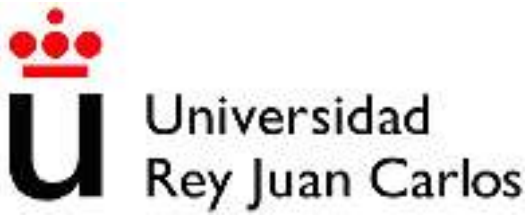


ESCUELA DE INGENIERÍA DE FUENLABRADA

GRADO EN INGENIERÍA DE ROBÓTICA SOFTWARE

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN DE DISTRACCIONES AL VOLANTE MEDIANTE INTELIGENCIA ARTIFICIAL

Autora: Arantxa García Benítez
Tutor: Roberto Calvo Palomino

Curso académico 2024 / 2025

Documento de **Arantxa García Benítez**.

A mi hermano Marcos;
que me ha acompañado en cada uno de mis logros.

Agradecimientos

A mis padres, Mikel y Amalia, porque siempre se han cerrado puertas para abrir las mías y porque sin su esfuerzo, este esfuerzo no habría sido posible.

A mi tutor Roberto Calvo, por estar siempre disponible, por su paciencia infinita y por enseñarme mucho más que los aspectos técnicos.

A mis amigos; los del colegio, los del instituto y los de la universidad. Gracias por ayudarme a descubrir quién soy y por acompañarme en este camino.

A Joan, por aguantarme como nadie, por entenderme incluso cuando no sé explicarme, y por alegrarse de mis victorias incluso más que yo.

Y a mi yaya Amalia. Si hay algo que agradezco de esta carrera, es el tiempo que he tenido contigo.

Madrid, 9 de julio de 2025
Arantxa García Benítez

Resumen

La *inteligencia artificial* (IA) está cada vez más presente en nuestra vida cotidiana, transformando tareas que antes parecían propias de la ciencia ficción en realidades tangibles y alcanzables. Entre las aplicaciones emergentes, en este *trabajo de fin de grado* (TFG) se destacan los denominados *Driver Monitoring Systems (sistemas de monitorización del conductor)* (DMS). Estos sistemas avanzados surgen como respuesta a uno de los principales problemas en seguridad vial: la distracción y la falta de atención al volante. La somnolencia, el uso de dispositivos móviles o no mantener la vista en la carretera son factores que incrementan considerablemente el riesgo de accidentes. Ante este escenario, resulta imprescindible contar con tecnologías que permitan supervisar de forma continua y precisa el comportamiento del conductor, con el objetivo de detectar precozmente cualquier signo de distracción y prevenir siniestros.

Por ello, en este proyecto se ha desarrollado un sistema de detección de distracciones al volante basado en IA, que utiliza únicamente dos cámaras estratégicamente ubicadas en el habitáculo. Las principales características incluyen un módulo para el análisis de acciones manuales, otro para la estimación de la dirección de la mirada y un tercero para la detección de distracciones físicas (como el uso de botellas o teléfonos). Además, se ha priorizado un diseño ligero a nivel computacional y con baja latencia, para facilitar su futura integración en sistemas empotrados.

Para su desarrollo se han empleado diversas herramientas y tecnologías avanzadas, como modelos de estimación de pose (Mediapipe), redes neuronales profundas (YOLO) y algoritmos de clasificación basados en árboles (Random Forest). Gracias a esta combinación, se ha implementado un sistema completo, capaz de analizar el comportamiento del conductor con baja latencia, evaluar su nivel de distracción y representar de forma clara y comprensible los resultados obtenidos.

Finalmente, se ha diseñado una representación gráfica cronológica que muestra detalladamente los niveles de distracción durante toda la sesión de conducción, permitiendo un análisis intuitivo y global. Esta visualización demuestra la efectividad y el correcto funcionamiento del sistema, que además destaca por su bajo consumo computacional, permitiendo su uso durante la conducción y posicionándose como una alternativa práctica y competitiva frente a métodos más complejos para su futura aplicación en entornos reales.

Con el sistema desarrollado se puede aportar seguridad a la conducción, ayudando a que los conductores mantengan la atención en la carretera y reduciendo significativamente el riesgo de accidentes. De este modo, se sientan las bases para futuras mejoras e integraciones en vehículos comerciales, contribuyendo al desarrollo de una movilidad más segura, inteligente y adaptada a las necesidades actuales y futuras de la sociedad.

Acrónimos

IA *inteligencia artificial*

GPU *Graphics Processing Unit (unidad de procesamiento gráfico)*

MOCAP *Motion Capture (captura de movimiento)*

CNN *Convolutional Neural Networks (redes neuronales convolucionales)*

ML *machine learning*

TFG *trabajo de fin de grado*

DMD *driving monitoring dataset*

RNN *Recurrent Neural Networks (redes neuronales recurrentes)*

URJC *Universidad Rey Juan Carlos*

DMS *Driver Monitoring Systems (sistemas de monitorización del conductor)*

Índice general

Índice de figuras

Listado de códigos

Índice de cuadros

Capítulo 1
Introducción

Si hay algo que define al ser humano, es su impulso constante por mejorar. A lo largo de la historia, nunca nos hemos conformado con lo que tenemos: si algo no existe, lo inventamos; y si ya existe, buscamos la forma de perfeccionarlo. Esta necesidad de avanzar ha dado lugar a descubrimientos que cambiaron el rumbo de nuestra sociedad, como la rueda, la imprenta, la electricidad o internet. Más recientemente, la inteligencia artificial se ha convertido en una herramienta clave para ampliar nuestras capacidades y afrontar nuevos desafíos. Gracias a estos avances, no solo ha cambiado nuestra forma de vivir, también la de movernos: del carro tirado por caballos pasamos al automóvil, y desde entonces no hemos dejado de hacerlo más rápido, eficiente y accesible. Ahora, la meta no es solo avanzar, sino hacerlo de forma segura. En ese camino, las nuevas tecnologías han abierto la puerta a soluciones capaces de cuidar de nosotros mientras conducimos. Sistemas capaces de observar, interpretar y reaccionar, que no solo entienden lo que ocurre fuera del vehículo, sino también lo que sucede dentro. Esta es la base de los sistemas de detección de distracciones del conductor, y el punto de partida de este trabajo.

Entre estas tecnologías, una de las más asentadas es la visión artificial. Históricamente, el ser humano ha buscado mecanismos para comprender y representar la realidad que lo rodea. Esta necesidad de interpretación ha sido clave para nuestra evolución y supervivencia, pues la capacidad de observar, analizar y tomar decisiones en función de lo que vemos ha determinado nuestra forma de interactuar con el entorno. Gracias a la vista, uno de nuestros sentidos más complejos, somos capaces de detectar peligros, reconocer patrones y adaptarnos al cambio. Con el avance de la tecnología, surgió el deseo de trasladar esta habilidad cognitiva a las máquinas, con el fin de automatizar tareas que antes dependían exclusivamente del juicio humano. En este contexto nace la visión artificial: una disciplina que busca dotar a los sistemas informáticos de la capacidad de interpretar imágenes y actuar en consecuencia, tal y como lo haría una persona. Por ejemplo, en el ámbito agrícola, antes era necesario revisar manualmente los cultivos para comprobar su estado y detectar posibles problemas, mientras que ahora estas tareas pueden realizarse de forma automatizada mediante sistemas de visión artificial, optimizando recursos y aumentando la precisión.

1.1. Visión Artificial

La visión artificial [13] es una rama de la inteligencia artificial que permite a los ordenadores interpretar y analizar imágenes digitales, vídeos y otras fuentes visuales

para extraer información significativa. Su propósito es imitar el funcionamiento del sistema visual humano, combinando sensores ópticos, algoritmos de procesamiento de datos y modelos de aprendizaje automático. A través de esta integración, las máquinas adquieren una forma de "visión" que les permite comprender su entorno, reconocer objetos o eventos, y tomar decisiones sin intervención humana directa. Entre sus múltiples aplicaciones se encuentran la detección de peatones para evitar atropellos, la asistencia al aparcamiento, el reconocimiento de señales de tráfico para ajustar la velocidad del vehículo, el análisis de imágenes médicas para apoyar diagnósticos clínicos, o el control de calidad en líneas de producción para identificar productos defectuosos o contaminados.

Los sistemas de visión artificial suelen estar compuestos por tres elementos fundamentales. En primer lugar, un sensor de imagen, como una cámara RGB, térmica o en profundidad, que captura las escenas del entorno. En segundo lugar, una unidad de procesamiento, que puede ser un ordenador tradicional, una *Graphics Processing Unit (unidad de procesamiento gráfico)* (GPU) de alto rendimiento o un sistema embebido, que ejecuta los algoritmos necesarios para el análisis visual. Por último, un software de análisis, basado en técnicas como el procesamiento digital de imágenes, redes neuronales convolucionales o aprendizaje profundo, que interpreta los datos capturados. El funcionamiento típico de estos sistemas sigue una secuencia estructurada: se adquiere la imagen, se mejora su calidad mediante preprocesamiento (como la eliminación de ruido o la conversión a escala de grises), se identifican regiones relevantes mediante detección y segmentación, y finalmente se extraen características que serán evaluadas para tomar decisiones. Además, en muchos casos, las imágenes se simplifican (por ejemplo, reduciéndolas a contornos o mapas de puntos clave) para que sean más fáciles de procesar computacionalmente y reducir la carga de cálculo. Esta secuencia de procesamiento permite, por ejemplo, detectar objetos en movimiento, clasificar componentes o identificar anomalías en tiempo real.

El uso de visión artificial aporta beneficios tangibles en términos de eficiencia, precisión y autonomía. Automatiza procesos visuales que serían costosos o imposibles de realizar manualmente a gran escala, asegura una mayor consistencia en tareas repetitivas, permite analizar grandes volúmenes de datos en muy poco tiempo, reduce la dependencia de supervisión humana continua y mejora la seguridad al intervenir en entornos peligrosos o inaccesibles para las personas.

Estas ventajas han impulsado su adopción en numerosos sectores, con aplicaciones cada vez más sofisticadas. Algunos ejemplos representativos incluyen:

- **Cadena de producción industrial:** los sistemas de inspección visual permiten identificar defectos microscópicos en líneas de producción de alta velocidad, comprobar tolerancias dimensionales con gran precisión y garantizar que cada pieza ensamblada cumpla con los estándares de calidad, todo ello sin detener la cadena de montaje. Podemos ver un ejemplo de esto en la figura 1.1a.

- **Automoción:** en el desarrollo de vehículos autónomos, la visión artificial es fundamental para percibir el entorno. A través de cámaras y sensores, el vehículo puede detectar peatones, interpretar señales de tráfico, seguir el trazado de los carriles o reaccionar ante obstáculos inesperados, lo que permite una conducción más segura y autónoma. Un caso ilustrativo se muestra en la figura 1.1b.

- **Medicina:** en el ámbito clínico, esta tecnología se aplica en el análisis automático de imágenes médicas (como radiografías, resonancias magnéticas o ecografías),

ayudando a los profesionales a detectar lesiones, tumores o anomalías de forma más rápida y precisa. También se usa en cirugías asistidas por computadora, donde la visión artificial guía al cirujano con información visual aumentada en tiempo real. Un buen ejemplo es el robot Da Vinci, visible en la figura 1.1c.

- **Agricultura:** mediante drones equipados con cámaras multiespectrales y algoritmos de visión, los agricultores pueden evaluar el estado de salud de sus cultivos, detectar plagas en etapas tempranas, estimar rendimientos e incluso automatizar tareas como la recolección de frutas en función de su madurez visual. Esto puede observarse en la figura 1.1d.

- **Seguridad:** los sistemas de videovigilancia inteligentes emplean visión artificial para reconocer rostros, detectar comportamientos anómalos o identificar matrículas de vehículos, permitiendo generar alertas en tiempo real sin necesidad de monitoreo humano constante. Esto es especialmente útil en aeropuertos, estaciones o infraestructuras críticas. Un ejemplo representativo aparece en la figura 1.1e.

- **Logística:** en almacenes y centros de distribución, esta tecnología permite clasificar automáticamente paquetes según su contenido o destino, leer códigos de barras con alta precisión incluso en condiciones desfavorables, y supervisar inventarios en tiempo real, optimizando los flujos de trabajo y reduciendo errores operativos. Podemos apreciar esta aplicación en la figura 1.1f.

- **Sector aeroespacial y exploración planetaria:** un ejemplo destacado del uso de visión artificial en el sector aeroespacial es el aterrizaje del rover Perseverance de la NASA en Marte en 2021. Durante la fase final de descenso, el sistema de navegación utilizó imágenes en tiempo real del terreno marciano para comparar con un mapa previamente almacenado, permitiendo identificar y evitar zonas peligrosas. Este caso se ilustra en la figura 1.1g.

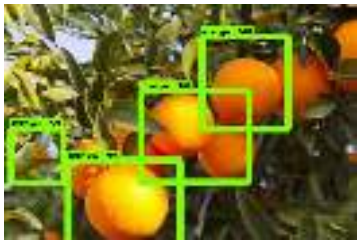
- **Análisis de movimientos:** el análisis automático de movimientos permite estudiar, interpretar y cuantificar de manera precisa los desplazamientos y gestos realizados por personas u objetos a partir de imágenes o secuencias de vídeo. Esta tecnología se apoya en técnicas avanzadas de visión artificial y aprendizaje automático para extraer información relevante sobre trayectorias, velocidades, patrones posturales y secuencias de acción. Gracias a ello, resulta posible detectar anomalías, optimizar procesos, evaluar el rendimiento o garantizar la seguridad en múltiples ámbitos, desde la industria y la robótica colaborativa hasta la supervisión de tareas manuales y la interacción hombre-máquina.

A medida que la visión artificial ha avanzado, sus aplicaciones han dejado de limitarse únicamente a la identificación de objetos o escenas. Hoy en día, estos sistemas son capaces de interpretar estructuras más complejas, como el movimiento humano o la disposición espacial del cuerpo. En este contexto, surgen los sistemas de detección de pose, una rama especializada de la visión artificial que se enfoca en identificar y seguir los puntos clave del cuerpo, como las articulaciones o extremidades, para comprender posturas, gestos o actividades en tiempo real.

(a) Control de calidad de piezas. Fuente: [14]

(b) Visión del Tesla. Fuente: [15]

(c) Robot Da Vinci. Fuente: [16]



(d) Selección de naranjas. Fuente: [17]

(e) Monitorización de cámaras. Fuente: [18]

(f) Control en almacén. Fuente: [19]

(g) Rover Perseverance. Fuente: [20]

(h) Análisis de movimiento. Fuente: [21]

Figura 1.1: Aplicaciones de visión artificial en distintos ámbitos.

1.2. Sistemas de detección de pose

Aunque no se trata de una tecnología reciente, la detección de pose sigue siendo poco utilizada fuera de ámbitos muy específicos debido a su elevado coste y la complejidad técnica que conlleva. Uno de los ejemplos más conocidos de esta tecnología es la *Motion Capture (captura de movimiento)* (MOCAP), una técnica ampliamente utilizada en cine, videojuegos y simuladores. Su objetivo es detectar y analizar los movimientos de personas reales para trasladarlos a modelos o personajes digitales en 3D, permitiendo así crear animaciones sumamente realistas que difícilmente podrían lograrse manualmente. A diferencia de los motion graphics, que son animaciones generadas por ordenador sin datos del mundo real, el MOCAP se basa en movimientos humanos auténticos, lo que le otorga una naturalidad y precisión fundamentales para escenas complejas de acción o interacción.

Existen diferentes métodos para llevar a cabo esta captura: por un lado, la captura óptica (como en la figura 1.2), en la que se utilizan cámaras combinadas con marcadores reflectantes o luces LED adheridas al cuerpo del actor; por otro, la captura inercial (se puede ver en la figura 1.3), que se basa en sensores colocados directamente sobre el

Figura 1.2: Captura óptica. Fuente: [1]

cuerpo para registrar los movimientos mediante acelerómetros y giróscopos. Ambas técnicas han demostrado una gran precisión, pero presentan limitaciones importantes: requieren infraestructuras específicas, equipamiento técnico costoso, y no son fácilmente escalables. La necesidad de utilizar trajes específicos limita su aplicabilidad en contextos cotidianos o con usuarios no especializados, lo que dificulta su adopción a gran escala. Todo ello, sumado a su elevado coste, contribuye a que sigan siendo tecnologías reservadas principalmente a sectores profesionales muy concretos.

Figura 1.3: Captura inercial. Fuente: [2]

En este contexto, y gracias a los avances en inteligencia artificial, han surgido soluciones mucho más accesibles: sistemas de detección de pose basados únicamente en cámaras convencionales, sin necesidad de trajes ni sensores. Estas soluciones aprovechan el poder de las redes neuronales, estructuras computacionales inspiradas en el cerebro humano, capaces de aprender patrones complejos a partir de grandes volúmenes de datos visuales.

Dentro de este campo, destacan las soluciones basadas en inteligencia artificial, especialmente diseñadas para procesar imágenes y vídeos. Estas tecnologías pueden detectar automáticamente características visuales relevantes, como contornos, texturas o puntos clave, sin necesidad de intervención manual. Gracias a su capacidad para identificar patrones complejos en diferentes niveles, se han convertido en herramientas esenciales para tareas de visión artificial avanzadas como la estimación de pose humana. Esta técnica genera una representación digital del esqueleto corporal a partir de imágenes o vídeos, identificando la ubicación de articulaciones y extremidades para comprender cómo se mueve una persona en tiempo real.

Las aplicaciones de la detección de pose son cada vez más amplias y transformadoras. En el ámbito deportivo, esta tecnología ha revolucionado el análisis del rendimiento y la técnica de los atletas, tanto profesionales como amateurs [22]. Mediante la extracción de puntos clave del cuerpo en tiempo real, es posible evaluar parámetros biomecánicos como los ángulos articulares, la alineación corporal o la simetría del movimiento. Esto permite ofrecer retroalimentación inmediata y objetiva sin necesidad de sensores físicos ni equipamiento especializado.

Un caso concreto sería el análisis de sentadillas o zancadas (se puede visualizar en la figura 1.4), donde un sistema de detección de pose puede medir con precisión los ángulos de cadera, rodilla y tobillo para corregir la postura y prevenir lesiones [23]. También se aplica en disciplinas como el running, el yoga o los deportes de equipo, donde se identifican errores técnicos, desequilibrios musculares o asimetrías. En aplicaciones de fitness en casa, la detección de pose permite ofrecer entrenamientos interactivos, en los que el sistema evalúa automáticamente la ejecución de los ejercicios y proporciona correcciones visuales o sonoras, emulando a un entrenador personal [24].

Figura 1.4: Análisis de calidad de zancada con detección de pose. Fuente: [3]

En el sector sanitario, estas tecnologías están transformando la rehabilitación física: permiten monitorizar el cumplimiento de ejercicios terapéuticos a distancia y detectar posibles movimientos compensatorios o posturas incorrectas (un ejemplo en la figura 1.5) que podrían ralentizar la recuperación del paciente. Esta capacidad de seguimiento no invasivo resulta especialmente útil en contextos de telemedicina y fisioterapia domiciliaria. Además, también se están explorando aplicaciones médicas más especializadas, como la monitorización de la posición de la cabeza en resonancias magnéticas fetales [25].

En el ámbito de la conducción, la detección de pose se emplea como medida de seguridad pasiva (puede apreciarse en la figura 1.6a). Los sistemas instalados en el habitáculo del vehículo pueden monitorizar si el conductor está distraído, somnoliento o si sus movimientos indican una posible pérdida de control, activando alertas preventivas o incluso interviniendo de forma automática en la conducción.

También en el entretenimiento digital [26], la detección de pose ha abierto nuevas formas de interacción. En videojuegos, realidad virtual y realidad aumentada, el cuerpo

Figura 1.5: Análisis de postura con detección de pose. Fuente: [4]

del usuario se convierte en el principal controlador (como en la figura 1.6b), eliminando la necesidad de mandos físicos y generando experiencias más inmersivas e intuitivas. Esto ha dado lugar a una categoría de juegos activos o "exergames", que fomentan la actividad física de forma lúdica.

(a) Detección de pose aplicada a conductores. Fuente: [27]

(b) Detección de pose en videojuegos sin mando físico. Fuente: [28]

Figura 1.6: Ejemplos destacados de sistemas de detección de pose en distintos ámbitos.

Más allá de los ámbitos más consolidados, la detección de pose está comenzando a tener un impacto creciente en otros contextos emergentes. En el ámbito educativo, por ejemplo, se utiliza para mejorar la enseñanza de actividades físicas como danza, teatro o artes marciales, mediante la corrección automática de movimientos y posturas. Además, en 2024, Carlota Vera desarrolló un sistema de detección y análisis de emociones y comportamientos de estudiantes en el aula, basado en técnicas de aprendizaje profundo (ejemplo en la figura 1.7a), como parte de su TFG [29], lo que demuestra el potencial de estas tecnologías en entornos pedagógicos más amplios. En la industria, la detección de pose permite supervisar la ergonomía de los trabajadores en líneas de producción [30] (se visualiza en la figura 1.7b), reduciendo el riesgo de lesiones derivadas de posturas incorrectas o repetitivas. Por su parte, en el ámbito de la seguridad laboral, se emplea para identificar comportamientos peligrosos en entornos industriales o de construcción, activando alertas automáticas ante indicios de caídas, fatiga o movimientos no autorizados.

Incluso en sectores como la moda o el comercio electrónico [32], esta tecnología se está aplicando para simular el ajuste de prendas en tiempo real sobre el cuerpo del usuario, mejorando la experiencia de compra online. También comienza a explorarse su uso en marketing, donde permite analizar el comportamiento del consumidor mediante el seguimiento de posturas y movimientos dentro de un espacio físico.

(a) Ámbito educativo. Fuente: [29]

(b) Ámbito seguridad laboral. Fuente: [31]

Figura 1.7: Sistemas de detección de pose en industrias emergentes.

Una vez que somos capaces de estimar la postura y el movimiento de una persona a partir de una imagen o un vídeo, se abre la puerta a una nueva gama de aplicaciones centradas en el seguimiento y análisis del comportamiento humano. Esto da lugar a los llamados sistemas de monitorización del usuario, cuyo objetivo es interpretar las acciones, el estado físico o incluso el estado cognitivo de una persona en tiempo real.

Este tipo de sistemas se beneficia directamente de la información proporcionada por los algoritmos de detección de pose, ya que a partir de la posición de las extremidades, la orientación corporal o los movimientos repetitivos, se puede inferir si una persona está activa, distraída, cansada o incluso si necesita asistencia. En escenarios como la conducción, la rehabilitación médica o la interacción con dispositivos inteligentes, el monitorización basado en pose se convierte en una herramienta poderosa para aumentar la seguridad, la eficiencia y la personalización de la experiencia.

1.3. Sistemas de detección de distracciones del conductor

La seguridad ha sido siempre una de las mayores preocupaciones del ser humano, y de esa necesidad constante por protegernos han nacido algunos de los avances tecnológicos más importantes. Entre ellos destacan los sistemas de detección de distracciones del conductor. Teniendo en cuenta que gran parte de los accidentes en carretera están relacionados con algún tipo de distracción [33] [34], resulta lógico que el siguiente paso en la evolución del automóvil haya sido centrarse en detectar el estado del conductor. Además, tal como plantea este artículo [35], el comportamiento del conductor influye directamente en la seguridad vial, el consumo de combustible y la emisión de contaminantes. Desde julio de 2022, todos los vehículos nuevos están obligados a incorporar de serie sistemas avanzados de asistencia a la conducción. De hecho, ya existen iniciativas europeas como Euro NCAP[1], que evalúan el nivel de seguridad de los vehículos y establecen criterios específicos para definir qué sistemas de detección de distracciones del conductor son realmente eficaces. Todo esto ha impulsado significativamente la investigación en este ámbito en los últimos años.

Uno de los pioneros en este campo fue Toyota, que ya en 2006 presentó su sistema de detección de distracciones en los modelos Lexus. Este utilizaba una cámara de

[1] `https://www.euroncap.com/en/`

infrarrojos situada cerca del volante para detectar la orientación de la cabeza y los ojos del conductor, activando el sistema precolisión si el conductor no miraba al frente[2]. Saab desarrolló un sistema con dos cámaras infrarrojas para seguir los parpadeos y la orientación de la cabeza; utilizando el software de SmartEye[3], era capaz de detectar fatiga y distracción, y emitir advertencias sonoras y vibraciones en el asiento si detectaba un problema. Volvo, con su Driver Alert Control en 2007, no monitorizaba directamente al conductor, sino que analizaba el patrón de conducción (como las desviaciones de carril o las correcciones de dirección) para identificar signos de fatiga. Por su parte, Mercedes-Benz lanzó en 2009 el sistema Attention Assist, que creaba un perfil personalizado del conductor en los primeros minutos de conducción y podía emitir alertas visuales y sonoras si detectaba indicios de cansancio. Estos ejemplos ofrecen una panorámica de cómo fueron los primeros sistemas de detección de distracciones del conductor disponibles en el mercado: con cámaras infrarrojas, análisis de la posición de la cabeza, movimientos del volante, etc. También hubo investigaciones más experimentales que exploraron el uso de señales biológicas, como la actividad cerebral (EEG) o el ritmo cardíaco (ECG) [36] [37], ya que pueden ofrecer información muy precisa sobre el estado de alerta. Sin embargo, estos métodos requerían sensores colocados directamente sobre el cuerpo, lo que los hacía poco prácticos para su aplicación real en vehículos comerciales.

Hoy en día, los sistemas de detección de distracciones del conductor han evolucionado considerablemente gracias a los avances en visión artificial y al desarrollo de sensores más sofisticados. Ya no se limitan a detectar hacia dónde miran los ojos: ahora aplican algoritmos inteligentes en tiempo real. Las cámaras RGB e infrarrojas son mucho más avanzadas y están integradas de forma discreta en el vehículo (en el volante, salpicadero, etc.). Se utilizan redes neuronales entrenadas para reconocer patrones de somnolencia o distracción, y estos sistemas pueden adaptarse incluso al perfil individual de cada conductor. Además, algunos modelos incorporan sensores en el volante, asiento o cinturón para medir el ritmo cardíaco y la respiración, sin necesidad de contacto directo [38]. Podemos ver un ejemplo en la figura 1.8.

Figura 1.8: Ejemplo de un sistema de monitorizacion al conductor. Fuente: [5]

Una de las claves de estos sistemas actuales es la fusión de datos [39] [40]: se

[2] https://global.toyota/en/detail/274094,
https://media.lexus.co.uk/lexus-ls-460-achieves-world-first-in-preventive-safety

[3] https://www.smarteye.se/

combinan señales visuales, fisiológicas y de comportamiento del propio vehículo para obtener una visión más completa del estado del conductor [41]. Algunos sistemas incluso tienen en cuenta el estado del tráfico, la hora del día o la duración del trayecto para ajustar su nivel de alerta. Y ya no se limitan a emitir avisos: también pueden activar funciones del coche, como el frenado automático o el mantenimiento en el carril.

Hay ejemplos muy claros en el mercado actual. Tesla, por ejemplo, incluye una cámara interior que detecta si el conductor está mirando a la carretera, y analiza la frecuencia del parpadeo y la orientación del rostro para detectar fatiga [42] (se puede apreciar en la figura 1.9a) [43]. Mercedes-Benz analiza más de 70 parámetros para detectar fatiga [44] y, en sus modelos de alta gama, una cámara interior es capaz de reconocer expresiones faciales y microgestos [45] (se ilustra en la figura 1.9b). Subaru, además del seguimiento ocular mediante una cámara infrarroja, cuenta con reconocimiento facial para ajustar automáticamente el asiento y los espejos al conductor registrado, y lanza alertas si detecta distracción prolongada [46] (un ejemplo es la figura 1.9c). En Toyota, el sistema puede incluso frenar o tomar el control temporalmente si detecta que el conductor está distraído o somnoliento [47].

(a) Tesla. Fuente: [43]

(b) Mercedes-Benz. Fuente: [44]

(c) Subaru. Fuente: [46]

Figura 1.9: Distintos sistemas de detección de distracciones.

1.3.1. Inteligencia artificial en sistemas de detección de distracciones

La incorporación de IA ha marcado un punto de inflexión [48] en el desarrollo de los sistemas de detección de distracciones. Estos sistemas ya no se limitan a una simple recolección y procesamiento de datos mediante reglas predefinidas; ahora son capaces de aprender de los comportamientos, adaptarse a distintos perfiles y anticipar riesgos antes de que se materialicen. Gracias al aprendizaje automático (*machine learning* (ML)), los modelos actuales pueden identificar patrones complejos asociados a la fatiga, distracción o estrés, incluso cuando estos varían de un conductor a otro.

Uno de los enfoques más extendidos en este ámbito es el uso de modelos supervisados, entrenados con grandes conjuntos de datos etiquetados donde se han registrado diferentes estados del conductor. Estos modelos permiten clasificar con gran precisión si una persona está atenta, distraída, somnolienta o incluso emocionalmente alterada. Por ejemplo, a partir de secuencias de imágenes faciales, *Convolutional Neural Networks (redes neuronales convolucionales)* (CNN) pueden detectar cambios sutiles en la expresión, la dirección de la mirada o el ritmo del parpadeo. A su vez, las *Recurrent Neural Networks (redes neuronales recurrentes)* (RNN) o los modelos basados en Transformers pueden aprovechar la información temporal para comprender mejor la evolución del comportamiento a lo largo del tiempo.

En este contexto, existen soluciones que recurren a arquitecturas más pesadas, como los sistemas DMS basados en la nube [49], donde los datos se transmiten continuamente a servidores externos para ser procesados mediante modelos de gran tamaño y alta capacidad. Este enfoque ofrece una gran precisión y la posibilidad de actualizar los modelos de manera centralizada, pero implica una alta dependencia de la conectividad, mayor latencia, un considerable consumo de recursos y poca privacidad, tanto en el vehículo como en la infraestructura de red.

Además, el uso de IA no se limita únicamente al análisis visual. También se aplica en la fusión multimodal de datos, donde se integran variables visuales, fisiológicas y contextuales. Por ejemplo, un sistema puede combinar los datos de la cámara con los del sensor de ritmo cardíaco y con información contextual (como el tipo de vía o el tráfico actual) para ofrecer una evaluación mucho más precisa del estado del conductor. Algunos estudios recientes exploran incluso la posibilidad de utilizar aprendizaje profundo para modelar la atención del conductor y predecir zonas de riesgo en su campo visual [50].

La personalización también se ha visto potenciada por la IA. Cada conductor tiene patrones únicos de comportamiento: cómo sujeta el volante, cuánto parpadea o qué movimientos realiza al conducir. Los sistemas modernos pueden aprender estos patrones y crear modelos personalizados que ajustan los umbrales de detección en función del individuo, reduciendo los falsos positivos y aumentando la eficacia de las alertas.

No obstante, la aplicación de inteligencia artificial en estos sistemas plantea también importantes desafíos. Entre ellos, destacan la necesidad de proteger la privacidad de los datos del usuario, garantizar la robustez ante condiciones cambiantes (como iluminación o uso de gafas), y asegurar una interpretación ética de las decisiones automatizadas. Por eso, cada vez es más habitual que los sistemas incluyan mecanismos de transparencia (explicabilidad de los modelos) y permitan cierto grado de intervención o configuración por parte del conductor.

En esta línea, este TFG se centra en el desarrollo de un sistema de detección de distracciones del conductor utilizando inteligencia artificial, basado en el análisis detallado de la cara, el torso superior y las manos. Este sistema apuesta por una solución accesible y eficiente, orientada a reducir al máximo la carga computacional, de modo que pueda ejecutarse en sistemas de bajo coste que normalmente van integrados en los vehículos, evitando así el envío de datos a sub-sistemas en la nube, lo cual supondría un problema de privacidad para los usuarios. La meta de este trabajo es demostrar que es posible detectar señales de distracción sin recurrir a infraestructuras complejas ni a equipamiento costoso, facilitando así su integración en vehículos de gama media o en entornos donde la eficiencia, la simplicidad y la protección de la privacidad resultan fundamentales.

Este proyecto se ha llevado a cabo gracias a la colaboración del Departamento de Sistemas Conectados, Cooperativos y Autónomos de Vicomtech[4], un centro tecnológico de referencia especializado en inteligencia artificial y visión por computador. Su apoyo ha sido fundamental, ya que nos han facilitado el acceso al Dataset *driving monitoring dataset* (DMD), lo que ha permitido desarrollar y validar las distintas fases del trabajo de manera efectiva. Agradecemos especialmente la confianza depositada y la cesión de los recursos necesarios para hacer posible este proyecto.

[4] `https://www.vicomtech.org/es/`

Capítulo 2
Objetivos

En este capítulo se describe el problema que trata de resolver este TFG, definiendo los objetivos y requisitos establecidos durante el desarrollo del proyecto.

2.1. Descripción del problema

La distracción al volante es uno de los factores que más influyen en la siniestralidad vial[1]. Se produce cuando el conductor desvía su atención de la tarea de conducir, incluso durante breves lapsos, como apartar la mirada de la carretera. Entre las causas más comunes se encuentra el uso del teléfono móvil, que incrementa considerablemente el riesgo de accidente, con niveles de peligrosidad comparables a la conducción bajo la influencia del alcohol. Además, actividades como mantener una conversación durante la conducción afectan la capacidad para mantener una velocidad constante, respetar la distancia de seguridad y responder con eficacia ante situaciones imprevistas.

Conscientes de la relevancia de la distracción como factor de riesgo, el objetivo principal de este proyecto es diseñar y desarrollar un sistema de detección de distracciones del conductor capaz de evaluar de forma continua su comportamiento. Este sistema buscará detectar signos de distracción mediante el análisis de indicadores visuales y comportamentales, con el fin de alertar y prevenir posibles situaciones de riesgo antes de que se produzcan. Para ello, se emplearán técnicas de visión artificial que permitan un seguimiento no intrusivo y eficiente del conductor durante la conducción.

Para alcanzar este objetivo general, se han definido los siguientes subobjetivos:

1. Desarrollar un sistema para detectar la dirección de la mirada y así identificar distracciones al volante.

2. Implementar un sistema capaz de reconocer acciones específicas del conductor a través de la posición de sus brazos y manos.

3. Detección de uso del teléfono móvil para identificar distracciones severas en el conductor.

4. Diseñar un sistema que analice el estado de distracción del conductor a partir de la información proporcionada por los sistemas anteriores para proporcionar información útil y valiosa.

[1] `https://www.dgt.es/muevete-con-seguridad/evita-conductas-de-riesgo/distracciones-al-conducir/`

2.2. Requisitos

Los requisitos necesarios para este trabajo son:

1. El sistema debe operar con baja latencia para asegurar análisis y respuestas rápidas.
2. Implementación de un sistema de bajo coste computacional.
3. Evaluación del sistema utilizando conjuntos de datos y pruebas con personas reales en entornos de conducción reales.
4. Tolerancia a cambios ligeros en la posición de la cámara.
5. Compatibilidad con dispositivos borde (edge), permitiendo su integración en sistemas embarcados sin depender de infraestructuras externas.

2.3. Metodología

El desarrollo de este TFG se llevó a cabo entre mayo de 2024 y junio de 2025, siguiendo la metodología ágil Scrum[2]. Este enfoque organiza el trabajo en ciclos cortos denominados sprints, que en este proyecto tuvieron una duración variable entre una y cuatro semanas. La división en sprints facilitó una gestión ordenada de las tareas, permitiendo establecer objetivos claros y alcanzables en cada ciclo.

Cada sprint finaliza con una revisión del progreso para analizar resultados, detectar dificultades y ajustar las siguientes actividades según fuera necesario. Esta estructura iterativa fomenta una mejora continua y una adaptación dinámica a los retos surgidos durante el desarrollo.

Para asegurar una comunicación constante y efectiva, se mantuvieron reuniones semanales a través de Microsoft Teams[3], con una duración aproximada de 30 a 60 minutos. Estas reuniones permitieron hacer seguimiento del avance, resolver incidencias y coordinar los objetivos para los sprints siguientes. Además, se empleó el correo electrónico institucional para atender cuestiones urgentes y compartir información relevante fuera de las sesiones programadas.

Este enfoque basado en Scrum contribuye a mantener un ritmo de trabajo constante, a mejorar la organización y a garantizar la entrega progresiva de resultados a lo largo del proyecto.

Durante el proyecto se ha utilizado un repositorio en GitHub[4] para controlar las distintas versiones del código fuente y los datos generados, facilitando la colaboración y la trazabilidad de los cambios. En este repositorio se puede encontrar todo el código fuente del trabajo.

Con el fin de promover la difusión, reutilización y mejora continua del trabajo, se ha decidido publicar el código fuente bajo la licencia MIT[5], la cual permite su uso, modificación y distribución incluso con fines comerciales, siempre que se conserve el aviso de copyright y la declaración de la licencia original.

[2] `https://www.atlassian.com/es/agile/scrum`
[3] `https://www.microsoft.com/es-es/microsoft-teams/log-in`
[4] `https://github.com/RoboticsLabURJC/2024-tfg-arantxa-garcia`
[5] `https://opensource.org/licenses/MIT`

2.4. Plan de trabajo

El desarrollo del proyecto se estructuró en varias fases, que se detallan a continuación:

1. **Inicio del proyecto:**
   - Se comenzó realizando un análisis preliminar del conjunto de datos disponible. Este primer paso fue fundamental para familiarizarse con la estructura interna del dataset, identificar el formato en el que se encontraban los datos y comprender las etiquetas que acompañaban a cada muestra.
   - Posteriormente, se estudió en detalle cómo debía organizarse y tratarse la información para poder aprovecharla adecuadamente en las siguientes fases del proyecto. Esto incluyó decisiones sobre limpieza de datos, segmentación de información relevante y estrategias para preparar el dataset de forma eficiente.
   - Paralelamente, se gestionaron los permisos y credenciales necesarios para acceder al servidor de trabajo, en el que se llevaría a cabo el desarrollo completo del proyecto.
2. **Desarrollo del proyecto:**
   - Una vez preparado el entorno de trabajo, se procedió a la extracción de información relevante a partir de los vídeos que componían el dataset. Esta fase implicó recorrer los archivos multimedia para obtener los datos clave que serían utilizados posteriormente. Todo este contenido fue almacenado de forma estructurada para facilitar su uso futuro.
   - Para mejorar la calidad del conjunto de entrenamiento, se diseñó y desarrolló una herramienta específica que permitiera filtrar únicamente los datos válidos. Esta herramienta también se encargaba de balancear las etiquetas, reduciendo el sesgo y mejorando así la capacidad de generalización de los modelos que se entrenarían posteriormente.
   - Con los datos ya procesados, se procedió a entrenar el primer modelo del sistema. Este modelo estaba orientado a la clasificación de acciones del conductor, como por ejemplo si tenía ambas manos en el volante o estaba realizando alguna acción que indicara distracción.
   - A continuación, se repitió un proceso similar de tratamiento de datos y entrenamiento para desarrollar un segundo modelo. En este caso, el objetivo era predecir la dirección de la mirada del conductor, lo cual ofrecía una información desde otro punto de vista sobre su nivel de atención.
   - Posteriormente, se implementó un modelo adicional que trabajaba directamente sobre las imágenes capturadas, con el fin de detectar la presencia de teléfonos móviles. Esta funcionalidad era clave para identificar una de las causas más comunes de distracción al volante.
   - Finalmente, se realizó la integración de los tres modelos desarrollados. Esta integración permitía que los resultados de las distintas tareas se combinaran, ofreciendo una valoración conjunta del comportamiento del conductor y permitiendo una evaluación más completa de su nivel de atención.

3. **Evaluación del sistema:**

    - Para validar la eficacia de los modelos entrenados, se generaron gráficas y métricas detalladas. Estas representaciones permitieron analizar el comportamiento de cada modelo, identificar posibles errores o inconsistencias y cuantificar el nivel de precisión alcanzado en cada tarea.

4. **Redacción de la memoria:**

    - Como cierre del proyecto, se procedió a la elaboración de la memoria. En este documento se recogieron todos los aspectos relevantes del trabajo realizado, desde la justificación y objetivos iniciales hasta los detalles técnicos del desarrollo, el análisis de los resultados obtenidos y las conclusiones finales.

Capítulo 3
Plataforma de desarrollo

A lo largo del desarrollo de este proyecto se han utilizado diversas herramientas. En este capítulo se describe cada una de ellas, abarcando desde los recursos empleados para la programación hasta las bibliotecas y módulos que sustentan la funcionalidad del trabajo.

3.1. Lenguajes de programación

3.1.1. Python

Python[1] es un lenguaje de programación de alto nivel e interpretado, lo que significa que no requiere de un proceso de compilación para ejecutar su código. Se trata de un lenguaje multiparadigma, ya que permite trabajar tanto con programación estructurada como orientada a objetos. Es ampliamente reconocido por su sintaxis clara, su facilidad de aprendizaje y su extensa colección de bibliotecas, lo que lo convierte en una herramienta especialmente adecuada para el desarrollo de este proyecto.

En este proyecto, Python 3 ha sido el lenguaje elegido para el desarrollo completo del código. Se ha empleado tanto en los programas que integran los modelos de detección de postura como en aquellos destinados al análisis y representación de datos para el entrenamiento de modelos.

OpenCV

OpenCV[2] (Open Source Computer Vision Library) es una biblioteca de código abierto para Python diseñada para facilitar el desarrollo de aplicaciones de visión por computadora. Fue desarrollada por Intel[3] en 1999 y lanzada como proyecto de código abierto en el año 2000. Sus algoritmos permiten llevar a cabo una amplia variedad de tareas, desde la detección de bordes hasta el reconocimiento de objetos, y destacan por su rapidez y eficiencia, especialmente en el procesamiento de vídeo en tiempo real. El código en 3.1 nos muestra un ejemplo del uso de esta biblioteca.

En este proyecto, OpenCV se ha empleado para capturar imágenes desde vídeos y cámaras, ajustarlas en tamaño y añadir información visual sobre las mismas, funciones

[1] `https://www.python.org/`
[2] `https://opencv.org/`
[3] `https://www.intel.com/content/www/us/en/homepage.html`

```
def draw_connections(self, frame, keypoints, connections):
    for idx1, idx2 in connections:

        x1, y1, x2, y2 = None, None, None, None
        for x, y, inx in keypoints:
            if inx == idx1:
                x1 = int(x * frame.shape[1])
                y1 = int(y * frame.shape[0])
            elif inx == idx2:
                x2 = int(x * frame.shape[1])
                y2 = int(y * frame.shape[0])

        if x1 and y1 and x2 and y2:

            cv2.line(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
```

Código 3.1: Ejemplo de uso de la biblioteca OpenCV

clave en el procesamiento y visualización de los datos del sistema de detección de distracciones.

NumPy

NumPy[4] es una biblioteca de código abierto para Python, ampliamente utilizada en el ámbito científico y matemático. Proporciona estructuras de datos eficientes, como los arrays multidimensionales, y una amplia colección de funciones para realizar operaciones numéricas de forma rápida y sencilla. A menudo se emplea como alternativa a MATLAB[5] en tareas de análisis numérico. En nuestro caso, la hemos utilizado principalmente para realizar los cálculos matemáticos necesarios y para generar matrices formadas por ceros o unos. Contamos con un ejemplo de su uso en **??**.

Pandas

Pandas[6] es una biblioteca de código abierto para Python que facilita el análisis y la manipulación de datos. Ofrece estructuras de datos muy eficientes, como las Series (estructuras unidimensionales con etiquetas) y los DataFrames (estructuras bidimensionales similares a tablas con filas y columnas etiquetadas). Con Pandas se pueden leer y escribir archivos en distintos formatos como CSV, Excel o JSON, así como filtrar, ordenar, agrupar y transformar datos de forma sencilla. También permite limpiar conjuntos de datos, por ejemplo, eliminando duplicados o rellenando valores faltantes, y realizar análisis estadísticos básicos. Mediante el uso de DataFrames en Pandas, llevamos a cabo la conversión de nuestros datos al formato necesario para su procesamiento. Se puede apreciar un ejemplo en 3.2.

[4] https://numpy.org/
[5] https://www.mathworks.com/products/matlab.html
[6] https://pandas.pydata.org/

```python
import numpy as np

a = np.array([1, 2, 3, 4, 5])
print("Original array:", a)

b = a + 10
print("Array plus 10:", b)

matrix = np.array([[1, 2], [3,
    4]])
transposed = matrix.T
print("Transposed of the matrix:")
print(transposed)
```

(a) Ejemplo de uso de NumPy.

```
arantxa@Vulcano: $ python3 ejemplo.py
Original array: [1 2 3 4 5]
Array plus 10: [11 12 13 14 15]
Trasposed of the matrix:
[[1 3]
 [2 4]]
```

(b) Resultado generado por el código.

Figura 3.1: Código y resultado visual del ejemplo con NumPy.

```python
import pandas as pd

data = {
    "Name": ["Ana", "Luis",
        "Carlos"],
    "Age": [23, 34, 29],
    "City": ["Madrid", "Seville",
        "Barcelona"]
}

df = pd.DataFrame(data)

print("Full DataFrame:")
print(df)

print("\nAges:")
print(df["Age"])

print("\nPeople older than 25:")
print(df[df["Age"] > 25])
```

(a) Ejemplo de uso de la biblioteca Pandas.

```
arantxa@Vulcano: $ python3 ejemplo.py
Full DataFrame:
     Name  Age       City
0     Ana   23     Madrid
1    Luis   34    Seville
2  Carlos   29  Barcelona

Ages:
0    23
1    34
2    29
Name: Age, dtype: int64

People older than 25:
     Name  Age       City
1    Luis   34    Seville
2  Carlos   29  Barcelona
```

(b) Resultado generado por el código.

Figura 3.2: Código y salida visual del ejemplo usando Pandas.

Matplotlib

Matplotlib[7] es una biblioteca de código abierto para Python desarrollada en 2002, cuyo objetivo inicial fue la visualización de señales eléctricas del cerebro en estudios sobre epilepsia, replicando las capacidades gráficas de Matlab. Con el tiempo, ha evolucionado hasta convertirse en una herramienta muy versátil y ampliamente

[7] https://matplotlib.org/

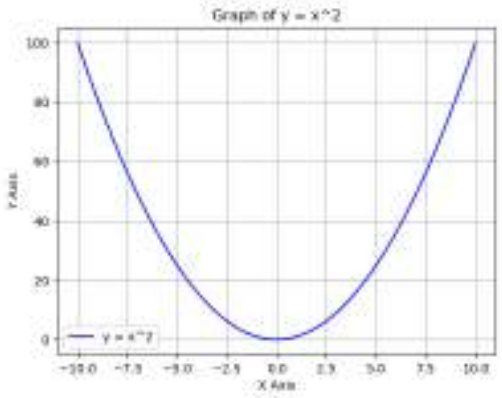
utilizada para la creación de gráficos y diagramas de alta calidad. En este proyecto, la he empleado para generar todas las gráficas necesarias, lo que me ha permitido visualizar y analizar de forma clara los resultados obtenidos a lo largo del desarrollo. En el ejemplo 3.3 se puede apreciar como se podría usar.

```python
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-10, 10, 100)
y = x**2

plt.plot(x, y, label="y = x^2",
    color="blue")
plt.title("Graph of y = x^2")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.legend()
plt.grid(True)
plt.show()
```

(a) Código Python utilizando Matplotlib.



(b) Resultado generado por el código.

Figura 3.3: Código y resultado visual generado con Matplotlib.

Scikit-learn

Scikit-learn[8] es una biblioteca de código abierto para Python diseñada específicamente para el aprendizaje automático (machine learning). Está construida sobre bibliotecas como NumPy, SciPy y matplotlib, lo que le permite integrarse fácilmente en entornos científicos y de análisis de datos en Python. Ofrece una amplia colección de algoritmos y herramientas para tareas como clasificación, regresión, clustering (agrupamiento), reducción de dimensionalidad, selección de características y validación de modelos. Su interfaz simple y coherente permite entrenar, evaluar y ajustar modelos de forma eficiente y con pocas líneas de código. En este trabajo se ha utilizado el algoritmo de **Random Forest**:

El algoritmo **Random Forest** (figura 3.4) es una técnica de aprendizaje automático que se basa en la construcción de múltiples árboles de decisión, en lugar de utilizar uno solo. La idea principal es que, al combinar varios árboles, se obtiene un modelo más preciso y menos propenso al sobreajuste. Para construir cada árbol, el algoritmo selecciona aleatoriamente una parte del conjunto de datos original, esto significa que algunos ejemplos pueden aparecer varias veces en esa muestra, mientras que otros pueden no ser seleccionados y, por tanto, no participar en la construcción de ese árbol concreto. Además, en cada nodo de decisión del árbol, no se consideran todas las variables disponibles, sino solo un subconjunto elegido aleatoriamente. Esta aleatoriedad introduce diversidad entre los árboles y evita que todos aprendan exactamente los mismos patrones.

[8] https://scikit-learn.org/stable/

Una vez construidos todos los árboles, sus predicciones se combinan: en tareas de clasificación, se selecciona la clase con mayor número de votos; en problemas de regresión, se calcula la media de los resultados. Gracias a este enfoque, **Random Forest** produce modelos más estables y menos sensibles a las variaciones del conjunto de entrenamiento que un único árbol de decisión. Además, ofrece un rendimiento más robusto frente al ruido en los datos y más confiable en presencia de características irrelevantes o correlacionadas, lo que lo convierte en una opción eficaz para una amplia gama de problemas de aprendizaje supervisado.

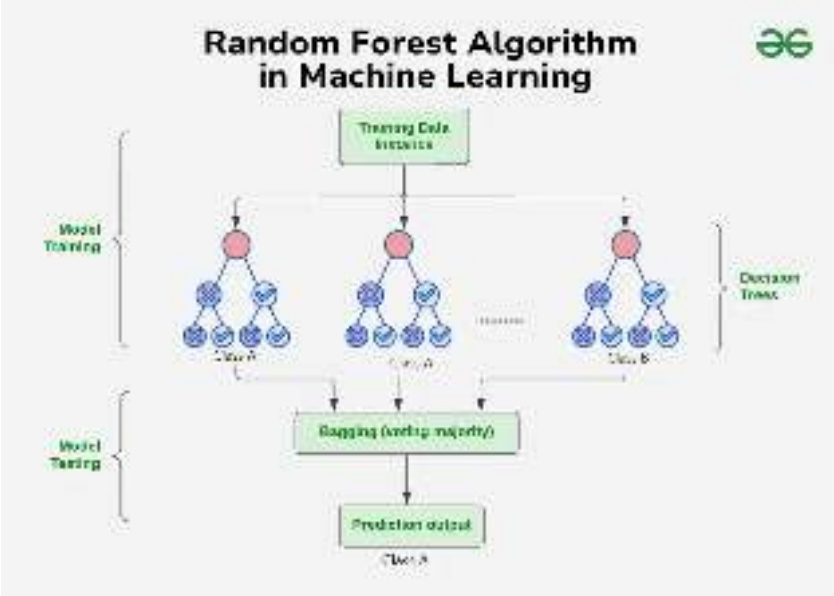


Figura 3.4: Funcionamiento del algoritmo de Random Forest. Fuente: [6]

3.2. Entornos de desarrollo

3.2.1. Anaconda

Anaconda[9] es una plataforma pensada para simplificar la gestión de entornos y librerías en proyectos desarrollados con Python. Su principal ventaja es que permite crear entornos independientes, donde se pueden definir versiones concretas tanto del intérprete de Python como de las bibliotecas necesarias, lo que evita conflictos entre proyectos y asegura que cada uno funcione con las versiones exactas que requiere. Esto resulta especialmente útil en trabajos que dependen de múltiples herramientas o versiones específicas de ciertos paquetes. Nosotros la hemos utilizado para facilitarnos la tarea de instalar todas las librerías que hemos ido necesitando a lo largo del proyecto, agilizando así el proceso de configuración y evitando problemas de compatibilidad.

[9] `https://www.anaconda.com/`

3.2.2. Visual Studio Code

Visual Studio Code (VSCode) es un editor de código fuente gratuito, lanzado en 2015 por Microsoft[10], compatible con múltiples sistemas operativos. Está diseñado para facilitar la creación, edición y depuración de programas en diversos lenguajes de programación, e incluye soporte integrado para Git y manejo sencillo del código. Además, cuenta con una amplia variedad de extensiones que optimizan la experiencia de codificación, ofreciendo funciones como autocompletado inteligente y terminales integrados, lo que permite un flujo de trabajo más ágil y eficiente. VSCode ha sido empleado como editor principal del código del proyecto.

Figura 3.5: Interfaz de visual studio code. Fuente: [7]

3.3. Redes neuronales de detección de pose

3.3.1. MediaPipe

MediaPipe[11] es un framework de código abierto orientado al desarrollo de aplicaciones de visión por computador en tiempo real. Proporciona un conjunto de soluciones preentrenadas y optimizadas para tareas complejas como reconocimiento facial, seguimiento de gestos, detección de objetos y análisis del movimiento corporal. Su arquitectura modular y multiplataforma facilita la creación de aplicaciones avanzadas con alta flexibilidad y eficiencia. Esta modularidad no solo simplifica la integración de componentes específicos, sino que también permite personalizar y extender las funcionalidades según las necesidades del proyecto. Además, MediaPipe está diseñado para ejecutarse eficientemente en dispositivos con recursos limitados, sin necesidad de contar con GPU dedicada, lo que permite su uso en entornos embebidos o sistemas con hardware modesto. En GitHub[12] está el código con todas sus soluciones.

En este proyecto hemos utilizado la detección de pose, la de malla facial y la detección de manos.

MediaPipe Pose

Esta herramienta permite identificar puntos clave del cuerpo humano en imágenes o vídeos, lo que facilita tareas como el análisis postural y la clasificación de movimientos.

[10] `https://www.office.com/`
[11] `https://ai.google.dev/edge/mediapipe/solutions/guide?hl=es-419`
[12] `https://github.com/google-ai-edge/mediapipe`

Ofrece diversas opciones de configuración, como el número máximo de personas a detectar y un umbral mínimo de confianza para validar cada punto clave. El sistema utiliza dos modelos principales para su funcionamiento: uno encargado de detectar la presencia de cuerpos y algunos puntos clave iniciales, y otro que refina este mapeo estimando las coordenadas tridimensionales de todos los puntos identificados. Su base es una red neuronal convolucional diseñada para optimizar el procesamiento en tiempo real. El resultado final consiste en un conjunto de 33 puntos clave con sus coordenadas normalizadas entre 0 y 1 en los ejes x e y, junto con un valor que indica la probabilidad de que cada punto sea visible. Además, incluye una estimación de la profundidad de cada punto mediante una coordenada z, lo que aporta información tridimensional adicional para un análisis más completo. Podemos ver una representación de los puntos en la figura 3.6

(a) Ejemplo Mediapipe Pose (b) Puntos de Mediapipe Pose

Figura 3.6: Mediapipe Pose. Fuente: [8]

MediaPipe Face Mesh

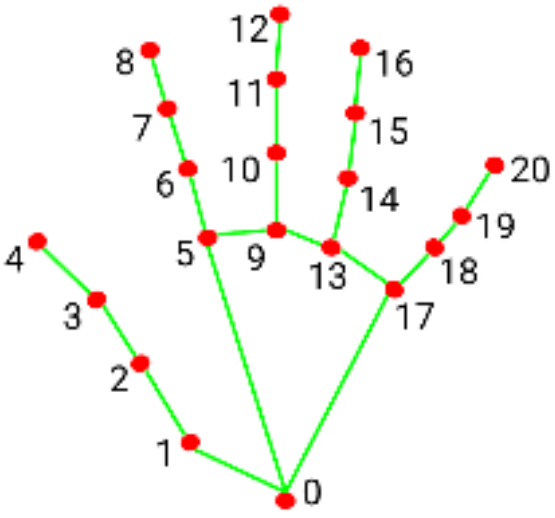
Este modelo de MediaPipe permite detectar una malla facial detallada, compuesta por 468 puntos que representan distintas zonas del rostro. Gracias a esta representación precisa, es posible realizar análisis de expresiones faciales en imágenes y vídeos, aplicar filtros y efectos sobre el rostro, o crear avatares virtuales con movimientos realistas. El sistema funciona mediante la combinación de varios modelos especializados. Primero, un modelo de detección facial identifica la presencia del rostro y algunos puntos clave iniciales. A continuación, un segundo modelo se encarga de completar el mapeo, generando la malla completa con los 468 puntos, como podemos ver en la figura 3.7. Finalmente, un tercer modelo analiza esta información para predecir las expresiones faciales, permitiendo interpretar emociones y movimientos faciales con gran detalle.

MediaPipe Hand Landmarker

Este sistema de MediaPipe para la detección de manos utiliza un conjunto integrado de modelos que incluyen un detector de palmas y un detector de puntos clave de la mano. El detector de palmas localiza las manos dentro de la imagen de entrada, mientras que el modelo de puntos clave identifica con precisión 21

Figura 3.7: Puntos de Mediapipe Face. Fuente: [9]

coordenadas correspondientes a nudillos y articulaciones dentro de las regiones de la mano previamente detectadas. Para optimizar el rendimiento en vídeo o transmisiones continuas, el sistema emplea un mecanismo inteligente que, tras detectar inicialmente las manos con el modelo de palmas, utiliza las regiones delimitadas por el modelo de puntos clave para realizar el seguimiento en los fotogramas siguientes. Solo si el sistema deja de detectar o seguir las manos en el vídeo, vuelve a activar el detector de palmas, lo que reduce la carga computacional y mejora la eficiencia durante la ejecución en tiempo real. Podemos ver un ejemplo de su funcionamiento en la figura 3.8



(a) Puntos de Mediapipe Hands. Fuente: [51]

(b) Ejemplo Mediapipe Hands. Fuente: [52]

Figura 3.8: Mediapipe Hands.

3.3.2. ViTPose

ViTPose[13] [10] es un modelo de código abierto basado en arquitecturas tipo Vision Transformer (ViT), diseñado específicamente para la tarea de estimación de pose humana. A diferencia de otros enfoques tradicionales basados en redes convolucionales, ViTPose utiliza mecanismos de atención para capturar relaciones espaciales complejas entre las articulaciones del cuerpo, logrando una mayor precisión en la localización de los puntos clave, incluso en escenas con oclusiones o poses inusuales. El modelo ha sido entrenado en grandes conjuntos de datos como COCO [53] y MPII [54], y se ha integrado dentro del ecosistema de herramientas de `OpenMMLab`, lo que facilita su uso en distintos entornos de desarrollo. Debido a la complejidad y tamaño del modelo, ViTPose suele requerir hardware con GPU para un procesamiento eficiente y tiempos de inferencia adecuados en tiempo real. Se aprecia un ejemplo de su funcionamiento en la figura 3.9.

(a) Ejemplo 1 (b) Ejemplo 2 (c) Ejemplo 3

Figura 3.9: Ejemplos del funcionamiento de VitPose. Fuente: [10]

3.4. YOLO

YOLO[14] (You Only Look Once) es un algoritmo de detección de objetos en tiempo real basado en redes neuronales convolucionales. A diferencia de otros métodos tradicionales como R-CNN, que requieren múltiples etapas para localizar y clasificar objetos, YOLO procesa toda la imagen en una sola pasada por la red, lo que permite realizar detecciones de forma extremadamente rápida y precisa. El modelo divide la imagen en una cuadrícula y, para cada celda, predice simultáneamente las clases de los objetos, la probabilidad de su presencia y las coordenadas de sus cajas delimitadoras (bounding boxes). Gracias a su velocidad, YOLO es especialmente útil en aplicaciones donde el tiempo de respuesta es crítico, como la conducción autónoma, la videovigilancia, la interacción con robots o el análisis de vídeo en tiempo real. Desde su primera versión en 2016, YOLO ha evolucionado significativamente a través de distintas versiones, mejorando tanto en precisión como en eficiencia. En nuestro caso lo

[13] `https://github.com/ViTAE-Transformer/ViTPose`
[14] `https://docs.ultralytics.com/es/`

hemos utilizado para detectar teléfonos móviles en cada frame de la cámara. Podemos ver su funcionamiento en la figura 3.10.

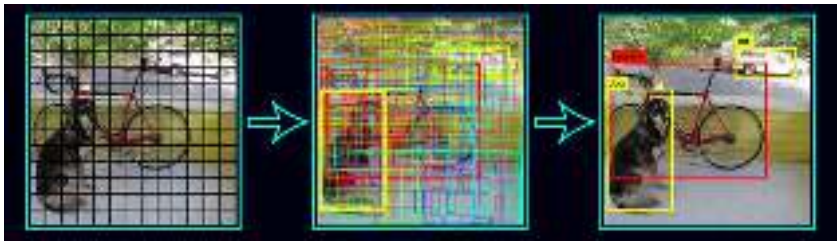


Figura 3.10: Funcionamiento de YOLO. Fuente: [11]

3.5. Dataset

El conjunto de datos utilizado para el entrenamiento y validación de los modelos desarrollados en este trabajo es el DMD[15] [55], una base de datos pública publicada por la empresa Vicomtech[16], en el marco del proyecto europeo VI-DAS. Se trata de un dataset multimodal que recoge información detallada sobre el comportamiento del conductor mediante el uso de múltiples cámaras instaladas en el interior del vehículo. Estas cámaras capturan simultáneamente tres tipos de datos: imagen RGB, profundidad (depth) e infrarrojo (IR), lo que permite obtener una representación más completa y robusta del entorno y del estado del conductor.

El conjunto está compuesto por grabaciones de 37 personas, con una duración total aproximada de 41 horas de vídeo y un volumen de datos que alcanza los 42 terabytes. La recopilación del material se diseñó con especial atención a la diversidad. El 27 % de los participantes eran mujeres y el 73 % hombres, todos mayores de edad y con una edad media de 30 años. Diez de ellos llevaban gafas, y en algunos casos fueron registrados tanto con como sin ellas. Las sesiones de grabación se realizaron en distintos momentos del día (mañana y tarde), permitiendo variaciones naturales en las condiciones de iluminación.

Cada grabación se estructura en sesiones definidas por un protocolo concreto que especificaba la actividad a realizar, el entorno de grabación, el participante involucrado y la condición lumínica. En cada sesión se utilizaron tres cámaras sincronizadas, orientadas respectivamente al cuerpo, al rostro y a las manos del conductor. Cada una de estas cámaras proporcionó datos en las tres modalidades mencionadas (RGB, IR y profundidad), generando un conjunto rico y complementario de información visual.

Además, las grabaciones fueron anotadas manualmente con más de 50 etiquetas que identifican distintas acciones o comportamientos del conductor, como desvíos de la mirada, uso del teléfono móvil, manipulación del volante, parpadeo, bostezos o cambios de postura. Estas anotaciones están organizadas conforme al estándar OpenLABEL VCD[17], lo que facilita su tratamiento y análisis con herramientas especializadas.

[15] `https://dmd.vicomtech.org/`

[16] `https://www.vicomtech.org/es/`

[17] `https://vicomtech.gitlab.io/v4/libraries/vcd/vcd-python/latest/vcd/`

Gracias a su riqueza multimodal, la variedad de condiciones recogidas y la exhaustividad de sus anotaciones, el dataset DMD constituye una base idónea para el desarrollo de sistemas avanzados de detección de distracciones, como el que se plantea en este TFG.

3.6. Hardware

3.6.1. Servidor Thor

El servidor Thor se encuentra en el CPD de la Escuela de Ingeniería Superior de Fuenlabrada de la *Universidad Rey Juan Carlos* (URJC) y forma parte de la infraestructura computacional avanzada del grupo RoboticsLab[18]. Está diseñado para ofrecer un alto rendimiento gracias a su procesador AMD Ryzen Threadripper 7960X y a su arquitectura de 64 bits. Además, incorpora una GPU NVIDIA GeForce RTX 4090 con 24 GB de memoria, lo que le permite abordar sin problema tareas como las pruebas de funcionamiento de sistemas como Vitpose y YOLO.

3.6.2. Jetson AGX Orin

La Jetson AGX Orin (se puede visualizar en la figura 3.11) es un dispositivo de computación embebida desarrollado por NVIDIA[19], diseñado específicamente para aplicaciones de inteligencia artificial en el borde (edge AI). Incorpora una GPU NVIDIA Ampere con 2048 núcleos CUDA y 64 núcleos Tensor, lo que le permite ejecutar modelos de visión por computador y aprendizaje profundo de forma eficiente sin necesidad de recurrir a servidores externos.

Además, dispone de una CPU ARMv8 de 64 bits, compuesta por 12 núcleos lógicos, de los cuales habitualmente permanecen activos 8 para optimizar el consumo energético y la gestión térmica. Su frecuencia máxima alcanza los 2,2 GHz, y cuenta con un sistema jerárquico de memoria caché que incluye 512 KiB de caché L1 para datos y para instrucciones, 2 MiB de caché L2 y 4 MiB de caché L3.

El dispositivo ofrece hasta 32 GB de memoria RAM LPDDR5, lo que resulta fundamental para procesar grandes volúmenes de datos y ejecutar modelos complejos de forma local. Estas características convierten a la Jetson AGX Orin en una plataforma idónea para aplicaciones que requieren procesamiento en el propio dispositivo y baja latencia, como los sistemas avanzados de monitorización del conductor.

[18] `https://roboticslaburjc.github.io/`
[19] `https://www.nvidia.com/es-es/`

Figura 3.11: Jetson AGX Orin.

Capítulo 4
Diseño e Implementación

En este proyecto se busca desarrollar un sistema de detección de distracciones del conductor, capaz de analizar con baja latencia las acciones que éste realiza al volante, con el fin de contribuir a la mejora de la seguridad vial mediante la identificación temprana de comportamientos potencialmente peligrosos.

En este capítulo se detalla el proceso completo de diseño e implementación llevado a cabo para alcanzar dicha meta, describiendo tanto el diseño de la arquitectura general como la implementación de algoritmos y las decisiones técnicas fundamentales adoptadas.

Asimismo, se presentan la evaluación y los resultados obtenidos en cada uno de los subsistemas desarrollados. Finalmente, se expone un estudio práctico de distracción de un conductor, realizado con datos reales, que ilustra el funcionamiento global del sistema y demuestra su capacidad para evaluar y representar gráficamente el nivel de distracción del conductor a lo largo del tiempo.

4.1. Arquitectura del sistema

En esta sección se describe el diseño de la arquitectura del sistema desarrollada para este TFG. La figura 4.1 ofrece una visión global de dicha arquitectura, sirviendo como punto de partida para detallar cada uno de los bloques que la componen. La fuente principal de datos del sistema proviene de dos imágenes que registran al conductor desde perspectivas complementarias: una centrada en el rostro y otra enfocada al cuerpo desde una vista lateral. Estas imágenes constituyen la base para el posterior procesamiento y análisis, y cada perspectiva se gestiona de manera independiente, ya que aporta información específica que sirve como entrada para los distintos módulos del sistema.

La imagen lateral (superior en el diagrama de la figura 4.1) permite analizar la postura general del conductor y el movimiento de sus manos. A partir de esta vista se extraen los puntos clave del cuerpo y de las manos, que se utilizan en dos módulos principales: el módulo de análisis de acciones, encargado de identificar la actividad que realiza el conductor (por ejemplo, beber, usar la radio, o realizar otras acciones con las manos), y el módulo de detección de distracciones, orientado a determinar si el conductor está manipulando objetos potencialmente peligrosos, como una botella o un teléfono móvil.

Por otro lado, la imagen frontal (inferior en el diagrama de la figura 4.1) se centra en el análisis facial. De esta vista se obtienen los puntos clave del rostro, que son

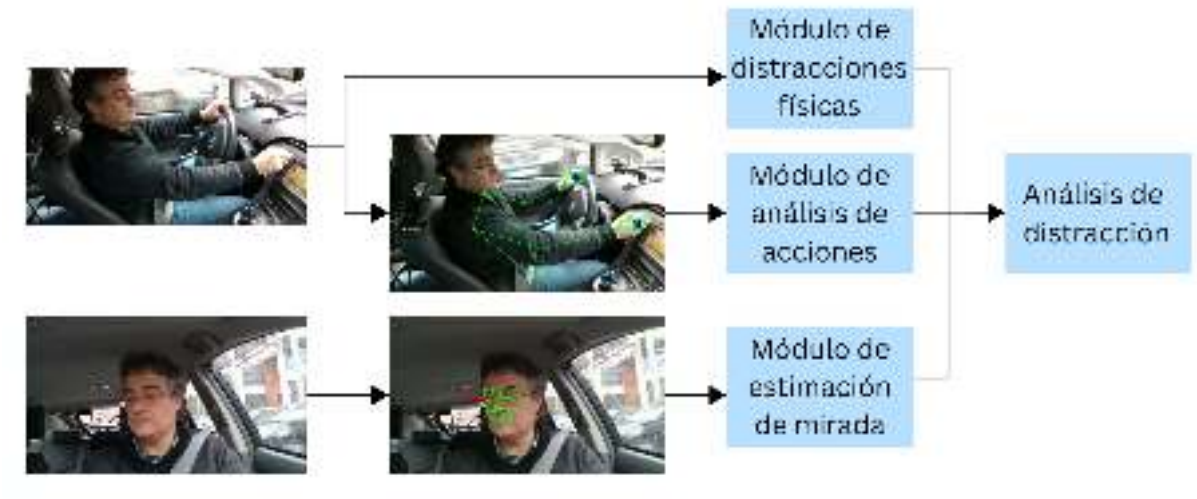


Figura 4.1: Arquitectura del sistema.

procesados en el módulo de estimación de mirada. Este módulo tiene como objetivo estimar la dirección de la mirada del conductor y valorar si se encuentra prestando atención a la carretera.

Finalmente, los resultados generados por los tres módulos (distracciones, acciones y mirada) se integran en el bloque de análisis de distracción. Este bloque actúa como núcleo centralizador de toda la información y permite evaluar, el nivel de distracción del conductor en cada instante. A partir de esta integración se obtiene una base sólida para la representación gráfica y el análisis global del comportamiento al volante, facilitando así la comprensión y la interpretación del estado del conductor de forma clara y estructurada.

4.2. Preparación del entorno

El sistema está diseñado para funcionar con dos cámaras RGB colocadas en posiciones estratégicas dentro del habitáculo del vehículo. La disposición recomendada de las cámaras se muestra en la figura 4.2, donde se representa tanto la cámara orientada al rostro del conductor como la cámara centrada en el cuerpo. También podemos ver un ejemplo real de ambas cámaras en la figura 4.3.

Esta configuración permite obtener una visión completa del comportamiento del conductor desde dos perspectivas distintas: una centrada en la cara, fundamental para la estimación de la mirada, y otra orientada al cuerpo, necesaria para el reconocimiento de acciones y la detección del uso de distintas distracciones físicas. El correcto posicionamiento de las cámaras es clave para garantizar la calidad y la utilidad de los datos capturados.

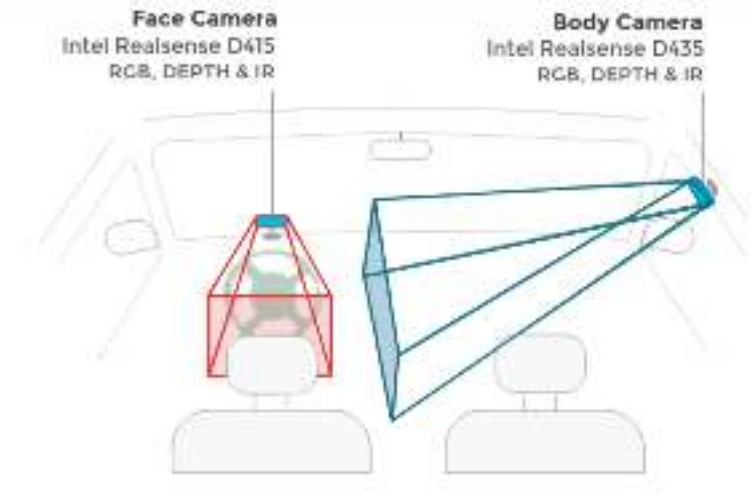


Figura 4.2: Esquema representativo de las cámaras dentro del coche. Fuente: [12]

(a) Visualización de la cámara facial ("Face Camera").

(b) Visualización de la cámara corporal lateral ("Body Camera").

Figura 4.3: Visualización de las cámaras representadas en la figura 4.2. Fuente: [12]

4.2.1. Dataset

Para la preparación de los módulos utilizados en este trabajo se ha empleado el conjunto que se ha descrito en la sub-sección 3.5. El uso de este dataset ha resultado especialmente útil, ya que ha evitado la necesidad de disponer de un vehículo real ni de realizar modificaciones físicas en él para adaptarlo a los requisitos del sistema, lo que ha facilitado significativamente la fase de desarrollo. Además, los datos empleados provienen de escenarios reales y no de simulaciones, lo que aporta mayor validez y realismo al desarrollo del sistema.

Conversión de los datos del dataset

El dataset original está compuesto por grabaciones en vídeo, procedentes de diferentes cámaras que registran el comportamiento del conductor desde varios ángulos y perspectivas. No obstante, dado que uno de los objetivos principales de este trabajo es diseñar un sistema lo más ligero y eficiente posible a nivel computacional, se ha optado por extraer únicamente los puntos clave (keypoints) de cada frame (que se obtienen de un modelo de detección de pose que se detallará más adelante). Los keypoints son un conjunto de coordenadas específicas que representan de manera abstracta la posición de partes anatómicas relevantes, como las articulaciones principales del cuerpo

(a) Imagen del dataset. (b) Representación de keypoints.

Figura 4.4: Procesamiento de un frame.

(hombros, codos, muñecas, caderas, etc.), las manos o los rasgos faciales (ojos, nariz, boca). Gracias a esta representación simplificada, es posible describir la postura y los movimientos del conductor sin necesidad de procesar toda la información contenida en la imagen completa. Podemos ver el procesamiento de un frame para calcular sus keypoints en la figura 4.4.

Esta estrategia permite prescindir del uso íntegro de las imágenes, reduciendo de forma significativa la cantidad de datos que se procesan y almacenan. Además, facilita guardar dichos puntos clave en archivos json estructurados, lo que simplifica tanto el procesamiento posterior como el análisis detallado de la información recogida. Gracias a ello, el sistema puede enfocarse en analizar el comportamiento y las acciones del conductor con un uso mucho más reducido de recursos computacionales, sin sacrificar la calidad de la información extraída.

Para llevar a cabo esta tarea, se ha desarrollado una aplicación específica, implementada en el script `converter.py`, cuya función inicial es sincronizar los vídeos correspondientes a cada sesión, ya que provienen de cámaras distintas y no comparten necesariamente la misma línea temporal exacta. Este proceso de sincronización es esencial para garantizar la coherencia temporal entre las diferentes fuentes de vídeo y asegurar que las imágenes capturadas en un mismo instante se correspondan correctamente entre todas las vistas disponibles.

Concretamente, el sistema compara las marcas temporales de cada archivo de vídeo y ajusta el punto de inicio y la frecuencia de extracción de frames, de forma que se pueda identificar y extraer el frame equivalente en cada uno de ellos. De este modo, se obtiene un conjunto de imágenes sincronizadas que representan exactamente el mismo momento desde perspectivas diferentes (rostro y cuerpo).

Una vez extraídos los frames sincronizados, el script `converter.py` aplica los modelos de estimación de pose correspondientes a cada tipo de cámara para obtener los keypoints relevantes de cada vista. Posteriormente, todos los puntos clave generados a partir de los diferentes frames se organizan y almacenan en ficheros JSON independientes. En concreto, para cada sesión se generan dos ficheros JSON principales: uno que contiene la información del cuerpo, incluyendo tanto la pose como los datos de las manos, y otro que recoge los keypoints faciales correspondientes al rostro. Se pueden ver las estructuras de estos ficheros en las figuras 4.5 y 4.6.

```
{
  "iterations": [
    {
      "frame": 1,
      "pose": [
        [0.325, 0.405, 11],
        [0.330, 0.589, 12],
        [0.421, 0.647, 13],
        ...
      ]
    }
  ]
}
```

Figura 4.5: Estructura del fichero de pose JSON.

```
{
  "iterations": [
    {
      "frame": 1,
      "face": [
        [0.532, 0.574, 0],
        [0.533, 0.497, 4],
        [0.534, 0.609, 17],
        ...
      ],
      "gaze": [
        [593, 331],
        [642, 279]
      ]
    }
  ]
}
```

Figura 4.6: Estructura del fichero de face JSON.

Un problema identificado en las primeras etapas del análisis es el fuerte desbalance presente en los datos del dataset, como se ilustra en la figura 4.7. Existen, en el dataset, acciones como *both hands on wheel* que cuentan con una cantidad significativamente mayor de muestras, mientras que otras apenas están representadas, como *texting left*, *reach backseat* o *hair and makeup*. Este desbalance supone un riesgo importante para el entrenamiento de modelos, ya que puede inducir un sesgo notable y limitar la capacidad de generalización.

Con el objetivo de mitigar este problema, se ha desarrollado una aplicación específica, implementada en el script `balancer.py`, orientada al balanceo de clases.

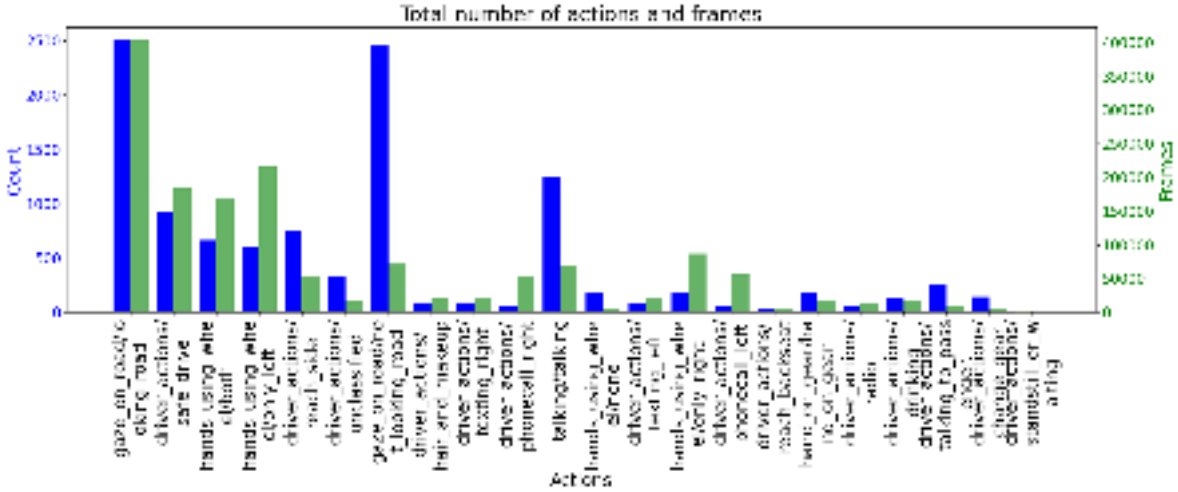


Figura 4.7: Cantidad de frames por acción.

Este script ha sido diseñado para procesar y organizar de manera automática los datos recogidos en cada sesión, garantizando que todas las acciones queden representadas con un número similar de muestras. En primer lugar, el script localiza y agrupa los diferentes ficheros JSON y vídeos asociados a cada sesión. Posteriormente, para cada acción definida en el dataset, se filtran y seleccionan únicamente aquellos frames que cumplen ciertos criterios de calidad (por ejemplo, ausencia de keypoints nulos) y que resulten representativos. Para ello, se implementan funciones específicas de validación que revisan la integridad y la coherencia de los puntos clave tanto del cuerpo como de las manos y el rostro.

Una vez identificados y validados los frames, el script aplica un criterio de muestreo configurable para limitar el número final de muestras por acción a un valor máximo N, evitando así la sobre-representación de determinadas categorías. Los frames seleccionados se almacenan en un directorio estructurado, donde cada acción dispone de su propio subdirectorio. Dentro de cada subdirectorio, se guardan tanto los ficheros JSON que contienen la información detallada de los keypoints asociados al frame como las imágenes extraídas de los vídeos correspondientes (pose y rostro), que sirven como referencia visual y facilitan futuras revisiones.

Gracias a esta estructura organizada y al control exhaustivo de la selección, se asegura una distribución equilibrada de ejemplos por clase, lo que contribuye significativamente a mejorar la robustez y la fiabilidad del proceso de entrenamiento y validación. Además, este procedimiento automatizado permite mantener la trazabilidad completa de cada muestra, facilitando el análisis detallado o posibles auditorías posteriores.

4.3. Módulo de análisis de acciones

El análisis de acciones constituye el primer módulo funcional de los tres que conforman la arquitectura global del sistema propuesto en este trabajo, que se puede visualizar en la figura 4.1. Este módulo se encarga de identificar y clasificar las diferentes actividades realizadas por el conductor durante la conducción, basándose en la información recogida de los keypoints (explicados en detalle en la subsección 4.2.1 del cuerpo y las manos. Se puede observar un diagrama del funcionamiento del modelo en

Figura 4.8: Diagrama del funcionamiento del módulo de análisis de acciones.

la figura 4.8. Para el diseño y entrenamiento del módulo de análisis de acciones, hemos seleccionado un conjunto reducido pero representativo de seis acciones principales que reflejan comportamientos relevantes durante la conducción. Las acciones consideradas son las siguientes:

- **drinking** (beber).
- **reaching side** (alcanzar un objeto hacia un lateral).
- **operating the radio** (manipular o interactuar con la radio del vehículo).
- **both hands on the wheel** (mantener ambas manos sobre el volante).
- **only left hand on the wheel** (mantener únicamente la mano izquierda sobre el volante).
- **only right hand on the wheel** (mantener únicamente la mano derecha sobre el volante).

La razón fundamental de esta elección radica en la relevancia directa de estas acciones para evaluar el nivel de atención y seguridad del conductor. Las acciones que implican retirar las manos del volante, como drinking o reaching side, son indicativas de distracción manual, un tipo de distracción que ocurre cuando el conductor aparta físicamente las manos del volante y, por tanto, pierde el control directo del vehículo. Por otro lado, operating the radio refleja una distracción cognitiva y manual simultánea, ya que no solo requiere retirar una mano del volante, sino también implica desviar la atención mental hacia la tarea secundaria, reduciendo la capacidad de concentración en la conducción.

Las etiquetas both hands on the wheel, only left hand on the wheel y only right hand on the wheel se incluyen para representar diferentes grados de control sobre el

vehículo y permitir discriminar entre situaciones seguras y potencialmente peligrosas. Asimismo, estas acciones se eligieron por su frecuencia y claridad en el dataset original, lo que facilita el entrenamiento y la validación del modelo.

Existe una relación de exclusividad lógica entre algunas de estas acciones. En particular, las tres últimas (referidas a la posición de las manos en el volante) son mutuamente excluyentes, ya que un conductor no puede tener al mismo tiempo ambas manos sobre el volante y, a la vez, solo una de ellas. Además, la acción *both hands on the wheel* resulta incompatible con las tres primeras, ya que beber, manipular la radio o alcanzar objetos obligan a retirar al menos una mano del volante.

A pesar de estas restricciones, hemos decidido abordar el problema desde una perspectiva *multilabel*, pudiendo así asignar múltiples etiquetas simultáneamente a un mismo fotograma. Esta aproximación es la ideal debido a que tenemos situaciones donde, por ejemplo, un conductor podría estar realizando una acción secundaria (como beber) mientras mantiene una sola mano en el volante (por ejemplo, la izquierda), siendo ambas acciones relevantes y coexistentes desde el punto de vista semántico.

Gracias a esta configuración, el modelo no solo puede identificar acciones aisladas, sino también interpretar combinaciones relevantes, aportando una capacidad de análisis más rica y alineada con los objetivos de seguridad y monitorización avanzada que motivan este trabajo.

4.3.1. Análisis de modelos de detección de pose

Con el propósito de seleccionar el modelo más adecuado para el sistema desarrollado en este TFG, hemos llevado a cabo un estudio comparativo entre los modelos **MediaPipe Pose** (Sección 3.3.1) y **ViTPose** (Sección 3.3.2). Para ello, hemos diseñado y desarrollado una aplicación específica capaz de ejecutar ambos modelos sobre los mismos fotogramas, asegurando así que la evaluación se realiza bajo condiciones totalmente equivalentes. La herramienta recibe como entrada un vídeo que simula la visión desde el interior del habitáculo del vehículo, y procesa cada fotograma para extraer las estimaciones de pose, podemos ver su funcionamiento en la figura 4.9.

En cada fotograma procesado, el modelo ViTPose se ejecuta inicialmente para obtener la predicción de los puntos clave corporales. Secuencialmente, el mismo fotograma es analizado por el modelo MediaPipe Pose, generando su propia predicción independiente. Posteriormente, se extraen las coordenadas normalizadas (con valores comprendidos en el rango $[0, 1]$) de los puntos clave de interés. La selección de estas regiones se centra principalmente en las extremidades superiores (hombros, codos y muñecas) y en el tronco (caderas), ya que son zonas anatómicas especialmente relevantes para la monitorización de la postura y la interacción manual durante la conducción.

La comparación detallada entre ambos modelos se lleva a cabo mediante el cálculo de la distancia euclídea normalizada entre cada par de puntos clave equivalentes detectados por MediaPipe y ViTPose en cada fotograma. Estas diferencias se agrupan y organizan en listas específicas, separadas según la articulación y el lado del cuerpo (izquierdo o derecho), lo que permite realizar un análisis estadístico granular y evaluar de forma pormenorizada la consistencia de ambos modelos para cada región anatómica evaluada. Cabe destacar que, al no disponer de un ground truth (verdadero valor de referencia) obtenido a partir de anotaciones manuales o sensores de alta precisión, este análisis se basa en la comparación directa entre ambos métodos, lo que permite

identificar discrepancias relativas, pero no cuantificar errores absolutos respecto a una referencia objetiva.

Además del análisis geométrico directo, hemos llevado a cabo un estudio estadístico complementario que incluye diversos indicadores clave, tales como:

- El porcentaje de fotogramas en los que alguno de los modelos no logra detectar correctamente los puntos clave correspondientes, indicando posibles fallos de robustez. En este caso, cabe destacar que ambos modelos han conseguido detectar los puntos clave en el 100 % de los fotogramas analizados.
- El tiempo de inferencia medio necesario para procesar un fotograma, dato crucial para aplicaciones en tiempo real. El modelo Vitpose presenta un tiempo medio de 0.034s, mientras que Mediapipe alcanza 0.039s.

Los resultados obtenidos se representan gráficamente mediante diagramas de caja (boxplots), los cuales ilustran la distribución de las distancias normalizadas para cada articulación evaluada. Este tipo de visualización resulta particularmente útil, ya que permite identificar de forma intuitiva y clara las regiones del cuerpo en las que ambos modelos presentan mayor concordancia, así como aquellas en las que podrían existir discrepancias significativas o mayor dispersión en las estimaciones.

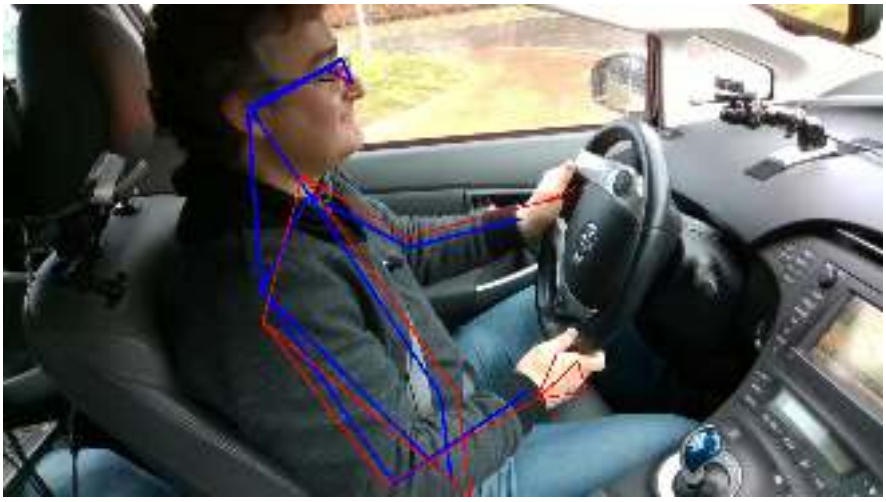


Figura 4.9: Frame con Mediapipe Pose (rojo) y Vitpose (azul)

A partir del análisis comparativo realizado y reflejado en la figura 4.10, se concluye que tanto ViTPose como MediaPipe Pose ofrecen un rendimiento notablemente similar en términos de precisión en la detección de los puntos clave corporales. Las distancias euclídeas entre los puntos clave homólogos son, en la mayoría de los casos, mínimas, lo que evidencia una alta consistencia entre ambas predicciones. Además, la mayor discrepancia se observa en la cadera izquierda, con una diferencia máxima del 8 %. Este resultado es esperable, ya que se trata de uno de los puntos más difíciles de visualizar y detectar para ambos modelos.

De manera particular, en las regiones anatómicas más críticas para el presente proyecto (como el tronco superior y las extremidades superiores), las discrepancias entre ambos modelos resultan prácticamente despreciables desde un punto de vista práctico. Esto confirma la validez de ambos enfoques para el contexto de monitorización

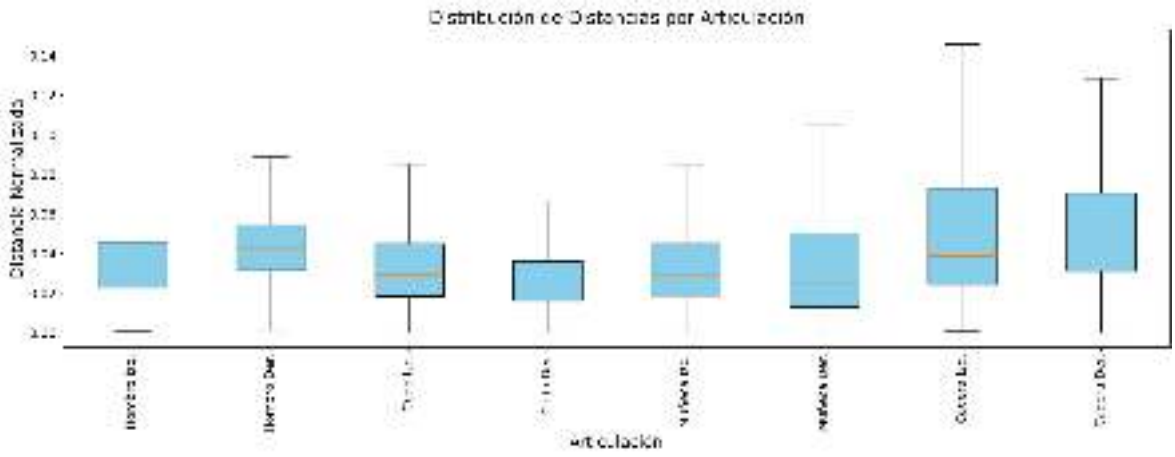


Figura 4.10: Distribución de las distancias por articulación

de conductor, especialmente en lo que respecta al análisis de la postura y la interacción manual.

No obstante, considerando factores adicionales como la eficiencia computacional, la facilidad de integración en el pipeline de procesamiento y la disponibilidad de soporte técnico y documentación, hemos decidido finalmente optar por **MediaPipe Pose** como modelo principal para el sistema desarrollado. Esta elección se justifica principalmente porque, a pesar de presentar un tiempo de inferencia ligeramente superior al de ViTPose (39ms frente a 34ms), ofrece un rendimiento igualmente robusto y preciso, y además presenta una ventaja decisiva: no requiere el uso de una GPU para su ejecución.

Esta característica resulta fundamental para el contexto de este TFG, ya que permite implementar el sistema en dispositivos con recursos computacionales limitados (por ejemplo, en entornos embebidos o sistemas integrados en el vehículo), facilitando así su despliegue real y reduciendo tanto el coste como la dependencia de hardware especializado. En definitiva, MediaPipe Pose representa la opción más adecuada y equilibrada entre precisión, eficiencia y viabilidad práctica para los objetivos planteados.

4.3.2. Clasificador

En esta subsección se describe el diseño y la configuración del modelo de clasificación empleado para identificar las acciones del conductor a partir de los puntos clave extraídos. Tras evaluar diferentes alternativas, se ha optado por utilizar un modelo **Random Forest**, dado su buen equilibrio entre precisión, robustez frente al sobreajuste y facilidad de interpretación. A continuación, se detallan las características y parámetros utilizados en este modelo, así como las razones que justifican su elección en el contexto de este trabajo.

Características elegidas para el modelo

A la hora de seleccionar las características que definen el modelo de análisis de acciones, hemos realizado un análisis detallado considerando las necesidades específicas de la tarea y las limitaciones observadas durante las primeras fases de experimentación. Finalmente, se ha determinado que son necesarias un total de 20 características (valores

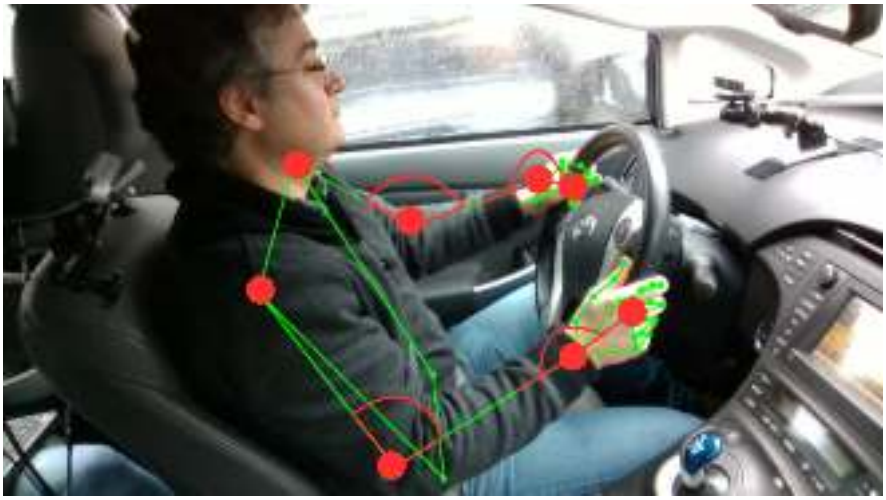


Figura 4.11: Los puntos en rojo representan las características elegidas para el modelo de acción.

X e Y de 8 puntos y 4 ángulos) para representar de manera efectiva la postura y la actividad del conductor (véase en la figura 4.11).

En primer lugar, hemos seleccionado seis puntos principales correspondientes a la pose corporal: ambos hombros, codos y muñecas. Hemos descartado los puntos asociados a las piernas y las caderas, ya que en la mayoría de las grabaciones dichas regiones se encuentran parcialmente o totalmente ocultas. Además, hemos constatado que la información proveniente de las extremidades inferiores y de la cadera no aporta valor relevante para inferir el comportamiento de las manos, que es el foco principal del análisis.

A continuación, hemos incorporado los puntos clave de ambas manos. Sin embargo, en lugar de utilizar de manera directa los 21 puntos que MediaPipe proporciona para cada mano, hemos optado por calcular el centro de masas de cada una, simplificando así la representación sin perder información esencial sobre su posición global. Este enfoque permite reducir la dimensionalidad y, al mismo tiempo, conservar la capacidad de diferenciar actividades en función de la localización aproximada de las manos.

El cálculo del centro de masas se ha implementado mediante la función que se presenta en el Código 4.1. En ella, se promedian las coordenadas x y y normalizadas de los 21 puntos para cada mano, obteniendo un único punto representativo por extremidad:

```
def calculate_center_of_mass(self, landmarks):
        x_coords = [landmark.x for landmark in landmarks]
        y_coords = [landmark.y for landmark in landmarks]

        center_x = np.mean(x_coords)
        center_y = np.mean(y_coords)

        return center_x, center_y
```

Código 4.1: Función para calcular los centros de masa de las manos

De este modo, hemos sustituído los 21 puntos individuales por un único punto

central por mano, reduciendo considerablemente la complejidad sin comprometer la interpretabilidad.

Durante la fase experimental, identificamos una posible dificultad para el modelo relacionada con la discriminación entre acciones que presentan configuraciones posturales muy similares, en concreto *operating the radio*, *both hands on the wheel* y *only right hand on the wheel*. Como se observa en la figura 4.12, las acciones *operating the radio* y *both hands on the wheel* muestran una distancia entre ambas manos muy parecida, coincidiendo en un rango significativo que complica la diferenciación por esta característica. Por otro lado, tal y como se aprecia en la figura 4.13, la posición de la mano derecha en el eje X es muy similar entre *operating the radio* y *only right hand on the wheel*, presentando también un solapamiento claro en los valores. Esta coincidencia en los rangos puede dificultar que el modelo pueda asignar correctamente la clase en base únicamente a estas medidas, generando confusiones recurrentes.

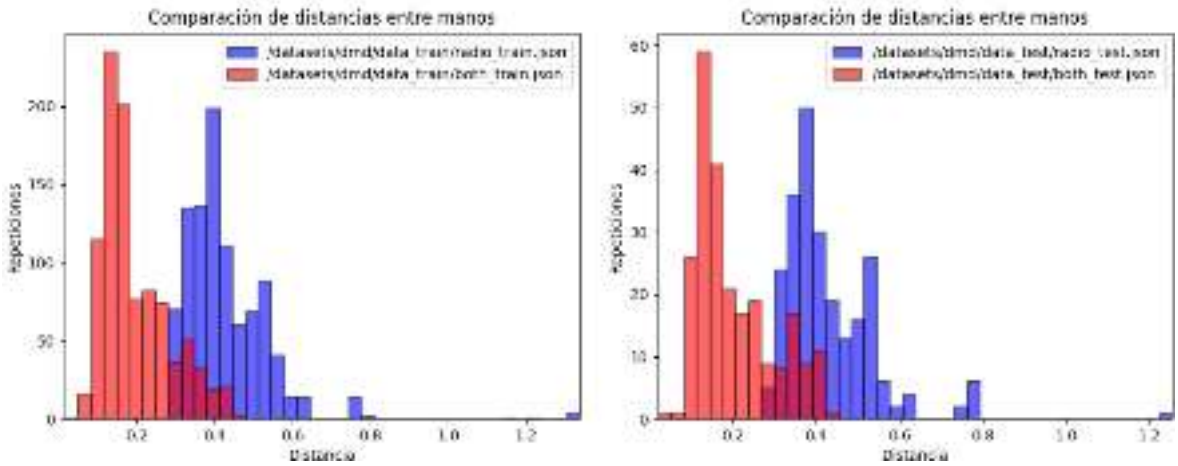


Figura 4.12: Comparación de distancias entre ambas manos de las acciones usando la radio y ambas manos al volante

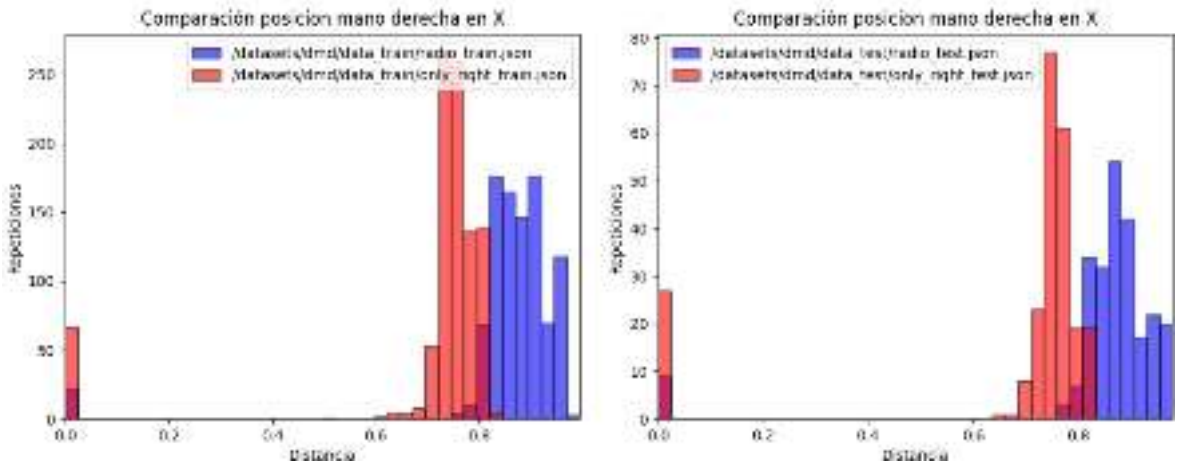


Figura 4.13: Comparación de la posición en el eje X de la mano derecha en las acciones usando la radio y mano derecha únicamente en el volante

Con el objetivo de solventar esta carencia, hemos decidido incorporar nuevas características basadas en ángulos articulares, concretamente el ángulo del brazo y el

ángulo de la muñeca, calculados para cada extremidad superior. La inclusión de estos ángulos aporta información adicional muy valiosa, ya que permite capturar diferencias sutiles pero relevantes en la postura, imposibles de identificar únicamente mediante la posición de los centros de masa. Gracias a esta mejora, el modelo puede lograr distinguir de forma más precisa entre acciones que, a nivel global, pueden parecer visualmente semejantes.

El cálculo de estos ángulos lo hemos llevado a cabo mediante la función que se presenta en el código 4.2. Dicha función recibe como entrada tres puntos: $p1$, $p2$ y un punto de referencia central (denominado *center*). A partir de estos puntos, se construyen los vectores u y v, definidos como las diferencias entre $p1$ y *center*, y $p2$ y *center*, respectivamente. Posteriormente, se calcula el coseno del ángulo θ mediante el producto escalar normalizado entre ambos vectores. Finalmente, el ángulo se expresa en grados utilizando la función $\arccos$, y se garantiza la validez numérica del coseno limitando su valor al rango $[-1, 1]$ mediante la función `np.clip`, evitando así posibles errores derivados de redondeos.

```
def calc_ang(p1, p2, center):
u, v = np.array(p1) - np.array(center), np.array(p2) - np.array(center)
cos_theta = np.dot(u, v) / (np.linalg.norm(u) * np.linalg.norm(v))
return np.degrees(np.arccos(np.clip(cos_theta, -1.0, 1.0)))
```

Código 4.2: Función para calcular el ángulo formado entre dos segmentos respecto a un punto central

Finalmente, se observó que el rendimiento del modelo podía disminuir al procesar grabaciones de personas con complexiones físicas significativamente diferentes (por ejemplo, personas más menudas o de menor altura). Este fenómeno se debe principalmente a la variabilidad en la localización absoluta de los puntos clave en el frame, que afecta negativamente a la generalización.

Para abordar este desafío, hemos diseñado un procedimiento de normalización relativa que consiste en trasladar todos los puntos clave de cada frame respecto al hombro derecho, el cual se identifica como el punto más estable y fiable en todas las grabaciones. Con este enfoque, el hombro derecho se fija en la posición $(0, 0)$, y el resto de los puntos se ajustan en consecuencia. De este modo, se consigue una representación homogénea de las poses, independiente de la complexión física o del desplazamiento de la persona en la imagen, lo que incrementa notablemente la robustez y la capacidad de generalización del modelo.

Separación en conjuntos de entrenamiento y prueba

El proceso de separación de los datos en conjuntos de entrenamiento (train) y prueba (test) se ha realizado en varias fases, adaptándose a las necesidades y limitaciones que se han ido presentando.

En una primera etapa, hemos optado por una división aleatoria del conjunto de sesiones de 12 personas. La elección de este número se justifica principalmente por razones prácticas y logísticas: estos sujetos presentan grabaciones de buena calidad, con datos completos y correctamente sincronizados en las diferentes cámaras. Concretamente, hemos asignado el 80 % de los datos al conjunto de entrenamiento y

el 20 % restante al conjunto de prueba. Esta aproximación permite realizar un primer prototipo funcional y validar de manera preliminar el rendimiento del modelo. Gracias a esta configuración inicial, hemos podido ajustar hiperparámetros, identificar posibles problemas en el preprocesamiento y establecer una base de referencia para futuras mejoras.

Una vez obtenidos resultados satisfactorios en esta fase exploratoria, se ha procedido a realizar una separación más rigurosa y representativa, basada en la identidad de los participantes y en los diferentes escenarios. Para ello, hemos seleccionado de forma aleatoria 10 participantes para conformar el conjunto de entrenamiento y hemos reservado otros 2 participantes para el conjunto de prueba. Esta estrategia tiene como objetivo evaluar la capacidad real de generalización frente a sujetos y condiciones no vistos durante el entrenamiento, asegurando que el sistema no dependa de características individuales específicas de cada persona.

Entrenamiento del modelo

Para finalizar el desarrollo del sistema, se ha procedido al entrenamiento del modelo de detección de acciones. Con el objetivo de mejorar la capacidad de generalización y evitar el sobreajuste, se han incorporado técnicas de *data augmentation* durante la fase de preparación de los datos.

En primer lugar, se ha aplicado una traslación aleatoria a los puntos clave, simulando pequeñas variaciones en la posición del conductor dentro del habitáculo. Esta transformación busca reflejar situaciones en las que el conductor no ocupa exactamente la misma posición en todos los trayectos, aumentando así la diversidad espacial de las muestras.

A continuación, se ha añadido ruido gaussiano a las coordenadas de los puntos clave, lo que permite emular leves imprecisiones propias de los modelos de detección, así como movimientos naturales y sutiles del cuerpo. Con ello, se entrena al modelo para ser más robusto frente a errores de detección o ligeras variaciones posturales.

Finalmente, en lugar de la traslación inicial, se ha optado por introducir una ligera rotación controlada, representando posibles cambios en la orientación del torso o en el ángulo de las cámaras. Esta transformación adicional enriquece todavía más la variabilidad del conjunto sintético y ayuda a cubrir casos menos frecuentes que podrían presentarse en un escenario real.

Un ejemplo visual del efecto combinado de estas técnicas se muestra en la figura 4.14. Como se puede ver en la imagen, los puntos pintados en verde dan la información obtenida de Mediapipe Pose, el rojo epresenta el ruido Gaussiano y el azul la rotación. El hombre derecho sigue representando el $(0, 0)$.

Gracias a estas transformaciones, cada muestra original genera dos variantes adicionales, lo que supone un incremento aproximado del 200 % en el tamaño final del conjunto de entrenamiento. Esta estrategia permite dotar al modelo de una mayor robustez frente a perturbaciones menores y le proporciona la capacidad de adaptarse mejor a distintas configuraciones anatómicas y posturales del conductor.

Una vez construido el conjunto aumentado, se procede al entrenamiento del modelo, incorporando hiperparámetros específicos seleccionados tras un proceso iterativo de experimentación y análisis de las primeras matrices de confusión obtenidas. En la Tabla 4.1 se resumen los valores finales establecidos, junto con una breve explicación de cada uno.

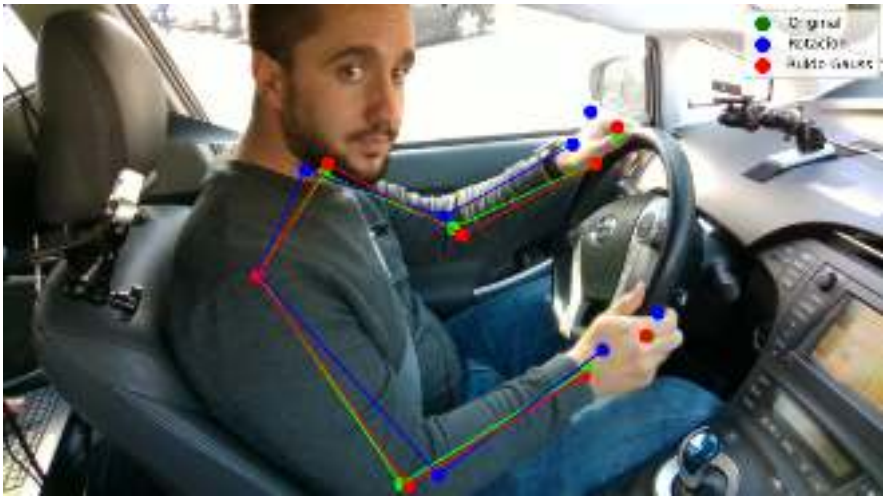


Figura 4.14: Ilustración de la técnica de *data augmentation* aplicada a los puntos clave.

Cuadro 4.1: Hiperparámetros seleccionados para el ajuste fino del modelo de análisis de acciones.

Hiperparámetro	**Valor**	**Explicación**
random_state	1	Garantiza la reproducibilidad de los resultados
class_weight	balanced	Compensa posibles desbalances entre clases
n_estimators	3	Controla el número de árboles empleados en el modelo

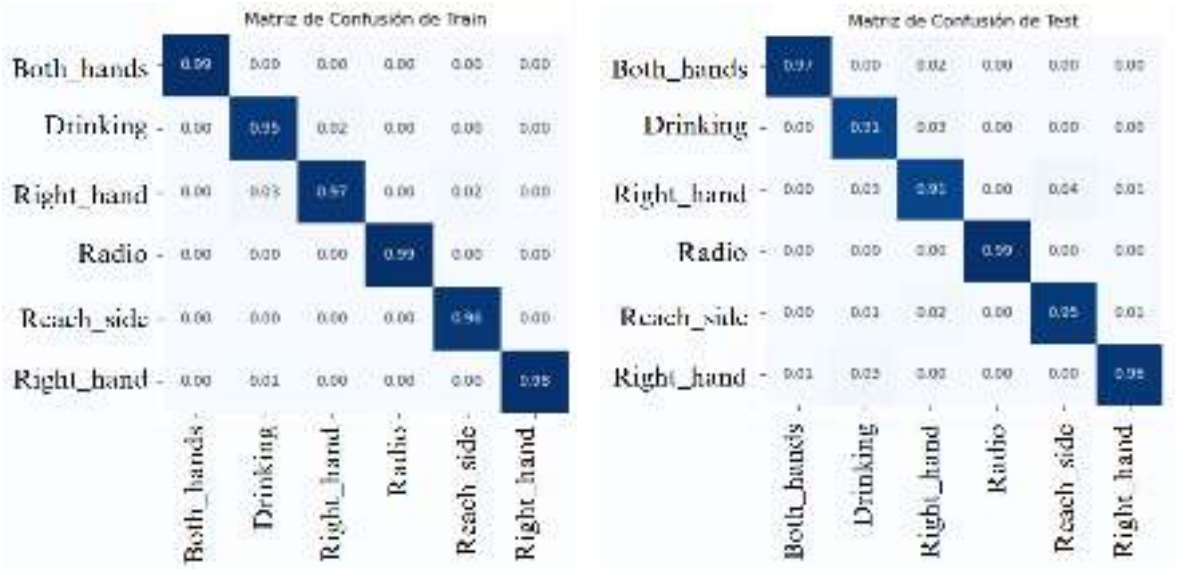
Estas configuraciones contribuyen a mejorar de forma significativa la capacidad de generalización del sistema, facilitando una clasificación más precisa y estable de las diferentes acciones evaluadas en el contexto de conducción.

Resultados del modelo

Para validar el rendimiento del modelo desarrollado, hemos empleado un conjunto de datos independiente, no utilizado durante la fase de entrenamiento. Esta estrategia permite evaluar la capacidad de generalización del modelo, es decir, su habilidad para realizar predicciones precisas sobre muestras no vistas previamente, evitando así el sobreajuste.

Con el objetivo de analizar de manera exhaustiva el comportamiento del modelo frente a este conjunto de validación, hemos utilizado la matriz de confusión como herramienta principal de evaluación. La matriz de confusión constituye una representación tabular que compara las etiquetas reales de los datos con las predicciones generadas por el modelo, proporcionando una visión detallada de la distribución de aciertos y errores. Esta herramienta resulta especialmente útil para identificar patrones de confusión entre clases específicas, detectando aquellas categorías en las que el modelo tiende a equivocarse, lo que facilita el diagnóstico y la posterior optimización del sistema.

En la figura 4.15 se muestran las matrices de confusión correspondientes a los conjuntos de train (entrenamiento) y test (prueba). Tal y como se observa, los resultados obtenidos sobre el conjunto de entrenamiento presentan un elevado número de aciertos, lo cual es esperado, dado que el modelo ha sido ajustado con estos datos.



(a) Train (b) Test

Figura 4.15: Matrices de confusión del modelo de acciones.

Sin embargo, el análisis más relevante recae en la matriz de confusión obtenida sobre el conjunto de prueba, donde se evidencia que el modelo mantiene un rendimiento robusto y consistente. Esto indica su capacidad para generalizar adecuadamente y clasificar correctamente nuevas muestras en un escenario real, validando la eficacia del enfoque propuesto.

4.4. Módulo de estimación de mirada

Figura 4.16: Diagrama del funcionamiento del módulo de estimación de mirada.

Para el diseño y entrenamiento del módulo de estimación de mirada (diagrama del módulo en la figura 4.16), hemos definido un conjunto de clases bien diferenciadas con las etiquetas del dataset, orientadas a capturar de manera precisa los puntos de atención más relevantes que puede adoptar un conductor durante la conducción.

Las direcciones seleccionadas se corresponden con zonas críticas en el habitáculo y el entorno del vehículo, y son las siguientes:

- **left mirror** (retrovisor izquierdo).
- **right mirror** (retrovisor derecho).
- **center mirror** (retrovisor central).
- **left** (izquierda).
- **right** (derecha).
- **front** (frente).
- **front right** (frente derecho).
- **steering wheel** (volante).

Cada una de estas clases refleja de manera explícita posibles puntos de enfoque visual que un conductor puede adoptar, tanto para tareas relacionadas directamente con la conducción (como mirar al frente o a los espejos) como para acciones internas en el vehículo (por ejemplo, mirar al volante). La definición clara de estas direcciones permite al sistema inferir no solo el estado de alerta del conductor, sino también posibles comportamientos de distracción o atención desviada.

A diferencia del módulo de análisis de acciones, en este caso se plantea el problema como un escenario de *multiclase* puro. Esto significa que, en cada fotograma analizado, el modelo debe asignar una única clase que represente la dirección predominante de la mirada en ese instante. Esta restricción resulta coherente con la naturaleza de la tarea, dado que un conductor no puede mirar simultáneamente en más de una dirección de manera efectiva.

4.4.1. Análisis y reducción de puntos clave faciales

Para estimar de manera precisa la dirección de la mirada del conductor, resulta imprescindible contar con una representación detallada, robusta y eficiente de la geometría facial. En el presente trabajo se ha empleado el modelo **MediaPipe Face Mesh**, una solución ampliamente utilizada que permite generar una malla densa formada por 468 puntos clave (keypoints) en coordenadas tridimensionales. Esta malla ofrece una descripción exhaustiva de la morfología facial, abarcando regiones críticas como los contornos oculares, la nariz, la boca y otras áreas anatómicas relevantes.

No obstante, el uso de la malla completa, con la totalidad de sus 468 puntos clave, implica un coste computacional considerable, además de introducir una complejidad innecesaria para el objetivo específico de este proyecto, que se centra exclusivamente en la estimación de la dirección de la mirada. Procesar un volumen tan elevado de información no solo puede ralentizar el sistema, sino que también aumenta el riesgo de sobreajuste, dado que incorpora detalles faciales secundarios que no aportan información significativa para esta tarea concreta.

Con el fin de optimizar la eficiencia y centrarse en la información estrictamente necesaria, se ha optado por realizar una reducción sustancial del conjunto de puntos clave considerados. En particular, se han seleccionado únicamente aquellos puntos que

(a) Puntos de Mediapipe Face Mesh (b) Reducción propuesta

Figura 4.17: Reducción de malla facial.

resultan esenciales para describir con precisión la posición y orientación de la región ocular, así como la estructura general del área periocular y nasal. Estas zonas son determinantes para inferir correctamente la dirección de la mirada, dado que contienen la información geométrica más relevante sobre la disposición y el ángulo de los ojos en relación con la cabeza.

Esta estrategia de reducción no solo mejora la velocidad de procesamiento y disminuye la carga computacional del sistema, sino que además favorece la generalización y la estabilidad del modelo, al eliminar variables innecesarias que podrían introducir ruido. La selección de los puntos clave se ha basado en la reducción propuesta en el artículo [56] (figura 4.17), donde se demuestra que un subconjunto reducido de vértices puede ser suficiente para representar de forma robusta y fiable ciertas expresiones o movimientos faciales específicos. En este caso, dichos puntos permiten centrarse exclusivamente en la orientación ocular, evitando el análisis de movimientos musculares u otras características faciales accesorias.

Esta aproximación encuentra su fundamento teórico en el Sistema de Codificación Facial, conocido como Facial Action Coding System (FACS), que describe los movimientos de los músculos faciales mediante unidades de acción (Action Units, AU). Aunque el FACS se emplea tradicionalmente para la codificación de expresiones y emociones, en este trabajo se adapta la filosofía de simplificación y focalización en regiones críticas para reducir la dimensionalidad de los datos y mantener únicamente la información esencial para la tarea de seguimiento ocular.

Una vez seleccionados los puntos clave más relevantes, se procede a extraer sus coordenadas normalizadas (dentro del rango $[0, 1]$), con el objetivo de garantizar que la representación obtenida sea independiente tanto de la escala como de la posición absoluta del rostro en la imagen.

Adicionalmente, y con el propósito de mejorar la capacidad de generalización del sistema frente a diferentes sujetos y condiciones de captura, se realiza un proceso de normalización geométrica adicional mediante traslación. En concreto, se selecciona el punto clave central (correspondiente al vértice ubicado en la región de la nariz) como referencia o punto de anclaje. A continuación, se efectúa una traslación de todas las coordenadas, de forma que este punto central se sitúe en el origen $(0, 0)$.

De esta manera, se consigue que la estructura relativa de los puntos clave permanezca invariante respecto a la ubicación global del rostro en el fotograma. Este paso resulta fundamental para asegurar que las variaciones en la posición del rostro no afecten a la posterior estimación de la dirección de la mirada, permitiendo que la

representación facial quede siempre alineada y centrada independientemente de la pose o desplazamiento en la escena.

4.4.2. Integración de un sistema de estimación de la dirección de la mirada

Con el objetivo de mejorar las características disponibles para la detección de la dirección de la mirada, hemos llevado a cabo una investigación que nos ha permitido encontrar un método efectivo descrito en el artículo *Eye Gaze Estimation Using a Webcam in 100 Lines of Code*[1]. Este enfoque facilita la integración en nuestro programa de una técnica capaz de estimar la dirección de la mirada con alta fiabilidad, combinando esta información con los puntos faciales extraídos previamente mediante MediaPipe Face Mesh.

La metodología propuesta enfrenta el complejo problema de determinar la dirección visual a partir de una única cámara convencional, sin necesidad de hardware especializado ni equipamiento costoso. Para ello, se parte de la detección de 468 puntos clave faciales en 2D, proporcionados por la librería MediaPipe Face Mesh, que ofrecen una representación detallada del rostro en tiempo real y con bajo coste computacional, como ya hemos hablado.

Dado que la estimación de la mirada es, por naturaleza, un problema tridimensional, pero únicamente se dispone de imágenes en 2D, se recurre al modelo geométrico de cámara estenopeica (pinhole camera model [57]), que describe la relación entre puntos del espacio tridimensional y su proyección sobre el plano bidimensional de la imagen capturada.

Se define un sistema de coordenadas 3D basado en un modelo genérico de rostro humano, estableciendo la punta de la nariz como el origen. Sobre este sistema, se seleccionan seis puntos faciales claves que se corresponden con ubicaciones equivalentes en el modelo 3D. Utilizando la función `solvePnP` de OpenCV, se calcula la rotación y traslación de la cabeza, permitiendo estimar su orientación relativa a la cámara.

Posteriormente, las coordenadas 2D de la pupila, obtenidas también a partir de MediaPipe, se proyectan en el espacio 3D mediante la función `estimateAffine3D`, que estima la posición tridimensional aproximada de la pupila a partir de su posición en la imagen. Esta aproximación, aunque no exacta, es suficiente para derivar la dirección visual relativa.

Para asegurar que la estimación sea robusta frente a movimientos y cambios de orientación de la cabeza, se ajusta la dirección de la mirada compensando el movimiento de la cabeza con la información de pose obtenida. Este ajuste consiste en calcular la intersección de la línea visual con un plano de referencia, utilizando los vectores de rotación y traslación previamente calculados.

Podemos ver una representación de lo explicado en la figura 4.18

4.4.3. Clasificador

En esta subsección se explica el planteamiento seguido para el desarrollo del modelo encargado de estimar la dirección de la mirada del conductor a partir de los puntos clave faciales (keypoints).

[1] `https://medium.com/@amit.aflalo2/eye-gaze-estimation-using-a-webcam-in-100-lines-of-code-570d4683fe23`

Figura 4.18: Visualización del funcionamiento de Gaze con sus puntos.

Para abordar este problema, se opta por un enfoque basado en **Random Forest**, un algoritmo que demuestra ser especialmente eficaz en la clasificación de patrones complejos y no lineales, como los que derivan de los movimientos oculares y faciales. En lugar de considerar únicamente la posición absoluta de los keypoints, el modelo tiene en cuenta también las relaciones relativas y las distancias entre puntos clave relevantes (por ejemplo, la disposición de los ojos respecto al centro del rostro).

En los apartados siguientes se describen las características seleccionadas como entrada al modelo, los parámetros ajustados durante el entrenamiento y las razones que motivan la elección de este enfoque frente a otras alternativas, con el fin de garantizar la fiabilidad en la detección de la dirección de la mirada y su aplicación en el contexto de la detección de distracciones.

Elección de características

El modelo ha sido entrenado utilizando un total de 58 características, compuestas por dos fuentes de información complementarias. En primer lugar, se incluyen los 27 puntos clave extraídos de la malla facial filtrada, seleccionados específicamente por su relevancia para la estimación de la orientación facial y la expresión general. Estos puntos permiten capturar la configuración geométrica fundamental del rostro, reduciendo la dimensionalidad original de 468 puntos y optimizando el rendimiento computacional sin comprometer la capacidad descriptiva.

En segundo lugar, se añaden los dos puntos derivados del módulo de Gaze, diseñados para aportar información explícita sobre la dirección de la mirada. Estos puntos se obtienen a partir de la estimación de la proyección de las pupilas en el espacio tridimensional, integrando tanto la orientación de la cabeza como la posición relativa de los ojos respecto al plano de la cámara.

La combinación de estas 58 características (valores X e Y de cada uno de los 29 puntos elegidos) proporciona al modelo un conjunto de características robusto y representativo, que permite capturar tanto la orientación global del rostro como la dirección precisa de la mirada. Podemos visualizarla en la figura 4.19.

Figura 4.19: Representación de las características elegidas para el modelo de la mirada.

Separación en conjuntos de entrenamiento y prueba

Hemos optado por implementar una partición aleatoria convencional de los datos, segmentando el conjunto total de muestras provenientes de las 13 personas disponibles en un 80 % para entrenamiento y un 20 % para prueba. Esta división asegura un reparto equilibrado y representativo de las distintas clases y participantes entre ambos subconjuntos, incrementando la diversidad y la cantidad de datos accesibles durante el entrenamiento.

Las características faciales y la geometría de los rostros presentan una alta variabilidad interindividual, lo que exige un volumen suficiente de ejemplos para que el modelo pueda abstraer las diferencias particulares y generalizar correctamente. En ausencia de esa cantidad adecuada de muestras representativas por participante, una estrategia basada en separar los datos por sujetos (dejando individuos completos fuera del entrenamiento) resulta contraproducente, ya que limita severamente la capacidad de aprendizaje y afecta negativamente al rendimiento global.

Por este motivo, aunque esta estrategia sacrifica parcialmente la evaluación estricta de generalización intersujeto, se considera una elección pragmática y necesaria dadas las limitaciones inherentes al dataset. De este modo, se garantiza que el modelo disponga de suficientes ejemplos para converger adecuadamente y se favorece el desarrollo de un sistema funcional y robusto para la estimación de la dirección de la mirada.

Aumento de datos

Con la finalidad de potenciar la capacidad de generalización del modelo y reducir la posibilidad de sobreajuste, se implementaron técnicas de aumento de datos (data augmentation) durante la preparación del conjunto de entrenamiento.

Tras evaluar diversas estrategias, se determinó que la adición de dos instancias de ruido gaussiano aleatorio sobre los puntos clave ofrecía el mejor rendimiento. De esta forma, cada muestra original generó dos nuevas variantes, incrementando en un 200 % el volumen total de datos disponibles para el entrenamiento.

Cuadro 4.2: Hiperparámetros seleccionados para el ajuste fino del modelo de estimación de mirada.

Hiperparámetro	**Valor**	**Explicación**
random_state	1	Garantiza la reproducibilidad de los resultados
class_weight	balanced	Compensa posibles desbalances entre clases
n_estimators	3	Controla el número de árboles empleados en el modelo

Entrenamiento del modelo

Finalmente, se procede al entrenamiento del modelo destinado a la detección de la dirección de la mirada. Una vez construido el conjunto de datos aumentado, se lleva a cabo el ajuste fino del modelo mediante la selección de hiperparámetros específicos, definidos tras un proceso iterativo de experimentación y análisis de las matrices de confusión preliminares. La tabla 4.2 presenta los valores finales establecidos, acompañados de una breve explicación de su función en el modelo.

Estas configuraciones contribuyeron significativamente a optimizar el desempeño del modelo, permitiendo una clasificación más estable y fiable de las diferentes direcciones de mirada.

Resultados del modelo

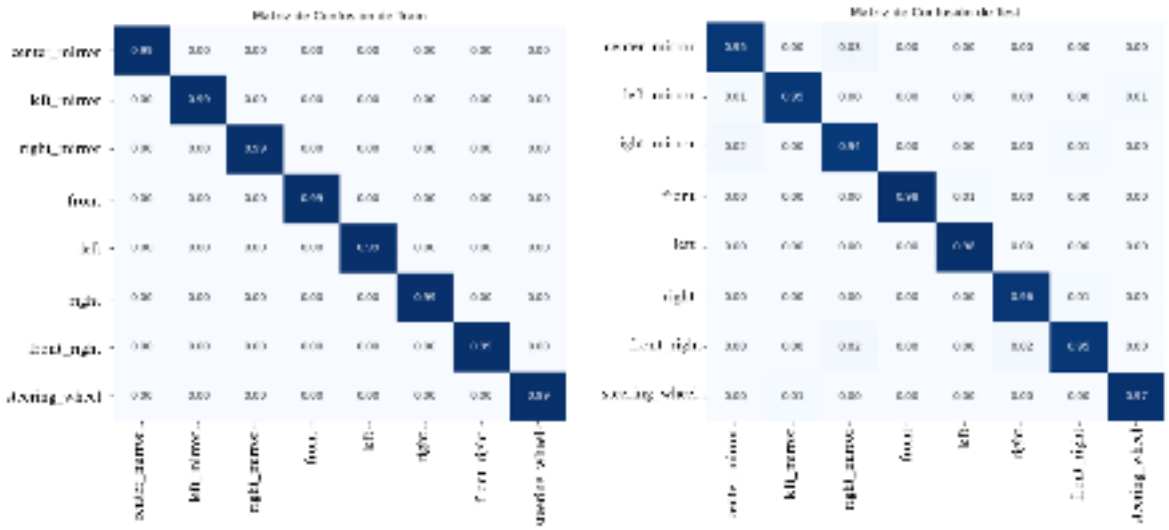
Como se ha expuesto previamente, debido a las limitaciones en el tamaño del conjunto de datos disponible, así como a la considerable variabilidad facial existente entre los distintos sujetos incluidos en el estudio, la división del conjunto de datos se realizó siguiendo una proporción del 80 % destinada a la fase de entrenamiento y un 20 % destinada a la fase de prueba.

Se puede observar que los resultados presentados en las matrices de confusión correspondientes al conjunto de entrenamiento, mostradas en la figura 4.20a, y al conjunto de prueba, ilustradas en la figura 4.20b, presentan ciertas diferencias, aunque estas no son especialmente pronunciadas. Esta relativa estabilidad en los resultados es atribuible a la estrategia de división de datos adoptada para el modelo. Asimismo, es posible inferir que el desempeño del modelo es satisfactorio, lo que indica que el sistema logra generalizar adecuadamente a partir de la información disponible, manteniendo un rendimiento consistente tanto en los datos usados para el aprendizaje como en los destinados a la evaluación.

Estos resultados confirman la capacidad del modelo para clasificar las diferentes direcciones de la mirada, permitiendo su integración efectiva en el sistema global de detección de las distracciones al volante.

4.5. Módulo de distracciones físicas

Finalmente, se decide crear un prototipo de modelo capaz de identificar objetos que pueden resultar en distracciones. Al tratarse de una prueba de concepto, se centra inicialmente en la detección de teléfonos móviles, dejando abierta la posibilidad de ampliar su funcionalidad en el futuro. La detección se basa en identificar la presencia de teléfonos en la imagen, aprovechando que el conductor es el único individuo visible



(a) Train (b) Test

Figura 4.20: Matrices de confusión del modelo de miradas.

en la escena; por lo tanto, si aparece un teléfono, se asume que está siendo manipulado por él.

Para abordar esta tarea, hemos optado por emplear el modelo **YOLO** (*You Only Look Once*), cuya arquitectura se describe en detalle en la Sección 3.4. Esta red neuronal convolucional destaca por su elevada velocidad de inferencia y su capacidad para realizar detecciones en tiempo real con una precisión notable, características que la convierten en una solución idónea para aplicaciones de monitorización en entornos dinámicos como la conducción.

La integración de YOLO en el sistema permite analizar cada fotograma y determinar de forma fiable la presencia o ausencia de teléfonos móviles, complementando así la información obtenida a partir de los modelos de clasificación de acciones y de estimación de la mirada.

A la hora de seleccionar el modelo de YOLO más adecuado para nuestro sistema, se consideran diferentes variantes disponibles en la librería, concretamente: *nano*, *medium*, *large* y *extra large*. Cada una de estas versiones ofrece un equilibrio distinto entre precisión, velocidad de inferencia y consumo de recursos computacionales.

En este contexto, se opta por la variante *extra large* debido a que es la que proporciona la mayor precisión en la detección, un aspecto fundamental para nuestro proyecto, ya que el objetivo principal es maximizar la fiabilidad y exactitud del sistema.

4.5.1. Detección específica de teléfonos móviles

El modelo YOLO, por diseño, posee la capacidad de detectar y clasificar de manera simultánea múltiples objetos presentes en una misma escena, proporcionando para cada uno de ellos tanto una etiqueta de clase como la correspondiente localización espacial mediante *bounding boxes*. No obstante, en el contexto específico de este trabajo, el interés se centra exclusivamente en identificar la presencia de teléfonos móviles, dado que su uso por parte del conductor constituye un indicador crítico de distracción al volante.

Con el objetivo de restringir el proceso de detección únicamente a los teléfonos

móviles, se procedió a identificar la clase específica asignada por YOLO a este objeto, la cual corresponde al identificador `67` dentro del conjunto de clases COCO (Common Objects in Context). A partir de esta información, el sistema ha sido diseñado para recorrer todas las detecciones generadas por YOLO en cada fotograma y verificar si alguna corresponde a la clase *cell phone*.

En la implementación desarrollada, que se encuentra en el script `phone_model.py`, se utiliza la siguiente lógica en Python. En ella, `self.model` hace referencia al modelo YOLO previamente cargado y configurado. Para cada imagen analizada, se obtienen las predicciones y se recorren todas las cajas detectadas junto con sus identificadores de clase. Si alguna de estas cajas presenta un identificador igual a 67, se interpreta como la presencia de un teléfono móvil en la escena, como podemos ver en el código 4.3

```
results = self.model(image, verbose=False)

for result in results:
boxes = result.boxes
for cls, box in zip(boxes.cls, boxes.xyxy):
if int(cls) == 67:
print("Cell phone found")
```

Código 4.3: Código para verificar la presencia de un teléfono móvil en la escena

Gracias a este proceso de filtrado específico, se logra garantizar que el sistema permanezca completamente centrado en la detección y el seguimiento del uso del teléfono móvil por parte del conductor, sin verse afectado por el resto de objetos potencialmente presentes en el entorno vehicular.

4.5.2. Problemas con las *bounding boxes*

Durante la fase de validación y prueba del módulo de detección de teléfonos móviles, hemos observado un comportamiento inesperado: el modelo YOLO genera detecciones falsas en regiones donde no hay un dispositivo móvil real. En concreto se identifica que, en numerosos fotogramas, el sistema clasifica incorrectamente la pantalla de la radio del vehículo como un teléfono móvil, generando así falsos positivos.

Para analizar con mayor precisión el origen de estas detecciones erróneas, hemos optado por visualizar cada fotograma, superponiendo sobre la imagen las *bounding boxes* generadas por YOLO. Esta visualización permite comprobar que, en muchos casos, las cajas que corresponden a la radio presentan un tamaño considerablemente superior al de las cajas asociadas a un teléfono móvil real (véase la figura 4.21).

Además de la inspección visual, hemos decidido registrar y analizar cuantitativamente el área de cada *bounding box*, calculando la superficie relativa normalizada respecto al tamaño total del fotograma. Este análisis evidencia una diferencia clara y sistemática entre las detecciones correctas y las incorrectas: las cajas que envuelven teléfonos móviles reales tienden a ocupar una superficie notablemente menor.

Con base en esta observación, hemos diseñado una estrategia de filtrado por área para descartar automáticamente las detecciones cuyo tamaño superara un umbral

(a) Clasificación incorrecta

(b) Clasificación correcta

Figura 4.21

predefinido. De esta forma, se reduce de manera significativa la tasa de falsos positivos y se mantiene el foco exclusivamente en los teléfonos móviles sostenidos o manipulados por el conductor.

A continuación, se muestra la función implementada en Python para llevar a cabo esta filtración. La función calcula el área de cada *bounding box*, la normaliza respecto al área total de la imagen y verifica si dicha área es inferior a un umbral empíricamente determinado (en este caso, 0.05). Si la condición se cumple, se considera una detección válida, como vemos en el código 4.4. Gracias a este enfoque, se logra mitigar los errores de clasificación asociados a objetos de gran tamaño presentes en el habitáculo (como la pantalla de la radio), consiguiendo que el modelo prácticamente no genere falsos positivos y garantizando así un funcionamiento más robusto y fiable del sistema de monitorización del uso del teléfono móvil en el contexto de conducción.

```
def detect(self, image):
        results = self.model(image, verbose=False)

        for result in results:
            boxes = result.boxes
            for cls, box in zip(boxes.cls, boxes.xyxy):
                if int(cls) == 67: # 67 = cell phone en COCO
                    x1, y1, x2, y2 = map(int, box)

                    area = (x2 - x1) * (y2 - y1)
                    normalized_area = area / (image.shape[0] * image.shape[1])
                    normalized_area_S = f"{normalized_area:.2f}"

                    if normalized_area < 0.05:
                        return 1
                    else:
                        break
        return 0
```

Código 4.4: Función para detectar móviles y filtrarlos por tamaño

4.6. Análisis de la distracción

Una vez desarrollados y validados satisfactoriamente los tres módulos principales del sistema, surge la necesidad de contar con un módulo adicional que integre y procese de manera conjunta la información generada por cada uno de ellos. Esta necesidad da lugar al diseño del módulo de análisis de distracciones, cuya función esencial es aprovechar los modelos previamente implementados para interpretar y fusionar los datos individuales, ofreciendo así una visión global y coherente del comportamiento del conductor. Para posibilitar un análisis detallado y una visualización clara de los resultados, se opta por almacenar toda la información correspondiente a una sesión completa de conducción en un archivo estructurado en formato JSON, asegurando la organización y accesibilidad de los datos recopilados durante el proceso.

Con el propósito de facilitar la interpretación de esta información y presentar los resultados de manera clara y comprensible, se implementa una representación gráfica que permite visualizar la evolución temporal de las distintas acciones detectadas a lo largo de la sesión. Dicha gráfica exhibe las acciones en función del tiempo, utilizando un sistema de codificación por colores intuitivo que refleja el nivel estimado de distracción del conductor en cada instante, favoreciendo así la identificación rápida y visual de patrones y momentos críticos durante la conducción.

Concretamente, el fondo de la gráfica se colorea en rojo cuando el sistema detecta que el conductor se encuentra en una situación de distracción severa; en amarillo, cuando no está completamente centrado en la carretera pero todavía no se considera una distracción crítica; y en verde, cuando se interpreta que mantiene una conducción segura y atenta, como podemos apreciar en la figura 4.22. A efectos prácticos, cualquier instante en el que se detecte la presencia de un teléfono móvil en el campo de visión se considera automáticamente como una situación de distracción crítica, por lo que se representa en color rojo, un ejemplo lo tenemos en la figura 4.25.

Por otro lado, las acciones relacionadas con el posicionamiento de las manos se interpretan con un criterio progresivo: toda acción que no implique mantener ambas manos sobre el volante se muestra en amarillo durante los primeros *3* segundos, como en la figura 4.23, indicando un nivel de atención reducido pero aún no crítico. Si dicha acción se prolonga en el tiempo más allá de este umbral, pasa a considerarse una distracción severa y se representa en rojo, como pasa en la figura 4.24.

El mismo criterio se aplica a la dirección de la mirada: cualquier desviación respecto a la mirada al frente se clasifica inicialmente en amarillo durante un periodo de 3 segundos, tras el cual se considera distracción severa si persiste, pasando a colorearse en rojo en la representación gráfica. Gracias a este enfoque visual y progresivo, se facilita la comprensión global del comportamiento del conductor y se ofrece una herramienta eficaz para identificar patrones de distracción y evaluar el nivel de atención a lo largo de toda la sesión analizada.

Podemos ver un ejemplo de una sesión entera en la siguiente gráfica 4.26. En el enlace [2] se puede ver la perspectiva de ambas cámaras durante parte de la sesión representada en la misma gráfica.

[2] `https://youtu.be/6hLyLizW14U`

Figura 4.22: Conducción atenta (verde), segundo 15.

Figura 4.23: Conducción ligeramente distraída (amarillo), segundo 122.

Figura 4.24: Conducción críticamente distraída (rojo), segundo 56.

Figura 4.25: Conducción con el móvil (rojo), perteneciente a una sesión con el coche detenido.

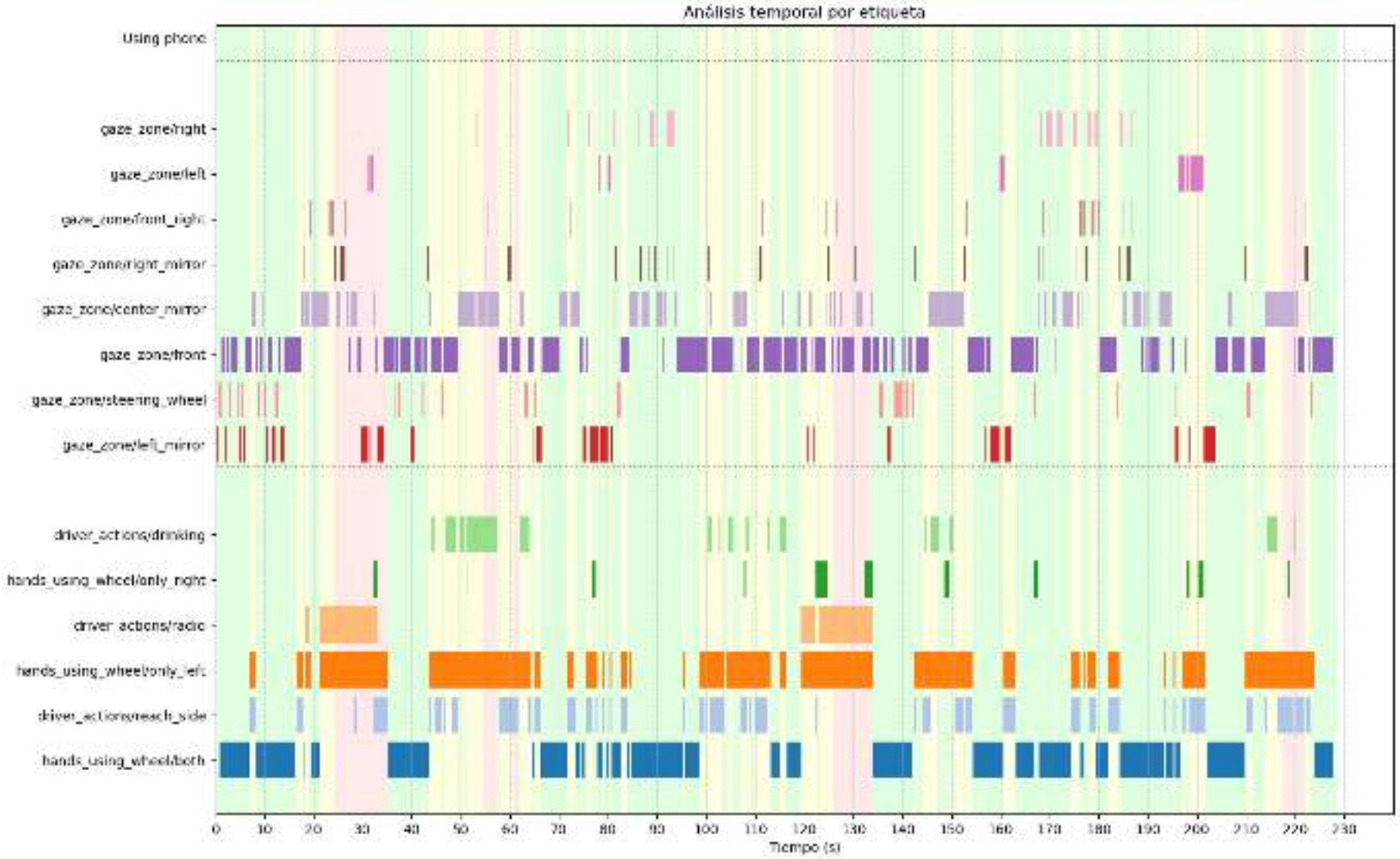


Figura 4.26: Gráfica temporal del transcurso de una sesión

Cuadro 4.3: Tiempos medios de inferencia (en milisegundos) por módulo y plataforma

Módulo	**Servidor Thor**	**Jetson AGX Orin**
Análisis de acciones	3.6	14.5
Estimación de la mirada	2.9	12.2
Detección de distracciones físicas (teléfono)	8	50
Mediapipe para pose y manos	32.1	138.6
Mediapipe para la cara	3.5	11.4
TOTAL	50,1	226,7

4.6.1. Rendimiento computacional

Se ha evaluado el rendimiento de los diferentes módulos que conforman el sistema de detección de distracción, tanto en el servidor Thor (explicado en la sección 3.6.1) como en la plataforma Jetson AGX Orin (explicada en la sección 3.6.2). Para cada caso, se midieron los tiempos medios de inferencia por predicción, obteniéndose los resultados que se recogen en la Tabla 4.3.

Es importante destacar que el tiempo medio de inferencia correspondiente al módulo de detección de distracciones físicas (uso del teléfono) en la plataforma Jetson AGX Orin se ha obtenido a partir de un vídeo publicado en YouTube[3], el cual se ha utilizado como referencia debido a que no ha sido posible realizar pruebas directas con este módulo en dicha plataforma. Por tanto, este valor debe considerarse orientativo y puede variar en función de la implementación específica y las condiciones de ejecución.

Considerando que los módulos de detección se ejecutan de forma secuencial, la latencia total por fotograma se obtiene sumando los tiempos medios de inferencia de cada uno. En el servidor Thor, esta suma da aproximadamente 0.05 segundos por frame, lo que equivale a un procesamiento cercano a 20 fps. En la Jetson AGX Orin, el tiempo total es mayor, alrededor de 0.23 segundos, correspondiente a unos 4.4 fps.

El servidor Thor ofrece una tasa de procesamiento significativamente elevada, mientras que la Jetson AGX Orin presenta una velocidad inferior debido a sus limitaciones computacionales. No obstante, esta capacidad sigue siendo adecuada para aplicaciones de monitorización del conductor que no requieren actualizaciones extremadamente rápidas. Cabe destacar que las acciones que detectamos, como beber agua o usar el móvil, normalmente tienen una duración superior al segundo, por lo que la tasa de procesamiento disponible es suficiente para capturar estos comportamientos de manera fiable y efectiva. En ambos casos, la baja latencia alcanzada resulta adecuada para mantener una monitorización constante del conductor y detectar de forma temprana posibles distracciones, contribuyendo así a mejorar la seguridad durante la conducción.

[3] `https://www.youtube.com/watch?v=QGeP-Y6KMLM&ab_channel=Ultralytics`

Capítulo 5
Conclusiones

El objetivo principal de este TFG se ha logrado satisfactoriamente, el cual consiste en el desarrollo de un sistema capaz de evaluar de manera precisa y fiable el nivel de distracción al volante de un conductor. Para alcanzar este fin, se han planteado como subobjetivos la creación de un módulo de análisis de acciones manuales, un módulo de estimación de la dirección de la mirada y un módulo de detección de distracciones físicas. Asimismo, se ha diseñado un método de representación de resultados que permite mostrar la información de forma clara, estructurada y fácilmente interpretable, facilitando así su análisis y comprensión. A lo largo del trabajo se ha puesto de manifiesto el potencial de este tipo de soluciones para contribuir a la mejora de la seguridad vial y se ha abierto la puerta a posibles desarrollos y aplicaciones futuras en el ámbito de la automoción inteligente.

En este capítulo se recogen y analizan los objetivos y requisitos planteados al inicio. Además, se describen las posibles líneas de trabajo futuro que podrían continuar y ampliar los avances realizados en este proyecto.

5.1. Objetivos cumplidos

En la sección 2.1 se expusieron los objetivos principales que se plantearon al inicio de este TFG. A continuación, se detallan los logros alcanzados y cómo estos contribuyen al cumplimiento de dichos objetivos:

1. Se ha desarrollado un sistema basado en Random Forest capaz de estimar de manera fiable la dirección de la mirada de distintos conductores, permitiendo identificar si la atención visual está orientada hacia la carretera u otros elementos del entorno.

2. Se ha implementado un modelo robusto, también basado en Random Forest, para la detección de seis acciones diferenciadas de las manos, proporcionando información clave sobre la postura y el nivel de implicación manual durante la conducción.

3. Se ha integrado un sistema eficaz para la detección de la presencia de teléfonos móviles en los frames captados utilizando YOLO, posibilitando así la identificación de una de las principales causas de distracción al volante.

4. Finalmente, se ha diseñado un módulo unificador capaz de coordinar los modelos anteriores, ofreciendo un análisis global del comportamiento del conductor. Este

sistema integrado permite determinar de forma automática y con baja latencia si el conductor se encuentra plenamente atento a la conducción, ligeramente distraído o completamente distraído.

5.2. Requisitos satisfechos

En la sección 2.2 propusimos una serie de requisitos que debía cumplir nuestro proyecto, ahora vamos a analizar si están satisfechos:

1. El sistema ha sido implementado con un flujo de procesamiento optimizado y módulos ligeros, lo que permite mantener una latencia baja y una respuesta ágil. De este modo, se garantiza la actualización continua y fluida del estado del conductor, cumpliendo los requisitos de reactividad y monitorización constante.

2. Se ha logrado desarrollar dos de los módulos de forma ligera, permitiendo su ejecución sin necesidad de hardware especializado (como una GPU), lo que facilita su uso en dispositivos embebidos y sistemas con recursos limitados. Únicamente el módulo de detección de teléfonos móviles presenta una mayor demanda computacional, derivada del uso del modelo YOLO, cuyo rendimiento óptimo se obtiene con aceleración por GPU.

3. Gracias a que el dataset empleado está formado por grabaciones de personas reales en un entorno de conducción auténtico, el sistema ha sido entrenado y validado con datos que reflejan fielmente la realidad, cumpliendo así este requisito de representatividad.

4. La inclusión de técnicas de aumento de datos mediante la adición de ruido gaussiano ha permitido reducir la dependencia de una colocación exacta de la cámara. De este modo, no es necesario que el dispositivo esté ubicado en unas coordenadas estrictamente definidas, sino que resulta suficiente con que se encuentre en la zona aproximada prevista dentro del vehículo, facilitando así su integración práctica.

5. Finalmente, el sistema ha sido probado de manera satisfactoria en un dispositivo Jetson AGX Orin, demostrando su capacidad de funcionamiento estable y eficiente en plataformas embebidas, lo que confirma la viabilidad de su uso en escenarios reales y su potencial de aplicación en el ámbito de la automoción inteligente.

5.3. Balance global y competencias adquiridas

A lo largo de este TFG, asumir el reto de crear un sistema que permita evaluar la distracción de un conductor durante la conducción se ha convertido en una experiencia exigente, pero también muy valiosa y formativa. A pesar de no contar inicialmente con experiencia previa en este ámbito específico, el esfuerzo constante y la dedicación me han permitido adentrarme en el mundo de los sistemas de detección de distracciones de conductores y comprender en profundidad su relevancia.

A lo largo de este trabajo he podido constatar que se trata de un campo con un enorme potencial de crecimiento y múltiples aplicaciones futuras. Estoy convencida

de que, con un mayor tiempo de investigación y un estudio más exhaustivo, estas tecnologías podrían evolucionar hasta convertirse en herramientas esenciales que contribuyan de manera significativa a la mejora de la seguridad vial y a la conducción del futuro.

Quiero destacar el adquirimiento de las siguientes competencias:

- Perfeccionamiento de la elaboración de representaciones visuales claras y eficaces para la interpretación de resultados en entornos aplicados
- Fortalecimiento de la autonomía investigadora mediante la búsqueda, comprensión e integración de publicaciones y recursos científicos relevantes.
- Evolución de la capacidad de documentar de forma precisa y estructurada los procesos técnicos desarrollados, fomentando la claridad y la transferencia de conocimiento.
- Mejora en el análisis crítico de métricas de rendimiento (por ejemplo, matrices de confusión), identificando limitaciones del modelo y proponiendo estrategias de mejora.
- Mejora en la comprensión y aplicación de técnicas de *data augmentation* para optimizar la generalización de modelos de aprendizaje automático.
- Desarrollo de habilidades avanzadas en la integración de modelos heterogéneos para su funcionamiento conjunto en un sistema unificado.
- Profundización en el manejo de herramientas de visión por computador, como *MediaPipe*, y en el uso de redes de detección rápida (por ejemplo, YOLO) para aplicaciones en tiempo real.

5.4. Líneas futuras

A partir de los resultados obtenidos y de la experiencia adquirida durante el desarrollo de este TFG, se pueden identificar diversas líneas de trabajo que permitirían continuar y enriquecer el proyecto en el futuro. A continuación, se detallan algunas propuestas destacadas:

- **Entrenamiento con conjuntos de datos adicionales:** Una de las mejoras más relevantes consistiría en entrenar los modelos actuales utilizando nuevos conjuntos de datos (datasets) que presenten mayor diversidad en cuanto a condiciones de iluminación, tipos de vehículo, perfiles de conductores y escenarios de conducción. Esta ampliación permitiría incrementar la capacidad de generalización del sistema, haciéndolo más robusto y fiable en contextos reales y variados.
- **Ampliación del módulo de detecciones físicas:** En futuras versiones, el módulo de detecciones físicas podría extenderse para identificar una mayor variedad de objetos dentro del vehículo. De este modo, además de detectar teléfonos móviles, sería posible reconocer otros elementos como botellas, maquillaje o cigarrillos, que también pueden influir negativamente en la atención y seguridad del conductor. Esta ampliación permitiría abarcar un rango más amplio de situaciones de distracción, mejorando la capacidad preventiva del sistema.

- **Personalización por conductor:** Otra línea futura interesante sería dotar al sistema de la capacidad de reconocer de forma individual a cada conductor y aprender progresivamente de sus patrones y técnicas de conducción. De este modo, se podrían generar análisis y recomendaciones mucho más personalizadas, adaptadas a los hábitos concretos de cada usuario, incrementando la precisión del sistema y fomentando una mayor confianza y aceptación por parte de los conductores.

El desarrollo de estas líneas futuras no solo contribuirían a mejorar la precisión y la funcionalidad del sistema, sino que también abrirían la puerta a nuevas aplicaciones en el ámbito de la seguridad vial y la automoción inteligente.

Bibliografía

[1] Proyecto IDIS, "Motion capture." Página web (Proyecto IDIS). https://proyectoidis.org/motion-capture/.

[2] Trigital, "Mocap: cómo funciona la captura de movimiento y qué aplicaciones tiene para la industria." Artículo en línea (Trigital). https://trigital.es/mocap-como-funciona-la-captura-de-movimiento-y-que-aplicaciones-tiene-para-l

[3] H. Navarro, "Pose estimation made effortless: Simplifying key point detection and spatial understanding." Artículo en línea (henrynavarro.org), 09 2023. https://henrynavarro.org/pose-estimation-made-effortless-simplifying-key-point-detection-and-spatial-

[4] Sensor Medica, "Análisis de postura." Página web (Sensor Medica). https://www.sensormedica.mx/anlisis-de-postura.

[5] Euro NCAP, "Towards vision zero: The future of driver monitoring." https://cdn.euroncap.com/media/77182/27esv-000289.pdf, 2023.

[6] GeeksforGeeks, "Random forest algorithm in machine learning." Artículo en línea (GeeksforGeeks). https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/.

[7] Microsoft, "Tutorial: Comenzar con visual studio code." Página web (Visual Studio Code). https://code.visualstudio.com/docs/getstarted/getting-started.

[8] Google AI, "Guía de detección de puntos de referencia de la postura." Página web (Google AI Edge). https://ai.google.dev/edge/mediapipe/solutions/vision/pose$_l$*andmarker*?*hl* = *es* − 419.

[9] Google AI Edge, "Guía de detección de puntos de referencia faciales." Página web (Google AI Edge). Accedido: 10 de julio de 2025.

[10] X. Chu, Z. Zhu, Z. Li, and J. Zhang, "Fighting fire with fire: Using antagonistic samples to improve human parsing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

[11] L. Belokon, "Improving real-time object detection with yolo." DataScienceCentral. https://www.datasciencecentral.com/improving-real-time-object-detection-with-yolo/.

[12] Vicomtech, "Driver monitoring dataset (dmd)." Página web (Vicomtech). https://dmd.vicomtech.org/.

[13] R. Szeliski, *Computer Vision: Algorithms and Applications*. Berlin, Heidelberg: Springer-Verlag, 1st ed., 2010.

[14] EDS Robotics, "Sistemas de visión artificial: tipos y aplicaciones," 2023. https://www.edsrobotics.com/blog/sistemas-de-vision-artificial-tipos-aplicaciones/.

[15] E. Pérez, "La carretera a ojos del autopilot de tesla: esto es todo lo que capta su sistema de conducción autónoma." Artículo en línea (Xataka), 01 2020. https://www.xataka.com/vehiculos/carretera-a-ojos-autopilot-tesla-esto-todo-que-capta-su-sistema-conduccion-autonoma.

[16] L. M.Morente, "Todo lo que debes saber sobre da vinci, el robot quirúrgico," 04 2017. https://www.expansion.com/tecnologia/2017/04/15/58f24ada22601d67308b460b.html.

[17] Interempresas, "Drones y visión artificial para la estimación de producción en cítricos." Artículo en línea, 06 2019. https://www.interempresas.net/Grandes-cultivos/Articulos/246173-Drones-y-vision-artificial-para-la-estimacion-de-produccion-en-citricos.html.

[18] A. Portal, "La visión artificial aplicada a la videovigilancia." Artículo en línea (Cuadernos de Seguridad), 10 2022. https://cuadernosdeseguridad.com/2022/10/la-vision-artificial-aplicada-a-la-videovigilancia/.

[19] Ultralytics, "Computer vision in logistics." Página web (Ultralytics), 2025. https://www.ultralytics.com/solutions/ai-in-logistics.

[20] Ciencia Canaria, "Perseverance: el siguiente paso de la conquista de marte." Artículo en línea (Ciencia Canaria), 02 2021. `https://www.cienciacanaria.es/secciones/a-fondo/1140-perseverance-el-siguiente-paso-de-la-conquista-de-marte`.

[21] Instituto de Biomecánica de Valencia (IBV), "Deep lab: deep learning aplicado al análisis biomecánico de movimientos humanos." Página web (IBV), 2022. `https://www.ibv.org/proyecto/deep_lab-deep-learning-aplicado-al-analisis-biomecanico-de-movimientos-humanos-2/`.

[22] M. Mundt, S. Colyer, L. Wade, L. Needham, M. Evans, E. Millett, and J. Alderson, "Automating video-based two-dimensional motion analysis in sport? implications for gait event detection, pose estimation, and performance parameter analysis," *Scandinavian journal of medicine science in sports*, vol. 34, no. 7, pp. e14693–n/a, 2024.

[23] T. Rangari, S. Kumar, P. P. Roy, D. P. Dogra, and B.-G. Kim, "Video based exercise recognition and correct pose detection," *Multimedia Tools and Applications*, vol. 81, no. 21, pp. 30267–30282, 2022.

[24] C. Arrowsmith, D. Burns, T. Mak, M. Hardisty, and C. Whyne, "Physiotherapy exercise classification with single-camera pose detection and machine learning," *Sensors (Basel, Switzerland)*, vol. 23, no. 1, p. 363, 2022.

[25] M. Hoffmann, E. Abaci Turk, B. Gagoski, L. Morgan, P. Wighton, M. D. Tisdall, M. Reuter, E. Adalsteinsson, P. E. Grant, L. L. Wald, and A. J. W. Kouwe, "Rapid head-pose detection for automated slice prescription of fetal-brain mri," *International journal of imaging systems and technology*, vol. 31, no. 3, pp. 1136–1154, 2021.

[26] S. R. Modi, H. K. Gevariya, R. Dayma, A. V. Panchal, and H. L. Chaudhary, "Augmented and virtual reality based segmentation algorithm for human pose detection in wearable cameras augmented and virtual reality based segmentation algorithm for human pose detection in wearable cameras," *Measurement. Sensors*, vol. 36, p. 101402, 2024.

[27] A. Liszewski, "Estas cámaras inteligentes para el auto pueden detectar lo que estás haciendo mientras conduces," *Gizmodo en Español*, Sept. 2021. Consultado el 11 de julio de 2025.

[28] Xataka, "Del joystick a jugar con el cuerpo." Artículo en línea (Xataka), 03 2013. https://www.xataka.com/videojuegos/del-joystick-a-jugar-con-el-cuerpo.

[29] C. Vera, "Sistema de detección y análisis de emociones y comportamientos de estudiantes en un aula docente basado en deep learning," trabajo de fin de grado, Universidad Rey Juan Carlos de Madrid, Madrid, España, 2024.

[30] M. Menanno, C. Riccio, V. Benedetto, F. Gissi, M. M. Savino, and L. Troiano, "An ergonomic risk assessment system based on 3d human pose estimation and collaborative robot," *Applied Sciences*, vol. 14, no. 11, 2024.

[31] Ultralytics, "Visión artificial en la fabricación." Página web (Ultralytics). `https://www.ultralytics.com/es/solutions/ai-in-manufacturing`.

[32] Y. Kobayashi, S. Saito, and T. Murahori, "Classification of fashion models' walking styles using publicly available data, pose detection technology, and multivariate analysis: From past to current trendy walking styles," *Sensors (Basel, Switzerland)*, vol. 24, no. 12, p. 3865, 2024.

[33] A. Kashevnik, R. Shchedrin, C. Kaiser, and A. Stocker, "Driver distraction detection methods: A literature review and framework," *IEEE Access*, vol. 9, pp. 60063–60076, 2021.

[34] K. A. Brookhuis and D. de Waard, "Monitoring drivers' mental workload in driving simulators using physiological measures," *Accident analysis and prevention*, vol. 42, no. 3, pp. 898–903, 2010.

[35] M. Malik, R. Nandal, S. Dalal, V. Jalglan, *et al.*, "Deriving driver behavioral pattern analysis and performance using neural network approaches.," *Intelligent Automation & Soft Computing*, vol. 32, no. 1, 2022.

[36] S. F. A. Razak, S. Yogarayan, A. A. Aziz, M. F. A. Abdullah, and N. H. Kamis, "Physiological-based driver monitoring systems: A scoping review," *Civil Engineering Journal*, vol. 8, no. 12, pp. 3952–3967, 2022.

[37] S. Begum, "Intelligent driver monitoring systems based on physiological sensor signals: A review," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 282–289, 2013.

[38] H. Hayashi, M. Kamezaki, and S. Sugano, "Toward health-related accident prevention: Symptom detection and intervention based on driver monitoring and verbal interaction," *IEEE open journal of intelligent transportation systems*, vol. 2, pp. 240–253, 2021.

[39] K. S, S. S. Nath, N. T, and B. S. M, "Enhancing heavy vehicle safety with intelligent driver behaviour monitoring via multi-sensor fusion and embedded systems," in *2024 International Conference on System, Computation, Automation and Networking (ICSCAN)*, pp. 1–5, 2024.

[40] Y. Liu, C. Yu, and X. Xiao, "Study of wireless safety monitoring system for driver based on information fusion technology," in *Proceedings of 2013 IEEE International Conference on Vehicular Electronics and Safety*, pp. 23–26, 2013.

[41] Y. Qu, H. Hu, J. Liu, Z. Zhang, Y. Li, and X. Ge, "Driver state monitoring technology for conditionally automated vehicles: Review and future prospects," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–20, 2023.

[42] Tesla, "Advertencia de somnolencia del conductor." Manual del propietario del Model 3. https://www.tesla.com/ownersmanual/model3/es$_e$s/$GUID-65BF21B8-50C5-4FA5-86A4-DA363DCD0484.html$.

[43] Tesla, "Cámara del habitáculo." Manual del propietario del Model 3. https://www.tesla.com/ownersmanual/model3/es$_e$u/$GUID-EDAD116F-3C73-40FA-A861-68112FF7961F.html$.

[44] Mercedes-Benz, "¿cómo funciona el sistema attention assist® de mercedes-benz?." Página
web (Mercedes-Benz of Easton). https://www.mercedesbenzofeaston.com/mercedes-benz-attention-assist/.

[45] Mercedes-Benz of Arrowhead, "How does the driver camera feature in the new mercedes-benz e-class sedan help?." Página web (Mercedes-Benz of Arrowhead). https://www.arrowheadmb.com/blog/how-does-the-driver-camera-feature-in-the-new-mercedes-benz-e-class-sedan-help/.

[46] Subaru, "Driver monitoring system – driver focus." Página web (Subaru Australia). https://www.subaru.com.au/driver-monitoring-system.

[47] Toyota, "T-mate driving assistance." Página web (Toyota Europe). https://www.toyota-europe.com/brands-and-services/toyota/t-mate-driving-assistance.

[48] S. K. P S, A. B. H, A. M C, R. Ilambirai, S. L. Jame, and R. Pongiannan, "Development of artificial intelligence system for driver drowsiness detection," in *2025 International Conference on Multi-Agent Systems for Collaborative Intelligence (ICMSCI)*, pp. 1812–1816, 2025.

[49] A. Kashevnik, I. Lashkov, A. Ponomarev, N. Teslya, and A. Gurtov, "Cloud-based driver monitoring system using a smartphone," *IEEE Sensors Journal*, vol. 20, no. 12, pp. 6701–6715, 2020.

[50] Z. Li, C. Chen, J. Zhang, B. Chen, and Y. Xiang, "Deep attention network for driver attention prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4685–4694, 2019.

[51] D. L. Brown, "Mediapipe hands: 21 landmarks." Figura en ResearchGate. https://www.researchgate.net/figure/MediaPipe-Hands-21-landmarks-$13_fig1_362871842$.

[52] Google AI Edge, "Guía de detección de puntos de referencia de la mano en python." Página web (Google AI Edge). https://ai.google.dev/edge/mediapipe/solutions/vision/$hand_landmarker/python$.

[53] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision (ECCV)*, pp. 740–755, Springer, 2014.

[54] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[55] J. D. Ortega, N. Kose, P. Cañas, M.-A. Chao, A. Unnervik, M. Nieto, O. Otaegui, and L. Salgado, "DMD: A Large-Scale Multi-modal Driver Monitoring Dataset for Attention and Alertness Analysis," in *Computer Vision – ECCV 2020 Workshops* (A. Bartoli and A. Fusiello, eds.), pp. 387–405, Cham: Springer International Publishing, 2020.

[56] A. I. Siam, N. F. Soliman, A. D. Algarni, F. E. Abd El-Samie, and A. Sedik, "Deploying machine learning techniques for human emotion detection," *Computational intelligence and neuroscience*, vol. 2022, pp. 1–16, 2022.

[57] I. De Boi, S. Pathak, M. Oliveira, R. Penne, S. Paredes, I. Domingues, and V. Vasconcelos, "How to turn your camera into a perfect pinhole model," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, vol. 14469 of *Lecture Notes in Computer Science*, pp. 90–107, Switzerland: Springer, 2023.

[58] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artificial Intelligence Review*, vol. 57, no. 4, p. 99, 2024.

[59] M. Miyaji, M. Danno, and K. Oguri, "Analysis of driver behavior based on traffic incidents for driver monitor systems," in *2008 IEEE Intelligent Vehicles Symposium*, pp. 930–935, 2008.

[60] W.-H. Cheng, S. Song, C.-Y. Chen, S. C. Hidayati, and J. Liu, "Fashion meets computer vision: A survey," *ACM computing surveys*, vol. 54, no. 4, pp. 1–41, 2022.

[61] A.-M. Ilisei and L. Bruzzone, "A method for automatic three-dimensional reconstruction of ice sheets by using radar sounder and altimeter data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 2, pp. 401–415, 2018.