO

Se van a estudiar dos aspectos principales: el rostro y la cabeza completa. En el rostro se definen dos áreas de interés, la boca y los ojos, para caracterizar los bostezos, la apertura de los ojos y el porcentaje de tiempo con los ojos cerrados. Por otra parte, la cabeza como conjunto se utilizará para obtener la posición y la inclinación de esta. Como dichos aspectos se encuentran inevitablemente solapados, la posición de la cabeza se estimará a partir de la obtención del rostro. Por lo tanto, el primer paso consiste en detectar el rostro.

Los algoritmos que se dedican a esto suelen empezar localizando los ojos, pues es un elemento especialmente característico en el rostro. Tras esto, localizan la boca, la nariz y las cejas. Si se encuentran los elementos suficientes, se considera un rostro. Para detectar rostros y obtener las coordenadas faciales de un rostro en una imagen existen diferentes recursos de programación, basados en los métodos expuestos en el Estado del Arte, al alcance de cualquier usuario de internet.

En primer lugar, se utilizará un detector de rostro para reconocer el rostro del conductor en la imagen y, en segundo lugar, a esta región que contenga el rostro se le aplicará un predictor de puntos faciales para localizar los diferentes elementos de un rostro.

Dentro de los métodos expuestos en el apartado de Estado del Arte, el escogido para este trabajo ha sido el método basado en apariencia del descriptor HOG junto al clasificador SVM. La razón es que proporciona una buena precisión y un tiempo de cómputo bajo, lo cual permite que se pueda utilizar prácticamente en tiempo real. Las redes neuronales convolucionales (CNN) no se han considerado para este trabajo porque requieren una capacidad de cómputo que, con los recursos disponibles, no permitiría utilizarlo a tiempo real.

La herramienta escogida para aplicar HOG es Dlib, librería que contiene un conjunto de algoritmos de aprendizaje automático y de visión artificial. Su algoritmo más conocido es el que se utiliza para detectar los 68 puntos clave en un rostro, realizando así un mapeo facial.

Se va a utilizar en primer lugar el detector de rostro de Dlib llamado "get_frontal_face_detector" y luego el predictor de los puntos faciales en los rostros encontrados de Dlib llamado "shape_predictor_68_face_landmarks".

Figura 4. Ejemplo de detección de rostro con Dlib.

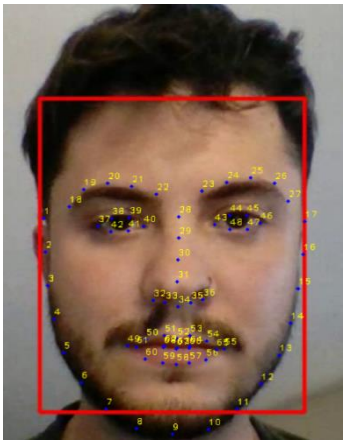


Figura 5. Ejemplo de predictor de los 68 puntos faciales clave de Dlib en el rostro detectado.

En la Figura 4 se observa cómo se ha detectado una cara en la imagen y se ha resaltado con un rectángulo de color rojo. En la Figura 5, se puede observar un ejemplo de cómo se predicen los 68 puntos clave de este mismo rostro.

4.3. ELECCIÓN DE INDICADORES

En este punto ya quedan identificadas las áreas definidas anteriormente como objetivo: los ojos y la boca, y a partir de este momento se pueden definir los indicadores que ayudarán a caracterizar los bostezos, la apertura de los ojos, el porcentaje de tiempo con los ojos cerrados y la posición e inclinación de la cabeza.

4.3.1. Indicador del grado de apertura de la boca (MAR)

El parámetro por excelencia para contemplar la apertura de la boca es el MAR (Mouth Aspect Ratio). Consiste en el cálculo de la relación entre la longitud y el ancho de la región que representa la boca en una imagen o en conjunto de puntos de referencias faciales.

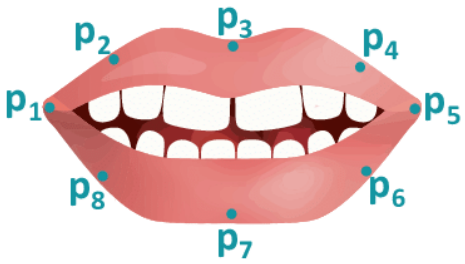


Figura 6. Boca con los puntos de referencia faciales indicados.

Su cálculo requiere medir la distancia entre los puntos de referencia que representan los extremos de los labios y la distancia entre aquellos que representan la parte superior e inferior de los labios. La fórmula utilizada es la siguiente:

$$MAR = \frac{|p_2 - p_8| + |p_3 - p_7| + |(p_4 - p_6)|}{2 \cdot |p_1 - p_5|} \quad (1)$$

Cuando la boca se encuentra cerrada, el valor del parámetro MAR tiende a ser más bajo, mientras que cuando está abierta, el valor tiende a ser más alto.

4.3.2. Indicador del grado de apertura de los ojos (EAR)

Para contemplar el grado de apertura de los ojos, se utiliza el parámetro EAR (Eye Aspect Ratio), y es similar al parámetro MAR. Se basa es el cálculo de la relación entre las distancias de los puntos de referencia asociados con los ojos.

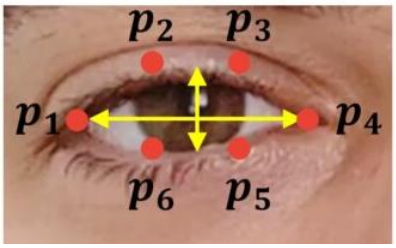


Figura 7. Ojo con los puntos de referencia representados.

La fórmula que se utiliza es la siguiente:

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

4.3.3. Indicador del porcentaje de tiempo con ojos cerrados (PERCLOS)

Para obtener el porcentaje de tiempo en que los ojos permanecen cerrados se utiliza el parámetro PERCLOS. Este término hace referencia a las siglas **PER**centaje of the Time Eyelids are **CLOS**ed. Es un índice que calcula el porcentaje de tiempo que permanecen los ojos cerrados por debajo de un determinado nivel de referencia de apertura. Es uno de los criterios más conocidos y aplicados en la literatura para detectar fatiga o somnolencia en los conductores, por ejemplo, en diferentes investigaciones llevadas a cabo por la National Highway Traffic Safety Administration (NHTSA) [13].

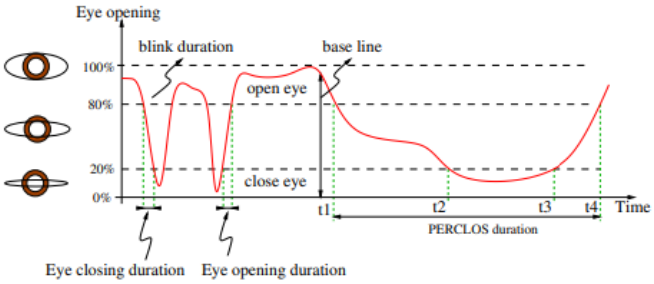


Figura 8. Evolución del cierre de ojos. Referencia: [10]

En la Figura 8 se puede observar cómo en la Tesis de García Daza el PERCLOS se computa siempre que la apertura de los ojos esté por debajo del 80 % de su valor nominal.

Se calcula dividiendo el tiempo en que el párpado está cerrado por debajo del umbral establecido entre el total del periodo de tiempo observado. Luego se multiplica por 100 para obtener el resultado en porcentaje.

$$PERCLOS = \frac{tiempo\ de\ cierre\ de\ párpados}{duración\ total\ del\ periodo\ de\ tiempo} \cdot 100 \quad (3)$$

Un valor alto de PERCLOS indica un mayor porcentaje de tiempo en que los ojos están cerrados y esto se asocia con la somnolencia o fatiga, sugiriendo que el sujeto se encuentra en un estado de alerta reducido.

4.3.4. Indicador de la posición e inclinación de la cabeza (Head Pose)

Para estimar la posición de la cabeza se utiliza un método que calcula los tres ángulos de Euler: ángulo de cabeceo, ángulo de alabeo y ángulo de guiñada, más conocidos en inglés por *pitch*, *roll* y *yaw*, respectivamente. Estos tres ángulos describen la orientación tridimensional de un objeto en relación con una referencia fija. Se exponen a continuación:

- Pitch (cabeceo): describe el movimiento de inclinación hacia arriba o hacia abajo del eje transversal, es decir, el movimiento de la cabeza cuando asentimos.
- Roll (alabeo): describe el movimiento de balanceo alrededor del eje longitudinal, es decir, el movimiento de inclinación hacia un lado de la cabeza.

- Yaw (guiñada): describe el movimiento del giro alrededor del eje vertical, es decir, el movimiento de la cabeza cuando giramos hacia la izquierda o derecha, como cuando negamos.

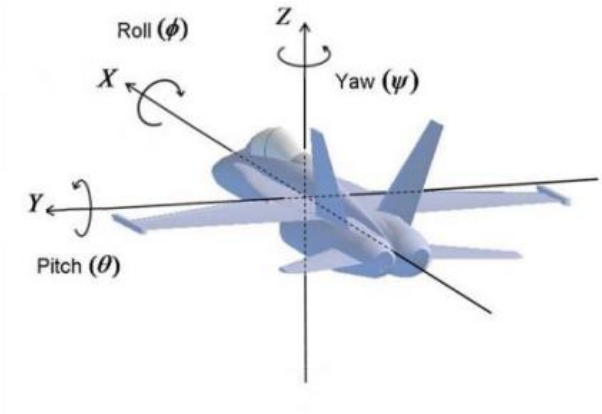


Figura 9. Ángulos de Euler utilizando de ejemplo un avión.

Situando el objeto de frente, siendo de la Figura 9 el eje Z el que atraviesa el objeto en vertical, el eje Y el transversal y el eje X el que atraviesa de manera transversal, se pueden reconocer con más facilidad los movimientos descritos.

Se utiliza un modelo 3D de una cabeza humana genérica y la posición de los puntos faciales de referencia obtenidos con el predictor de landmarks, se calcula la rotación y la translación de la cabeza.

4.3.5. Indicador de la dirección de la mirada (Gaze)

A este indicador se le conoce como Gaze Score, y se obtiene a partir de la distancia euclidiana entre el punto central del ojo y el punto central de la pupila. De esta manera, se puede saber si el conductor, cuando se encuentra con la cabeza orientada hacia el frente, se encuentra mirando también hacia el frente o si se encuentra con la mirada distraída.

La región de interés (ROI, por sus siglas en inglés) del ojo ya queda definida por los puntos clave del predictor, sin embargo, para identificar la pupila y su centro se utiliza la transformada de Hough.

Se utiliza la siguiente fórmula:

$$Gaze\ Score = \frac{distancia\ L2}{ancho\ del\ ojo} \quad (4)$$

Donde:

- Distancia L2: distancia euclidiana entre el punto central del ojo y el punto central de la pupila.
- Ancho del ojo: distancia de extremo a extremo del ojo.

5. DESARROLLO DEL SISTEMA

En este apartado se va a describir el proceso de desarrollo del sistema de detección de la somnolencia basado en la monitorización del rostro.

5.1. HARDWARE

El sistema de adquisición y procesamiento de imágenes está compuesto por una Raspberry Pi, encargada de ejecutar el programa, de una cámara y de una lámpara infrarroja para sortear los problemas de iluminación deficiente. La lámpara permite que el sistema pueda utilizarse en condiciones de iluminación diversas, tanto en ambientes diurnos como nocturnos.

5.1.1. Raspberry Pi

Se ha utilizado una Raspberry Pi 4 Modelo B de 8 GB de RAM, un ordenador de pequeño tamaño y bajo coste en el que se realizará el procesamiento de las imágenes recogidas por la cámara.

Es el modelo que posee mejores características técnicas y de hardware que el resto de las versiones anteriores, manteniendo un precio competitivo y asequible. Incorpora una RAM mayor (el doble que el modelo Raspberry Pi 3 Modelo B), puertos USB 3.0 y USB-C, Bluetooth 5.0 BLE, conectividad Wi-Fi, una CPU con 1,5 GHz de frecuencia de reloj, conector GPIO de 40 pines y, como detalle a destacar, incluye conectores CSI, DSI y HDMI que hacen a este modelo tener una gran compatibilidad con la mayoría de los accesorios. El puerto CSI es específico para realizar la conexión con una cámara, característica que es ideal para este proyecto. El puerto HDMI permite la conexión a un monitor para utilizar la Raspberry como si de un PC se tratase, y así junto a la conectividad Wi-Fi facilitar la instalación de todo lo necesario a través de Internet en la Raspberry.

Figura 10. Imagen de la Raspberry Pi 4 Modelo B.

La alimentación de esta placa funciona a 5,1 V, lo que implica que, si se va a utilizar en un entorno en el que solo existan tomas de 12 V, como en un vehículo, hará falta un conversor de 12 V a 5 V.

La Raspberry hace uso de un sistema operativo, el Raspberry Pi OS, que se basa en Linux y está muy optimizado. De hecho, la mayoría de las distribuciones de Linux incorporan Python por defecto, que es el lenguaje de programación que se va a utilizar en este proyecto, como es el caso de Raspberry Pi OS. La versión de SO utilizada ha sido la de 32 bits.

La tarjeta de memoria que se ha escogido para este proyecto ha sido una microSDHC con 256 GB, de clase 10 con una velocidad de transferencia de hasta 130 MB/s, de clase de vídeo V30 apta para resolución de 4K y de clase de rendimiento A2.

5.1.2. Cámara

Para capturar las imágenes dentro del vehículo se va a utilizar el dispositivo óptico de una cámara. En las aplicaciones de visión artificial, la cámara es un elemento clave a la hora de diseñar el hardware.

El ámbito de aplicación de este proyecto requiere una cámara que sea capaz de adquirir imágenes en condiciones de luminosidad diversas, ya que se conduce tanto durante el día como durante la noche, con nubes o sin ellas; a cualquier hora del día y en cualquier condición meteorológica. Las cámaras que más habituales son aquellas que incluyen un filtro de luz infrarroja que solamente permiten capturar la luz del espectro visible. La principal razón es evitar que la imagen tenga tonos rojizos debido a la captación de las longitudes de onda infrarrojas de las imágenes.

En el caso de este proyecto hará falta una cámara que sí tenga la capacidad de adquirir imágenes en el espectro infrarrojo, porque cuando sea de noche y no haya luz visible, se necesitará capturar igualmente el rostro del conductor. Para no deslumbrar al conductor durante la conducción, no se ha considerado la opción de añadir una lámpara de luz visible, y se ha decidido incorporar una lámpara infrarroja descrita en el siguiente apartado. Gracias a la acción de esta lámpara, cuando no exista luz visible, el habitáculo se iluminará con luz infrarroja, que no será visible al ojo humano pero que permitirá a la cámara continuar con la detección del rostro del conductor.

Para ello, se ha utilizado la cámara Pi NoIR Camera v2, que es la cámara oficial que tiene la Raspberry para realizar fotografía infrarroja. Esta cámara no incorpora el filtro de luz infrarroja mencionado, por lo que deja pasar las longitudes de onda de este espectro. Con esta cámara la imagen tomada tendrá los tonos rojizos mencionados, pero esto no supondrá ningún contratiempo porque el color de las imágenes no resulta importante en esta aplicación.

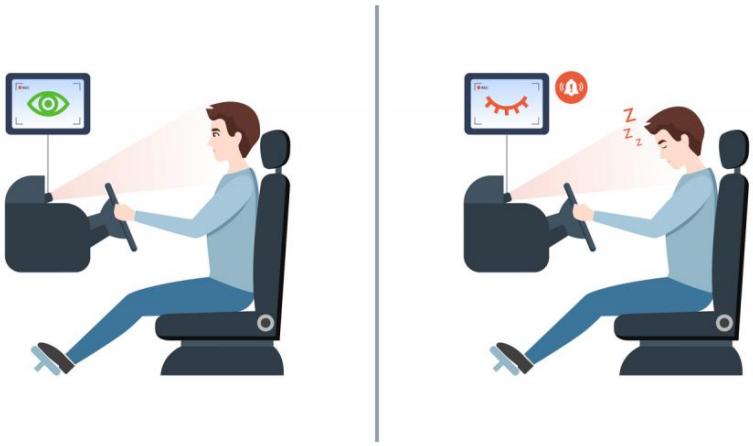


Figura 11. Imagen de ejemplo de funcionamiento del sistema de detección de somnolencia.

5.1.3. Lámpara IR

Dentro del espectro electromagnético completo, el ojo humano únicamente es capaz de percibir una pequeña parte llamada espectro visible. A esta radiación en el rango de longitudes de onda de 400 nm hasta los 750 nm se le conoce comúnmente como luz visible.

La zona de luz infrarroja comprende las longitudes de onda desde los 750 nm hasta los 1000 µm. De todo ese rango, se ha escogido un modelo con longitud de onda de 850 nm. Será una lámpara de LED infrarrojos formada por 24 LED que puede iluminar hasta a 20 m de distancia. Incluye una fotorresistencia, conocida como LDR, cuya resistencia varía en función de la cantidad de luz visible que recibe su superficie. Esto significa que será capaz de regular su funcionamiento, de manera que cuando exista luz visible, no se encenderán los LED y lo hará cuando ya no detecte dicha luz.

La elección de esta longitud de onda se fundamenta en que, al quedar cercana al espectro visible, emitirá un pequeño resplandor rojo cuando esté activa que permitirá saber durante los ensayos que está funcionando correctamente.

Respecto a su alimentación para el montaje en el vehículo, es de 12 Vdc, lo cual supone una gran ventaja para su montaje porque como ya se ha mencionado anteriormente, las tomas habituales en los vehículos son de 12 V.

5.2. SOFTWARE

Para desarrollar el algoritmo de procesamiento de las imágenes en este proyecto se han tenido en cuenta diferentes aspectos que se desarrollan a continuación, como la plataforma de desarrollo del código y el lenguaje de programación utilizado.

5.2.1. Python

Python es actualmente el lenguaje más popular para aplicaciones de visión artificial. Cuenta además con una gran comunidad de software libre en la que apoyarse para llevar a cabo cualquier desarrollo.

Es un lenguaje fácil de aprender, sencillo de utilizar debido a su legible sintaxis, eficiente y muy versátil a la hora de escoger una plataforma en la que ejecutarlo. Es fácilmente portable a plataformas como la Raspberry Pi. Dispone de una gran cantidad de librerías de visión artificial y aprendizaje automático a su alcance, como son OpenCV o Dlib, claves en este proyecto.

El sistema operativo Raspberry Pi OS trae por defecto instalado *Thonny*, un entorno de programación dedicado a Python. Se puede utilizar también para depurar código directamente desde la Raspberry Pi.

5.2.2. VSCode

El editor de código escogido para este proyecto es Visual Studio Code (VSCode), en el que se puede desarrollar y ejecutar código en diferentes lenguajes de programación, como por ejemplo Python. Es gratuito y está disponible tanto para Windows como para masOS o Linux.

Es fácilmente personalizable, además de tener una interfaz intuitiva y cómoda. Una parte realmente importante para escoger este editor es que posee una buena cantidad de extensiones que se pueden descargar desde este mismo. Existen extensiones de todo tipo: ayudas para depurar y limpiar el código, autocompletado, personalización del formato de comentario, reorganización de código, etc.

Se ha utilizado este editor para desarrollar el código en un entorno con sistema operativo Windows, pero que luego es fácilmente trasladable a la Raspberry Pi para su ejecución.

5.2.3. Github

Github es una de las principales plataformas en las que poder alojar código de cualquier desarrollador que quiera compartirlo. A estos portales se les conoce como repositorios, y la finalidad es que se cree una comunidad colaborativa para hacer aportaciones y mejorar los códigos.

Se ha decidido alojar este proyecto en esta plataforma para ofrecer su uso y divulgación a cualquier usuario de internet.

5.3. CÓDIGO FUENTE

En primer lugar, se muestra en la Figura 12 el diagrama de flujo que representa el proceso que realiza el algoritmo desarrollado para dar una primera descripción general. En el diagrama se indican los pasos que se han considerado en el código y la toma de decisiones que se requieren en él.

El objetivo principal del algoritmo es clasificar en 5 alarmas distintas no excluyentes entre sí, que son las siguientes: dormido, distraído, mirada distraída, somnoliento y bostezando. Estas alarmas han sido las elegidas para definir un cuadro de somnolencia en un sujeto humano. Cada una de estas alarmas está directamente relacionada con uno o más indicadores anteriormente expuestos, que son el EAR, MAR, PERCLOS, Gaze y Head Pose.

En la Tabla 2 se muestran los umbrales de los indicadores de este proyecto. Respecto a los umbrales de indicador, el EAR_thresh y el MAR_thresh se obtienen de la rutina de inicialización que se explica más adelante. Este último, si no es posible obtenerlo de dicha rutina, se ha decidido de manera experimental que valga 0,35, pues la apertura máxima promedio de una boca se encuentra alrededor de la unidad. Los umbrales relacionados con la mirada y la cabeza se han obtenido de manera experimental durante las pruebas de funcionamiento, ya que dependen de los ejes de referencia que se utilizan para calcular sus posiciones. Respecto a los umbrales de tiempo, el EAR_time_thresh se ha decidido situar en 3 segundos por corresponderse con la duración de un microsueño y el MAR_time_thresh se ha decidido situar en 3 segundos porque la duración de los bostezos suele ubicarse entre 3 y 10 segundos. Los umbrales relacionados con la mirada y la cabeza se han determinado de manera

experimental, concluyendo que una distracción de la mirada de 4 segundos sería excesiva y una distracción de la cabeza de 6 segundos consecutivos también. El umbral del PERCLOS se ha fijado en 20 % basando esta decisión en los estudios de investigación y trabajos como [10] en los que se hace uso de este indicador.

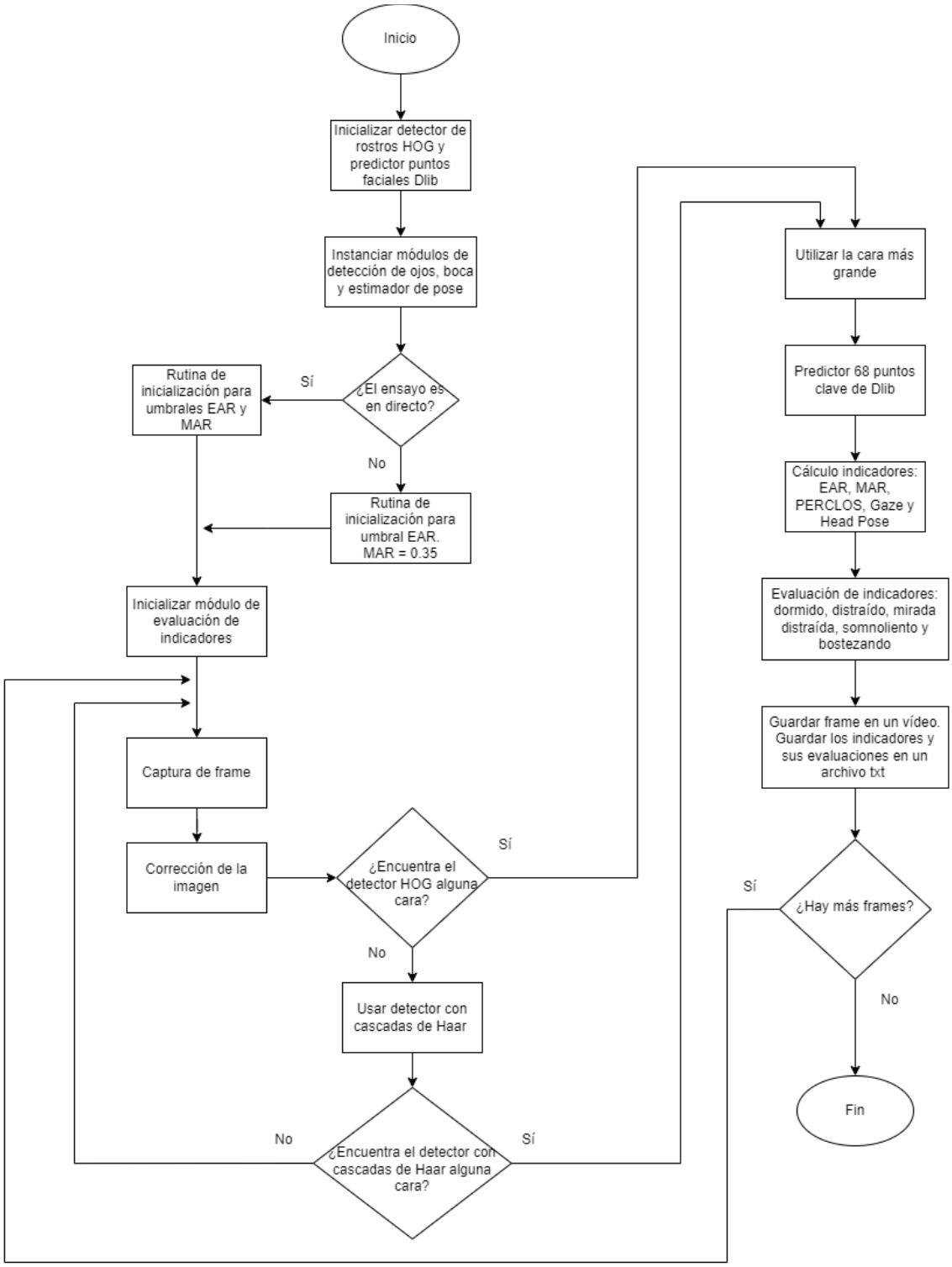


Figura 12. Diagrama de flujo del algoritmo desarrollado.

Tabla 2. Umbrales utilizados para el algoritmo de detección de somnolencia.

Área de aplicación	Umbral	Definición	Valor
	de indicador		
Ojos	EAR_thresh	Límite del EAR por debajo del cual se considera que los ojos están cerrados	Adaptativo en función de la rutina de inicialización
Boca	MAR_thresh	Límite del MAR por encima del cual se considera que la boca está muy abierta	Adaptativo en función de la rutina de inicialización en los casos que pueda utilizarse. Si no, será 0.35
Mirada	Gaze_thresh	Límite del Gaze por encima del cual se considera que la mirada no está centrada	0.4
Cabeza	Pitch_thresh	Límite del ángulo pitch para considerar la cabeza no centrada	165
	Yaw_thresh	Límite del ángulo yaw para considerar la cabeza no centrada	30
	Roll_thresh	Límite del ángulo roll para considerar la cabeza no centrada	20
	de tiempo		
Ojos	EAR_time_thresh	Límite de tiempo para tener los ojos cerrados por debajo de EAR_thresh de manera consecutiva	3 segundos
Boca	MAR_time_thresh	Límite de tiempo para tener la boca abierta por encima de MAR_thresh de manera consecutiva	3 segundos
Mirada	Gaze_time_thresh	Límite de tiempo para tener la mirada no centrada, según Gaze_thresh, de manera consecutiva	4 segundos
Cabeza	Pose_time_thresh	Límite de tiempo para tener poses de la cabeza distraídas de manera consecutiva	6 segundos
Ojos	PERCLOS_thresh	Límite porcentual del tiempo permitido para tener los ojos cerrados por debajo del umbral EAR_thresh durante 60 segundos	0,2 (20 %, que son 12 segundos)

La definición de los umbrales constituye una parte clave en el desarrollo de este sistema, ya que de rebasarlos dependerá si se detecta un cuadro de somnolencia o no en el conductor. Los umbrales que se corresponden con los índices EAR y MAR, es decir, EAR_thresh y MAR_thresh, se han considerado definir en la rutina de inicialización del sujeto. La razón principal es que la apertura de los ojos y la boca son características muy dependientes de la persona, porque según sus rasgos tendrán unos límites u otros. Se le da prioridad al EAR_thresh porque es el que más influencia tiene, además de ser más sencillo de calcular. El MAR_thresh es posible definirlo mediante la rutina de inicialización si el sujeto al que se le realiza el ensayo mantiene la boca abierta durante el tiempo que dura esta rutina. Si dichas condiciones no son posibles, como en el caso de realizar ensayos con datasets, entonces se define como valor umbral 0,35.

Cabe destacar que los parámetros que hacen referencia a la dirección de la cabeza y la mirada se han considerado centrados cuando el sujeto esté orientado hacia la cámara. Se ha decidido así porque también los detectores de rostros obtienen mejores resultados con el rostro lo más frontal posible. Por esto, la prioridad en el montaje en el vehículo será situar la cámara lo más centrada posible al conductor.

El código consta de diferentes módulos, con la finalidad de ordenarlo y separarlo por funcionalidades. A continuación, se expone detalladamente cada uno de estos módulos, que pueden consultarse en los Anexos de este trabajo.

Main

Es en el módulo principal donde se realizan las tareas principales del código y desde donde este se ejecuta. En este módulo se activa la cámara para tomar las imágenes del conductor, se inicializa el detector y el predictor de rostro, se llama a las funciones que calculan los índices de somnolencia y se guarda la información relevante.

El punto de inicio es la detección del rostro. Tal y como se ha comentado anteriormente en el apartado de la propuesta de desarrollo, para detectar el rostro se va a utilizar el modelo pre-entrenado de Dlib llamado “get_frontal_face_detector”, basado en el método de los gradientes orientados HOG. El detector de Dlib tiene un buen funcionamiento, pero tiene limitaciones al intentar detectar rostros que no se encuentren en una posición relativamente frontal. Para solventar esto, se ha realizado una combinación de HOG con el método de las Cascadas de Haar. El detector con prioridad será HOG debido a que es el algoritmo que ofrece mejores resultados, pero en el caso de que este no consiga detectar una cara, se utilizará el clasificador de Cascadas de Haar.

La captura de la imagen de entrada se realiza con la librería OpenCV, que captura imágenes en formato BGR, lo que significa que el orden de las bandas de color se representa en ese orden: azul, verde y rojo. Para que la detección del rostro sea satisfactoria, al algoritmo de detección se le entrega una imagen con la mayor calidad posible, y para ello se utilizan técnicas de preprocesamiento de imagen. Se han utilizado las siguientes:

- Conversión de la imagen en formato BGR a escala de grises: la imagen que toma la cámara está representada en un modelo de color de tres canales, sin embargo, para el detector escogido se necesita la imagen en un solo canal, es decir, en escala de grises.
- Filtro bilateral: se trata de un filtro no lineal que preserva los detalles y reduce el ruido en las imágenes.

La imagen obtenida después del preprocesado de la imagen se utiliza de entrada al detector de rostros. La salida de este detector es un vector cuyo contenido son las coordenadas de los rectángulos que tienen en su interior un rostro. En este caso, solo se le dará relevancia a la cara más grande que se haya encontrado, que se presupone será la del conductor. De esta manera, se acota la zona de la imagen en la que hay que buscar y localizar los elementos de un rostro. Esta tarea la lleva a cabo el predictor "shape_predictor_68_landmarks", que como ya se ha mencionado en este trabajo es el predictor de los 68 puntos faciales clave de Dlib. La salida de este predictor será un vector, llamado "landmarks", con las posiciones de los puntos faciales detectados en la cara del conductor.

A partir de este vector se localizan los ojos y la boca y se representan en la imagen los contornos de ambos. Después, este mismo vector se introduce en las llamadas de los módulos: Eye_Mouth_Detector_Module para obtener los indicadores EAR, MAR y Gaze; Pose_Estimation_Module para obtener los valores de roll, pitch y yaw; Attention_Scorer_Module para calcular el PERCLOS y evaluar el resto de los indicadores. Las puntuaciones de los indicadores se muestran en el frame analizado que luego se guardará en un vídeo. El hecho de guardar en un vídeo los frames no es necesario una vez se compruebe su eficacia, pero en los ensayos puede ser muy relevante disponer del vídeo y poder reproducirlo con posterioridad para realizar comprobaciones.

Finalmente, se escribe en un fichero de texto los resultados obtenidos en cada frame para luego poder realizar un análisis de estos.

Eye_Mouth_Detector_Module

Es el módulo en el que, a partir de los puntos faciales clave, se calculan el índice EAR, el índice MAR y el Gaze Score del sujeto en cada frame. Para ello, se introduce como entrada los puntos clave de la cara del conductor.

Pose_Estimation_Module

Con el código contenido en este módulo, se realiza la estimación de la pose de la cabeza del conductor, calculando los ángulos de Euler. De la misma forma que en el módulo anterior, se introduce como entrada los puntos clave de la cara del conductor.

Attention_Scorer_Module

En este módulo se realiza la evaluación de los parámetros calculados en el módulo anterior. También calcula el índice PERCLOS a partir del índice EAR de los ojos. Se evalúa al sujeto de manera que se pueden dar cinco alarmas distintas: dormido, distraído, con la mirada distraída, bostezando y somnoliento. Estas se pueden dar simultáneamente.

- La alarma de **dormido** está relacionada con la apertura de los ojos, con el índice EAR. Hace referencia a un microsueño. Cuando un sujeto permanece con los ojos cerrados por debajo del umbral establecido durante un tiempo establecido, por ejemplo, por debajo del 75 % de la apertura y durante 3 segundos, se considera que está dormido.

- La alarma de **distraído** se obtiene a partir de la pose del sujeto, observando la posición de su cabeza. Si alguno de los ángulos de Euler (roll, pitch y yaw) que se calculan para estimar la posición de la cabeza se encuentra por encima de algún umbral, durante determinado tiempo, se considera que el sujeto no tiene la cabeza orientada hacia el frente y por tanto está distraído.
- La alarma de **mirada distraída** hace referencia a cuando el sujeto tiene la cabeza posicionada hacia el frente, pero la dirección de su mirada no está centrada. Para ello se utiliza la puntuación de la mirada (gaze en inglés), cuando esta se encuentra por encima de un umbral establecido, se considera que hay distracción en la mirada.
- La alarma de **bostezando** se produce cuando la apertura de la boca se encuentra por encima de determinado umbral durante un tiempo determinado. Por ejemplo, si la boca está abierta más de un 50% durante 3 segundos consecutivos. En este caso, se considera que el sujeto se encuentra bostezando.
- La alarma de cansado, que en este caso se utiliza como **somnoliento**, hace referencia a la cantidad de tiempo que pasa el sujeto con los ojos cerrados por debajo de determinado umbral, es decir, el índice PERCLOS. Por ejemplo, si el sujeto permanece más de un 20% del tiempo en que se produce el ensayo con los ojos cerrados por debajo de dicho umbral, se considera que se encuentra somnoliento.

Respecto al cálculo del PERCLOS, en el apartado de *Elección de Indicadores* se expone cómo este se realiza, y resulta ser dividiendo el tiempo en que los ojos permanecen cerrados por debajo de determinado umbral entre el total del tiempo observado. Sin embargo, se puede plantear de diferentes maneras.
Si quiere obtenerse el PERCLOS todo el tiempo en que se utiliza el sistema y no se sabe cuánto durará el trayecto, puede calcularse con otra unidad de tiempo que se decida. Si se utilizase el tiempo actual en cada momento, al principio del trayecto se obtendrían puntuaciones muy altas debido a que el tiempo observado es muy corto, y sería complicado detectar en qué momentos se han producido más oclusiones o de más duración.
En este trabajo se ha planteado realizar el cálculo cada minuto por ser la referencia de tiempo más habitual. Se acumulará el tiempo en que los ojos se encuentran cerrados por debajo del umbral establecido, y luego se dividirá este dato entre 60 segundos. De esta manera, se puede observar el PERCLOS durante cada minuto, pero también se puede calcular el PERCLOS global promediando las puntuaciones obtenidas al final de cada minuto, y además se pueden detectar los momentos en que ha habido episodios de más oclusiones analizando el registro del trayecto.

Se presenta a continuación un diagrama de flujo de la función que calcula el PERCLOS. Ha de tenerse en cuenta que previa a esta función se han definido las siguientes variables globales:

- prev_time = 0, es una variable auxiliar para luego calcular el tiempo que ha pasado.
- eye_closure_counter = 0, es el contador de los frames en que los ojos han estado cerrados, es decir, el parámetro EAR ha sido menor que su umbral, el EAR_thresh.
- delta_time_frame = 1/capture_fps, es el tiempo estimado que dura un frame, donde capture_fps son los FPS (frame per second) de captura de los frames.
- perclose_time_period = 60, es el tiempo en segundos definido para calcular el PERCLOS, en este caso, 60 segundos

La función del PERCLOS se calcula para cada frame que se procesa.

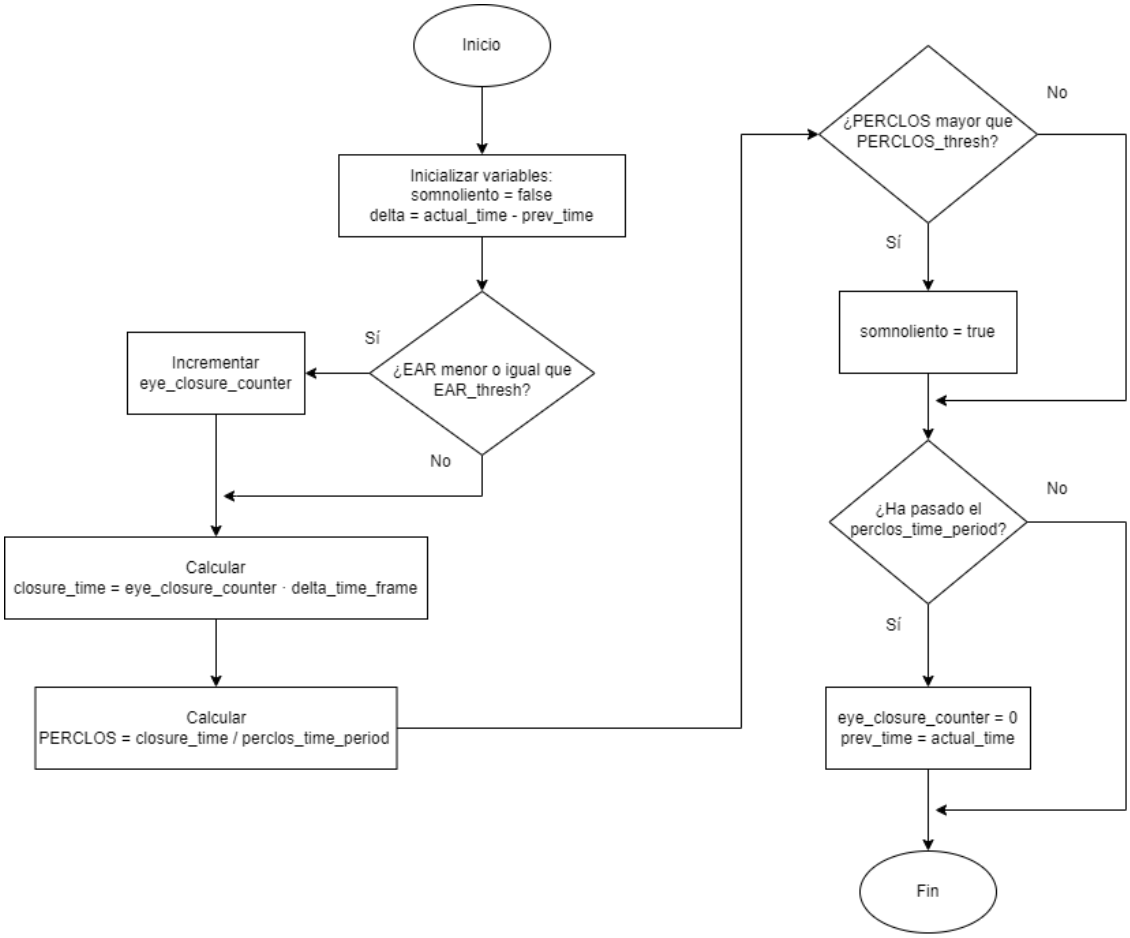


Figura 13. Diagrama de flujo de la función que calcula el PERCLOS.

Initialization

Este módulo se utiliza para realizar una rutina de inicialización al sujeto que va a comportarse como conductor en el vehículo. La razón de esta rutina es que cada persona tiene unas características fisiológicas diferentes, por lo que se ha incorporado un cálculo personalizado de los índices EAR y MAR para monitorizar la apertura de los ojos y la boca de la manera más precisa posible. Es imprescindible que para que el índice MAR funcione correctamente, el sujeto debe permanecer con la boca lo más abierta posible durante los 5 primeros segundos de ejecución del sistema. De esta manera puede obtenerse una estimación de la capacidad de apertura de la boca de dicha persona.

El umbral EAR_thresh se define como tres cuartas partes de la media del EAR durante los segundos en que dura la inicialización:

$$EAR_{thresh} = EAR_{mean} \cdot 3/4 \tag{5}$$

Se define por tanto que, si los ojos se encuentran cerrados por debajo del 75 % de su capacidad de apertura, se encuentran demasiado cerrados. Si la puntuación está por debajo de este umbral durante el tiempo estipulado para tener los ojos cerrados, se relacionará con los estados de dormido y somnoliente.

El umbral MAR_thresh se calcula como la mitad de la media del MAR durante los segundos en que dura la rutina de inicialización:

$$MAR_{thresh} = MAR_{mean\_init}/2 \quad (6)$$

Se decide de esta manera considerar que cuando la boca esté abierta por encima del 50 % de su capacidad de apertura, está lo suficientemente abierta como para en caso de superar el tiempo de bostezo, considerarse un bostezo.

6. PRUEBA DE FUNCIONAMIENTO

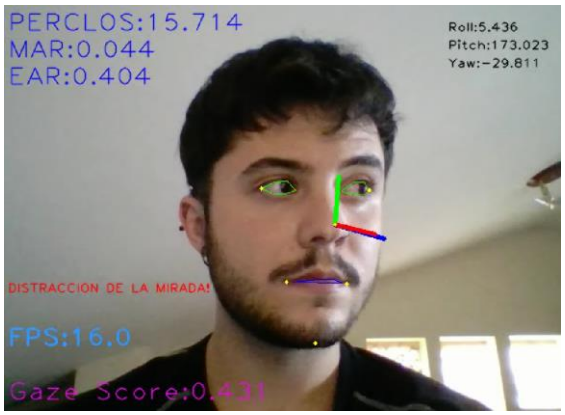
Previamente a los ensayos, se han realizado dos pruebas de funcionamiento para obtener una primera toma de contacto con el desempeño del algoritmo:

- Utilizando un ordenador portátil ASUS VivoBook de 16 GB de RAM, con un procesador AMD Ryzen 7 4700U, y la webcam que lleva incorporada, una USB2.0 HD UVC WebCam.
- Utilizando la Raspberry Pi y la NoIR Camera.

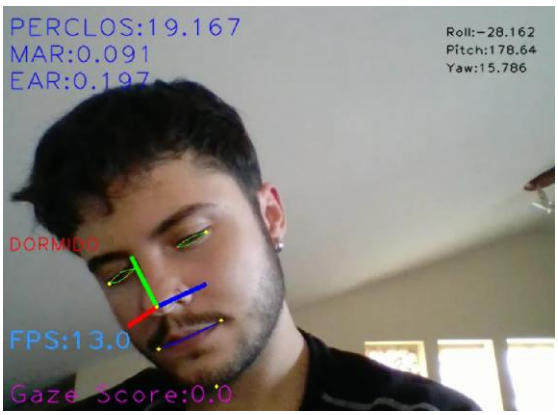
6.1. CON ORDENADOR Y WEBCAM

A la par que se ha realizado el desarrollo del algoritmo, se han llevado a cabo pruebas de funcionamiento con el ordenador portátil y su webcam.

En estas pruebas el sujeto se encuentra en posición frontal hacia la cámara y trata de reproducir un comportamiento somnoliente. Se prueba a bostezar con diferentes tiempos, entrecerrar y cerrar los ojos, mover la cabeza hacia los lados y de arriba abajo, desviar la mirada durante varios segundos, etc. Mientras se van procesando los frames, se muestran por pantalla en forma de vídeo, en el que se resaltan los contornos de la boca y los ojos, y se escriben las puntuaciones de diferentes indicadores y las alarmas que se van produciendo. Estos vídeos se guardan para poder revisarlos con posterioridad.



a)



b)

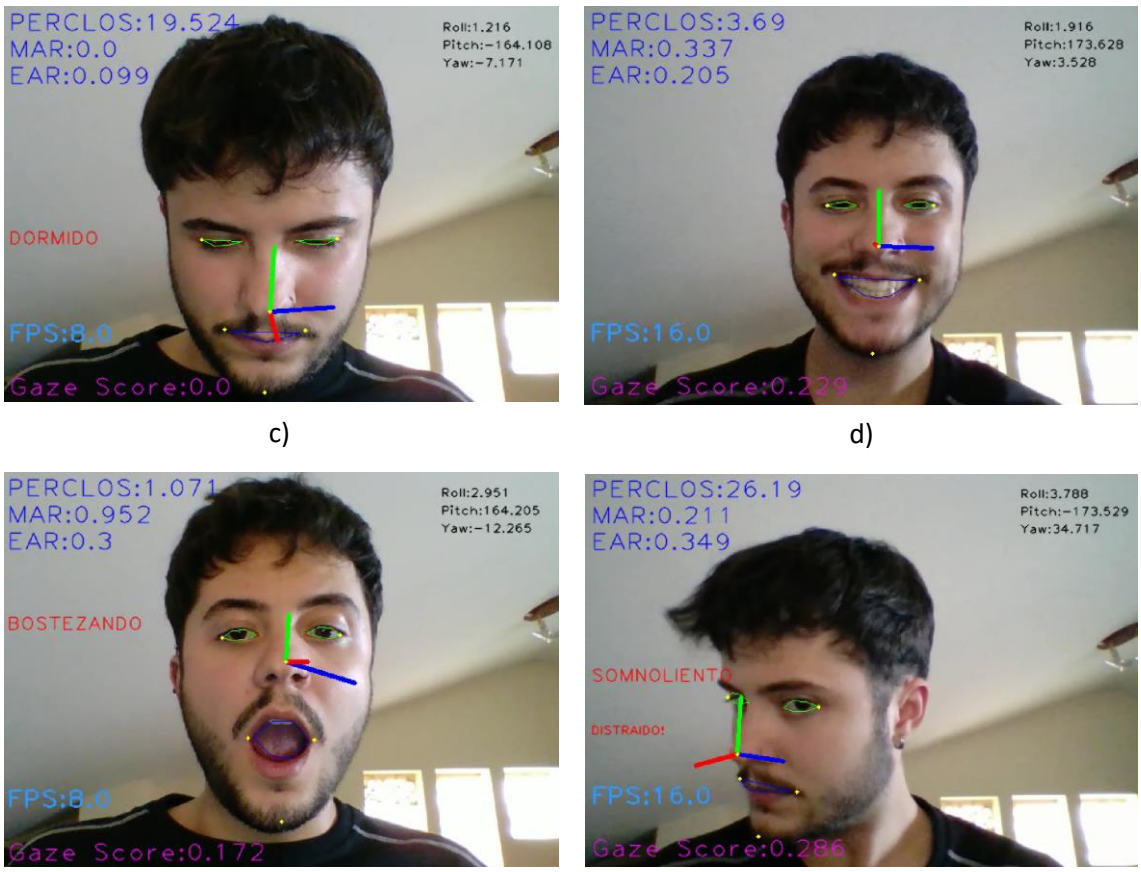
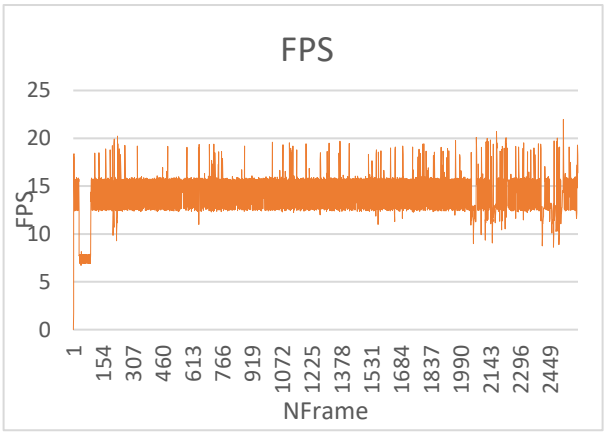


Figura 14. Ejemplos de prueba de funcionamiento del algoritmo con el ordenador portátil y la webcam.

En las imágenes anteriores se puede ver como las puntuaciones de los indicadores y las alarmas son coherentes con lo que está sucediendo. Entre todas las imágenes se han simulado las cinco diferentes alarmas: dormido, somnoliento, bostezando, distraído y con la mirada distraída. En la Figura 14, en la imagen d) se ha comprobado que, aunque el sujeto sonría, si no lo hace por encima del umbral del bostezo ni durante demasiado tiempo no saldrá la alarma de bostezando. Se han comprobado también los límites de detección del algoritmo, como es el caso de la imagen f).



Aunque se presentan picos en la velocidad de procesamiento, la velocidad media es de 14,8 frames por segundo (FPS), o lo que es lo mismo, se procesa un frame cada 0,067 segundos.

Figura 15. Gráfico representando los FPS del algoritmo ejecutándose en el ordenador portátil.

6.2. CON RASPBERRY Y NOIR CAMERA

La prueba de funcionamiento es la misma que con el ordenador portátil y su webcam, pero sustituyendo el hardware por la Raspberry y la NoIR Camera. De esta manera, se prueba la captación de imágenes por parte de la cámara y la capacidad de la Raspberry para procesarlas.

Figura 16. Ejemplos de prueba de funcionamiento del algoritmo con la Raspberry Pi y la NoIR Camera.

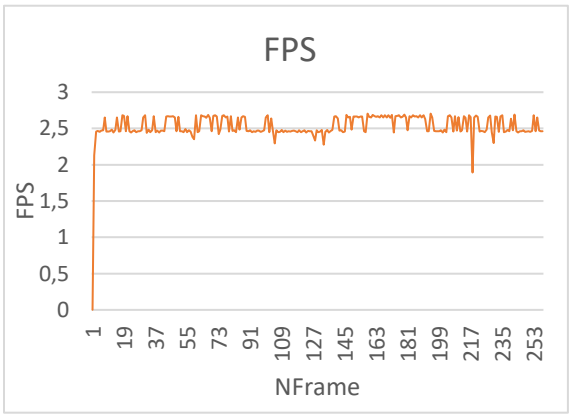


Figura 17. Gráfico representando los FPS del algoritmo ejecutándose en la Raspberry Pi.

En el caso de la Raspberry, la tasa de FPS se encuentra comprendida entre 2,5 y 3 FPS, lo que se traduce en que se procesa aproximadamente un frame cada 0,36 segundos.

7. ENSAYOS

Los ensayos se han dividido en tres partes con el objetivo de probar el sistema desde tres perspectivas diferentes. La fase de pruebas se ha realizado a partir de los diferentes medios expuestos a continuación:

- Con datasets “YawDD dataset”, “Nitymed dataset” y “UTA Real-Life Drowsiness dataset”
- En conductores físicos utilizando un juego de conducción como simulador.
- En conductores físicos en condiciones reales de conducción.

7.1. ENSAYOS CON DATASETS

El objetivo de los ensayos con datasets es probar el algoritmo en variedad de sujetos antes de realizar el montaje en el vehículo de la Raspberry.

Los ensayos en datasets se han realizado con el ordenador portátil porque no había necesidad de utilizar un dispositivo portátil de dimensiones reducidas y con cámara, ya que los vídeos ya se tienen guardados y solo hay que procesarlos. Además, la velocidad de procesamiento es mayor en este caso que con la Raspberry.

7.1.1. YawDD dataset

El primero de los datasets que se ha probado es el YawDD dataset [14], un conjunto de vídeos tomados por una cámara en el interior de un vehículo, con conductores con características faciales muy diversas hablando, cantando, en silencio y bostezando. Está dedicado especialmente a la detección de bostezos. Las condiciones de iluminación son de luz natural y variable. El formato de vídeo es de 640x480 píxeles en color RGB, a 30 FPS, en formato AVI y sin audio.

El dataset está formado por dos conjuntos de datos según el lugar donde se ha instalado la cámara. En el primero, la cámara se ha instalado debajo del espejo frontal del vehículo, de manera que el conductor no queda frontal a la misma, y consta de 322 vídeos. El segundo conjunto tiene la cámara instalada en el tablero del conductor, de manera más frontal, y consta de 29 vídeos.

Antes de probar el dataset ya se puede predecir que la detección con el segundo conjunto de datos obtendrá mejores resultados que en el primero, ya que la posición de la cámara en el vehículo es un aspecto crucial, y los detectores funcionan mucho mejor de manera frontal. Además, los umbrales de los indicadores para la orientación de la cabeza y la mirada tendrán que adaptarse en el primer conjunto de datos, según la posición de la cámara.

Se ha decidido escoger al azar 10 vídeos del primer conjunto de datos, donde la cámara se ubica en el espejo frontal del habitáculo, y 15 vídeos del segundo conjunto de datos, donde la cámara se ubica frontalmente. Se le ha dado prioridad al segundo por ser el que se alinea más con las especificaciones del sistema desarrollado en este trabajo. Con el procesamiento de estos vídeos se han recogido datos y se ha tratado de validar la coherencia de los resultados.

Cámara frontal

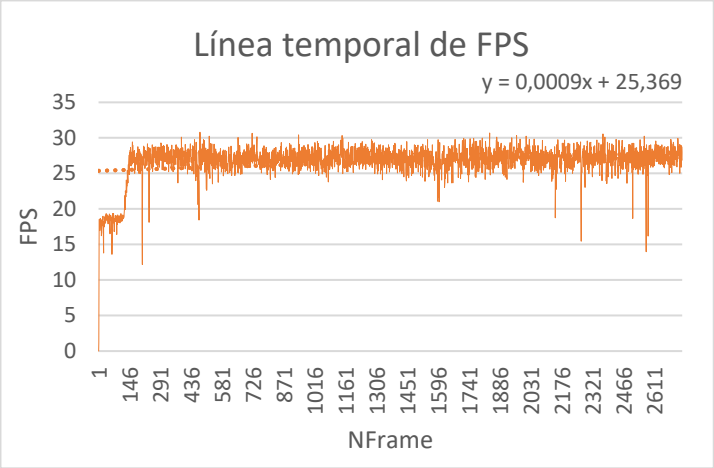
Figura 18. Ejemplos de rostros del YawDD dataset con la cámara frontal con detecciones exitosas.

En la Figura 18 se muestran 4 frames de diferentes vídeos del segundo conjunto de datos, con la cámara en posición frontal. Los sujetos tienen características faciales distintas, etnias diferentes (oriente medio, caucásicos, asiáticos) y algunos llevan gafas.

Son casos de detecciones exitosas en las que el detector ha encontrado la cara del conductor y el predictor ha ubicado los puntos clave faciales del rostro satisfactoriamente. Además, se han obtenido alarmas coherentes con lo que sucede en las imágenes. Por ejemplo, en a) y d) las participantes se encuentran bostezando y aparecen las alarmas correspondientes. En b) el sujeto se encuentra distraído, con la cabeza ladeada, durante varios segundos y así lo refleja la alarma. Además, en el caso anterior se obtienen buenas detecciones del rostro a pesar de la sombra que hay en el rostro debido al ángulo de incisión del sol.

Se puede observar como las gafas de vista no suponen un problema para este algoritmo porque los ojos consiguen detectarse con éxito.

Se analizan los resultados de, por ejemplo, el vídeo de a) en la Figura 18. A continuación, se presentan diferentes gráficas para analizar el comportamiento del algoritmo.



En cuanto a los FPS alcanzados por el algoritmo, se puede observar en la gráfica como se han mantenido con un comportamiento relativamente estable entre 25 y 30 FPS, a excepción de los primeros 121 frames. Los vídeos están grabados a 30 FPS, lo que significa que el frame 121 del vídeo es el segundo 4 del vídeo.

Figura 19. Gráfico de la línea temporal de FPS en el vídeo a) de la Figura 18.

Con la ecuación de la línea de tendencia se puede comprobar que la pendiente es muy próxima a cero, lo que significa que la tendencia es lineal alrededor de 25,369. Calculando la media de FPS se obtiene que son 26,66 FPS de media, por lo tanto, se procesan 26 frames cada segundo, o lo que es lo mismo, se procesa un frame cada $1/26{,}66 = 0{,}037$ segundos. Casi a la velocidad en la que se ha grabado el vídeo.

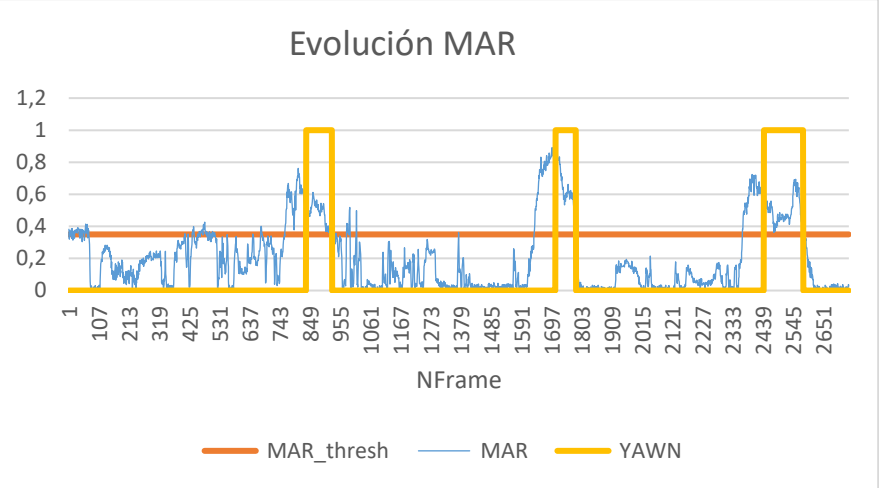


Figura 20. Gráfico de la evolución del MAR y la alarma de bostezo en el vídeo a) de la Figura 18.

Respecto a la evolución a lo largo de los frames del indicador MAR y de las alarmas de bostezos (Yawn), en el gráfico se puede observar que cuando el MAR supera su umbral (MAR_thresh) durante el tiempo establecido para la alarma de bostezo (MAR_time_thresh), esta se activa hasta decaer el valor MAR por debajo del umbral. El tiempo que tarda en activarse se puede aproximar tomando el número de frame en que MAR supera a MAR_thresh y el número de frame en que aparece la alarma de bostezo. Por ejemplo, de 761 a 840 hay 79 frames, que se corresponden con $79/26{,}66 = 2{,}96$ segundos, que es aproximadamente 3 segundos, lo que

se ha establecido en el umbral de tiempo MAR_time_thresh. Se ve como el algoritmo ha detectado 3 bostezos durante el vídeo.

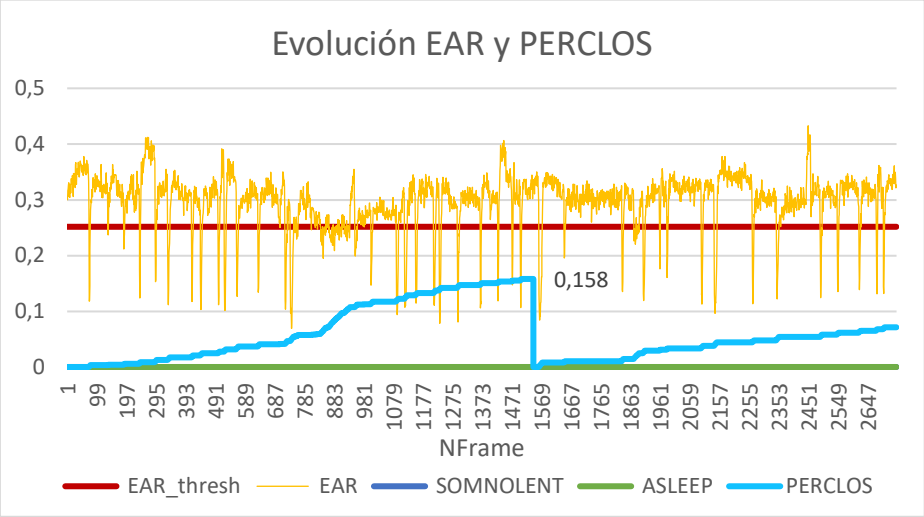
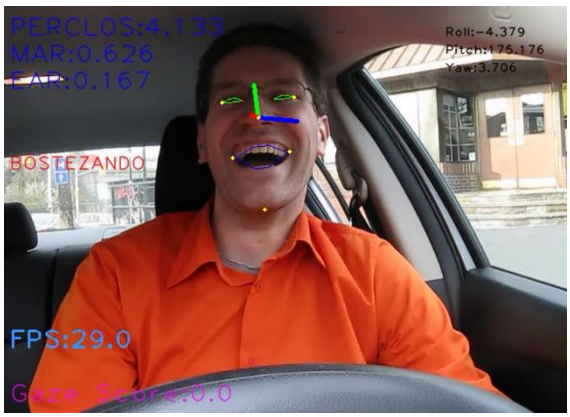


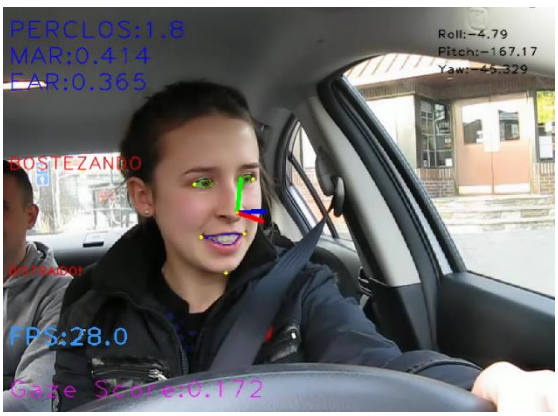
Figura 21. Gráfico de la evolución del EAR, el PERCLOS y las alarmas asociadas en el vídeo a) de la Figura 18.

En la gráfica anterior se observa la evolución del EAR a lo largo de los frames, junto a su umbral (EAR_thresh) y la evolución del PERCLOS. El indicador EAR se encuentra en algunos frames por debajo de su umbral, pero no aparece la alarma de dormido (Asleep) porque no supera el tiempo definido por el umbral de tiempo (EAR_time_thresh).

El PERCLOS, por otra parte, se ha representado en decimal y no en porcentaje para facilitar la representación en el gráfico y no ampliar en exceso el rango del eje vertical. El PERCLOS se va incrementando a medida que se producen oclusiones de los ojos y alrededor del frame 1521 alcanza el minuto de vídeo, con una puntuación de 0,158, que es el 15,8 %. Como se encuentra por debajo del umbral establecido para el PERCLOS, 20 %, no aparece la alarma de somnoliento (Somnolent).



a)



b)

c) d)

Figura 22. Ejemplos de rostros del YawDD dataset con la cámara frontal con detecciones erróneas.

En la Figura 22 se muestran también 4 frames de diferentes vídeos del segundo conjunto de datos, cuyos sujetos también tienen características físicas distintas. En este caso, se trata de detecciones erróneas por diferentes motivos. Sin embargo, hace falta resaltar que el detector ha encontrado en todos los frames el rostro del conductor, aunque luego el predictor de los puntos faciales no haya sido satisfactorio.

En a) y b) se produce un falso positivo de bostezo debido a que el sujeto sonríe con bastante apertura de la boca y durante al menos 3 segundos seguidos, que es el mínimo considerado para un bostezo. Es posible que dicha situación se pudiese corregir con la rutina de inicialización en la que el umbral de apertura de la boca se ajustase al individuo. A pesar de esto, se puede ver como en b) se produce la alarma de distraído correctamente.

En c) se produce la alarma de bostezo correctamente, sin embargo, el contorno de la boca no se ajusta a la boca del sujeto, por lo que el predictor no acaba de tener buenos resultados.

En d) el predictor no encuentra correctamente los puntos faciales clave del conductor, seguramente debido a la sombra que se produce en el interior del vehículo.

Figura 23. Ejemplo de rostro del YawDD dataset con la cámara frontal de mujer con gafas de sol.

En la Figura 23 se muestran dos frames de un vídeo del dataset en el que la conductora lleva gafas de sol. Se trata de un caso especialmente sensible porque las gafas de sol dificultan la monitorización de los ojos, que es un aspecto clave en la detección de somnolencia. En este caso, en la imagen izquierda se ve como el predictor intenta ubicar los ojos a pesar de las gafas de sol. Sin embargo, en la imagen derecha se aprecia como el predictor no ubica bien los puntos clave del rostro de la mujer, el eje de referencia en lugar de situarse sobre la nariz queda desplazado hacia abajo, y finalmente no es capaz de dar positivo en bostezo.

Cámara no frontal

Con la cámara ubicada en el espejo frontal de vehículo, se han ajustado los umbrales referentes al posicionamiento de la cabeza y la dirección de la mirada.

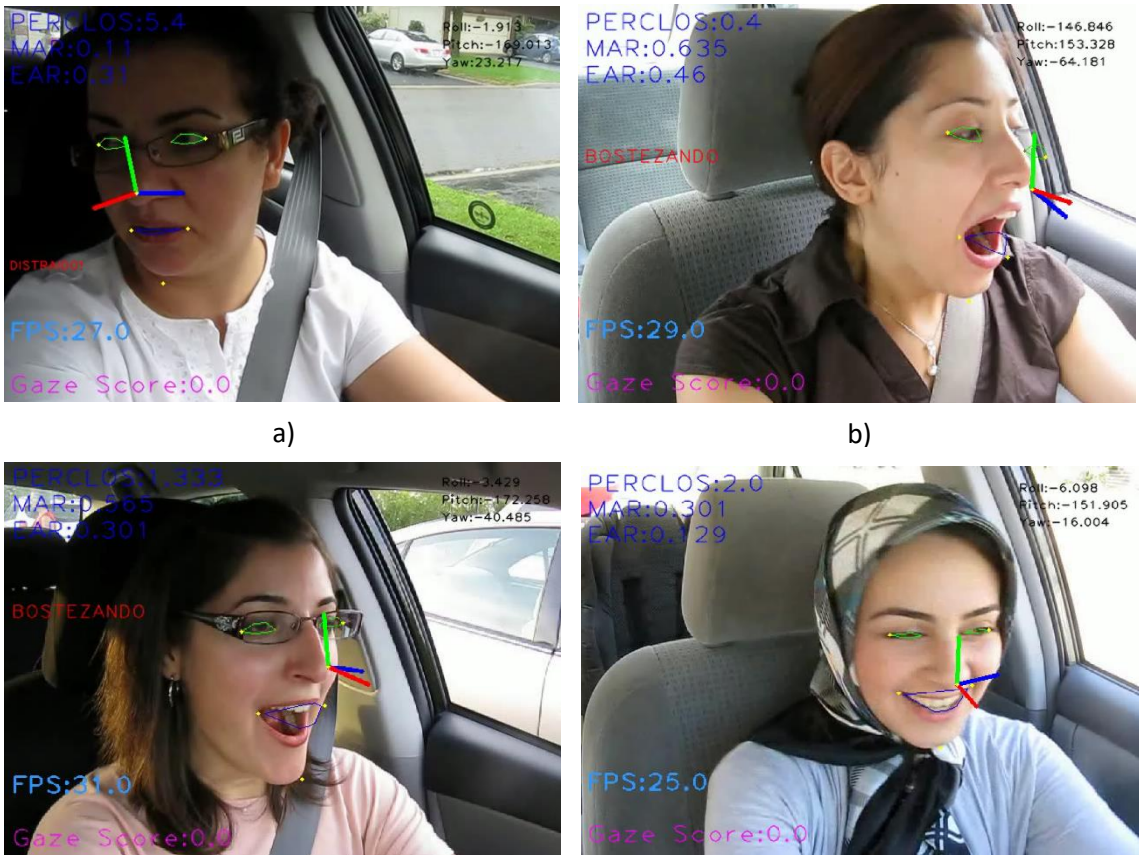


Figura 24. Ejemplos de rostros del YawDD dataset con la cámara no frontal con detecciones exitosas.

En la Figura 24 se puede apreciar como a pesar de la posición de la cámara, situada en el espejo frontal del vehículo, se obtienen buenos resultados. Tanto en b) como en c) no se acaban de ubicar bien puntos clave de la boca, pero aun así se obtiene la suficiente apertura como para detectar los bostezos. En a) se consigue detectar la distracción de la conductora incluso con una iluminación un poco deficiente sobre el rostro. En d) se consigue una detección de los elementos de la cara bastante estable, seguramente debido a la buena iluminación y a la posición más frontal de la cámara que en otros vehículos.

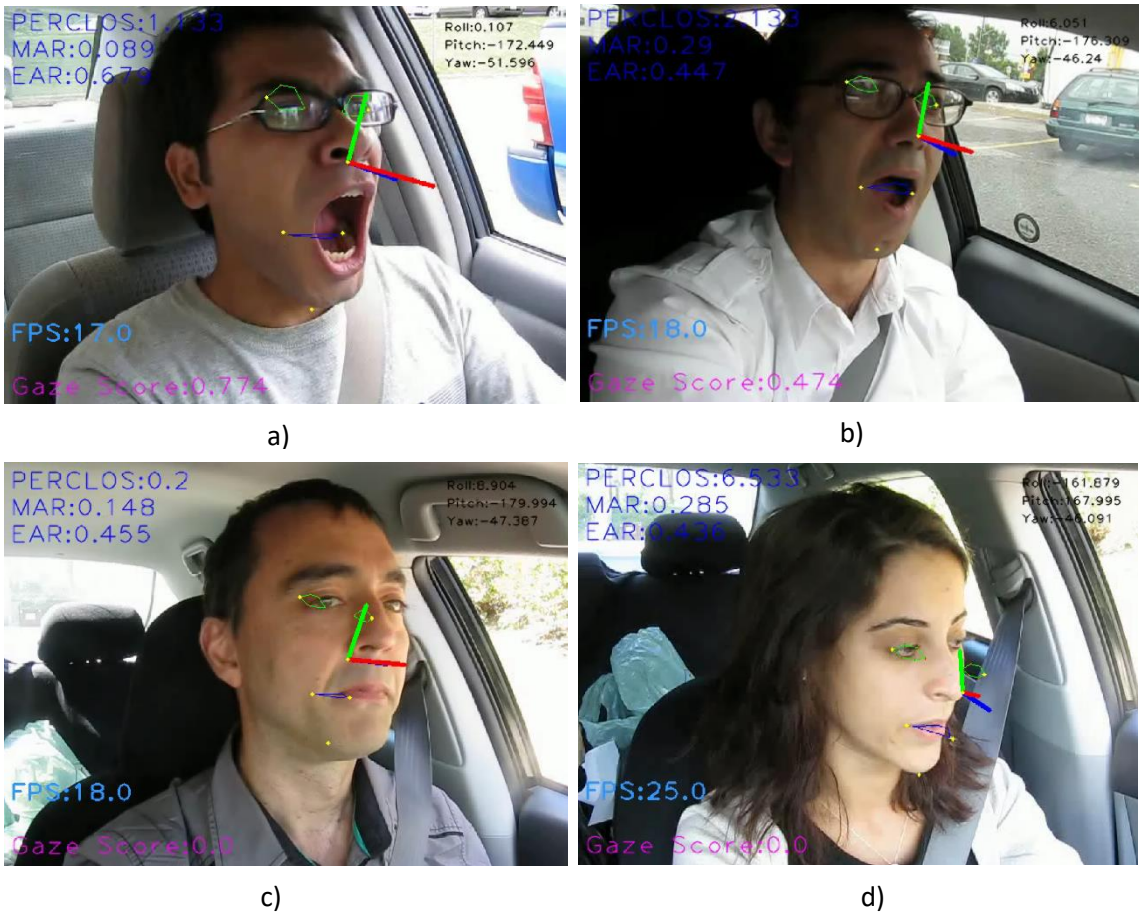


Figura 25. Ejemplos de rostros del YawDD dataset con la cámara no frontal con detecciones erróneas.

En las imágenes anteriores de la Figura 25 se muestran detecciones erróneas con la cámara situada debajo del espejo frontal del habitáculo. En los vídeos se puede comprobar que son detecciones con cierta inestabilidad, en los que hay frames como los que se muestran en estas imágenes, en los que el algoritmo no es capaz de identificar correctamente la posición de la cara y los ojos del sujeto.

7.1.2. Nitymed dataset

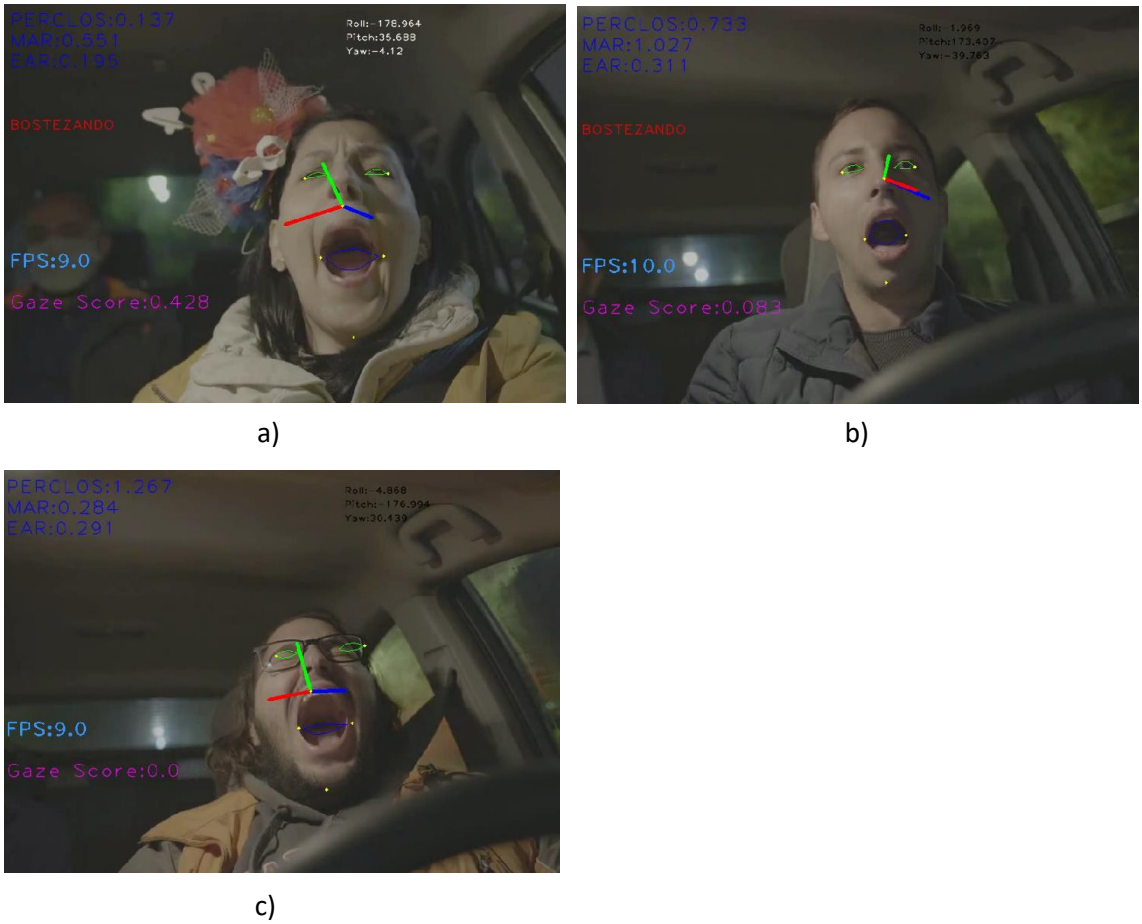
El dataset Nitymed [15], realizado en Patras, Grecia, es un dataset que muestra conductores en vehículos reales y en movimiento, en condiciones de iluminación nocturnas. Los participantes son 11 hombres y 10 mujeres con diferentes características físicas, y consta de un total de 130 vídeos. Estos vídeos se separan en dos conjuntos: yawning (bostezando), donde los conductores bostezan 3 veces en cada vídeo, y microsleep (microsueño), donde los conductores hablan, miran hacia los lados y tienen episodios de microsueños. Los vídeos tienen formato mp4, a una velocidad de 25 FPS y con resolución 1280x720.

Este dataset está indicado para probar algoritmos en condiciones nocturnas, con una iluminación natural y un ligero refuerzo de las luces interiores más débiles del automóvil para

poder simular condiciones de iluminación de una avenida. La cámara se ubica de manera frontal al conductor.

Se ha decidido utilizar 10 vídeos al azar de cada conjunto de datos, bostezando y microsueños, para reducir el tamaño del dataset y poder analizar brevemente los resultados.

Bostezando



a) b)

c)

Figura 26. Ejemplos de rostros del Nitymed dataset en el conjunto de bostezos, con resultados exitosos.

En los vídeos del conjunto de bostezos se han encontrado resultados bastante satisfactorios, consiguiendo detectar la mayoría de los bostezos. Sin embargo, también ha habido detecciones erróneas, como en el caso de la imagen b) de la Figura 26.

Microsueños

Un microsueño es un episodio de sueño repentino en el que el cerebro desconecta del entorno, puede durar desde una fracción de segundo hasta un par de minutos.

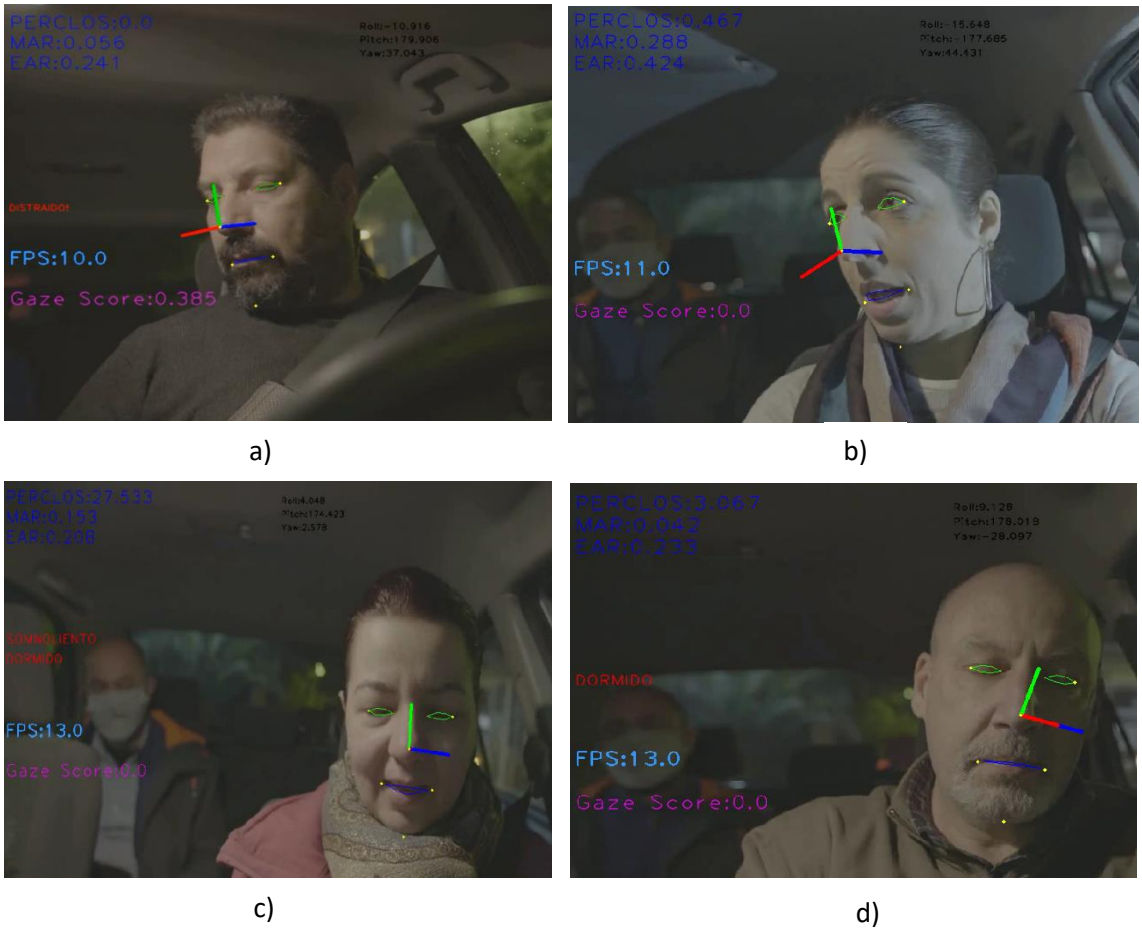


Figura 27. Ejemplos de rostros del Nitymed dataset en el conjunto de microsueños, con resultados exitosos.

En las imágenes de la Figura 27 se pueden observar casos de éxito en los que se ha monitorizado correctamente el rostro del conductor o conductora y se han detectado diferentes estados. En el caso de a) el conductor se encuentra distraído y la alarma es coherente con su comportamiento. En el caso de c) y d), los sujetos presentan microsueños que el algoritmo es capaz de detectar. En el c), además, aparece la alarma de somnolencia porque la conductora tiene una puntuación de PERCLOS por encima del 20 %, más concretamente, cerca del 27,5 %.

Figura 28. Ejemplos de rostros del Nitymed dataset en el conjunto de microsueños, con resultados erróneos.

Contrariamente, en las imágenes de la Figura 28, se presentan casos en los que ha habido fallos en la detección. En a) y c) el algoritmo no es capaz de seguir los puntos clave del sujeto y en b) el algoritmo ni tan solo consigue detectar el rostro del conductor.

En los vídeos de este dataset ha sido complicado encontrar detecciones de los episodios de microsueños debido principalmente a dos factores: la iluminación es deficiente, lo que dificulta la tarea de ubicación de los ojos, y los episodios son de muy corta duración, la mayoría alrededor de 1 segundo. Tal y como se han fijado los umbrales, considerando que el sujeto estará dormido cuando supere el tiempo EAR_time_thresh, que es igual a 3 segundos, no se consiguen detectar estos microsueños.

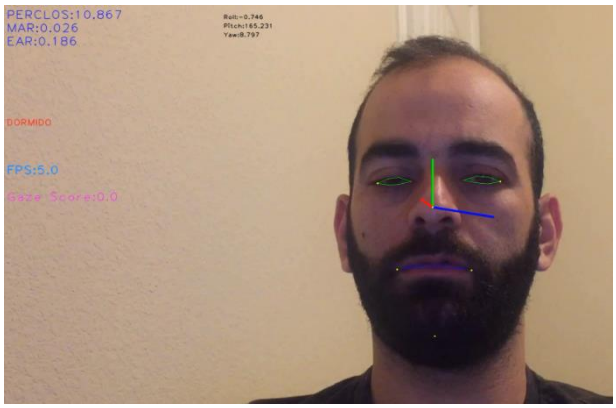
En algunos vídeos se produce además una oclusión parcial del rostro por el volante, como es el caso de la imagen b) Figura 28, lo que hace perder la correcta ubicación de los puntos clave del rostro.

7.1.3. UTA Real-Life Drowsiness dataset

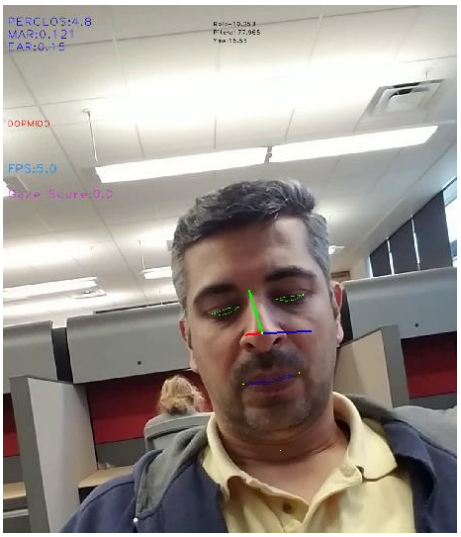
Es el dataset de datos de somnolencia de la vida real de la Universidad de Texas en Arlington [16]. Se creó para la tarea de detección de somnolencia en diferentes etapas y está enfocado sobre todo en aquellos casos sutiles en los que las microexpresiones son los factores determinantes de la somnolencia.

Consta de alrededor de 30 horas de vídeos RGB repartidas en 180 vídeos de 60 participantes distintos, de los cuales 51 eran hombres y 9 mujeres, de diferentes etnias y edades, algunos con gafas, etc. Para cada participante existe un vídeo de alerta, otro de baja vigilancia y otro de somnolencia. Estas tres clases están basadas en la escala KSS, en la que alerta se consideran los tres primeros estados, baja vigilancia son los niveles 6 y 7, y somnolencia los niveles 8 y 9.

El peso de este dataset es de casi 100 GB, por lo que solamente se han seleccionado al azar 5 vídeos, que duran aproximadamente 10 minutos cada uno, para analizar brevemente si el algoritmo es capaz de realizar detecciones satisfactorias.



a)

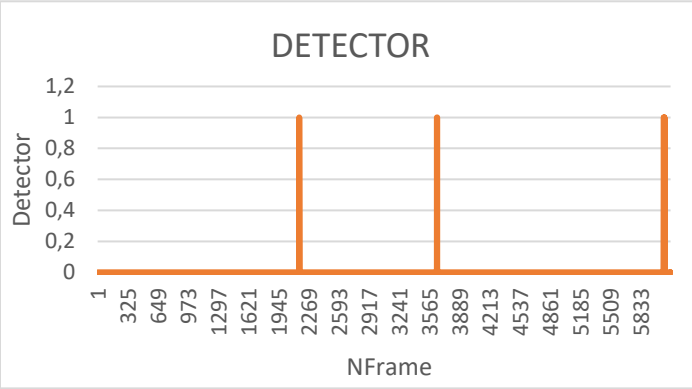


b)

Figura 29. Ejemplos de rostros del UTA dataset, con resultados exitosos.

Los vídeos analizados de este dataset han obtenido muy buenos resultados, posiblemente porque las condiciones de iluminación son óptimas y los sujetos presentan muy poco movimiento. En ambas imágenes presentadas en la Figura 29 los sujetos estaban sentados frente al ordenador.

La baja tasa de FPS en estos casos con el algoritmo se debe al tamaño de los frames, que se ha decidido no ajustar a 640x480 para no distorsionar los rostros.



En la imagen b) de la Figura 29, se ha realizado la detección del rostro mayoritariamente con el detector clasificado como 1, que es el de los gradientes orientados, HOG. Sin embargo, ha habido momentos en los que este no ha funcionado y sí lo han hecho las cascadas de Haar.

Figura 30. Gráfico de tipo de detector utilizado en cada frame en el vídeo b) de la Figura 29.

7.2. ENSAYOS CON RASPBERRY Y NOIR CAMERA

Este grupo de ensayos hace referencia a los que se han realizado con sujetos en físico, y se dividen en dos escenarios distintos: simulación y conducción real. Para estos ensayos se ha diseñado un formulario previo a su realización para conocer el contexto y circunstancias específicas de la persona que va a conducir.

Formulario:

- ¿Cuál es su edad?
- ¿Cuál es su género?
- Descripción de sus rasgos faciales, etnia.
- ¿Cuánto hace que tiene el carné de conducir?
- ¿Con qué frecuencia conduce?
- ¿Su estilo de vida es más sedentario o activo?
- ¿Cuántas horas ha trabajado hoy y cuántas ha descansado la noche anterior?
- ¿Qué hábitos de sueño tiene?
- ¿Tiene historial de accidentes previos?

La conversación entre el conductor y el ocupante del vehículo se restringe solamente para que el sujeto de estudio se autoevalúe en la escala KSS cada 10 minutos de conducción. De esta manera, se evita distraer o entretener al conductor.

7.2.1. Ensayos en simulación

El simulador que se ha propuesto utilizar en este trabajo es un juego de conducción para PC llamado City Car Driving. Desarrollado por Forward Development, es un juego centrado en ofrecer una experiencia realista de la conducción en entornos urbanos y en carreteras. Está diseñado para que los jugadores tengan la oportunidad de aprender, practicar y disfrutar de la experiencia de conducir automóviles en un entorno urbano virtual.

El juego simula de manera precisa el comportamiento de los vehículos, teniendo en cuenta factores como la aceleración, la frenada, la tracción, la inercia, etc. También se tienen en cuenta las reglas de tráfico. Por todo esto, es un juego que ofrece una experiencia inmersiva, que es justo lo que se busca en este caso para realizar pruebas en simulación. Una gran ventaja de la simulación es que los sujetos involucrados en el ensayo pueden encontrarse en cualquier situación respecto a la somnolencia sin que esto suponga un peligro para su seguridad.

Figura 31. Posicionamiento de la cámara en el PC para los ensayos en simulación.

Los ensayos se han realizado con un PC de sobremesa y un volante para simular un escenario parecido al del interior de un vehículo. La cámara se ha situado justo encima del monitor del PC apuntando al conductor, tal y como se muestra en la Figura 31.

Se han realizado 4 ensayos en esta modalidad, con una duración de 20 minutos cada uno. Los participantes han conducido en el juego en condiciones de iluminación media y alta, y predominantemente por escenarios de poblado, aunque también por autovía.

Figura 32. Ejemplos de la conducción en ensayos en formato simulación.

Los resultados de la simulación se aproximan a los hallados con algunos de los datasets anteriores, sobre todo con los vídeos en los que la iluminación era óptima y la posición de la cámara lo suficientemente frontal. Las detecciones son en general satisfactorias, con un porcentaje muy bajo de falsos positivos, que se atribuyen sobre todo al detector de las Cascadas de Haar. Se alcanzan porcentajes del 95 % de detecciones, con cerca del 92 % con el detector de HOG.

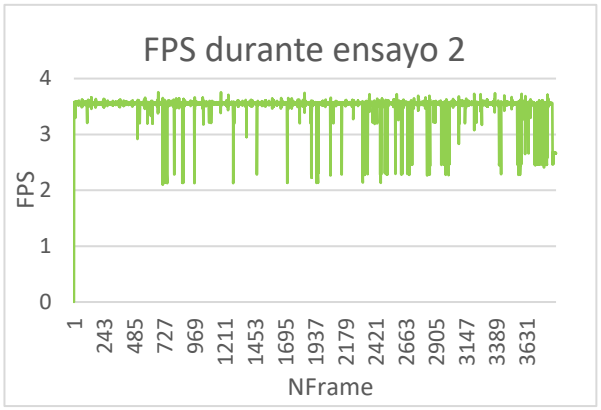
La rutina de inicialización cumple su función adaptando los umbrales del MAR y el EAR según el sujeto del ensayo. Se muestra la variabilidad en los 4 ensayos realizados en esta modalidad en la tabla que se muestra a continuación.

		Ensayo			
		Ensayo 1	**Ensayo 2**	**Ensayo 3**	**Ensayo 4**
Umbrales	**MAR_thresh**	0,421	0,355	0,348	0,580
	MAR_thresh	0,273	0,210	0,194	0,322

Tabla 3. Umbrales personalizados en los ensayos en simulación realizados.

Los sujetos de estos ensayos son dos hombres y dos mujeres, de entre 22 y 60 años, con experiencia al volante superior a 2 años. Dichos conductores se han autoevaluado, empezando por el ensayo 1 y acabando por el ensayo 4, en las siguientes posiciones de la escala KSS: somnoliento, algún esfuerzo por mantenerse despierto; ni alerta ni somnoliento; muy alerta y algunos síntomas de somnolencia. Todos los conductores han repetido su puntuación en la autoevaluación a los 10 minutos de iniciar la prueba.

Figura 33. Ejemplos de una conductora en los ensayos en simulación.



La tasa de FPS del ensayo referido en la Figura 33 se mantiene en la media de los 3,5 FPS durante los 4 ensayos. En los casos en que es más baja es cuando a los detectores les cuesta reconocer la cara en los frames.

Figura 34. Gráfica de los FPS durante el ensayo 2 de simulación.

7.2.2. Ensayos en condiciones reales

Los ensayos en condiciones reales son los más realistas por tratarse de situaciones de cotidianeidad en el ámbito de la conducción. Con estos ensayos se pretende probar el sistema ante diferentes circunstancias para valorar su desempeño, por lo que será más extenso y variado que el apartado anterior con los ensayos en condiciones de simulación. Se han llevado a cabo ensayos en diferentes sujetos, con diferentes accesorios, con variabilidad de condiciones de iluminación, etc.

El montaje en el vehículo es uno de los puntos críticos de este trabajo, ya que por las características de los detectores de rostros que se han utilizado, debe darse prioridad a situarlo de la manera más frontal posible al conductor. El sistema se ha instalado en un vehículo Opel Corsa 1.2T XHL Elegance. Tras probar diferentes opciones de colocación del sistema, como encima del salpicadero, se ha optado por situarlo en la visera parasol del asiento del conductor, procurando que la cámara apunte correctamente al conductor. En las imágenes siguientes de la Figura 35 se muestra el montaje.

Figura 35. Montaje en vehículo del sistema de detección de somnolencia.

De esta manera, la visera parasol no obstaculiza el campo de visión del conductor y a su vez permite que la cámara quede a la altura de los ojos del sujeto, lo que mejora considerablemente los resultados de la detección de rostros.

Para el caso de los ensayos en condiciones de iluminación bajas, como por la noche, se ha montado la lámpara de LEDs infrarrojos realizando las conexiones con una de las luces de cortesía alimentada a 12V de la zona delantera del vehículo, tal y como se muestra en la Figura 36.

Cabe destacar que finalmente, a esta lámpara se le ha eliminado la LDR para regular los LED en función de la iluminación ambiente, ya que tal y como se reporta en [4], existían problemas con farolas o elementos que proporcionan luz durante los ensayos nocturnos, lo que provocaba que no se encendiesen los LED. Para esquivar esta problemática, se ha decidido cortocircuitar la LDR y de esta manera, que los LED queden encendidos todo el tiempo en que estén alimentados.

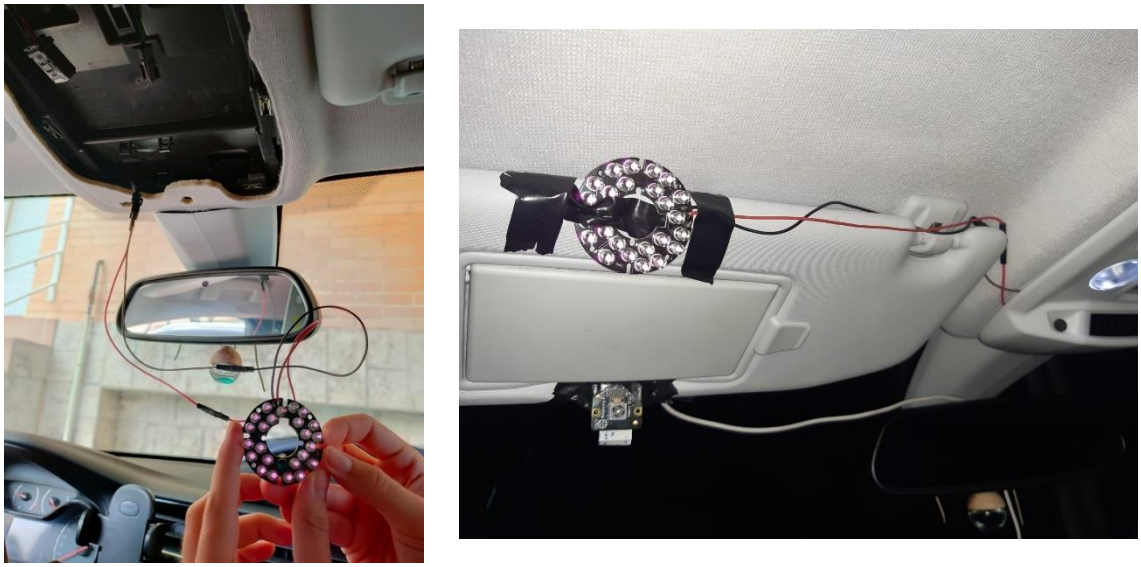


Figura 36. Montaje de la lámpara LED de infrarrojos en el vehículo.

Se ha contado con un total de 11 participantes, con entre una y dos sesiones para cada usuario de una duración de 20 minutos cada una. El género de los conductores de los ensayos es de un 64 % hombres y un 36 % mujeres. El rango de edades se encuentra comprendido entre 23 y 58 años. Cada participante tiene unas circunstancias concretas que se ha considerado pueden influir en un cuadro de somnolencia, y que se verán reflejadas en el formulario que han rellenado previamente a los ensayos.

Los ensayos se han realizado en diferentes escenarios de conducción y con diferentes itinerarios para darle más variabilidad en este aspecto, algunos se han realizado en poblado y otros en autovía. La conversación entre el participante y el acompañante queda restringida a solamente la respuesta a la estimación subjetiva del índice KSS, que se realiza cada 10 minutos. En este caso, al empezar el ensayo y justo a mitad del mismo.

Los primeros segundos del ensayo se utilizan para la rutina de inicialización del rostro del conductor, gracias a la cual se establecen los umbrales para los indicadores EAR y MAR.

ENSAYO 1

La participante es una mujer blanca de 23 años de etnia caucásica, con carnet de conducir de 8 meses de antigüedad. Conduce con una frecuencia prácticamente diaria y se define como una persona moderadamente activa. La noche anterior al ensayo, que se inició alrededor de las 18 h de la tarde, había dormido unas 5 horas. Sus hábitos de sueño son irregulares, durmiendo entre 6 y 7 horas diarias. No reporta ningún antecedente de accidente vial.

Tabla 4. Tabla recopilatoria de los datos del ensayo 1.

ENSAYO	Ensayo 1		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	6 (Algunos síntomas de somnolencia) en ambas ocasiones	
	Luminosidad	Media: nublado y lloviendo	
RESULTADOS	Porcentaje de detección facial	92,4%	
	Detector	HOG	96,13%
		Cascadas de Haar	3,87%
	Velocidad de detección	Máximo	3,77 fps
		Mínimo	2,13 fps
		Promedio	3,49 fps
	Umbrales personalizados	MAR	0,45
		EAR	0,23
	Puntuación PERCLOS	Promedio	19,94%
		Máximo	33,33%
	Alarmas activadas	Bostezo	0
		Somnoliento	7
		Dormido	2
		Distraído	10
		Distracción mirada	1
COMENTARIOS	Se han producido algunas alarmas de dormido y somnoliento que son coherentes con el nivel de somnolencia de la escala KSS en que se ha situado el sujeto.		

En este ensayo se ha alcanzado un 92 % de detección facial, de las cuales el 96,13 % se han realizado con el detector de HOG, sin tener que recurrir a las Cascadas de Haar para encontrar el rostro en la imagen. El promedio de velocidad de procesamiento de frames ha sido de 3,49 fps. Los umbrales personalizados mediante la rutina de inicialización son de 0,45 para el MAR y de 0,23 para el EAR, puntuaciones que se adaptan muy adecuadamente a la persona del ensayo. La puntuación PERCLOS que se ha alcanzado como máximo en un minuto ha sido de 33,33 %, lo que ha hecho saltar la alarma de somnoliento. Se ha sobrepasado el 20 % del PERCLOS en 7 de los 20 minutos que dura el ensayo. El promedio, sin embargo, ha sido de 19,94 %, lo que significa que en otros minutos la conductora se ha encontrado más atenta. Se han producido además 10 alarmas de distracción, algunas relacionadas con movimientos para mirar hacia los lados del vehículo, 1 alarma de distracción de la mirada y ningún bostezo.

Se han producido algunos falsos positivos, generalmente en momentos en los que la conductora tiene la cabeza muy girada hacia un lado y el detector no es capaz de reconocer el rostro. Sin embargo, los falsos positivos han sido escasos, por lo que se considera un ensayo con resultados bastante satisfactorios.

La conductora manifiesta que lleva lentillas y que eso le causa una sequedad excesiva de los ojos, lo que implica aumentar la cantidad de pestañeos y, por lo tanto, la puntuación del PERCLOS, que da lugar a la alarma de somnoliento cuando se supera el 20 %. Además, a pesar de ser un día nublado, en momentos en los que ha salido el sol, los reflejos de la conductora

han sido entornar los ojos, lo que puede aumentar también la puntuación del PERCLOS sin significar necesariamente somnolencia.

ENSAYO 2

La participante es una mujer blanca de 58 años, de etnia caucásica, con carnet de conducir con 37 años de antigüedad. La frecuencia de conducción es diaria, aunque de trayectos cortos y en poblado. Perfil de actividad física frecuente. El ensayo se inició sobre las 18:30 h de la tarde, habiendo descansado 8 horas la noche anterior. Sus hábitos de sueño son medianamente regulares, durmiendo alrededor de 8 horas diarias, aunque con episodios de desvelo en la madrugada. No reporta ningún antecedente de accidente vial.

Tabla 5. Tabla recopilatoria de los datos del ensayo 2.

ENSAYO	Ensayo 2		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	5 (ni alerta ni somnoliento) y 6 (algunos síntomas de somnolencia)	
	Luminosidad	Media: nublado	
RESULTADOS	Porcentaje de detección facial	96,36%	
	Detector	HOG	96,70%
		Cascadas de Haar	3,30%
	Velocidad de detección	Máximo	3,75 fps
		Mínimo	1,88 fps
		Promedio	3,49 fps
	Umbrales personalizados	MAR	0,32
		EAR	0,25
	Puntuación PERCLOS	Promedio	9,24%
		Máximo	28,57%
	Alarmas activadas	Bostezo	1
		Somnoliento	2
		Dormido	2
		Distraído	12
		Distracción mirada	0
COMENTARIOS	Se han producido alguna alarma de dormido y somnoliento, pero sobre todo destacan las distracciones. La detección de rostro ha sido precisa, sin falsos positivos.		

En este ensayo se ha alcanzado un 96,36 % de detección facial, por lo que los detectores han funcionado bastante bien. De estos, el 96,7 % se han realizado con el detector de HOG. No ha habido falsos positivos, por lo que los resultados en referencia a la detección del rostro son óptimos.

El promedio de velocidad de procesamiento de frames ha sido de 3,49 fps, igual que en el primer ensayo. Los umbrales personalizados mediante la rutina de inicialización son de 0,32

para el MAR y de 0,25 para el EAR, puntuaciones que también se adaptan muy adecuadamente a la persona del ensayo. La puntuación PERCLOS promedio ha sido de 9,24 % y la que se ha alcanzado como máximo en un minuto ha sido de 28,57 %, lo que ha hecho saltar la alarma de somnoliento. Se ha sobrepasado el 20 % del PERCLOS solamente en 2 de los 20 minutos que dura el ensayo. Se han producido además 12 alarmas de distracción, algunas de ellas relacionadas con movimientos para mirar hacia los lados del vehículo, 2 de dormido, ninguna distracción de la mirada y 1 bostezo.

ENSAYO 3

El participante es un hombre blanco de 58 años, de etnia caucásica, con carnet de conducir con 39 años de antigüedad. La frecuencia de conducción es diaria, con diferentes tipos de trayectos. Tiene un perfil de actividad física medio. El ensayo se inició sobre las 19:30 h de la tarde, habiendo descansado 8 horas la noche anterior. Sus hábitos de sueño son medianamente regulares, durmiendo alrededor de 8 horas diarias. No reporta ningún antecedente de accidente vial. El conductor presenta gafas de vista.

Tabla 6. Tabla recopilatoria de los datos del ensayo 3.

ENSAYO	Ensayo 3		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	5 (ni alerta ni somnoliento) en ambas ocasiones	
	Luminosidad	Alta: soleado	
RESULTADOS	Porcentaje de detección facial	81,85%	
	Detector	HOG	36,37%
		Cascadas de Haar	63,63%
	Velocidad de detección	Máximo	3,72 fps
		Mínimo	1,77 fps
		Promedio	2,64 fps
	Umbrales personalizados	MAR	0,37
		EAR	0,2
	Puntuación PERCLOS	Promedio	14,98%
		Máximo	35,24%
	Alarmas activadas	Bostezo	3
		Somnoliento	1
		Dormido	2
		Distraído	6
		Distracción mirada	1
COMENTARIOS	Se han producido pocas alarmas y algunas de ellas erróneas debido a la alta cantidad de falsos positivos.		

Se ha alcanzado un 81,85 % de detección facial, de las cuales solamente el 36,37 % se han realizado con el detector de HOG. Significa que la detección del rostro ha sido costosa y se ha tenido que recurrir a las Cascadas de Haar en la mayoría de las ocasiones, presentando una cantidad de falsos positivos elevada. Los resultados no son demasiado satisfactorios. Es posible que dichos resultados se deban a la distancia del conductor a la cámara, empeorando los resultados de detección a medida que el sujeto se encuentra más alejado.

El promedio de velocidad de procesamiento ha sido de 2,64 fps, más bajo que los anteriores, debido a que la dificultad de detectar un rostro conlleva más tiempo de procesamiento. Los umbrales personalizados son de 0,37 para el MAR y de 0,2 para el EAR.

ENSAYO 4

La participante es una mujer blanca de 24 años de etnia caucásica, con rasgos nórdicos. Tiene el carnet de conducir con 1 año de antigüedad. Conduce con una frecuencia prácticamente diaria y se define como una persona más sedentaria que activa. La noche anterior al ensayo, que se inició alrededor de las 20 h de la tarde, había dormido alrededor de 5 horas y ese día había trabajado 8 horas. Sus hábitos de sueño son regulares, durmiendo entre 7 y 8 horas diarias. No reporta ningún antecedente de accidente vial.

Tabla 7. Tabla recopilatoria de los datos del ensayo 4.

ENSAYO	Ensayo 4		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	6 (algunos síntomas de somnolencia) y 7 (somnoliento, sin esfuerzo por mantenerse despierto)	
	Luminosidad	Alta	
RESULTADOS	Porcentaje de detección facial	93,75%	
	Detector	HOG	98,13%
		Cascadas de Haar	1,87%
	Velocidad de detección	Máximo	3,79 fps
		Mínimo	2,12 fps
		Promedio	3,5 fps
	Umbrales personalizados	MAR	0,49
		EAR	0,17
	Puntuación PERCLOS	Promedio	12,53%
		Máximo	28,10%
	Alarmas activadas	Bostezo	2
		Somnoliento	1
		Dormido	2
		Distraído	3
		Distracción mirada	0
COMENTARIOS	Se han producido pocas alarmas a pesar de que la conductora se ha autoevaluado con cierta somnolencia		

El porcentaje de detección es muy positivo, sobre todo porque se encuentra acompañado de una detección del rostro con HOG del casi 100 %. Además, no hay falsos positivos. Los umbrales de los indicadores EAR y MAR se ajustan al sujeto del ensayo adecuadamente. El PERCLOS es relativamente bajo para lo que la conductora ha considerado que es su estado de somnolencia. Como ya se ha mencionado anteriormente, es posible que la autopercepción se vea alterada y no refleje realmente el estado del sujeto, por lo que seguramente la conductora se encontraba más bien en el estado número 5 de la escala KSS, ni alerta ni somnolienta.

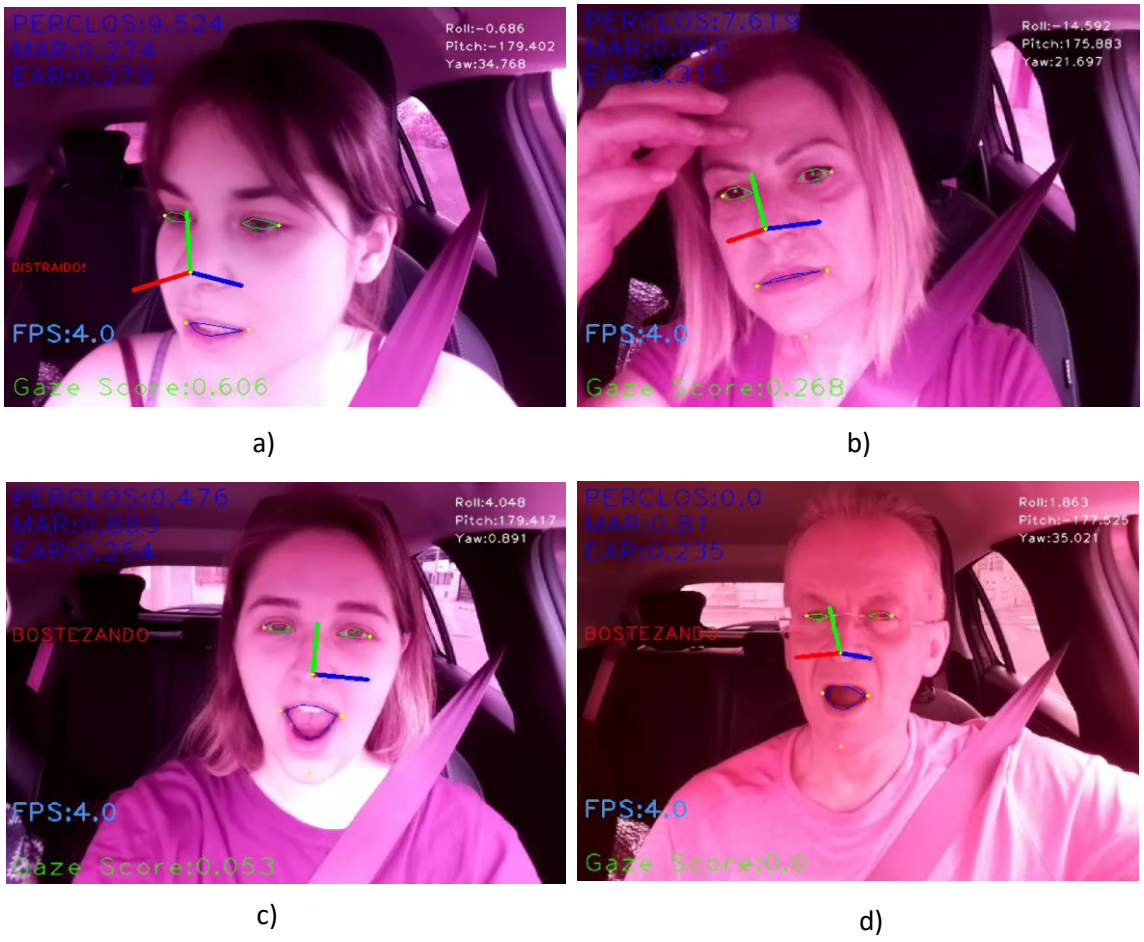


Figura 37. Ejemplos de frames de algunos sujetos de los ensayos en condiciones reales.

ENSAYO 5

El participante es un hombre blanco de 22 años de etnia caucásica, con una antigüedad conduciendo de 4 años. Conduce todos los días para desplazarse al trabajo y a la universidad. Su estilo de vida es activo, realizando deporte entre 4 y 5 días a la semana. La noche anterior al ensayo ha dormido 7 horas y durante ese día ha trabajado 5 horas. Justo antes de realizar el ensayo, que ha empezado sobre las 17 h, ha dormido una siesta de poco más de media hora. No reporta ningún antecedente de accidente vial.

Tabla 8. Tabla recopilatoria de los datos del ensayo 5.

ENSAYO	Ensayo 5		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	4 (más bien alerta) y 5 (ni alerta ni somnoliento)	
	Luminosidad	Alta: soleado	
RESULTADOS	Porcentaje de detección facial	98,22%	
	Detector	HOG	94,70%
		Cascadas de Haar	5,30%
	Velocidad de detección	Máximo	3,76 fps
		Mínimo	2,00 fps
		Promedio	3,47 fps
	Umbrales personalizados	MAR	0,35
		EAR	0,28
	Puntuación PERCLOS	Promedio	8,39%
		Máximo	11,43%
	Alarmas activadas	Bostezo	0
		Somnoliento	1
		Dormido	0
		Distraído	3
		Distracción mirada	1
COMENTARIOS	A penas se han producido alertas porque el conductor estaba en un estado de alerta promedio, concentrado en la conducción		

El porcentaje de detección facial resulta ser muy exitoso, superando el 98 %, del cual cerca de un 95 % se han producido con HOG. Además, no se han producido falsos positivos. Respecto a los umbrales del EAR y MAR, se han ajustado al sujeto del ensayo adecuadamente; son de 0,35 para el MAR y de 0,28 para el EAR. El PERCLOS está acorde con lo que el conductor percibe de sí mismo durante el ensayo, teniendo un promedio de 11,43 % del tiempo los ojos cerrados.

En cuanto a la velocidad de procesamiento de frames, ha tenido un promedio de 3,47 fps, parecido al resto a excepción del Ensayo 3.

ENSAYO 6

El participante es un hombre blanco de 23 años de etnia caucásica. Tiene el carnet de conducir con 3,5 años de antigüedad. Conduce con una frecuencia diaria, mucho por entornos monótonos como la autovía para ir a trabajar, y se define como una persona más sedentaria que activa. La noche anterior al ensayo, había dormido alrededor de 5 horas y ese día había trabajado 8 horas. Sus hábitos de sueño son regulares, durmiendo entre 5 y 6 horas diarias, por lo que padece de cansancio acumulado. No reporta ningún antecedente de accidente vial.

Tabla 9. Tabla recopilatoria de los datos del ensayo 6.

ENSAYO	Ensayo 6		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	8 (somnoliento, algún esfuerzo por mantenerse despierto) en ambas ocasiones	
	Luminosidad	Alta: soleado	
RESULTADOS	Porcentaje de detección facial	98,20%	
	Detector	HOG	95,12%
		Cascadas de Haar	4,88%
	Velocidad de detección	Máximo	3,77 fps
		Mínimo	2,11 fps
		Promedio	3,49 fps
	Umbrales personalizados	MAR	0,35
		EAR	0,18
	Puntuación PERCLOS	Promedio	14,19%
		Máximo	29,05%
	Alarmas activadas	Bostezo	3
		Somnoliento	4
		Dormido	2
		Distraído	3
		Distracción mirada	1
COMENTARIOS	Se han producido algunas alarmas de dormido y somnoliento debido al estado de somnolencia del conductor. También se han producido 3 bostezos que se han detectado correctamente. No ha presentado mucha distracción, pero sí síntomas de estar cansado.		

El participante en este ensayo se autoevaluó en un nivel de somnolencia alto, con una puntuación de 8 sobre 9 según la KSS. El PERCLOS se encuentra acorde con lo que el conductor percibe de sí mismo durante el ensayo, teniendo un promedio de 14,19 % del tiempo los ojos cerrados, y un máximo de 29,05 %. Se han producido por tanto todas las alarmas, mayoritariamente la de somnoliento, seguida de las distracciones y los bostezos.

El porcentaje de detección facial resulta ser también muy exitoso, como en casos anteriores, superando el 98 %. De dichas detecciones, más del 95 % son con el detector de HOG, sin presencia de falsos positivos. Respecto a los umbrales del EAR y MAR, se han ajustado al sujeto del ensayo adecuadamente; son de 0,35 para el MAR y de 0,18 para el EAR.

La velocidad de procesamiento de frames ha sido regular como en la mayoría del resto de ensayos, teniendo un promedio de 3,49 fps.

ENSAYO 7

El participante es un hombre blanco de 53 años, de etnia caucásica, que tiene una experiencia conduciendo de 35 años. Es conductor frecuente y tiene un estilo de vida activo, realizando deporte asiduamente. El día del ensayo trabajó 7,5 horas y durmió unas 7 horas. Tiene una rutina de sueño estable de entre 7 y 8 horas de descanso. No reporta ningún antecedente de accidente vial.

Tabla 10. Tabla recopilatoria de los datos del ensayo 7.

ENSAYO	Ensayo 7		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	3 (alerta) en ambas ocasiones	
	Luminosidad	Alta: soleado	
RESULTADOS	Porcentaje de detección facial	98,33%	
	Detector	HOG	98,70%
		Cascadas de Haar	1,30%
	Velocidad de detección	Máximo	3,76 fps
		Mínimo	1,99 fps
		Promedio	3,52 fps
	Umbrales personalizados	MAR	0,33
		EAR	0,22
	Puntuación PERCLOS	Promedio	11,10%
		Máximo	19,05%
	Alarmas activadas	Bostezo	0
		Somnoliento	0
		Dormido	0
		Distraído	3
		Distracción mirada	2
COMENTARIOS	A penas se han producido alarmas, solamente de distracción al mirar durante determinado tiempo hacia los lados.		

En este ensayo el conductor se auto percibe como alerta, lo que resulta estar acorde con las alarmas producidas durante la conducción. Solamente se han producido algunas distracciones y una distracción de la mirada. El promedio del PERCLOS se encuentra sobre el 11 %, que es una puntuación promedio que encaja con su estado.

El porcentaje de detección facial resulta como en casos anteriores, superando el 98 %, de los que más del 98 % son con el detector de HOG, sin presencia de falsos positivos. Los umbrales calculados para el EAR y el MAR son 0,22 y 0,33 respectivamente, puntuaciones que encajan en un rostro como el del conductor.

La velocidad promedio del ensayo se ha mantenido como el resto, hallándose en 3,52 fps de media.

ENSAYO 8

El participante es un hombre blanco de 57 años, de etnia caucásica. Tiene el carnet de conducir desde hace casi 40 años. Es conductor diario y su estilo de vida es activo, realizando trabajos físicos con frecuencia. El día del ensayo trabajó 5 horas y durmió unas 6 horas. Padece de insomnio alternativo. No reporta ningún antecedente de accidente vial.

Tabla 11. Tabla recopilatoria de los datos del ensayo 8.

ENSAYO	Ensayo 8		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	3 (alerta) y 4 (más bien alerta)	
	Luminosidad	Alta: soleado	
RESULTADOS	Porcentaje de detección facial	98,43%	
	Detector	HOG	92,30%
		Cascadas de Haar	7,70%
	Velocidad de detección	Máximo	3,77 fps
		Mínimo	2,11 fps
		Promedio	3,44 fps
	Umbrales personalizados	MAR	0,30
		EAR	0,19
	Puntuación PERCLOS	Promedio	8,19%
		Máximo	17,62%
	Alarmas activadas	Bostezo	0
		Somnoliento	0
		Dormido	0
		Distraído	9
		Distracción mirada	0
COMENTARIOS	El conductor se ha autoevaluado como alerta y los resultados coinciden. Tan solo se han producido alertas de distracción y algunas posiblemente debido a la observación en cruces.		

Parecido al Ensayo 7, el conductor se percibe como alerta. El PERCLOS se encuentra acorde con lo que el conductor percibe de sí mismo, teniendo un promedio de 8,19 % del tiempo los ojos cerrados, y un máximo de 17,62 %. Solamente se han producido alarmas de distracción, aunque han sido 7.

El porcentaje de detección facial también resulta por encima del 98 %, atribuyendo más de un 92 % al detector de HOG. Respecto a los umbrales del EAR y MAR, se han ajustado al sujeto del ensayo adecuadamente, siendo de 0,3 para el MAR y de 0,19 para el EAR.

La velocidad de procesamiento de frames ha sido regular como en la mayoría del resto de ensayos, presentando un promedio de 3,44 fps.

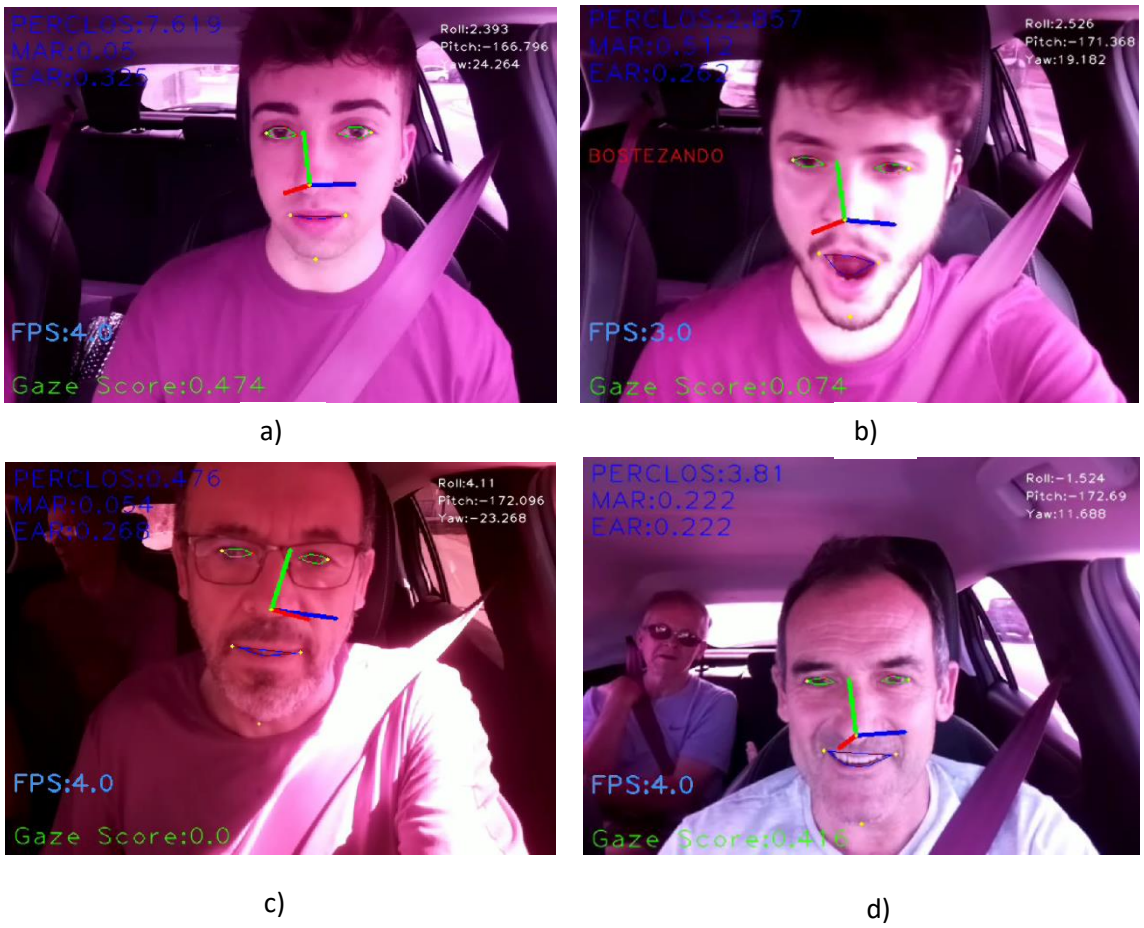


Figura 38. Ejemplos de frames de algunos sujetos de los ensayos en condiciones reales.

En la Figura 38 se observan ejemplos de la monitorización del rostro con el algoritmo diseñado. En todas se puede apreciar cómo se ubican correctamente los ojos y la boca de los conductores. En el caso de la imagen b), se produce la alarma de bostezo cuando el sujeto se encuentra bostezando, por lo que el algoritmo lo detecta correctamente.

En la mayoría de los casos el desempeño del algoritmo ha sido bueno, garantizando una monitorización del rostro realista la mayor parte del tiempo, adaptándose a los diferentes sujetos sometidos a los ensayos.

ENSAYO 9

El participante es una mujer blanca de 55 años, de etnia caucásica. Posee el carnet de conducir desde hace 37 años. Es una conductora habitual conduciendo diariamente y tiene un estilo de vida activo. El día del ensayo no trabajó y durmió unas 7 horas. Sigue una rutina de sueño estable de entre 7 y 8 horas de descanso por noche. No reporta ningún antecedente de accidente vial.

Tabla 12. Tabla recopilatoria de los datos del ensayo 9.

ENSAYO	Ensayo 9		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	5 (ni alerta ni somnoliento) en ambas ocasiones	
	Luminosidad	Alta: soleado	
RESULTADOS	Porcentaje de detección facial	95,56%	
	Detector	HOG	84,35%
		Cascadas de Haar	15,65%
	Velocidad de detección	Máximo	3,79 fps
		Mínimo	1,88 fps
		Promedio	3,33 fps
	Umbrales personalizados	MAR	0,41
		EAR	0,22
	Puntuación PERCLOS	Promedio	12,76%
		Máximo	19,76%
	Alarmas activadas	Bostezo	3
		Somnoliento	0
		Dormido	1
		Distraído	4
		Distracción mirada	3
COMENTARIOS	Se han producido algunas alarmas, algunas erróneas como algún bostezo, que se ha confundido con una sonrisa		

Se ha alcanzado un 95,56 % de detección facial, de los cuales, el 84,35 % se han realizado con el detector de HOG, resultando menos efectivo que en algunos de los ensayos anteriores. También se han presentado algunos falsos positivos. La puntuación PERCLOS promedio ha sido de 12,76 % y la que se ha alcanzado como máximo en un minuto ha sido de 19,76. Se han producido varias alarmas, destacando 4 en distracción y 3 en bostezos.

El promedio de velocidad de procesamiento de frames ha sido de 3,33 fps, disminuyendo un poco respecto a los anteriores ensayos, seguramente debido a los falsos positivos. Los umbrales personalizados mediante la rutina de inicialización son de 0,31 para el MAR y de 0,22 para el EAR, puntuaciones que también se adaptan muy adecuadamente a la persona del ensayo.

ENSAYO 10

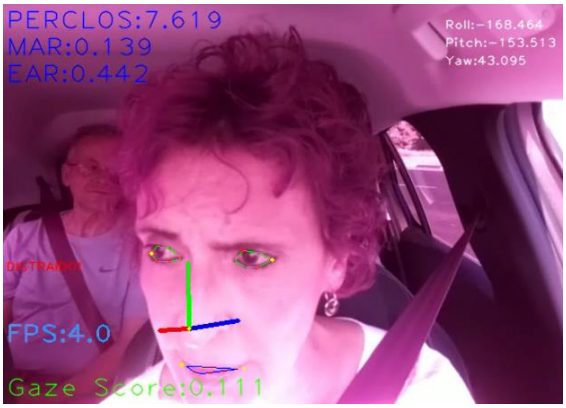
El participante es un hombre blanco de 57 años, de etnia caucásica. Tiene el carnet de conducir desde hace 38 años. Es conductor diario de distancias largas, recorriendo un mínimo de 100 km al día. Su estilo de vida es activo y el día del ensayo trabajó 10 horas y durmió unas 6 horas. Sigue una rutina de sueño inestable, acostándose tarde y durmiendo pocas horas y de manera intermitente. Reporta dos accidentes de tráfico, aunque sin culpa propia y sin relación con la fatiga al volante.

Tabla 13. Tabla recopilatoria de los datos del ensayo 10.

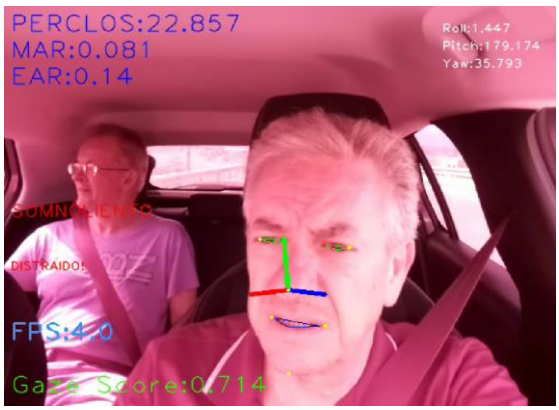
ENSAYO	Ensayo 10		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	6 (algunos síntomas de somnolencia) en ambas ocasiones	
	Luminosidad	Alta: soleado	
RESULTADOS	Porcentaje de detección facial	95,07%	
	Detector	HOG	95,84%
		Cascadas de Haar	4,16%
	Velocidad de detección	Máximo	3,79 fps
		Mínimo	2 fps
		Promedio	3,46 fps
	Umbrales personalizados	MAR	0,30
		EAR	0,20
	Puntuación PERCLOS	Promedio	29,30%
		Máximo	50,00%
	Alarmas activadas	Bostezo	0
		Somnoliento	8
		Dormido	5
		Distraído	6
		Distracción mirada	0
COMENTARIOS	Se han producido bastantes alarmas, destacando las de somnoliento, seguidas de distraído y dormido.		

En la línea de los ensayos anteriores, se ha conseguido en este una alta tasa detecciones, alcanzando el 95,07 % de detección facial de los cuales el 95,84 % son con el detector de HOG. Respecto al PERCLOS, la puntuación máxima es muy alta, del 50 %, y también lo es la puntuación promedio, del 29,3 %, concluyendo que el sujeto se encuentra en un estado de fatiga considerable durante este ensayo. Sin embargo, en la encuesta subjetiva éste solo se ha percibido con algunos síntomas de somnolencia. Repasando sus respuestas al formulario personal, es una persona cuyos hábitos de sueño son irregulares, lo que podría provocar que se encontrase fatigado la mayor parte del tiempo.

El promedio de velocidad de procesamiento de frames ha sido de 3,46 fps. Los umbrales personalizados mediante la rutina de inicialización son coherentes con su fisionomía, siendo de 0,302 para el MAR y de 0,204 para el EAR.



a)



b)

c)

Figura 39. Ejemplos de frames de algunos sujetos de los ensayos en condiciones reales.

En la Figura 39 se encuentran más ejemplos de detecciones satisfactorias durante los ensayos. En la imagen a) la conductora se encuentra distraída mirando hacia un lado, lo que hace aparecer la alarma de distracción.

En la imagen b), el conductor aparece distraído y además se produce una alarma de somnolencia debido a que el PERCLOS es superior al 20 %.

En la imagen c) el conductor se encuentra concentrado en la carretera y no muestra síntomas de somnolencia que conduzcan a la activación de una alarma.

ENSAYO 11

El sujeto del ensayo es el mismo que en el ensayo 6. Lo que ha cambiado en este ensayo es que se ha realizado de noche, sobre las 23:30 h.

Tabla 14. Tabla recopilatoria de los datos del ensayo 11.

ENSAYO	Ensayo 11		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	8 (somnoliento, algún esfuerzo por mantenerse despierto) en ambas ocasiones	
	Luminosidad	Muy baja: de noche	
RESULTADOS	Porcentaje de detección facial	96,14%	
	Detector	HOG	99,67%
		Cascadas de Haar	0,33%
	Velocidad de detección	Máximo	3,76 fps
		Mínimo	2,28 fps
		Promedio	3,54 fps
	Umbrales personalizados	MAR	0,31
		EAR	0,21
	Puntuación PERCLOS	Promedio	13,61%
		Máximo	31,90%
	Alarmas activadas	Bostezo	5
		Somnoliento	3
		Dormido	3
		Distraído	9
		Distracción mirada	0
COMENTARIOS	Se han producido todas las alarmas menos la de distracción de la mirada. El conductor presenta un cuadro de somnolencia medio.		

La particularidad de este ensayo es que es en condiciones de iluminación muy bajas, de noche. La lámpara de LEDs infrarroja ha estado activa durante todo el trayecto debido a la eliminación intencional de la LDR.

Ha sido un ensayo muy exitoso alcanzándose una detección por encima del 96 %, del cual prácticamente el 100 % ha sido con el detector de HOG. Se ha conseguido monitorizar correctamente al conductor, produciéndose distintas alarmas que se corresponden con un cuadro de somnolencia, aunque quizás no tan excesivo como el propio conductor se ha autoevaluado en la escala KSS. Ha presentado un PERCLOS promedio de 13,61 % y un máximo de 31,90 %.

La velocidad de procesamiento promedio es de 3,54 fps, lo que significa que se ha mantenido como en los ensayos con condiciones de iluminación altas.

ENSAYO 12

La conductora de este ensayo es la misma que en el ensayo 4. Sin embargo, en este caso, el ensayo se ha realizado por la noche, con condiciones de iluminación bajas.

Tabla 15. Tabla recopilatoria de los datos del ensayo 12.

ENSAYO	Ensayo 12		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	7 (somnoliento, sin esfuerzo por mantenerse despierto) y 6 (algunos síntomas de somnolencia)	
	Luminosidad	Muy baja: de noche	
RESULTADOS	Porcentaje de detección facial	96,55%	
	Detector	HOG	94,79%
		Cascadas de Haar	5,21%
	Velocidad de detección	Máximo	3,77 fps
		Mínimo	2,13 fps
		Promedio	3,48 fps
	Umbrales personalizados	MAR	0,47
		EAR	0,17
	Puntuación PERCLOS	Promedio	9,40%
		Máximo	18,57%
	Alarmas activadas	Bostezo	2
		Somnoliento	0
		Dormido	2
		Distraído	12
		Distracción mirada	0
COMENTARIOS	Se han producido sobre todo alarmas de distracción. La conductora se encuentra relativamente alerta.		

Este ensayo ha sido realizado de noche, con condiciones de iluminación muy bajas. En la tabla recopilatoria anterior se puede observar como el algoritmo ha tenido un buen desempeño, alcanzado una detección del rostro del 96,55 %, con el 94,79 % usando el detector de HOG.

En su mayoría, se han producido alarmas de distracción de la conductora. También ha presentado algún bostezo. El PERCLOS promedio ha sido de 9,4 % y el máximo de 18,57 %, no indicando demasiada somnolencia con este parámetro. Sin embargo, la conductora sí se ha autoevaluado con cierta somnolencia, lo que induce a pensar que cuando es de noche hay una tendencia a pensar en que se está somnoliento.

La velocidad de procesamiento promedio es de 3,48 fps.

ENSAYO 13

El conductor es un hombre blanco de 36 años, de etnia caucásica. La antigüedad de su carnet de conducir es de 16 años. Conduce semanalmente. Su estilo de vida es activo y el día del ensayo no trabajó y durmió unas 6 o 7 horas. Sigue una rutina de sueño estable de unas 6 o 7 horas por noche, aunque a veces presenta interrupciones del sueño. No reporta accidentes de tráfico.

Tabla 16. Tabla recopilatoria de los datos del ensayo 13.

ENSAYO	Ensayo 13		
CONDICIONES	Nivel de somnolencia (valorado por el sujeto con el índice KSS)	4 (más bien alerta) en ambas ocasiones	
	Luminosidad	Alta: soleado	
RESULTADOS	Porcentaje de detección facial	99,56%	
	Detector	HOG	98,57%
		Cascadas de Haar	1,43%
	Velocidad de detección	Máximo	3,77 fps
		Mínimo	2,23 fps
		Promedio	3,53 fps
	Umbrales personalizados	MAR	0,31
		EAR	0,28
	Puntuación PERCLOS	Promedio	7,76%
		Máximo	18,57%
	Alarmas activadas	Bostezo	1
		Somnoliente	0
		Dormido	0
		Distraído	2
		Distracción mirada	0
COMENTARIOS	El conductor ha permanecido en un estado de alerta, detectando solamente un bostezo ocasional y dos distracciones.		

En este ensayo se ha conseguido una alta tasa detecciones, casi del 100 % y casi únicamente con el detector de HOG. Respecto al PERCLOS, la puntuación máxima es de 18,57 % y la promedio de 7,76 %, presentando por tanto el conductor un cuadro de no somnolencia, hallándose en estado de alerta. Su autoevaluación parece corresponderse con los resultados obtenidos.

El promedio de velocidad de procesamiento de frames ha sido de 3,53 fps. Los umbrales personalizados mediante la rutina de inicialización son coherentes con su fisionomía, siendo de 0,310 para el MAR y de 0,278 para el EAR.

Figura 40. Ejemplos de frames de algunos sujetos de los ensayos en condiciones reales y de noche.

En la Figura 40 se presentan ejemplos de los ensayos realizados de noche. La lámpara LED está situada frente al conductor y esa circunstancia es la que provoca el resplandor blanco que se aprecia en el rostro de los conductores. Sin embargo, es también dicho resplandor el que seguramente permite detectar mejor los puntos faciales clave porque genera más contraste entre el color de piel y los ojos y la boca de las personas.

En la imagen a) se puede apreciar como el algoritmo es capaz de detectar el rostro a pesar de estar girado hacia un lado más de 45 grados. De la misma manera e indicando distracción, en la imagen c) sucede lo mismo.

En la imagen b) se produce un bostezo que el algoritmo es capaz de detectar y reflejar con la alarma de bostezo.

Se presentan a continuación imágenes en las que ha habido errores en el algoritmo, aunque estos han sido poco frecuentes.

a) b)

Figura 41. Ejemplos de detecciones o interpretaciones erróneas del algoritmo.

Durante los ensayos, en ocasiones se han presentado falsos positivos o errores en la monitorización del rostro, tal y como se ha ido comentando brevemente en los ensayos. En la Figura 41 se presentan casos de dichos errores. En el caso de la imagen a), el algoritmo ha encontrado una cara falsa en lugar de la del conductor, produciéndose además una alarma de distracción que es errónea. En la imagen b), la sonrisa de la conductora ha sido confundida con un bostezo.

8. EVALUACIÓN DEL SISTEMA

En este apartado se realiza un análisis de los resultados de los indicadores utilizados en el algoritmo a lo largo de los diferentes tipos de pruebas y ensayos, mayoritariamente en los ensayos en condiciones reales.

Como se ha visto a lo largo de todos los ensayos, los porcentajes de detección del rostro son generalmente positivas, arrojando datos por encima del 90 % en la gran mayoría de ocasiones. El detector de HOG resulta mucho más efectivo, tal y como se presuponía desde el principio de este trabajo. La decisión de incluir las Cascadas de Haar como detector secundario era intentar cubrir momentos en los que el detector de HOG no era suficiente, sin embargo, en la mayoría de los casos esto solo ha aportado falsos positivos.

Cabe destacar que el desempeño del algoritmo mejora cuando se circula por entornos con pocas distracciones y mayormente rectos, en lugar de por poblado, debido a que en los primeros se encuentran muchos cruces y curvas que provocan movimientos rápidos y severos de los conductores. Durante esos segundos, en muchas ocasiones al algoritmo se le hace complicado detectar el rostro debido a su ángulo de giro. Sin embargo, el algoritmo sí responde a cierto rango de giro de la cabeza y es capaz de cuantificar el ángulo y el tiempo para relacionarlo con distracciones, lo que puede resultar interesante.

Los ensayos con condiciones de luminosidad bajas, por la noche, han sido un éxito debido a la lámpara de LED infrarrojos que eliminando la LDR, permiten capturar imágenes del rostro del conductor de manera muy efectiva durante todo el trayecto.

Se ha considerado muy relevante en este proyecto la variación de rasgos faciales entre sujetos, por lo que se han calculado umbrales personalizados para cada uno de ellos con la rutina de inicialización previa a cada ensayo. Esta incorporación ha vuelto el algoritmo más robusto y versátil. La recopilación de estos resultados se presenta en la Tabla 17.

Tabla 17. Umbrales personalizados obtenidos en los ensayos realizados en condiciones reales.

		Umbrales	
		MAR_thresh	**EAR_thresh**
Ensayo	**Ensayo 1**	0,45	0,23
	Ensayo 2	0,32	0,25
	Ensayo 3	0,37	0,20
	Ensayo 4	0,49	0,17
	Ensayo 5	0,35	0,28
	Ensayo 6	0,35	0,18
	Ensayo 7	0,33	0,22
	Ensayo 8	0,30	0,19
	Ensayo 9	0,41	0,22
	Ensayo 10	0,30	0,20
	Ensayo 11	0,31	0,21
	Ensayo 12	0,47	0,17
	Ensayo 13	0,31	0,28

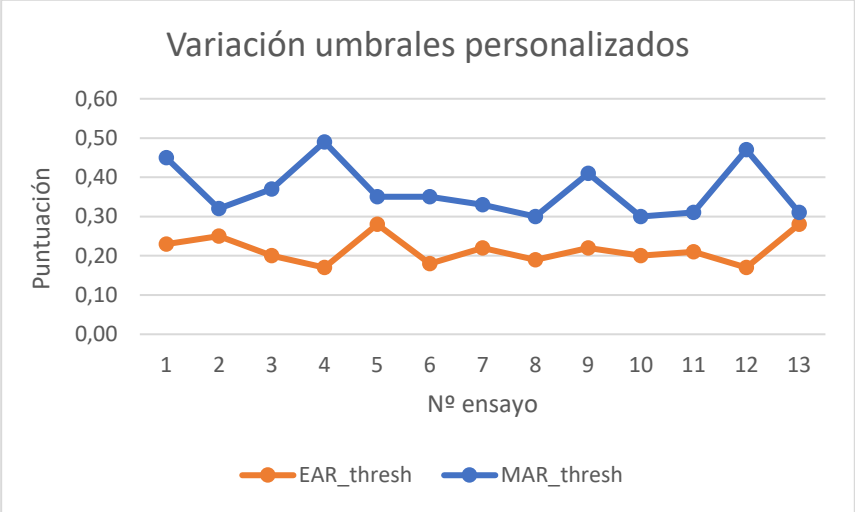


Figura 42. Gráfica de la variación de los umbrales personalizados en los ensayos en condiciones reales.

Indicador MAR

La personalización del umbral del indicador MAR ha permitido ajustar la referencia de apertura de la boca y, en consecuencia, la aparición de las alarmas de bostezos, a cada sujeto. En la Tabla 17 se pueden observar los diferentes resultados del MAR_thresh que se han producido a lo largo de los ensayos.

Se muestra un ejemplo en la Figura 43 en el que se representa conjuntamente el indicador MAR, el umbral MAR_thresh y las alarmas de bostezo (Yawn). En tres ocasiones se ha producido la alarma de bostezo debido a que la puntuación del MAR ha superado a su umbral durante el tiempo considerado un bostezo.

En todos los ensayos se han recogido los datos que permiten generar estas gráficas, cuya función es facilitar el análisis visual de los resultados. En la Figura 43 se detecta con claridad cuándo se producen los sucesos de interés, que son los bostezos.

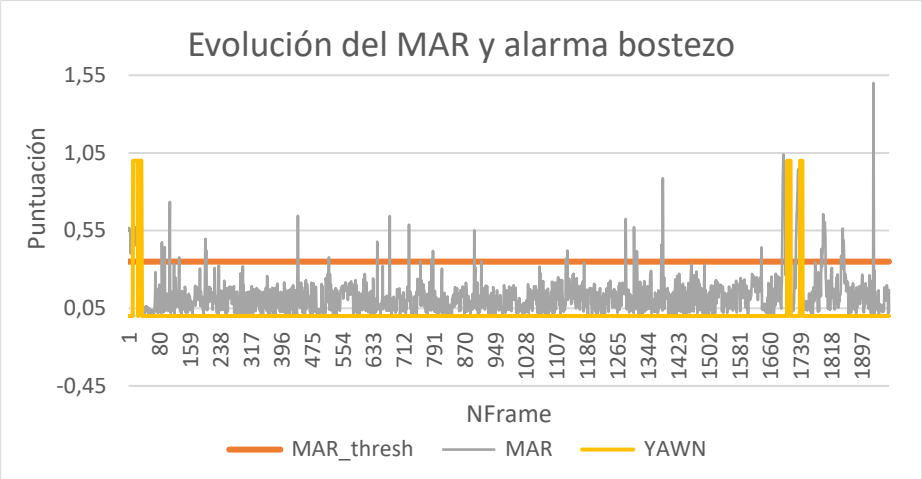


Figura 43. Evolución del MAR en el ensayo 6 en condiciones reales.

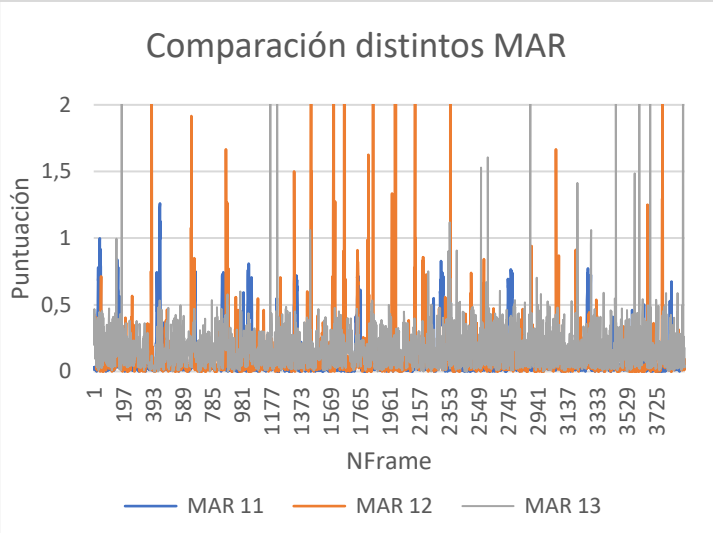


Figura 44. Comparación de las puntuaciones del MAR en los ensayos 11, 12 y 13 en condiciones reales.

Comparando los resultados en distintos ensayos del indicador MAR en la Figura 44, se observa que cada sujeto tiene un comportamiento diferente. El ensayo 12 tiene más momentos en los que la boca está abierta que en los ensayo 11 y 13. Sin embargo, es posible que no todos esos episodios se correspondan con bostezos, porque estos están también definidos por un tiempo mínimo.

Indicador EAR

Por su parte, la personalización del umbral EAR ha sido decisiva para adecuar la referencia en la apertura de los ojos a cada sujeto. Tal y como se puede apreciar en la Tabla 17, se ha producido cierta variabilidad en los resultados de los umbrales personalizados obtenidos. Además, incluso en el mismo sujeto se han podido obtener diferentes puntuaciones, seguramente debido a la combinación del momento del ensayo y el estado del sujeto en ese momento.

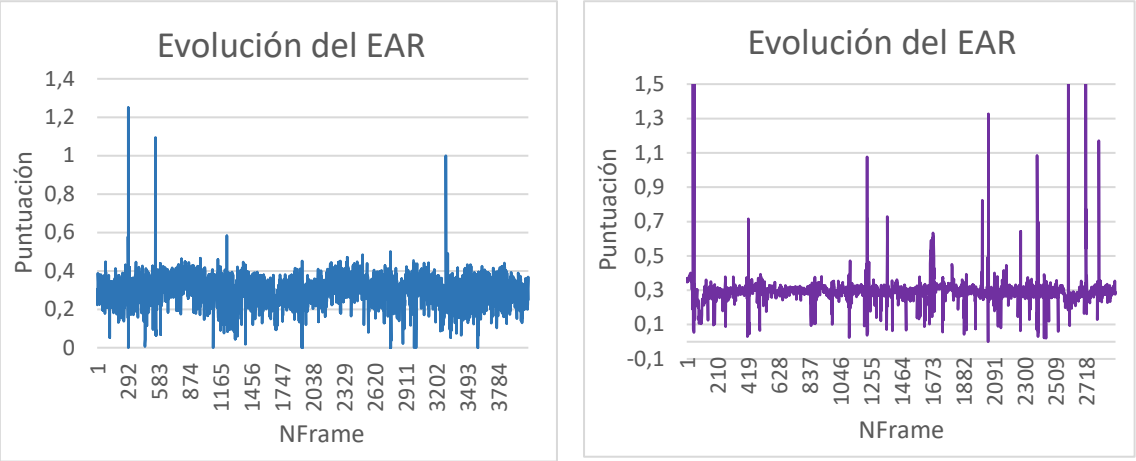
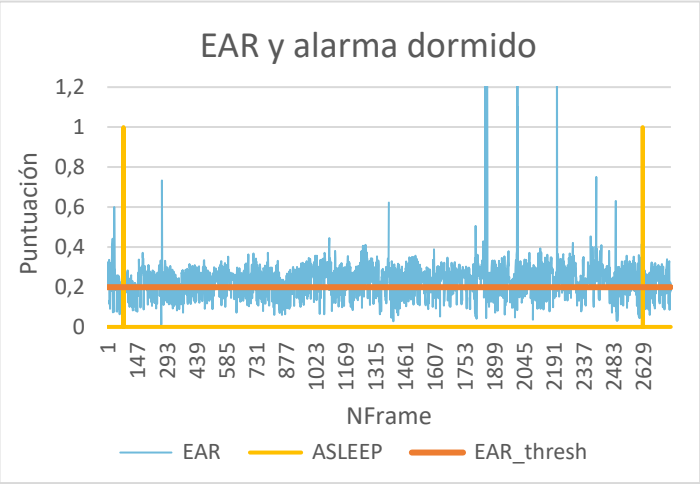


Figura 45. Evolución del EAR en los ensayos 7 y 2 respectivamente, en condiciones reales.

En la primera gráfica de la Figura 45, correspondiente al ensayo 7, la variación del indicador EAR se encuentra entre 0,1 y 0,4 en su mayoría. Existen algunos puntos en los que los ojos se encontraban mucho más abiertos, llegando a superar la puntuación de la unidad, o mucho más cerrados, aproximándose a cero. El EAR_thresh de este ensayo era 0,22, que a pesar de encontrarse por debajo la puntuación del EAR de este en algunas ocasiones, en ninguna ha durado el tiempo suficiente como para activar la alarma referente a dormido.

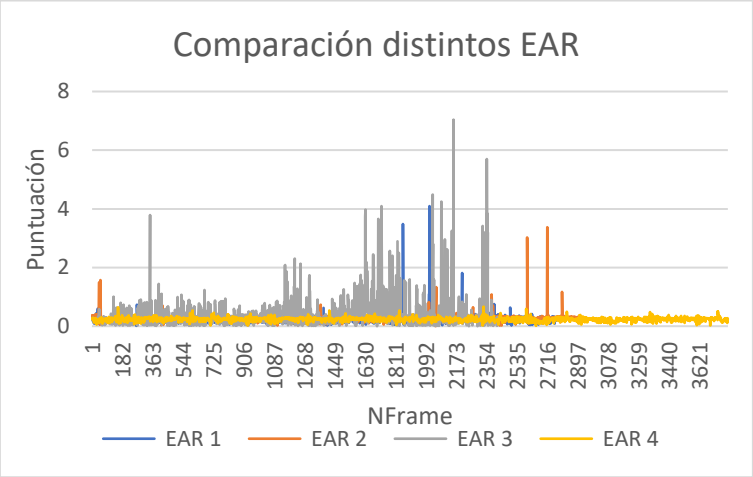
En la segunda gráfica se ha ajustado el eje vertical al máximo de 1,5 en la puntuación del EAR para apreciar mejor la variación en la mayor parte del ensayo. Solamente hay tres valores que superan dicha puntuación y se consideran aislados. En este ensayo, el número 2, ha existido menor variación alrededor de la puntuación promedio, que se encuentra sobre 0,3, por encima de su EAR_thresh que ha sido de 0,25.

El indicador EAR se ha utilizado, además de para calcular el indicador PERCLOS, para la activación de la alarma dormido. Si la puntuación EAR se encuentra durante el tiempo estipulado en los parámetros por debajo de su umbral, el EAR_thresh, se produce la alarma de dormido.



En la Figura 46 se muestra la evolución del EAR en el ensayo 1 en condiciones reales junto al EAR_thresh, de valor aproximado 0,2, y las activaciones de la alarma dormido. Estas alarmas de producen justo cuando han pasado 3 segundos de manera consecutiva en los que los ojos se encontraban cerrados por debajo del umbral.

Figura 46. EAR y alarma dormido en el ensayo 1 en condiciones reales.



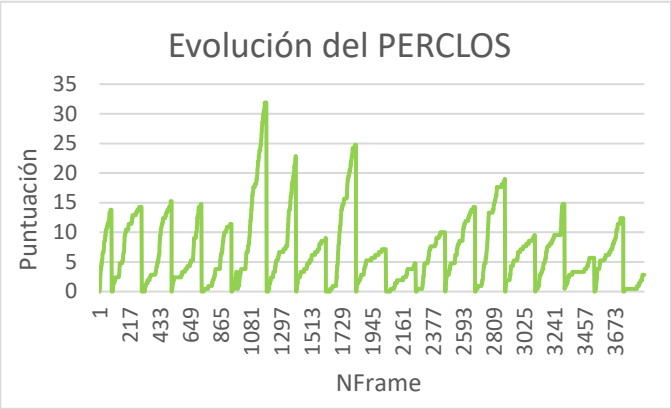
Recopilando los resultados de diferentes ensayos, como el caso de la Figura 47, en la que se comparan los ensayos 1, 2, 3 y 4 en condiciones reales. Se pueden apreciar las diferencias en la evolución del indicador EAR en cada uno de los ensayos, presentando el sujeto del ensayo 4 una puntuación baja y constante del EAR.

Figura 47. Comparación de las puntuaciones EAR de distintos ensayos en condiciones reales.

A diferencia del ensayo 4, en el ensayo 3 el sujeto ha presentado distintos y notables picos en la puntuación que evalúa la apertura de los ojos a lo largo de toda la prueba.

Indicador PERCLOS

El indicador PERCLOS permite evaluar el estado de somnolencia del conductor a través de la cantidad de tiempo en que los ojos permanecen cerrados, por lo que el umbral del EAR es crucial para un buen cálculo del PERCLOS. Habiendo personalizado estos umbrales, el PERCLOS se ajusta pues a las características físicas de cada conductor.

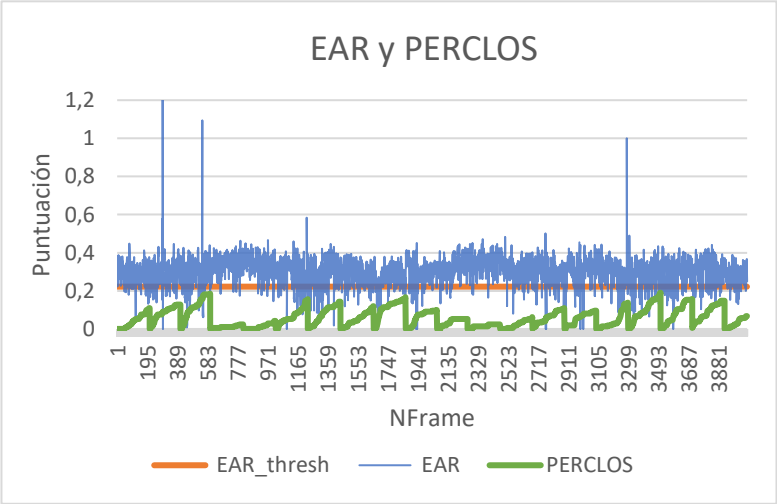


En la Figura 48 se aprecia la evolución del PERCLOS en uno de los ensayos en condiciones reales, donde cada minuto presenta una puntuación acumulativa de este indicador.

En el frame número 1081, alrededor del minuto 6 a 3 fps, se produce el máximo de la puntuación, estando alrededor del 32 %.

Figura 48. Evolución del PERCLOS en el ensayo 11 en condiciones reales.

La relación entre el indicador PERCLOS y el EAR es muy estrecha ya que, como se ha expuesto en los cálculos de los indicadores, el primero se obtiene a partir del segundo. En la Figura 49 se presenta el ejemplo de la relación que guardan ambos en el ensayo 7 en condiciones reales.



En la Figura 49 se observa que cuando la puntuación del EAR se encuentra por debajo de su umbral (EAR_thresh, de color naranja) el PERCLOS aumenta. Es lógico este comportamiento ya que el PERCLOS contabiliza los frames en que los ojos se encuentran cerrados.

Figura 49. Relación entre el indicador EAR y el indicador PERCLOS en el ensayo 7 en condiciones reales.

Realizando una comparación entre distintos ensayos en condiciones reales en la Figura 50, se observa que cada sujeto presenta un comportamiento particular durante la conducción, en algunos casos llegando a puntuaciones altas, como es el caso del ensayo 11 superando el 30 % en un minuto. En los ensayo 8 y 13 los sujetos se encuentran con un promedio inferior al 10 %.

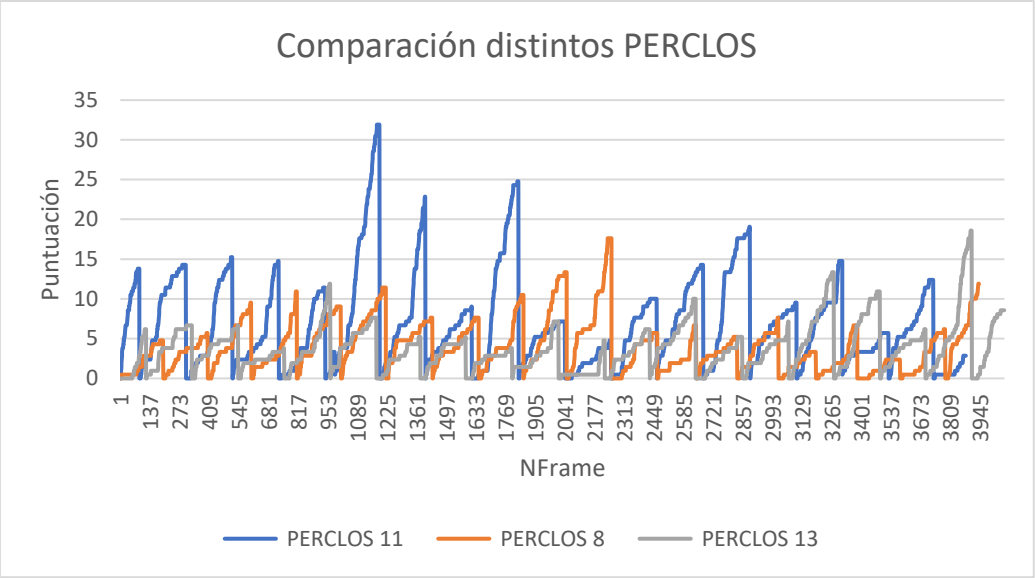
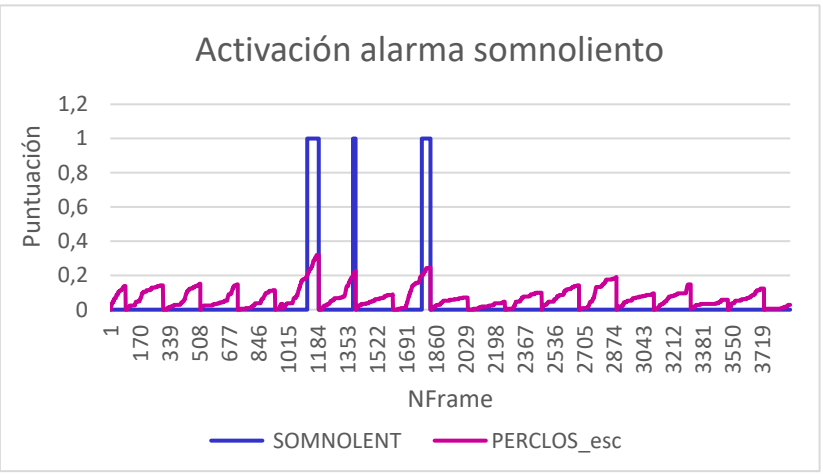


Figura 50. Comparación entre las puntuaciones PERCLOS de los ensayos 11, 8 y 13 en condiciones reales.

Como se presenta en apartados anteriores, es a partir de este indicador que se produce o no la alarma de somnoliento, la más representativa en este trabajo. Para representar la activación de dicha alarma junto al PERCLOS en una gráfica y poderse visualizar correctamente, se ha escalado el PERCLOS entre 0 y 1 (PERCLOS_esc).



En la Figura 51 se puede observar cómo se activa la alarma somnoliento cuando la puntuación del PERCLOS supera el 20%. Esto resulta ser en 3 minutos diferentes, y en el resto la alarma se mantiene desactivada.

Figura 51. Activación de alarma somnoliento en el ensayo 11 en condiciones reales.

Como se ha mostrado en el apartado de ensayos, la puntuación del PERCLOS se ha promediado utilizando las puntuaciones acumuladas en cada minuto de ensayo. Además, se ha destacado también la puntuación máxima en un minuto. Los resultados recogidos en los ensayos en condiciones reales se presentan a continuación.

Tabla 18. Recopilación de las puntuaciones del PERCLOS y las activaciones de la alarma somnoliento en los ensayos en condiciones reales.

		Puntuación PERCLOS		Activaciones alarma
		Promedio	Máximo	Somnoliento
Ensayo	Ensayo 1	19,94%	33,33%	7
	Ensayo 2	9,24%	28,57%	2
	Ensayo 3	14,98%	35,24%	1
	Ensayo 4	12,53%	28,10%	1
	Ensayo 5	8,39%	11,43%	1
	Ensayo 6	14,19%	29,05%	4
	Ensayo 7	11,10%	19,05%	0
	Ensayo 8	8,19%	17,62%	0
	Ensayo 9	12,76%	19,76%	0
	Ensayo 10	29,30%	50,00%	8
	Ensayo 11	13,61%	31,90%	3
	Ensayo 12	9,40%	18,57%	0
	Ensayo 13	7,76%	18,57%	0

En la Tabla 18 se exponen las distintas puntuaciones obtenidas a lo largo de los ensayos en condiciones reales. La puntuación promedio del PERCLOS abarca desde el 7,76 % hasta el 29,30 %.

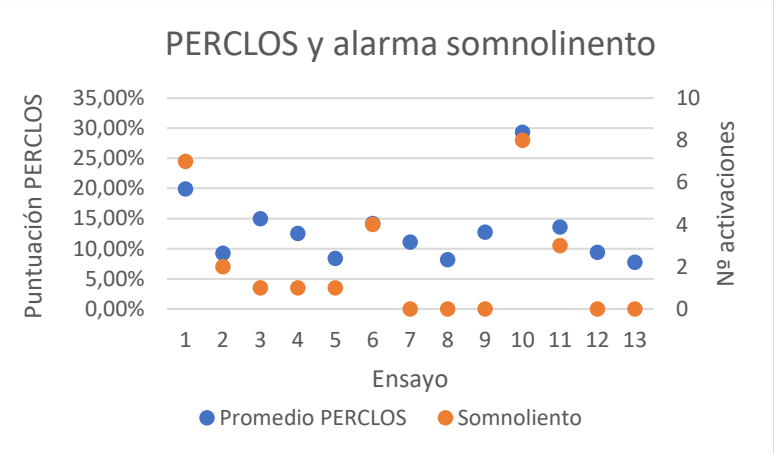
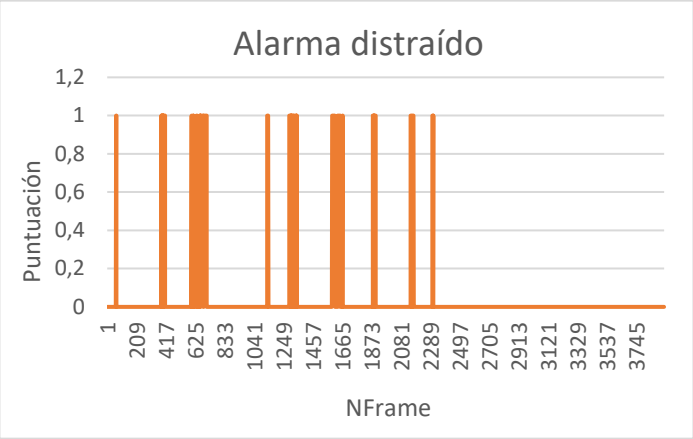


Figura 52. Puntuación promedio PERCLOS y alarma somnoliento en los ensayos en condiciones reales.

En la Figura 52 se observa la relación existente entre las puntuaciones promedio del indicador PERCLOS y el número de activaciones de la alarma somnoliento. Cuanto más cerca se encuentren los marcadores, más se parecen, lo que indicaría que a mayor promedio de puntuación PERCLOS, mayor número de activaciones de la alarma.

Indicador Head Pose

Para reconocer la posición de la cabeza se han utilizado los ángulos de Euler. A partir de estos ángulos, se ha definido un rango de movimiento habitual en el que se considera que el conductor está enfocado en la carretera. Fuera de dicho rango y durante más de 6 segundos consecutivos, se ha considerado que el conductor padece una distracción, situación que activa la alarma de distraído.



En la Figura 53 se presentan las activaciones de la alarma distraído basadas en los ángulos de Euler para ubicar la cabeza del conductor. Se trata del ensayo número 8 en condiciones reales, llegando a registrar 9 activaciones de la alarma.

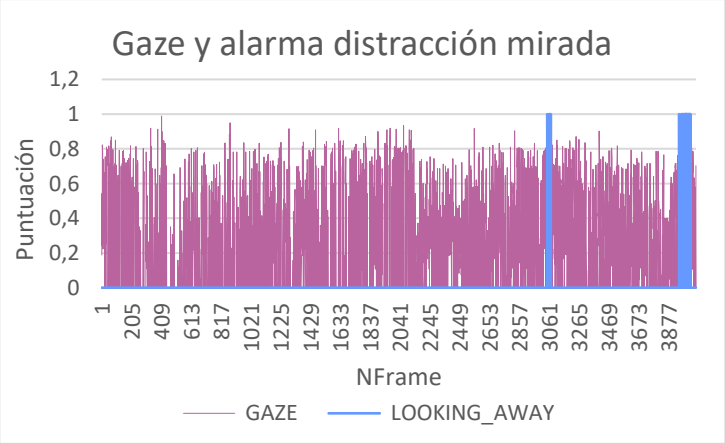
Figura 53. Activaciones alarma distraído en el ensayo 8 en condiciones reales.

Sin embargo, cabe destacar que el cálculo de los ángulos de Euler se ha visto en ocasiones limitado por el detector de rostro, que cuando mejores resultados obtiene es cuando la cabeza se encuentra de manera frontal a la cámara. Por eso y porque la conducción está repleta de estímulos a veces difíciles de determinar, estas alarmas se han considerado de las menos relevantes.

Indicador Gaze

El indicador de la mirada (Gaze en inglés) se ha utilizado para activar la alarma de distracción de la mirada, con el objetivo de establecer una diferencia entre mover la cabeza y cambiar la dirección de la mirada. Según la bibliografía estudiada, las personas pueden realizar más movimientos de la cabeza cuando se encuentran cansadas, pero no necesariamente desviar la mirada.

En la mayoría de los ensayos no se han producido distracciones de la mirada, pues se requiere que el detector de rostro ubique correctamente la cara del conductor, se localicen los ojos y se diferencie lo suficientemente bien el iris de la esclerótica del ojo. Además, para activar la alarma debe producirse una desviación de la mirada durante un mínimo de 4 segundos consecutivos.



En la Figura 54, perteneciente al ensayo 7 en condiciones reales, se producen 2 activaciones de la alarma de distracción de la mirada.

Figura 54. Gaze y alarma distracción de la mirada en ensayo 7 en condiciones reales.

Haciendo zoom en la primera de dichas activaciones:

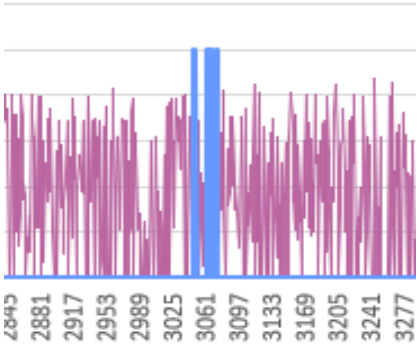


Figura 55. Zoom Figura 54 para observar el detalle de la activación de la alarma distracción de la mirada.

Se observa cómo se produce la activación tras varios frames, correspondientes a más de 4 segundos consecutivos, superándose el umbral Gaze, Gaze_thresh.

El valor de dicho umbral ha sido definido previamente en el algoritmo de 0,4. Se corresponde con la desviación excesiva de la mirada para considerarla no centrada.

Escala de somnolencia de Karolinska

Respecto a la escala de somnolencia de Karolinska (KSS), cada sujeto se ha autoevaluado un total de 2 veces a lo largo de cada ensayo. Se expone a continuación una tabla resumen de los resultados obtenidos, añadiendo una columna de PERCLOS promedio y otra de las activaciones de la alarma somnoliento.

Tabla 19. Recopilación de las autoevaluaciones de los sujetos en los ensayos en condiciones reales en la escala KSS. También se incluye el PERCLOS promedio obtenido y las activaciones de la alarma somnoliento.

		KSS_1	KSS_2	PERCLOS promedio	Somnoliento
Ensayo	**Ensayo 1**	6	6	19,94%	7
	Ensayo 2	5	6	9,24%	2
	Ensayo 3	5	5	14,98%	1
	Ensayo 4	6	7	12,53%	1
	Ensayo 5	4	5	8,39%	1
	Ensayo 6	8	8	14,19%	4
	Ensayo 7	3	3	11,10%	0
	Ensayo 8	3	4	8,19%	0
	Ensayo 9	5	5	12,76%	0
	Ensayo 10	6	6	29,30%	8
	Ensayo 11	8	8	13,61%	3
	Ensayo 12	7	6	9,40%	0
	Ensayo 13	4	4	7,76%	0

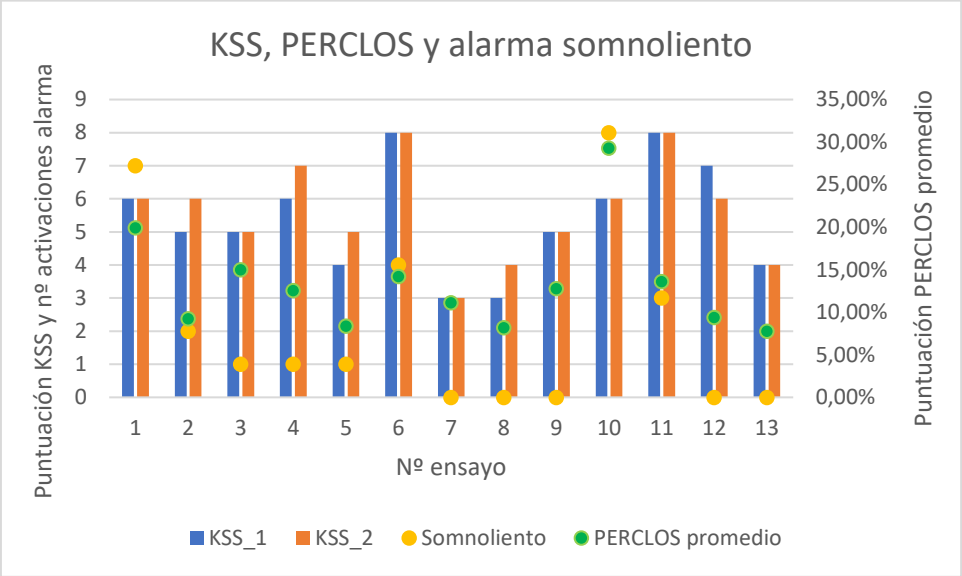


Figura 56. KSS, PERCLOS y alarma somnoliento en los ensayos en condiciones reales.

Analizando la Figura 56, se percibe cierta relación entre las puntuaciones promedio del PERCLOS y las autoevaluaciones de los sujetos en la escala KSS. Sin embargo, al tratarse de una escala subjetiva, es posible que los sujetos hayan tenido una percepción que no coincide con la realidad que arrojan los datos. Es el caso del ensayo 6, en el que el sujeto se ha evaluado en el nivel 8, pero ha presentado un promedio de PERCLOS cercano al 15 %. A pesar de la relativa baja puntuación promedio de PERCLOS, se han producido 4 activaciones de la alarmas somnoliento. Puede ser que el promedio del PERCLOS sea bajo en comparación con la cantidad de activaciones de la alarma, lo que significaría que ha habido minutos en los que ha presentado bastante somnolencia, la suficiente para superar el 20 % del umbral de PERCLOS, y en otros ha presentado muy poca.

En varios de los ensayos se sigue una correlación proporcional entre estos parámetros, como se puede apreciar por ejemplo en los ensayos 1 y 10. Con una muestra más amplia de ensayos, podría determinarse dicha relación para ponderar la importancia de la escala de KSS en este algoritmo para utilizar como señal de referencia del estado de somnolencia del conductor.

9. CONCLUSIONES

Los objetivos de este trabajo eran principalmente los dos siguientes: desarrollar e introducir mejoras en el algoritmo de detección de somnolencia y realizar un apartado de ensayos más detallado y profundo.

Se han introducido varios indicadores nuevos como son el Gaze, que analiza la dirección de la mirada, la posición e inclinación de la cabeza (Head Pose) definida por los tres ángulos de Euler yaw, roll y pitch, y finalmente el PERCLOS. Este último es el más relevante en la bibliografía de la detección de somnolencia en personas a través del análisis facial. No solamente se han calculado y registrado estos indicadores en cada frame de cada ensayo, si no que, además se han planteado 5 tipos de alarmas relacionadas con estos, que expresan con más claridad el significado de dichos datos. Las alarmas que se han implementado son las siguientes: dormido,

somnoliento, distraído, bostezando y con la mirada distraída. Con ellos, se consigue definir a lo largo del tiempo un cuadro de somnolencia en el conductor.

La rutina de inicialización, que era de gran peso en la mejora del algoritmo, ha tenido resultados satisfactorios arrojando una personalización necesaria para evaluar correctamente los indicadores en cada sujeto. En la tabla recopilatoria de los umbrales de los ensayos en condiciones reales se puede apreciar la variabilidad en el umbral del EAR, el EAR_thresh, con puntuaciones desde 0,17 hasta 0,28 y en el MAR_thresh, desde 0,32 hasta 0,49.

El apartado de ensayos es el más extenso de este trabajo y en él se reflejan los resultados del algoritmo diseñado ante diferentes circunstancias. Tanto en los ensayos realizados en datasets como aquellos realizado en directo, utilizando la Raspberry Pi y la NoIR Camera en simulación y sobre todo en condiciones reales, se ha recogido una gran cantidad de datos que en su mayoría indican el buen desempeño del algoritmo. Los ensayos se han llevado a cabo en sujetos de diferente género y diferentes edades comprendidas entre 22 y 58 años, con distintas características físicas. Sin embargo, para la muestra solo se han conseguido personas blancas de etnia caucásica, lo que claramente constituye un sesgo.

Por otra parte, se ha mejorado la detección del rostro en condiciones de iluminación bajas eliminando la LDR de la lámpara LED de infrarrojos. Esto supone un coste de energía mayor ya que la lámpara de LED se encontrará siempre encendida, sin embargo, permite que en condiciones de poca luminosidad, la iluminación no sea un problema. Esto se ha considerado prioritario en el ámbito de aplicación de este proyecto ya que, aunque la somnolencia se puede presentar en cualquier momento, es predominantemente por la noche cuando se acentúa más.

Se destaca que existen diferentes situaciones en las que la detección de ojos se vuelve más confusa y complicada. Una persona puede cerrar o semicerrar los ojos por diferentes motivos, por ejemplo, porque se encuentra bostezando, quiere enfocar un objeto lejano o le está dando directamente el sol. En este último caso, la mayoría de persona utilizan gafas de sol, que por su opacidad impiden que se puedan monitorizar los ojos y no son aptas para el algoritmo.

10. TRABAJOS FUTUROS

Para trabajos futuros en este campo se propone:

- Adquirir un sistema comercial para poder realizar un estudio comparativo entre los resultados de ambos sistemas.
- Incorporación de las señales EEG del conductor y análisis de señales del vehículo para llevar a cabo una detección de la somnolencia más robusta.
- Utilizar las puntuaciones de los indicadores de este trabajo para obtener un sistema binario de clasificación para la detección de somnolencia, donde se clasifique entre somnolencia y no somnolencia. Esto supone un problema de aprendizaje supervisado, en el que se utilizarían los resultados de los indicadores como datos de entrada y la señal de referencia, que es la obtenida con el índice de KSS, como etiqueta. El algoritmo de aprendizaje supervisado buscará que la salida sea lo más parecida posible a la etiqueta.

11. REFERENCIAS

[1] Michael Page, «El desplazamiento al lugar de trabajo en España,» [En línea]. Available: https://www.michaelpage.es/prensa-estudios/estudios/transport-commute/resultados-de-espa%C3%B1a#:~:text=En%20Espa%C3%B1a%20el%20tiempo%20medio,debajo%20de%20la%20media%20europea..

[2] DGT, «Censo de conductores - Tablas estadísticas 2021,» 2021.

[3] Klauer et al., «The impact of dirver inattention on near-crash/crash risk: An analysis using the 100-car naturalistic driving study data,» National Highway Traffic Safety Administration, 2006.

[4] J. Cano Bernet, «Diseño de un sistema de bajo coste para la detección de la somnolencia en la conducción basado en reconocimiento de expresiones faciales,» Valencia, 2021.

[5] Fundación MAPFRE, «ADAS (Advanced Driver-Assistance Systems) Sistemas Avanzados de Conducción,» 2018.

[6] DGT, «Conducir con sueño o cansancio,» 30 noviembre 2022. [En línea]. Available: https://www.dgt.es/muevete-con-seguridad/evita-conductas-de-riesgo/Conducir-con-sueno-o-cansancio#:~:text=La%20somnolencia%20interviene%2C%20directa%20o,somnolencia%20es%20un%20factor%20implicado..

[7] A. Shahid, K. Wilkinson, S. Marcu y C. M. Shapiro, «STOP, THAT and One Hundred Other Sleep Scales,» Springer New York, NY, 2012.

[8] Zhong et al, «Localized energy study for analyzing driver fatigue state based on wavelet analysis,» Wavelet Analysis and Pattern Recognition, 2007.

[9] Bergasa et al, «Real-time system for monitoring driver vigilance,» 2006.

[10] García Daza, Iván, «Detección de fatiga en conductores mediante fusión de sistemas ADAS,» 2011.

[11] López Noelia , «La nueva tecnología de Ford para detectar cuando un conductor se duerme al volante,» 7 octubre 2021. [En línea]. Available: https://www.autobild.es/noticias/nueva-tecnologia-ford-detectar-cuando-conductor-duerme-volante-944111.

[12] P. Viola and M. Jones, «Rapid object detection using boosted cascade of simple features,» IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Kauai, HI, USA, 2001.

[13] Dinges David et all, «Evaluation of techniques for ocular measurement as an index of fatigue and the basis for alertness management,» Virginia, USA, 1998.

[14] S. Abtahi, M. Omidyeganeh, S. Shirmohammadi y B. Hariri, *YawDD: A Yawning Detection Dataset,* Proc. ACM Multimedia Systems, Singapore, 2014.

[15] N. Petrellis, N. Voros, C. Antonopoulos, G. Keramidas, P. Christakos y P. Mousouliotis, *NITYMED Dataset,* 2022.

[16] R. Ghoddoosian, M. Galib y V. Athitsos, *A Realistic Dataset and Baseline Temporal Model for Early Drowsiness Detection,* 2019.

12. BIBLIOGRAFÍA

Viola & Jones, «Robust real-time face detection,» 2004.

DGT, «Otros factores de riesgo: la fatiga,» [En línea]. Available: https://www.dgt.es/export/sites/web-DGT/.galleries/downloads/conoce_la_dgt/que-hacemos/educacion-vial/adultos/no-formal/fatiga.pdf.

S. Darshana, D. Fernando, S. Jayawardena, S. Wickramanyake y Dr. C. DeSilva, «Efficient PERCLOS and Gaze Measurement Methodologies to Estimate Driver Attention in Real Time,» Department of Computer Science & Engineering, University of Moratuwa, Sri Lanka, 2014.

D. Cárdenas, J. Conde-González y J.C. Perales, «La fatiga como estado motivacional subjetivo,» Revista Andaluza de Medicina del Deporte, 2017.

S. El Kaddouhi, A. Saaidi y M. Abarkan, «Eye Detection based on Viola & Jones Detector, Skin Color, and Eye Template,» International Journal of Control and Automation, Vol. 11, No. 5, pp. 59-72, 2018.

M. J. Flores, J. M. Armingol y A. de la Escalera, «Sistema Avanzado de Asistencia a la Conducción para la Detección de la Somnolencia,» Revista Iberoamericana de Automática e Informática Industrial RIAI, Vol. 8, pp. 216-228, 2011.

ANEXOS

Anexo A. ODS - Objetivos y metas de desarrollo sostenible

El 25 de septiembre de 2015 se adoptaron una serie de objetivos globales en la agenda de 2030 con el compromiso de velar por el desarrollo sostenible. Su principal propósito reside en mejorar la vida de las personas y proteger el planeta a lo largo de todo el mundo.

Se define en la Tabla 20 la implicación de este trabajo final de grado en cada uno de los objetivos de desarrollo sostenible, denominados ODS.

Tabla 20. Implicación de los ODS en el presente trabajo final de grado.

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**	X			
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades.**				X
ODS 11. **Ciudades y comunidades sostenibles.**			X	
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**		X		

Este trabajo final de grado se encuentra alineado con distintos ODS marcados con una X en la Tabla 20, que se detallan a continuación:

- ODS 8. **Trabajo decente y crecimiento económico.**

  Este trabajo plantea un prototipo de dispositivo capaz de detectar somnolencia en la conducción, lo que supone oportunidades de crecimiento económico para las personas que se dediquen a mejorar su desempeño y para las empresas que en un futuro próximo se dediquen a comercializarlo.

Siendo el sector del transporte de mercancías por carretera uno de los más importantes para la industria en España, un sistema de detección de somnolencia contribuye al trabajo decente influyendo en las condiciones dignas y de seguridad de sus trabajadores.

- ODS 9. **Industria, innovación e infraestructuras.**

  La implementación del sistema desarrollado en este trabajo lleva implícito la aplicación de tecnologías novedosas y avanzadas como es la visión artificial.

  Además, forma parte de las innovaciones en automatización para la industria, constituyendo un sistema que vela por la seguridad de las personas en el interior de los vehículos que circulan por carretera.

- ODS 11. **Ciudades y comunidades sostenibles.**

  La sostenibilidad y la seguridad van de la mano cuando se trata de reducir la accidentalidad en la carretera y sus posibles consecuencias en el medioambiente. Este trabajo presenta su foco en la mejora de la vida de las personas, disminuyendo el riesgo de padecer un accidente que desencadene en daños materiales y humanos.

- ODS 17. **Alianzas para lograr los objetivos.**

  El trabajo desarrollado tiene un carácter colaborativo entre instituciones de educación e investigación, que dedican sus esfuerzos a desarrollar tecnologías que mejoran la vida de las personas y a realizar una labor de divulgación y concienciación de la población, con el sector de la industria que produce y comercializa vehículos. Fruto de su colaboración surgen sistemas como el propuesto en este trabajo, cuyo objetivo es que en un futuro se implementen en dichos vehículos para aportar protección y seguridad a la ciudadanía.

Anexo B. Documentación técnica

Anexo B 1. Hoja resumen de características de la Raspberry Pi 4 Modelo B.

Características de Raspberry Pi 4B	
Procesador	ARM Cortex-A72
Capacidad RAM	8 GB
Frecuencia de reloj	1,5 GHz
GPU	VideoCore VI (con soporte para OpenGL ES 3.x)
Temperatura de funcionamiento	0 °C a 50 °C
Conectividad	Bluetooth 5.0, Wi-Fi 2.4 GHz y 5.0 GHz IEE 802.11b/g/n/ac, Gigabit Ethernet
Puertos	GPIO 40 pines, 2x micro HDMI con soporte 4K 60 fps, 2x USB 2.0, 2x USB 3.0, CSI (cámara Raspberry Pi), DSI (pantalla táctil), Micro SD, salida jack para audio y vídeo, USB-C (alimentación)
Alimentación	Requiere alimentador de 5.1V, 3A
Dimensiones	85 mm x 56 mm x 21 mm

Anexo B 2. Hoja resumen de características de la Pi NoIR Camera v2.

Características de Pi NoIR Camera v2	
Compatibilidad	Raspberry Pi 1,2,3 y 4
Conexión con Raspberry	Cable plano de 15 cm que se conecta al puerto CSI de la Raspberry
Enfoque	Fijo de 8 megapíxeles
Sensor	IMX219PQ de Sony
Temperatura funcionamiento	-20 °C y 60 °C
Imagen fija	3280 x 2464
Captura vídeo	1080p30, 720p60, 640x480p90
Formato óptico	1/4 pulgadas
Tipo de captura	Sin filtro IR
Peso	3 g
Dimensiones	25 mm x 24 mm x 1 mm

Anexo C. Código fuente de programación en Raspberry

Anexo C 1. Módulo Main.py

```
import time
import os

import cv2
import dlib
import numpy as np

from imutils import face_utils
from time import sleep

from picamera import PiCamera
from picamera.array import PiRGBArray

from Utils import get_face_area
from Eye_Mouth_Detector_Module import EyeMouthDetector as EyeMouthDet
from Pose_Estimation_Module import HeadPoseEstimator as HeadPoseEst
from Attention_Scorer_Module import AttentionScorer as AttScorer

from Initialization import startup_routine as startrout

CAPTURE_SOURCE = 0 # Webcam

def main():

    t1 = time.time()
    ctime = 0  # tiempo actual (utilizado para calcular los FPS)
    ptime = 0  # tiempo pasado (utilizado para calcular los FPS)
    count_frame = 1

    # Adaptar el límite de FPS según la fuente de entrada
    if CAPTURE_SOURCE == 0:
        fps_lim = 3.5 # Para la webcam
    else:
        fps_lim = 25 # Para los vídeos de los dataset

    cv2.setUseOptimized(True)  # Activar la optimización de OpenCV

    # Inicializar el detector de caras de dlib
    DetectorHOG = dlib.get_frontal_face_detector()
    DetectorHaar =
cv2.CascadeClassifier(r'/home/lucas/Desktop/Lucas/haarcascade_frontalface_defa
ult.xml')

    # Inicializar el predictor de dlib
    Predictor =
dlib.shape_predictor(r'/home/lucas/Desktop/Lucas/shape_predictor_68_face_landm
arks.dat')

    # Instanciar los módulos de detector de ojos y boca, y el estimador de
pose
    Eye_mouth_det = EyeMouthDet(show_processing=False)
    Head_pose = HeadPoseEst(show_axis=True)

    # Decisión de activar la rutina de inicialización personalizada
    if CAPTURE_SOURCE != 0:
        MAR_thresh = 0.35
        ear_thresh = 0.2


    Initialization = startrout(DetectorHOG, Predictor, CAPTURE_SOURCE)
    ear_thresh, MAR_thresh = Initialization.personalized_ear_and_mar_thresh()
```

```
    #ear_thresh = 0.20
    #MAR_thresh = 0.35

    # Escribir los umbrales de los indicadores MAR y EAR obtenidos
    print("MAR_thresh: " + str(MAR_thresh) + "; EAR_thresh: " +
str(ear_thresh))

    # Inicialización del módulo de calcular la puntuación de atención
    Scorer = AttScorer(capture_fps=fps_lim, ear_tresh=ear_thresh,
mar_thresh=MAR_thresh, ear_time_tresh=3, gaze_tresh=0.4, perclos_tresh=0.2,
                       gaze_time_tresh=4, pitch_tresh=180, roll_tresh=20,
yaw_tresh=30, pose_time_tresh=6, verbose=False, mar_time_tresh=3)

    # Inicializar la captura de imágenes
    camera = PiCamera()
    camera.resolution = (640, 480)
    camera.framerate = 32
    cap = PiRGBArray(camera, size=(640, 480))
    time.sleep(2)

    # Para crear archivos distintos sucesivos en los que guardar el vídeo y el
txt
    i = 0
    Path = "/home/lucas/Desktop/Lucas/ensayo"
    PathAvi = Path + ".avi"
    PathText = Path + ".txt"

    while (os.path.exists(PathAvi) == True):
        PathAvi = Path + str(i) + ".avi"
        PathText = Path + str(i) + ".txt"

        i = i + 1

    # Para grabar el vídeo
    ancho = 640
    alto = 480
    codec = cv2.VideoWriter_fourcc(*'XVID')
    grabador = cv2.VideoWriter(os.path.abspath(PathAvi), codec, fps_lim,
(ancho, alto))

    # Para escribir en un archivo los datos del vídeo
    f = open(os.path.abspath(PathText), 'w')
    #TIEMPO EAR MAR PERCLOS GAZE YAWN SOMNOLENT ASLEEP LOOKING_AWAY DISTRACTED
    f.write('NFRAME\t FPS\t DETECTOR\t MAR_thresh\t EAR_thresh\t EAR\t MAR\t
PERCLOS\t GAZE\t YAWN\t SOMNOLENT\t ASLEEP\t LOOKING_AWAY\t DISTRACTED\t\n')

    # Bucle infinito para capturar los frames con la cámara
    for frames in camera.capture_continuous(cap, format="bgr", use_video_port
= True):

        # Leer un frame
        frame = frames.array

        # Calcular los FPS actuales y mostrarlos
        ctime = time.perf_counter()
        fps = 1.0 / float(ctime - ptime)
        ptime = ctime
        cv2.putText(frame, "FPS:" + str(round(fps, 0)), (10, 400),
cv2.FONT_HERSHEY_PLAIN, 2, (255, 150, 50), 2)

        # Transformar el frame en BGR a escala de grises
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # Aplicar un filtro bilateral para reducir el ruido y resaltar los
detalles
        gray = cv2.bilateralFilter(gray, 5, 10, 10)

        # Uso el detector frontal HOG
```

```
        faces = DetectorHOG(gray)

        # Para ver si usa HOG o Haar
        detectortype = 0

        # Pero si este no encuentra ninguna cara, utilizo el de cvlib
(cascadas de Haar)

        if len(faces) == 0:
            faces = DetectorHaar.detectMultiScale(gray, scaleFactor = 1.1,
minNeighbors = 5, minSize = (30,30))

            detectortype = 1

            try:
                # Lo convierto a el formato rectángulo de dlib para luego
facilitar el tratamiento
                left, top, right, bottom = faces[0]
                faces = dlib.rectangle(int(left), int(top), int(right),
int(bottom))
            except:
                None


        # Procesa el frame si encuentra al menos una cara
        if faces:

            # Solo cojo la cara más grande, que será la del conductor
            try:
                faces = sorted(faces, key=get_face_area, reverse=True)
                faces = faces[0]
            except:
                None

            # Utilizo el predictor de los 68 puntos clave y lo muestro en el
frame
            landmarks = Predictor(frame, faces)
            Eye_mouth_det.show_eye_keypoints(color_frame=frame,
landmarks=landmarks)

            #### Para mostrar los contornos ####

            # Crear mapa de puntos
            puntos_coordenadas = face_utils.shape_to_np(landmarks)
            # Se extraen las coordenadas de cada ojo
            ojo_izq = puntos_coordenadas[42:48]
            ojo_der = puntos_coordenadas[36:42]

            # Se extraen las coordenadas internas de la boca
            mouth = puntos_coordenadas[60:68]

            # Mostrar el contorno de los ojos
            contorno_ojo_izq = cv2.convexHull(ojo_izq)
            contorno_ojo_der = cv2.convexHull(ojo_der)
            cv2.drawContours(frame, [contorno_ojo_izq], -1, (0, 255, 0), 1)
            cv2.drawContours(frame, [contorno_ojo_der], -1, (0, 255, 0), 1)
            # Mostrar el contorno de la boca
            contorno_boca = cv2.convexHull(mouth)
            cv2.drawContours(frame,[contorno_boca],-1, (255, 0, 0), 1)

            #### Cálculo de los indicadores y evaluación ####

            # Cálculo del EAR
            ear = Eye_mouth_det.get_EAR(frame=gray, landmarks=landmarks)

            # Cálculo del MAR
            MAR = Eye_mouth_det.get_MAR(mouth)
```

```
            # Calcular el PERCLOS y el estado de somnolencia
            somnolent, perclos_score = Scorer.get_PERCLOS(ear)

            # Calcular el Gaze Score
            gaze = Eye_mouth_det.get_Gaze_Score(frame=gray,
landmarks=landmarks)

            # Calcular la posición de la cabeza
            frame_det, roll, pitch, yaw = Head_pose.get_pose(frame=frame,
landmarks=landmarks)

            # Evaluar EAR, MAR, GAZE y HEAD POSE
            asleep, looking_away, distracted, yawn = Scorer.eval_scores(
                ear, gaze, roll, pitch, yaw, MAR)

            # Si la estimación de la pose es satisfactoria, mostrar los
resultados
            if frame_det is not None:
                frame = frame_det

            # Mostrar EAR
            if ear is not None:
                cv2.putText(frame, "EAR:" + str(round(ear, 3)), (10, 110),
                            cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 1,
cv2.LINE_AA)

            # Mostrar MAR
            if MAR is not None:
                cv2.putText(frame, "MAR:" + str(round(MAR, 3)), (10, 80),
                            cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 1,
cv2.LINE_AA)

            # Mostrar Gaze Score
            if gaze is not None:
                cv2.putText(frame, "Gaze Score:" + str(round(gaze, 3)), (10,
460),
                            cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 51), 1,
cv2.LINE_AA)
            else:
                cv2.putText(frame, "Gaze Score:0.0", (10, 460),
                            cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 51), 1,
cv2.LINE_AA)

            # Mostrar PERCLOS
            cv2.putText(frame, "PERCLOS:" + str(round(perclos_score, 3)), (10,
50),
                        cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 1,
cv2.LINE_AA)

            #### Mostrar evaluaciones ####

            # Si el conductor está somnoliento, mostrar
            if somnolent:
                cv2.putText(frame, "SOMNOLIENTO", (10, 260),
                            cv2.FONT_HERSHEY_PLAIN, 1.5, (0, 0, 255), 1,
cv2.LINE_AA)

            # Si el conductor está dormido, mostrar
            if asleep:
                cv2.putText(frame, "DORMIDO", (10, 290),
                            cv2.FONT_HERSHEY_PLAIN, 1.5, (0, 0, 255), 1,
cv2.LINE_AA)

            # Si el conductor tiene la mirada distraída, mostrar
            if looking_away:
                    cv2.putText(frame, "DISTRACCION DE LA MIRADA!", (10, 340),
                                cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 255), 1,
cv2.LINE_AA)
```

```
            # Si el conductor está mirada distraído, mostrar
            if distracted:
                cv2.putText(frame, "DISTRAIDO!", (10, 320),
                            cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 255), 1,
cv2.LINE_AA)

            # Si el conductor está bostezando, mostrar
            if yawn == True:
                cv2.putText(frame,"BOSTEZANDO", (10, 200),
cv2.FONT_HERSHEY_PLAIN,
                        1.5, (0, 0, 255), 1, cv2.LINE_AA)

            # Escribir en el fichero de texto
            # NFRAME FPS DETECTOR MAR_THRESH EAR_THRESH EAR MAR PERCLOS GAZE
YAWN SOMNOLENT ASLEEP LOOKING_AWAY DISTRACTED
            f.write(str(count_frame) + '\t' + str(fps) + '\t' +
str(detectortype) + '\t' + str(MAR_thresh) + '\t' + str(ear_thresh) + '\t' +
str(ear) + '\t' + str(MAR) + '\t' + str(perclos_score) + '\t' + str(gaze) +
'\t' + str(yawn) + '\t' + str(somnolent) + '\t' + str(asleep) + '\t' +
str(looking_away) + '\t' + str(distracted) + '\n')

        # Guardar frame en el vídeo
        grabador.write(frame)

        count_frame = count_frame + 1 # Contar los frames

        cap.truncate(0)

        # Sale del bucle si se acaba el tiempo
        if ((time.time() - t1 >= 20*60) or (cv2.waitKey(1) & 0xFF ==
ord('q'))):
            break

    # Cerrar webcam
    camera.close()
    cv2.destroyAllWindows()

    return

# Llamada al main
if __name__ == "__main__":
    main()
```

Anexo C 2. Módulo Eye_Mouth_Detector_Module.py

```
import cv2
import numpy as np
from numpy import linalg as LA
from Utils import resize
from scipy.spatial import distance as calculo_distancia

class EyeMouthDetector:

    def __init__(self, show_processing: bool = False):
        """
        La clase Eye detector contiene varios metodos para estimar la apertura
de los ojos, de la boca y del gaze

        Parámetros
        ----------
        show_processing: bool
            Si se pone a True, muestra los frames durante el procesamiento en
algunos pasos
```

```
        Métodos
        ----------
        - show_eye_keypoints: muestra los keypoints de los ojos en el frame
        - get_EAR: calcula el parámetro EAR para los dos ojos
        - get_MAR: calcula el parámetro MAR para la boca
        - get_Gaze_Score: calcula el Gaze (la distancia euclídea normalizada
entre el centro del ojo y la pupila)
        """

        self.keypoints = None
        self.frame = None
        self.show_processing = show_processing
        self.eye_width = None

    def show_eye_keypoints(self, color_frame, landmarks):
        """
        Muestra los keypoints de los ojos encontrados en la cara, dibujando
círculos rojos en esas posiciones del frame

        Parámetros
        ----------
        color_frame: numpy array
            Frame a color
        landmarks: list
            Lista de los 68 dlib keypoints de la cara
        """

        self.keypoints = landmarks

        for n in range(36, 48):
            x = self.keypoints.part(n).x
            y = self.keypoints.part(n).y
            cv2.circle(color_frame, (x, y), 1, (0, 0, 255), -1)
        return

    def get_EAR(self, frame, landmarks):
        """
        Calcula el eye aperture rate de la cara

        Parámetros
        ----------
        frame: numpy array
            Frame a color
        landmarks: list
            Lista de los 68 dlib keypoints de la cara

        Returns
        --------
        ear_score: float
            EAR average entre los dos ojos
            El EAR o Eye Aspect Ratio se calcula como la apertura del ojo
dividida entre la longitud del ojo
            Cada ojo tiene una puntuación y las dos puntuaciones se promedian
        """

        self.keypoints = landmarks
        self.frame = frame
        pts = self.keypoints

        i = 0  # contador auxiliar
        # array para almacenar las posiciones de los keypoints del ojo
izquierdo
        eye_pts_l = np.zeros(shape=(6, 2))
        # array para almacenar las posiciones de los keypoints del ojo derecho
        eye_pts_r = np.zeros(shape=(6, 2))

        for n in range(36, 42):  # los dlib keypoints del 36 al 42 se refieren
al ojo izquierdo
```

```
            point_l = pts.part(n)  # guardar los i-keypoint del ojo izquierdo
            point_r = pts.part(n + 6)  # guardar los i-keypoint del ojo
derecho
            # array de las coordenadas x,y para el punto de referencia del ojo
izquierdo
            eye_pts_l[i] = [point_l.x, point_l.y]
            # array de las coordenadas x,y para el punto de referencia del ojo
derecho
            eye_pts_r[i] = [point_r.x, point_r.y]
            i += 1  # incrementar contador

        def EAR_eye(eye_pts):
            """
            Calcular el EAR score para un solo ojo dados sus keypoints
            :param eye_pts: numpy array of shape (6,2) que contienen los
keypoints de un ojo considerando dlib
            :return: ear_eye
                EAR del ojo
            """
            ear_eye = (LA.norm(eye_pts[1] - eye_pts[5]) + LA.norm(
                eye_pts[2] - eye_pts[4])) / (2 * LA.norm(eye_pts[0] -
eye_pts[3]))
            '''
            EAR se calcula como la media de las dos medidas de apertura del
ojo dividido entre la longitud del ojo
            '''
            return ear_eye

        ear_left = EAR_eye(eye_pts_l)  # calcular EAR para el ojo izquierdo
        ear_right = EAR_eye(eye_pts_r)  # calcular EAR para el ojo derecho

        # calcular la media del EAR
        ear_avg = (ear_left + ear_right) / 2

        return ear_avg

    def get_MAR(self, mouth):
        """
        Calcular el mouth aspect ratio (MAR)
        Se establece la relación entre la longitud y el ancho de la región de
la boca

        Parámetros
        ----------
        mouth: numpy array
            Keypoints de la boca

        Returns
        -------
        MAR: float
            Puntuación del MAR
        """

        # calcular la distancia entre los puntos que definen la posición de la
boca interna según el dlib68
        A = calculo_distancia.euclidean(mouth[1], mouth[7])
        B = calculo_distancia.euclidean(mouth[2], mouth[6])
        C = calculo_distancia.euclidean(mouth[3], mouth[5])
        D = calculo_distancia.euclidean(mouth[0], mouth[4])
        # Cálculo MAR
        MAR = (A+B+C)/2.0/D
        return MAR

    def get_Gaze_Score(self, frame, landmarks):
        """
        Calcular el promedio Gaze para los ojos
        El Gaze Score es la media de la norma l2 (distancia euclidiana) entre
el punto central de la ROI del ojo
```

```
        (la bounding box del ojo) y el centro de la pupila

        Parámetros
        ----------
        frame: numpy array
            Frame en el que se encuentran los ojos
        landmarks: list
            Lista de los 68 dlib keypoints de la cara

        Returns
        --------
        avg_gaze_score: float
            If successful, returns the float gaze score
            Si no es cero, devuelve la puntuación calculada
            Si es cero, devuelve None
        """
        self.keypoints = landmarks
        self.frame = frame

        def get_ROI(left_corner_keypoint_num: int):
            """
            Obtiene la ROI bounding box del ojo a partir del keypoint del ojo

            :param left_corner_keypoint_num:  el keypoint del ojo más a la
izquierda
            :return: eye_roi
                Sub-frame de la región del ojo del frame
            """

            kp_num = left_corner_keypoint_num

            eye_array = np.array(
                [(self.keypoints.part(kp_num).x,
self.keypoints.part(kp_num).y),
                 (self.keypoints.part(kp_num+1).x,
                  self.keypoints.part(kp_num+1).y),
                 (self.keypoints.part(kp_num+2).x,
                  self.keypoints.part(kp_num+2).y),
                 (self.keypoints.part(kp_num+3).x,
                  self.keypoints.part(kp_num+3).y),
                 (self.keypoints.part(kp_num+4).x,
                  self.keypoints.part(kp_num+4).y),
                 (self.keypoints.part(kp_num+5).x,
self.keypoints.part(kp_num+5).y)], np.int32)

            min_x = np.min(eye_array[:, 0])
            max_x = np.max(eye_array[:, 0])
            min_y = np.min(eye_array[:, 1])
            max_y = np.max(eye_array[:, 1])

            eye_roi = self.frame[min_y-2:max_y+2, min_x-2:max_x+2]

            return eye_roi

        def get_gaze(eye_roi):
            """
            Calcula la norma L2 entre el punto central de la ROI del ojo y el
centro de la pupila
            :param eye_roi: float
            :return: (gaze_score, eye_roi): tuple
                tuple
            """

            eye_center = np.array(
                [(eye_roi.shape[1] // 2), (eye_roi.shape[0] // 2)])  #
posición central de la ROI del ojo
            gaze_score = None
            circles = None
```

```
            # filtro bilateral para reducir el sonido y resaltar los detalles
            if eye_roi.any():
                eye_roi = cv2.bilateralFilter(eye_roi, 4, 40, 40)

                circles = cv2.HoughCircles(eye_roi, cv2.HOUGH_GRADIENT, 1, 10,
                                           param1=90, param2=6, minRadius=1,
maxRadius=9)
            # Transformada de Hough para encontrar el iris y su centro (la
pupila) en la eye_roi de la imagen en escala de grises

            if circles is not None and len(circles) > 0:
                circles = np.uint16(np.around(circles))
                circle = circles[0][0, :]

                cv2.circle(
                    eye_roi, (circle[0], circle[1]), circle[2], (255, 255,
255), 1)
                cv2.circle(
                    eye_roi, (circle[0], circle[1]), 1, (255, 255, 255), -1)

                # la posición de la pupula es el primer círculo que se
encuentra con la transformada de Hough
                pupil_position = np.array([int(circle[0]), int(circle[1])])

                cv2.line(eye_roi, (eye_center[0], eye_center[1]), (
                    pupil_position[0], pupil_position[1]), (255, 255, 255), 1)

                gaze_score = LA.norm(
                    pupil_position - eye_center) / eye_center[0]
                # calcula la L2 distancia entre el centro del ojo y el centro
de la pupila

            cv2.circle(eye_roi, (eye_center[0],
                                 eye_center[1]), 1, (0, 0, 0), -1)

            if gaze_score is not None:
                return gaze_score, eye_roi
            else:
                return None, None

        left_eye_ROI = get_ROI(36)  # calcula la ROI para el ojo izquierdo
        right_eye_ROI = get_ROI(42)  # calcula la ROI para el ojo derecho

        # calcula el gaze para los ojos
        gaze_eye_left, left_eye = get_gaze(left_eye_ROI)
        gaze_eye_right, right_eye = get_gaze(right_eye_ROI)

        # si show_processing es True, muestra la ROI de los ojos, el centro
del ojo, el centro de la pupila y las líneas de distancia
        if self.show_processing and (left_eye is not None) and (right_eye is
not None):
            left_eye = resize(left_eye, 1000)
            right_eye = resize(right_eye, 1000)
            cv2.imshow("left eye", left_eye)
            cv2.imshow("right eye", right_eye)

        if gaze_eye_left and gaze_eye_right:

            # calcula el promedio del gaze score para los 2 ojos
            avg_gaze_score = (gaze_eye_left + gaze_eye_left) / 2
            return avg_gaze_score

        else:
            return None
```

Anexo C 3. Módulo Pose_Estimation_Module.py

```
import cv2
import numpy as np

from Utils import rotationMatrixToEulerAngles, draw_pose_info


class HeadPoseEstimator:

    def __init__(self, camera_matrix=None, dist_coeffs=None, show_axis: bool =
False):
        """
        Head Pose estimator contiene el método get_pose para calcular los 3
ángulos de Euler (roll, pitch, yaw)
        de la cabeza. Usa el frame, los landmarks de la cabeza detectados con
dlib y, opcionalmente, los
        parámetros de la cámara

        Parámetros
        ----------
        camera_matrix: numpy array
            La matriz de la cámara usada
        dist_coeffs: numpy array
            Los coeficientes de distorsión de la cámara
        show_axis: bool
            Para enseñar los ejes proyectados sobre la nariz
        """

        self.verbose = show_axis
        self.camera_matrix = camera_matrix
        self.dist_coeffs = dist_coeffs

    def get_pose(self, frame, landmarks):
        """
        Estima la posición de la cabeza usando el estimador de pose

        Parámetros
        ----------
        frame: numpy array
            Frame capturado por la cámara
        landmarks: dlib.rectangle
            Los 68 landkmarks de la cabeza detectados con dlib

        Devuelve
        --------
        - if successful: image_frame, roll, pitch, yaw (tuple)
        - if unsuccessful: None,None,None,None (tuple)

        """
        self.keypoints = landmarks  # dlib 68 landmarks
        self.frame = frame

        self.axis = np.float32([[200, 0, 0],
                                [0, 200, 0],
                                [0, 0, 200]])
        # array que especifica el largo de los 3 ejes que se proyectan sobre
la nariz

        if self.camera_matrix is None:
            # Si no hay una matriz de la cámara, estimar los parámetros
utilizando un frame
            self.size = frame.shape
            self.focal_length = self.size[1]
            self.center = (self.size[1] / 2, self.size[0] / 2)
            self.camera_matrix = np.array(
                [[self.focal_length, 0, self.center[0]],
```

```
                [0, self.focal_length, self.center[1]],
                [0, 0, 1]], dtype="double"
            )

        if self.dist_coeffs is None:  # Asumir que no hay distorsión de lente
si no se proporcionan los coeficientes de distorsión
            self.dist_coeffs = np.zeros((4, 1))

        # Modelo 3D de la cabeza génerica humana
        self.model_points = np.array([
            (0.0, 0.0, 0.0),  # Nariz
            (0.0, -330.0, -65.0),  # Mentón
            (-225.0, 170.0, -135.0),  # Left eye left corner
            (225.0, 170.0, -135.0),  # Right eye right corner
            (-150.0, -150.0, -125.0),  # Left Mouth corner
            (150.0, -150.0, -125.0)  # Right mouth corner
            ])

        # Posición 2D de los keypoints faciales de dlib usados para la
estimación de la pose
        self.image_points = np.array([
            (landmarks.part(30).x, landmarks.part(30).y),  # Nariz
            (landmarks.part(8).x, landmarks.part(8).y),  # Mentón
            (landmarks.part(36).x, landmarks.part(36).y),  # Left eye left
corner
            (landmarks.part(45).x, landmarks.part(45).y),  # Right eye right
corner
            (landmarks.part(48).x, landmarks.part(48).y),  # Left Mouth corner
            (landmarks.part(54).x, landmarks.part(54).y)  # Right mouth corner
            ], dtype="double")

        # Calcular la pose de la cabeza
        (success, rvec, tvec) = cv2.solvePnP(self.model_points,
self.image_points,
                                             self.camera_matrix,
self.dist_coeffs, flags=cv2.SOLVEPNP_ITERATIVE)
        '''
        OpenCV Solve PnP calcula los vectores de rotación y traslación con
respecto al sistema de coordenadas de la cámara
        con los puntos de la imagen que tienen de referencia el modelo 3D de
la cabeza
        '''

        if success:  # Si funciona, calcular la posición de la cabeza

            # Refinar rvec y tvec
            rvec, tvec = cv2.solvePnPRefineVVS(
                self.model_points, self.image_points, self.camera_matrix,
self.dist_coeffs, rvec, tvec)

            # Punto de la nariz en la imagen
            nose = (int(self.image_points[0][0]),
int(self.image_points[0][1]))

            # Calcular la proyección de los 3 ejes desde la nariz
            (nose_end_point2D, _) = cv2.projectPoints(self.axis, rvec, tvec,
self.camera_matrix, self.dist_coeffs)

            # Calcular la matriz de rotación a partir del vector de rotación
con la fórmula Rodrigues
            Rmat = cv2.Rodrigues(rvec)[0]

            pitch, yaw, roll = rotationMatrixToEulerAngles(Rmat) * 180/np.pi

            """
            La función rotationMatrixToEulerAngles se utiliza para calcular
los ángulos de euler a partir de la
            matriz de rotación. Los ángulos se convierten a radianes.
```

```
            """

            # Para mostrar los datos de la función
            if self.verbose:
                self.frame = draw_pose_info(
                    self.frame, nose, nose_end_point2D, roll, pitch, yaw)
                # draws 3d axis from the nose and to the computed projection
points
                for point in self.image_points:
                    cv2.circle(self.frame, tuple(
                        point.ravel().astype(int)), 2, (0, 255, 255), -1)
                # Dibuja los 6 keypoints usados para la pose estimation

            return self.frame, roll, pitch, yaw

        else:
            return None, None, None, None
```

Anexo C 4. Módulo Attention_Scorer_Module.py

```
import time

class AttentionScorer:

    def __init__(self, capture_fps: int, ear_tresh, gaze_tresh, mar_thresh,
perclos_tresh=0.20, ear_time_tresh=4.0, pitch_tresh=165,
                 yaw_tresh=30, gaze_time_tresh=4.0, roll_tresh=20,
pose_time_tresh=4.0, verbose=False, mar_time_tresh=3.0):
        """
        La clase Attention Scorer contiene métodos de estimación del EAR,
Gaze_Score, PERCLOS y Head Pose a través del tiempo,
        con los thresholds dados (tanto de tiempo como de valores umbrales)

        Parámetros
        ----------
        capture_fps: int
            FPS considerados

        ear_tresh: float or int
            El valor de EAR threshold (si el EAR es menor que este valor, los
ojos son considerados como cerrados)

        mar_thresh: float or int
            El valor de MAR threshold (si el MAR es mayor que este valor, se
considera que está bastante abierta la boca)

        gaze_tresh: float or int
            El valor de Gaze threshold (si el Gaze Score es mayor que este
valor, la mirada se considera no centrada)

        perclos_tresh: float (ranges from 0 to 1)
            El PERCLOS threshold indica el máximo tiempo permitido en 60
segundos para tener los ojos cerrados sin
            considerar que hay somnolencia
            (suele utilizarse el 0.2, que sería el 20% de 1 minuto)


        pitch_tresh: int
            Threshold para el ángulo pitch para considerar a una persona
distraída
            (por defecto son 35 grados desde la posición central)

        yaw_tresh: int
            Threshold para el ángulo yaw para considerar a una persona
distraída (no teniendo la cabeza orientada hacia el frente)
```

```
            (por defecto son 30 grados desde la posición central de la cabeza)

        roll_tresh: int
            Threshold para el ángulo roll para considerar a una persona
distraída
            (por defecto es None: no se considera


        pose_time_tresh: float or int
            Tiempo máximo permitido para poses de la cabeza distraídas
consecutivas
            (por defecto son 4.0 segundos)

        ear_time_tresh: float or int
            Máximo tiempo permitido para tener los ojos cerrados de manera
consecutiva
            (por defecto está a 4.0 segundos)

        mar_time_tresh: float or int
            Máximo tiempo permitido para tener la boca abierta de manera
consecutiva

        gaze_time_tresh: float or int
            Máximo tiempo permitido para tener la mirada no centrada


        verbose: bool
            Si está a True, escribir información adicional sobre las
puntuaciones


        Métodos
        ----------

        - eval_scores: usado para evaluar el estado del conductor
        - get_PERCLOS: específicamente usado para evaluar la somnolencia del
conductor
        """

        self.tiempo_t = 0

        self.fps = capture_fps
        self.delta_time_frame = (1.0 / capture_fps)  # tiempo estimado de un
frame
        self.prev_time = 0  # variable auxiliar para la función de estimación
del PERCLOS

        self.perclos_time_period = 60 # tiempo por defecto del PERCLOS (60
segundos)
        self.perclos_tresh = perclos_tresh

        # los tresholds de tiempo
        self.ear_tresh = ear_tresh
        self.ear_act_tresh = ear_time_tresh / self.delta_time_frame
        self.ear_counter = 0
        self.eye_closure_counter = 0

        self.gaze_tresh = gaze_tresh
        self.gaze_act_tresh = gaze_time_tresh / self.delta_time_frame
        self.gaze_counter = 0

        self.roll_tresh = roll_tresh
        self.pitch_tresh = pitch_tresh
        self.yaw_tresh = yaw_tresh
        self.pose_act_tresh = pose_time_tresh / self.delta_time_frame
        self.pose_counter = 0

        self.verbose = verbose
```

```
        self.mar_tresh = mar_thresh
        self.mar_act_tresh = mar_time_tresh / self.delta_time_frame
        self.mar_counter = 0

    def eval_scores(self, ear_score, gaze_score, head_roll, head_pitch,
head_yaw, mar_score):
        """
        :param ear_score: float
            EAR (Eye Aspect Ratio) score obtenido a partir de la apertura de
los ojos
        :param gaze_score: float
            Gaze Score obtenido del gaze del ojo
        :param head_roll: float
            Roll ángulo obtenido del head pose
        :param head_pitch: float
            Pitch ángulo obtenido del head pose
        :param head_yaw: float
            Yaw ángulo obtenido del head pose

        :return:
            Devuelve una tupla de valores booleanos que indican el estado del
conductor
            tuple: (asleep, looking_away, distracted, yawn)
        """
        # inicialización variables del estado del conductor
        asleep = False
        looking_away = False
        distracted = False
        yawn = False

        if self.ear_counter >= self.ear_act_tresh:  # comprobar si el EAR
acumulativo sobrepasa el threshold
            asleep = True

        if self.gaze_counter >= self.gaze_act_tresh:  # comprobar si el gaze
acumulativo sobrepasa el threshold
            looking_away = True

        if self.pose_counter >= self.pose_act_tresh:  # comprobar si el pose
acumulativo sobrepasa el threshold
            distracted = True

        if self.mar_counter >= self.mar_act_tresh:  # comprobar si el MAR
acumulativo sobrepasa el threshold
            yawn = True

        '''
        Los 3 bloques if que siguen están escritos de manera que cuando
tenemos una puntuación que está por encima de su umbral de valor,
        un contador de puntuación respectivo (ear counter, gaze counter, pose
counter) se incrementa y puede alcanzar un máximo dado
        con el tiempo.
        Cuando una puntuación no supera un umbral, se disminuye y puede llegar
a un mínimo de cero.

        Ejemplo:

        Si la puntuación del oído del ojo del conductor supera el umbral para
un ÚNICO cuadro, el contador de oídos aumenta.
        Si se supera la puntuación de oído del ojo en varios fotogramas,
ear_counter aumentará y alcanzará
        un máximo dado, entonces no aumentará, pero la variable "dormido" se
establecerá en Verdadero.
        Cuando ear_score no supera el umbral, ear_counter disminuye. Si hay
varios marcos
        donde la puntuación no supera el umbral, ear_counter puede alcanzar el
mínimo de cero
```

```
        De esta forma, tenemos una puntuación acumulada para cada una de las
características controladas (OÍDO, MIRADA y POSE DE LA CABEZA).
        Si se alcanza una puntuación alta para un contador acumulativo, esta
función conservará su valor y necesitará un
        poco de "tiempo de enfriamiento" para volver a cero
        '''

        if (ear_score is not None) and (ear_score <= self.ear_tresh):
            if not asleep:
                self.ear_counter += 1
        elif self.ear_counter > 0:
            self.ear_counter -= 1

        if (gaze_score is not None) and (gaze_score >= self.gaze_tresh):
            if not looking_away:
                self.gaze_counter += 1
        elif self.gaze_counter > 0:
            self.gaze_counter -= 1

        if ((head_pitch is not None and ((head_pitch < -self.pitch_tresh) or
((0 < head_pitch) and (head_pitch < self.pitch_tresh)))) or (
            head_roll is not None and abs(head_roll) > self.roll_tresh) or (
            head_yaw is not None and abs(head_yaw) > self.yaw_tresh)):
            if not distracted:
                self.pose_counter += 1
                None
        elif self.pose_counter > 0:
            self.pose_counter -= 1

        if (mar_score is not None) and (mar_score >= self.mar_tresh):
            if not yawn:
                self.mar_counter += 1
        elif self.mar_counter > 0:
            self.mar_counter -= 1

        if self.verbose:
            print(
                f"ear counter:{self.ear_counter}/{self.ear_act_tresh}\ngaze
counter:{self.gaze_counter}/{self.gaze_act_tresh}\npose
counter:{self.pose_counter}/{self.pose_act_tresh}\nmar
counter:{self.mar_counter}/{self.mar_act_tresh}")
            print(
                f"eye closed:{asleep}\tlooking
away:{looking_away}\tdistracted:{distracted}\yawn:{yawn}")

        return asleep, looking_away, distracted, yawn

    def get_PERCLOS(self, ear_score):
        """

        :param ear_score: float
            EAR (Eye Aspect Ratio) score obtenido a partir de la apertura de
los ojos
        :return:
            tuple:(tired, perclos_score)

            tired:
                es un valor booleando que indica si el conductor está cansado
o no
            perclos_score:
                es un valor que indica el PERCLOS durante un minuto
                después de un minuto, se resetea el valor
        """

        delta = time.time() - self.prev_time  # calcular delta timer
        tired = False  # inicializar variable de estado del conductor
```

```
        # si el EAR es menor o igual que el threshold, incrementar
eye_closure_counter
        if (ear_score is not None) and (ear_score <= self.ear_tresh):
            self.eye_closure_counter += 1

        # calcular el closure_time (el tiempo en que los ojos está cerrado)
acumulativo
        closure_time = (self.eye_closure_counter * self.delta_time_frame)

        # calcular el PERCLOS en un periodo de tiempo dado
        perclos_score = ((closure_time) / self.perclos_time_period) * 100

        if perclos_score >= (self.perclos_tresh*100):  # si el PERCLOS es
mayor que el threshold, tired = True
            tired = True

        if self.verbose:
            print(
                f"Closure
Time:{closure_time}/{self.perclos_time_period}\nPERCLOS: {round(perclos_score,
3)}")

        if delta >= self.perclos_time_period:  # cada vez que finaliza el
periodo de tiempo establecido, se resetea el contador y el timer
            self.eye_closure_counter = 0
            self.prev_time = time.time()

        return tired, perclos_score
```

Anexo C 5. Módulo Initilization.py

```
import cv2
import time
import statistics

from picamera import PiCamera
from picamera.array import PiRGBArray

from Utils import get_face_area
from Eye_Mouth_Detector_Module import EyeMouthDetector as EyeMouthDet

from imutils import face_utils

class startup_routine():

    def __init__(self, Detector, Predictor, CAPTURE_SOURCE):
        """
        Esta función permite calcular los indicadores EAR y MAR de manera
personalizada

        Parámetros
        ----------
        Detector
        Predictor
        CAPTURE_SOURCE

        IMPORTANTE: en esta rutina, se ha de tener la boca abierta para poder
calcular bien el MAR
        """

        self.Detector = Detector
        self.Predictor = Predictor
        self.CAPTURE_SOURCE = CAPTURE_SOURCE

        self.ptime = 0
```

```
        self.frame_count = 0

        self.ears = []
        self.MARs = []

    def personalized_ear_and_mar_thresh(self):

        # Llamada al detector de ojos y bocas
        Eye_mouth_det = EyeMouthDet(show_processing=False)

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

        for frames in camera.capture_continuous(cap, format="bgr",
use_video_port = True):

            # Leer un frame
            frame = frames.array

            # Si el frame viene de la webcam, girarlo para que sea como un
espejo
            if self.CAPTURE_SOURCE == 0:
                frame = cv2.flip(frame, 2)

            # Calcular los FPS actuales y mostrarlos
            self.ctime = time.perf_counter()
            self.fps = 1.0 / float(self.ctime - self.ptime)
            self.ptime = self.ctime
            cv2.putText(frame, "FPS:" + str(round(self.fps, 0)), (10, 400),
cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 0), 1)

            # Transformar el frame en BGR a escala de grises
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            # Aplicar un filtro bilateral para reducir el ruido y resaltar los
detalles
            gray = cv2.bilateralFilter(gray, 5, 10, 10)

            # Uso el detector frontal
            faces = self.Detector(gray)

            if faces:  # Procesa el frame si encuentra al menos una cara

                # Solo cojo la cara más grande, que será la del conductor
                faces = sorted(faces, key=get_face_area, reverse=True)
                driver_face = faces[0]

                # Utilizo el preductor de los 68 puntos clave y lo muestro en
el frame
                landmarks = self.Predictor(gray, driver_face)
                Eye_mouth_det.show_eye_keypoints(color_frame=frame,
landmarks=landmarks)

                # Obtener la apertura de los ojos en posición de reposo
                self.ear = Eye_mouth_det.get_EAR(frame=gray,
landmarks=landmarks)
                self.ears.append(self.ear) # Añadir EAR de este frame al array

                # Obtener los puntos de la cara
                puntos_coordenadas = face_utils.shape_to_np(landmarks)

                # Se extraen las coordenadas internas de la boca
                mouth = puntos_coordenadas[60:68]

                # Mostrar el contorno de la boca
                contorno_boca = cv2.convexHull(mouth)
```

```
                cv2.drawContours(frame,[contorno_boca],-1, (255, 0, 0), 1)

                # Calcular MAR
                self.MAR = Eye_mouth_det.get_MAR(mouth)
                self.MARs.append(self.MAR) # Añadir MAR de este frame al array

            # Sumar uno al conteo de frames
            self.frame_count += 1

            cap.truncate(0)

            # Salir del bucle si se pulsa la tecla "q" o si pasan 50 frames
(unos 3-4 segundos, depende de los FPS)
            if (cv2.waitKey(1) & 0xFF == ord('q')) or self.frame_count > 20:

                # Cuando pasen x frames, se corta el vídeo y se calcula la
media
                mean_ear = statistics.mean(self.ears)
                mean_MAR = statistics.mean(self.MARs)

                # Calculamos el ear_thresh como 3/4 partes de la media, para
evitar tener en cuenta los parpadeos
                ear_thresh = mean_ear*(3/4)

                # Calculamos el MAR_thresh como la mitad de la media
estadística, ya que se tomará con la boca abierta, y así
                # consideraremos que a partir de la mitad de apertura (y
durante determinado tiempo), es un bostezo
                MAR_thresh = mean_MAR/2

                break

        camera.close()
        cv2.destroyAllWindows()

        return ear_thresh, MAR_thresh
```

Anexo C 6. Módulo Utils.py

```
import numpy as np
import cv2


def resize(frame, scale_percent):
    """
    Resize the image maintaining the aspect ratio
    :param frame: opencv image/frame
    :param scale_percent: int
        scale factor for resizing the image
    :return:
    resized: rescaled opencv image/frame
    """
    width = int(frame.shape[1] * scale_percent / 100)
    height = int(frame.shape[0] * scale_percent / 100)
    dim = (width, height)

    resized = cv2.resize(frame, dim, interpolation=cv2.INTER_LINEAR)
    return resized


def get_face_area(face):
    """
    Computes the area of the bounding box ROI of the face detected by the dlib
face detector
    It's used to sort the detected faces by the box area
```

```
    :param face: dlib bounding box of a detected face in faces
    :return: area of the face bounding box
    """
    return abs((face.left() - face.right()) * (face.bottom() - face.top()))


def show_keypoints(keypoints, frame):
    """
    Draw circles on the opencv frame over the face keypoints predicted by the
dlib predictor

    :param keypoints: dlib iterable 68 keypoints object
    :param frame: opencv frame
    :return: frame
        Returns the frame with all the 68 dlib face keypoints drawn
    """
    for n in range(0, 68):
        x = keypoints.part(n).x
        y = keypoints.part(n).y
        cv2.circle(frame, (x, y), 1, (0, 0, 255), -1)
        return frame


def midpoint(p1, p2):
    """
    Compute the midpoint between two dlib keypoints

    :param p1: dlib single keypoint
    :param p2: dlib single keypoint
    :return: array of x,y coordinated of the midpoint between p1 and p2
    """
    return np.array([int((p1.x + p2.x) / 2), int((p1.y + p2.y) / 2)])


def get_array_keypoints(landmarks, dtype="int", verbose: bool = False):
    """
    Converts all the iterable dlib 68 face keypoint in a numpy array of shape
68,2

    :param landmarks: dlib iterable 68 keypoints object
    :param dtype: dtype desired in output
    :param verbose: if set to True, prints array of keypoints (default is
False)
    :return: points_array
        Numpy array containing all the 68 keypoints (x,y) coordinates
        The shape is 68,2
    """
    points_array = np.zeros((68, 2), dtype=dtype)
    for i in range(0, 68):
        points_array[i] = (landmarks.part(i).x, landmarks.part(i).y)

    if verbose:
        print(points_array)

    return points_array


def isRotationMatrix(R):
    """
    Checks if a matrix is a rotation matrix
    :param R: np.array matrix of 3 by 3
    :return: True or False
        Return True if a matrix is a rotation matrix, False if not
    """
    Rt = np.transpose(R)
    shouldBeIdentity = np.dot(Rt, R)
    I = np.identity(3, dtype=R.dtype)
```

```
    n = np.linalg.norm(I - shouldBeIdentity)
    return n < 1e-6


def rotationMatrixToEulerAngles(R):
    """
    Computes the Tait-Bryan Euler angles from a Rotation Matrix.
    Also checks if there is a gymbal lock and eventually use an alternative
formula
    :param R: np.array
        3 x 3 Rotation matrix
    :return: (pitch, yaw, roll) tuple of float numbers
        Euler angles in radians
    """
    # Calculates Tait-Bryan Euler angles from a Rotation Matrix
    assert (isRotationMatrix(R))  # check if it's a Rmat

    sy = np.sqrt(R[0, 0] * R[0, 0] + R[1, 0] * R[1, 0])
    singular = sy < 1e-6

    if not singular:  # check if it's a gymbal lock situation
        x = np.arctan2(R[2, 1], R[2, 2])
        y = np.arctan2(-R[2, 0], sy)
        z = np.arctan2(R[1, 0], R[0, 0])

    else:  # if in gymbal lock, use different formula for yaw, pitch roll
        x = np.arctan2(-R[1, 2], R[1, 1])
        y = np.arctan2(-R[2, 0], sy)
        z = 0

    return np.array([x, y, z])


def draw_pose_info(frame, img_point, point_proj, roll=None, pitch=None,
yaw=None):
    """
    Draw 3d orthogonal axis given a frame, a point in the frame, the
projection point array.
    Also prints the information about the roll, pitch and yaw if passed

    :param frame: opencv image/frame
    :param img_point: tuple
        x,y position in the image/frame for the 3d axis for the projection
    :param point_proj: np.array
        Projected point along 3 axis obtained from the cv2.projectPoints
function
    :param roll: float, optional
    :param pitch: float, optional
    :param yaw: float, optional
    :return: frame: opencv image/frame
        Frame with 3d axis drawn and, optionally, the roll,pitch and yaw
values drawn
    """
    frame = cv2.line(frame, img_point, tuple(
        point_proj[0].ravel().astype(int)), (255, 0, 0), 3)
    frame = cv2.line(frame, img_point, tuple(
        point_proj[1].ravel().astype(int)), (0, 255, 0), 3)
    frame = cv2.line(frame, img_point, tuple(
        point_proj[2].ravel().astype(int)), (0, 0, 255), 3)

    if roll is not None and pitch is not None and yaw is not None:
        cv2.putText(frame, "Roll:" + str(round(roll, 3)), (500, 50),
                    cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 0), 1, cv2.LINE_AA)
        cv2.putText(frame, "Pitch:" + str(round(pitch, 3)), (500, 70),
                    cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 0), 1, cv2.LINE_AA)
        cv2.putText(frame, "Yaw:" + str(round(yaw, 3)), (500, 90),
                    cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 0), 1, cv2.LINE_AA)
return frame
```

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA DETECTAR Y ALERTAR DE LA SOMNOLENCIA DURANTE LA CONDUCCIÓN MEDIANTE LA MONITORIZACIÓN DE LA CABEZA Y EL ROSTRO.

DOCUMENTO II: PRESUPUESTO

ÍNDICE DEL PRESUPUESTO

1. PRESUPUESTO

Se presenta a continuación el desglose de los diferentes costes derivados de este proyecto para realizar así una valoración económica de este.

Los costes son de diferente tipo: coste computacional por el tiempo dedicado al procesamiento de los datos, coste de recursos humanos por el tiempo dedicado al desarrollo del algoritmo, planificación y realización de los ensayos, y coste material, referido a los recursos físicos requeridos para el desarrollo del proyecto.

1.1. CUADRO DE PRECIOS

1.1.1. Costes de recursos humanos

En este apartado se muestran las tablas de precios relativos a los costes de recursos humanos y de materiales incurridos en este TFG.

CUADRO DE PRECIOS BÁSICOS: RECURSOS HUMANOS			
Código	**Unidad**	**Descripción**	**Precio (€)**
RH1	h	Doctor Ingeniero Industrial (Tutor del proyecto)	50
RH2	h	Técnico de laboratorio de electrónica	30
RH3	h	Estudiante aspirante a Ingeniero en Electrónica Industrial y Automática	20

El Doctor Ingeniero Industrial ha tutorizado este trabajo para su buen desarrollo y el Técnico de Laboratorio de Electrónica ha realizado tareas de ayuda y supervisión para el montaje y la reparación de elementos de hardware necesarios para llevar a cabo los ensayos. Por otra parte, el estudiante ha realizado el desarrollo de este proyecto, pasando por la revisión bibliográfica y de referencias pertinente, la implementación de mejoras en la programación del algoritmo heredado, el montaje del sistema, la realización de los ensayos y su posterior análisis y la redacción de este documento.

1.1.2. Costes de recursos materiales

En este apartado se incluyen los costes relacionados con los equipos utilizados en este proyecto. Sin embargo, se marca una diferencia entre aquellos recursos destinados a este proyecto y aquellos que se han utilizado para llevarlo a cabo, pero también están destinados a otros usos.

En cuanto al software utilizado para el desarrollo del algoritmo, se trata de programas de software libre, lo que significa que son gratuitos. Estos son Visual Studio Code para el desarrollo en el ordenador portátil y Thonny para el desarrollo en la Raspberry. También se han utilizado aplicaciones como VNC Viewer y Putty para el acceso remoto a la Raspberry cuyo coste es nulo por ser también gratuitas.

En el caso del software utilizado para otras labores como las herramientas para la elaboración de tablas y la redacción de este documento, son herramientas que se destinan también a otros usos, por lo que se calcula el coste de amortización. Sucede lo mismo con otros elementos materiales como son el ordenador portátil y el multímetro empleados. Además, se calcula el coste energético y el coste relativo al combustible durante la realización de los ensayos.

- Los costes de amortización del ordenador portátil se obtienen del siguiente cálculo:

  Teniendo en cuenta que el coste del ordenador portátil ASUS VivoBook de 16 GB de RAM, con un procesador AMD Ryzen 7 4700U es de 799 €, con un periodo de amortización de 10 años. Considerando que el año 2022 tiene 251 días laborables con jornadas de 8 h, los costes de amortización son los siguientes.

$$\frac{799\ €}{10\ años} \cdot \frac{1\ año}{251\ días\ laborables} \cdot \frac{1\ día}{8\ h\ laborables} = 0{,}039\ €/h \quad (1)$$

- Los costes de amortización de la licencia de Microsoft Office 365 se obtienen del siguiente cálculo:

  Teniendo en cuenta que la licencia, según la página oficial, es de 7 €/mes y considerando que un mes tiene 20 días laborables con jornadas de 8 h, los costes de amortización son los siguientes.

$$\frac{7\ €}{1\ mes} \cdot \frac{1\ mes}{20\ días\ laborables} \cdot \frac{1\ día}{8\ h\ laborables} = 0{,}044\ €/h \quad (2)$$

- Los costes de amortización del multímetro se obtienen del siguiente cálculo:

  Teniendo en cuenta que el precio del multímetro es de 14,99 € y suponiendo que tiene una vida útil aproximada de 7 años con un uso habitual y considerando que el año 2022 tiene 251 días laborables con jornadas de 8 h, los costes de amortización son los siguientes.

$$\frac{14{,}99\ €}{7\ años} \cdot \frac{1\ año}{251\ días\ laborables} \cdot \frac{1\ día}{8\ h\ laborables} = 0{,}001\ €/h \quad (3)$$

- El coste energético:

  Considerando que la Raspberry Pi tienen un consumo de 600 mA a 5 V, es decir, $P = V \cdot I = 5 \cdot 600 \cdot 10^{-3} = 3\ W$, el consumo del ordenador portátil según las especificaciones de su cargador oficial es de 45 W y el precio medio de la electricidad es de 0,2913 €/kWh:

$$(3\ W + 45\ W) \cdot \frac{0{,}2913\ €}{kWh} = 48\ W \cdot \frac{0{,}2913\ €}{10^3 \cdot Wh} = 0{,}14\ €/h \quad (4)$$

- El coste de combustible:

  Considerando que el precio del combustible, que en este caso es gasolina sin plomo 95, es actualmente 1,58 €/l, que el consumo medio del vehículo es de 5,5 l/100 km y que el total de los kilómetros recorridos entre todos los ensayos, que los ensayos han tenido una media de 15 km/ensayo y que cada ensayo ha tenido una duración de 20 minutos:

$$\frac{1{,}58\ €}{l} \cdot \frac{5{,}5\ l}{100\ km} \cdot \frac{15\ km}{1\ ensayo} \cdot \frac{1\ ensayo}{20\ \text{min}} \cdot \frac{60\ min}{1\ h} = 3{,}91\ €/h \qquad (5)$$

CUADRO DE PRECIOS BÁSICOS: RECURSOS MATERIALES			
Código	**Unidad**	**Descripción**	**Precio (€)**
RM1	h	Ordenador portátil ASUS VivoBook	0,039
RM2	h	Microsoft Office 365	0,044
RM3	h	Multímetro	0,001
RM4	h	Coste energético del ordenador portátil y la Raspberry	0,140
RM5	h	Coste de combustible	3,910
RM6	ud	Coste Raspberry Pi 4 Modelo B 8 GB de RAM	85,000
RM7	ud	Coste Pi Camera NoIR v2	27,000
RM8	ud	Coste cable de imagen micro-HDMI a HDMI	9,000
RM9	ud	Coste cargador Raspberry Pi 15,3 W	9,000
RM10	ud	Coste tarjeta micro SDHC 32 GB 100 MB/s	6,900
RM11	ud	Coste cable de alimentación USB a USB tipo C	8,000
RM12	ud	Coste cargador para mechero del coche 12 V a 5 V	10,000
RM13	ud	Coste lámpara de LED infrarrojos	7,000
RM14	ud	Kit ventilador Raspberry Pi	7,000

1.2. UNIDADES DE OBRA

En este apartado se expone el precio unitario de cada unidad de obra a través de un cuadro de precios descompuesto correspondiente.

Se ha considerado el 3 % del valor de cada unidad de obra en concepto de coste directos para intentar cuantificar justamente el gasto de algunos medios utilizados que no son exclusivos de este TFG. Se ha considerado tres unidades de obra distintas.

❖ Unidad de obra UO-01 referida a las tareas de desarrollo y programación del algoritmo.

Código	Unidad	Descripción	Rendimiento	Precio	Importe (€)
UO-01	h	Tareas de desarrollo y programación del algoritmo			
RH3	h	Estudiante aspirante a Ingeniero en Electrónica Industrial y Automática	1,000	20,000	20,000
RM1	h	Ordenador portátil ASUS VivoBook	1,000	0,039	0,039
RM2	h	Microsoft Office 365	0,600	0,044	0,026
RM4	h	Coste energético del ordenador portátil y la Raspberry	1,000	0,140	0,140
%	%	Costes Directos Complementarios	0,030	20,206	0,606
		Costes Directos			20,811
	%	Costes Indirectos	0,010	20,811	0,208
		Coste Total			**21,020**

❖ La unidad de obra UO-02, respecto a los costes considerados para la fase de ensayos:

Código	Unidad	Descripción	Rendimiento	Precio	Importe (€)
UO-02	h	Realización de ensayos			
RH3	h	Estudiante aspirante a Ingeniero en Electrónica Industrial y Automática	1,000	20,000	20,000
RH1	h	Doctor Ingeniero Industrial (Tutor del proyecto)	0,100	50,000	5,000
RH2	h	Técnico de laboratorio de electrónica	0,100	30,000	3,000
RM1	h	Ordenador portátil ASUS VivoBook	1,000	0,039	0,039
RM2	h	Microsoft Office 365	0,600	0,044	0,026
RM3	h	Multímetro	0,400	0,001	0,004
RM4	h	Coste energético del ordenador portátil y la Raspberry	1,000	0,140	0,140
RM5	h	Coste de combustible	1,000	3,910	3,910
%	%	Costes Directos Complementarios	0,030	32,115	0,963
		Costes Directos			33,082
	%	Costes Indirectos	0,010	33,082	0,331
		Coste Total			**33,410**

- La unidad de obra UO-03, referida a los costes de los equipos y herramientas adquiridos para la formación del sistema:

Código	Unidad	Descripción	Rendimiento	Precio	Importe (€)
UO-03	ud	Equipos y herramientas para la confección del sistema			
RM6	ud	Coste Raspberry Pi 4 Modelo B 8 GB de RAM	1,000	85,000	85,000
RM7	ud	Coste Pi Camera NoIR v2	1,000	27,000	27,000
RM8	ud	Coste cable de imagen micro-HDMI a HDMI	1,000	9,000	9,000
RM9	ud	Coste cargador Raspberry Pi 15,3 W	1,000	9,000	9,000
RM10	ud	Coste tarjeta micro SDHC 32 GB 100 MB/s	1,000	6,900	6,900
RM11	ud	Coste cable de alimentación USB a USB tipo C	1,000	8,000	8,000
RM12	ud	Coste cargador para mechero del coche 12 V a 5 V	1,000	10,000	10,000
RM13	ud	Coste lámpara de LED infrarrojos	1,000	7,000	7,000
RM14	ud	Kit ventilador Raspberry Pi	1,000	7,000	7,000
%	%	Costes Directos Complementarios	0,030	168,900	5,067
		Costes Directos			173,967
	%	Costes Indirectos	0,010	173,967	1,740
		Coste Total			**175,707**

1.3. MEDICIONES

Se estima que la duración del Trabajo Final de Grado correspondiente a 12 ECTS es de 300 horas. De estas, se han dedicado cerca de 200 al desarrollo y programación del algoritmo, considerando que en desarrollo se incluye también todo el proceso de redacción del proyecto. Las 100 horas restantes se han dedicado a la preparación y cursado de los ensayos, además de su posterior análisis.

1.4. RESUMEN PRESUPUESTO

El presupuesto final es el resultado de agrupar los costes totales de las diferentes unidades de obra, tal y como se muestra a continuación:

Código	Unidad	Descripción	Rendimiento	Precio	Importe (€)
UO-01	h	Tareas de desarrollo y programación del algoritmo	200	21,020	4204,019
UO-02	h	Realización de ensayos	100	33,410	3340,965
UO-03	ud	Equipos y herramientas para la confección del sistema	1	175,707	175,707
		Presupuesto de Ejecución Material		.....	7720,690
		Gastos Generales 13 %		.....	1003,690
		Beneficio Industrial 6 %		.....	463,241
		Presupuesto de Ejecución por Contrata		.....	9187,622
		IVA 21 %		.....	1929,401
		Presupuesto base de licitación		.....	11117,022

Tras contabilizar los gastos generales y el beneficio industrial, además de aplicar el IVA que en 2023 se sitúa en un 21 %, el presupuesto final de este proyecto asciende a la cifra de ONCE MIL CIENTO DIECISIETE EUROS CON DOS CÉNTIMOS.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA DETECTAR Y ALERTAR DE LA SOMNOLENCIA DURANTE LA CONDUCCIÓN MEDIANTE LA MONITORIZACIÓN DE LA CABEZA Y EL ROSTRO.

DOCUMENTO III: PLIEGO DE CONDICIONES

ÍNDICE DEL PLIEGO DE CONDICIONES

1. OBJETO

El pliego de condiciones es el documento técnico que describe de manera detallada los requisitos y especificaciones que deben cumplirse para llevar a cabo este Trabajo Final de Grado. Constituye un elemento esencial en la planificación y ejecución de proyectos al establecer las pautas y expectativas que deben seguirse a lo largo de todo el proceso.

El proyecto se ha realizado en el ámbito de desarrollo de software y la realización de ensayos en condiciones reales en automóviles. Por lo tanto, este documento incluye las condiciones generales de trabajo requeridas para el proyecto, así como las condiciones necesarias para llevar a cabo los ensayos.

2. CONDICIONES Y NORMAS GENERALES

En el presente apartado se describen las condiciones generales que rigen las obligaciones y derechos de las personas involucradas. Se deben seguir unas pautas específicas de seguridad tanto en el desarrollo del proyecto como en la realización de los ensayos.

2.1. CONDICIONES DEL LUGAR DE TRABAJO

Debido a que este proyecto posee un fuerte componente analítico y de desarrollo mediante software informático, se detallan a continuación las condiciones mínimas de seguridad y salud por parte del personal que trabaja en el desarrollo.

2.1.1. Ergonomía

La tarea de desarrollo de software implica mantener posturas estáticas en periodos de tiempo prolongados, por lo que el puesto en el que se vaya a realizar debe cumplir con ciertas condiciones, con el fin de evitar problemas de postura.

Un puesto de trabajo de calidad consta de los siguientes puntos:

- Asiento: debe incorporar un sistema que permita ajustar debidamente la altura y la inclinación, para así permitir una vista centrada a la pantalla del ordenador.

- Mesa y espacio de trabajo: las dimensiones deben ser suficientes para que contenga una pantalla, los accesorios requeridos, así como la documentación que se vaya a utilizar. El color del entorno debe evitar los reflejos.

- Posturas: las piernas deben formar un ángulo recto y los brazos deben quedar a la altura del espacio de trabajo. La columna vertebral debe permanecer lo más recta posible, utilizando de apoyo el respaldo del asiento.

- Pantalla: la distancia mínima recomendada desde los ojos hasta la pantalla deberá ser como mínimo de 40 cm y no superar los 75 cm. Además, quedará dentro de un ángulo de 120^{o} del campo de visión del sujeto.

- Teclado: puede ser un accesorio del ordenador que podrá colocarse en cualquier lugar y posición de la mesa, permitiendo una buena postura del usuario. En su defecto, podrá formar parte de un ordenador portátil.

2.1.2. Iluminación

La iluminación en el área de trabajo es un aspecto fundamental para la salud del trabajador. Esta puede ser artificial o natural, siendo la natural más recomendable, pero evitando la incisión directa sobre el sujeto. En conjunto, se debe garantizar un nivel de iluminación adecuado a la luminancia de la pantalla y el entorno, así como a las necesidades visuales del usuario. Se deben evitar los contrastes altos y deslumbramientos, por lo que la pantalla debe ubicarse en dirección perpendicular a las ventanas.

2.1.3. Ruido

Se debe garantizar que los niveles de ruido no superen los 85 dB durante una jornada de 8 horas. En caso de que no se pueda cumplir con este requisito, será necesario aplicar las medidas de protección correspondiente en el emisor, medio y receptor con el objetivo de minimizarlo, tal y como establece el Real Decreto 286/2006, de 10 de marzo, sobre la protección de salud y seguridad de los trabajadores contra los riesgos relacionados con la exposición del ruido.

2.2. CONDICIONES DE LOS ENSAYOS

En lo que respecta a los ensayos, se permite la instalación de un sistema de detección de somnolencia en los vehículos, siempre que no interfiera con la seguridad del vehículo. Por lo tanto, la colocación del hardware para la realización de los ensayos debe hacerse en un lugar que no obstaculice la correcta visualización de la carretera al conductor ni cause distracciones. El montaje se describe en el apartado 4, referido al montaje en el vehículo.

2.3. OBLIGACIONES Y DERECHOS DE LOS SUJETOS DE LOS ENSAYOS

Se deben tomar medidas para garantizar la seguridad de los conductores voluntarios durante los ensayos, incluyendo la instalación segura del hardware utilizado, que en este caso incluye una Raspberry Pi, una cámara Pi NoIR y una lámpara IR.

Los sujetos de los ensayos deben permanecer en el interior del vehículo durante todo el ensayo, con las medidas de seguridad requeridas por el vehículo y prestando atención al entorno de la carretera en todo momento. De acuerdo con el Reglamento General de Circulación elaborado por la DGT, donde se establecen los derechos y obligaciones de cada uno de los usuarios que interactúan en las vías públicas, el sujeto no manipulará ningún elemento del hardware de este sistema durante la conducción.

3. CONDICIONES PARTICULARES

3.1. ESPECIFICACIONES TÉCNICAS

Este trabajo contiene una fase de desarrollo que requiere equipos informáticos y determinado software de desarrollo y hardware para la fase de ensayos, que se detallan en este apartado.

3.1.1. EQUIPOS INFORMÁTICOS

El equipo informático empleado para la realización de este proyecto debe ser lo suficientemente potente para ejecutar el software requerido. Las características del equipo utilizado son las de un equipo informático de gama media.

- Sistema operative: Windows 11 Home 22H2
- Procesador: AMD Ryzen 7 4700U con Radeon Graphics de 2 GHz
- Velocidad de reloj: 2.9 GHz
- Memoria RAM: 16 GB
- Disco duro: 475 GB

3.1.2. SOFTWARE

Para la realización de este proyecto se han utilizado principalmente las siguientes herramientas:

- Python como lenguaje de desarrollo.
- Visual Studio Code como entorno de desarrollo del algoritmo.
- Microsoft Excel como herramienta de cálculo para extraer conclusiones de los archivos de datos de cada ensayo.
- Microsoft PowerPoint como herramienta ilustrativa para llevar a cabo la presentación del desarrollo del trabajo.
- Microsoft Word como herramienta de redacción de texto.

El paquete de Microsoft utilizado es el correspondiente a la versión de educación proporcionada por la Universitat Politècnica de València.

3.1.3. HARDWARE

El hardware utilizado para los ensayos ha sido el siguiente:

- Raspberry Pi 4 Modelo B: 8 GB de RAM, puertos USB 3.0 y USB-C, Bluetooth 5.0 BLE, conectividad Wi-Fi, CPU con 1,5 GHz de frecuencia de reloj, conector GPIO de 40 pines y conectores CSI, DSI y HDMI.
- Pi NoIR Camera v2, que no posee filtro infrarrojo.
- Lámpara IR de longitud de onda 850 nm, consta de 24 LED que pueden iluminar a 20 m de distancia. La alimentación es de 12 V.

El control de calidad de los elementos hardware de manera independiente vendrá realizado por el proveedor, que certificará el correcto funcionamiento.

4. MONTAJE EN EL VEHÍCULO

Tal y como se ha comentado en las condiciones de los ensayos, el montaje en el vehículo deberá garantizar que la actividad de la conducción se desarrolle de manera cómoda y segura para el conductor. Para ello, se detalla a continuación el correcto montaje de los elementos de hardware utilizados para los ensayos del sistema.

4.1. MONTAJE DE LA RASPBERRY PI

La Raspberry Pi se colocará anclada en la parte posterior de la visera parasol del vehículo, de manera que al doblar la visera para mantenerla lo más alta posible, quede cubierta. La Raspberry vendrá alimentada por un cable tipo USB-C, insertando el conector C en el puerto de alimentación de la Raspberry y el conector USB en un adaptador tipo cargador de mechero conversor de 12 V a 5 V. Este adaptador irá conectado a la toma de mechero del vehículo. El cable de alimentación se fijará a la estructura del vehículo por la parte superior de este pasando por el espejo interior para que quede disimulado a la vista.

4.2. MONTAJE DE LA CÁMARA

La cámara se anclará justo debajo del espejo que se encuentra en la visera parasol en el lugar más centrado de esta, procurando situarla en la paralela a la ubicación del torso y cabeza del conductor. Debe ser así para garantizar la captura completa de la cabeza del sujeto de la manera más frontal posible.

4.3. MONTAJE DE LA LÁMPARA IR

La lámpara IR se sujetará en la parte superior del espejo de la visera parasol, de manera que su incisión sea directa sobre el sujeto del ensayo. La alimentación de la lámpara vendrá dada por alguna de las luces de cortesía del vehículo que posea 12 V, respetando la polaridad.

5. PRUEBA DE SERVICIO

Después del montaje en el vehículo del sistema hardware se procederá a validar el funcionamiento de este ejecutando el algoritmo diseñado. Para ello, se seguirán los pasos siguientes:

- Se conectará la alimentación de Raspberry y se comprobará visualmente que el LED rojo PWR de la alimentación está encendido permanentemente, lo que significará que la potencia recibida es adecuada.
- Se conectará la alimentación de la lámpara IR y se comprobará de manera visual que existe un ligero resplandor de color rojizo en los LED de lámpara, lo que indicará que está funcionando correctamente.

- Para saber si el algoritmo se está ejecutando con éxito, bastará con observar el LED verde ACT, y si está parpadeando constantemente, significará que la aplicación se está ejecutando en el sistema.

DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA DETECTAR Y ALERTAR DE LA SOMNOLENCIA DURANTE LA CONDUCCIÓN MEDIANTE LA MONITORIZACIÓN DE LA CABEZA Y EL ROSTRO.

DOCUMENTO IV: PLANOS

Este Trabajo Final de Grado se centra exclusivamente en el desarrollo de algoritmia, sin involucrar ninguna actividad de diseño o construcción física. Dado que la investigación se basa en principios teóricos y metodológicos relacionados con la algoritmia, la inclusión de planos no aportaría información relevante ni contribuiría al alcance y los objetivos de este proyecto, por lo que se ha considerado que no procede.

En su lugar, este trabajo se ha enfocado en la descripción detallada de los algoritmos desarrollados, su análisis y la implementación, además de la realización de pruebas de funcionamiento y ensayos, lo cual es esencial para entender esta área específica de estudio.