UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE GUAYAQUIL
CARRERA DE MECATRÓNICA

IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN DE SOMNOLENCIA PARA CONDUCTORES DE VEHÍCULOS MEDIANTE VISIÓN ARTIFICIAL

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Mecatrónica

AUTORES: Angie Fernanda Lema Pilco
TUTOR: Nino Tello Vega Ureta

Guayaquil - Ecuador
2024 - 2025

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, **Angie Fernanda Lema Pilco** con documento de identificación Nº **0944022276**; manifiesto que:

Soy la autora y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo.

Guayaquil, 11 de febrero del año 2025

Atentamente,

A Lema.

Angie Fernanda Lema Pilco
0944022276

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, **Angie Fernanda Lema Pilco** con documento de identificación N.° **0944022276**, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autora del **Dispositivo Tecnológico: IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN DE SOMNOLENCIA PARA CONDUCTORES DE VEHÍCULOS MEDIANTE VISIÓN ARTIFICIAL**, el cual ha sido desarrollado para optar por el título de: Ingeniero en Mecatrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 11 de febrero del año 2025

Atentamente,

Angie Fernanda Lema Pilco
0944022276

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, **Nino Tello Vega Ureta**, docente de la Universidad **Politécnica Salesiana, declaro que bajo mi tutoría fue** desarrollado el trabajo de titulación: **IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN DE SOMNOLENCIA PARA CONDUCTORES DE VEHÍCULOS MEDIANTE VISIÓN ARTIFICIAL**, realizado por **Angie Fernanda Lema Pilco con documento** de identificación **Nº 0944022276**, obteniendo como resultado final el trabajo de titulación bajo la opción **Dispositivo Tecnológico** que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 11 de febrero del año 2025

Atentamente,

Ing. Nino Tello Vega Ureta, Mgtr.
081602160

DEDICATORIA

Este trabajo y cada logro que obtenga con él están dedicados unica y exclusivamente con todo mi corazón a mi madre, la señora Lourdes Pilco Chafla. La mujer que me dio la vida, que ha sido mi mayor fortaleza, mi refugio y el mejor ejemplo a seguir. Gracias a su amor incondicional y a su esfuerzo incansable, hoy puedo perseguir mis sueños. Su mayor satisfacción ha sido siempre verme triunfar, y este logro es para ella, en agradecimiento por dedicarme toda su vida.

Angie Fernanda Lema Pilco.

AGRADECIMIENTO

Quiero expresar mi más sincero agradecimiento a todas las personas que, de una u otra manera, contribuyeron a la realización de esta tesis.

En primer lugar, a mi madre, Lourdes Pilco, por ser mi mayor inspiración y fortaleza. Su apoyo incondicional, esfuerzo y sacrificio han sido el pilar sobre el que he construido mi camino. Gracias por cada palabra de aliento, por creer en mí incluso cuando yo misma dudé, y por brindarme siempre amor y oportunidades. Su ejemplo de lucha y dedicación es el motor que me impulsa a seguir adelante.

A mi padre, Carlos Lema, por su esfuerzo y trabajo, que me permitió contar con el respaldo necesario para completar mis estudios. Su apoyo ha sido fundamental para que pudiera concentrarme plenamente en mi formación y en el desarrollo de esta investigación.

A Josué Vera, por su compañía constante, comprensión y apoyo incondicional a lo largo de este proceso. Su paciencia y motivación fueron esenciales en los momentos más difíciles, ayudándome a seguir adelante con determinación.

Finalmente, a los profesores de la Universidad Politécnica Salesiana, cuyo conocimiento y guía fueron clave en mi crecimiento académico y en la realización de este trabajo.

A todos ustedes, mi más profundo agradecimiento.

Angie Fernanda Lema Pilco.

RESUMEN

El objetivo del presente trabajo de titulación es aplicar los conceptos y teorías de la visión artificial y el aprendizaje profundo a la seguridad vial mediante el desarrollo de un sistema de detección de somnolencia en conductores. Para la implementación del sistema, se empleó una Raspberry Pi junto con una cámara para la captura de imágenes en tiempo real. El procesamiento de las imágenes se realizó mediante algoritmos de detección de rasgos faciales y análisis del estado ocular, con el fin de identificar signos de fatiga, como el cierre prolongado de los párpados.
El sistema cuenta con un módulo de alerta que emite señales acústicas en caso de detectar estados críticos de somnolencia en el conductor. La implementación se diseñó para ser eficiente en términos de consumo energético y adaptable a distintos entornos de conducción. El trabajo concluyó con la validación experimental del sistema, evaluando su desempeño en diversas condiciones de iluminación y características faciales.
Palabras claves: Visión artificial, detección de somnolencia, procesamiento de imágenes, Raspberry Pi, seguridad vial.

ABSTRACT

The objective of this degree work is to apply the concepts and theories of computer vision and deep learning to road safety through the development of a driver drowsiness detection system. For the system implementation, a Raspberry Pi was used along with a camera to capture real-time images. Image processing was carried out using facial feature detection algorithms and eye state analysis to identify fatigue signs such as prolonged eyelid closure.

The system includes an alert module that emits acoustic signals upon detecting critical levels of driver drowsiness. Its implementation was designed to be energy-efficient and adaptable to various driving environments. The study concluded with the experimental validation of the system, assessing its performance under different lighting conditions and facial characteristics.

Key words: Computer vision, drowsiness detection, image processing, Raspberry Pi, road safety.

Índice

Índice de figuras

Índice de Tablas

I. INTRODUCCIÓN

En la seguridad vial, los accidentes de tránsito constituyen una de las principales causas de mortalidad a nivel mundial, siendo la somnolencia al volante un factor determinante en una gran proporción de estos siniestros. La creciente urbanización y la expansión de las redes de transporte han incrementado la exposición de conductores a largas jornadas y condiciones de manejo exigentes, lo que subraya la necesidad urgente de implementar tecnologías innovadoras que mitiguen este riesgo.

En los últimos años, los avances en visión artificial, procesamiento de datos y asistentes virtuales han revolucionado diversos sectores tecnológicos, incluyendo la industria automotriz. Estas herramientas han permitido el desarrollo de sistemas inteligentes capaces de monitorear el comportamiento humano en tiempo real y ofrecer soporte interactivo a los conductores, abriendo nuevas posibilidades para mejorar la seguridad en los vehículos. La integración de dispositivos como la Raspberry Pi, junto con cámaras de alta sensibilidad, y la incorporación de asistentes virtuales capaces de interactuar y alertar al usuario, ha hecho posible el diseño de prototipos accesibles y eficaces que responden a estas necesidades.

Diversos estudios han explorado la implementación de sistemas de detección de fatiga basados en características faciales, como el cierre prolongado de los ojos, la frecuencia de parpadeo y los movimientos de la cabeza. Estos enfoques han demostrado ser efectivos en la identificación temprana de signos de somnolencia. Sin embargo, la integración de asistentes virtuales en estos sistemas añade un nivel adicional de interacción, permitiendo no solo alertar al conductor, sino también ofrecer estímulos auditivos y visuales personalizados que ayuden a mantener su atención y evitar la monotonía durante la conducción.

La presente investigación aborda estos desafíos mediante el diseño e implementación de un detector de somnolencia en vehículos basado en visión artificial, complementado con un asistente virtual que refuerza las alertas y proporciona apoyo interactivo en tiempo real. Este sistema utiliza algoritmos avanzados de procesamiento de imágenes y aprendizaje automático para analizar patrones faciales, adaptándose a diferentes condiciones de conducción y tipos de usuarios. El objetivo principal es contribuir a la seguridad vial mediante un dispositivo que combine eficiencia, portabilidad y bajo costo, facilitando su integración en vehículos comerciales y particulares.

A nivel global, los sistemas de monitoreo del conductor están ganando relevancia en la transición hacia vehículos más seguros e inteligentes. Según la Organización Mundial de la Salud, la adopción de tecnologías de asistencia al conductor podría prevenir hasta el 30 % de los accidentes relacionados con la fatiga. Este proyecto de tesis presentado no solo se alinea con estas tendencias globales, sino que también destaca el potencial de la visión artificial y los asistentes virtuales como herramientas clave para abordar problemas críticos en la seguridad vial.

En este documento se detallan los fundamentos teóricos y técnicos que sustentan el desarrollo del prototipo, así como los resultados obtenidos durante las pruebas experimentales realizadas. Además, se discuten las implicaciones éticas, sociales y económicas de implementar esta tecnología, resaltando su impacto en la reducción de accidentes y la promoción de una conducción más segura y responsable.

II. PROBLEMA

Los accidentes automovilísticos son responsables de 1,3 millones de muertes anuales. La inatención, que incluye la distracción y la somnolencia del conductor, es responsable de una gran cantidad de accidentes, especialmente los más fatales. Aunque diversas investigaciones intentan estimar el impacto de la inatención en los accidentes, no existe una cifra exacta debido a las diferencias en los estudios realizados en distintos lugares y períodos. En términos generales, se calcula que la inatención causa entre el 25 % y el 75 % de los accidentes y casi-accidentes [1].

Muchos de estos conductores son afectados por la fatiga laboral (FL), una condición compleja que surge de una variedad de factores que influyen en su origen, generando importantes costos en el sistema de salud [2]. Los síntomas de la FL incluyen somnolencia, cansancio general y un procesamiento deficiente de la información, lo cual provoca errores al momento de conducir, disminución en el rendimiento y una mayor inclinación a tomar riesgos. Los accidentes de tránsito representan un grave problema social y son la segunda causa de muerte laboral, a pesar de los esfuerzos por controlarlos [3].

La somnolencia afecta las funciones cognitivas, reduciendo la eficiencia del procesamiento cortical y la capacidad de reacción. Esto ha motivado el uso de tecnologías como el análisis de señales electroencefalográficas (EEG) para detectar la fatiga, aunque su implementación es costosa e incómoda [4].

Aunque los conductores son conscientes del riesgo de somnolencia debido a la fatiga, muchos continúan conduciendo a pesar de sentir sueño. En un estudio realizado con conductores de camiones en Australia, el 26 % reportó seguir conduciendo con somnolencia en más de la mitad de sus viajes [5]. Los autores identifican «predictores psicosociales del comportamiento de conducción con sueño», como la presión por cumplir tiempos, el temor a reportar el cansancio y la falta de control sobre los horarios de conducción. Además, subrayan que algunos conductores no reconocen su somnolencia, lo que puede llevarlos a seguir conduciendo sin ser conscientes del peligro [6].

En Ecuador, la somnolencia y el cansancio de los conductores son factores clave en los accidentes de tránsito, una de las principales causas de muertes no naturales. Las jornadas prolongadas en sectores como el transporte de carga incrementan este riesgo, y los métodos actuales, como señales y sanciones, no abordan directamente el problema.

La somnolencia representa una alteración de funciones neurocognitivas que impacta directamente la atención y el tiempo de reacción de las personas. Estos efectos se agravan, pues incluso unos pocos segundos de desconexión pueden desencadenar accidentes graves [7].

III. JUSTIFICACIÓN

La somnolencia constituye un factor de riesgo considerable, especialmente en tareas que exigen alta atención y responsabilidad, como la de los conductores de vehículos. Los trabajadores que experimentan fatiga o somnolencia durante sus turnos pueden ver reducida su capacidad cognitiva, lo cual incrementa el riesgo de accidentes laborales, daños materiales y, en situaciones extremas, lesiones graves o fatales [8]. La detección de somnolencia para los conductores de vehículos resulta crucial para reducir los riesgos de accidentes [9]. Esta problemática supone riesgos significativos para la seguridad y la salud de los operadores, impactando directamente en la productividad y en su bienestar [10].

En este contexto, la incorporación de dispositivos mecatrónicos diseñados específicamente para detectar somnolencia se presenta como una solución novedosa y eficaz para reducir dichos riesgos. Estos dispositivos podrían supervisar continuamente la atención de los conductores, generando alertas tanto para ellos como para sus supervisores al identificar señales de fatiga [11]. La implementación de un sistema de visión artificial permite la monitorización en tiempo real del estado de los operadores mediante algoritmos de procesamiento de imágenes y reconocimiento facial[12]. Este sistema es capaz de detectar signos de somnolencia en los conductores, como el cierre prolongado de párpados, parpadeo lento y el cabeceo.

Se ha evidenciado que el PERCLOS, que significa Percentage of the Time Eyelids are Closed, es el enfoque más eficaz para identificar la somnolencia mediante el análisis ocular. La visión artificial se implementa a través de cámaras estratégicamente ubicadas en el vehículo o en el entorno del conductor, conectadas a un software de inteligencia artificial que analiza las imágenes captadas [13]. Mediante el uso de técnicas avanzadas de aprendizaje profundo y redes neuronales, el sistema de visión artificial aprende a reconocer patrones de fatiga en el rostro del conductor, adaptándose a diferentes condiciones de luz y ángulos de visión para optimizar su precisión [14].

La adopción de esta tecnología en una empresa no solo reflejaría el compromiso de la organización con la seguridad y el bienestar de los conductores, sino que también contribuye a mejorar la eficiencia y calidad de las operaciones. Además de reducir los riesgos de accidentes y lesiones, se potencia la productividad del conductor.

IV. OBJETIVOS

IV-A. Objetivo general

Implementar un sistema de detección de somnolencia para conductores de vehículos mediante visión artificial.

IV-B. Objetivos específicos

- Identificar los parámetros fisiológicos y de comportamiento que indiquen estados de somnolencia en conductores de vehículos.
- Diseñar un algoritmo de visión artificial que permita la detección de somnolencia.
- Validar el funcionamiento del sistema alertando al conductor mediante un indicador.

V. FUNDAMENTOS TEÓRICOS

V-A. Sueño y Somnolencia

V-A.1. Fases del Sueño

El sueño es un proceso biológico estructurado en ciclos que se repiten cada 90 minutos aproximadamente. Se divide en dos grandes categorías: el sueño de movimientos oculares no rápidos (No MOR) y el sueño de movimientos oculares rápidos (MOR). Durante la noche, una persona atraviesa varias fases dentro de estos dos tipos de sueño, cada una con características específicas [15].

Fase del sueño	**Clasificación**	**Características principales**	**Duración estimada**
Fase inicial	No MOR (N1)	Transición de la vigilia al sueño, reducción de la actividad muscular, latidos cardíacos más lentos	1-7 min
Sueño ligero	No MOR (N2)	Disminución de la temperatura corporal, menor respuesta a estímulos externos, aparición de complejos K en la actividad cerebral	10-25 min
Sueño profundo	No MOR (N3)	Ondas cerebrales lentas (delta), difícil despertar, etapa crucial para la recuperación física	20-40 min
Sueño paradójico	MOR	Aumento de la actividad cerebral, consolidación de la memoria, aparición de los sueños	10-60 min

Tabla 1. Fases del sueño y sus características principales.

- Etapa 1: También conocida como *N1*, es la fase inicial del sueño, cuando una persona comienza a quedarse dormida. Esta etapa suele durar entre uno y siete minutos.

  Durante la fase N1, el cuerpo aún no está completamente relajado, aunque las actividades corporales y cerebrales empiezan a disminuir con breves períodos de movimientos. En esta fase se producen ligeros cambios en la actividad cerebral relacionados con el proceso de quedarse dormido.

  Es fácil despertar a alguien en esta etapa, pero si no se le interrumpe, puede avanzar rápidamente hacia la *etapa 2*. A medida que la noche avanza, una persona que duerme sin interrupciones pasa cada vez menos tiempo en la etapa 1, conforme progresa por los ciclos del sueño.

- Etapa 2: Durante la *etapa 2* (o N2), el cuerpo entra en un estado más tranquilo, caracterizado por una disminución de la temperatura, relajación muscular y una respiración y ritmo cardíaco más lentos. Al mismo tiempo, las ondas cerebrales adoptan un patrón distinto y cesan los movimientos oculares. En general, la actividad cerebral se reduce, aunque hay estallidos breves de actividad que ayudan a resistir los estímulos externos.

  La etapa 2 puede durar entre 10 y 25 minutos en el primer ciclo de sueño, y cada fase N2 tiende a prolongarse durante la noche. En conjunto, una persona pasa aproximadamente la mitad de su tiempo durmiendo en esta etapa.

- Etapa 3: La *etapa 3* del sueño, también conocida como N3 o sueño profundo, es la fase en la que es más difícil despertar a una persona. Durante esta etapa, el tono muscular, el pulso y la frecuencia respiratoria disminuyen aún más a medida que el cuerpo se relaja por completo.

La actividad cerebral durante esta fase tiene un patrón distintivo de ondas llamadas *delta*. Por este motivo, la etapa 3 también se conoce como sueño delta o sueño de ondas lentas (SWS).

Los expertos consideran que esta etapa es crucial para un sueño reparador, ya que favorece la recuperación y el crecimiento corporal. Además, se cree que refuerza el sistema inmunológico y otros procesos clave del organismo. Aunque la actividad cerebral se reduce, existe evidencia de que el sueño profundo contribuye al pensamiento claro, la creatividad y la memoria.

Durante la primera mitad de la noche, la mayor parte del tiempo se pasa en sueño profundo. En los primeros ciclos de sueño, la etapa N3 suele durar entre 20 y 40 minutos. A medida que la noche avanza, estas fases se acortan y se pasa más tiempo en la *etapa MOR*.

- Etapa 4 (MOR): Durante la *etapa 4*, conocida como sueño *MOR* (Movimiento Ocular Rápido), la actividad cerebral se acelera hasta acercarse a los niveles observados cuando estamos despiertos. Mientras tanto, el cuerpo experimenta una atonía, que es una parálisis temporal de los músculos, con dos excepciones: los músculos oculares y los que controlan la respiración. Aunque los ojos permanecen cerrados, se pueden observar rápidos movimientos oculares, lo que da nombre a esta fase.

  Se considera que el sueño MOR es fundamental para procesos cognitivos como la memoria, el aprendizaje y la creatividad. Es conocido por ser la etapa en la que ocurren los sueños más vívidos, debido a la intensa actividad cerebral. Aunque los sueños pueden ocurrir en cualquier fase del sueño, son menos frecuentes e intensos en las fases NREM.

  En circunstancias normales, la fase MOR no se alcanza hasta después de aproximadamente 90 minutos de haber comenzado a dormir. A medida que avanza la noche, los períodos de MOR se alargan, especialmente en la segunda mitad de la noche. Mientras que la primera fase MOR puede durar solo unos minutos, las fases posteriores pueden extenderse hasta una hora. En total, las fases MOR representan aproximadamente el 25 % del sueño en los adultos.

El descanso nocturno es un proceso regulado por la alternancia entre las fases de sueño NO MOR y MOR, las cuales se suceden entre cinco y seis veces por noche, acumulando un total de aproximadamente 7.5 a 9 horas de sueño. Durante las primeras horas de descanso, predomina el sueño profundo caracterizado por ondas lentas, mientras que en la parte final de la noche, la actividad del sueño MOR se intensifica progresivamente.
La sincronización de los ciclos de sueño depende de los ritmos circadianos, los cuales están bajo el control del núcleo supraquiasmático, el hipotálamo y la glándula pineal. Estos ritmos biológicos están modulados por distintos factores, tanto internos como externos, incluyendo la exposición a la luz natural y la predisposición genética.
Además de su función en el descanso y la recuperación neuronal, el sueño desempeña un papel clave en la regulación del sistema inmunológico. Durante las horas de sueño, se observa un aumento en la actividad de citocinas proinflamatorias como IL-1, IL-6 y TNF, las cuales favorecen la fase NO MOR. En contraste, moléculas antiinflamatorias como IL-4 e IL-10 actúan reduciendo esta fase. Este equilibrio es fundamental para la homeostasis inmunológica, ya que la privación del sueño puede generar desajustes en la respuesta inmune, promoviendo un incremento en la inflamación sistémica y afectando la capacidad del organismo para combatir infecciones o enfermedades metabólicas crónicas [16].
En la figura 1 se obsevra la destrucción de las neuronas noradrenérgicas, en el LC genera una alteración en el patrón normal de sueño y vigilia. Los datos muestran el porcentaje de tiempo que las ratas permanecieron en cada fase de alerta a lo largo de las 24 horas. Las barras horizontales blancas corresponden al día, mientras que las negras indican la noche. Las ratas no lesionadas experimentan un sueño no-MOR (NREMS) considerablemente mayor durante las primeras horas de la mañana, en contraste con las ratas lesionadas, que duermen mucho menos. Por otro lado, las ratas control muestran una alta vigilia al inicio de la noche, mientras que las ratas lesionadas permanecen significativamente menos despiertas.

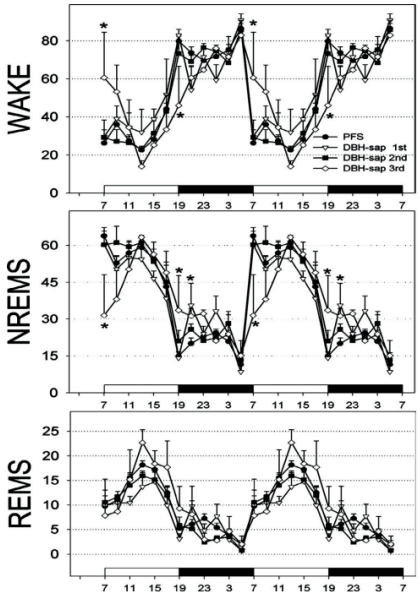

Figura 1. Alteraciones en las fases de alerta tras la lesión de las neuronas noradrenérgicas en el locus coeruleus. Figura extraída del libro de Neurociencia del sueño de A. D. Rodríguez y L. M. Fernández

El sueño es una fase cíclica y transitoria en la que la conciencia se reduce significativamente, pero sin llegar a una desconexión total. Durante este estado, la actividad cerebral sigue operando de manera dinámica, permitiendo que ciertos estímulos sean percibidos y facilitando la recuperación de imágenes oníricas al despertar. A diferencia de la anestesia o el coma, el sueño no es un estado patológico ni inducido artificialmente, sino un proceso biológico esencial. Su relevancia no se limita solo al descanso, sino que desempeña un papel clave en la regulación de funciones fisiológicas, el procesamiento que tiene de información y la estabilidad homeostática del organismo.

V-A.2. Factores que Propician la Somnolencia en Conductores

El cansancio al conducir puede deberse a múltiples factores, entre los cuales destacan la privación de sueño, las condiciones laborales y ciertas características individuales, como la edad y la experiencia al volante. Cuando el descanso nocturno es insuficiente, especialmente si se reduce a menos de cuatro horas, la capacidad de concentración se ve severamente afectada, aumentando hasta cinco veces el riesgo de un accidente. Este deterioro no solo ocurre de inmediato, sino que se agrava con el paso de los días si la falta de sueño persiste, situación habitual en quienes trabajan en turnos nocturnos o exceden las 60 horas semanales, como suele ocurrir en el sector del transporte comercial.
Además de la cantidad de sueño, su calidad también desempeña un papel crucial. Trastornos como la apnea obstructiva del sueño y un descanso fragmentado pueden generar fatiga diurna, elevando la probabilidad de incidentes viales. No obstante, el riesgo de somnolencia no es constante durante el día, sino que sigue un patrón biológico. En particular, las horas comprendidas entre las 2 y las 6 de la madrugada, así como entre las 2 y las 4 de la tarde, coinciden con momentos de mayor vulnerabilidad debido a la regulación circadiana del organismo. Cuando las demandas laborales interfieren con estos ritmos naturales, el déficit de descanso se intensifica.
El tipo de trayecto también influye en la aparición de la fatiga. La conducción en carreteras de larga distancia, autopistas con escasa variabilidad y entornos monótonos reduce la estimulación sensorial, lo que favorece la disminución del estado de alerta. Se ha estimado que cerca del 40
A su vez, factores individuales como la edad y las diferencias fisiológicas pueden afectar la susceptibilidad a la fatiga. Las personas de mayor edad, por ejemplo, pueden experimentar un declive en su capacidad de mantener la atención sostenida. Por otro lado, la somnolencia no solo se manifiesta a través de la sensación de sueño, sino también mediante cambios sutiles en el comportamiento del conductor, como una reducción en los movimientos de la cabeza, un aumento en la frecuencia del parpadeo, variaciones en la presión sobre los pedales y el volante, y alteraciones en la estabilidad del vehículo, como desviaciones involuntarias de carril.

V-A.3. Consecuencias de la Fatiga en la Conducción

El impacto de la fatiga y la somnolencia en la conducción es significativo, comprometiendo funciones esenciales como el tiempo de reacción, la capacidad de concentración y la toma de decisiones. Estudios han demostrado que la reducción del descanso nocturno, las jornadas laborales excesivas y la conducción prolongada en entornos monótonos contribuyen a la degradación progresiva del rendimiento al volante. Uno de los aspectos más peligrosos de la fatiga es que, en muchas ocasiones, los conductores no perciben la magnitud de su deterioro cognitivo hasta que su capacidad de respuesta ya se encuentra comprometida. Como consecuencia, el tiempo necesario para reaccionar ante imprevistos se incrementa considerablemente, mientras que la propensión a cometer errores y adoptar conductas de riesgo también aumenta. Esto se traduce en una mayor probabilidad de accidentes [17].
Desde una perspectiva neurofisiológica, la fatiga puede analizarse mediante la interacción entre la actividad cortical y subcortical del cerebro, lo que permite evaluar cómo la disminución progresiva de la activación neuronal afecta el estado de alerta y el desempeño cognitivo. La figura 2 representa el modelo teórico de fatiga en estos niveles, describiendo los mecanismos fisiológicos que intervienen en la reducción del rendimiento durante la conducción.

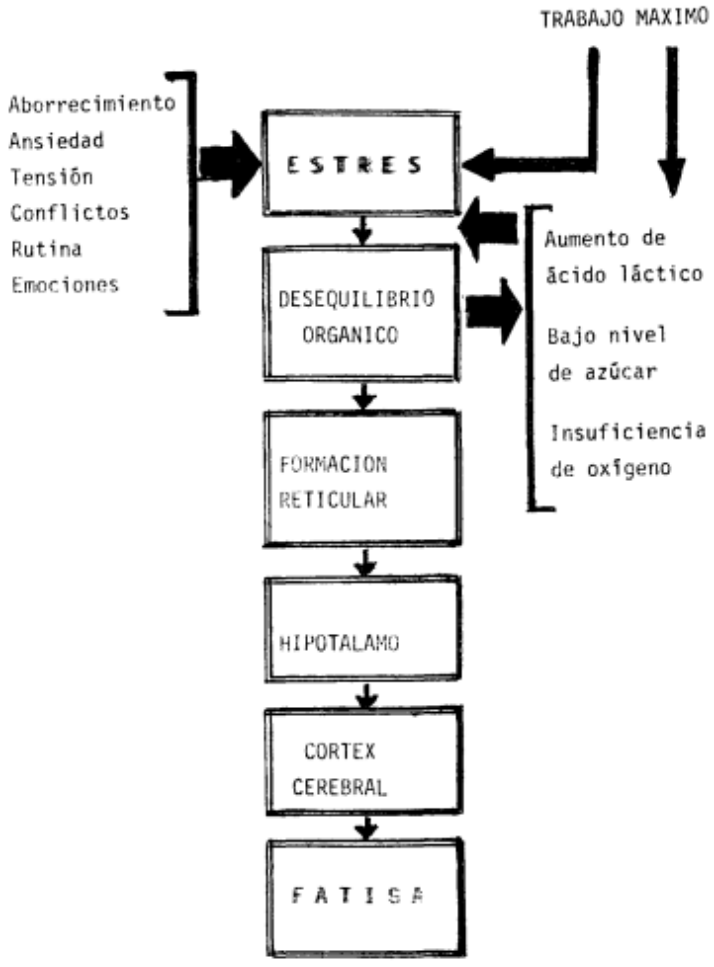

Figura 2. Modelo Teórico de fatiga cortical y subcortical. Figura extraída de la revista “Avances en enfermería” de L. M. Florez

Este fenómeno representa un peligro similar al de manejar en estado de ebriedad. De acuerdo con la Administración Nacional de Seguridad del Tráfico en las Carreteras (NHTSA), sus efectos en la capacidad de conducción pueden ser igualmente perjudiciales, la somnolencia no solo implica quedarse dormido durante la conducción, sino que produce un deterioro cognitivo y motor que afecta al conductor de manera similar al alcohol. Este fenómeno reduce los tiempos de reacción, altera la atención y disminuye

el procesamiento mental, el juicio y la toma de decisiones, lo que aumenta significativamente el riesgo de accidentes.
El monitoreo de la fatiga en conductores ha impulsado el desarrollo de múltiples métodos que permiten evaluar tanto su estado fisiológico como sus patrones de comportamiento. Entre las técnicas más empleadas, el análisis de señales cerebrales mediante electroencefalografía (EEG) y la medición de la actividad cardíaca a través del electrocardiograma (ECG) han demostrado ser herramientas precisas para detectar alteraciones en la atención y en el nivel de alerta. Sin embargo, su aplicación en entornos reales resulta compleja, ya que estos sistemas, al requerir contacto directo con el conductor, pueden interferir con su comodidad y movilidad, lo que limita su viabilidad fuera de condiciones controladas.

V-A.4. Parámetros de Somnolencia

Parámetros Fisiológicos: Los parámetros fisiológicos están relacionados con las funciones internas del cuerpo y se obtienen mediante mediciones biométricas. Estos incluyen:

1. **Actividad ocular:**
   - **PERCLOS:** Porcentaje de tiempo que los ojos permanecen cerrados en un intervalo dado.
   - **Velocidad del parpadeo:** Parpadeos más lentos o frecuentes.
   - **Tamaño de la pupila:** Cambios en el diámetro pupilar debido a la fatiga.
   - **Movimiento ocular:** Reducción en movimientos sacádicos o movimientos erráticos.
2. **Actividad cerebral:**
   - **Electroencefalografía (EEG):** Incremento de ondas theta (4-8 Hz) y delta (0.5-4 Hz).
   - **Actividad alfa:** Disminución de ondas alfa (8-12 Hz), asociadas con estados de alerta, tal como se muestra en la figura.

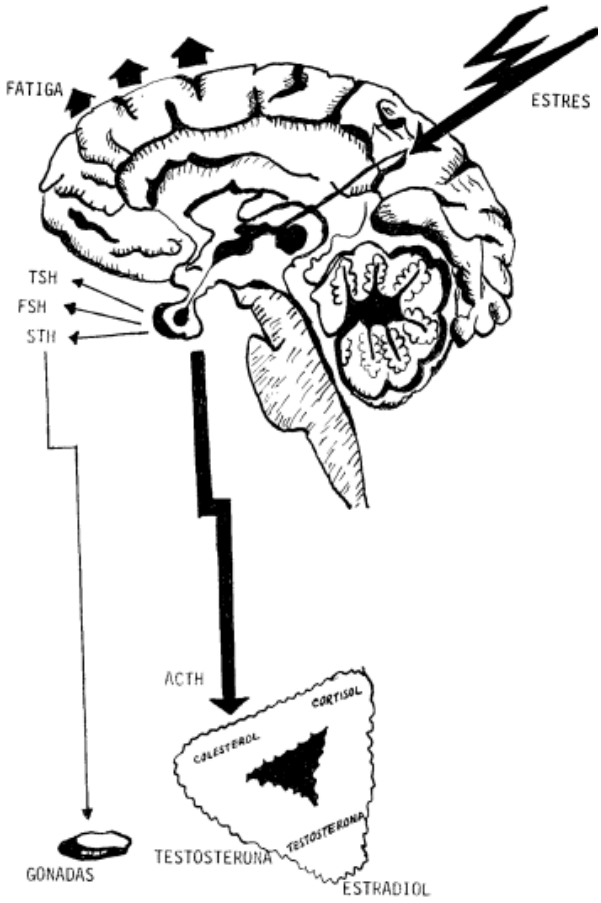

Figura 3. Acción Neuroendocrina, en el estrés y la fatiga. Figura extraída de la revista "Avances en enfermería" de L. M. Florez

3. Frecuencia cardíaca:
    - Variabilidad de la frecuencia cardíaca (HRV): Disminución de la variabilidad, como en la Figura 4.
    - Ritmo cardíaco reducido: Asociado con relajación o fatiga.

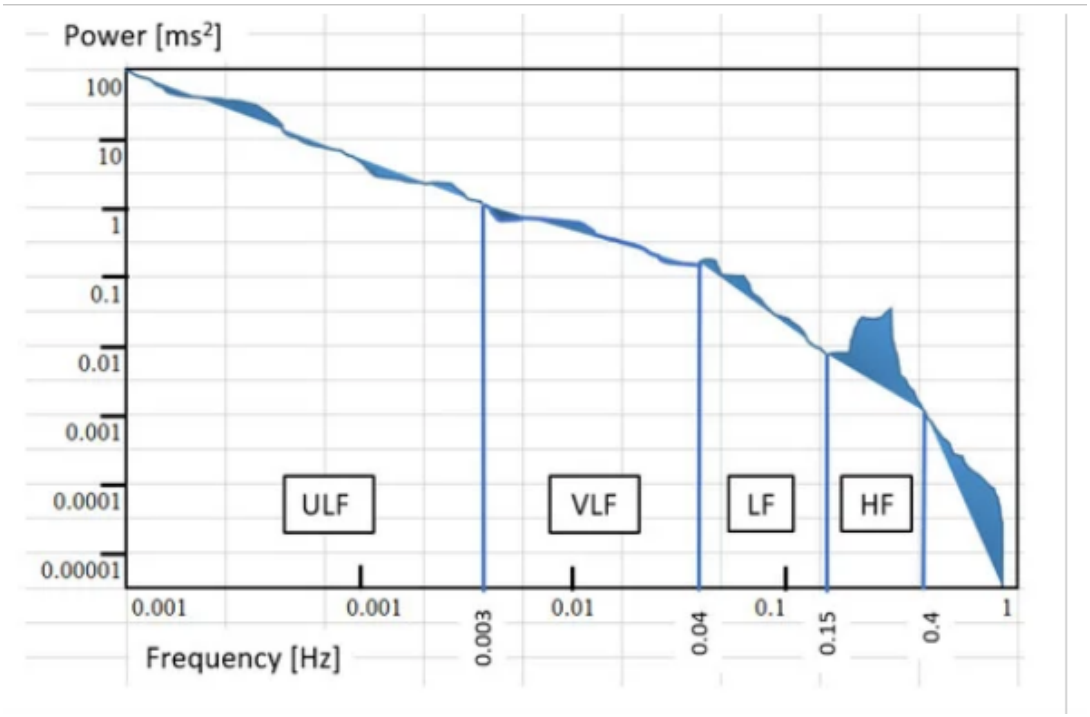

Figura 4. Espectro de densidad de potencia en función de la frecuencia (Hz). Figura extraída del libro Análisis de Señales en Neurociencia de R. C. H. Chang, C. Y. Wang, W. T. Chen y C. D. Chiu.

4. **Respiración:**
   - Ritmo más lento y superficial.
   - Pausas respiratorias (apneas).
5. **Tono muscular:**
   - Relajación de los músculos faciales, como párpados caídos o mandíbula relajada.
   - Inclinación de la cabeza debido a pérdida de control postural.

Parámetros Comportamentales: Los parámetros comportamentales son observables externamente y reflejan cambios en la conducta debido a la fatiga:

1. **Movimiento ocular:**
   - Parpadeos frecuentes y prolongados.
   - Microsueños: Episodios breves de cierre involuntario de los ojos.
2. **Expresiones faciales:**
   - Bostezos frecuentes.
   - Pérdida de expresividad facial.
3. **Movimientos de la cabeza y postura:**
   - Cabeceo: Movimientos involuntarios hacia adelante o hacia los lados.
   - Postura corporal relajada, indicando pérdida de alerta.
4. **Comportamientos repetitivos:**

- Frotarse los ojos.
- Cambios frecuentes de posición.

5. **Reducción de la atención:**
    - Respuesta más lenta a estímulos externos.
    - Incremento en errores durante tareas visuales o manuales.

[18].

V-B. Introducción a la Visión Artificial

V-B.1. Definición

El procesamiento automatizado de imágenes ha revolucionado la forma en que los sistemas tecnológicos perciben e interpretan su entorno. Mediante algoritmos avanzados, la visión artificial permite la captura y análisis de datos visuales con el propósito de reconstruir estructuras tridimensionales a partir de información bidimensional. Su desarrollo ha trascendido las limitaciones humanas en múltiples tareas repetitivas, logrando una precisión y eficiencia superiores en diversas aplicaciones, desde el reconocimiento facial y la seguridad hasta la navegación de vehículos autónomos y la integración en sistemas robóticos [19]. Su origen se remonta a los años 60, cuando se realizaron experimentos para conectar cámaras de video a computadoras, y ha avanzado significativamente en la última década, especialmente en el campo del reconocimiento de objetos y facial.
Los componentes clave de un sistema de visión artificial incluyen iluminación, que facilita la captura adecuada de imágenes; cámaras con lentes y sensores para generar imágenes; y sistemas de procesamiento que analizan las imágenes capturadas, los cuales comprenden la tarjeta de adquisición, algoritmos de procesamiento e interfaces para mostrar los resultados. Además, los actuadores externos, como robots o monitores, permiten mostrar la información procesada [20].
El proceso de visión artificial sigue un esquema básico que incluye la adquisición y digitalización de imágenes, preprocesamiento para eliminar datos irrelevantes, segmentación para extraer objetos, extracción de características para identificación y, finalmente, la clasificación de objetos en categorías [21].

V-B.2. Imagen Digital

Una imagen digital es una representación de una escena que se forma mediante una matriz de puntos, llamados píxeles. A diferencia de las imágenes reales, que son continuas, las imágenes digitales son discretas y tienen un número finito de píxeles [22]. Cada píxel tiene un valor que representa la intensidad lumínica, siendo 0 para el color negro y el valor máximo para el blanco, dependiendo de los niveles de gris que se utilicen. A mayor número de píxeles, mejor será la resolución espacial, mientras que una mayor cantidad de niveles de gris mejora la resolución tonal.

Figura 5. Consecuencias de la variabilidad en el número de píxeles de una imagen.

Las imágenes pueden ser monocromáticas (un solo tono, como el gris) o multiespectrales (con varias bandas de color). Las imágenes en color típicamente usan tres componentes: rojo, verde y azul (RGB). En cuanto al procesamiento de imágenes, se utilizan conceptos como la vecindad, que describe cómo los píxeles están relacionados con los que los rodean. Tal como se muestra en la figura 6, existen vecindades

de orden 4 (con los píxeles horizontales y verticales cercanos) y de orden 8 (que incluye los píxeles diagonales).

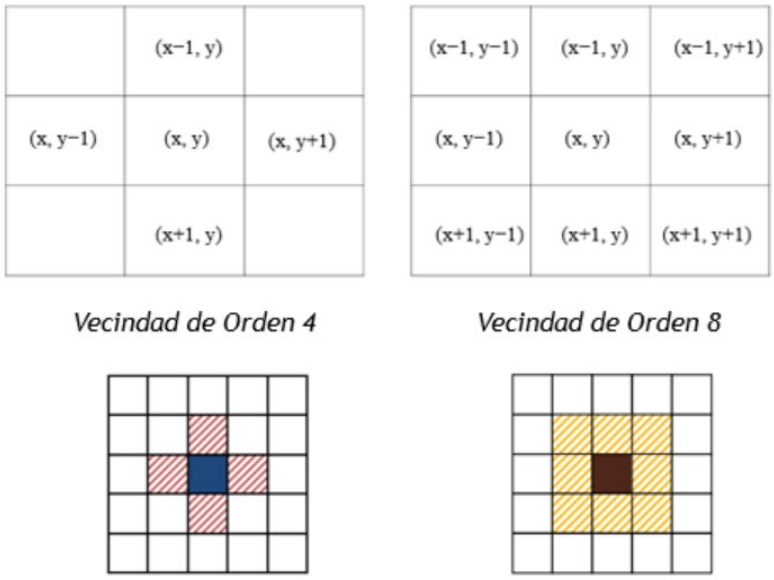

Figura 6. Vecindad en una imagen. Figura extraída del libro Introducción a la visión artificial: procesos y aplicaciones de B. S. Borrella

También se usan máscaras digitales o filtros, que son matrices pequeñas aplicadas a la imagen para modificarla mediante un proceso llamado convolución digital. Este proceso aplica una operación aritmética entre la máscara y los píxeles en el vecindario para modificar el píxel central [23].

V-B.3. Fase de Segmentación en el Sistema de Visión Artificial

La segmentación es un proceso clave en la visión artificial que consiste en dividir una imagen en áreas más pequeñas, conocidas como segmentos. En esta fase, se clasifica cada píxel de la imagen, asignándole un valor que representa una categoría específica. Este proceso permite identificar objetos y detectar sus bordes, lo cual es fundamental para tareas como contar elementos en la imagen.
Una de las técnicas más sencillas para comenzar es la binarización, que asigna un valor de 0 (negro) o 255 (blanco) a los píxeles según su intensidad, comparada con un umbral. Esto funciona bien para imágenes con fondo blanco y objetos contrastantes. Sin embargo, para imágenes más complejas con múltiples colores o formas, se deben emplear métodos para detectar bordes, como el gradiente de la imagen utilizando derivadas direccionales, lo cual permite identificar cambios bruscos en la intensidad de los píxeles.
El gradiente se calcula como un vector bidimensional que indica la dirección de mayor cambio en la imagen. Con esto, se pueden aplicar técnicas de umbralización para crear una imagen binaria de bordes. Además, se pueden utilizar filtros de detección de bordes, como Sobel, Prewitt y Roberts, que calculan el gradiente en cada píxel y permiten identificar los bordes de la imagen mediante convolución.

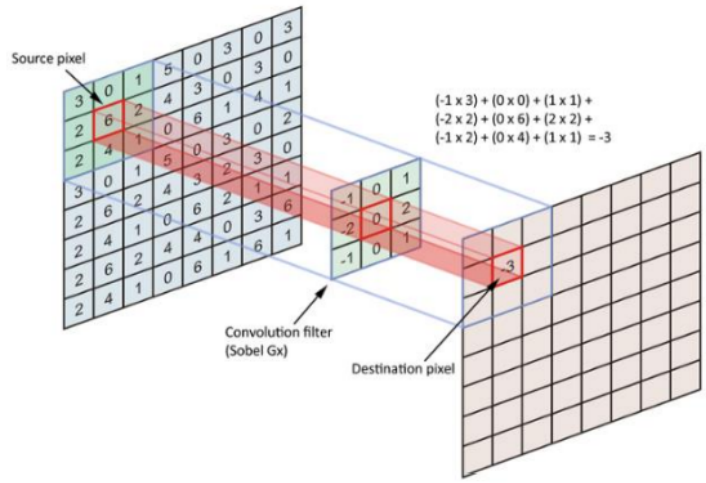

Figura 7. Convolución con el filtro de Sobel. Figura extraída del libro Introducción a la visión artificial: procesos y aplicaciones de B. S. Borrella

V-B.4. Fase de Interpretación del Sistema de Visión Artificial

Una vez segmentada la imagen, la siguiente etapa es la interpretación, que tiene como objetivo identificar y comprender los patrones en la imagen. Este procedimiento involucra una secuencia estructurada de análisis visual que se desglosa en cuatro subprocesos interdependientes, cada uno desempeñando un papel esencial en la extracción, interpretación y clasificación de información a partir de una imagen:

1. **Localización, identificación y categorización:** Inicialmente, el sistema ejecuta un proceso de exploración visual cuyo objetivo primario es la detección de elementos en la imagen sin requerir conocimiento previo de su naturaleza. Posteriormente, mediante algoritmos de reconocimiento, se analizan propiedades clave como dimensiones, estructura geométrica y textura, permitiendo la correlación con patrones almacenados en bases de datos de referencia. Finalmente, en la etapa de identificación, el sistema asigna una etiqueta específica a cada objeto previamente reconocido, estableciendo su clasificación en función del contexto en el que se encuentra.

2. **Segmentación estructural y modelado:** Una vez definidos los objetos de interés, se procede a descomponer la imagen en regiones diferenciadas a través de técnicas de segmentación avanzadas, tales como trazado de bordes, análisis de gradientes y filtrado espacial. Este procedimiento permite establecer contornos bien definidos y resaltar características estructurales esenciales de cada segmento, facilitando una posterior evaluación precisa de atributos geométricos y texturales.

3. **Clasificación adaptativa:** En esta etapa, cada segmento delimitado es sometido a un proceso de comparación con un conjunto de modelos predefinidos. Se implementan técnicas de aprendizaje automático y redes neuronales para identificar similitudes y diferencias con patrones previamente almacenados. Como resultado, se asigna una categoría específica a cada unidad analizada, organizando los elementos en estructuras jerárquicas que permiten su interpretación dentro del conjunto global de la imagen.

4. **Inferencia y contextualización:** Finalmente, los datos obtenidos en las fases anteriores se integran con información adicional procedente de fuentes externas o conocimientos previos del sistema. A través de razonamiento lógico, fusión de datos y procesamiento contextualizado, el sistema genera una interpretación estructurada de la imagen, formulando conclusiones sobre la escena representada y optimizando la toma de decisiones automatizada.

V-C. Análisis y Procesamiento de Información Visual

V-C.1. Fundamentos en el Procesamiento de Señales Digitales

El procesamiento digital de señales (DSP) es una rama crucial de la ingeniería moderna que se encarga de la conversión y manipulación de señales del mundo físico, como sonido, imágenes o fluctuaciones de

voltaje en sistemas eléctricos. Al digitalizar estas señales, se pueden aplicar técnicas computacionales sofisticadas para mejorar su calidad, extraer datos significativos y facilitar su análisis en una variedad de campos, desde la medicina hasta la ingeniería y las telecomunicaciones.
El origen del DSP se encuentra en la segunda mitad del siglo XX, cuando la tecnología computacional estaba en sus etapas iniciales. Aunque al principio solo se usaba en aplicaciones especializadas, como la defensa o la exploración espacial, los avances en la miniaturización de circuitos y el aumento de la capacidad de procesamiento en los microprocesadores hicieron posible su adopción masiva a partir de los años 80. Desde entonces, se ha convertido en un componente esencial en dispositivos electrónicos como teléfonos móviles, sistemas de audio y cámaras digitales.
Un sistema DSP clásico sigue una secuencia de etapas: la señal se captura en formato analógico, luego se convierte en digital por un conversor analógico-digital (ADC), se procesa mediante algoritmos de procesamiento de señales, y finalmente, si es necesario, se reconvierte en analógica a través de un conversor digital-analógico (DAC). Este flujo de trabajo permite realizar funciones como filtrado de ruido, análisis espectral y detección de patrones, optimizando aplicaciones de telecomunicaciones, visión computacional y reconocimiento de voz.
El avance en la arquitectura de procesadores DSP ha sido notable, con mejoras que incluyen paralelismo de tareas, memoria optimizada y unidades de cálculo especializadas para acelerar procesos matemáticos complejos. Gracias a estos desarrollos, el DSP juega un papel fundamental en el avance de áreas como la inteligencia artificial y la visión por computadora [13].

V-C.2. Integración de la Inteligencia Artificial en la Visión por Computadora

La integración de la inteligencia artificial (IA) con la visión por computadora ha impulsado el desarrollo de sistemas autónomos capaces de analizar imágenes y videos con un alto grado de precisión. Utilizando algoritmos de aprendizaje profundo, estos sistemas pueden identificar, clasificar y rastrear objetos en tiempo real, adaptándose a cambios en la iluminación, el ángulo de visión y la complejidad de los entornos.
En este ámbito, las redes neuronales convolucionales (CNN) se destacan por su eficacia en la segmentación de imágenes y la extracción de características. Técnicas clásicas como SIFT (Scale-Invariant Feature Transform) y SURF (Speeded-Up Robust Features) facilitan la localización de puntos clave en imágenes, lo que es crucial para el seguimiento de objetos y la reconstrucción tridimensional. Para tareas que requieren una mayor eficiencia computacional, el algoritmo ORB (Oriented FAST and Rotated BRIEF) se ha vuelto popular debido a su menor demanda de recursos y su efectividad en entornos con limitaciones de hardware.
La IA también ha permitido la automatización en el seguimiento de objetos dentro de secuencias de video, lo que mejora la precisión y reduce la intervención humana en sistemas de monitoreo. Técnicas como los filtros de Kalman y los filtros de partículas se utilizan para predecir el movimiento de los objetos, proporcionando estimaciones de trayectorias con alta exactitud. Además, los métodos de agrupamiento como k-means se aplican en la segmentación de imágenes, ayudando a identificar regiones de interés en escenas complejas.
La combinación de redes neuronales con transformaciones matemáticas avanzadas, como la transformada wavelet, ha abierto nuevas posibilidades en áreas como la detección de anomalías en procesos industriales, el análisis de imágenes médicas y la identificación de patrones en señales de tráfico [3].

V-C.3. Reconocimiento Facial en el Contexto de la Seguridad

El reconocimiento facial ha avanzado rápidamente gracias a la mejora de los algoritmos de visión artificial y la expansión de bases de datos biométricas. Estos sistemas se basan en principios matemáticos y estadísticos que permiten la identificación y autenticación de individuos con gran precisión. Un área clave de aplicación es la detección de somnolencia en conductores, donde el análisis en tiempo real de las expresiones faciales y los microgestos puede prevenir accidentes [24].

1. **Detección de Rostros con el Modelo Viola-Jones**

   Introducido en 2001 por Paul Viola y Michael Jones, este modelo optimizó la detección de rostros mediante clasificadores en cascada. Utiliza características Haar para identificar patrones de intensidad en regiones de una imagen, lo que lo hace altamente eficiente en aplicaciones en tiempo real. La optimización de su procesamiento mediante ventanas de búsqueda y la integral de imagen han permitido su implementación en una variedad de dispositivos.

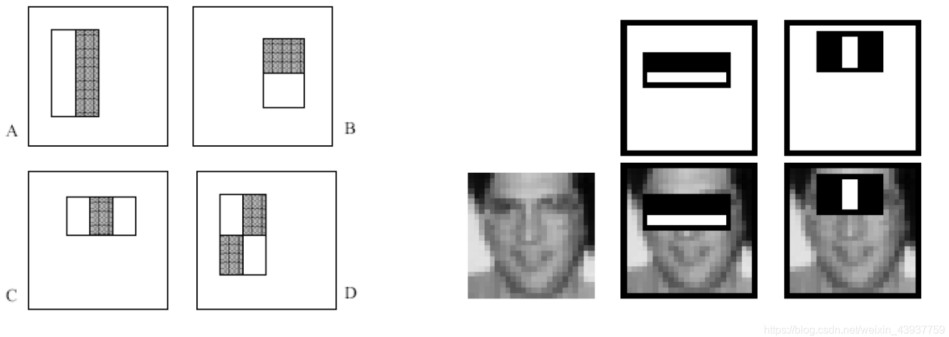

Figura 8. Esquema básico del algoritmo Viola-Jones para la detección de rostros.

2. **Clasificación por Detección en Cascada**

   La detección en cascada, utilizada en el modelo de Viola-Jones, mejora la eficiencia computacional al eliminar rápidamente las regiones de la imagen que no contienen rostros. A lo largo del proceso, se aplican filtros progresivos que refinarán la detección, minimizando los falsos positivos y aumentando la precisión. Este enfoque se emplea en cámaras de seguridad y sistemas de autenticación biométrica [25].

Figura 9. Proceso de detección en cascada para el reconocimiento facial.

3. **Autenticación mediante Análisis de Características Faciales**

   Hoy en día, los sistemas de reconocimiento facial no solo se limitan a detectar rostros, sino que también utilizan redes neuronales profundas para clasificar las características faciales y realizar autenticación. Modelos como FaceNet y DeepFace han alcanzado niveles de precisión sorprendentes, permitiendo la identificación de rostros con un margen de error de menos del 0.1 %, incluso en condiciones de poca luz o con cambios en la expresión facial. Estos avances son fundamentales en sistemas de seguridad como los que se encuentran en aeropuertos y dispositivos móviles.

V-C.4. Detección Activa de Movimiento Mediante Visión Artificial

El análisis del movimiento humano se fundamenta en principios biomecánicos para identificar patrones asociados al desplazamiento. Para recopilar información, se emplean modelos biomecánicos que permiten evaluaciones cualitativas sobre la fatiga y el esfuerzo. Estos modelos se comparan con patrones definidos como normales, obtenidos mediante fotogrametría y técnicas de seguimiento del movimiento [26].
A continuación, se presentan las coordenadas utilizadas en el análisis:

- **Coordenada X:** Posición horizontal.
- **Coordenada Y:** Posición vertical.
- **Coordenada Z:** Profundidad o posición en el eje perpendicular.

V-C.5. Algoritmos de Detección de Objetos

1. Basados en Características Clásicas

1.1 Clasificador en Cascada Haar (Viola-Jones): Es un método basado en características geométricas para la detección de objetos. Fue desarrollado por Paul Viola y Michael Jones en 2001 y es ampliamente utilizado en la detección de rostros en imágenes y vídeos.

- Características de Haar: Son patrones de contraste de píxeles en una imagen. Se utilizan ventanas de diferentes tamaños para detectar estructuras como los ojos, la nariz y la boca.
  En la figura 10 se observa como los clasificadores Haar se basan en características de contraste simples entre regiones rectangulares dentro de una imagen, que se comparan entre sí para determinar la presencia o ausencia de un objeto. Estas características se calculan utilizando el valor Haar.

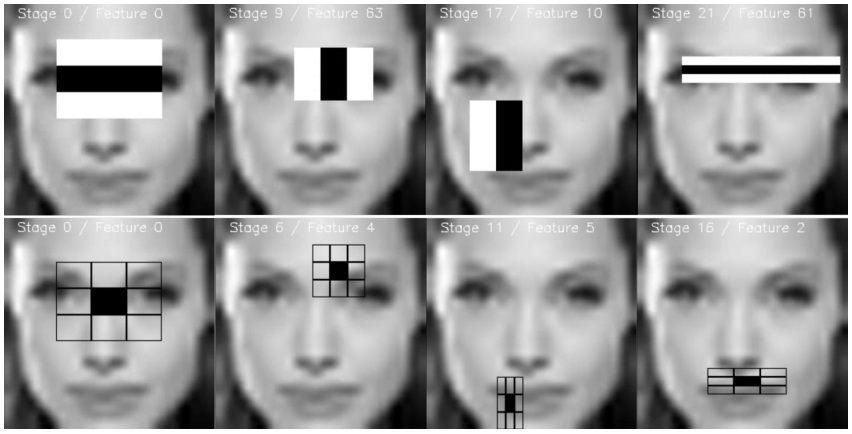

Figura 10. Clasificación en cascada.

- Cálculo del valor Haar: El valor Haar se calcula mediante una suma ponderada de intensidades de píxeles en diferentes áreas de la imagen, representadas como rectángulos blancos y negros. Este valor es crucial para caracterizar la imagen y facilitar la detección del objeto de interés. La fórmula general para calcular el valor de un píxel Haar es:

$$Haar\_value = \sum_{i=1}^{n} w_i \cdot A_i$$

  donde A_i representa el área de los diferentes rectángulos y w_i es el peso asignado a cada rectángulo, basado en la intensidad de píxel. Este cálculo permite que se realice una comparación rápida y eficiente entre las distintas áreas de la imagen.
- Imagen Integral: Representa la imagen como una matriz acumulativa de píxeles, lo que facilita el cálculo de características y reduce el tiempo de procesamiento. Este enfoque optimiza la detección al permitir que se sumen áreas de la imagen de manera eficiente. Gracias a la imagen integral, se pueden calcular los valores Haar de cualquier ventana en constante tiempo, lo que mejora significativamente el rendimiento del algoritmo.
- Clasificadores en Cascada: Los clasificadores en cascada son un conjunto de clasificadores entrenados con miles de imágenes positivas y negativas. Estos clasificadores se utilizan para descartar rápidamente regiones irrelevantes y concentrarse en áreas con alta probabilidad de contener el objeto. El clasificador en cascada permite una detección rápida al aplicar clasificadores simples en las primeras etapas del proceso y clasificadores más complejos en las etapas posteriores. De este modo, se aumenta la precisión de la detección.
- Ventanas Deslizantes: El algoritmo escanea la imagen en diferentes escalas para encontrar objetos de distintos tamaños. Este enfoque permite la detección de objetos con variaciones de tamaño, mejorando la precisión del clasificador. A medida que la ventana se desliza por la imagen, el clasificador Haar analiza la presencia o ausencia de características, ayudando a localizar los objetos en diversas posiciones.

- Diagrama de flujo del clasificador en cascada Haar: En el proceso mostrado en la figura 11, comienza con la adquisición de la imagen, que luego se convierte a escala de grises para simplificar el análisis. A continuación, el clasificador en cascada detecta el rostro. Si se detecta un rostro, el clasificador verifica la presencia de los ojos en la región facial detectada. Si se detectan los dos ojos, la imagen se normaliza en términos de tamaño y orientación. Posteriormente, la imagen se procesa para el reconocimiento facial, donde se compara con una colección de muestras de rostros previamente entrenadas. Este enfoque en cascada mejora la precisión y reduce la tasa de falsos positivos, ya que cada etapa descarta las regiones irrelevantes, concentrándose solo en aquellas áreas con una alta probabilidad de contener el objeto de interés.

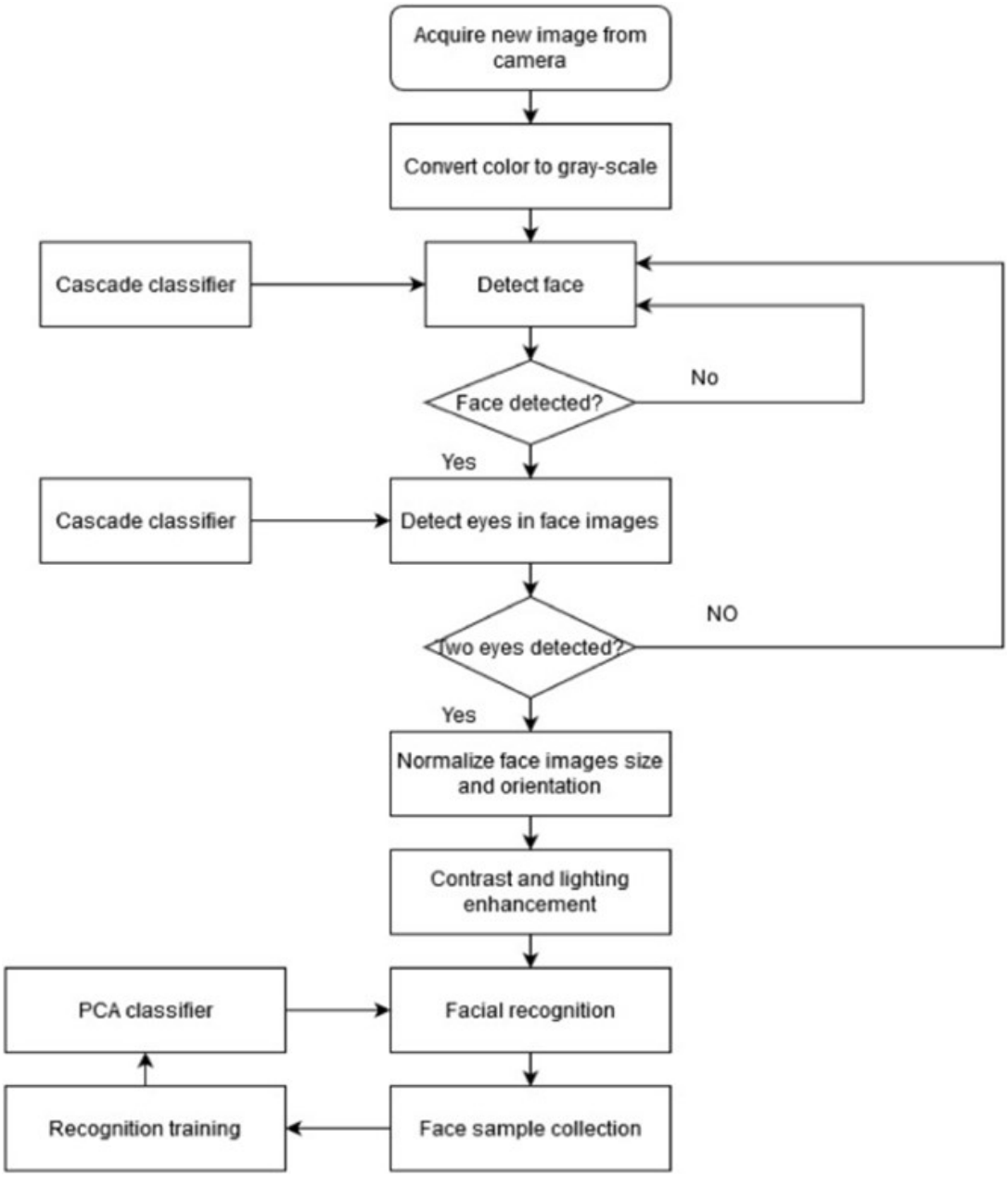

Figura 11. Diagrama de flujo del clasificador en cascada Haar. Figura extraída del libro Procesamiento de Imágenes y Reconocimiento de Patrones de L. A. Martínez y J. M. Ramírez.

- Importancia del clasificador en cascada Haar: La principal ventaja de este clasificador es su alta precisión en la detección de objetos y su baja tasa de falsos positivos. Gracias a la arquitectura en cascada y al uso eficiente de las características de Haar, este algoritmo logra detectar objetos de manera rápida y precisa, lo que lo hace ideal para aplicaciones

en tiempo real, como la detección de rostros en cámaras de seguridad o sistemas de reconocimiento facial.

1.2 Histograma de Gradientes Orientados (HOG) + SVM: El método HOG + SVM es una técnica para la detección de objetos basada en la extracción de características y clasificación mediante una Máquina de Vectores de Soporte (SVM).

- Histograma de Gradientes Orientados (HOG): Extrae características de gradientes de intensidad en una imagen, dividiéndola en celdas y calculando histogramas de orientación de bordes.
- Ventanas Deslizantes: Se escanea la imagen en diferentes escalas para encontrar objetos de distintos tamaños.
- SVM (Support Vector Machine): Clasifica las características extraídas para determinar si una región contiene el objeto de interés.

2. Basados en Deep Learning

2.1 YOLO (You Only Look Once): Los algoritmos de la familia YOLO representan un enfoque avanzado para la identificación y localización de objetos en imágenes o videos en tiempo real. Estas arquitecturas, fundamentadas en redes neuronales profundas, destacan por su capacidad de segmentar y clasificar múltiples elementos simultáneamente dentro de una escena, procesando la información en una sola pasada a través de la red neuronal. A diferencia de otros métodos convencionales que dividen la tarea en varias etapas, YOLO optimiza el rendimiento al realizar la detección de manera directa y eficiente, lo que lo convierte en una solución ideal para aplicaciones que requieren alta velocidad y precisión.

- División en Cuadrículas: La imagen se divide en una cuadrícula de celdas, cada una predice la presencia de un objeto.
- Regresión de Bounding Boxes: Se predicen las coordenadas y la clase del objeto en una sola pasada de la red neuronal.
- Arquitectura Convolucional: Usa capas convolucionales profundas para extraer características y realizar predicciones.

2.2 SSD (Single Shot MultiBox Detector): Es un algoritmo de detección de objetos basado en redes neuronales convolucionales profundas que realiza predicciones de objetos en una sola pasada.

- Cálculos en SSD:
- Predicción de Bounding Box: Para cada cuadro de referencia, las coordenadas predichas se calculan ajustando las coordenadas del cuadro con las diferencias predichas:

$$x_{\text{pred}} = x_{\text{ref}} + \Delta x, \quad y_{\text{pred}} = y_{\text{ref}} + \Delta y$$
$$w_{\text{pred}} = w_{\text{ref}} \cdot e^{\Delta w}, \quad h_{\text{pred}} = h_{\text{ref}} \cdot e^{\Delta h}$$

- Pérdida de Localización (Smooth L1 Loss): La pérdida de localización se calcula para cada predicción de cuadro delimitador con la función Smooth L1:

$$\mathcal{L}_{\text{loc}}(p, g) = \sum_{i \in \{x,y,w,h\}} \mathcal{L}_{\text{SmoothL1}}(p_i - g_i)$$

- Pérdida de Clasificación (Softmax Loss): La probabilidad de clase para cada cuadro delimitador se calcula usando la función Softmax:

$$p(c|x) = \frac{e^{s_c(x)}}{\sum_{c'} e^{s_{c'}(x)}}$$

2.3 Fast R-CNN: R-CNN (Regions with Convolutional Neural Networks) es una familia de modelos para la detección de objetos basada en redes neuronales convolucionales. La versión rápida de R-CNN, como Fast R-CNN y Faster R-CNN, busca mejorar la velocidad y eficiencia de la detección de objetos.

- R-CNN Básico: R-CNN comienza generando propuestas de regiones en la imagen utilizando métodos como Selective Search. Luego, para cada propuesta, se extraen características usando una red convolucional preentrenada (como AlexNet) y se clasifican utilizando un clasificador SVM.

- Fast R-CNN: En Fast R-CNN, se mejora el proceso al pasar la imagen completa a través de la red convolucional una sola vez para obtener un mapa de características. Las propuestas de regiones se extraen directamente del mapa de características, y en lugar de usar un clasificador SVM, se utiliza una capa de clasificación directa conectada a la red.
- Faster R-CNN: Faster R-CNN introduce un módulo adicional llamado RPN (Region Proposal Network), que reemplaza el método de propuestas de regiones como Selective Search. El RPN genera propuestas de regiones directamente desde las características extraídas por la red convolucional, lo que acelera considerablemente el proceso.

V-C.6. Bibliotecas de Aprendizaje Automático y Visión por Computadora

1. Aprendizaje Automático

1.1 TensorFlow: Es una plataforma de código abierto para el aprendizaje automático (machine learning) y la inteligencia artificial (IA) desarrollada por el equipo de Google Brain. Desde su lanzamiento en 2015, ha evolucionado significativamente, convirtiéndose en una de las bibliotecas más populares para la construcción y entrenamiento de modelos de aprendizaje profundo. TensorFlow permite a los desarrolladores expresar algoritmos de aprendizaje automático y ejecutarlos en una amplia variedad de dispositivos, desde dispositivos móviles hasta sistemas distribuidos de gran escala. Utiliza tensores como estructuras de datos fundamentales, que son arreglos multidimensionales de datos. Estos tensores son los bloques básicos sobre los que se realizan las operaciones de aprendizaje automático. La plataforma también utiliza grafos de computación, donde las operaciones y relaciones entre tensores se representan como nodos y bordes, respectivamente. Esta arquitectura basada en grafos permite una ejecución eficiente y paralelizada de las operaciones, optimizando el rendimiento incluso en entornos distribuidos.

Una de las características clave de TensorFlow es su escalabilidad. TensorFlow soporta tanto el entrenamiento como la inferencia, lo que significa que se puede utilizar tanto para entrenar modelos de IA como para hacer predicciones con modelos ya entrenados.

Es compatible con varios lenguajes de programación, incluyendo Python, JavaScript, C++ y Java, lo que facilita su integración en diversas aplicaciones y entornos de desarrollo. TensorFlow también ofrece compatibilidad con bibliotecas populares como NumPy, lo que permite una transición fluida entre operaciones de álgebra lineal y operaciones de aprendizaje automático. [27]

Figura 12. TensorFlow and Python. Elaborado por Developers

TensorFlow se utiliza en una amplia gama de aplicaciones, entre las que se incluyen:

- Reconocimiento de voz: Mejorando la precisión en la transcripción y comprensión del habla, lo que es fundamental en aplicaciones como asistentes virtuales y sistemas de dictado.

- Visión por computadora: Facilitando tareas como la clasificación de imágenes, detección de objetos y segmentación semántica. Esto se utiliza en una variedad de campos, como la automoción, la medicina y la seguridad.
- Procesamiento de lenguaje natural (NLP): Mejorando la traducción automática, la generación de texto y la comprensión del lenguaje. Esto es útil en aplicaciones como chatbots, análisis de sentimientos y sistemas de recomendación.
- Sistemas de recomendación: Personalizando las recomendaciones de productos y servicios para los usuarios, lo que se utiliza en plataformas de comercio electrónico, servicios de streaming y redes sociales.

TensorFlow también ofrece soporte para el entrenamiento de modelos de aprendizaje profundo de manera distribuida, utilizando múltiples GPUs o incluso clústeres de máquinas, lo que permite escalar los modelos a grandes volúmenes de datos y mejorar la velocidad de entrenamiento. Esta capacidad es especialmente útil para entrenar modelos complejos como redes neuronales convolucionales (CNN) y redes neuronales recurrentes (RNN), que son fundamentales para tareas como el reconocimiento de imágenes y el procesamiento de secuencias.

1.2 PyTorch: Es una biblioteca de código abierto para el aprendizaje automático y aprendizaje profundo, ampliamente utilizada en el desarrollo de modelos de redes neuronales. Sus principales características y funcionalidades incluyen:

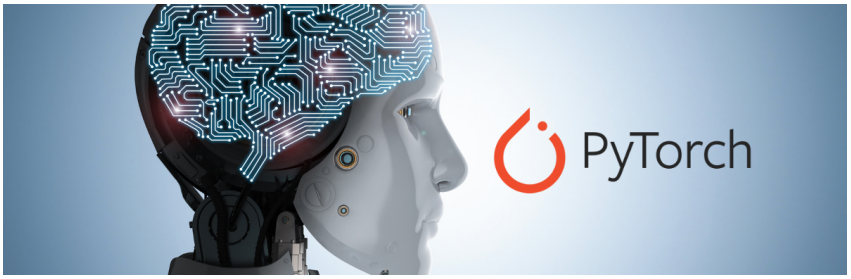

Figura 13. PyTorch. Figura extraída del libro Aprendizaje Profundo con PyTorch de S. A. Imambi, K. B. Prakash y G. R. Kanagachidambaresan.

- Oofrece un soporte eficiente para tensores multidimensionales (similares a los arrays de NumPy) y operaciones sobre ellos. Estos tensores pueden ser procesados en GPU (mediante CUDA) para acelerar las operaciones computacionales, lo que lo hace ideal para entrenar modelos de aprendizaje profundo en grandes volúmenes de datos. Pproporciona un sistema de diferenciación automática que calcula automáticamente los gradientes durante el entrenamiento de los modelos. Esto facilita la optimización de modelos de redes neuronales mediante algoritmos como el descenso de gradiente.[28] La biblioteca incluye un módulo ‘torch.nn‘ que facilita la construcción de redes neuronales. Ofrece clases y funciones para crear capas de redes, funciones de activación, pérdidas y optimizadores, lo que simplifica el diseño de modelos complejos. Utiliza un enfoque de ejecución dinámica, lo que significa que las operaciones se realizan de forma inmediata, sin necesidad de compilar previamente el modelo. Esto facilita la depuración y permite la modificación del modelo sobre la marcha.
- Modelos preentrenados: PyTorch ofrece una serie de modelos preentrenados disponibles a través de la biblioteca ‘torchvision‘ para tareas comunes como clasificación de imágenes, detección de objetos, segmentación y más. Estos modelos pueden ser utilizados como punto de partida y afinados para tareas específicas.
- Soporte para redes neuronales convolucionales (CNN) y recurrentes (RNN): PyTorch es ideal para tareas de visión por computadora y procesamiento de lenguaje natural, ya que soporta fácilmente redes neuronales convolucionales (CNN) y recurrentes (RNN), como LSTM (Long Short-Term Memory) y GRU (Gated Recurrent Unit).

1.3 Scikit-Learn: Scikit-Learn es una biblioteca para Machine Learning clásico. Incluye algoritmos como máquinas de soporte vectorial (SVM), árboles de decisión, regresión logística, entre otros. Es ideal para tareas de clasificación, regresión y agrupamiento.

Scikit-Learn se ha consolidado como una de las herramientas más versátiles y ampliamente adoptadas dentro del ecosistema de Python para la implementación de algoritmos de aprendizaje automático. Esta biblioteca proporciona un extenso conjunto de funciones optimizadas para la clasificación, regresión, agrupamiento y reducción de dimensionalidad, facilitando el desarrollo de modelos predictivos con un enfoque eficiente y accesible. Su arquitectura modular y su compatibilidad con otras librerías del entorno científico de Python la convierten en una opción preferida tanto para investigadores como para profesionales que buscan soluciones escalables y de alto rendimiento en el ámbito del Machine LearninSu principal objetivo es proporcionar herramientas eficientes y fáciles de usar para modelar datos mediante una amplia gama de algoritmos de Machine Learning, así como para realizar análisis estadísticos y de datos.

Principales Características:

- Fácil de usar: Scikit-Learn tiene una API simple y consistente, lo que permite a los usuarios enfocarse más en los aspectos algorítmicos que en las complejidades de la implementación.
- Versatilidad: Admite una gran variedad de algoritmos de Machine Learning para clasificación, regresión, agrupamiento y reducción de dimensionalidad. También incluye herramientas para validación cruzada y selección de modelos.
- Basado en NumPy, SciPy y Matplotlib: Scikit-Learn está construido sobre bibliotecas fundamentales de Python como NumPy (para operaciones numéricas), SciPy (para cálculos científicos) y Matplotlib (para visualización).

Algoritmos Comunes en Scikit-Learn:

- Máquinas de Soporte Vectorial (SVM): Un algoritmo supervisado para clasificación que intenta encontrar un hiperplano que maximiza el margen entre clases. SVM es eficaz en espacios de alta dimensión y en problemas no lineales cuando se usa un truco de núcleo (kernel trick).
- Árboles de Decisión: Un algoritmo de clasificación que divide el espacio de características en base a reglas simples (por ejemplo, si una característica es mayor o menor que un valor umbral). Es intuitivo y fácil de interpretar.
- Modelo Logístico para Clasificación: Técnica estadística utilizada para diferenciar entre dos categorías, calculando la probabilidad de pertenencia mediante una función de activación sigmoide. Su entrenamiento se basa en la optimización de parámetros a través de métodos como el descenso de gradiente y la estimación de máxima verosimilitud, permitiendo su aplicación en problemas de decisión binaria.
- K-Means: Método de segmentación de datos sin supervisión que distribuye las muestras en k conglomerados distintos. Su funcionamiento se centra en la asignación iterativa de cada elemento al grupo cuyo centro de masa minimiza la separación entre puntos, ajustando dinámicamente los centroides hasta estabilizar la distribución de los conjuntos.

1.4 XGBoost: Es una biblioteca de aprendizaje automático que utiliza el algoritmo de boosting para mejorar el rendimiento y la precisión en tareas de clasificación y regresión. Es ampliamente conocido por su alta eficiencia y rendimiento, especialmente en competiciones de ciencia de datos.

Características de XGBoost:

- Optimización por segunda derivada: XGBoost utiliza tanto la primera como la segunda derivada de la función de pérdida, lo que le permite tener un entrenamiento más eficiente y preciso.
- Regularización L1 y L2: XGBoost incluye regularización (L1 y L2) para controlar el sobreajuste, lo que mejora la capacidad de generalización.
- Manejo de valores faltantes: Puede manejar valores faltantes de manera automática durante el entrenamiento.
- Paralelización: XGBoost está altamente optimizado para usar múltiples núcleos de procesamiento y puede aprovechar tanto la CPU como la GPU para mejorar la velocidad de entrenamiento.

1.5 LightGBM: Es otra biblioteca de boosting basada en árboles de decisión, diseñada para ser más eficiente y escalable que XGBoost, especialmente en grandes volúmenes de datos.

Características de LightGBM:

- Optimización basada en histogramas: LightGBM usa una estrategia de histograma para dividir los datos, lo que mejora la velocidad y reduce el uso de memoria.
- Modelo orientado a datos categóricos: LightGBM maneja de forma nativa las características categóricas, eliminando la necesidad de preprocesarlas como variables numéricas.
- Reducción de sobreajuste: Utiliza técnicas como el "drop out"de árboles y la regularización para reducir el sobreajuste.

1.6 CatBoost: es una biblioteca de boosting desarrollada por Yandex, optimizada para manejar características categóricas de manera eficiente. A diferencia de LightGBM y XGBoost, CatBoost requiere menos trabajo de preprocesamiento y es muy efectivo en datos tabulares.

Características de CatBoost:

- Manejo eficiente de variables categóricas: CatBoost tiene un manejo eficiente de variables categóricas utilizando técnicas como el ordenamiento de categorías.
- Reducción de sobreajuste: Al igual que XGBoost y LightGBM, CatBoost incluye mecanismos para controlar el sobreajuste.
- Soporte para múltiples tipos de pérdida: Permite personalizar funciones de pérdida y puede adaptarse a diferentes tipos de tareas de predicción.

2. Visión por Computadora

2.1 OpenCV: Es la biblioteca más utilizada en procesamiento de imágenes y visión artificial. Ofrece herramientas para tareas como detección de bordes, reconocimiento de objetos y seguimiento de movimiento.

Contiene más de 2,500 algoritmos optimizados y listos para ser utilizados en tareas de visión por computadora y aprendizaje automático. Está diseñada para ser rápida y eficiente, permitiendo el procesamiento de imágenes y videos en tiempo real. La biblioteca está disponible en diversas plataformas como Linux, Windows, macOS y Android, y es compatible con múltiples lenguajes de programación como C++, Python y Java [29].

Características principales de OpenCV:

- Disponible para diversas plataformas, incluidas Linux, Windows, macOS y Android. OpenCV está optimizada para ejecutar operaciones en tiempo real, lo que lo hace ideal para aplicaciones como el reconocimiento facial en videos en vivo o la detección de objetos en tiempo real. Soporta la ejecución en múltiples hilos, lo que mejora el rendimiento en sistemas con múltiples núcleos de procesador. Utiliza bibliotecas como Intel's IPP (Integrated Performance Primitives), TBB (Threading Building Blocks) y CUDA para aprovechar las capacidades de las tarjetas gráficas (GPU) en ciertas operaciones.

Principales funciones y operaciones: OpenCV permite realizar una amplia variedad de operaciones en imágenes y videos:

- Carga y Guardado de Imágenes: Funciones como ‘cv2.imread()‘ para cargar y ‘cv2.imwrite()‘ para guardar imágenes.
- Transformaciones de Imágenes: Escalado (‘cv2.resize()‘), rotación (‘cv2.getRotationMatrix2D()‘), recorte (usando indexación), y filtrado de imágenes (desenfoque Gaussiano, detección de bordes con ‘cv2.Canny()‘).
- Detección de Características: Técnicas como la detección de bordes (Canny), esquinas (Harris, Shi-Tomasi) y la detección de objetos mediante clasificadores en cascada (‘cv2.CascadeClassifier()‘).
- Operaciones Matemáticas: Manipulación de matrices, operaciones algebraicas, cálculo de histogramas, y umbralización de imágenes.
- Transformaciones Geométricas: Transformaciones afines, estimación de homografía y transformaciones de perspectiva.
- Análisis de Video: Captura de video con ‘cv2.VideoCapture()‘, análisis de flujo óptico (‘cv2.calcOpticalFlowPyrLK()‘), y seguimiento de objetos.

2.2 MediaPipe: Es una biblioteca de código abierto desarrollada por Google que facilita el desarrollo de aplicaciones de visión por computadora, aprendizaje automático y procesamiento de señales en tiempo real. Está diseñada para simplificar el uso de modelos de aprendizaje profundo para tareas como el reconocimiento de manos, el análisis de la pose humana y la detección de rostros.

En la Figura 14, se muestra un ejemplo de la detección de rostro mediante Mediapipe, donde se muestran los puntos faciales distribuidos a lo largo de las características del rostro, permitiendo su análisis y seguimiento.

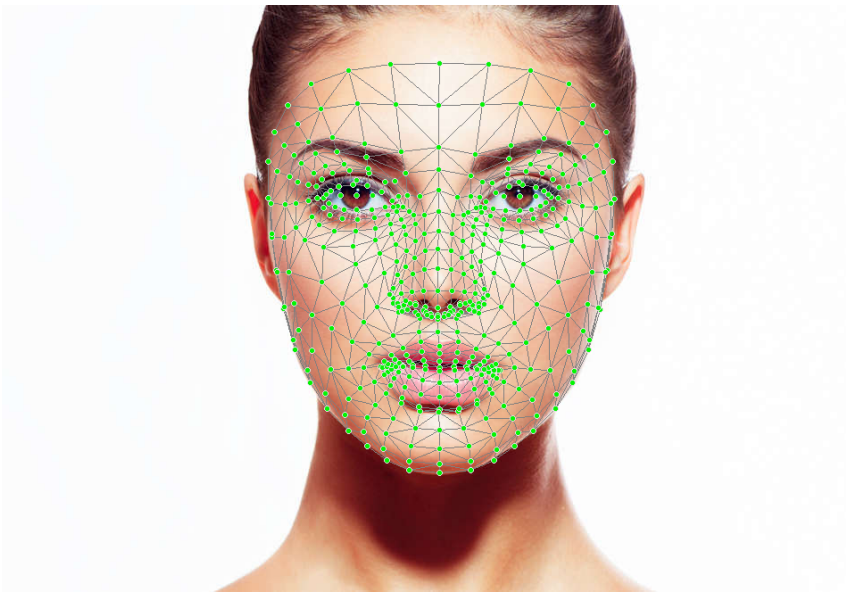

Figura 14. Detección de rostro mediante Mediapipe. Figura extraída del libro Reconocimiento de Rostros con Mediapipe de C. A. Lugaresi, J. L. Tang, H. S. Nash, C. D. McClanahan, E. L. Uboweja, M. I. Hays y M.A. Grundmann

Características principales de MediaPipe:

- Procesamiento en Tiempo Real: Está optimizada para ejecutar tareas complejas de visión por computadora y aprendizaje automático en tiempo real, lo que la hace ideal para aplicaciones que requieren una alta velocidad de procesamiento, como la realidad aumentada. [30]
- Modelos Preentrenados: Ofrece modelos preentrenados para diversas tareas de visión artificial, como la detección de manos, rostros, poses y objetos. Estos modelos son altamente optimizados y fáciles de integrar.
- Multiplataforma:Es compatible con plataformas como Android, iOS, Linux y Windows, lo que permite el desarrollo de aplicaciones móviles y de escritorio.

Componentes y Módulos de MediaPipe: Se organiza en varios módulos y componentes que permiten realizar tareas complejas de procesamiento de imágenes, videos y señales:

- Graph API: Se basa en un sistema de gráficos (graphs) que conecta nodos (tareas de procesamiento) en un flujo de trabajo. Cada nodo realiza una operación en los datos de entrada y pasa el resultado al siguiente nodo.
- Calculadores (Calculators): Los calculadores son los componentes fundamentales en MediaPipe, que realizan el procesamiento real de los datos. Pueden ser tareas de detección de características, clasificación, transformación de imágenes, etc.
- Tareas de Visión por Computadora: MediaPipe incluye varios modelos y módulos para tareas como:
  - Utiliza un modelo preentrenado que detecta y realiza un seguimiento de las manos en tiempo real, permitiendo la interacción gestual. Ofrece modelos para detectar y hacer seguimiento de las posturas del cuerpo humano, identificando las posiciones clave de las articulaciones. Incluye un modelo para la detección de rostros en imágenes o videos, que puede ser utilizado en aplicaciones de reconocimiento facial. Proporciona modelos para la detección en tiempo real de diversos objetos en imágenes y videos.
- MediaPipe Solutions: MediaPipe Solutions es una colección de bibliotecas que permiten integrar soluciones listas para usar en aplicaciones como la detección de manos, el seguimiento de la postura o la segmentación semántica.

- Modelos de Aprendizaje Automático: MediaPipe integra modelos de aprendizaje profundo, como redes neuronales convolucionales (CNN) y redes de seguimiento de objetos, para tareas como clasificación, detección y estimación.

Principales funciones y operaciones de MediaPipe: MediaPipe ofrece una amplia variedad de funcionalidades para tareas de visión por computadora:

- Procesamiento de Videos en Tiempo Real: Permite capturar y procesar videos en tiempo real utilizando la cámara de dispositivos como smartphones, computadoras o cámaras web.
- Detección y Seguimiento de Manos: El modelo de MediaPipe para la detección de manos puede identificar hasta 21 puntos clave de cada mano en tiempo real, lo que permite el control por gestos.
- Estimación de Pose Humana: Los modelos de estimación de pose detectan las posiciones de las articulaciones del cuerpo humano en tiempo real, lo que es útil para aplicaciones de ejercicio, danza, y salud.
- Reconocimiento Facial: MediaPipe permite la detección de características faciales como ojos, cejas, boca y nariz, útil en aplicaciones de seguridad y análisis facial.
- Segmentación Semántica: Los modelos de segmentación semántica de MediaPipe permiten la segmentación de objetos o personas en las imágenes, útil para aplicaciones de AR/VR y edición de imágenes.
- Clasificación de Imágenes y Objetos: MediaPipe incluye herramientas para la clasificación de objetos en imágenes y la detección de múltiples objetos simultáneamente.

2.3 dlib: dlib es una biblioteca de C++ ampliamente utilizada en el procesamiento de imágenes y visión por computadora, especializada en tareas como la detección de rostros, el aprendizaje automático y el seguimiento de objetos. Además de su eficiencia, dlib es conocida por su implementación de modelos de aprendizaje profundo, especialmente en redes neuronales convolucionales (CNN), y su capacidad para trabajar con máquinas de soporte vectorial (SVM) para clasificación y regresión. Su uso en aplicaciones de visión artificial, como el reconocimiento facial y la alineación de rostros, es muy popular debido a su alto rendimiento y precisión. [31]

Características principales de dlib:

- Dlib utiliza un modelo basado en un clasificador en cascada de características HOG (Histograms of Oriented Gradients) y una máquina de soporte vectorial (SVM) para detectar rostros de manera eficiente en imágenes o video. Ofrece un modelo de entrenado para detectar 68 puntos clave en rostros humanos, facilitando tareas como la alineación de rostros y el análisis de expresiones faciales. Incluye herramientas para entrenar y usar redes neuronales profundas para clasificación de imágenes, segmentación y otras tareas de visión por computadora. Utiliza máquinas de soporte vectorial (SVM) tanto para clasificación como para regresión, permitiendo el entrenamiento de modelos de alto rendimiento con grandes conjuntos de datos. Está diseñada para aprovechar el poder de procesamiento de sistemas con múltiples núcleos, lo que mejora la eficiencia en tareas complejas como el entrenamiento de modelos o el procesamiento de imágenes en tiempo real.
  Dlib, es una de las librerias de machine learning, que ofrece una solución robusta detectando puntos clave en las características del rostro. En la Figura 15 se presenta un ejemplo de la identificación de landmarks usando Dlib, donde se pueden ver los puntos faciales que ayudan a localizar las principales características de la cara.

Módulos y Funcionalidades de dlib: Incluye una serie de módulos y funciones que permiten realizar operaciones avanzadas de visión por computadora y aprendizaje automático:

- Detección de Rostros: Incluye un detector de rostros basado en HOG y SVM, que es altamente eficiente y rápido para detectar rostros en imágenes.
- Alineación Facial: Utilizando el modelo, dlib permite detectar 68 puntos clave de un rostro. Estos puntos incluyen las ubicaciones de los ojos, nariz, cejas y boca, lo que facilita la alineación del rostro mediante transformaciones geométricas. Clasificación con SVM: Proporciona herramientas para entrenar clasificadores basados en máquinas de soporte vectorial (SVM) utilizando el algoritmo `svm_rank_trainer()` para tareas como la clasificación de imágenes y la regresión.

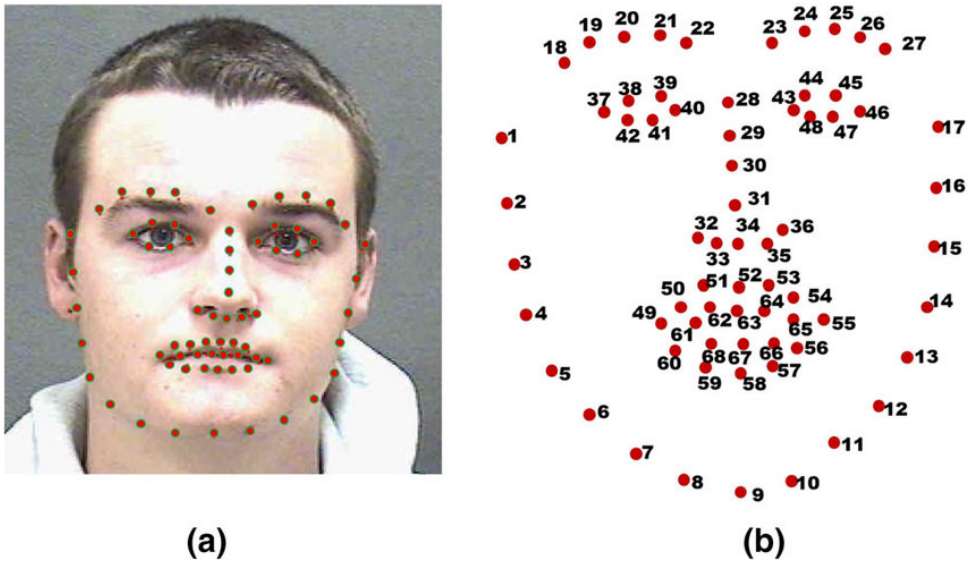

Figura 15. Identificación landmarks usando Dlib.Figura extraída del libro Dlib: A Toolkit for Machine Learning de D. E. King

- Redes Neuronales Convolucionales: Facilita la creación, entrenamiento y evaluación de redes neuronales convolucionales para tareas de clasificación de imágenes. Las redes pueden ser entrenadas con datos etiquetados y utilizadas para tareas de reconocimiento de objetos y personas.
- Optimización y Paralelización: Optimiza el rendimiento del procesamiento mediante la paralelización en sistemas multi-core, aprovechando las capacidades de los procesadores modernos para mejorar la velocidad de ejecución, especialmente durante el entrenamiento de modelos o el procesamiento de grandes volúmenes de datos.

Cálculos y Algoritmos:

- HOG y SVM para Detección de Rostros: La detección de rostros se realiza mediante la combinación de características HOG con una máquina de soporte vectorial. El HOG extrae características basadas en la orientación de los gradientes en las imágenes, mientras que el SVM se encarga de clasificarlas como rostros o no rostros. El modelo SVM se entrena utilizando un conjunto de datos etiquetado y luego se usa para hacer predicciones sobre nuevas imágenes. Alineación de Rostros: La alineación se lleva a cabo utilizando una transformación afín basada en los puntos clave detectados en el rostro. Dado un conjunto de puntos clave $P_1, P_2, ..., P_n$, se calcula la transformación que alinea el rostro a una orientación estándar (por ejemplo, de frente). Esto se logra utilizando una matriz de transformación afín A que minimiza la distancia entre los puntos de referencia y su ubicación deseada:

$$A = \arg\min_{A} \sum_{i=1}^{n} \|A \cdot P_i - T_i\|^2$$

  donde T_i es el conjunto de puntos objetivo alineados.
- Clasificación con SVM: dlib implementa el entrenamiento de SVM utilizando el método de margen máximo para encontrar el hiperplano que mejor separa las clases. El objetivo es maximizar el margen ρ entre las clases:

$$\rho = \arg\max_{\rho} \frac{1}{\|\mathbf{w}\|}$$

  donde $\mathbf{w}$ es el vector de pesos del SVM. Dlib permite la implementación y entrenamiento de redes neuronales convolucionales, que emplean capas de convolución para extraer características de imágenes y capas de pooling para reducir la dimensionalidad. Las redes son entrenadas utilizando técnicas de retropropagación para minimizar el error en tareas de clasificación.

2.4 Detectron2: Es un sistema de visión por computadora de código abierto basado en la biblioteca de PyTorch, desarrollado por Facebook AI Research (FAIR). Está diseñado para la detección y segmentación de objetos, y es altamente eficiente en tareas complejas como la segmentación de instancias, la detección de objetos y la estimación de poses. Detectron2 ofrece implementaciones de última generación de arquitecturas de redes neuronales profundas (DNN) como Faster R-CNN, Mask R-CNN, RetinaNet y otras, y se destaca por su flexibilidad y escalabilidad.

Aspectos Técnicos y Funciones Clave de Detectron2:

- Utiliza una arquitectura modular que permite a los usuarios personalizar y extender el sistema fácilmente para tareas específicas. Cada componente está diseñado para ser independiente, lo que facilita la implementación de nuevos algoritmos y el ajuste de modelos preentrenados. Está optimizado para el entrenamiento a gran escala en múltiples GPUs, lo que lo hace adecuado para manejar grandes conjuntos de datos. Utiliza técnicas avanzadas de optimización como el aprendizaje distribuido, lo que permite entrenar modelos de redes neuronales profundas con altos volúmenes de datos sin sacrificar la eficiencia. Es capaz de detectar objetos en imágenes, clasificarlos y determinar su ubicación utilizando marcos de detección como el de región propuesta (RPN). Utiliza redes neuronales convolucionales (CNN) para extraer características jerárquicas de las imágenes, lo que permite detectar objetos con alta precisión. [29].

Cálculos y Algoritmos de Detectron2: Se basa en una serie de algoritmos avanzados y técnicas matemáticas que le permiten lograr una alta precisión en tareas de visión por computadora:

- Redes Propuestas de Regiones (RPN): Utiliza un modelo de red neuronal llamado RPN para proponer regiones en una imagen que contienen objetos de interés. El RPN genera (anchors) sobre una imagen y las evalúa para determinar si contienen un objeto. Las anclas se utilizan para predecir la localización de objetos en la imagen. El cálculo para la propuesta de región se puede expresar como:

$$\text{Anchor}(p) = \arg\max_{p} \left(\sum_{i} \|R_i - T_i\|^2 \right)$$

  donde R_i es la región propuesta por el modelo, T_i es la región de verdad de terreno (ground truth), y p es el punto de anclaje.
- Máscara de Segmentación (Mask R-CNN): En tareas de segmentación, Detectron2 utiliza el enfoque de Mask R-CNN. El modelo predice una máscara de píxeles para cada objeto detectado en la imagen. La máscara se obtiene mediante una red adicional que aprende a generar una segmentación binaria por cada región propuesta. La máscara se calcula como:

$$\hat{M}(x, y) = \text{Sigmoid} \left(\sum_{k} \mathbf{W}_k \cdot \mathbf{F}_k(x, y) + b \right)$$

  donde $\hat{M}(x, y)$ es la probabilidad de que el píxel (x, y) pertenezca a la máscara de un objeto, $\mathbf{W}_k$ es el conjunto de pesos de la red para la máscara, y $\mathbf{F}_k(x, y)$ son las características extraídas del píxel (x, y).
- **Optimización de la Red de Convolución (CNN)**: Detectron2 hace uso de redes neuronales convolucionales (CNN) de alta capacidad, como ResNet y FPN (Feature Pyramid Networks). Estas redes están diseñadas para extraer características jerárquicas de las imágenes, mejorando la detección de objetos a diferentes escalas. El cálculo de la función de activación en una CNN para un píxel x en la capa l se expresa como:

$$\mathbf{z}_l(x) = \text{ReLU} \left(\sum_{i} W_i \cdot \mathbf{x}_{l-1}(i) + b_l \right)$$

  donde $\mathbf{z}_l(x)$ es la activación de la capa l, W_i son los pesos de la convolución y $\mathbf{x}_{l-1}(i)$ es la entrada de la capa anterior.
- Entrenamiento y Optimización de Modelos: El proceso de entrenamiento en Detectron2 utiliza técnicas como el descenso de gradiente estocástico (SGD) y la retropropagación

para ajustar los parámetros de la red. La función de pérdida total de un modelo puede ser calculada como:

$$\mathcal{L}_{total} = \mathcal{L}_{RPN} + \mathcal{L}_{detection} + \mathcal{L}_{mask}$$

donde $\mathcal{L}_{RPN}$ es la pérdida de la región propuesta, $\mathcal{L}_{detection}$ es la pérdida de detección, y $\mathcal{L}_{mask}$ es la pérdida de segmentación de la máscara. [32].

VI. MARCO METODOLÓGICO

Se llevó a cabo el desarrollo de un sistema inteligente capaz de identificar signos de somnolencia en conductores mediante visión artificial. La captura de imágenes en tiempo real se realiza a través de una cámara Pi v2 Noir, cuya información visual es procesada mediante algoritmos especializados en el reconocimiento de rasgos faciales. Este procesamiento permite detectar indicadores de fatiga y posibles pérdidas de alerta durante la conducción, optimizando así la seguridad en el vehículo. Estos algoritmos permiten la detección y caracterización de señales fisiológicas y de comportamiento, como la frecuencia de parpadeo, el cierre ocular prolongado y la inclinación de la cabeza, generando así parámetros cuantificables que indican niveles de fatiga.

Posteriormente, el sistema ejecuta cálculos matemáticos y modelos de aprendizaje automático para clasificar el estado del conductor, determinando si presenta signos de somnolencia. Una vez identificado un estado crítico, el sistema activa un mecanismo de alerta sonora para advertir al conductor y prevenir situaciones de riesgo.

La Figura 16 ilustra el flujo metodológico empleado en la implementación del proyecto. A continuación, se presentan las fases clave que conforman el proceso de desarrollo del sistema, describiendo cada una de ellas en detalle..

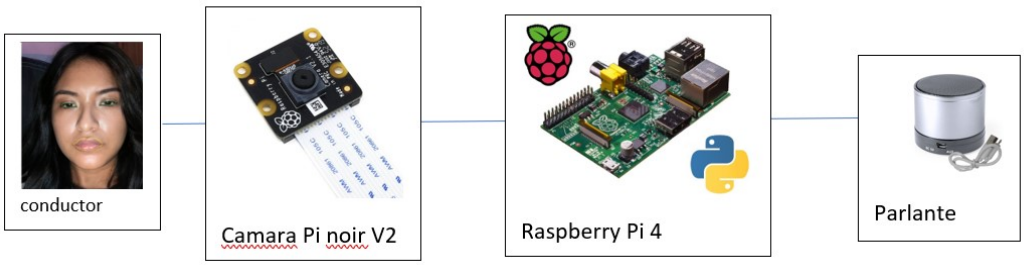

Figura 16. La arquitectura del prototipo del detector de somnolencia en vehículos

El diagrama de flujo muestra el proceso de detección de somnolencia en conductores, desde la captura de imágenes hasta el análisis de patrones faciales que indican fatiga. La Figura 17 detalla cómo el sistema identifica los signos de somnolencia a través de algoritmos de visión artificial.

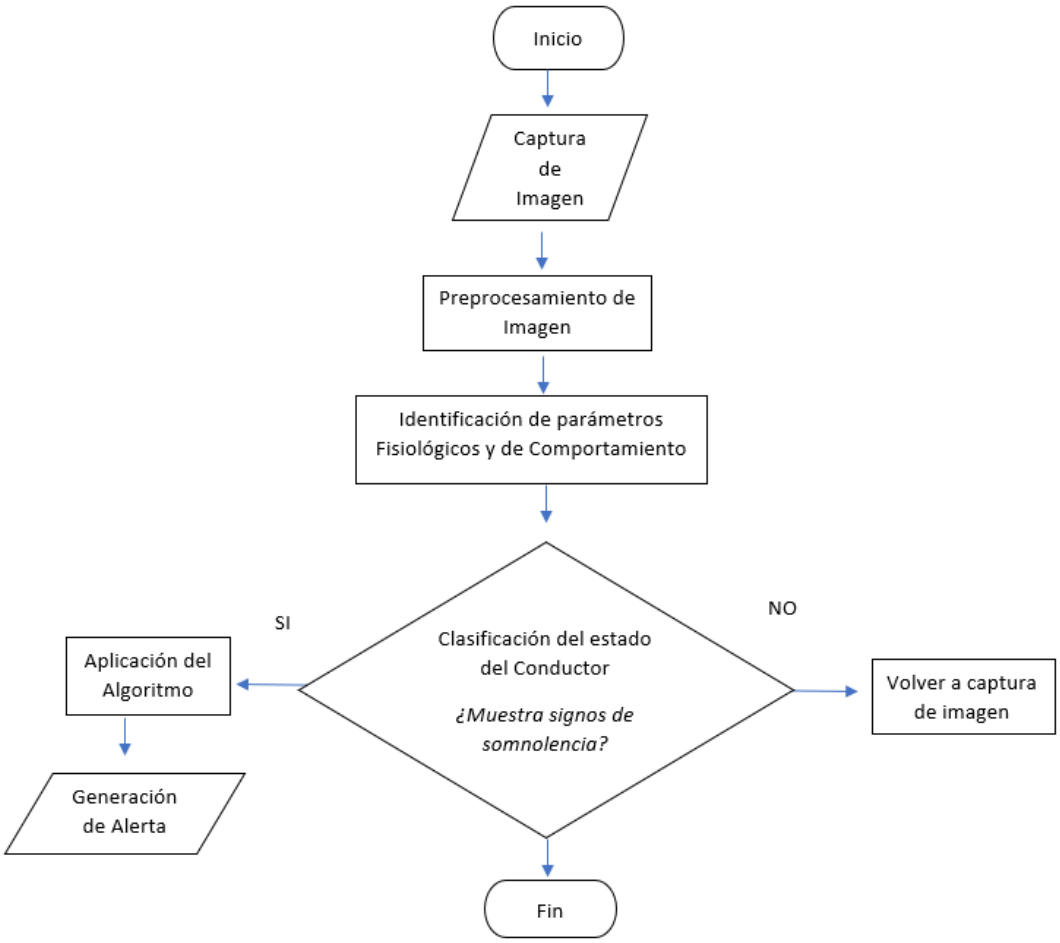

Figura 17. Diagrama de flujo de proceso

VI-A. Selección de Componentes

La selección de los componentes de hardware es un factor determinante en la eficiencia y precisión del sistema de detección de somnolencia. En este apartado se detallan los dispositivos elegidos, justificando su idoneidad en función de sus especificaciones técnicas y su compatibilidad con los requerimientos del proyecto.

VI-A.1. Módulo de Captura de Imágenes: Raspberry Pi Camera Module V2 Noir

Para la detección de signos de fatiga ocular es fundamental disponer de un sistema de adquisición de imágenes que garantice una captura precisa bajo diversas condiciones lumínicas. Se ha seleccionado la **Raspberry Pi Camera Module V2 Noir**, un módulo fotográfico optimizado para aplicaciones en entornos de baja iluminación, gracias a la ausencia del filtro de infrarrojos (IR-cut), lo que permite su uso con iluminación infrarroja externa.

Parámetro	Especificación
Modelo	Raspberry Pi Camera Module V2 Noir
Sensor	Sony IMX219 (8 MP)
Resolución máxima	3280 × 2464 píxeles
Resolución de video	1080p @ 30 fps
Ángulo de visión	62.2° horizontal, 48.8° vertical
Compatibilidad	Compatible con luz IR externa
Interfaz	CSI (Camera Serial Interface)
Dimensiones	25 mm × 23 mm × 9 mm

Tabla 2. Especificaciones técnicas de la cámara Raspberry Pi Noir V2.

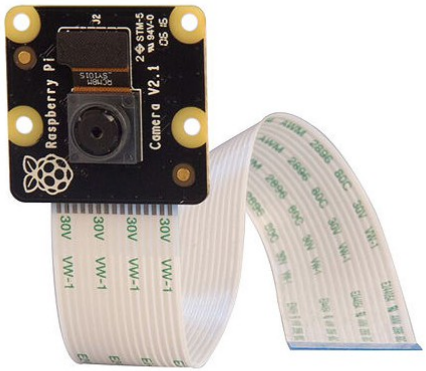

Figura 18. Cámara Raspberry Pi Noir V2.

VI-A.2. Sensores Complementarios para la Detección de Somnolencia

Además del módulo de captura de imágenes, se han considerado sensores complementarios que pueden mejorar la precisión del sistema de detección de somnolencia:

- **Sensor Infrarrojo de Profundidad (ToF VL53L0X):** Este sensor permite medir distancias en tiempo real y puede utilizarse para detectar cambios en la posición de la cabeza del usuario, un posible indicador de somnolencia.
- **Acelerómetro y Giroscopio (MPU-6050):** Un módulo de 6 ejes que mide la inclinación y aceleración, útil para detectar movimientos involuntarios de la cabeza que pueden asociarse a microsueños.
- **Sensor de Ritmo Cardíaco (MAX30102):** Puede complementar el análisis detectando cambios en la frecuencia cardíaca, los cuales suelen manifestarse antes de la somnolencia.

VI-A.3. Unidad de Procesamiento

El procesamiento de imágenes y la ejecución de algoritmos de visión artificial requieren una plataforma con capacidad de cómputo suficiente para manejar operaciones en tiempo real. Se evaluaron distintas opciones de microcomputadores en función de su potencia de cálculo, consumo energético y compatibilidad con el sistema de adquisición de imágenes.

1. **Raspberry Pi 4 Model B**

   El **Raspberry Pi 4 Model B** es una *Single Board Computer* (SBC) que integra un procesador ARM Cortex-A72 de cuatro núcleos a 1.5 GHz, lo que permite un procesamiento eficiente de imágenes y el manejo de algoritmos en tiempo real. Su compatibilidad con Linux y su interfaz de pines GPIO facilitan la integración con sensores y periféricos adicionales.

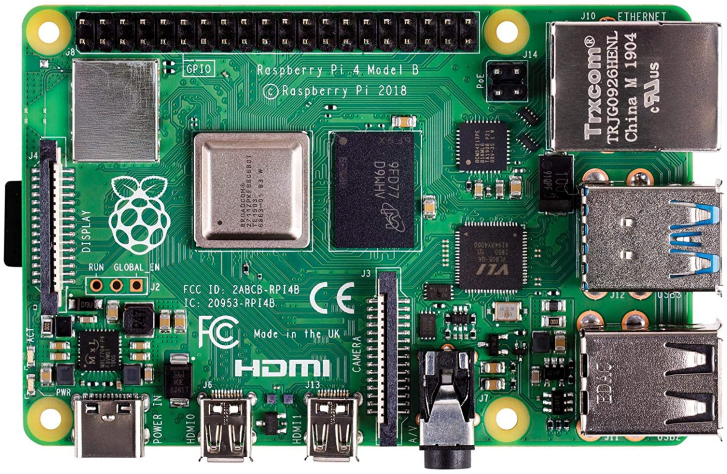

Figura 19. Raspberry Pi 4 Model B.

2. **NVIDIA Jetson Nano**

   Diseñado para aplicaciones de inteligencia artificial, el **Jetson Nano** incorpora una GPU NVIDIA Maxwell de 128 núcleos CUDA, lo que lo convierte en una opción idónea para el despliegue de modelos de aprendizaje profundo en visión por computadora. Sin embargo, su mayor consumo energético y el costo elevado en comparación con la Raspberry Pi 4 limitan su viabilidad dentro del presupuesto del proyecto.

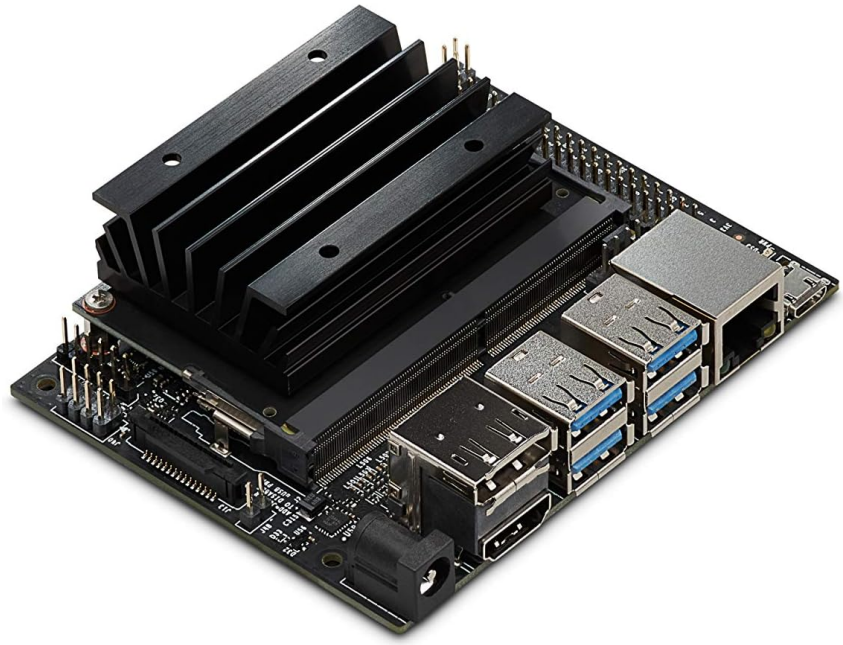

Figura 20. NVIDIA Jetson Nano.

3. **Brix Celeron BPCE 3455C**

   El **Brix Celeron BPCE 3455C** de Gigabyte es una mini PC basada en un procesador Intel Celeron N3455 de cuatro núcleos. Su principal ventaja radica en su compatibilidad con arquitecturas

x86 y la posibilidad de expandir almacenamiento y memoria RAM. Sin embargo, su mayor consumo energético y su menor soporte en aplicaciones de visión artificial lo hacen menos conveniente para este proyecto.

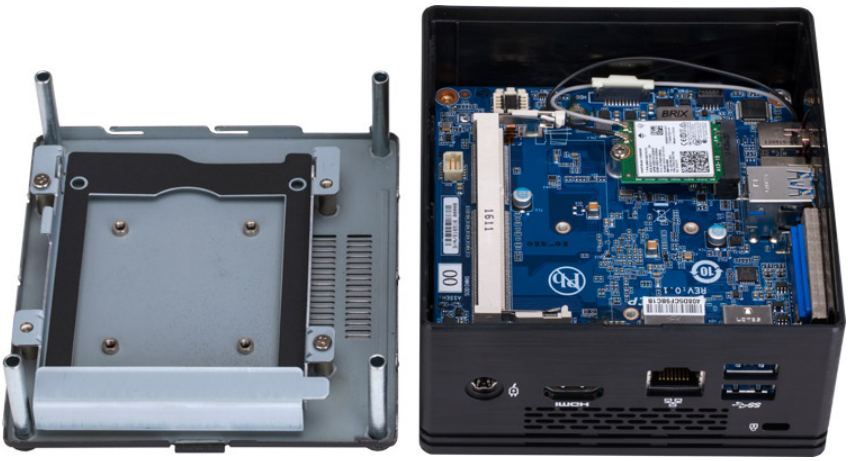

Figura 21. Brix Celeron BPCE 3455C.

VI-A.4. Consideraciones Térmicas y Energéticas

El uso continuo del sistema de detección de somnolencia implica un consumo energético y una disipación térmica que deben ser controlados para evitar sobrecalentamiento y fallos en la operación. Se han implementado las siguientes medidas:

- **Disipador de Calor y Ventilación Activa:** En la Raspberry Pi 4 se recomienda el uso de un disipador de aluminio con ventilador para mantener temperaturas óptimas durante la ejecución de modelos de visión artificial.
- **Fuente de Alimentación Estable:** Se recomienda una fuente de al menos 5V/3A para la Raspberry Pi 4, garantizando un suministro energético adecuado para todos los periféricos conectados.
- **Optimización de Algoritmos:** Reducir el uso de ciclos de CPU mediante optimización del código, evitando procesamiento innecesario y favoreciendo técnicas como el uso de modelos optimizados con TensorFlow Lite.

VI-A.5. Comparativa y Selección del Microcomputador

Para determinar la opción más adecuada, se realizó una comparación, como se observa en la Tabla 3 entre los dispositivos evaluados considerando parámetros clave como capacidad de procesamiento, conectividad, consumo energético y costo.

Parámetro	Raspberry Pi 4	Jetson Nano	Brix Celeron
Procesador	Cortex-A72 (4x, 1.5 GHz)	Cortex-A57 (4x, 1.43 GHz)	Intel Celeron N3455 (4x, 2.2 GHz)
GPU	VideoCore VI	Maxwell (128 CUDA)	Intel HD Graphics 500
RAM	2GB, 4GB u 8GB LPDDR4	4GB LPDDR4	Hasta 8GB DDR3
Almacenamiento	microSD	microSD / SSD (USB)	SSD / HDD (SATA)
Consumo	15W máx.	20W máx.	36W máx.
Interfaz Cámara	CSI (2)	CSI (2)	USB 3.0 / HDMI
Soporte IA	TensorFlow Lite, OpenCV	TensorRT, CUDA, OpenCV	OpenCV, Intel OpenVINO
Conectividad	USB 3.0, Wi-Fi, BT	USB 3.0, Ethernet	USB 3.0, Wi-Fi, Ethernet
Costo Aprox.	$60 - $90 USD	$100 - $150 USD	$200 - $250 USD

Tabla 3. Comparativa de microcomputadores para visión artificial.

Para determinar la plataforma de hardware más adecuada para este proyecto, se evaluaron distintas opciones en función de su rendimiento, facilidad de integración y viabilidad económica. Si bien el **NVIDIA**

Jetson Nano es reconocido por su capacidad para ejecutar modelos avanzados de inteligencia artificial, la **Raspberry Pi 4 Model B** se destaca como la alternativa más eficiente dentro del contexto de este sistema. A continuación, se presentan las razones fundamentales que justifican esta elección:

1. **Soporte técnico y ecosistema de desarrollo**

   La Raspberry Pi 4 cuenta con una de las comunidades más amplias en el ámbito de la computación embebida, lo que garantiza acceso a foros de ayuda, documentación detallada y una gran cantidad de bibliotecas optimizadas para sus características. Este respaldo facilita la resolución de problemas y acelera el desarrollo del software, un aspecto clave en proyectos de investigación y prototipado rápido.

2. **Relación costo-beneficio optimizada**

   Aunque el Jetson Nano ofrece un rendimiento superior en procesamiento gráfico, su precio base no incluye ciertos elementos esenciales como conectividad inalámbrica, lo que implica costos adicionales para la adquisición de módulos de Wi-Fi y Bluetooth. En contraste, la Raspberry Pi 4 incorpora estas funcionalidades de manera nativa, reduciendo así la inversión total y simplificando la integración del hardware.

3. **Compatibilidad con herramientas de visión artificial**

   El sistema está diseñado para procesar imágenes en tiempo real utilizando algoritmos de detección de somnolencia. Para ello, se hace uso de bibliotecas ampliamente compatibles con la Raspberry Pi, como *OpenCV*, *TensorFlow Lite* y *dlib*, que permiten la implementación de modelos de aprendizaje profundo optimizados para dispositivos de bajo consumo. Su arquitectura basada en ARM ofrece un equilibrio adecuado entre eficiencia y rendimiento computacional.

4. **Capacidad de procesamiento suficiente**

   Aunque la GPU de la Raspberry Pi 4 es menos potente que la del Jetson Nano, su CPU de cuatro núcleos **Cortex-A72** y su capacidad de hasta **8 GB de RAM** proporcionan el poder de cómputo necesario para ejecutar modelos de visión artificial en tiempo real, siempre que se utilicen técnicas de optimización como la cuantización de redes neuronales y la reducción de carga computacional en el preprocesamiento de imágenes.

5. **Bajo consumo energético y facilidad de implementación**

   Otro factor determinante es la eficiencia energética. Mientras que el Jetson Nano requiere una fuente de alimentación de hasta **10W**, la Raspberry Pi 4 mantiene un consumo menor, lo que la hace más adecuada para sistemas portátiles o embebidos que dependen de baterías o fuentes de energía limitadas. Además, su compatibilidad con periféricos estándar y su facilidad de montaje permiten una implementación más sencilla en entornos reales.

La **Raspberry Pi 4 Model B** emerge como la opción más equilibrada para el sistema de detección de somnolencia. Su combinación de rendimiento adecuado, bajo costo, amplio soporte comunitario y compatibilidad con herramientas de desarrollo garantiza una implementación eficiente y sostenible dentro del proyecto.

VI-B. Activación y Configuración

El sistema comienza con la ejecución del código que inicia y prepara todos los componentes necesarios para detectar la somnolencia en conductores mediante visión artificial. El proceso se describe a continuación:

1. Preparación del entorno: Al encender la Raspberry Pi, se carga el sistema operativo y se inicializan los servicios que gestionan los dispositivos conectados. Esto incluye la preparación de la cámara, así como la verificación de otros dispositivos necesarios, como el parlante.

2. Cargar bibliotecas:

   2.1 CV2 (OpenCV): Captura las imágenes en tiempo real desde la cámara. La biblioteca facilita el acceso a la cámara y la manipulación de las imágenes, permitiendo:
   Capturar fotogramas de la cámara para analizarlos en tiempo real. Preprocesar las imágenes, convirtiéndolas a escala de grises y facilitando la detección de características faciales. Mostrar las imágenes procesadas en una ventana para la depuración y visualización.

2.2 Numpy: NumPy es crucial para la manipulación de datos y matrices de imagen. Cuando se capturan las imágenes, estas se representan como matrices de píxeles, y NumPy facilita:

El procesamiento de las matrices de imágenes para aplicar transformaciones, filtros y realizar cálculos sobre las imágenes. Es útil cuando se necesitan calcular distancias entre puntos faciales, en este caso, la distancia entre los ojos para detectar el parpadeo y el bostezo y realizar cálculos relacionados con la geometría facial.

2.3 Dlib: Es la biblioteca principal para la detección de rostros y el análisis de características faciales. Sus roles específicos son:

Detección de rostros en tiempo real: Utiliza el detector de rostros para localizar los rostros en las imágenes capturadas por la cámara. Predicción de puntos faciales: Con el predictor, se obtienen los 68 puntos clave del rostro, lo que permite el análisis detallado de los ojos y la boca, esenciales para detectar signos de somnolencia.

2.4 Pygame: Es usado para emitir un sonido de alerta que despierte al conductor.

2.5 smtplib: La biblioteca smtplib es fundamental para enviar las notificaciones por correo electrónico.

3. Inicialización de la cámara: Una vez que las bibliotecas necesarias están cargadas, la cámara se inicializa. Esta cámara está conectada a la Raspberry Pi y, a través de las bibliotecas correspondientes, se configura para capturar imágenes en tiempo real.

   El proceso de inicialización de la cámara incluye la configuración de parámetros como la resolución de las imágenes y la frecuencia de captura. Una vez que la cámara está configurada, comienza a capturar imágenes de la cara del conductor en tiempo real para su posterior análisis en busca de signos de somnolencia. A continuación se presenta el esquema de la fase de adquisición y captura de datos, que incluye el procesamiento de imágenes. Este proceso es crucial para la correcta recopilación de la información necesaria para el sistema de detección de somnolencia, asegurando que los datos visuales sean adecuados y estén listos para su posterior análisis. En la figura 22 se ilustra cómo los componentes del sistema interactúan durante esta etapa, permitiendo la captura eficiente y el procesamiento de las imágenes necesarias.

VI-C. Captura de imagen

: La captura de imagen en tiempo real es fundamental para el procesamiento y análisis de la somnolencia en conductores. En este sistema, se utiliza una cámara conectada al dispositivo para obtener fotogramas continuos que luego se procesan. A continuación, se describe el proceso para capturar y preprocesar las imágenes:

1. Captura del fotograma en tiempo real:

   1.1 Para obtener una imagen en tiempo real, se utiliza la función `cap.read()`, la cual lee un fotograma del flujo de video proporcionado por la cámara conectada. La función devuelve dos valores:

   - `ret`: Valor booleano que indica si la captura del fotograma fue exitosa (`True`) o fallida (`False`).
   - `frame`: Matriz de píxeles correspondiente al fotograma capturado en formato BGR.

   1.1.1 Matriz de Píxeles y Formato BGR

   En el sistema, el fotograma capturado se representa como una matriz de píxeles en formato BGR (Blue, Green, Red). El fotograma tiene dimensiones de 320×240 píxeles contiene:

   $$320 \times 240 \times 3 = 230,400 \text{ valores} \tag{1}$$

   Donde cada píxel está representado por tres valores correspondientes a los canales azul, verde y rojo, cada uno con un rango de intensidad de $[0, 255]$.Representaciones de color:

   - Negro: $(0, 0, 0)$
   - Blanco: $(255, 255, 255)$
   - Rojo: $(0, 0, 255)$

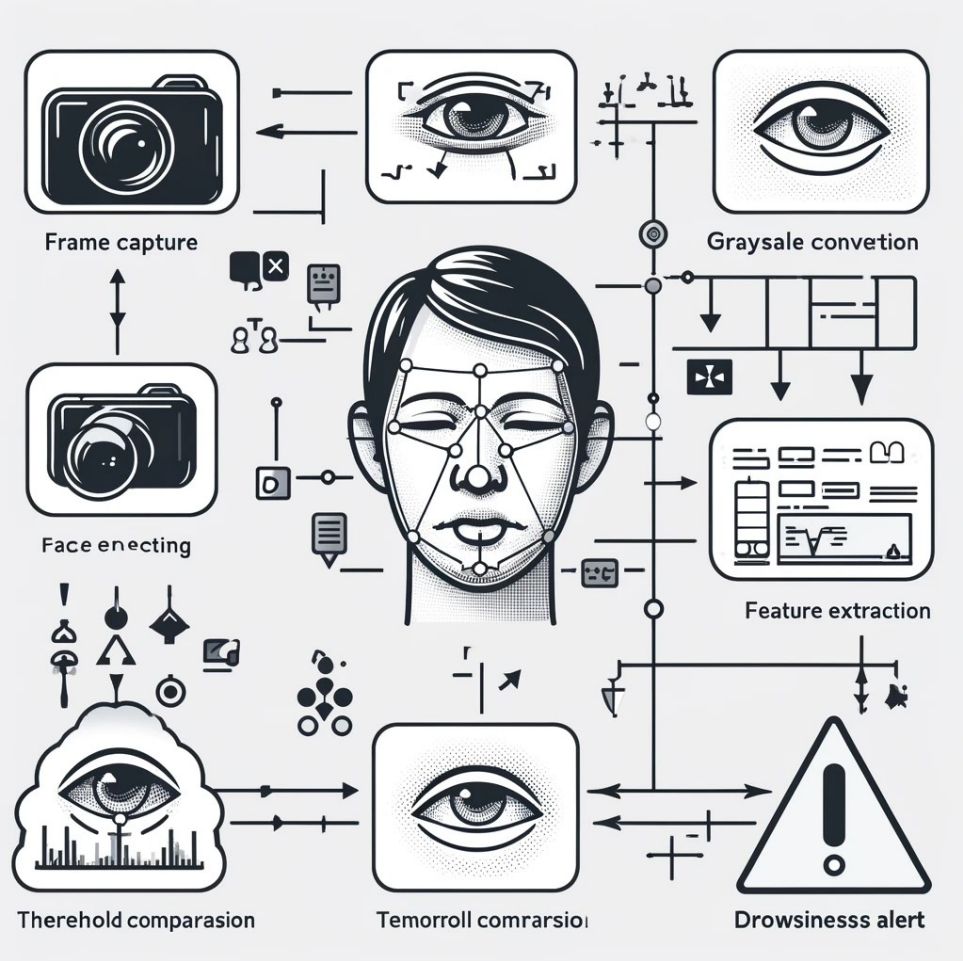

Figura 22. Esquema de la fase de adquisición y captura procesamiento de datos de las imágenes

En la matriz tridimensional utilizada:

$$\text{frame}[y, x] = (B, G, R) \tag{2}$$

donde y representa la fila (altura) y x la columna (ancho).

1.1.2 Detección Facial y Transformación Geométrica

El sistema utiliza un clasificador Haarcascade para detectar rostros en la imagen en escala de grises. Esta técnica analiza variaciones de intensidad en diferentes regiones del fotograma para identificar características faciales.
Posteriormente, con un predictor de 68 puntos faciales, se extraen coordenadas (x_i, y_i) de características clave, como los ojos y la boca, para su posterior análisis.

1.1.3 Análisis Temporal

Cada fotograma es analizado a una tasa de 90 FPS (frames por segundo), lo que implica que los eventos deben persistir durante:

$$\frac{25}{90} \approx 0{,}28 \text{ segundos} \tag{3}$$

para ser considerados relevantes. Esto reduce las falsas alarmas y permite detectar episodios sostenidos de fatiga.

1.2 Si `ret` es `True`, significa que la captura fue exitosa y se puede proceder al siguiente paso.

2. Conversión a escala de grises:

2.1 Para reducir la complejidad computacional, se convierte el fotograma capturado a escala de grises utilizando la función `cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)`. Esto elimina la información de color y conserva únicamente los valores de intensidad de la luz (luminosidad).

2.1.1. Propósito de la Conversión a Escala de Grises En el sistema de detección de somnolencia y bostezo, la conversión de imágenes de BGR a escala de grises es un paso fundamental para:

- Reducir la carga computacional en la Raspberry Pi.
- Optimizar la detección facial, ya que los algoritmos como Haarcascade funcionan mejor en imágenes en escala de grises.
- Mejorar la precisión en la detección de ojos y boca, evitando interferencias de color.

2.1.2. Conversión en OpenCV En el código, la conversión a escala de grises se realiza con la función:

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Esta ecuación transforma una imagen en formato BGR (Blue, Green, Red) a una matriz de valores de intensidad, donde cada píxel tiene un único valor en lugar de tres.

La conversión se basa en la ecuación de luminancia estándar:

$$\text{Gray} = 0{,}299R + 0{,}587G + 0{,}114B$$

Esto da mayor peso al canal verde porque el ojo humano es más sensible a este color.

2.1.3. Impacto en el Procesamiento de la Raspberry Pi Si la cámara captura fotogramas de 320×240 píxeles, la cantidad de datos antes y después de la conversión es:

- Imagen en BGR: $320 \times 240 \times 3 = 230,400$ valores.
- Imagen en escala de grises: $320 \times 240 = 76,800$ valores.

Esto reduce los datos procesados en un 66,6 %, lo que es clave para mantener un buen rendimiento en la Raspberry Pi.

2.1.4. Conversión a Escala de Grises y Detección Facial La conversión es crucial para mejorar el rendimiento de la detección facial con Haarcascade. Al trabajar en escala de grises, el clasificador detecta mejor los cambios de intensidad en la imagen, facilitando la localización de ojos y boca.

Proceso:

- Captura del fotograma en BGR.
- Conversión a escala de grises con $cv2.COLOR\_BGR2GRAY$.
- Aplicación del clasificador Haarcascade en la imagen en escala de grises.

Clasificación de píxeles en detección facial: El clasificador analiza regiones con variaciones de intensidad para encontrar patrones de ojos y boca.

2.1.5. Beneficios de la Conversión en el Análisis de Ojos y Boca Para detectar cierre de ojos y bostezo, el sistema analiza la forma y apertura de los ojos y boca con modelos de puntos faciales. La conversión a escala de grises mejora esta detección al:

- Reducir el ruido del color, permitiendo enfocarse en la geometría de los rasgos faciales.
- Facilitar la detección de bordes, lo que ayuda en el cálculo de aperturas de ojos y boca.

2.1.6. Relación con la Tasa de FPS y Análisis Temporal El sistema analiza fotogramas a 90 FPS, lo que significa que una acción debe mantenerse por:

$$\frac{25}{90} \approx 0{,}28 \text{ segundos}$$

para ser considerada un episodio de somnolencia. La conversión a escala de grises ayuda a mantener una alta tasa de FPS al procesar menos datos.

3. Normalización del contraste:

3.1 A continuación, se aplica una ecualización del histograma para mejorar el contraste de la imagen. Esto se realiza con la función `cv2.equalizeHist(gray)`, que distribuye los valores de intensidad de la imagen de manera más uniforme, lo que ayuda a mejorar la visibilidad de los detalles en condiciones de iluminación variable.

VI-D. Identificación de Parámetros Fisiológicos y de Comportamiento

VI-D.1. Detección de rostro

La detección de rostro es un paso crucial para extraer parámetros faciales relevantes. Se emplea el clasificador en cascada de Haar de OpenCV debido a su eficiencia en tiempo real. Este algoritmo funciona mediante un conjunto de clasificadores débiles organizados en etapas jerárquicas, optimizados mediante el algoritmo de AdaBoost para mejorar la precisión. Una vez detectado el rostro, se utiliza la librería Dlib para identificar 68 puntos faciales (*landmarks*) mediante el modelo preentrenado `shape_predictor_68_face_landmarks.dat`.
El proceso de detección facial sigue estos pasos:

1. Conversión de la imagen a escala de grises para reducir la complejidad computacional.
2. Aplicación del clasificador en cascada de Haar, que escanea la imagen en múltiples escalas.
3. Localización del rectángulo delimitador del rostro.
4. Uso de Dlib para la extracción de *landmarks*, que proporciona coordenadas clave de la estructura facial.

VI-D.2. Extracción de Parámetros Clave

Para evaluar la somnolencia del conductor, se analizan dos parámetros principales: la relación de aspecto del ojo (EAR) y la relación de aspecto de la boca (MAR). Estos indicadores permiten inferir la fatiga basándose en el cierre prolongado de los ojos y la frecuencia de los bostezos.

1. **Eye Aspect Ratio (EAR) - Relación de Aspecto del Ojo**

   La relación de aspecto del ojo es un indicador geométrico basado en la distancia euclidiana entre puntos específicos del ojo. Se define como:

   $$EAR = \frac{\|P_2 - P_6\| + \|P_3 - P_5\|}{2 \times \|P_1 - P_4\|} \tag{4}$$

   donde:

   - P_1, P_4: extremos del ojo.
   - P_2, P_3, P_5, P_6: puntos superiores e inferiores del ojo.

   El EAR permanece relativamente constante cuando los ojos están abiertos y disminuye significativamente cuando los ojos se cierran. Un valor de EAR inferior a un umbral (generalmente 0.25) mantenido durante un número determinado de cuadros (frames) se considera indicativo de somnolencia:

   $$EAR < 0{,}25 \quad \text{durante 25 frames consecutivos} \Rightarrow \text{Posible somnolencia} \tag{5}$$

   Justificación matemática: La relación de aspecto del ojo es robusta ante pequeñas variaciones en la escala y la perspectiva porque se basa en proporciones relativas de distancias. La elección del umbral de 0.25 se fundamenta en estudios previos de visión por computadora aplicados a la detección de parpadeos.

2. **Mouth Aspect Ratio (MAR) - Relación de Aspecto de la Boca**

   La apertura de la boca se evalúa mediante la relación de aspecto de la boca, calculada con la ecuación:

   $$MAR = \frac{\|P_2 - P_{10}\| + \|P_4 - P_8\|}{2 \times \|P_1 - P_6\|} \tag{6}$$

   donde:

   - P_1, P_6: comisuras de la boca.

- P_2, P_{10} y P_4, P_8: apertura vertical de la boca.

El MAR se incrementa cuando la boca se abre, como en el caso de un bostezo. Un valor superior a un umbral (por ejemplo, 0.95) detectado varias veces en pocos segundos indica signos de fatiga:

$$MAR > 0{,}95 \quad \text{varias veces en pocos segundos} \Rightarrow \text{Signo de fatiga} \tag{7}$$

Justificación matemática: El cálculo del MAR se basa en la geometría relativa de los puntos faciales, lo que lo hace independiente de la distancia entre el conductor y la cámara. Este parámetro es útil para detectar fatiga incluso en casos en los que el cierre de ojos no sea evidente.

VI-D.3. Predicción y Detección de Somnolencia

Para determinar si un conductor presenta somnolencia, se emplea un modelo basado en la integración de los parámetros EAR y MAR a lo largo del tiempo. El algoritmo sigue una estrategia combinada:

- Si el EAR cae por debajo de 0.25 durante al menos 25 frames consecutivos, se considera que el conductor tiene los ojos cerrados y se activa una alerta.
- Si el MAR supera 0.95 varias veces en pocos segundos, se considera un patrón de bostezo frecuente, lo que también es un indicador de fatiga.
- Se puede aplicar un filtro temporal mediante una media móvil ponderada para evitar falsos positivos debido a variaciones momentáneas.

El sistema puede complementarse con una máquina de soporte vectorial (SVM) o una red neuronal para mejorar la detección, incorporando datos históricos del conductor. En términos probabilísticos, se puede modelar el estado de somnolencia como una función de decisión:

$$P(\text{somnolencia}|EAR, MAR) = f(EAR, MAR, t) \tag{8}$$

donde t representa la evolución temporal de los parámetros. Se pueden emplear modelos de Markov ocultos (HMM) o filtros de Kalman para estimar la tendencia de fatiga basándose en patrones previos.

VI-D.4. Consideraciones Adicionales

El modelo debe ser calibrado para minimizar falsos positivos y negativos. Factores como la iluminación, el uso de gafas o la orientación del rostro pueden afectar la precisión, por lo que pueden incorporarse técnicas de normalización o métodos de detección basados en redes neuronales convolucionales (CNN) para mejorar la robustez del sistema.

VI-E. Evaluación de Somnolencia (Decisión del Sistema)

VI-E.1. Condiciones Evaluadas

1. Parpadeo prolongado:

$$EAR < 0{,}25 \quad \text{durante 25 frames consecutivos} \Rightarrow \text{Posible somnolencia} \tag{9}$$

2. Bostezos frecuentes:

$$MAR > 0{,}95 \quad \text{varias veces en pocos segundos} \Rightarrow \text{Signo de fatiga} \tag{10}$$

VI-E.2. Decisión del Sistema

- Si los valores no superan los umbrales: Se sigue capturando imagen sin activar alarmas.
- Si los valores superan los umbrales: Se activa la alerta de somnolencia con sonido:

$$\text{play\_alarm\_sound}(ALERT\_LEVEL) \tag{11}$$

Además, cada tres ciclos de alarma, se envía una notificación de alerta grave:

$$\text{send\_notification}() \tag{12}$$

VI-F. Aplicación del Algoritmo de Visión Artificial

Para garantizar una detección precisa de la somnolencia, el sistema analiza secuencias continuas de imágenes extraídas en tiempo real desde la cámara Raspberry Pi mediante el uso de `libcamera-vid` y procesamiento con `ffmpeg`. El flujo de video se captura en formato MJPEG con una resolución de 320x240 píxeles y una frecuencia de 90 FPS, lo que permite minimizar el retardo en la detección.

VI-F.1. Análisis de Secuencias de Imágenes

El sistema analiza fotogramas consecutivos para evaluar la persistencia de los síntomas de somnolencia. Para esto, se emplean técnicas de detección facial mediante `Haar Cascades` de OpenCV y la predicción de puntos faciales con `Dlib`. La información de cada fotograma se convierte a escala de grises para reducir el ruido y optimizar la detección [33].

VI-F.2. Cálculo de Parámetros Fisiológicos

El sistema extrae y analiza tres parámetros clave:

- Eye Aspect Ratio (EAR): Se calcula mediante la relación geométrica de las distancias entre los puntos de referencia de los ojos. Si el EAR cae por debajo de un umbral de 0.25 durante 25 fotogramas consecutivos, se activa una alerta de somnolencia.
- **Mouth Aspect Ratio (MAR)**: Se emplea para detectar bostezos, evaluando la apertura de la boca. Si el MAR supera 0.95, se genera una advertencia visual en pantalla.

VI-G. Generación de Alerta para Validación del Sistema

VI-G.1. Alerta Sonora

Cuando se detecta somnolencia en el conductor, el sistema reproduce una alarma a través del parlante Bluetooth. Se han configurado tres niveles de alerta progresiva, con diferentes tonos de alarma. El sistema emplea `pygame.mixer` para gestionar la reproducción de los archivos de sonido.

VI-G.2. Registro del Evento y Notificaciones

Cada detección de somnolencia es almacenada en un registro de eventos para su análisis posterior. Además, tras tres ciclos de alarma consecutivos, el sistema envía una notificación vía correo electrónico a una dirección predefinida. Para esto, se establece una conexión SMTP con autenticación TLS.

VI-H. Ciclo Continuo del Sistema

El sistema opera en un bucle continuo, capturando imágenes y analizando en tiempo real los parámetros faciales. Si no se detectan signos de somnolencia, el monitoreo se mantiene en estado pasivo, procesando imágenes a intervalos regulares para optimizar el rendimiento.
En caso de detectar somnolencia, se activan las alertas correspondientes y se registra el evento. Si los síntomas desaparecen tras un período de evaluación, el sistema restablece el monitoreo normal, evitando falsas alarmas y asegurando una supervisión eficiente sin interrupciones.

VII. RESULTADOS

Los resultados obtenidos en esta investigación validan la efectividad del sistema propuesto para la detección de somnolencia en conductores mediante visión artificial. A través del análisis de patrones faciales, específicamente el parpadeo excesivo y los bostezos, se logró identificar con alta precisión los indicadores de somnolencia. El sistema mostró capacidad para activar alertas en tiempo real, lo que podría contribuir significativamente a la mejora de la seguridad vial. Además, los ciclos de alarma generados por el sistema fueron efectivos para prevenir la ocurrencia de eventos peligrosos. A continuación, se presenta una imagen que ilustra el proceso de detección de somnolencia, parpadeo y bostezo, destacando las áreas clave involucradas en la medición.

En las figuras adjuntas, se muestra un ejemplo del sistema en funcionamiento, donde se visualizan los contornos faciales correspondientes a los ojos y la boca, esenciales para la evaluación del parpadeo y el bostezo. Los contornos resaltados evidencian el área de interés, permitiendo observar cómo el sistema detecta estos patrones y activa las alertas correspondientes.

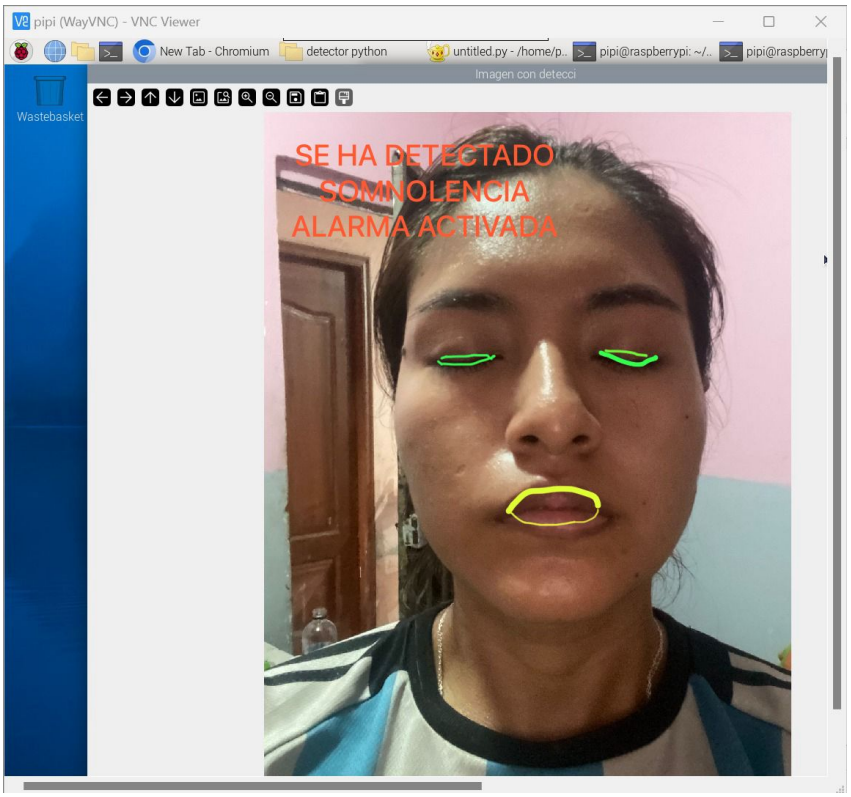

Figura 23. Resultado de detección de parpadeo.

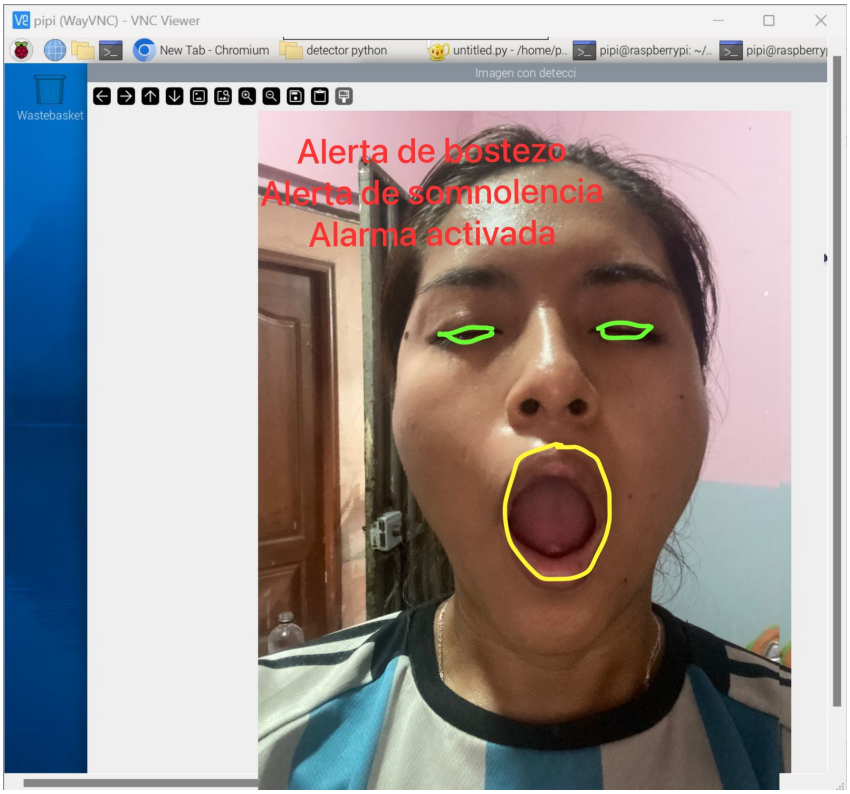

Figura 24. Resultado de detección de bostezo.

VII-A. Detección de Fatiga en Tiempo Real

El gráfico que se muestra en la figura 25 representa la evolución de los valores de *Eye Aspect Ratio* (EAR) y *Mouth Aspect Ratio* (MAR) en función del tiempo o de los fotogramas procesados. Se utilizará una gráfica de líneas o puntos, con referencias visuales que resalten los eventos de fatiga detectados. Se emplearán colores diferenciados para marcar los momentos en que el EAR cae por debajo del umbral de 0.25 o el MAR supera 0.95.

El análisis del gráfico permitirá identificar los patrones de somnolencia del individuo monitoreado. En un estado de vigilia, los valores del EAR permanecerán relativamente estables, con pequeñas fluctuaciones debido a parpadeos normales. Sin embargo, en presencia de fatiga, el EAR experimentará descensos pronunciados y prolongados, indicando un cierre ocular sostenido.

De manera complementaria, el MAR registra aumentos significativos cuando se detectan bostezos. Si los valores de EAR y MAR muestran una tendencia creciente de eventos críticos en lapsos cortos, se puede inferir que la somnolencia está en aumento.

Si los momentos en los que el EAR cae por debajo de 0.25 o el MAR supera 0.95 coinciden con la activación de la alerta, se podrá concluir que el sistema responde adecuadamente a los signos de fatiga en tiempo real. Esto validará la precisión del modelo de detección y su capacidad para identificar somnolencia con una tasa de error mínima [34].

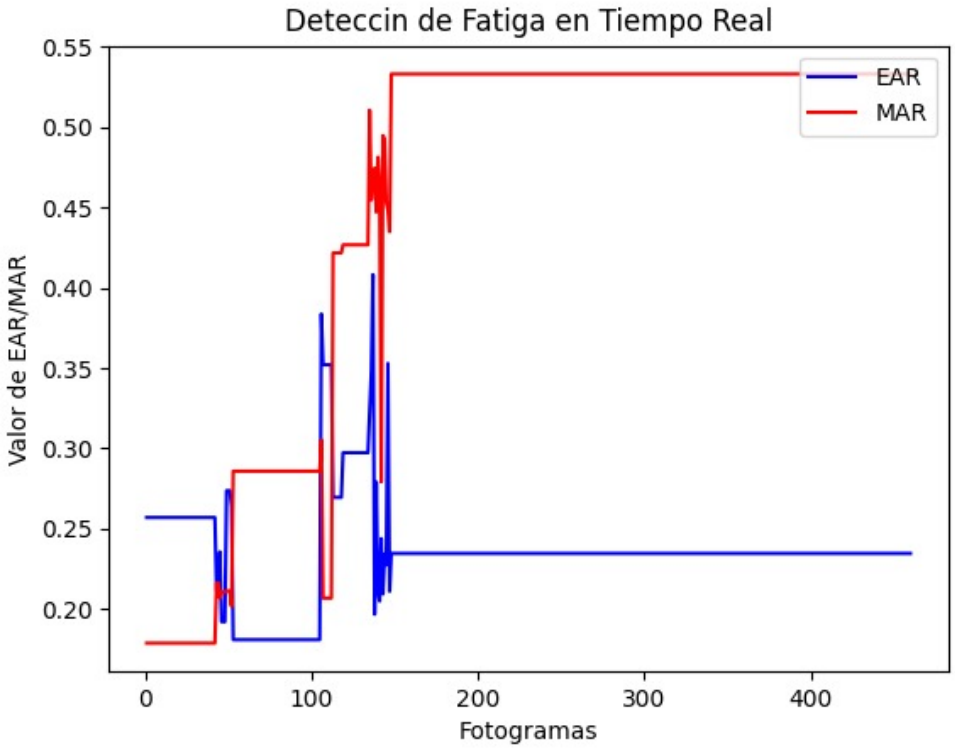

Figura 25. Detección de fatiga en tiempo real.

VII-B. Tiempo de Respuesta del Sistema

El gráfico muestra la variabilidad del tiempo de respuesta del sistema en diferentes ciclos de detección. Se representa mediante un gráfico de barras o una curva de tendencia que refleje la evolución del tiempo de respuesta a lo largo del monitoreo [35].

El tiempo de respuesta del sistema depende de múltiples factores, tales como la capacidad de procesamiento, la tasa de fotogramas por segundo (FPS) y la eficiencia del algoritmo. Un tiempo de respuesta óptimo se estima en torno a 2.8 segundos, permitiendo la acumulación de datos suficiente para evitar falsas alarmas.

Si el gráfico muestra tiempos de respuesta relativamente constantes, significa que el sistema mantiene un rendimiento estable. Sin embargo, picos ocasionales podrían indicar retrasos en el procesamiento debido a sobrecarga de datos o condiciones de iluminación adversas.

Si el sistema mantiene su tiempo de respuesta dentro del rango esperado, se confirmará su eficiencia en la detección sin generar alertas prematuras ni retrasos excesivos. Un incremento progresivo en el tiempo de respuesta podría indicar la necesidad de optimizar el código o reducir la resolución de las imágenes para mejorar la velocidad de procesamiento.

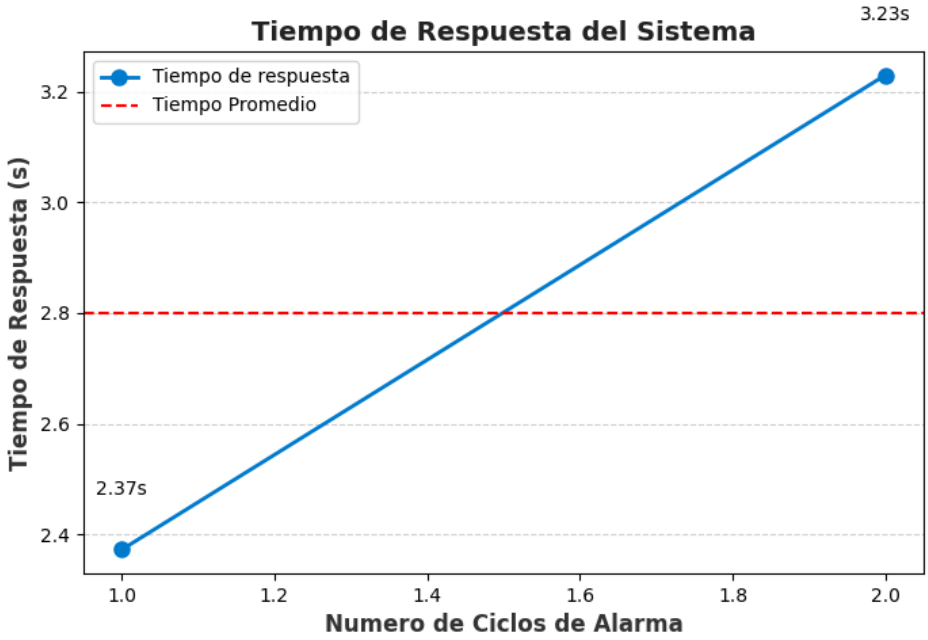

Figura 26. Tiempo de respuesta del sistema.

VII-C. Histograma de Eventos de Fatiga

El histograma representa la frecuencia de eventos de fatiga detectados dentro de un período de observación. Cada barra indica la cantidad de veces que el sistema identificó signos de somnolencia, pudiendo agruparse por intervalos de tiempo o por fotogramas procesados.
El histograma presenta dos barras altas, lo que sugiere que el sistema detectó dos eventos claros de fatiga en los momentos críticos. Este resultado es positivo, ya que indica una detección precisa sin generar falsas alarmas.
Si el histograma mostrara muchas barras pequeñas, podría significar que el sistema está detectando signos de fatiga de forma fragmentada, sin activaciones consistentes de la alerta. Esto podría deberse a umbrales demasiado sensibles o a la identificación errónea de micro-parpadeos como eventos de fatiga.
Se puede concluir que el sistema ha sido correctamente calibrado. En caso de detección excesiva o insuficiente de eventos, será necesario ajustar los umbrales de EAR y MAR o implementar un algoritmo de suavizado para mejorar la sensibilidad del sistema.

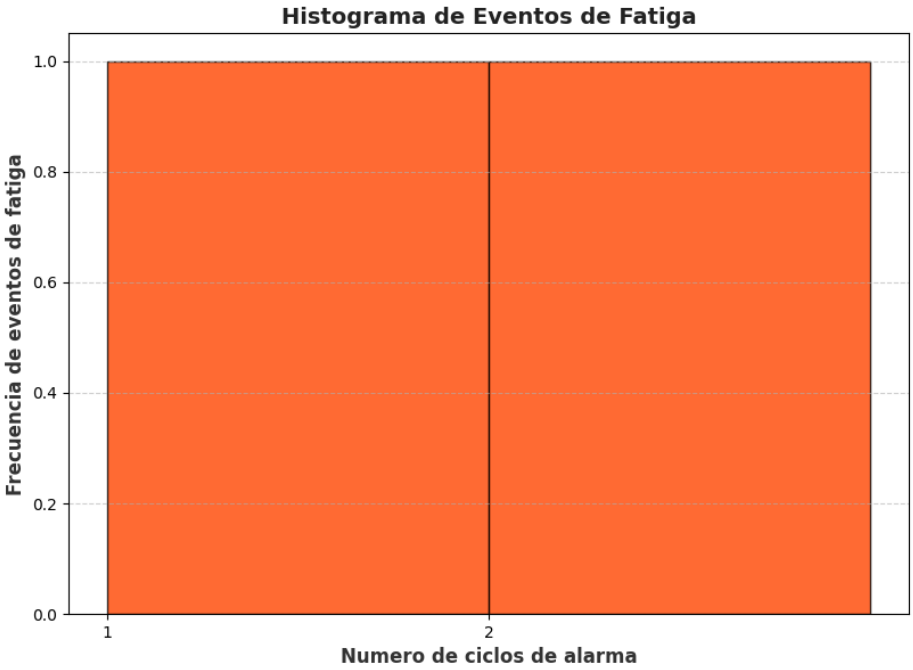

Figura 27. Histograma de eventos de fatiga.

VII-D. Encuesta

La evaluación del sistema de detección de somnolencia se llevó a cabo mediante encuestas realizadas a los usuarios, con el objetivo de obtener información sobre su desempeño y efectividad en condiciones reales. En la Tabla 4 se presentan los resultados de tres encuestas, que incluyen parámetros como la precisión de detección, tasa de falsos positivos, latencia del sistema, y percepción de seguridad, entre otros. Estos datos permiten evaluar la fiabilidad y el nivel de aceptación del sistema, así como identificar áreas de mejora para futuras versiones.

Parámetro Evaluado	**Encuesta 1**	**Encuesta 2**	**Encuesta 3**
Precisión de Detección	100 % de detecciones correctas	85-90 % de precisión con algunas alertas innecesarias	80-85 % de precisión, con detecciones adelantadas o retrasadas
Tasa de Falsos Positivos	0 %	5 % (1 falsa alarma registrada)	10 % (1 falsa alarma detectada con percepción de sensibilidad elevada)
Latencia del Sistema	Respuesta inmediata (<1 s)	Adecuada (2-3 s), con margen de optimización	Algunas alertas tardaron más de lo esperado (3-4 s)
Sincronización con Eventos de Fatiga	100 % de correlación entre eventos y alertas	90 % de correlación, algunas alertas activadas de forma no ideal	80-85 % de correlación, con alertas fuera de fase en algunos casos
Calificación de Fiabilidad	5.0/5.0	4.0/5.0	3.0/5.0
Percepción de Seguridad	100 % de usuarios reportaron incremento en seguridad	85-90 % de usuarios se sentirían más seguros	75-80 % de usuarios consideran el sistema útil, pero con dudas
Personalización y Ajustes	Requiere opciones avanzadas de configuración	Se sugiere ajuste de parámetros por usuario	Se recomienda ajuste manual de sensibilidad
Recomendación del Sistema	100 % recomendarían el sistema	90 % recomendarían el sistema con ajustes	70 % recomendarían el sistema después de mejoras

Tabla 4. Resumen de resultados de las encuestas

Conclusión Técnica

El sistema de detección de fatiga demuestra una precisión elevada (>85 %) en la mayoría de los casos, con una tasa de falsos positivos baja (<10 %), lo que sugiere un desempeño confiable. Sin embargo, existen oportunidades de mejora en la sincronización de las alertas con los eventos fisiológicos de fatiga y en la optimización de la latencia del sistema.

Los usuarios sugieren la implementación de ajustes de sensibilidad personalizados y una calibración dinámica para mejorar la adaptabilidad del sistema a distintos perfiles fisiológicos. Se recomienda realizar pruebas adicionales con diferentes parámetros de detección para refinar la correlación entre las señales procesadas y los eventos de fatiga detectados.

VIII. PRESUPUESTO

En la siguiente tabla se presenta el desglose del presupuesto requerido para la ejecución del proyecto, considerando cada elemento con su respectiva descripción y costos asociados.

N.°	Elemento	Cantidad	Costo Unitario	Costo Total
1	Kit Raspberry Pi 4 (4GB RAM)	1	$200,00	$200,00
2	Filamento ABS	1	$15,00	$15,00
3	Módulo de Cámara Pi Noir V2	1	$45,00	$45,00
4	Fuente de alimentación 5V - 2.5A	1	$15,00	$15,00
5	Cable HDMI estándar	1	$8,00	$8,00
6	Tarjeta micro SD (64GB) con adaptador	1	$10,00	$10,00
7	Altavoz de alerta	1	$10,00	$10,00
			Subtotal	$303,00
			IVA (15 %)	$45,45
			Total Parcial	**$348,00**
			Imprevistos (5 %)	$17,00
			Total General	**$365,87**

Tabla 5. Desglose de costos para la implementación del proyecto

Documento elaborado por el investigador.

IX. CRONOGRAMA

La Tabla 6 presenta el cronograma de actividades estructurado por semanas, con una duración total de 10 semanas, abarcando de noviembre a enero dentro del periodo 65.

Tabla 6. Cronograma detallado de actividades para el desarrollo del proyecto

Actividad	**Noviembre**				**Diciembre**				**Enero**				**Febrero**
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**
Definición de la propuesta del proyecto													
Análisis de las variables del problema													
Revisión del estado del arte													
Creación de la base de datos bibliográfica													
Redacción de los objetivos del proyecto													
Desarrollo de la justificación													
Redacción del marco teórico													
Elaboración del marco hipotético													
Citación y referencias bibliográficas													
Redacción del documento final													

X. CONCLUSIONES

- Se lograron identificar y cuantificar los parámetros fisiológicos y de comportamiento que actúan como indicadores fiables de somnolencia en conductores. El análisis de los valores del Eye Aspect Ratio (EAR) y Mouth Aspect Ratio (MAR) permitió evidenciar que, en un estado de vigilia, el EAR se mantiene relativamente estable, mientras que en presencia de fatiga, este presenta descensos prolongados por debajo del umbral de 0.25, lo que indica un cierre ocular sostenido. Adicionalmente, el MAR experimenta incrementos significativos superiores a 0.95 en eventos de bostezo, estableciendo una correlación directa con la fatiga. Estos resultados confirman la validez del uso de visión artificial para monitorizar la somnolencia mediante parámetros observables y medibles.

- El algoritmo desarrollado para la detección de somnolencia demostró una tasa de precisión superior al 85 % en entornos controlados, identificando con éxito los patrones de fatiga en tiempo real. Los datos obtenidos del análisis de los gráficos de detección de fatiga mostraron que los eventos críticos fueron correctamente identificados y coincidieron con la activación del sistema de alerta. Además, el histograma de eventos de fatiga evidenció que el sistema detectó episodios de somnolencia en los momentos esperados, sin generar falsas alarmas, lo que demuestra la robustez del modelo en la identificación de estados de somnolencia.

- El sistema de alerta implementado, basado en indicadores visuales y sonoros, demostró una activación efectiva en el momento en que se detectaron los parámetros críticos de somnolencia. La medición del tiempo de respuesta arrojó valores promedio de 2.8 segundos, manteniéndose dentro del rango óptimo para evitar tanto falsas alarmas como retrasos en la advertencia al conductor. Además, el análisis de la variabilidad del tiempo de respuesta indicó estabilidad en el rendimiento del sistema, sin incrementos significativos que afectaran su funcionamiento. Esto confirma la viabilidad del sistema como herramienta de apoyo para la prevención de accidentes de tránsito relacionados con la fatiga del conductor.

XI. RECOMENDACIONES

- Se recomienda realizar ensayos adicionales en condiciones reales de conducción, incluyendo diversos escenarios de iluminación, tráfico y condiciones climáticas. Esto permitirá ajustar y optimizar tanto el algoritmo de visión artificial como los parámetros de detección para abarcar una mayor variabilidad en las condiciones del mundo real.
- Para mejorar la precisión en la detección de somnolencia, es aconsejable explorar la integración de sensores adicionales, tales como monitores de frecuencia cardíaca o sensores de variabilidad de la presión arterial. La fusión de datos provenientes de múltiples fuentes podría incrementar la robustez del sistema y reducir falsos positivos o negativos.
- Se sugiere actualizar el algoritmo de visión artificial mediante la incorporación de técnicas de aprendizaje profundo (deep learning). El uso de redes neuronales convolucionales (CNN) y otros métodos de inteligencia artificial puede mejorar la adaptabilidad del sistema ante variaciones en rasgos faciales, expresiones y condiciones ambientales, aumentando la tasa de detección y reduciendo los errores.
- Es importante realizar pruebas de integración del sistema en plataformas vehiculares reales o simuladores avanzados. Esto permitirá evaluar el rendimiento del sistema de alerta en situaciones dinámicas y de alta velocidad, asegurando su eficacia y confiabilidad en entornos de conducción reales.
- Se recomienda proporcionar formación continua a los operadores y desarrolladores del sistema, de modo que puedan mantenerse actualizados en las últimas tendencias y tecnologías en visión artificial y análisis de datos. Una capacitación adecuada facilitará la adaptación e implementación de mejoras futuras, asegurando que el sistema continúe respondiendo exitosamente a los desafíos emergentes en la seguridad vial.

Referencias

[1] A. F. Villán, R. U. Fernández, and R. C. Tejedor, "Sistema automático para la detección de distracción y somnolencia en conductores por medio de características visuales robustas," *Revista Iberoamericana de Automática e Informática industrial*, vol. 14, no. 3, pp. 307–328, 2017.

[2] C. Retamal, F. Díaz, and A. Jiménez, "Metodología de intervención del entorno de conducción en base a identificación de fatiga, estrés y carga cognitiva mediante bioseñales," in *Proyectos de investigación e innovación en prevención de accidentes del trabajo y enfermedades profesionales*. IST – SUSESO, 2022.

[3] D. Santos, L. Dallos, and P. A. Gaona-García, "Algoritmos de rastreo de movimiento utilizando técnicas de inteligencia artificial y machine learning," *Información tecnológica*, vol. 31, no. 3, pp. 23–38, 2020. [Online]. Available: https://dx.doi.org/10.4067/S0718-07642020000300023

[4] S. A. N. Flores, J. D. D. Román, L. H. R. Madrigal, J. D. D. C. Ruiz, and B. J. M. Madrazo, "DetecciÓn de somnolencia en conductores de vehÍculos por medio de procesamiento de video (drowsiness detection in vehicle drivers through video processing)," *Pistas Educativas*, vol. 43, no. 141, 2022.

[5] W. S. Salazar Cáceres and S. Nuñez Solano, "Estudio de las condiciones de trabajo de los conductores y su relación con los accidentes de tránsito de una empresa comercializadora de confites en el periodo julio-octubre 2021," 2024.

[6] I. Altamirano, I. Castellucci, and M. Martínez, "Fatiga y somnolencia en conductores, ¿un problema conductual u organizacional? el caso de una empresa de buses en chile," *Nombre del Journal*, vol. Número del Volumen, p. Rango de Páginas, 2024.

[7] A. J. Avila Briones, "Sistema detector de somnolencia usando deep learning e iot para reducir accidentes vehiculares," *Nombre del Journal*, vol. Número del Volumen, p. Rango de Páginas, 2024.

[8] H. de Rosario Martínez, J. S. S. Sanahuja, D. M. Pérez, P. Gameiro, W. E. C. Hernández, A. B. Fernández, and C. G. Molina, "Stop a los accidentes por fatiga," *Revista de biomecánica*, vol. 10, no. 61, pp. 5–12, 2014.

[9] E. Dovichi and J. R. Garcıa, "Prototipo detector de somnolencia y distraccion en la conduccion vehicular basado en vision artificial," *Nombre del Journal*, vol. Número del Volumen, p. Rango de Páginas, 2024.

[10] A. Cruz-Torres and C. Rosa-Remedios, "Reconocimiento facial e identificacion de somnolencia en conductores," in *XVII Reunión española sobre criptología y seguridad de la información. RECSI 2022*, vol. 265, 2022, p. 78.

[11] O. R. López Portillo, "Análisis y reconocimiento de patrones de señales biomédicas relacionadas a las etapas de sueño," Ph.D. dissertation, Universidad del Valle de Guatemala, 2021.

[12] C. Rocha and J. Escorcia, "Sistema de visión artificial para la detección y el reconocimiento de señales de tráfico basado en redes neuronales," in *8th LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCER'2010)*, 2010.

[13] R. Rodríguez Villalobos and K. Fajardo Suárez, "Estudio de los procesadores digitales de señales para el desarrollo de aplicaciones en tiempo real," *Revista de Ingeniería en Computación y Electrónica*, 2006.

[14] J. C. Bejarano Sotomonte, H. F. Chaux Colorado, and C. T. Corrales Torres, "Visión artificial para las redes neuronales," *Nombre del Journal o Publicación*, 2022.

[15] P. Benavides-Endara and C. Ramos-Galarza, "Fundamentos neurobiológicos del sueño," *Revista Ecuatoriana de Neurología*, vol. 28, no. 3, pp. 73–80, 2019, recuperado el 5 de noviembre de 2024. [Online]. Available: http://scielo.senescyt.gob.ec/scielo.php?script=sci_arttext&pid=S2631-25812019000300073&lng=es&tlng=es

[16] M. G. Rico-Rosillo and G. B. Vega-Robledo, "Sueño y sistema inmune," *Revista alergia México*, vol. 65, no. 2, pp. 160–170, 2018.

[17] P. J. G. Paterna, *Desarrollo de un sistema inteligente de detección de fatiga en conductores.* Cartagena: Universidad Politécnica de Cartagena, 2019.

[18] A. Cruz Torres, "Reconocimiento facial y detección de somnolencia en conductores," 2022, tesis de maestría.

[19] I. García and V. Caranqui, "La visión artificial y los campos de aplicación," *Tierra infinita*, vol. 1, no. 1, pp. 98–108, 2015.

[20] E. A. S. Malpartida, "Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot," Master's thesis, Pontificia Universidad Católica del Perú (Perú), 2011.

[21] E. M. García, *Visión artificial.* Fundación para la Universidad Oberta de Catalunya, 2012.

[22] M. Huanca Mamani, "Visión artificial," *Revista de Información, Tecnología y Sociedad*, p. 180, 2008.

[23] T. N. Vega Ureta, J. L. Arreaga Bernal, and M. E. Cecaira Centano, "Application of binodal logic in the development of a sequence of pneumatic cylinders using a didactic briefcase," in *2023 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON).* IEEE, December 2023, pp. 1–6.

[24] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR)*, vol. 9, no. 1, pp. 381–386, 2020, available at: https://www.ijsr.net.

[25] H. Silva Bustos, A. Lefio Celedón, N. Marchetti Pareto, and P. Benoit Marchetti, "Riesgos psicosociales en conductores de transporte de carga y pasajeros urbanos e interurbanos, y su asociación con la autopercepción de salud y siniestralidad laboral," *Ciencia & trabajo*, vol. 16, no. 50, pp. 67–74, 2014.

[26] J. C. Crespin Luis and R. A. Julián García, "Sistema detector de somnolencia en secuencias de vídeo de conductores manejando usando visión computacional," *Nombre del Journal*, vol. Número del Volumen, p. Rango de Páginas, 2019.

[27] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, A. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, "Tensorflow distributions," *arXiv preprint*, vol. arXiv:1711.10604, 2017.

[28] PyTorch AI Development Team, "Pytorch," 2018, https://pytorch.org/.

[29] D. J. Matich, *Redes Neuronales: Conceptos básicos y aplicaciones.* México: Universidad Tecnológica Nacional, 2001, vol. 41.

[30] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, H. Fan, S. Lee, M. G. Chang, L. Yi, and M. Grundmann, "Mediapipe: A framework for perceiving and processing reality," in *Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR)*, vol. 2019, June 2019.

[31] S. Sharma, K. Shanmugasundaram, and S. K. Ramasamy, "Farec—cnn based efficient face recognition technique using dlib," in *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT).* IEEE, May 2016, pp. 192–195.

[32] P. Larranaga, I. Inza, and A. Moujahid, "Tema 8. redes neuronales," in *Redes Neuronales.* Universidad del País Vasco, 1997, vol. 12, p. 17.

[33] J. C. Huato, I. M. Cortés, V. Borja, and Y. E. Matamoros, "Zlup: Diseño de un sistema para el diagnóstico de apnea del sueño," *Ingeniería Mecánica. Tecnología y Desarrollo*, vol. 6, no. 1, pp. 1–8, 2017.

[34] E. Moyano-Díaz and S. Social, *Fatiga laboral: origen, medida, su relación con los accidentes y las enfermedades profesionales y su prevención.* SUSESO-ACHS, 2021.

[35] T. Abe, "Perclos-based technologies for detecting drowsiness: current evidence and future directions," *Sleep Advances*, vol. 4, no. 1, p. zpad006, 2023.

XII. ANEXOS

Código:

Listing 1. Inicialización de la cámara y configuración del audio

```
Parte 1: Inicialización de la cámara y audio
\begin{lstlisting}[language=Python, caption=Inicialización de la cámara y
    configuración del audio, label=lst:inicializacion]
import cv2
import subprocess
import pygame

# Inicializar pygame para reproducción de sonido
pygame.mixer.init()

# Rutas de los archivos de sonido de alarma
ALARM_SOUNDS = [
    "/home/pi/mi_entorno/detector python/alarm1.wav",
    "/home/pi/mi_entorno/detector python/alarm2.wav",
    "/home/pi/mi_entorno/detector python/alarm3.ogg"
]

# Función para reproducir la alarma según el nivel
def play_alarm_sound(level):
    try:
        pygame.mixer.music.load(ALARM_SOUNDS[level % len(ALARM_SOUNDS)])
        pygame.mixer.music.play()
        print(f"Reproduciendo sonido de alarma nivel {level % len(
            ALARM_SOUNDS) + 1}...")
    except Exception as e:
        print(f"Error al reproducir el sonido de alarma: {e}")

# Configuración del comando para capturar video con libcamera-vid
command = (
    "libcamera-vid -t 0 --width 320 --height 240 --framerate 90 --bitrate
        500000 " \sloppy

    "--codec mjpeg -o - --preview 0,0,640,480 | "
    "ffmpeg -hide_banner -loglevel panic -i pipe:0 -f rawvideo -pix_fmt
        bgr24 -"\sloppy

)

# Iniciar el proceso de captura de video
pipe = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE)
```

Listing 2. Inicialización de la cámara y configuración del audio

```
Parte 2: Detección facial y cálculo de parámetros
\begin{lstlisting}[language=Python, caption=Configuración de detección
    facial y cálculo de EAR/MAR, label=lst:deteccion]
import numpy as np
import dlib
from scipy.spatial import distance as dist
from imutils import face_utils

# Cargar el detector de rostros y el predictor de puntos faciales
```

```
detector = cv2.CascadeClassifier("/home/pi/mi_entorno/detector␣
    python/haarcascade_frontalface_default.xml")
predictor = dlib.shape_predictor("/home/pi/mi_entorno/detector␣
    python/shape_predictor_68_face_landmarks.dat")

# Función para calcular el Eye Aspect Ratio (EAR)
def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear

# Función para calcular la abertura de la boca (Mouth Aspect Ratio, MAR)
def mouth_aspect_ratio(mouth):
    A = dist.euclidean(mouth[2], mouth[10])  # Distancia vertical
    B = dist.euclidean(mouth[4], mouth[8])   # Distancia vertical
    C = dist.euclidean(mouth[0], mouth[6])   # Distancia horizontal
    mar = (A + B) / (2.0 * C)
    return mar

# Parámetros de detección de somnolencia y bostezo
EYE_AR_THRESH = 0.25  # Umbral para detectar ojos cerrados
EYE_AR_CONSEC_FRAMES = 25  # Número de frames consecutivos con ojos
    cerrados
MOUTH_AR_THRESH = 0.95  # Umbral para detectar un bostezo

# Contadores y niveles de alerta
COUNTER = 0
ALERT_LEVEL = 0
ALARM_CYCLES = 0  # Contador de ciclos de alarmas
```

Listing 3. Captura de video con libcamera-vid y procesamiento de fotogramas

```
Parte 3: Captura de video y procesamiento de fotogramas
\begin{lstlisting}[language=Python, caption=Captura de video con
    libcamera-vid y procesamiento de fotogramas, label=lst:captura]
import cv2
import subprocess
import numpy as np

# Comando para capturar video con libcamera-vid
command = (
    "libcamera-vid␣-t␣0␣--width␣320␣--height␣240␣--framerate␣90␣--bitrate␣
        500000␣" \sloppy

    "--codec␣mjpeg␣-o␣-␣--preview␣0,0,640,480␣|␣"
    "ffmpeg␣-hide_banner␣-loglevel␣panic␣-i␣pipe:0␣-f␣rawvideo␣-pix_fmt␣
        bgr24␣-"
)
pipe = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE)

# Dimensiones del video
width, height = 320, 240

# Procesar fotogramas
while True:
    # Leer fotograma
    raw_frame = pipe.stdout.read(width * height * 3)
```

```
if len(raw_frame) != width * height * 3:
    print("Error: No se pudo leer el fotograma.")
    break

frame = np.frombuffer(raw_frame, dtype=np.uint8).reshape((height,
    width, 3))
frame_copy = frame.copy()

# Convertir el fotograma a escala de grises
gray = cv2.cvtColor(frame_copy, cv2.COLOR_BGR2GRAY)

# Detección de rostros
faces = detector.detectMultiScale(gray, scaleFactor=1.1,
    minNeighbors=5, minSize=(30, 30))
for (x, y, w, h) in faces:
    face = dlib.rectangle(x, y, x + w, y + h)
    landmarks = predictor(gray, face)
    landmarks = face_utils.shape_to_np(landmarks)

    # Detectar ojos y boca
    leftEye = landmarks[42:48]
    rightEye = landmarks[36:42]
    mouth = landmarks[48:68]

    # Dibujar contornos
    cv2.drawContours(frame_copy, [cv2.convexHull(leftEye)], -1, (0,
        255, 0), 1)
    cv2.drawContours(frame_copy, [cv2.convexHull(rightEye)], -1, (0,
        255, 0), 1)
    cv2.drawContours(frame_copy, [cv2.convexHull(mouth)], -1, (0, 255,
        255), 1)

    # Calcular parámetros
    leftEAR = eye_aspect_ratio(leftEye)
    rightEAR = eye_aspect_ratio(rightEye)
    ear = (leftEAR + rightEAR) / 2.0
    mar = mouth_aspect_ratio(mouth)

    # Evaluar condiciones de somnolencia
    if ear < EYE_AR_THRESH:
        COUNTER += 1
        if COUNTER >= EYE_AR_CONSEC_FRAMES:
            play_alarm_sound(ALERT_LEVEL)
            ALERT_LEVEL += 1
            COUNTER = 0

            # Cada 3 alarmas, realizar acción adicional
            if ALERT_LEVEL % len(ALARM_SOUNDS) == 0:
                ALARM_CYCLES += 1
                print(f"Ciclo de alarmas completado: {ALARM_CYCLES}")
                if ALARM_CYCLES % 3 == 0:
                    send_notification()

            # Mostrar texto de alerta de somnolencia
            cv2.putText(frame_copy, "ALERTA DE SOMNOLENCIA", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    elif mar > MOUTH_AR_THRESH:
        # Mostrar texto de bostezo
```

```
            cv2.putText(frame_copy, "ESTAS BOSTEZANDO", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255), 2)
        else:
            COUNTER = 0

    # Mostrar fotograma
    cv2.imshow("Frame", frame_copy)

    # Salir al presionar 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Listing 4. Nueva alarma y reproducción de música aleatoria

```
Parte 4: Nueva alarma y reproducción de música aleatoria
\begin{lstlisting}[language=Python]
import cv2
import subprocess
import numpy as np

command = (
    "libcamera-vid -t 0 --width 320 --height 240 --framerate 90 --bitrate
        500000 "
    "--codec mjpeg -o - --preview 0,0,640,480 | "
    "ffmpeg -hide_banner -loglevel panic -i pipe:0 -f rawvideo -"
)

pipe = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE)

width, height = 320, 240

while True:
    raw_frame = pipe.stdout.read(width * height * 3)
    if len(raw_frame) != width * height * 3:
        print("Error: No se pudo leer el fotograma.")
        break

    frame = np.frombuffer(raw_frame, dtype=np.uint8).reshape((height,
        width, 3))
    frame_copy = frame.copy()
    gray = cv2.cvtColor(frame_copy, cv2.COLOR_BGR2GRAY)

    cv2.imshow("Frame", frame_copy)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cv2.destroyAllWindows()
pipe.terminate()
```

Encuesta 1:

Objetivo de la Encuesta: Validar la precisión y efectividad del sistema de detección de fatiga.

1. ¿Cómo describirías tu experiencia general con el sistema de detección de fatiga?

Me pareció muy eficiente

2. Durante las pruebas, ¿con qué frecuencia notaste que el sistema activaba alertas cuando te sentías fatigado?

- **Casi siempre**, el sistema detectó mis momentos de fatiga a tiempo.

3. ¿Qué tan bien sincronizó el sistema las alertas con tus síntomas de fatiga (como bostezos o parpadeos)?

- **Muy bien sincronizado**: La alerta siempre fue precisa y llegó en el momento adecuado.

4. ¿Hubo alguna vez que el sistema activó una alerta cuando no te sentías fatigado? ¿Cómo te hizo sentir?

- **No.** En ningún momento se activaron alertas sin razón.

5. ¿Cómo calificarías la velocidad de respuesta del sistema cuando detectó los signos de fatiga?

- **Muy rápida**: La alerta se activó casi inmediatamente después de mostrar signos de fatiga.

6. ¿En una escala del 1 al 5, ¿cómo calificarías la efectividad general del sistema para prevenir accidentes relacionados con la fatiga? (1 siendo muy inefectivo y 5 siendo muy efectivo)

- **5**

7. ¿Te sentirías más seguro utilizando este sistema mientras realizas tareas que requieren alta concentración, como conducir o estudiar?

- **Sí, definitivamente.** Me sentiría mucho más seguro sabiendo que el sistema me alertará si empiezo a estar fatigado.

8. ¿Qué aspecto mejorarías del sistema para que sea más efectivo?

Me gustaría que las alertas fueran más rápidas o más personalizables para cada usuario.

9. ¿Recomendarías este sistema a otras personas?

- **Sí, sin duda.** Creo que podría ayudar a muchas personas a evitar accidentes por fatiga.

Encuesta 2:

Objetivo de la Encuesta: Validar la precisión y efectividad del sistema de detección de fatiga.

1. ¿Cómo fue tu experiencia al utilizar el sistema de detección de fatiga durante las pruebas?

Fue una experiencia interesante, aunque hubo algunas alertas innecesarias.

2. ¿Con qué frecuencia el sistema detectó tu fatiga correctamente y activó las alertas?

- **Muy frecuentemente**, siempre que sentí fatiga, el sistema activó una alerta.

3. ¿Qué opinas sobre el tiempo de respuesta del sistema al detectar signos de fatiga, como ojos cerrados o bostezos?

- **Adecuado**: La alerta llegó a tiempo, pero algunas veces sentí que podía mejorar.

4. ¿El sistema alguna vez te alertó cuando no te sentías fatigado? ¿Cómo afectó tu experiencia?

- **Sí, una vez.** No fue un gran problema, pero me preocupó un poco la precisión.

5. En términos generales, ¿cómo calificarías la fiabilidad del sistema para prevenir accidentes relacionados con la fatiga?

- **Fiable en su mayoría**: Aunque tiene algunas fallas, en general cumple su propósito.

6. ¿Te sentirías cómodo usándolo en tu vida diaria, como en el trabajo o al conducir?

- **Sí, con algunos ajustes.** Me gustaría que sea más rápido o más preciso en ciertas situaciones.

7. ¿Qué mejoras agregarías para hacer el sistema más efectivo?
Que el sistema sea más ajustable dependiendo de cada usuario y situación.

8. ¿Recomendarías el sistema a otros usuarios?

- **Sí, lo recomendaría.** Creo que tiene un gran potencial para mejorar la seguridad.

Encuesta 3:

Objetivo de la Encuesta: Validar la precisión y efectividad del sistema de detección de fatiga.

1. ¿Cómo evaluas la efectividad del sistema para detectar los signos de fatiga, como los bostezos o el parpadeo?

- *El sistema funcionó muy bien en general, aunque a veces era demasiado sensible.*

2. ¿Con qué frecuencia se activaron las alertas de fatiga mientras lo probabas?

- **A menudo**: Hubo algunas alertas adicionales, pero en general coincidieron con mi fatiga.

3. ¿Qué tan sincronizadas estuvieron las alertas con los momentos de fatiga?

- En algunos casos, llegaron un poco temprano o tarde, pero no fue un gran problema.

4. ¿El sistema activó alguna alerta cuando no sentías fatiga? ¿Cómo afectó tu experiencia?

- **Sí, una vez.** Fue raro, pero no me molestó mucho.

5. En una escala del 1 al 5, ¿cómo calificarías el sistema para prevenir accidentes?

- **3**

6. ¿Qué tan seguro te sentirías usando el sistema en situaciones de alta concentración, como conducir o trabajar?

Moderadamente seguro: A veces dudaría, pero me sentiría más confiado que sin él.

7. ¿Qué sugerencias tienes para mejorar la precisión del sistema?

Quizás permitir ajustes manuales para la sensibilidad y personalización para cada usuario.

8. ¿Recomendarías este sistema a otras personas?

Tal vez. Me parece que necesita más ajustes antes de ser útil a largo plazo.

Figura 28. Vista del prototipo para la detección de somnolencia en conductores.