

APLICACIÓN MÓVIL PARA DETECCIÓN DE FATIGA A PARTIR DE VIDEOS DEL
ROSTRO DE UNA PERSONA, EN EL ESPECTRO VISIBLE

VICTOR VILLOTA QUINTERO

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
SANTIAGO DE CALI
2024

APLICACIÓN MÓVIL PARA DETECCIÓN DE FATIGA A PARTIR DE VIDEOS DEL
ROSTRO DE UNA PERSONA, EN EL ESPECTRO VISIBLE

VICTOR VILLOTA QUINTERO

Trabajo de grado para optar por el título de Ingeniero Electrónico

Directores

Sandra Esperanza Nope Rodríguez, Ph.D.

Andrés David Restrepo Girón, Ph.D.

UNIVERSIDAD DEL VALLE
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA
SANTIAGO DE CALI
2024

Nota de aceptación:

Firma del director del trabajo

Firma del director del trabajo

Firma del evaluador

Firma del evaluador

Firma del evaluador

DEDICATORIA

A Dios, por ser mi guía espiritual, por darme la fortaleza y sabiduría necesarias para superar cada desafío. Gracias por ser mi guía y por iluminar mi camino.

A mi madre, por su amor incondicional, su apoyo inquebrantable y por enseñarme el valor del esfuerzo y la perseverancia. Gracias por tus bendiciones y ser mi fuente de inspiración constante.

A mi padre, por su sabiduría, su paciencia y por siempre creer en mí. Gracias por tus consejos, tu fortaleza y por enseñarme a nunca rendirme.

A mi hermano, por su apoyo constante, su amistad y por estar siempre dispuesto a ayudarme. Gracias por ser un ejemplo a seguir y por hacer este viaje más llevadero.

CONTENIDO

	pág.
1. INTRODUCCIÓN	2
1.1. PLANTEAMIENTO DEL PROBLEMA	2
1.2. ANTECEDENTES	3
1.3. OBJETIVOS	7
1.3.1. Objetivo general.	7
1.3.2. Objetivos específicos.	7
2. JUSTIFICACIÓN	7
3. MARCO TEÓRICO	8
3.1. DEFINICIÓN DE FATIGA	8
3.2. EVALUACIÓN DE LA FATIGA	8
3.3. MANIFESTACIONES DE FATIGA EN EL ROSTRO DE PERSONAS	9
3.3.1. Alteración de parpadeo	10
3.3.2. Bostezo	10
3.3.3. Movimientos De Cabeza	11
3.3.4. Coloración de mejillas	11
3.3.5. Otras manifestaciones	11
3.3.6. Selección de manifestaciones	12
3.4. ALGORITMOS PARA DETECCIÓN DE REGIONES DE INTERÉS	13

3.4.1. Algoritmo de Viola & Jones	13
3.4.2. Histograma de gradiente orientados (HOG)	16
3.4.3. Localización de puntos del rostro (Landmark)	17
3.4.4. Transformada de Hough circular	17
3.4.5. Redes neuronales	18
3.4.6. Selección de algoritmos	19
3.5. MÉTODOS PARA DETECCIÓN DE PARPADEO Y BOSTEZOS	19
3.5.1. Ausencia de detección (Viola & Jones)	19
3.5.2. Relación de aspecto del ojo (EAR)	19
3.6. INDICADORES DE FATIGA	20
3.6.1. Porcentaje de cierre de ojo (PERCLOS)	20
3.6.2. Frecuencia del parpadeo	20
3.6.3. Duración del parpadeo	20
3.6.4. Presencia de bostezos	21
3.7. HERRAMIENTAS DE PROGRAMACIÓN	21
3.7.1. Opencv	22
3.7.2. Dlib	22
3.8. BASE DE DATOS	22
4. SISTEMA PARA DETECCIÓN DE FATIGA	23
4.1. ALGORITMOS PARA DETECCIÓN DE MANIFESTACIONES DE FATIGA	23

4.1.1. Detección de rostro	24
4.1.2. Detección de ojos y boca	25
4.1.3. Detección de parpadeos y bostezos	27
4.2. ESTRATEGIA PARA LA DETECCIÓN DE FATIGA	30
4.3. INTEGRACIÓN DEL SISTEMA FINAL PARA LA DETECCIÓN DE FATIGA	31
4.4. PRE-PROCESAMIENTO DE IMÁGENES	32
4.5. DISEÑO DE LA INTERFAZ DE LA APLICACIÓN	34
4.6. PROTOCOLO DE PRUEBAS	37
4.6.1. Variaciones de iluminación	37
4.6.2. Variaciones de inclinación de cámara	39
4.6.3. Variaciones de fondo	42
5. ANÁLISIS Y RESULTADOS	42
5.1. RESULTADOS OBTENIDOS EN EL PROTOCOLO DE PRUEBAS	42
5.1.1. Variaciones de iluminación	42
5.1.2. Variaciones de inclinación de cámara	56
5.2. RESULTADOS DEL SISTEMA FINAL PARA LA DETECCIÓN DE FATIGA	63
6. CONCLUSIONES	67
7. TRABAJOS A FUTURO	68
8. BIBLIOGRAFÍA	70

LISTA DE TABLAS

	pág.
Tabla 1. Parámetros para entrenamiento del clasificador HAAR (detección de boca)	25
Tabla 2. Resultados de la detección de parpadeos por ausencia de Viola & Jones.	29
Tabla 3. Resultados de la detección de bostezos por ausencia de Viola & Jones.	29
Tabla 4. Resultados de las pruebas con la simulación de fatiga (sujeto 1).	64
Tabla 5. Resultados de las pruebas con la simulación de fatiga (sujeto 2)	64
Tabla 6. Resultados de las pruebas con la simulación de fatiga (sujeto 3).	65
Tabla 7. Resultados de las pruebas con la simulación de fatiga (sujeto 4)	65
Tabla 8. Resultados de las pruebas con la simulación de fatiga (sujeto 5).	66

LISTA DE FIGURAS

	pág.
Figura 1. Parpadeo	12
Figura 2. Bostezo	13
Figura 3. Filtros HAAR	14
Figura 4. Aplicación de filtros HAAR sobre una imagen	14
Figura 5. Imagen integral	15
Figura 6. Diagrama de clasificadores en cascada	16
Figura 7. Histograma de gradientes orientados	16
Figura 8. Representación de los 68 puntos faciales del rostro humano	17
Figura 9. Fotograma de ojo tras aplicar algoritmo de Canny	18
Figura 10. Puntos de referencia para detección de parpadeo por medio de EAR	19
Figura 11. Niveles de porcentaje del ojo abierto	20
Figura 12. Etapas de sistema para detección de fatiga	23
Figura 13. Algoritmos para cada etapa para la detección de manifestaciones de fatiga	24
Figura 14. Detección de rostro usando clasificador HAAR	25
Figura 15. Detección de rostros, ojos y boca, usando clasificadores en cascada	26
Figura 16. División del rostro para detección de ojos (recuadro azul) y detección de boca (recuadro violeta)	26

Figura 17. Detección con variaciones de iluminación, inclinación de cámara y obstáculos visuales, usando Viola & Jones	27
Figura 18. Conteo de parpadeos a partir de ausencia de detección (Viola & Jones)	28
Figura 19. Conteo de bostezos a partir de ausencia de detección (Viola & Jones)	29
Figura 20. Etapas del sistema para la detección de manifestaciones de fatiga	30
Figura 21. Sistema final para la detección de fatiga en el rostro humano	32
Figura 22. Procesado de imagen inicial, aplicando escala de grises y filtro	33
Figura 23. Aplicación de ecualización de histograma a la imagen filtrada	33
Figura 24. Descripción de la interfaz de la aplicación móvil	34
Figura 25. Esquema de iluminación de tres puntos	38
Figura 26. Posición del celular sin inclinaciones con respecto al rostro	38
Figura 27. Rotación del dispositivo en plano horizontal (vista frontal)	40
Figura 28. Rotación del rostro con respecto al dispositivo en el plano horizontal (vista desde arriba)	40
Figura 29. Rotación del rostro en plano vertical con respecto al dispositivo.	41
Figura 30. Detección con variaciones de fondo	41
Figura 31. Detección de Rostro con variación de luz principal.	43
Figura 32. Detección de Rostro con variación de la luz de fondo.	44
Figura 33. Detección de Rostro con variación de la luz de relleno.	45
Figura 34. Detección de ojos con variación en la luz principal.	46

Figura 35. Detección de ojos con variación en la luz de fondo	47
Figura 36. Detección de ojos con variación de la luz de relleno.	48
Figura 37. Detección de la boca con variaciones de la luz principal.	49
Figura 38. Detección de la boca con variaciones de la luz de fondo.	50
Figura 39. Detección de la boca con variación de la luz de relleno	51
Figura 40. Detección de rostro (sin luz de fondo y luz de relleno constante)	52
Figura 41. Detección de ojos (sin luz de fondo y luz de relleno constante).	53
Figura 42. Detección de boca (sin luz de fondo y luz de relleno constante).	54
Figura 43. Iluminación para obtener un entorno controlado en términos de luz	55
Figura 44. Detección rostro con inclinación de la cámara en el eje horizontal.	57
Figura 45. Detección ojos inclinación de la cámara en el eje horizontal.	58
Figura 46. Detección boca inclinación de la cámara en el eje horizontal.	59
Figura 47. Detección de rostro con inclinación de la cámara en el eje vertical.	60
Figura 48. Detección de ojos con inclinación de la cámara en el eje vertical.	61
Figura 49. Detección de boca con inclinación de la cámara en el eje vertical.	62

LISTA DE ANEXOS

	pág.
Anexo A. Diagrama del algoritmo simplificado para la detección de fatiga	69
Anexo B. Casos de uso real	70
Anexo C. Tablas de porcentaje de detección con variaciones de iluminación	74
Anexo D. Tablas de porcentaje de detección con variaciones de inclinación de la cámara	74
Anexo E. Código de la clase principal	74
Anexo F. Código de la clase para la detección de fatiga	77
Anexo G. Código de la clase para la carga de clasificadores	82
Anexo H. Código de la clase para simulación de fatiga	83
Anexo I. Código de la clase para manejo de tiempos	85

RESUMEN

Este documento presenta el desarrollo de una aplicación móvil capaz de detectar fatiga en las personas y dar una alerta en tiempo real, mediante el uso de técnicas de visión artificial. Inicialmente se hace una recopilación de las señales fisiológicas que se manifiestan en el rostro humano y escogiendo las más relevantes, para ser cuantificadas usando procesamiento de imágenes. Adicionalmente se hace una exploración de algoritmos y métodos que pueden emplearse para detectarlas, teniendo en cuenta las limitaciones de procesamiento que presenta el dispositivo.

El estudio también presenta un protocolo de pruebas, realizado a un grupo de 5 personas, que considera variaciones de iluminación, inclinación de cámara y cambios de fondo para determinar las condiciones necesarias para las cuales el aplicativo presenta su mejor funcionamiento.

Finalmente, se diseña un método para determinar la presencia de fatiga, realizando una prueba final, en la cual, se simula el estado de fatiga para cada individuo del grupo de prueba, con ayuda de indicadores en la pantalla del dispositivo y verificando así, la precisión del sistema implementado en el móvil.

Palabra Clave: Fatiga, visión artificial, Viola & Jones, algoritmo, iluminación, inclinación, fondo, entorno.

SUMMARY

This document presents the development of a mobile application capable of detecting fatigue in individuals and providing real-time alerts using computer vision techniques. Initially, a collection of physiological signals manifested in the human face is compiled, selecting the most relevant ones to be quantified using image processing. Additionally, an exploration of algorithms and methods that can be employed to detect these signals is conducted, taking into account the processing limitations of the device.

The study also presents a testing protocol, carried out on a group of 5 people, which considers variations in lighting, camera angle, and background changes to determine the conditions under which the application performs best.

Finally, a method is designed to determine the presence of fatigue, conducting a final test in which the state of fatigue is simulated for each individual in the test group. This is achieved with the help of indicators on the device's screen, thereby verifying the accuracy of the system implemented on the mobile device.

Keywords: Fatigue, computer vision, Viola & Jones, algorithm, lighting, inclination, background, environment.

1. INTRODUCCIÓN

La presencia de agotamiento mental y físico en las personas, de forma excesiva se conoce comúnmente como fatiga y constituye uno de los principales problemas en la actualidad. Este estado influye de manera negativa en el rendimiento cognitivo y físico, lo que podría ocasionar bajo desempeño en actividades cotidianas, accidentes laborales o de tránsito e incluso, desencadenar problemas de salud a largo plazo como estrés, depresión y ansiedad. Tomando en consideración lo anterior, su identificación temprana es de vital importancia como estrategia de prevención ante sus consecuencias.

En este proyecto, se desarrolló una aplicación móvil con la capacidad de detectar la fatiga, a partir de tres manifestaciones provenientes del rostro, las cuales son: la frecuencia de parpadeo, la duración de los mismos y la aparición de bostezos. Debido a que este estado puede ser provocado por múltiples factores, el objetivo principal fue crear una herramienta con fácil acceso a las personas y que se pueda usar en diferentes entornos. Por consiguiente, el sistema implementado realiza primero, una segmentación del rostro y las regiones de interés (ojos y boca) usando técnicas basadas en el algoritmo desarrollado por Viola & Jones, posteriormente busca los cambios fisiológicos mencionados anteriormente y por último, procede a dar una alerta en caso de presencia de fatiga.

1.1. PLANTEAMIENTO DEL PROBLEMA

Actualmente el ser humano tiene un ritmo de vida bastante acelerado dadas las diferentes exigencias de la sociedad, el ambiente que lo rodea, los problemas económicos, el constante acceso a la tecnología y medios de comunicación logran sobreestimar y saturar a las personas, las largas y exhaustivas jornadas de trabajo y/o estudio provocan un estado de hipervigilancia o preocupación constante, estos factores, combinados o aislados, aunados a todas las actividades diarias, generan en las personas la necesidad de aprovechar cada minuto de su vida exigiendo al máximo sus capacidades físicas y mentales, produciendo un estado de agotamiento y cansancio al finalizar el día conocido comúnmente como fatiga [1][2].

De acuerdo a A. Hervás [3], la fatiga es un proceso en el que el sujeto pierde paulatinamente su capacidad de respuesta a los estímulos, deteriorando la calidad y la precisión, eficiencia y ejecución de una acción simple o compleja, las causas pueden ser múltiples e incluyen factores fisiológicos y psicológicos. En la actualidad B. Pedraz [4] la definió como: “Una lasitud o cansancio que se exterioriza en las esferas corporales y/o mentales”.

De la misma forma, el estudio llevado a cabo por la Universidad Complutense de Madrid [1], refiere que la fatiga tiene una gran incidencia en el proceso de toma de decisiones y en alteraciones de tipo atencional que conlleva, al aumento en la distracción del individuo que la padece.

A razón de lo anterior, se puede inferir que este fenómeno afecta a la persona en todas las esferas de su vida ya sea a nivel familiar, social o laboral, debido a que, puede repercutir en el estado óptimo de una persona e indirectamente puede desencadenar en la aparición de accidentes laborales o de tránsito, bajo rendimiento en el desempeño laboral, académico o en cualquier tipo de actividad en general [1].

En los últimos años, muchos trabajos de investigación han usado visión artificial para implementar un sistema de detección de fatiga en las personas a partir de las expresiones faciales, por ejemplo, J. P. Arias [5], señala que se puede determinar si una persona presenta cansancio de acuerdo a algunas alteraciones fisiológicas en el cuerpo humano, entre los que se encuentran: expresiones faciales, movimientos de la cabeza, bostezos, duración de parpadeo, cambios de flujo sanguíneo, apertura de los ojos. Sin embargo, es de interés resaltar que, la mayoría de sistemas se han desarrollado haciendo uso de algoritmos programados en computadora, principalmente porque requieren de buena capacidad de procesamiento de imágenes, limitando su funcionamiento a un ambiente fijo. Por otro lado, resulta oportuno destacar la evolución de la tecnología en dispositivos móviles durante los últimos años, la cual ha provocado una mayor capacidad procesamiento y adicionalmente tienen mayor incidencia en la vida cotidiana, brindando así, un escenario idóneo con el cual se plantea la pregunta:

¿Cómo detectar la fatiga en un aplicativo móvil, a partir de videos del rostro de una persona en el espectro visible?

1.2. ANTECEDENTES

Como se mencionó anteriormente, se ha encontrado evidencia de que algunas señales fisiológicas del cuerpo humano, pueden asociarse a la fatiga, como lo son: alteraciones bioquímicas en la sangre u orina, cambios de flujo sanguíneo facial, frecuencia de parpadeo y amplitud de apertura de los ojos, bostezos, entre otros [5]. Lo que permitió hacer un recuento de trabajos basados en visión artificial para la detección de dichas señales y que son la base para esta investigación. El estado de arte realizado abarca trabajos relacionados con la fatiga, en particular desarrollados en el contexto colombiano y extranjero.

Se considera importante destacar el estudio desarrollado por A. Juan Pablo, “Detección de signos de fatiga en humanos utilizando imágenes del rostro” (Colombia, 2020), en la Universidad del Valle [5]. Este trabajo recopila señales asociadas a la fatiga, como parpadeos, bostezos y coloración en las mejillas, realiza pruebas con distintos algoritmos para identificar estos signos, como por ejemplo; algoritmo de Viola & Jones, localización de puntos faciales (*Landmarks*) e histograma de gradientes (HOG), dando a conocer sus ventajas y desventajas. Finalmente, concluye que hay una relación directa entre el aumento de parpadeo y la aparición de bostezos con el estado de fatiga en las personas. Tales contribuciones proporcionan una base sólida para comprender cómo pueden ser identificados y cuantificados los signos de fatiga mediante técnicas de visión artificial.

En la investigación “Sistemas de detección de nivel de cansancio en conductores mediante técnicas de visión por computador” (Colombia, 2011), realizado por R. J. Moreno [6], se desarrolló un sistema de detección de fatiga en un conductor, considerando como características: la amplitud de los ojos, apertura de la boca y movimiento de la cabeza, dichos parámetros se introdujeron a un clasificador diseñado con una red neuronal realimentada el cual genera como respuesta uno de tres estados posibles (No fatiga, Alerta y Crítico). El clasificador construido presentó el mejor resultado cuando se incluyeron estados precedentes de la persona con un 91% de precisión, siendo necesario dar alertas tempranas, el aporte de este proceso es precisamente evidenciar cómo es posible detectar la fatiga a partir de signos faciales.

“Aplicación móvil para la detección de somnolencia de un conductor aplicando visión artificial” (Ecuador, 2016) [7]. En este proyecto se desarrolló una aplicación móvil de vanguardia para detectar la somnolencia en conductores. La aplicación consta de tres etapas fundamentales: la primera fase realiza la detección del rostro del conductor; la segunda, detecta los movimientos oculares; la etapa final determina el estado del conductor mediante el método del Porcentaje de Tiempo con los Ojos Cerrados (PERCLOS). Esta implementación se ha llevado a cabo en una aplicación nativa diseñada para el sistema operativo Android.

En el proyecto de “Aplicación para teléfono móvil con sistema operativo Android que permite detectar somnolencia y emitir una alarma sonora a conductores de automóvil mediante procesamiento de imágenes” (Colombia, 2019) [8], se realizó un aplicativo para el teléfono, que permite estimar el estado de somnolencia de un conductor mediante el análisis de los ojos, en el cual se implementa los algoritmos de Viola & Jones, HOG y landmarks para detección de rostro y ojos, además se usó PERCLOS para determinar la somnolencia, el cual evalúa la duración en la que los ojos permanecen cerrados basándose en su grado de apertura. Los resultados obtenidos en este trabajo tuvieron aproximadamente un 94.5% de eficiencia en la clasificación del estado de la persona, en un grupo de 10 personas con diferentes características faciales. Su desarrollo permitió obtener información sobre los el tipo de algoritmos soportados y compatibles para un dispositivo móvil.

“Predict Driver Fatigue Using Facial Features” (Marruecos, 2018) [9]. Este trabajo propone un método para predecir de forma temprana los signos de fatiga durante la conducción usando las características faciales que reflejan el estado que presentan los conductores, se usa el método de Viola & Jones para la detección de rostros, por ser efectiva y sencilla de implementar con una cámara de teléfono estándar equipada con LED IR. Posteriormente, todas las características faciales extraídas se presentan a una Máquina de Vectores de Soporte (SVM) para su clasificación, diferenciando entre el estado normal y el estado crítico, lo cual se logró analizando la velocidad/duración de parpadeo y PERCLOS, los resultados de esta investigación permitió conocer que la velocidad y la duración de los parpadeos también son indicadores significativos de la fatiga.

Por su parte en el estudio *“Research on tiredness detection method based on fusion of face image and EEG information”* (EE.UU, 2021) [10], se combinan métodos de detección de cansancio por medio de imágenes y señales electroencefalográficas (EEG), para evaluar el estado de fatiga en conductores, en el experimento se diseñó un ambiente simulado para los conductores, en el que se recopilaban estos datos de forma simultánea, los resultados obtenidos señalan que la fusión de estos dos métodos, presenta mayor precisión y adaptabilidad en comparación al uso de cada método por separado, esta investigación demuestra que la fatiga puede producir múltiples señales en el cuerpo humano y entre mayor sea el número de señales analizadas, será más fácil detectarla y con una precisión mayor.

“Driver Drowsiness Detection Using Fixed and Dynamic Thresholding” (EE.UU, 2022) es un trabajo realizado en colaboración entre *Southern University and A&M College* y la universidad de Bahrein [11]. Este estudio propuso dos métodos para detectar la somnolencia durante la conducción usando características faciales que reflejan el estado de alerta de los conductores. Se empleó el algoritmo de detección facial basado en Dlib, elegido por su eficacia y facilidad de implementación, utilizando una cámara estándar y procesamiento de imágenes en un ordenador. El sistema calcula dos parámetros clave: la Relación de Aspecto del Ojo (EAR) y la Relación de Apertura de la Boca (MOR). Se implementaron dos enfoques: un umbral fijo para la detección de parpadeos y bostezos y un umbral dinámico para los fotogramas. El umbral dinámico considera que, a mayor tiempo de conducción, menor es el número de fotogramas consecutivos necesarios para considerar a una persona somnolienta, adaptándose así a la fatiga acumulada en viajes largos y mejorando la precisión de la detección.

En la investigación *“Driver Drowsiness Detection Using DLIB”* (India, 2022) [12] realizada en la universidad Galgotias, se propone un método para detectar la somnolencia durante la conducción usando características faciales, específicamente los patrones de parpadeo, que reflejan el estado de alerta de los conductores. Se empleó el algoritmo de detección facial basado en DLIB, utilizando una cámara estándar y procesamiento de imágenes en Python. El sistema calcula un parámetro clave: EAR y posteriormente utiliza un clasificador de SVM para reducir las detecciones falsas

positivas de parpadeos y mejorar la eficiencia del modelo, clasifica el estado del conductor en tres etapas: activo, somnoliento y dormido, activando una alarma en este último caso. Los autores destacan la importancia de considerar las diferencias individuales, especialmente en el tamaño de los ojos, al desarrollar algoritmos basados en visión por computadora para la detección de somnolencia.

"Driver Assistance Systems with Driver Drowsiness Detection Using Haar-Cascade Algorithm" (2023) [13] es un trabajo llevado a cabo por investigadores del Pillai HOC College of Engineering and Technology. Este estudio propone un sistema de asistencia al conductor diseñado para detectar la somnolencia mediante el análisis de las características faciales del conductor, utilizando una cámara y una placa Raspberry Pi, el sistema captura imágenes en tiempo real, que son procesadas mediante el algoritmo Haarcascade. El sistema se enfoca principalmente en la detección de ojos para identificar el estado de fatiga del conductor. Además, cuando el sistema detecta somnolencia, activa un mecanismo que detiene automáticamente el vehículo.

El trabajo realizado por P. Sudarshan et al. "Real-time Driver Drowsiness Detection and Assistance System using Machine Learning and IoT" (India, 2023) [14] se diseñó un sistema de detección de somnolencia en tiempo real utilizando el algoritmo Haarcascade y redes neuronales combinacionales para la clasificación de fatiga, integrado con tecnologías IoT para la alerta y prevención de accidentes. Desarrollado en un Raspberry Pi 3 Model B+, para facilitar su portabilidad y bajo costo, se centra en detectar si los ojos del conductor permanecen cerrados durante un periodo prolongado, empleando una cámara para monitorizar el rostro. Cuando se detecta somnolencia, se activan mecanismos de alerta como la emisión de un sonido, el envío de un mensaje de alerta y, de manera innovadora, la activación de un rociador de agua para despertar físicamente al conductor. La investigación considera al algoritmo de Haarcascade como el más eficiente para la detección de imágenes positivas y no deseadas.

Por otro lado, el estudio "Driver Drowsiness Detection Using Machine Learning" (Mumbai, 2023) [15], desarrollado en el K J Somaiya Institute of Technology, propone un sistema basado en una aplicación móvil que emplea algoritmos de aprendizaje automático para detectar la somnolencia del conductor. El sistema utiliza una cámara para monitorizar las expresiones faciales, como la duración del cierre de los ojos y los movimientos de la cabeza. Además, se integran datos del acelerómetro para detectar vibraciones y movimientos del vehículo, lo que ayuda a identificar señales de fatiga. Los algoritmos empleados, como las Redes Neuronales Convolucionales (CNN), permiten el procesamiento en tiempo real de estas señales, generando predicciones sobre el estado de somnolencia del conductor, y emitiendo alertas visuales o auditivas para prevenir accidentes.

Los estudios revisados muestran una clara tendencia hacia el uso de dispositivos portátiles como medios para la implementación de sistemas de detección de fatiga. Algoritmos clásicos como el de Viola & Jones, Landmarks, han sido predominantes en estas aplicaciones debido a su eficacia en detección de rostros y características

faciales. En los últimos años, se presentan muchos avances tecnológicos que han permitido empezar a incorporar algoritmos con redes neuronales para mejorar la precisión y adaptabilidad de los sistemas, sin embargo los algoritmos tradicionales siguen teniendo base fundamental debido a su simplicidad, particularmente, en escenarios donde la capacidad de procesamiento y la energía son limitadas, como teléfonos móviles.

1.3. OBJETIVOS

1.3.1. Objetivo general. Desarrollar un aplicativo móvil para la detección de fatiga, a partir de videos del rostro de una persona en el espectro visible, con pequeñas variaciones de iluminación, fondo e inclinación de cámara.

1.3.2. Objetivos específicos.

- Implementar algoritmos para la detección de manifestaciones de fatiga en rostros de personas a partir de videos en el espectro visible, ejecutables en un dispositivo móvil.
- Integrar los algoritmos implementados para la detección de manifestaciones de fatiga para establecer la presencia o ausencia de fatiga.
- Establecer los alcances y limitaciones del proyecto con base en la capacidad de procesamiento de los dispositivos móviles.

2. JUSTIFICACIÓN

La indagación frente al fenómeno de la fatiga constituye actualmente un problema de salud pública a nivel mundial, informes realizados por entidades promotoras de salud y seguridad vial aseguran que cerca del 20% de siniestros viales y el 30% de accidentes graves están relacionados a esta condición [16], [17], así como las lesiones y accidentes en diferentes ambientes laborales [18]. Estos datos ponen en manifiesto la importancia de la detección temprana de la fatiga como mecanismo de prevención de percances usando métodos no invasivos (técnicas de visión artificial).

Anteriormente se mencionó que en la mayoría de los estudios se aborda esta problemática desarrollando sistemas para la detección de fatiga usando computadores. Sin embargo, dado que la fatiga puede presentarse en diversos contextos, es necesario que las investigaciones se guíen hacia el desarrollo tecnológico actual, utilizando

dispositivos de mayor alcance y fácil acceso para la población en diferentes entornos, como son los teléfonos celulares. El uso de estos dispositivos se ha incrementado considerablemente en la última década, en 2018, el Fondo Monetario Internacional (FMI) destacó el incremento en la venta de celulares, que superó al de los ordenadores como medio informativo [19]. Por otro lado, un informe más reciente, publicado por la asociación GSMA (*Global System for Mobile Communications*) a finales del año 2023, revela que más de la mitad de la población mundial posee un smartphone, implicando que más de 4.300 millones de personas usan estos dispositivos móviles [20]. Con base en todo lo anterior y teniendo en cuenta el gran avance tecnológico de los smartphones ha hecho de la visión artificial móvil un concepto realista, la implementación de un sistema para la detección de fatiga en un dispositivo móvil resulta ser novedoso para los usuarios despertando interés por su fácil acceso, portabilidad, uso generalizado y funcionalidad en diversos entornos.

Por otro lado, este estudio analizó las manifestaciones en el rostro de una persona, a fin de establecer si existe presencia o ausencia de fatiga en los individuos usando el algoritmo de Viola & Jones, considerando pequeñas variaciones del entorno tales como: iluminación, fondo e inclinación de cámara, desarrollando un sistema que puede dar una alerta a tiempo, generando así una herramienta para la prevención de accidentes, que puede mejorar la seguridad de las personas en el ámbito laboral y también personal.

El aporte a la Institución se da en cuanto al estudio en términos de investigación y desarrollo académico, lo que permite tener un bagaje más amplio de conocimientos y procedimientos útiles que fusionan la tecnología y la teoría en búsqueda de aportar al desarrollo, bienestar y calidad de vida y que a su vez sirva como un precedente, para futuras investigaciones, hecho que constituye un tema de interés para el programa de Ingeniería Electrónica de la Universidad del Valle, en especial, el grupo de investigación en Percepción y Sistemas Inteligentes (PSI), ya que, hasta el momento, no existen estudios previos que hayan explorado el diseño de un aplicativo móvil para la detección de fatiga.

3. MARCO TEÓRICO

3.1. DEFINICIÓN DE FATIGA

En el artículo “*Defining and measuring fatigue*”, publicado por la revista: *The Journal of Nursing Scholarship* (JNS), se abordan los diferentes esfuerzos realizados por estudios previos para encontrar un concepto claro de la fatiga, es definida como: “la conciencia de una capacidad disminuida para la actividad física y/o mental debido a un desequilibrio en la disponibilidad, utilización o restauración de los recursos necesarios

para realizar la actividad” [21], además, se presentan los métodos más relevantes encontrados para su medición, los cuales se basan en escalas que contemplan la aparición de diferentes manifestaciones físicas y mentales, aunque, es importante mencionar que dichos signos generados, son bastantes subjetivos.

En el ámbito de la medicina, este fenómeno se relaciona con el cansancio que presenta una persona, durante o luego de una actividad física y/o mental. Según la Biblioteca Nacional de Medicina de EE. UU. (NLM) la fatiga se puede definir como: “sensación de falta de energía, de agotamiento o de cansancio, generalmente como respuesta al esfuerzo físico, estrés emocional o falta de sueño” [22].

3.2. EVALUACIÓN DE LA FATIGA

La percepción de la fatiga puede ser subjetiva, motivo por el cual la comunidad científica la ha buscado definir, de manera que pueda ser cuantificable. Algunos componentes psicopatológicos considerados para validación de instrumentos que puedan estimar dicho estado son: la disminución de energía, cansancio, reducción de la resistencia física, debilidad generalizada, sensación de pesadez en los brazos o piernas, somnolencia, disminución de motivación y concentración [4].

Según el Manual MSD, una referencia médica reconocida a nivel mundial, los métodos realizados por los médicos para confirmar la presencia de fatiga de una forma objetiva, incluyen una exploración sobre síntomas como: debilidad, dificultad respiratoria y adormecimiento [23].

3.3. MANIFESTACIONES DE FATIGA EN EL ROSTRO DE PERSONAS

El estudio de la fatiga humana ha despertado un gran interés en distintos campos, dado que la detección y diagnóstico precisos de éste estado son esenciales para garantizar la seguridad y el rendimiento óptimo de las personas en diversas áreas, como la conducción, sectores de salud y/o trabajo industrial, en este sentido, el análisis de indicadores faciales se ha convertido en una línea de investigación prometedora para su detección.

Como se indicó en los antecedentes de este documento, se ha comprobado que el cansancio produce cambios fisiológicos en las personas y que pueden ser detectados haciendo uso de diferentes técnicas de visión artificial.

3.3.1. Alteración del parpadeo. El parpadeo es un fenómeno que consiste en el cierre y apertura rápida de los párpados de forma involuntaria y espontánea, esto quiere decir que no se necesitan estímulos externos que lo provoquen. Por lo general, este evento dura aproximadamente entre 100 y 200 milisegundos en condiciones normales con un promedio de 15 parpadeos por minuto. Estos parámetros sufren alteraciones cuando las personas comienzan a experimentar diferentes niveles de fatiga, aumentando tanto la frecuencia como la duración del parpadeo. Estos datos son respaldados por estudios como el de Caffier et al. [24], en el que se hizo una evaluación a sujetos que hacían parte de un control de tráfico aéreo y su relación con diferentes medidas del parpadeo obtenidas durante el tiempo que estos se dedicaron a dicha tarea, por su parte, en la investigación de G. Barbato et al. [25], se enfocaron en determinar la variación de la tasa de parpadeo en las personas a lo largo del día, como resultado, se obtuvo que la tasa de parpadeo espontáneo aumentó significativamente en las horas de la noche al igual que el nivel de cansancio en los participantes, de la misma manera en los trabajos de Stern et al. [26], Carpenter A. [27] y Lukesch M. [28] llegan a conclusiones similares.

3.3.2. Bostezo. Este síntoma se caracteriza por una apertura involuntaria de la boca, realizando una inhalación profunda y prolongada de aire, la duración promedio de este fenómeno es de 5 segundos [4]. La investigación realizada por Noboa et al. [29], analizó los signos y síntomas de la fatiga física y mental, dicho trabajo menciona que, entre los síntomas más frecuentes, se encuentra “el deseo de bostezar todo el tiempo cuando la persona tiene un nivel de fatiga considerable”. A su vez, en el trabajo de Abtahi et al. [30], se realizaron experimentos en los que se determinó que el acto de bostezar se relaciona con la somnolencia, como consecuencia del cansancio y la fatiga que presenta la persona y como un reflejo involuntario para estirar los músculos faciales con el fin de mantenerse despierto.

3.3.3. Movimientos De Cabeza. Los movimientos de como el cabeceo están estrechamente asociados con la somnolencia. Cuando una persona experimenta cansancio o fatiga, es común observar dificultades para mantener la cabeza erguida y la mirada enfocada en un punto fijo. El cabeceo se define como movimientos involuntarios y repetitivos de la cabeza hacia abajo y hacia arriba. Esta manifestación se considera un indicador claro de la disminución del estado de alerta y la presencia de somnolencia [9].

3.3.4. Coloración de mejillas. Algunos estudios han sugerido que el cambio de la coloración de las mejillas también están relacionados con la fatiga, por la reducción de circulación de sangre, justo como se evidencia en el trabajo realizado por A. E. Sulavko y S. S. Zhumazhanova [31]. Los investigadores llevaron a cabo mediciones de la perfusión sanguínea en diversas zonas faciales; como la frente, las mejillas y la nariz, mientras los participantes llevaban a cabo tareas que inducían fatiga mental y los resultados obtenidos revelaron una disminución en la perfusión sanguínea sobre las áreas faciales medidas durante los estados de fatiga, proponiendo que los cambios en el flujo sanguíneo facial pueden utilizarse como un indicador de fatiga.

Otro estudio que abordó la detección de la fatiga mediante cambios en el flujo sanguíneo facial, fue realizado por Hassoumi et al, en el cual se hizo uso de la termografía para medir cambios en el flujo sanguíneo facial como una herramienta para la detección de la carga de trabajo cognitivo de pilotos en entornos operativos, concluyendo que: “un nivel elevado de carga de trabajo cognitivo puede tener repercusiones negativas, como la fatiga mental, el estrés y la disminución de la precisión y la velocidad en la realización de la tarea” [32].

3.3.5. Otras manifestaciones. La investigación llevada a cabo por G. Sikander y S. Anwar [33], presentó un método para detección de fatiga del conductor basado en la identificación de unidades de acción facial (AU) del sistema FACS (*Facial Action Coding System*), el cual, se compone de códigos numéricos asignados a cada movimiento muscular facial para describir y clasificar las expresiones del rostro humano. En los resultados arrojados por el estudio se evidenciaron tres acciones faciales como principales indicadores de fatiga: parpadeo, levantamiento de cejas y depresión en la comisura de los labios.

- **Blink (Parpadeo) [AU45]**

Parpadeo (AU45): Se refiere al movimiento de cerrar y abrir rápidamente los párpados. La frecuencia y duración de los parpadeos pueden variar cuando una persona se encuentra cansada o somnolienta.

- **Inner Brow Raise (Levantamiento interno de cejas) [AU1]**

Movimiento implica el levantamiento del músculo frontal medial, lo que produce una elevación de las cejas internas.

- **Lip corner depressor (Depresión de las comisuras de los labios) [AU15]**

Se refiere al movimiento de descenso de las comisuras de los labios, lo que puede dar lugar a una expresión de tristeza o cansancio, cuando las comisuras de la boca se tiran hacia abajo.

3.3.6. Selección de manifestaciones. Toda la información necesaria para la extracción de las manifestaciones de fatiga del rostro debe capturarse usando técnicas de visión por computadora, esta rama de la inteligencia artificial tiene como propósito interpretar características de una imagen, como la forma y el color, con el objetivo de emular el reconocimiento que logran los seres humanos a través de los ojos [5], estos métodos dependen en gran medida de tener entornos parcialmente controlados, debido a que variables como la iluminación, inclinaciones de cámara y temperatura del ambiente, afectan su desempeño.

Debido a estas limitaciones, algunas señales fisiológicas mencionadas previamente no se pueden abordar en esta investigación. Los cambios de posición del dispositivo pueden llevar a un análisis incorrecto de los movimientos repetitivos de la cabeza (cabeceo), de la misma forma, la coloración de las mejillas puede verse afectada por la temperatura y la luz del lugar, por lo que se requiere de tecnologías más especializadas, como cámaras infrarrojas para capturar imágenes térmicas. Con respecto a indicadores como los movimientos de cejas, labios y arrugas en la nariz, presentan un gran reto debido a su sutileza y su seguimiento requiere de un coste computacional alto, otra limitación importante a tener en cuenta.

Después de un análisis para determinar la viabilidad de cada indicador en este proyecto, se decidió enfocar los esfuerzos en los signos de parpadeo y el bostezo, representados en las figuras 1 y 2 respectivamente. La elección se basó en su capacidad de proporcionar datos fiables y consistentes sobre el estado de fatiga en condiciones prácticas, y lo más importante, ambos pueden ser analizados mediante la cámara del dispositivo.

Figura 1. Parpadeo



Fuente: Tomado de BuzzFeed [52]

Figura 2. Bostezo



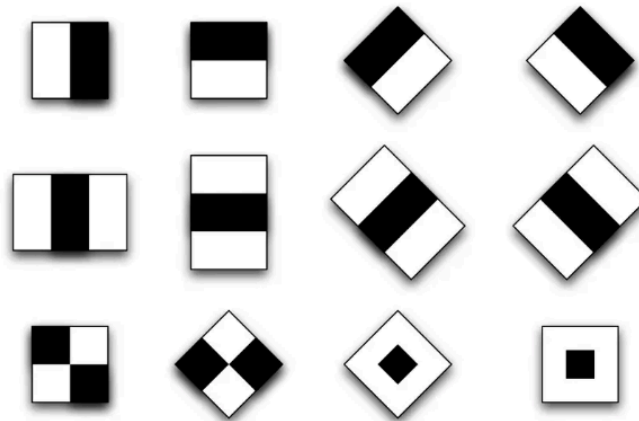
Fuente: Tomado de Shutterstock [53]

3.4. ALGORITMOS PARA DETECCIÓN DE REGIONES DE INTERÉS

Por lo general, los algoritmos empleados para la detección de diferentes expresiones faciales en una persona, usando visión artificial, se realizan en 4 fases. En la primera, se captura una secuencia de imágenes de la persona, en la segunda, se emplea un algoritmo para la detección de rostro, en la tercera, se realiza una segmentación de las regiones de interés y la última etapa, se encarga de analizar el comportamiento de las expresiones faciales que requiere el desarrollo del proyecto, para el contexto de esta investigación, el análisis de los indicadores de fatiga, de esta forma fue importante realizar una exploración de los algoritmos existentes en el ámbito de la visión por computadora para su posterior elección.

3.4.1. Algoritmo de Viola & Jones. Este, es posiblemente, el algoritmo más ampliamente difundido y reconocido para la detección de rostros fue desarrollado por P. Viola y M. Jones a partir de clasificadores **HAAR** y se destaca por su velocidad en la detección de objetos. Los clasificadores son entrenados a partir de muchas imágenes positivas y negativas, para su posterior uso, siendo una referencia icónica para la detección de rostros, debido a su gran capacidad para identificar patrones distintivos en las imágenes y tomar decisiones rápidas sobre la presencia o ausencia de un objeto. Su enfoque está basado en filtros **HAAR**, mostrados en la figura 3, y el uso de clasificadores en cascada que permiten un procesamiento eficiente y una detección precisa [34], [35], [36].

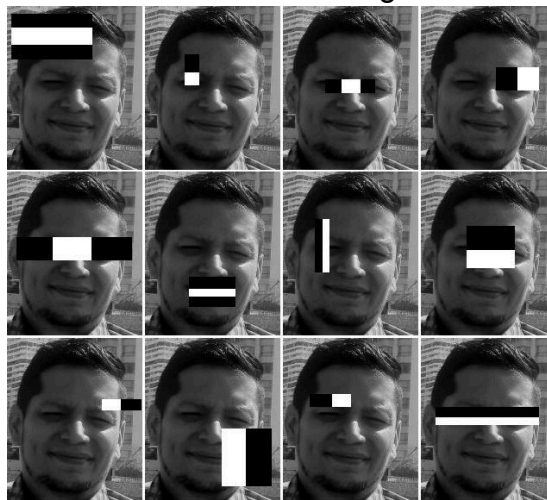
Figura 3. Filtros HAAR.



Fuente: Imagen tomada de C. J. Pardo [36]

Como se observa en la figura 4, los filtros se aplican sobre la imagen que se desea analizar y se procede a realizar cálculos sobre los píxeles afectados por dicho filtro, considerando que, aplicar en gran cantidad de filtros sobre la imagen afecta el desempeño del sistema, el algoritmo de Viola & Jones hace uso de la imagen integral y el algoritmo de **AdaBoost** combinado con los clasificadores en cascada para mejorar su velocidad de detección [34],[35],[36].

Figura 4. Aplicación de filtros HAAR sobre una imagen.



Fuente: Imagen tomada de C. J. Pardo [36].

Imagen integral:

La imagen integral se puede definir como una matriz de búsqueda en dos dimensiones, donde cada elemento contiene la suma de píxeles que se encuentran en la región superior izquierda, incluyendo el píxel de la posición indicada, reduciendo significativamente el tiempo de procesamiento computacional [34],[35],[36].

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (1)$$

$ii(x, y)$: imagen integral
 $i(x, y)$: imagen de entrada

Figura 5. Imagen integral.



Fuente: Imagen tomada de C. J. Pardo [36].

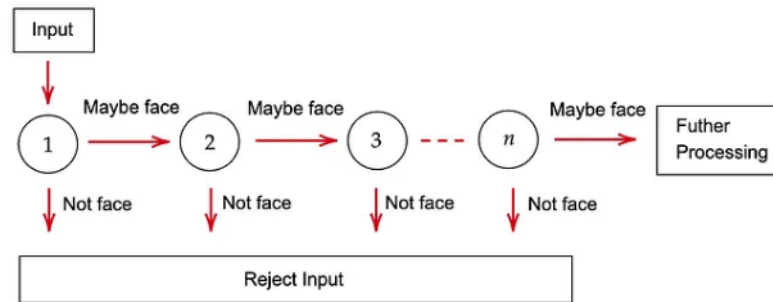
AdaBoost:

Este es el método de *boosting* que se emplea en el algoritmo creado por Viola & Jones. Su técnica consiste en establecer un clasificador de aprendizaje sencillo, posteriormente, cada característica relevante es asignada a uno de ellos, a medida que se emplea el procedimiento, las características menos útiles se eliminan dejando únicamente los que tengan las características más importantes [34], [35]. Su objetivo es encontrar un clasificador fuerte, a partir de la combinación del conjunto de clasificadores débiles, esto se logra mediante la asignación de pesos a cada uno de ellos en función de su rendimiento en la detección de ejemplos positivos y negativos durante el entrenamiento [34].

Clasificadores en cascada:

El concepto de clasificadores en cascada básicamente es un árbol de decisión, en el cual, se agrupa las características en diferentes etapas y se aplican una a una, si una ventana falla en la primera etapa, se desecha sin tener en cuenta las restantes, por el contrario, si pasa, se aplica la segunda característica y se continúa el proceso para todas las demás [34],[35],[36], en la figura 6 se ilustra su funcionamiento.

Figura 6. Diagrama de clasificadores en cascada.



Fuente: Tomado de M. Ghosh et. al [37]

3.4.2. Histograma de gradiente orientados (HOG). Es una técnica utilizada en la visión por computadora para la extracción de características de una imagen basándose en el análisis de los gradientes de intensidad, buscando información sobre la dirección y magnitud de los cambios de en diferentes regiones de la imagen [5],[38]. Para obtener un descriptor HOG, la imagen se divide en áreas rectangulares, como se ilustra en la figura 7, luego se calcula un histograma de gradientes sobre cada celda, y la combinación de todos esos gradientes constituyen el descriptor final [38].

Figura 7. Histograma de gradientes orientados.



Fuente: Imagen tomada de A. Cardona [38.]

El método de entrenamiento comúnmente empleado en este algoritmo son Máquinas de Vectores de Soporte (SVM) usando un kernel lineal con las imágenes positivas y negativas [5],[38], una vez completado este proceso, el modelo generado es usado junto al descriptor HOG para la detección de objetos en nuevas imágenes.

3.4.3. Localización de puntos del rostro (*Landmark*). Este proceso es capaz de extrapolar un conjunto de puntos clave a partir de una determinada imagen facial, estos puntos, conocidos como hitos, son aquellos que describen la forma de los atributos de la cara como: ojos, cejas, nariz, orejas, mentón, boca, etc, generando una visión general del rostro como se ilustra en la figura 8, llegando a ser de mucha utilidad para una amplia gama de aplicaciones de reconocimiento facial, animación, identificación de emociones, incluso en aplicaciones más avanzadas, como el modelado 3D de caras y el seguimiento facial [39],[40], [41]. *Face Landmark Detection* tiene una técnica que consiste en usar la diferencia de intensidad de píxeles para evaluar las posiciones de los puntos de referencia, las posiciones son reajustadas con procesos iterativos y se emplean regresiones en cascada para reducir el error lineal de los puntos estimados con cada iteración.

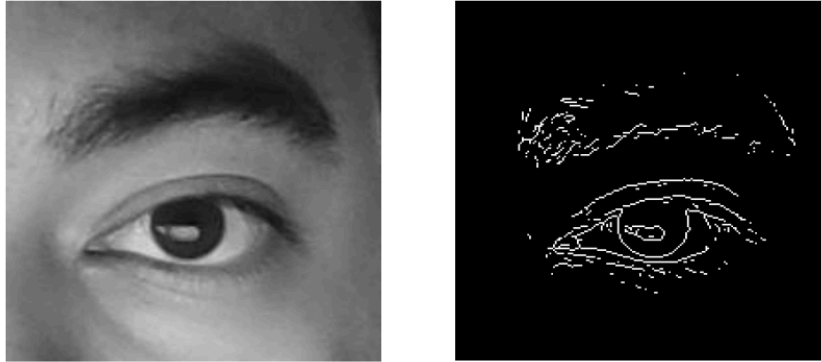
Figura 8. Representación de los 68 puntos faciales del rostro humano.



Fuente: Imagen tomada de i-bug [41]

3.4.4. Transformada de Hough circular. Es un algoritmo que permite detectar formas geométricas que puedan describirse con ecuaciones paramétricas sencillas, como rectas, parábolas, círculos, etc. [42]. Hough utiliza un algoritmo de detección de bordes, como Canny, con el objetivo de identificar y trazar los contornos de los elementos presentes en una imagen, obteniendo una representación simplificada de esta en forma su forma binaria, compuesta exclusivamente por píxeles negros y blancos como se muestra en la figura 9, esta transformación resulta sumamente útil para la detección de objetos.

Figura 9. Fotograma de ojo tras aplicar algoritmo de Canny.



Fuente: Imagen tomada de M. Antunez [42]

El contorno de la imagen no está compuesto por una línea única de píxeles, se presenta como una región completa, con un grosor irregular y una apariencia algo ruidosa en comparación con la forma paramétrica descrita por la ecuación (2).

$$(x - a)^2 + (y - b)^2 = R^2 \quad (2)$$

La transformada Hough se puede usar para determinar los parámetros de un círculo cuando se conocen varios puntos que están en el perímetro a partir de las ecuaciones (3) y (4). El objetivo del algoritmo consiste en determinar los parámetros necesarios para describir cada círculo de manera precisa (a, b, R). Cuando el ángulo θ completa una vuelta de 360 grados, los puntos (x, y) trazan el contorno del perímetro de un círculo [43].

$$x = a + R \cos(\theta) \quad (3)$$

$$y = b + R \sin(\theta) \quad (4)$$

3.4.5. Redes neuronales. En la actualidad, las redes neuronales han demostrado ser una de las tecnologías más efectivas para de detección de objetos en los últimos años, su capacidad de aprendizaje, detección de patrones y clasificación de características las ha convertido en una herramienta poderosa para aplicaciones de visión artificial, como el reconocimiento facial, por lo que muchos investigaciones recientes se han enfocado en el desarrollo de redes neuronales para la detección de rostros y otras características físicas de las personas [5]. Los trabajos realizados por Wang M. et al [44] y Cakiroglu O. et al [45] han desarrollado modelos para detección de rostros y características faciales basados en los detectores *Faster R-CNN* y *Mask R-CNN* respectivamente, demostrando un alto rendimiento y precisión en su detección.

3.4.6. Selección de algoritmos. Luego de evaluar las opciones, se optó por el algoritmo de Viola & Jones para la detección de rostro, ojos y boca, debido a que ofrece una solución adecuada en términos de velocidad y bajo coste computacional, esenciales para el desarrollo móvil. Aunque el algoritmo Landmarks cumple con estos requisitos, su implementación fue complicada debido a la falta de soporte nativo en OpenCV y Dlib para Android Studio. La necesidad de crear una interfaz JNI entre Java y C++ hizo el proceso más complejo, por lo que se decidió no usar Landmarks.

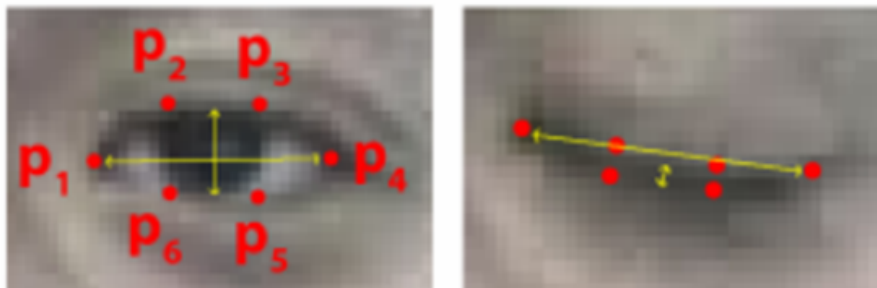
3.5. MÉTODOS PARA DETECCIÓN DE PARPADEO Y BOSTEZOS

3.5.1. Ausencia de detección (Viola & Jones). A través de este algoritmo es posible determinar si los ojos se encuentran cerrados o abiertos, utilizando 2 modelos entrenados para la detección de ojos conectados en “serie”, el primero detecta la presencia del ojo sin importar si está cerrado o abierto, mientras que el segundo verifica específicamente si el ojo está abierto o no. Este mismo enfoque puede aplicarse para detectar bostezos: primero se localiza la boca y luego se verifica si está abierta o cerrada.

3.5.2. Relación de aspecto del ojo (EAR). Este método usa un cálculo de la relación de distancia entre puntos de referencia de la forma del ojo, mostrados en la imagen de la figura 10, a partir de la ecuación (5). Generalmente se establece un umbral de esta relación para determinar si el ojo se encuentra abierto o cerrado, esto permite llevar un conteo de las veces que esa relación cae por debajo de dicho umbral, para determinar el número de parpadeos realizados por la persona [40][46].

$$EAR = \frac{kp2 - p6k + kp3 - p5k}{2kp1 - p4k} \quad (5)$$

Figura 10. Puntos de referencia para detección de parpadeo por medio de EAR.

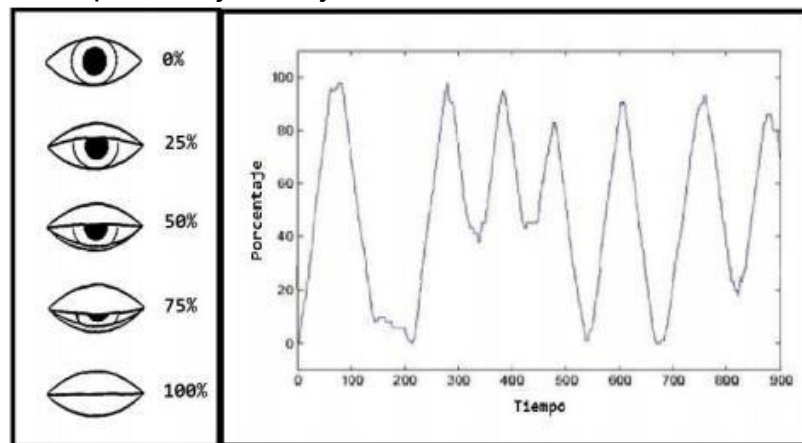


Fuente: Imagen tomada de B. Carton [40].

3.6. INDICADORES DE FATIGA

3.6.1. Porcentaje de cierre de ojo (PERCLOS). Basa su funcionamiento en la relación existente entre la fatiga y el porcentaje de la apertura del ojo de la persona que la presenta, durante un tiempo determinado como se muestra en la figura 9, un mayor nivel de fatiga representa un menor porcentaje de pupila visible [5], justo como se observa en la figura 11.

Figura 11. Niveles de porcentaje del ojo abierto.



Fuente: Imagen tomada de U. Trutschel et al [47].

3.6.2. Frecuencia del parpadeo. Según J. P. Arias [5], estimar la frecuencia de los parpadeos de un individuo, nos ofrece información valiosa sobre el nivel de su agotamiento. En condiciones normales, una persona parpadea aproximadamente 15 veces por minuto [1][5][6]. Este promedio sufre una alteración cuando se experimenta un estado de fatiga, siendo ligeramente mayor.

3.6.3. Duración del parpadeo. La duración promedio del parpadeo de un individuo en estado normal, se encuentra entre 100 y 200 milisegundos, la cual se mide, desde el momento en el cual la persona comienza a cerrar el ojo hasta que lo vuelve a abrir completamente. En contraste, este tiempo tiende a aumentar cuando la persona presenta cansancio, así lo evidencian las investigaciones como J. P. Arias [5] y P. P. Caffier. et al. [24].

3.6.4. Presencia de bostezos. Anteriormente, se mencionó que los bostezos están dentro de las manifestaciones más evidentes de cansancio en las personas. Esta correlación se basa en estudios que demuestran que el acto de bostezar está estrechamente ligado a la necesidad de aumentar la vigilia y la atención [5]. Por lo tanto, contar el número de bostezos que una persona realiza en un periodo específico se presenta como un método útil para evaluar el nivel de fatiga.

3.7. HERRAMIENTAS DE PROGRAMACIÓN

A pesar de la variedad de sistemas operativos desarrollados para los teléfonos inteligentes, sin duda alguna Android e iOS se destacan indiscutiblemente sobre los demás, siendo Android el que tiene mayor cuota en el mercado de teléfonos celulares, con una cuota de mercado del 72.15%, mientras que iOS representa el 27.19%, según cifras de un estudio realizado en junio del 2024 por Statista, un portal de estadísticas sobre el mercado mundial [48]. Esta preferencia se evidencia en su adopción generalizada por parte de diversos fabricantes de dispositivos móviles, mientras que el sistema operativo de iOS es exclusivo para dispositivos desarrollados por Apple Inc.

Android: *Android* es un sistema operativo basado en el núcleo de *Linux* de código abierto y diseñado por la empresa *Android Inc.* Se concibió principalmente para teléfonos celulares, pero su adopción se ha extendido de manera significativa a otros dispositivos, como tabletas, relojes e incluso televisores. La flexibilidad y versatilidad de *Android* han sido clave en su popularidad y amplio uso en el mercado tecnológico.

iOS: *iOS* es un sistema operativo de código cerrado desarrollado por *Apple Inc.* Originalmente creado para *iPhone*, y más tarde en dispositivos como *iPod* y *iPad*. A nivel mundial, *iOS* ocupa una sólida posición como el segundo sistema operativo más utilizado.

Una de las principales ventajas de Android en comparación con el sistema operativo de Apple radica en su naturaleza de código abierto, lo que permite un mayor grado de libertad en la creación de aplicaciones. Por tal razón, se ha determinado que el aplicativo móvil se desarrolle en el sistema operativo Android, aprovechando su flexibilidad y recursos que ofrece [8].

Por otro lado, el entorno de desarrollo integrado (IDE) más avanzado, para la elaboración de aplicaciones compatibles con sistemas *Android* es, *Android Studio*, dicho software, proporciona un editor de código, emuladores de dispositivos, depuración y análisis de rendimiento. Además, es el recurso oficial de Google y cuenta con una gran comunidad activa de programadores, lo que garantiza una amplia cantidad de documentación, permitiendo una experiencia más fluida y eficiente, para el desarrollo de este aplicativo.

Los entornos de desarrollo no ofrecen por sí solos herramientas de visión artificial necesarias para llevar a cabo el proyecto, por lo que se debe hacer uso de paquetes especializados en esta área. Las más destacables hasta el momento son OpenCV (*Open Source Computer Vision Library*) y Dlib.

3.7.1. Opencv. Esta es una librería de código abierto ampliamente utilizada en el campo de la visión artificial fue desarrollada originalmente por Intel y abarca una amplia gama de funciones y algoritmos para el procesamiento de imágenes y el análisis por computadora como detección y reconocimiento de objetos, seguimiento de objetos en tiempo real, calibración de cámaras, estimación de poses, segmentación de imágenes, entre otros [49], siendo compatible con lenguajes de programación como: C++, Python y Java.

3.7.2. Dlib. Es una biblioteca de software de código abierto que se utiliza en el área de aprendizaje autónomo y tratamiento de imágenes. Fue desarrollada principalmente por Davis King y ofrece una amplia gama de módulos con los cuales, se pueden aplicar técnicas avanzadas para identificación de patrones y objetos, seguimiento de puntos de interés, la estimación de pose y el análisis de características faciales [50].

3.8. BASE DE DATOS

“*YawDD*” es una colección de videos disponible en la plataforma de IEEE, grabados por una cámara dentro de un automóvil, con diversas características faciales de hombres y mujeres, y mostrando diversas expresiones: hablando, en silencio, bostezando, etc. Esta base de datos es útil para técnicas que involucren la detección de rostros, bocas y bostezos [44].

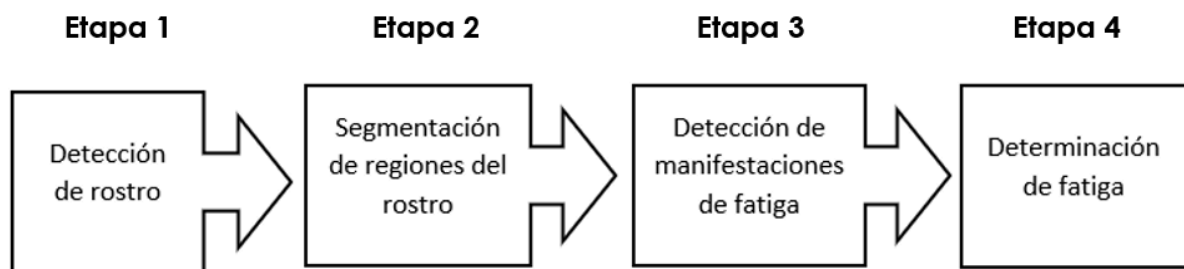
“*UTA-RLDD*” es una base de datos creada en la universidad de Texas en Arlington, con el objetivo de detectar somnolencia en varias etapas. Tiene un total de 180 videos RGB

de 60 participantes en diferentes estados: alerta, baja vigilancia y somnolencia, entre los participantes voluntarios para su creación, había 51 hombres y 9 mujeres de diferentes etnias, edades y facciones [51].

4. SISTEMA PARA DETECCIÓN DE FATIGA

El sistema para la detección de fatiga diseñado en este proyecto, se compone de 4 etapas, luego de obtener las imágenes capturadas por la cámara. Dichas etapas se realizan de forma consecutiva y cumpliendo una procedimiento específico cada una como se ilustra en la figura 12, esta estructura permite dar una visión clara y precisa para la implementación de algoritmos.

Figura 12. Etapas de sistema para detección de fatiga.

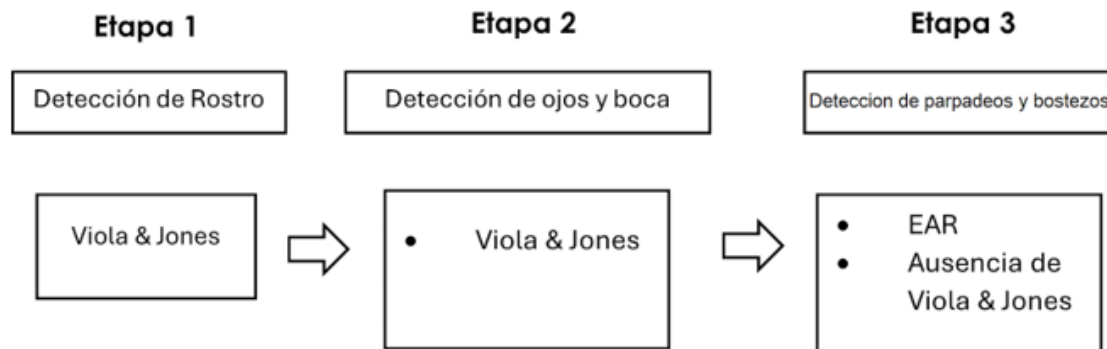


Fuente: Elaboración Propia.

4.1. ALGORITMOS PARA DETECCIÓN DE MANIFESTACIONES DE FATIGA

El diagrama de bloques mostrado en la figura 13 presenta los algoritmos que se consideran para su implementación en las primeras tres etapas del sistema. Dado que la determinación de fatiga (etapa 4) es independiente de los algoritmos utilizados en las etapas anteriores, se decidió implementar cada algoritmo en cada etapa para seleccionar el que ofreciera los mejores resultados.

Figura 13. Algoritmos para cada etapa para la detección de manifestaciones de fatiga



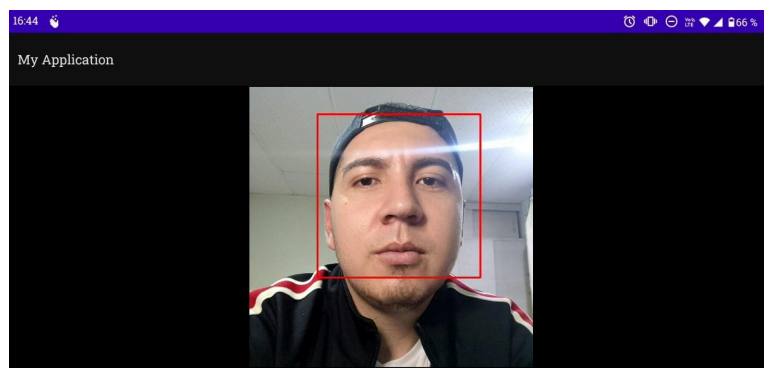
Fuente: Elaboración propia.

4.1.1. Detección de rostro. La primera etapa del sistema se encarga de realizar la detección del rostro del usuario, donde es crucial que el procesamiento sea rápido, en este sentido, se optó por usar el algoritmo desarrollado por P. Viola y M. Jones, considerando que tiene un coste computacional relativamente bajo [5]. Como se mencionó anteriormente, este algoritmo usa una serie de clasificadores aplicados secuencialmente a las imágenes para detectar la presencia de un objeto específico, en este caso el rostro, dichos clasificadores, utilizan patrones que permiten distinguir áreas oscuras y claras de la imagen y se utilizan para crear ventanas que escanean la imagen en diferentes escalas. Este enfoque en cascada hace que el algoritmo sea muy eficiente porque elimina rápidamente las regiones de la imagen que no tienen rostros y se concentra en las áreas prometedoras.

Para implementar esta técnica, se utilizó la librería OpenCV integrada en Android Studio. Específicamente, se empleó el archivo "*haarcascade_frontalface_alt.xml*", que es un modelo pre-entrenado para la detección de rostros frontales. Se aplicó a cada fotograma capturado por la cámara frontal del dispositivo utilizando la función ***detectMultiScale***, esta función toma la imagen de entrada y devuelve un vector de rectángulos que representan el objeto detectado, adicionalmente, los parámetros de esta función, permite ajustar el comportamiento de la detección. El tamaño mínimo del rostro detectado, establecido en 270 x 270 píxeles y, el factor de escala fijado en 1.4, permitió reducir el número de iteraciones del proceso, mejorando la rapidez de la aplicación, esto debido a que el rostro ocupa la mayor parte de la imagen, no es necesario que el algoritmo trate de identificar rostros muy pequeños.

Por último se dibujó un rectángulo alrededor del rostro utilizando las funciones de dibujo de OpenCV, la imagen de la figura 14 muestra el resultado obtenido.

Figura 14. Detección de rostro usando clasificador HAAR.



Fuente: Elaboración Propia.

4.1.2. Detección de Ojos y Boca. Si bien Viola & Jones proporciona un modelo pre-entrenado para detectar ojos, no se disponía de uno similar para la boca, y tras explorar varios repositorios en línea, no se encontró una solución que cumpliera con los requisitos esperados. Afortunadamente, OpenCV también ofrece una herramienta, la cual permite crear clasificadores personalizados, basados en características Haar o características LBP (*Local Binary Patterns*), este recurso exige un conjunto de datos compuesto por muestras positivas (imágenes que contienen el objeto que se pretende detectar) y negativas (imágenes en las que no se encuentra dicho objeto). En el proceso de entrenamiento se deben ajustar diferentes parámetros, como el tamaño de las imágenes, el tipo de clasificadores a utilizar, el número de etapas de aprendizaje y los umbrales que permiten optimizar la tasa de detección y reducir la tasa de falsos positivos. La tabla 1 presenta los parámetros usados, obtenidos luego de realizar varios procesos de entrenamiento y seleccionando el clasificador que presentó la mejor tasa de detección.

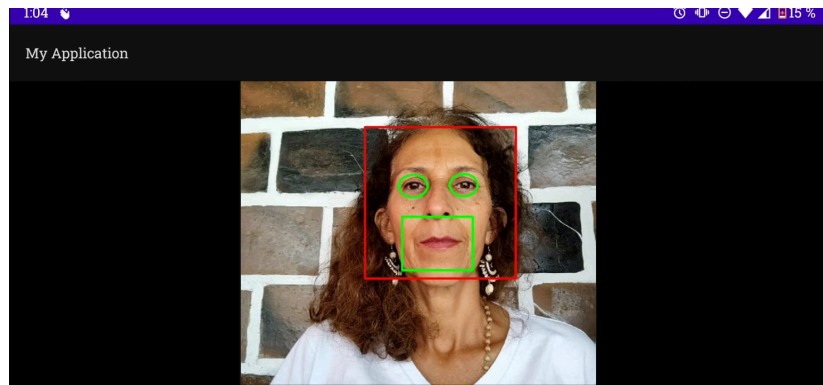
Tabla 1. Parámetros para entrenamiento del clasificador HAAR (detección de boca).

Parámetro	Configuración
Número de etapas	15
Dimensiones de las imágenes	320x240 pixels
Tipo de característica	HAAR
Tasa mínima de aciertos	0.995
Tasa máxima de falsos positivos	0.15

Fuente:Elaboración Propia.

Es importante tener en cuenta que para la creación de un modelo se requiere una considerable cantidad de tiempo y una capacidad de procesamiento bastante significativa. Debido a estas exigencias, se utilizó un código en Python para generar un archivo XML que contiene los parámetros del modelo entrenado para la detección de la boca. Este archivo es empleado por la aplicación de Android para realizar dicha detección. En la figura 15, se muestra la implementación de estos modelos en conjunto.

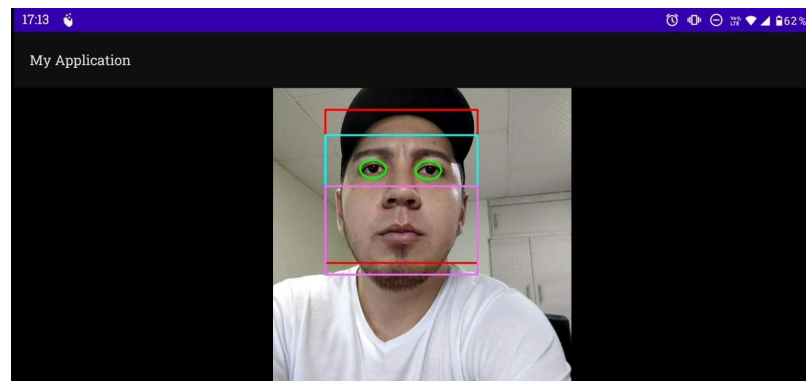
Figura 15. Detección de rostros, ojos y boca, usando clasificadores en cascada



Fuente: Elaboración Propia.

Inicialmente, el uso de los clasificadores para la detección de ojos y boca se realizó en toda la región del rostro, lo que resultó en un procesamiento lento (aproximadamente 10 FPS), generando una experiencia poco fluida para el usuario. Por esta razón, se decidió dividir la zona del rostro en dos partes: el clasificador de ojos aplicado a la parte superior, y el clasificador de boca a la parte inferior, como se ilustra en la figura 16. reducir las iteraciones del algoritmo, aumentando el procesamiento a 24 FPS.

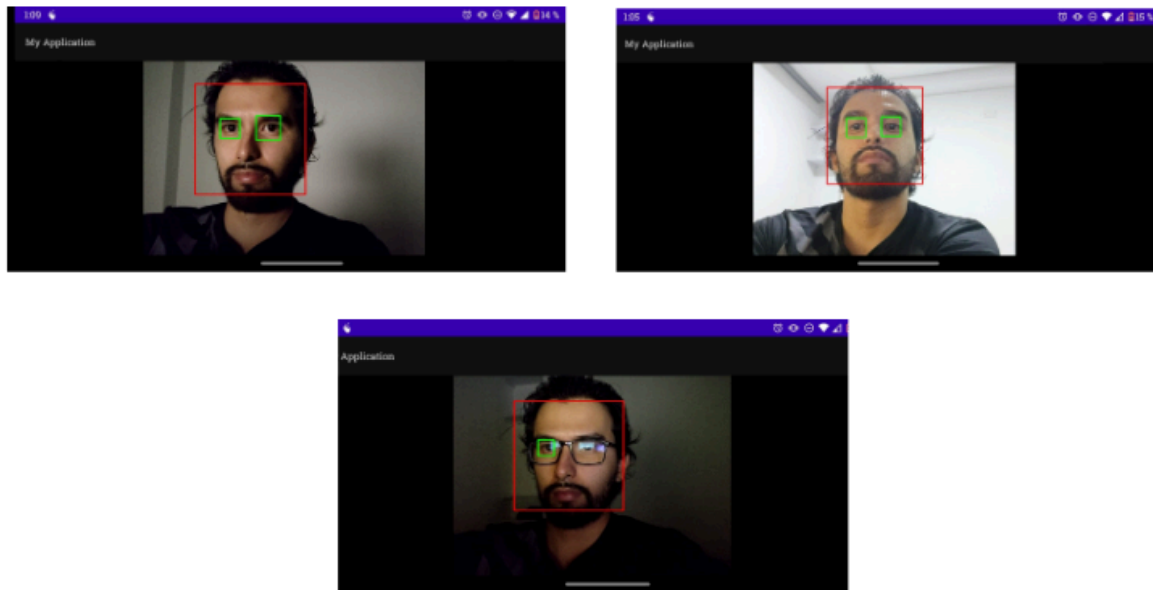
Figura 16. División del rostro para detección de ojos (recuadro azul) y boca (recuadro violeta).



Fuente: Elaboración Propia.

El algoritmo de Viola & Jones demostró ser capaz de llevar a cabo la detección de una forma bastante precisa a pesar de lo que se creía, incluso en presencia de pequeñas variaciones en la iluminación, así como en situaciones en las que la cámara se encuentra ligeramente inclinada, no obstante, presenta limitaciones cuando se enfrenta a obstáculos visuales, como el uso de gafas, tal como se ilustra en la figura 17.

Figura 17. Detección con variaciones de iluminación, inclinación de cámara y obstáculos visuales, usando Viola & Jones.



Fuente: Elaboración Propia.

4.1.3. Detección de parpadeos y bostezos. Tras realizar la detección y segmentación de las regiones de interés, se procedió a implementar la detección de parpadeos utilizando la estrategia de “Detección por ausencia de Viola & Jones”.

La estrategia de EAR arrojó problemas al momento de implementarlo debido al funcionamiento del clasificador pre-entrenado de OpenCV para la detección de ojos, el cual está diseñado exclusivamente para reconocer ojos abiertos. Esto significa que, cuando un usuario parpadea y sus ojos se cierran, el sistema no puede detectarlos como tales, como resultado, se pierden los datos necesarios para calcular EAR, y en consecuencia, la detección efectiva de parpadeo.

- **Detección de parpadeos y bostezos usando ausencia de Viola & Jones.**

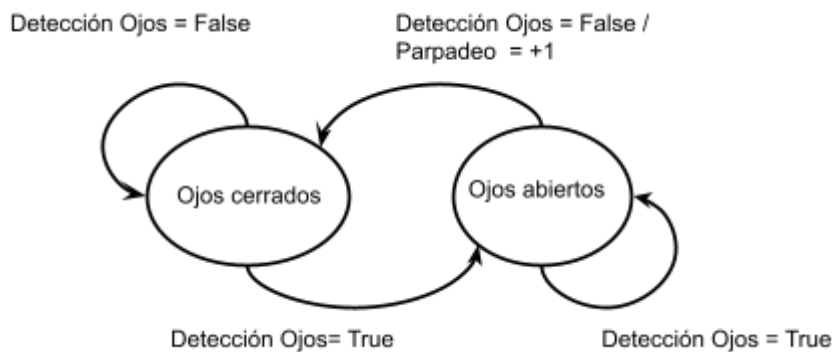
La detección de parpadeos mediante esta técnica resultó ser un proceso sencillo de implementar, bajo las condiciones que presenta el clasificador usado para detección de ojos, el cual detecta únicamente ojos abiertos, permitiendo contabilizar las ocasiones en que el sistema deja de detectar los ojos como parpadeos realizados por la persona.

Para poder aplicar este método se usaron dos variables booleanas, y un contador de parpadeos. La primera variable almacena el estado de los ojos en el fotograma previo (true: abiertos, false: cerrados), la segunda variable se define dependiendo de la detección de los ojos en el fotograma en proceso (true: ojos detectados, false: ojos no detectados).

Si en el fotograma procesado en tiempo real no se detectan ojos y adicionalmente, el estado anterior hizo la detección, indica que la persona ha cerrado los ojos y por lo tanto, el contador de parpadeos incrementa en 1, la figura 18 ilustra de forma básica el mecanismo implementado para contabilizar los parpadeos.

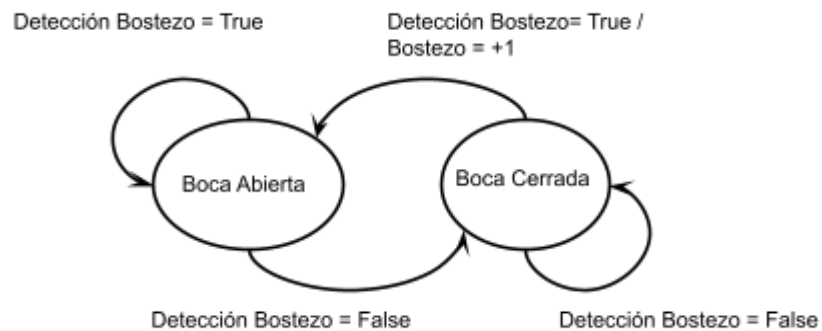
Observando que, el clasificador usado para la detección de ojos tiene facilidad para implementar este sistema, se procedió a generar un nuevo clasificador en cascada, el cual se enfoca en la detección directa de los bostezos, tomando imágenes de estos como muestras positivas e imágenes de fondos como muestras negativas. Por lo tanto, la contabilización de bostezos se logra solamente cuando el clasificador identifica el bostezo en los fotogramas procesados figura 19.

Figura 18. Conteo de parpadeos a partir de ausencia de detección (Viola & Jones).



Fuente: Elaboración Propia

Figura 19. Conteo de bostezos a partir de ausencia de detección (Viola & Jones).



Fuente: Elaboración Propia.

Previo al desarrollo de las pruebas con este algoritmo, y tomando en consideración las dificultades generadas debido al funcionamiento del clasificador pre-entrenado para detección de ojos, se optó por aprovechar esta desventaja como una oportunidad para llevar a cabo la detección por medio de ausencia de Viola & Jones. Para ello, se procedió al entrenamiento de un nuevo clasificador capaz de detectar directamente los bostezos (boca abierta), cabe aclarar que, para que esta estrategia funcione correctamente fue necesario diseñar un contador de bostezos que aumenta, solamente cuando el tiempo de detección de boca abierta sea mayor a 5 segundos (promedio de duración de un bostezo). Las tablas 2 y 3 muestran los resultados obtenidos con esta técnica.

Tabla 2. Resultados de la detección de parpadeos por ausencia de Viola & Jones.

	PARPADEOS REALIZADOS	PARPADEOS DETECTADOS	FALSOS POSITIVOS
Sujeto 1	98	94	5
Sujeto 2	101	98	4
Sujeto 3	115	110	3

Fuente:Elaboración Propia.

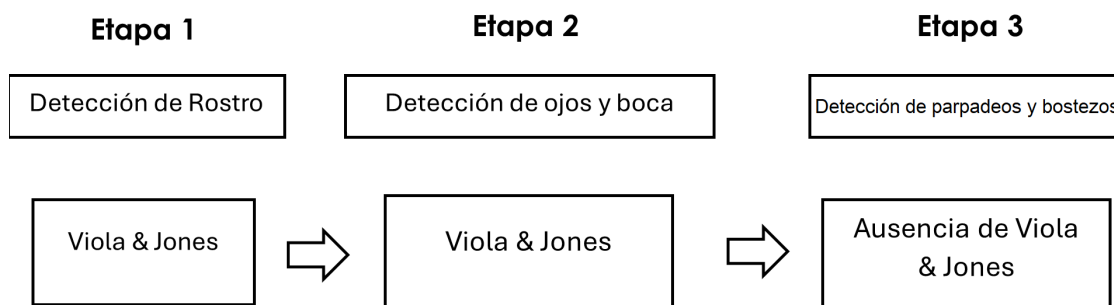
Tabla 3. Resultados de la detección de bostezos por ausencia de Viola & Jones.

	BOSTEZOS REALIZADOS	BOSTEZOS DETECTADOS	FALSOS POSITIVOS
Sujeto 1	8	4	5
Sujeto 2	7	3	2
Sujeto 3	6	4	3

Fuente:Elaboración Propia.

Los resultados muestran que este algoritmo cumple las exigencias del sistema, por lo que se procede a usar esta estrategia para la detección de parpadeos y bostezos e implementar los algoritmos presentados en la figura 20 al aplicativo móvil.

Figura 20. Etapas del sistema para la detección de manifestaciones de fatiga.



Fuente: Elaboración Propia..

4.2. ESTRATEGIA PARA LA DETECCIÓN DE FATIGA

Es necesario señalar que la fatiga es un fenómeno multidimensional y puede manifestarse de diferentes formas en cada persona, por lo que usar solamente uno de los indicadores descritos en la sección 3.6, no es suficiente para dar un veredicto certero del estado de la persona, por esto, se requiere su combinación. Se pretende hacer uso de todos, exceptuando a PERCLOS, ya que, para usarlo efectivamente se necesita del cálculo de EAR para establecer el porcentaje abierto del ojo, por otro lado, de acuerdo con J. P. Arias, “este criterio no es muy confiable, pues su desempeño comparado con otros métodos es muy bajo” [5], así también lo demuestra el estudio realizado por U. Trutschel et. al [47].

Se realizó la monitorización continua de tres señales clave: la frecuencia de parpadeos, la duración de los parpadeos y la presencia de bostezos. Cada uno de estos indicadores fue registrado y analizado en intervalos de 1 minuto y se decidió representar cada indicador con variables booleanas para facilitar la estrategia de detección de fatiga según las siguientes pautas:

- Frecuencia de parpadeo: Si la tasa de parpadeo es mayor al promedio de los parpadeos que realiza una persona cuando esta no presenta síntomas de fatiga (15) en un lapso de 1 minuto, este indicador se establece como “verdadero”, de lo contrario se marca como “falso”.

- Duración de parpadeo: Si el número de parpadeos que presentan una duración mayor a 200 milisegundos es mayor al 50% del total de parpadeos realizados en un lapso de 1 minuto, esta variable se fija como “verdadero”, caso contrario se establece como “falso”.
- Presencia de Bostezos: Si existe al menos un bostezo detectado por el sistema en un lapso de 1 minuto, esta señal se define como “verdadero”, caso contrario será “falso”

Es importante mencionar que los valores de estos criterios, se establecieron basándose en las investigaciones antecedentes.

Por otra parte, se definieron tres estados posibles del usuario los cuales se activan de acuerdo a los siguientes lineamientos:

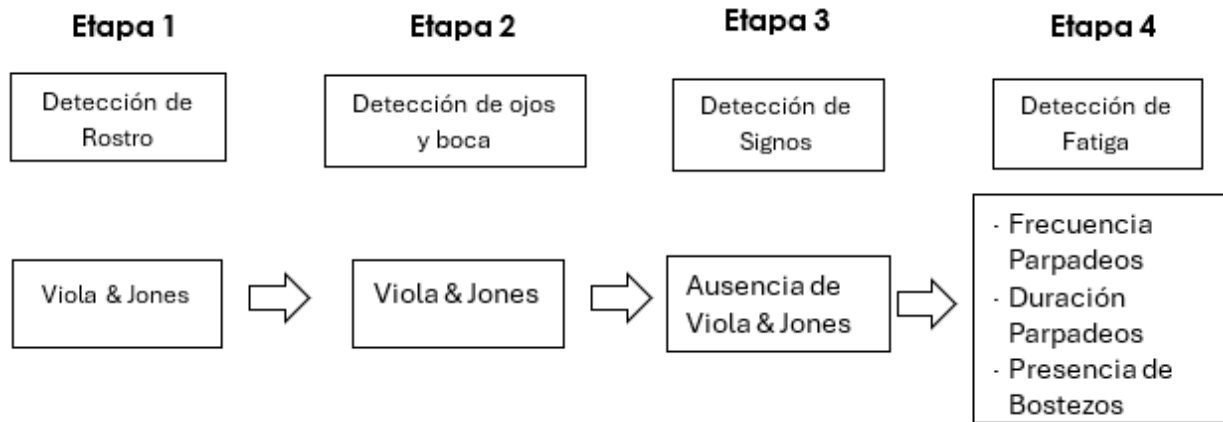
- SIN FATIGA: No se presentan irregularidades en los indicadores de fatiga.
- POSIBLE FATIGA: El sistema detecta anomalías en uno o dos indicadores.
- FATIGA: Se presenta cuando el sistema identifica discrepancias en los tres indicadores de fatiga.

En el anexo A, se presenta el diagrama del algoritmo implementado para la detección de fatiga.

4.3. INTEGRACIÓN DEL SISTEMA FINAL PARA LA DETECCIÓN DE FATIGA

La estrategia detallada en la sección 4.2, para el reconocimiento de la fatiga (etapa 4), se integra al sistema de detección de manifestaciones de fatiga obtenido en la sección 4.1, dando lugar así al sistema completo que se ilustra en la figura 21.

Figura 21. Sistema final para la detección de fatiga en el rostro humano.

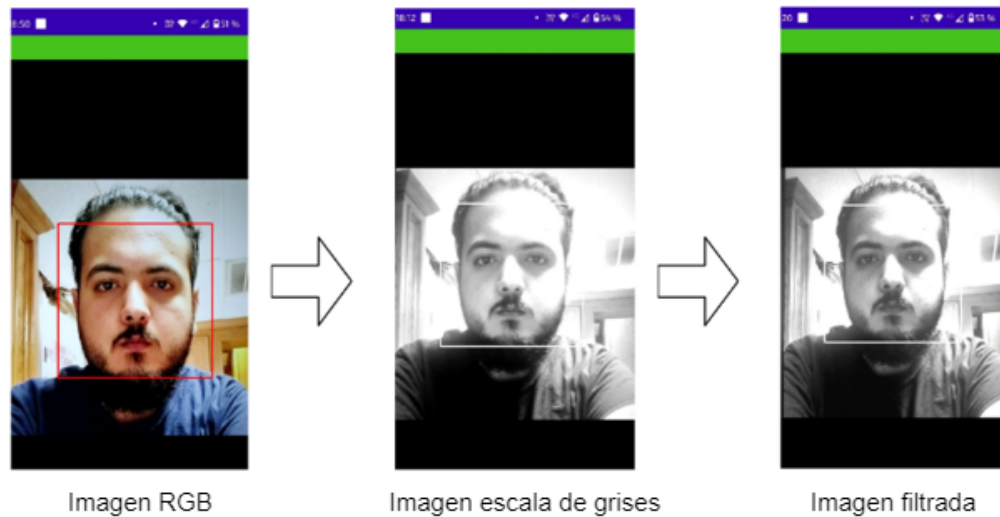


Fuente: Elaboración Propia.

4.4. PRE-PROCESAMIENTO DE IMÁGENES

Durante la implementación de los algoritmos para detección de rostro, ojos y boca, cada fotograma entregado por la cámara del dispositivo móvil fue convertido a escala de grises, para reducir el peso y facilitar su posterior procesamiento. Sin embargo, en aras de mejorar el sistema, se llevó a cabo un proceso experimental, que exploró diferentes técnicas de pre-procesamiento de imágenes ofrecidas por Android Studio y OpenCV, las cuales contribuyeron a mejorar la detección y aumentar su robustez. En primer lugar, se aplicó un filtro gaussiano en las imágenes, para realizar un suavizado y eliminar el ruido presente, obteniendo más nitidez de los rasgos faciales, figura 22. Luego, se implementó una ecualización de histograma, este proceso ajusta la distribución de intensidades de píxeles, lo cual implica calcular la función de transformación que mapea los valores de intensidad de entrada a nuevos valores de intensidad de salida, refinando su brillo y contraste, esto resultó particularmente beneficioso en condiciones de iluminación desigual, como ilustra la figura 23.

Figura 22. Procesado de imagen inicial, aplicando escala de grises y filtro.



Fuente: Elaboración propia.

Figura 23. Aplicación de ecualización de histograma a la imagen filtrada.



Fuente: Elaboración Propia.

Utilizando estas técnicas, se observó una mejora significativa en la detección, especialmente en la de los ojos, durante situaciones en donde la iluminación del rostro no era uniforme. A pesar de ello, el sistema aún presenta inestabilidades ante variaciones extremas de luz y movimientos del dispositivo móvil.

Para mitigar esto, se intentó implementar más estrategias, pese a ello, dado la variedad de entornos no controlados, resultó imposible que se ajustaran a cada uno de estos escenarios de forma satisfactoria. Además, el aumento en el procesamiento de la imagen afectó negativamente el rendimiento de la aplicación. Por ende, fue necesario diseñar un protocolo de pruebas en un ambiente semicontrolado para analizar el comportamiento ante diferentes tipos de iluminación, fondos y ángulos, con el objetivo de identificar sus limitaciones y determinar las condiciones del mejor desempeño el cual se describe en la sección 4.6 de este apartado.

4.5. DISEÑO DE LA INTERFAZ DE LA APLICACIÓN.

La interfaz de la aplicación, mostrada en la figura 24, fue diseñada con la intención de presentar al usuario los datos recopilados por el sistema de detección como el tiempo transcurrido de detección, números de parpadeos y bostezos, etc. Es bastante intuitiva y fácil de usar, contiene botones para controlar la detección de fatiga y activar los destellos de luz que simulan la fatiga, en la parte superior se presenta el resultado obtenido por el sistema, así como contadores de parpadeos y bostezos.

Figura 24. Descripción de la interfaz de la aplicación móvil.



Fuente: Elaboración propia.

La programación realizada en el aplicativo móvil consta de cinco clases diseñadas cuidadosamente para llevar a cabo la detección de fatiga. Esta estructura, refleja los principios de una programación orientada a objetos, la cual ofrece múltiples beneficios, como la modularidad y organización del código, su encapsulamiento, facilidad de mantenimiento, reutilización del código, interpretación de funcionalidades, etc.

Asimismo, se utilizaron casos de uso real para guiar el desarrollo del aplicativo, los cuales describen cómo los usuarios interactúan con la aplicación y el sistema desarrollado, especificando las acciones de cada actor que intervenga en diversos escenarios. Para este proyecto, los casos de uso real, descritos en el apartado de anexos, incluyen la detección de parpadeos y bostezos, la simulación de estados de fatiga, y la emisión de alertas, asegurando que todas las funcionalidades requeridas estén implementadas y que el sistema responda adecuadamente a las necesidades del usuario.

MainActivity:

La clase principal de esta aplicación se encarga de gestionar la interfaz del usuario y la inicialización de los periféricos que el dispositivo utiliza, como la cámara y el sensor de luz, implementando métodos auxiliares para solicitar permisos de configuración, además, se encarga de realizar la conexión entre las clases que se encargan de realizar la detección y la determinación de fatiga.

CascadeClassifierLoader:

Esta clase se encarga de cargar los clasificadores Haar desde los recursos de la aplicación y almacenarlos temporalmente en el dispositivo, dichos clasificadores son archivos XML, los cuales contienen la información necesaria para la detección rostro, ojos y bostezos, "*haarcascade_frontalface_alt.xml*", "*haarcascade_eye.xml*" y "*haarcascade_yawn.xml*", respectivamente.

Detection:

Esta clase personalizada implementa la interfaz '*CvCameraViewListener2*' de OpenCV, dicha interfaz proporciona el método callback '*onCameraFrame*' el cual se ejecuta cada vez que la cámara captura un fotograma nuevo, el cual es procesado uno a uno para lograr la detección precisa, tanto del rostro como de los ojos y la boca. Esta clase es el núcleo de la aplicación y se apoya principalmente en cuatro métodos esenciales para realizar su funcionalidad:

- *'processsingImagenFunction(Mat)'*: recibe la imagen en formato Mat de la cámara y se encarga del procesamiento de las imágenes, conversión a escala de grises, filtrado, ajuste de brillo y contraste aplicando ecualización de histograma para mejorar la imagen antes de aplicar el algoritmo de detección sobre ella.
- *'blinkDetectFunction(MatOfRect)'*: este método recibe una matriz de cuadros delimitadores de la región donde se encuentran los ojos del usuario y se encarga de detectar y contar los parpadeos que ha realizado la persona.
- *'yawnDetectFunction(MatOfRect)'*: recibe una matriz de recuadros que contiene la región de la boca y al igual que el método anterior, se encarga de detectar la boca abierta y contabilizar los bostezos que realiza el usuario.
- *'fatigaDetectionFunction(int)'*: está encargado de determinar si la persona presenta fatiga o no, basándose en el conteo de la frecuencia de los parpadeos, su duración y la presencia de bostezos.

StateSimulation:

Esta clase se encarga de activar indicadores de luz “destellos” en la pantalla del dispositivo, de forma aleatoria para simular los parpadeos y bostezos de una persona en estado normal y en estado de fatiga. Su funcionamiento depende del *checkbox* “fatiga”. El estado normal (no fatiga) activa solamente el indicador de parpadeos generando destellos de color amarillo que indican parpadeos de duración “normal” (duracion menor a 200 ms), por otro lado, el estado fatiga, activa el indicador de bostezos de color verde y el indicador de parpadeos con destellos amarillos y rojos (parpadeos con duracion mayor a 200 ms) de forma aleatoria.

Time: Esta clase utilitaria, ayuda a medir el tiempo transcurrido entre eventos, como control del tiempo de detección de las manifestaciones de fatiga.

4.6. PROTOCOLO DE PRUEBAS

El protocolo de pruebas se diseñó específicamente para establecer los rangos de variación en iluminación, inclinación y cambio de fondos que el sistema puede soportar sin comprometer la detección de características faciales clave, como el rostro, los ojos y la boca, esenciales para determinar signos de fatiga. Este enfoque se centra en crear condiciones controladas que simulan específicamente entornos habituales, como oficinas o habitaciones del hogar. En estos espacios, los niveles de iluminación suelen variar entre 50 y 400 lux [54]. En contraste, condiciones más extremas, como la luz solar directa en exteriores, pueden superar los 10000 lux, lo que dificulta la detección. Por lo tanto, la aplicación funciona en un rango del 2% al 5% en comparación con esta amplia gama de iluminación a la que podría estar expuesto el sistema, así como a la ausencia total de iluminación.

4.6.1. Variaciones de iluminación. Para gestionar eficazmente la iluminación del entorno, se empleó el sensor de luz integrado en el dispositivo móvil. La mayoría de los smartphones actuales incluyen este sensor, que evalúa la intensidad de la luz ambiente y emite un valor correspondiente. En áreas oscuras, los valores son bajos, pero aumentan en ambientes bien iluminados. Android Studio proporciona una medición en un rango de 0 a 5000 lux, aunque puede variar según el dispositivo específico.

Es relevante mencionar que el sensor tiene un ángulo de visión limitado de aproximadamente 100 a 120 grados, lo que puede presentar dificultades según la ubicación de la fuente de luz con respecto al sensor, el cual suele estar cerca de la cámara frontal del dispositivo. Para abordar mejor esta situación, se recurrió al esquema de iluminación de tres puntos (principal, fondo y relleno) que se plantea en el trabajo de J. P. Arias [5], con el fin de obtener una condición semi controlada.

El esquema considera tres fuentes fundamentales de iluminación, una luz principal que ilumina de forma frontal el rostro de la persona y debe ser la fuente más fuerte de iluminación en el ambiente, una luz de relleno, la cual suaviza las sombras generadas por la anterior y formando un ángulo de 180° con respecto a la anterior, ubicándose en la parte inferior y al frente del sujeto y por último una luz de fondo que aumenta el contraste, la cual debe estar a un ángulo de 180° con respecto a la principal, en la parte superior y detrás del sujeto como se ilustra en la figura 25.

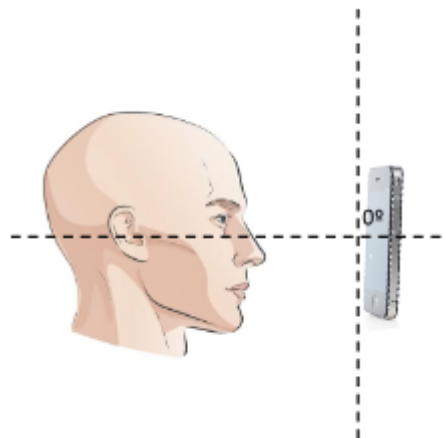
Figura 25. Esquema de iluminación de tres puntos.



Fuente: Tomado de J. P. Arias [5].

El entorno escogido para esta prueba fue una habitación pequeña con dimensiones de 3 metros de ancho por 4 de largo y muy bien iluminada. El celular se ajustó en una base, para evitar movimientos de éste, en un ángulo de 0 grados de inclinación tanto en el eje horizontal como en el eje vertical con respecto al rostro frontal del usuario, tal y como se aprecia en la figura 26. Adicionalmente se advirtió a los sujetos de prueba, evitar movimientos bruscos y mantener una distancia de 25 a 35 cm con el dispositivo móvil.

Figura 26. Posición del celular sin inclinaciones con respecto al rostro.



Fuente: Elaboración Propia

Las variaciones de las fuentes de luz se hicieron de la siguiente manera:

Se utilizó una lámpara LED regulable como fuente de luz principal, por lo que se ubicó al frente de los sujetos de prueba, lo que permite una iluminación directa y uniforme de sus rostros. Genera diferentes niveles de intensidad de luz, en un rango aproximado de 100 a 1200 lúmenes. Por otro lado, para las fuentes de luz de fondo y relleno se usaron lámparas LED de 12 watts de potencia que producen alrededor de 400 lúmenes cada una.

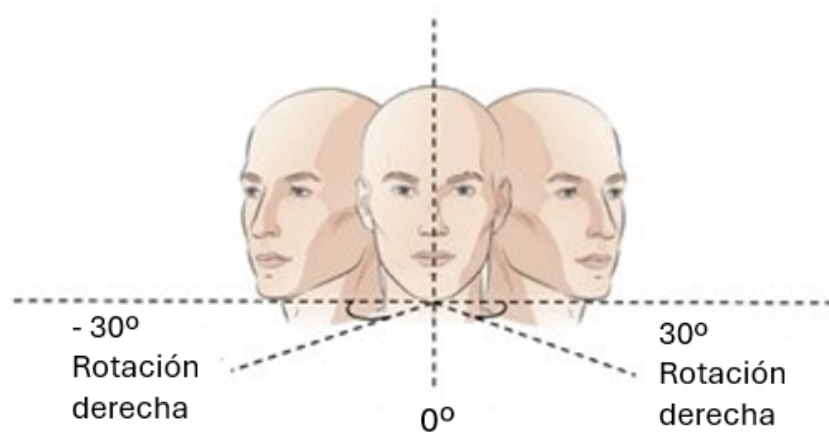
Con este escenario se procedió a realizar una variación de la luz principal desde su máxima intensidad hasta la mínima. Posteriormente se intercambia la posición de esta lámpara y las fuentes de luz de fondo y relleno, permitiendo realizar variaciones controladas en cada fuente de luz por separado y ver el comportamiento de la aplicación bajo una amplia gama de escenarios.

4.6.2. Variaciones de inclinación de cámara. Las inclinaciones del móvil generan una variedad de ángulos entre el rostro del usuario y la cámara del dispositivo, complicando la detección de sus regiones, por lo que es importante realizar pruebas para delimitar en qué ángulos el sistema es capaz de detectar los ojos y la boca de los usuarios.

Considerando la posición inicial como 0 grados de inclinación, tanto vertical como horizontalmente, detallada en la figura 26, se realizaron variaciones de 10 grados en un rango de 60° en un plano vertical y horizontal, generando diferentes ángulos de captura, las figuras 26, 27, y 28 logran mostrar de una manera más clara la descripción de estas inclinaciones en cada plano.

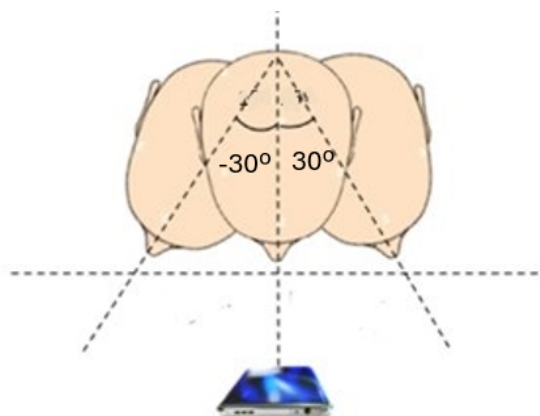
Los resultados obtenidos bajo las pruebas realizadas en este protocolo de pruebas se muestran en el apartado 5.

Figura 27. Rotación del dispositivo en plano horizontal (vista frontal).



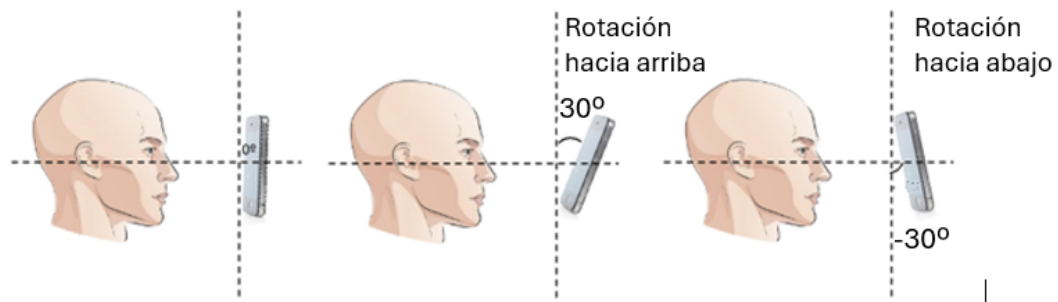
Fuente: Elaboración Propia.

Figura 28. Rotación del rostro con respecto al dispositivo en el plano horizontal (vista desde arriba)



Fuente: Elaboración Propia.

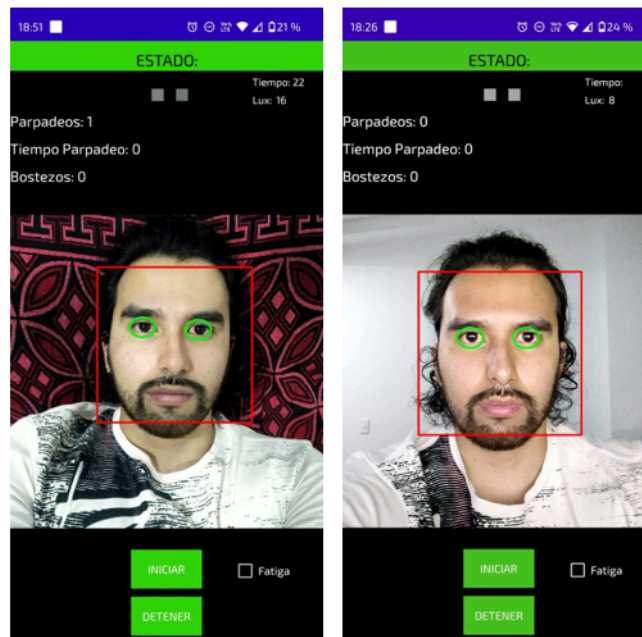
Figura 29. Rotación del rostro en plano vertical con respecto al dispositivo.



Fuente: Elaboración propia.

4.6.3. Variaciones de fondo. Si bien, inicialmente se consideró que estas variaciones podrían afectar la detección del rostro, las pruebas llevadas a cabo utilizando diferentes fondos, tanto claros y oscuros como se muestra en la figura 30, demostraron que éste no tenía un impacto significativo, factores como el color de piel de la persona u objetos circundantes no afectaron al desempeño del sistema, debido, principalmente, a que la distancia del dispositivo implica que la persona cubra la mayor parte de la imagen. Las variaciones en los resultados al cambiar los fondos no justifican su consideración como un factor determinante en este proceso.

Figura 30. Detección con variaciones de fondo.



Fuente: Elaboración propia.

5. ANÁLISIS Y RESULTADOS

5.1. RESULTADOS OBTENIDOS EN EL PROTOCOLO DE PRUEBAS

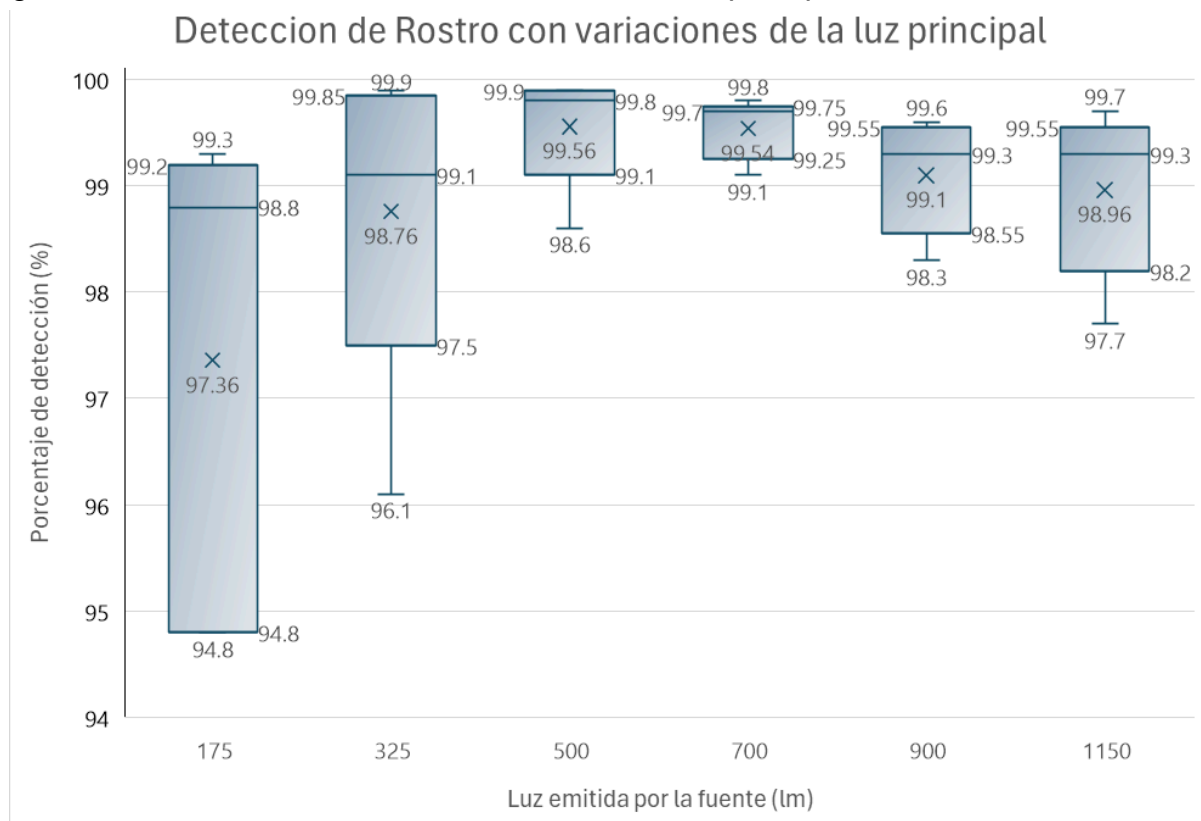
5.1.1. Variaciones de iluminación. Se realizaron tres pruebas, correspondientes a las 3 fuentes de iluminación (luz principal, luz de fondo, luz de relleno), con un grupo de 5 personas, en cada prueba se usaron 6 niveles de intensidad lumínica proporcionados por una lámpara LED regulable.

Los sujetos fueron expuestos a cada nivel de iluminación durante 2 minutos (lo que sumó un total de 12 minutos por prueba), por cada uno de estos segmentos el sistema llevó un registro del número total de fotogramas capturados por la cámara, la cantidad de fotogramas con detección exitosa del rostro, los ojos y la boca (abierta) por separado. Una vez obtenidos estos datos, se calculó la tasa de detección de cada parámetro (rostro, ojos, boca) dividiendo el número de detecciones exitosas por el total de fotogramas capturados en cada nivel de iluminación.

Durante las pruebas, los sujetos mantuvieron una postura constante para evitar que el movimiento influyera en los resultados de detección. De este modo, se garantizó que las variaciones en la tasa de detección estuvieran vinculadas únicamente a los cambios en las condiciones de iluminación. Adicionalmente a esto, se supervisó visualmente que el sistema identificara correctamente los parámetros evaluados para evitar falsos positivos, dado que el sistema no permite verificar, por sí solo, si las detecciones eran correctas, asegurando así que los datos registrados reflejaban detecciones genuinas.

A continuación se presentan los gráficos de porcentaje de detección del rostro, ojos y boca para cada nivel de variación de iluminación: luz principal, de fondo y de relleno respectivamente, como soporte las tablas de porcentajes se encuentran en el apartado de anexos del documento.

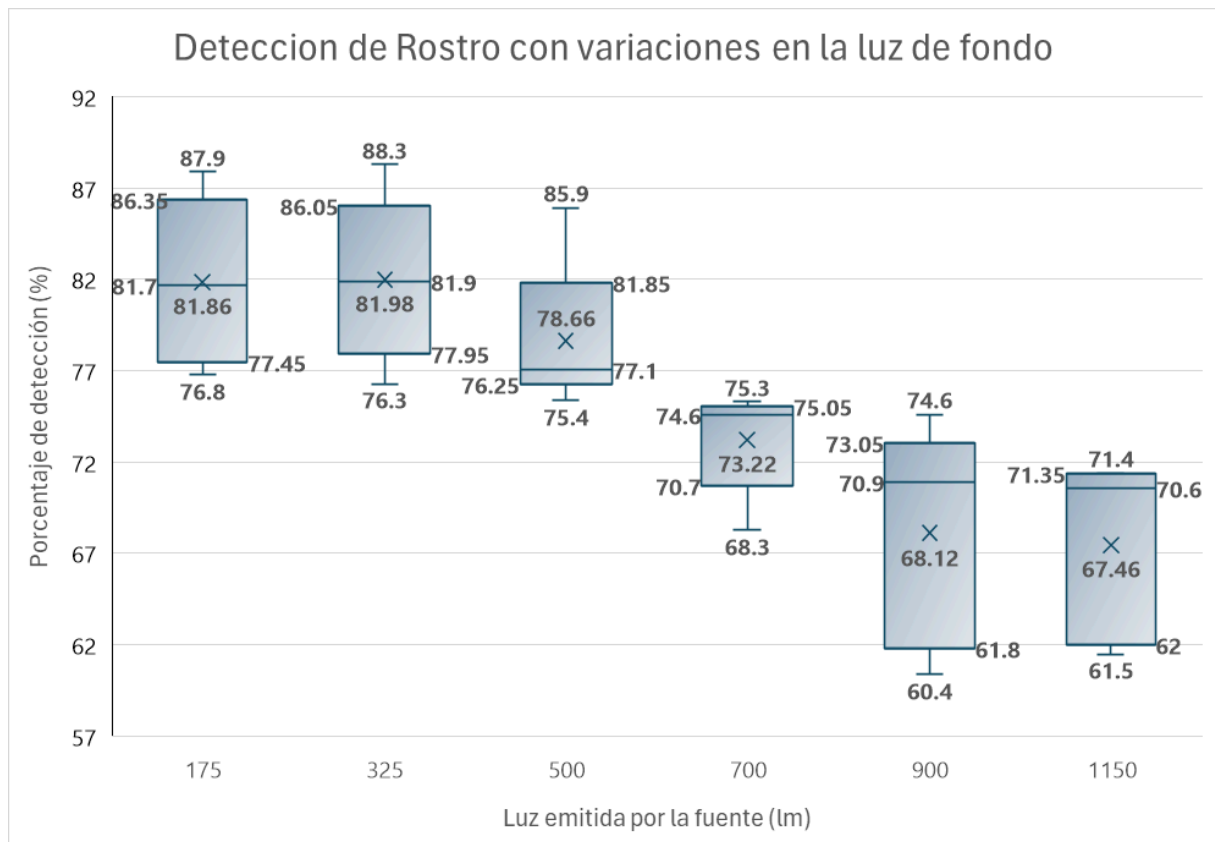
Figura 31. Detección de Rostro con variación de luz principal.



Fuente: Elaboración propia.

El sistema muestra una tendencia a mejorar la detección a medida que aumenta la intensidad de la fuente de luz. A partir de los 500 lúmenes, el rendimiento es óptimo, alcanzando aproximadamente un 99.5% de detección alrededor de los 700 lúmenes. Además, la dispersión de los datos es menor en comparación con niveles más bajos, como los 175 lúmenes, donde se observa una mayor variabilidad en las detecciones.

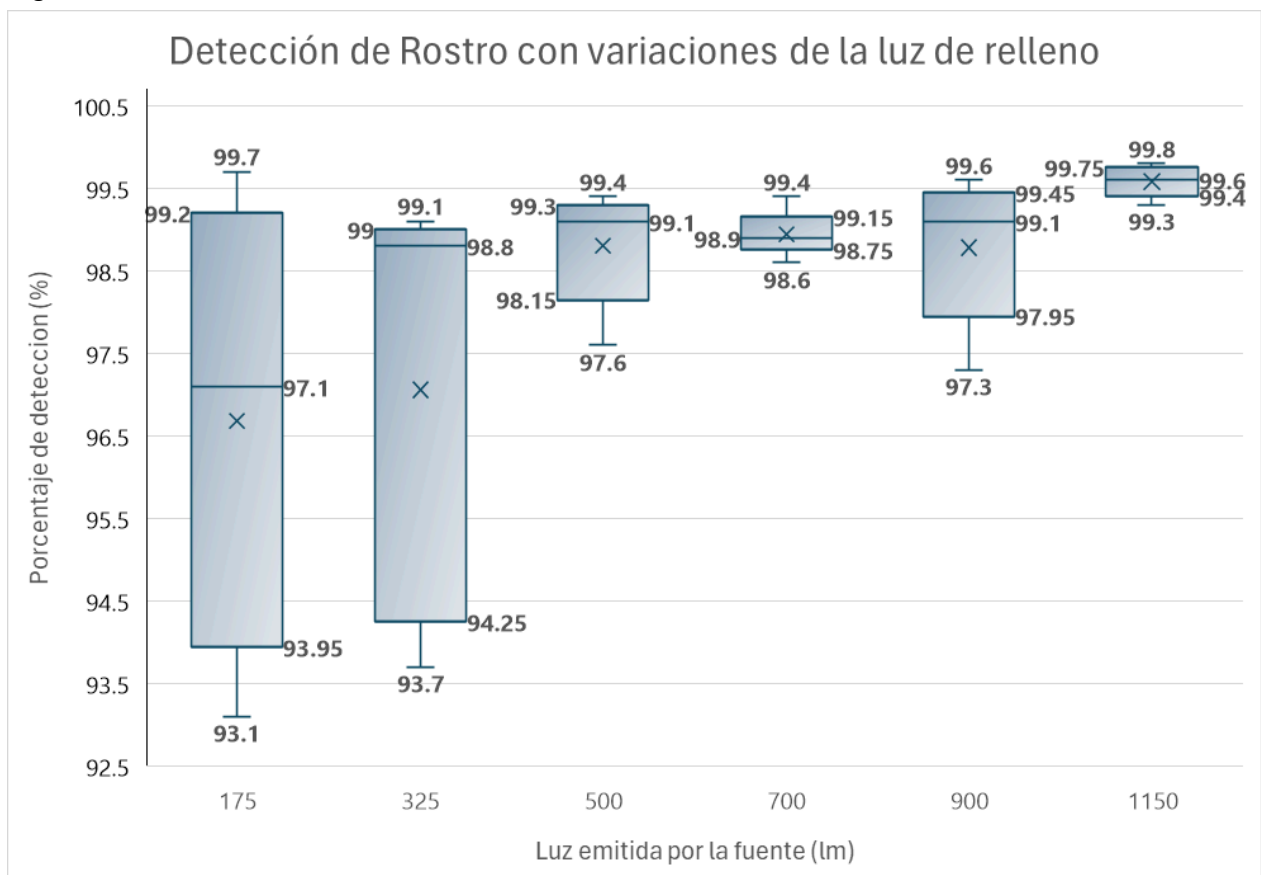
Figura 32. Detección de Rostro con variación de la luz de fondo.



Fuente: Elaboración propia.

Esta fuente de luz impacta negativamente en la detección, se observa una disminución en el porcentaje de detección de rostros a medida que aumentan los niveles de iluminación. Alrededor de los 500 lúmenes, la tasa de detección cae por debajo del 77%.

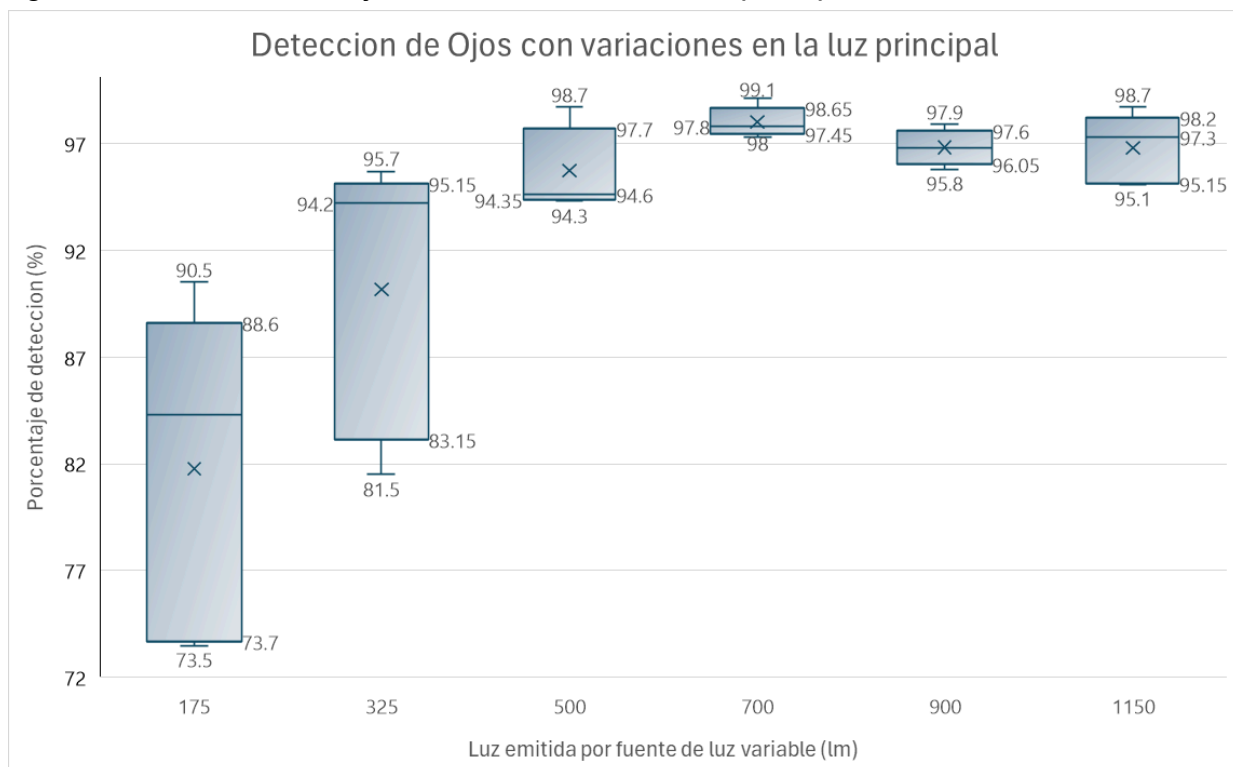
Figura 33. Detección de Rostro con variación de la luz de relleno.



Fuente: Elaboración propia.

El sistema presenta un comportamiento similar al registrado por la luz principal, mostrando mayor dispersión en los datos con niveles de iluminación bajos y una porcentaje de detección más alto y con menor variación de resultados entre los sujetos de prueba.

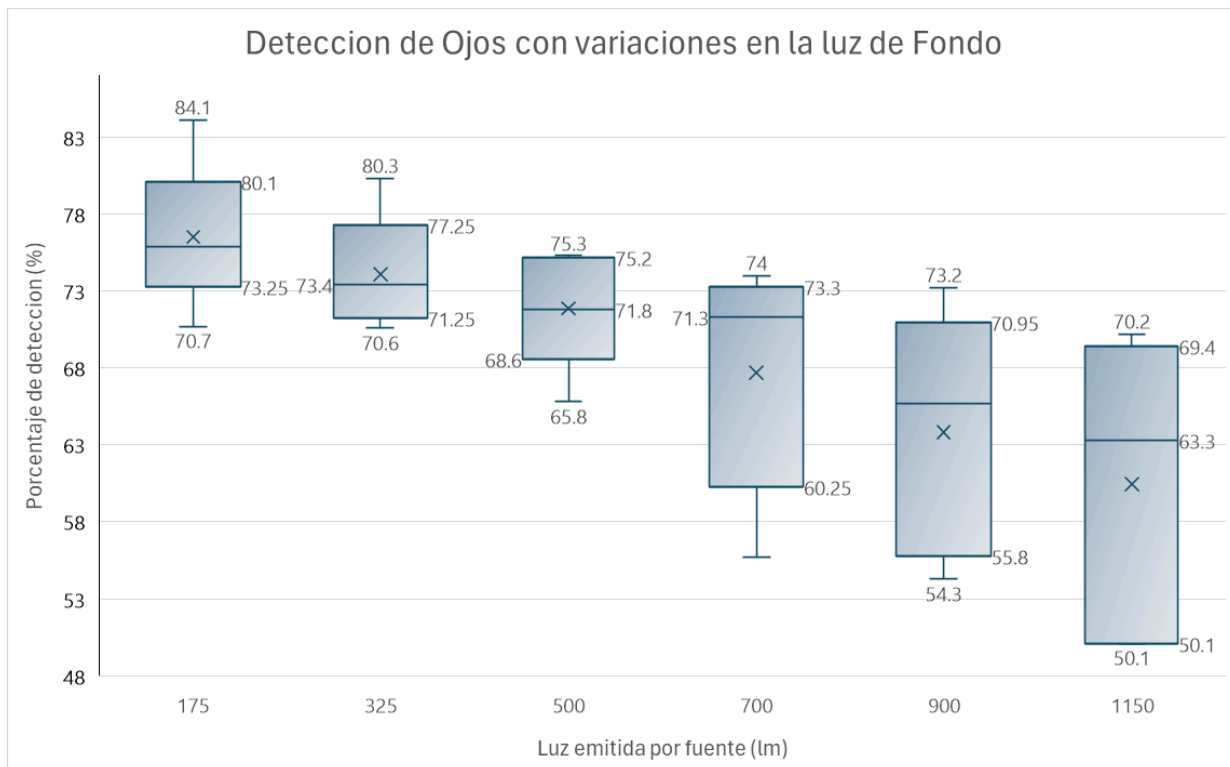
Figura 34. Detección de ojos con variación en la luz principal.



Fuente: Elaboración propia.

El sistema alcanza un mayor porcentaje de detección de ojos con una iluminación cercana a los 700 lúmenes, con una media del 97.8% y la menor variabilidad en los datos. Al igual que en el caso de la detección de rostros, niveles inferiores a los 500 lúmenes presentan una mayor dispersión y un menor porcentaje de detección. Curiosamente, por encima de los 900 lúmenes, el sistema vuelve a mostrar fallas en la detección y un aumento en la dispersión en comparación con los 700 lúmenes.

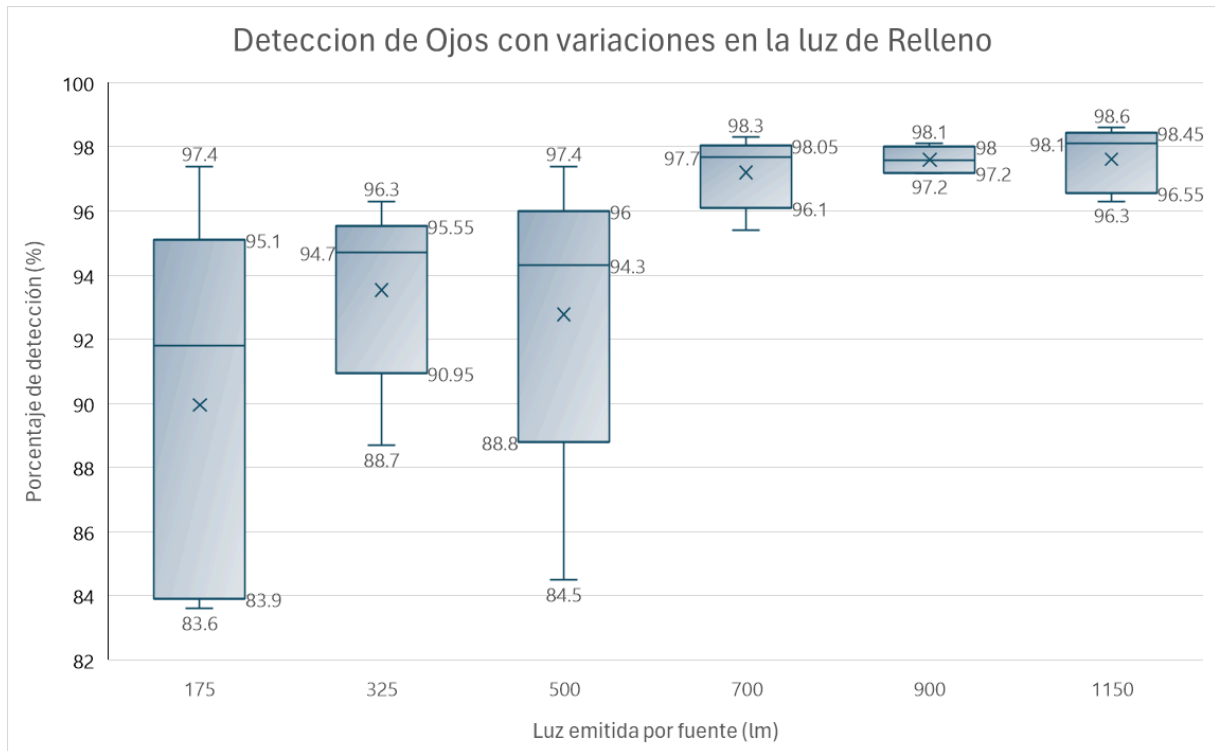
Figura 35. Detección de ojos con variación en la luz de fondo



Fuente: Elaboración propia.

De manera similar a la detección de rostros, el sistema vuelve a presentar afectaciones en la detección a niveles de iluminación más altos. Aunque los datos muestran poca dispersión en niveles bajos de luz, el porcentaje de detección no supera el 78% en promedio y disminuye a medida que aumentan los niveles de iluminación.

Figura 36. Detección de ojos con variación de la luz de relleno.

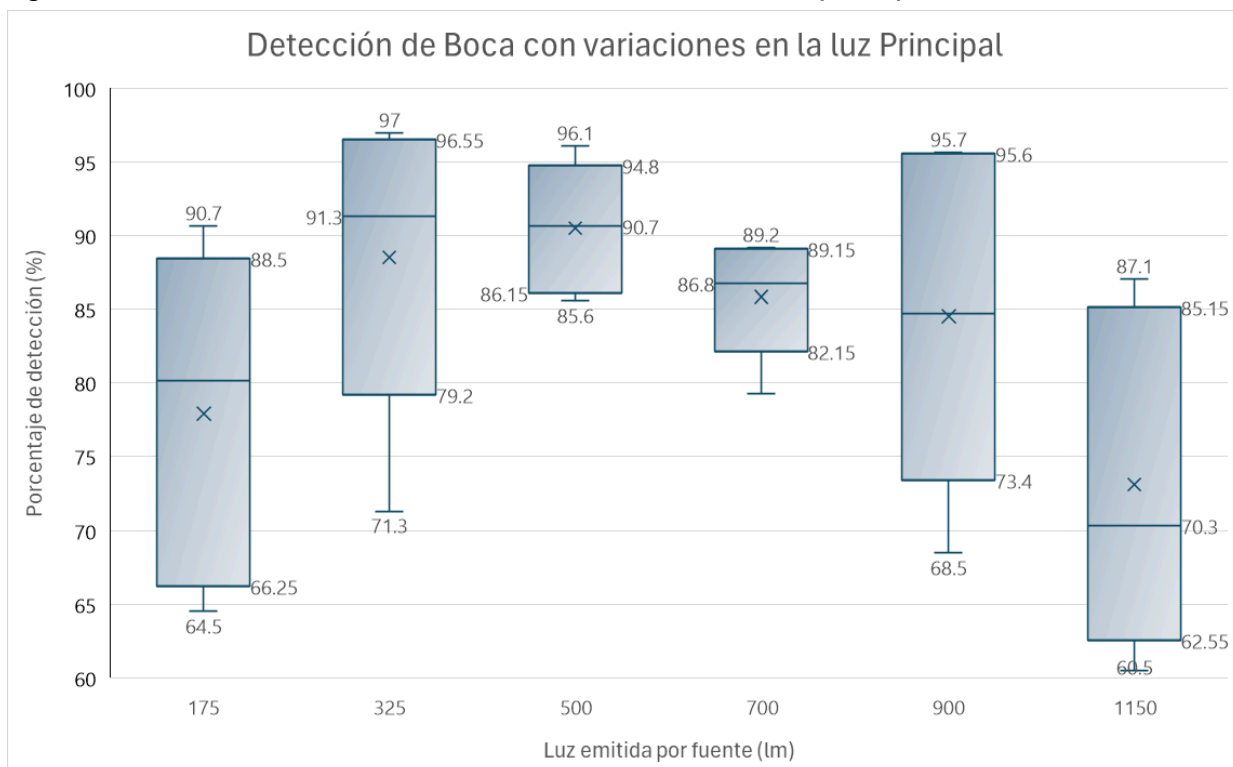


Fuente: Elaboración propia.

Análisis de la detección de Ojos:

El sistema muestra un buen desempeño cuando la iluminación se encuentra alrededor de los 900 lúmenes, con un porcentaje de detección cercano al 97% y baja variabilidad en los resultados. Nuevamente, los niveles bajos de iluminación presentan una mayor inestabilidad en la detección, con más personas experimentando tasas de detección reducidas.

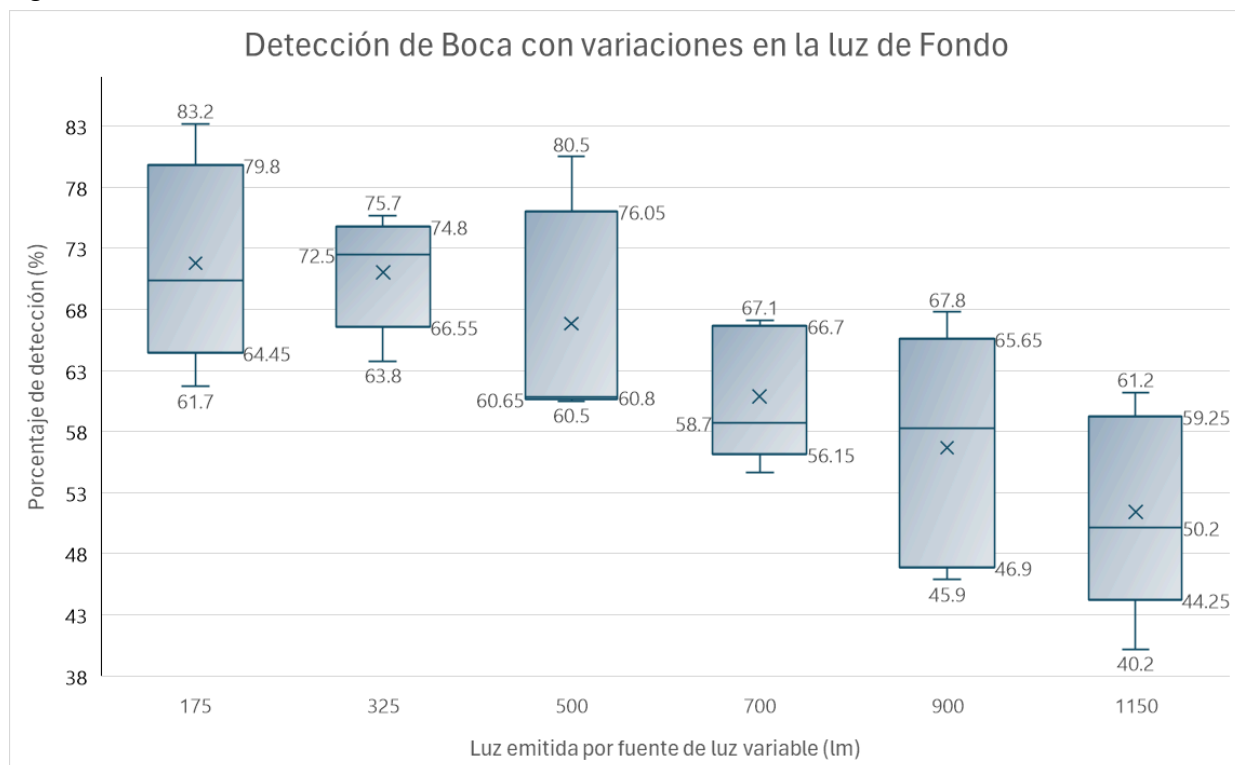
Figura 37. Detección de la boca con variaciones de la luz principal.



Fuente: Elaboración Propia.

El sistema alcanza su mayor porcentaje de detección con una iluminación cercana a los 500 lúmenes, logrando un promedio del 90.7%. Aunque los datos presentan cierta dispersión, este nivel de iluminación ofrece el mejor rendimiento en comparación con niveles inferiores a 325 lúmenes y superiores a 700 lúmenes.

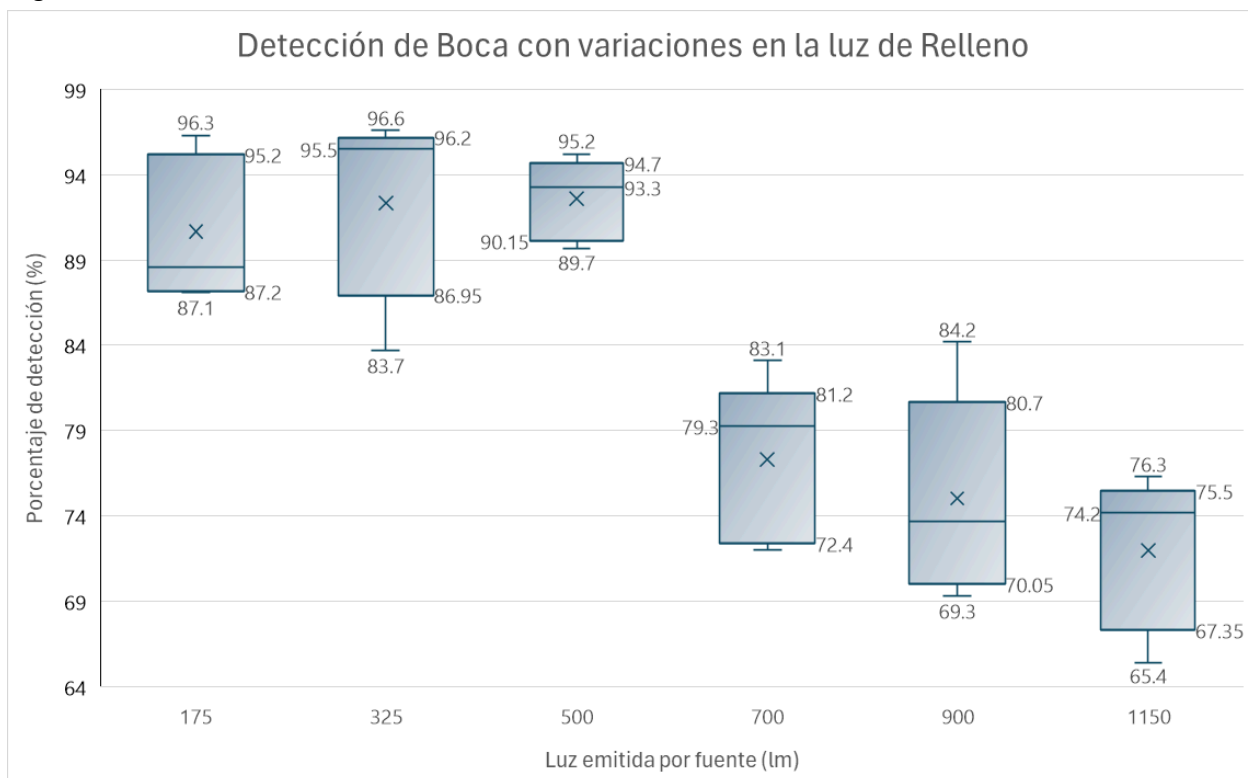
Figura 38. Detección de la boca con variaciones de la luz de fondo.



Fuente: Elaboración propia.

La luz de fondo genera problemas en la detección de bocas a medida que aumenta el nivel de iluminación. Sin embargo, en niveles bajos, el porcentaje de detección también es limitado, situándose alrededor del 73%.

Figura 39. Detección de la boca con variación de la luz de relleno



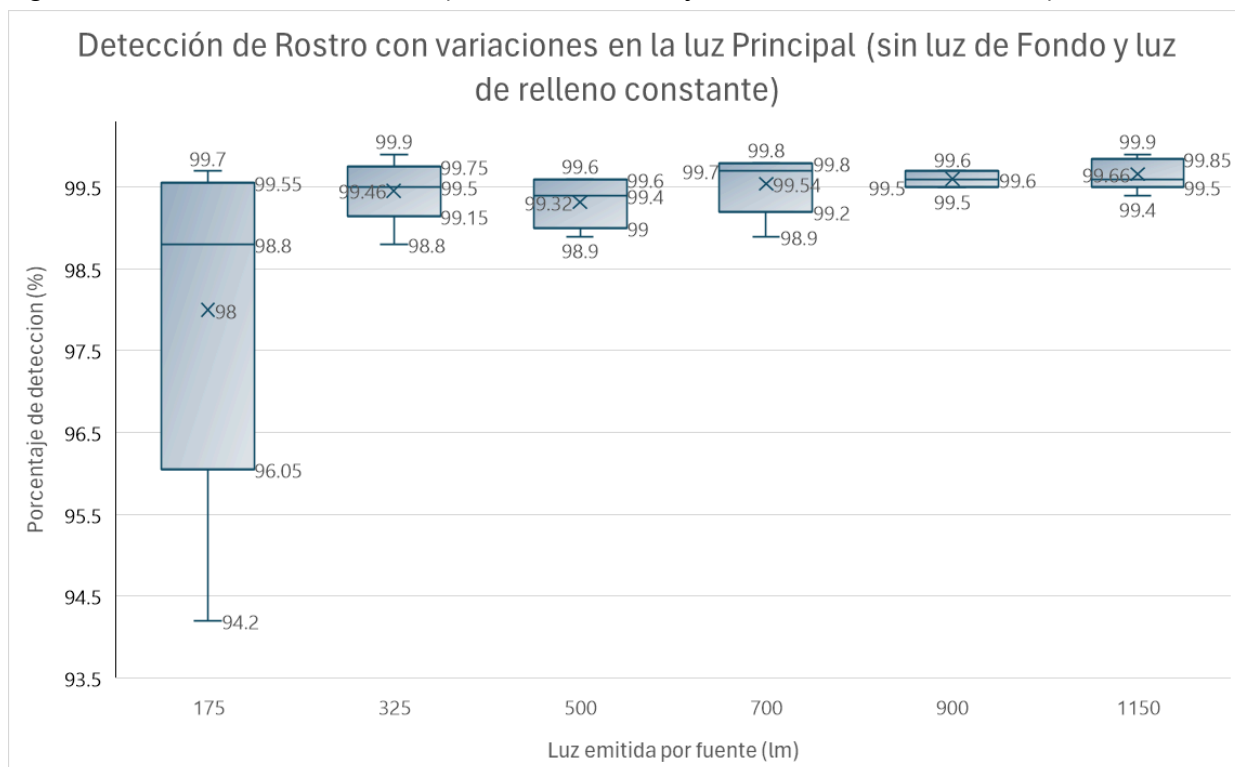
Fuente: Elaboración propia.

Se observa una detección promedio superior al 94% con niveles bajos de esta fuente de luz. Sin embargo, a medida que la iluminación supera los 500 lúmenes, el porcentaje de detección disminuye, llegando a un promedio del 72%, acompañado de una mayor dispersión en los datos.

Con el análisis realizado a estos datos, se puede concluir que, la detección de los 3 parámetros (rostro, ojos y boca) se presenta en un rango entre los 250 y 800 lúmenes generados por la luz principal y la luz de relleno, por otro lado, la luz de fondo parece afectar negativamente el desempeño del sistema, puesto que al aumentar su nivel de intensidad, se presentan menos detecciones en los 3 parámetros, esto debido, posiblemente a que generan contraluz en el rostro de la persona, dificultando la identificación facial y por consiguiente, la detección de los ojos y la boca. Por otro lado, se observó que la luz de relleno no generaba cambios sustanciales en la detección, solo se apreció una mejora en la detección de la boca cuando se empleaban niveles de intensidad de luz similares a la fuente principal.

A raíz de estos hallazgos, se determinó realizar una prueba adicional, en la cual, se eliminó la luz de fondo y una luz de relleno constante (aproximadamente de 400 lúmenes).

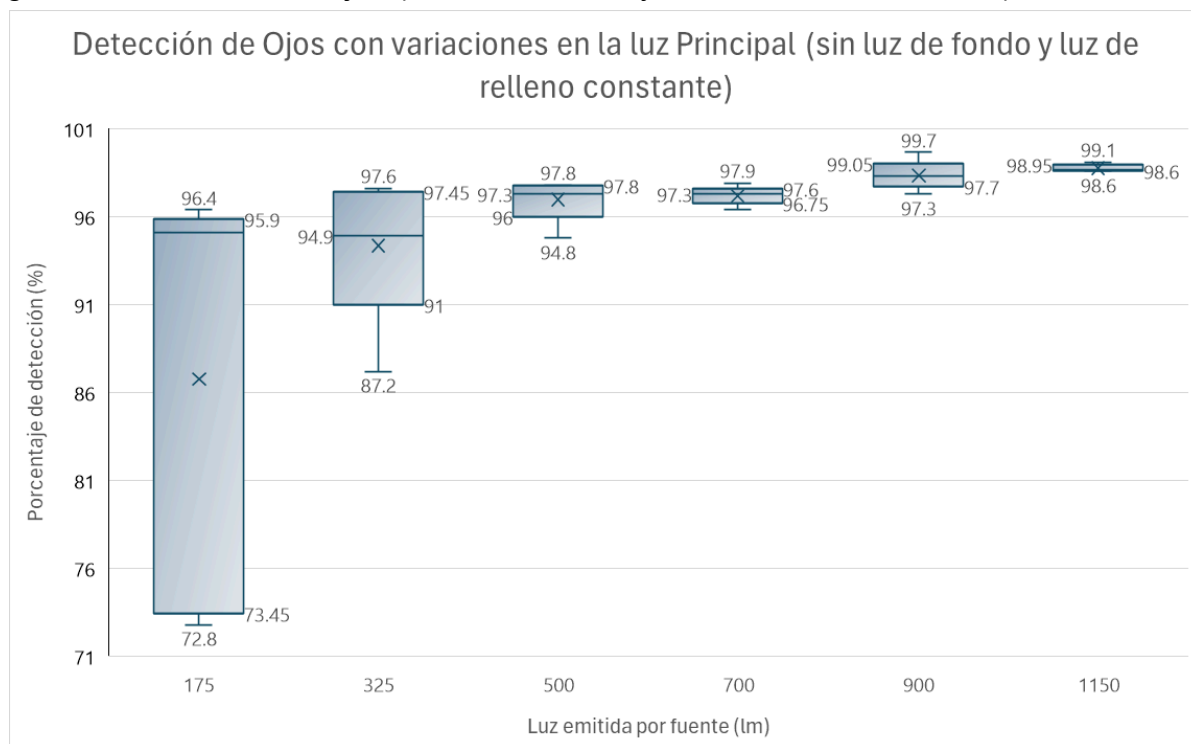
Figura 40. Detección de rostro (sin luz de fondo y luz de relleno constante)



Fuente: Elaboración Propia.

Se observa que la detección mejora considerablemente a partir de los 325 lúmenes, donde el porcentaje de detección se mantiene alto, alrededor del 99.5% y superior, con una menor dispersión en los datos. En cambio, para una iluminación de 175 lúmenes, la detección es más variable, con una media de detección más baja (98%) y una mayor dispersión, entre 94.2% y 99.7%. A partir de los 500 lúmenes, los resultados son más consistentes y cercanos al 99.9%.

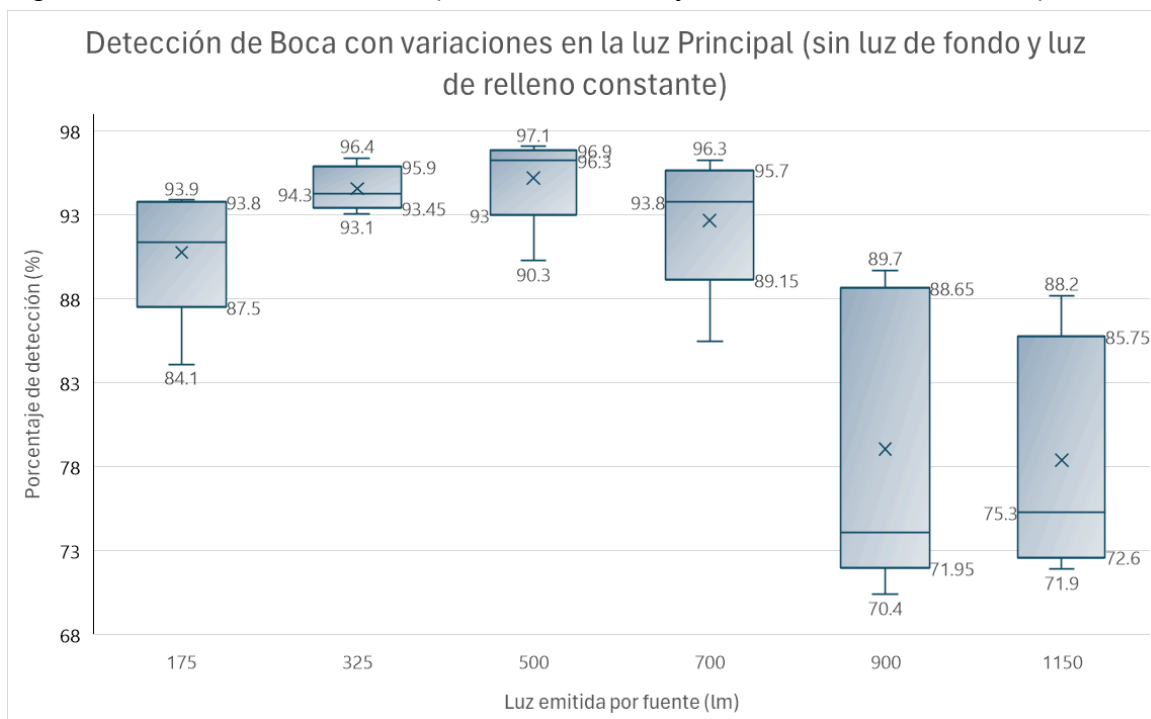
Figura 41. Detección de ojos (sin luz de fondo y luz de relleno constante).



Fuente: Elaboración Propia.

Los datos sugieren que, con una iluminación de 175 lúmenes, la detección es menos confiable, con una alta dispersión de los datos entre 72.8% y 96.4%, y una mediana de 73.45%. A medida que aumenta la luz, los resultados mejoran considerablemente. A partir de 500 lúmenes, la detección es mucho más estable y precisa, alcanzando más del 97% en promedio, con poca variabilidad.

Figura 42. Detección de boca (sin luz de fondo y luz de relleno constante).



Fuente: Elaboración Propia.

Los hallazgos muestran que la precisión de detección es más alta y consistente se encuentra a 500 lúmenes, con una mediana del 97.1% y una dispersión relativamente baja, entre 90.3% y 96.9%. Para niveles de luz más bajos, como 175 y 325 lúmenes, la detección es más variable, con medianas de 87.5% y 95.9%, respectivamente, pero con algunas detecciones por debajo de 84.1%. A partir de 700 lúmenes, aunque la detección es buena la dispersión empieza a aumentar, lo que indica que estas condiciones de luz no favorecen la detección de la boca.

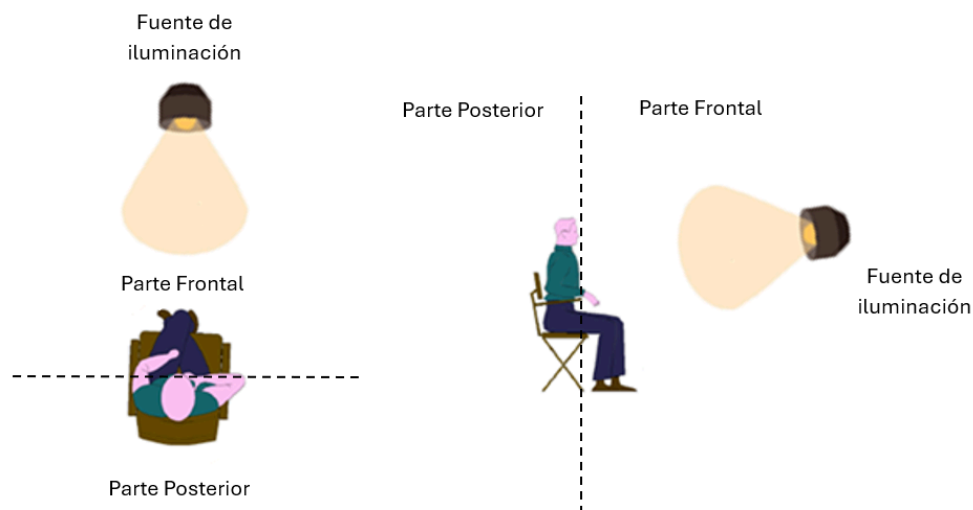
Al examinar el comportamiento de la prueba sin luz de fondo del rostro, los ojos y la boca, se puede afirmar que el entorno para llevar a cabo la detección, debe contar con una fuente lumínica principal proveniente de la parte frontal del usuario, la cual ilumine el rostro de la persona de manera uniforme, minimizando al máximo condiciones en donde exista una fuente de luz en la parte trasera del usuario, tal como se observa en la figura 43.

Por otro lado, al observar detalladamente cada prueba, se determinó que la detección del rostro fallaba en momentos en los cuales la persona movía la cabeza de forma brusca, produciendo falsos positivos o interrumpe la detección facial, lo que a su vez afectaba la detección de los ojos y boca abierta.

La detección de los ojos presenta un porcentaje ligeramente menor a la del rostro, debido a los parpadeos que la persona realizó durante la prueba, ya que al descartar los fotogramas en los cuales el ojo se encontraba cerrado, se identificó que los porcentajes aumentaban casi al mismo nivel que la detección del rostro.

La detección de la boca abierta presentó un mayor número de fallos, comparado al del rostro y la boca, principalmente porque el entrenamiento no contaba con suficientes muestras para igualar la precisión de los otros dos clasificadores. Se observó que las fallas ocurrían con mayor frecuencia cuando los niveles de luz caían por debajo de 250 lm o superaban los 800 lm. Además, algunos de estos fallos también se debían a la fisonomía de la persona, como a la barba o arrugas cerca de la boca.

Figura 43. Iluminación para obtener un entorno controlado en términos de luz.



Fuente: Elaboración Propia, en base a J. P. Arias [5].

Dado que se busca un sistema que pueda detectar rostros, ojos y boca de manera adecuada, es fundamental identificar un rango de niveles de iluminación donde los tres elementos presentan el mayor porcentaje de detección y menor variabilidad, asegurando que funcione con el mayor número de sujetos posible. Con las pruebas se logró determinar un rango entre 325 y 700 lúmenes emitidos por la luz principal.

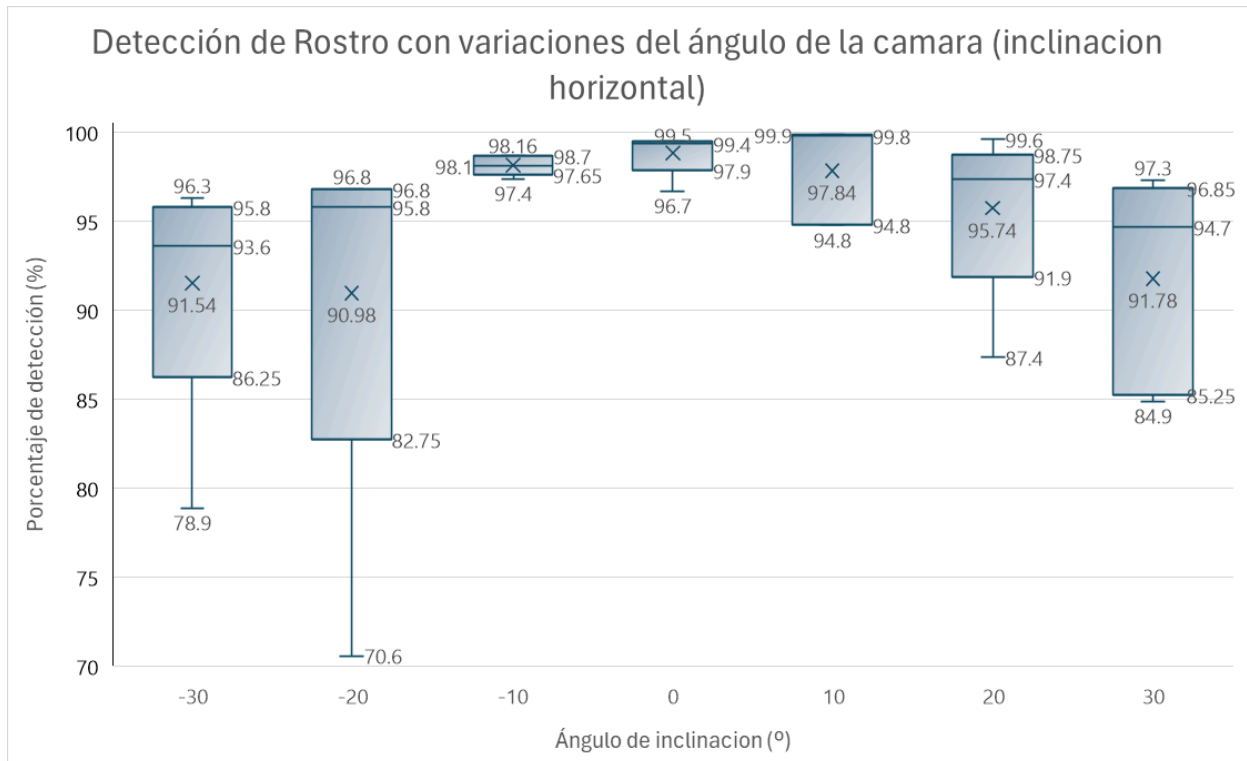
Por otra parte, los datos proporcionados por el sensor lumínico del dispositivo indican un rango de 60 a 180 lux, que se relaciona con los niveles de iluminación de lúmenes generados por las fuentes de luz. Sin embargo, este indicador sólo ofrece información sobre el nivel de iluminación general del área circundante al teléfono móvil, lo que puede ayudar al usuario a identificar un entorno adecuado, siempre que haya una fuente de iluminación principal que ilumine el rostro como se mencionó anteriormente.

5.1.2. Variaciones de inclinación de cámara. Estas pruebas se llevaron a cabo usando el escenario planteado en la sección 5.1.1, para evitar que el factor de iluminación afecta a los datos de esta sección.

Las pruebas involucraron a 5 sujetos, cada una tuvo una duración de 14 minutos en total y cada variación en el ángulo se mantuvo durante 2 minutos. De forma similar a la anterior prueba, el sistema registró el número total de fotogramas capturados por la cámara y el número de fotogramas con detección exitosa, del rostro, los ojos y la boca (abierta) por separado, para lograr encontrar una tasa de porcentaje de detección con cada uno de estos parámetros.

A continuación, se presentan los gráficos de porcentaje de detección de rostro, ojos y boca para inclinaciones verticales y horizontales de la cámara con respecto al rostro de la persona, como soporte las tablas de porcentajes se encuentran en el apartado de anexos del documento.

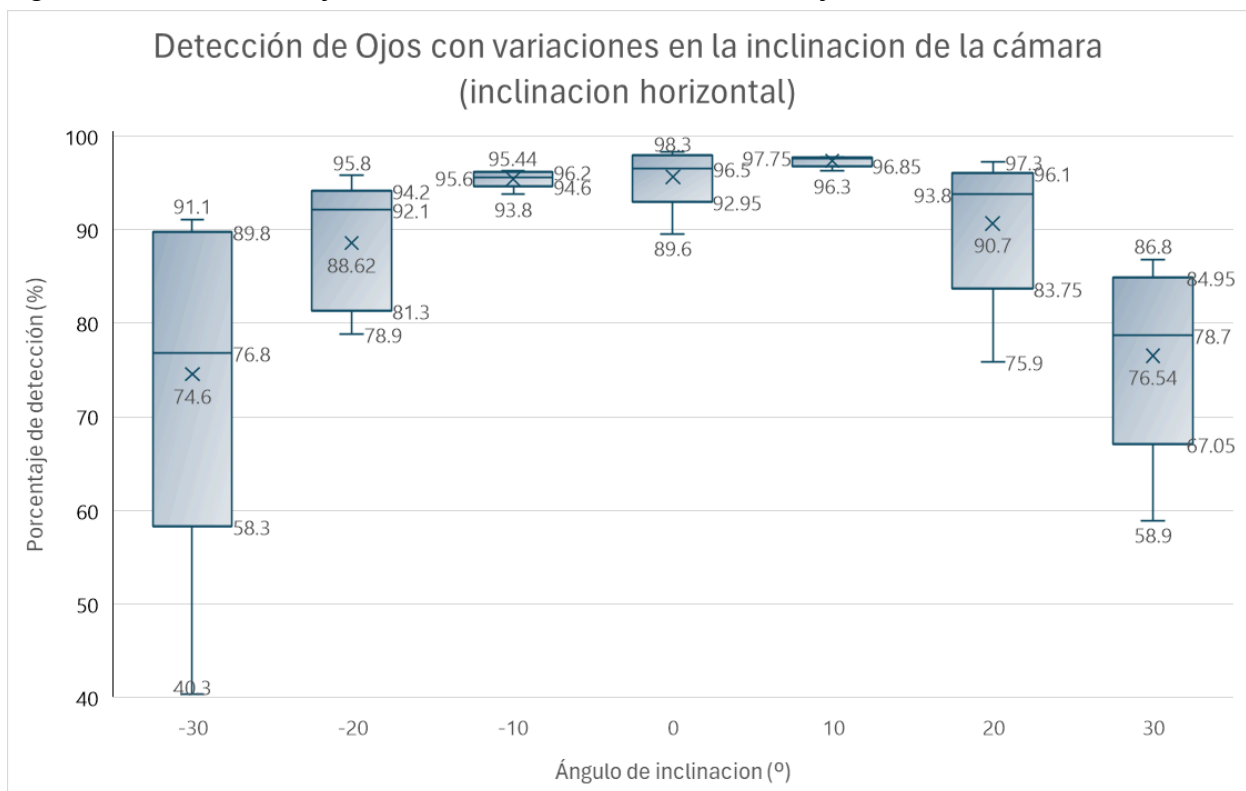
Figura 44. Detección rostro con inclinación de la cámara en el eje horizontal.



Fuente: Elaboración propia.

De acuerdo a los resultados, los mejores porcentajes de detección se obtuvieron con ángulos de inclinación cercanos a 0°, alcanzando una mediana de 99.5%, los ángulos de $\pm 10^\circ$, también demostraron un rendimiento bueno, con medianas por encima de 97%. Como se esperaba, a medida que el ángulo de inclinación aumentaba en cualquier dirección, la precisión tendía a disminuir.

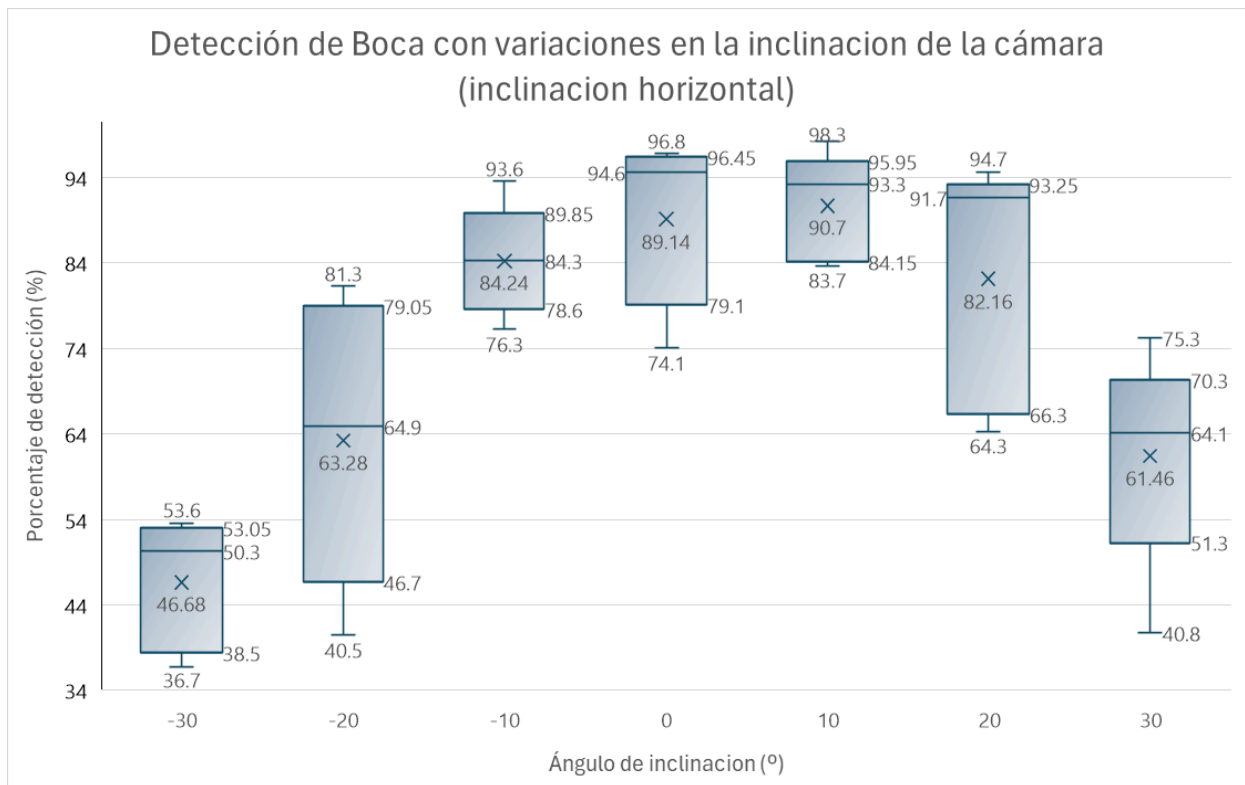
Figura 45. Detección ojos inclinación de la cámara en el eje horizontal.



Fuente: Elaboración propia.

Se observa que la detección más alta y consistente es cuando la cámara está cercana a la posición de 0°, con una mediana de 96.5%, nuevamente ángulos cercanos al cero presentan la menor dispersión de datos y mayor porcentaje de detección. los ángulos mayores a $\pm 10^\circ$ muestran mayor variabilidad y una disminución significativa de detección, con mínimos de 41.3% y 58.9%.

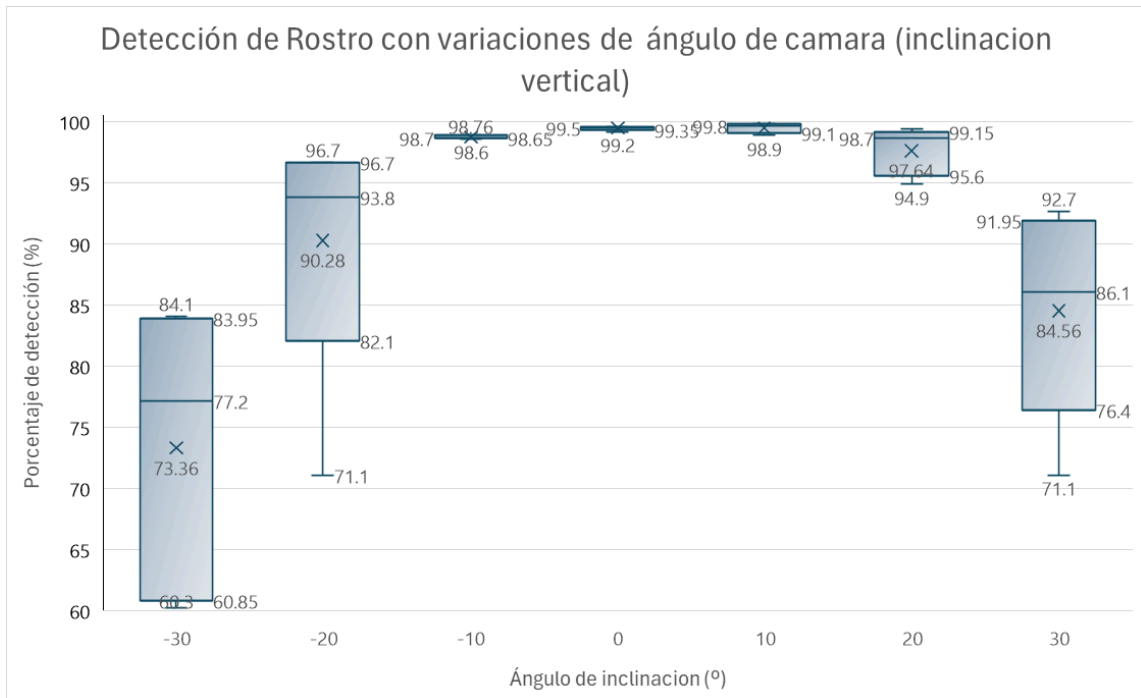
Figura 46. Detección boca inclinación de la cámara en el eje horizontal.



Fuente: Elaboración propia.

La precisión de detección alcanza valores de 98.3% cerca al ángulo cero y disminuye notablemente a medida que el ángulo de inclinación se aleja de éste, en ambas direcciones. Los ángulos extremos presentan mayor variabilidad y tasas de detección más bajas, con mínimos de 36.7% y 40.8%.

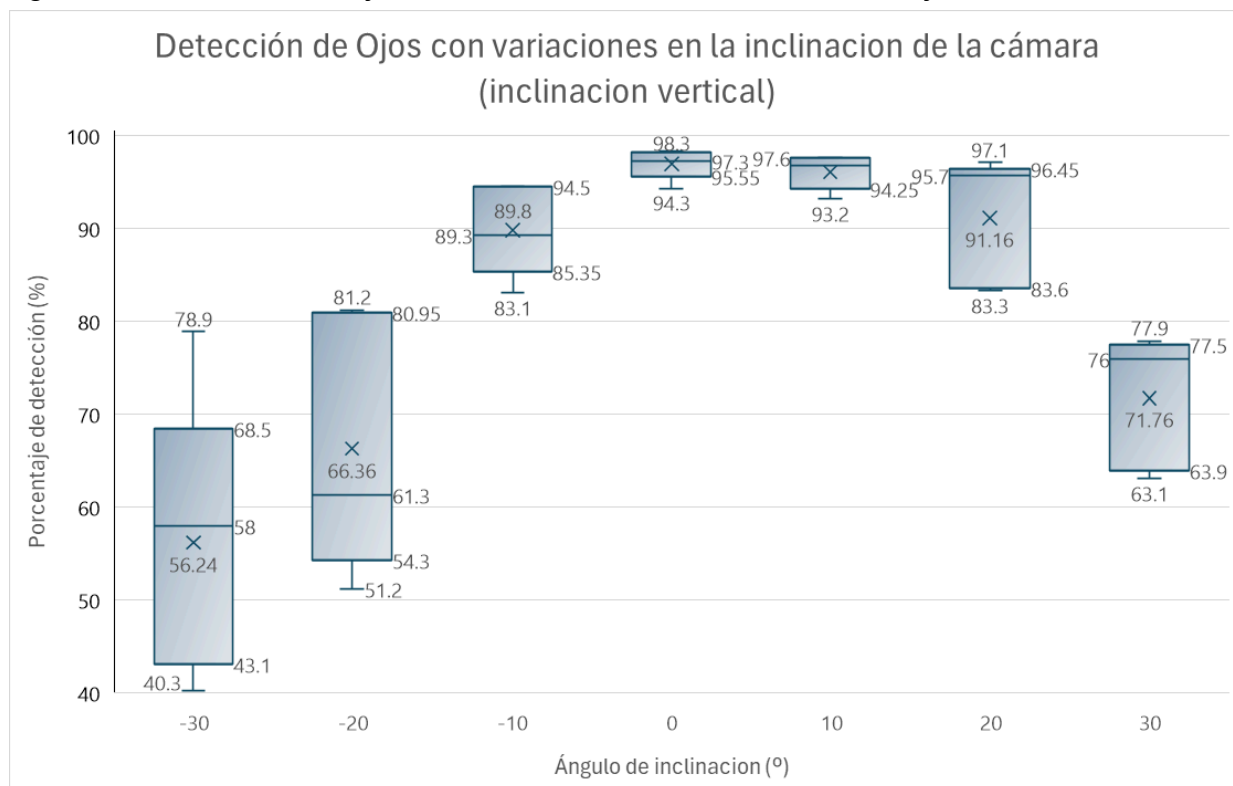
Figura 45. Detección de rostro con inclinación de la cámara en el eje vertical.



Fuente: Elaboración propia.

Como se observa en los datos, algo similar a los planos de orientación horizontal ocurre para el eje vertical, entre -10° y 10° la dispersión es tan pequeña, por ello las cajas se observan más achatadas, por el contrario, ángulos por fuera de este rango aumentan la variabilidad de detección en los sujetos de prueba y con más fallas.

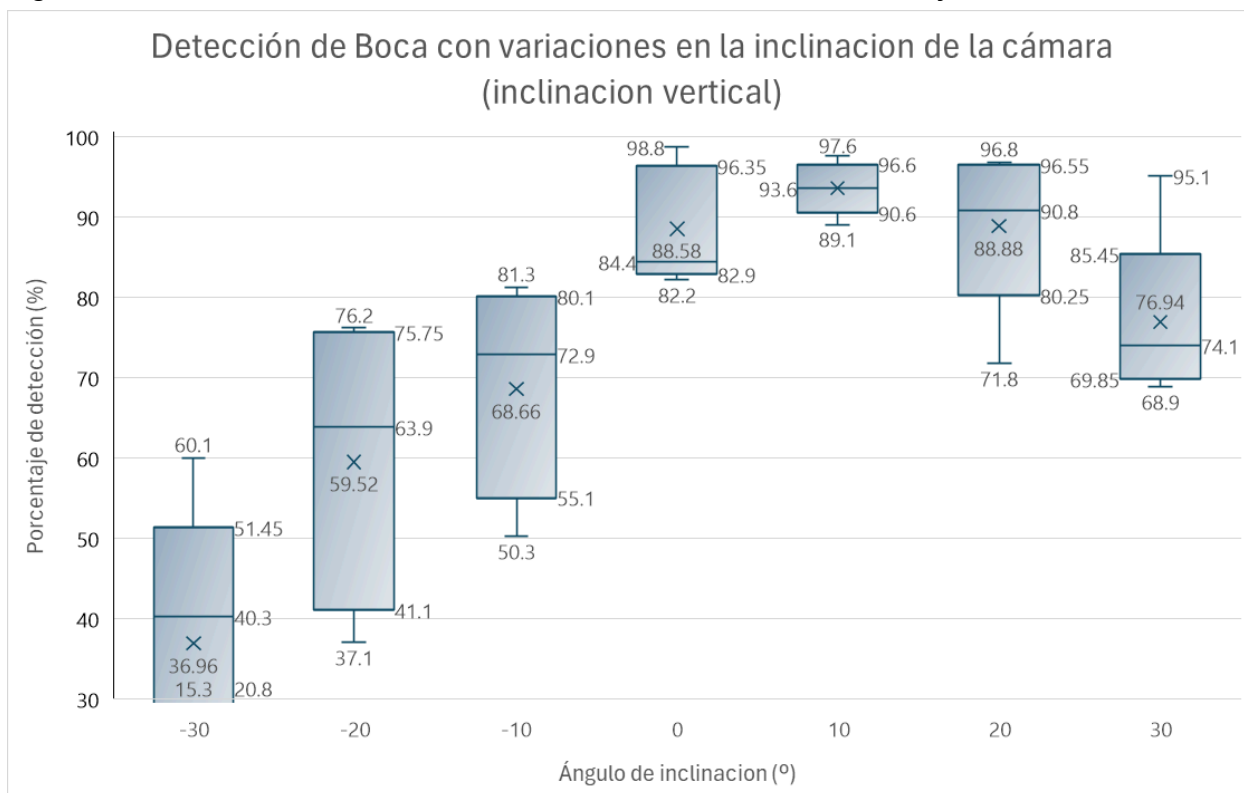
Figura 48. Detección de ojos con inclinación de la cámara en el eje vertical.



Fuente: Elaboración propia.

Se observa, en el rango de -10° a 10° , porcentajes de detección buenos y alcanzando el 98.3% en 0° . La precisión disminuye gradualmente a medida que el ángulo de inclinación se aleja de 0° en ambas direcciones. Los ángulos extremos, especialmente -30° y -20° , presentan la mayor variabilidad y las tasas de detección más bajas, con mínimos de 40.3% y 51.2% respectivamente. La detección parece ser más robusta para inclinaciones positivas (cámara hacia arriba) que para las negativas, manteniendo una mediana de detección por encima del 70% incluso a 30° .

Figura 49. Detección de boca con inclinación de la cámara en el eje vertical.



Fuente: Elaboración propia.

Los porcentajes de detección parecen ser más favorables para inclinaciones positivas (cámara hacia arriba). El sistema alcanzó un 96.3% de detección en torno a los 10°, con la menor dispersión de datos en comparación con otros ángulos. Incluso hasta los 30°, la detección sigue siendo superior en comparación con inclinaciones negativas (cámara hacia abajo), logrando un 76.9% a 30°. Una inclinación de -10°, la mediana de detección alcanza un 68.7%, mostrando una tendencia a valores más bajos.

Según los datos reunidos durante las pruebas de inclinación, se concluye que el sistema exhibe un mayor índice de aciertos, en los siguientes rangos de ángulo:

Inclinación respecto al plano horizontal:

Un promedio de detección de rostro, mayor a 97.8% con ángulos entre -10° y 10°.

Un promedio de detección de Ojos mayor a 95.4% con ángulos entre -10° y 10°.

Un promedio de detección de Boca mayor a 84.2% con ángulos entre -10° y 10°.

Inclinación respecto al plano vertical:

Un promedio de detección de rostro, mayor a 97.6% con ángulos entre -10° y 20° .

Un promedio de detección de Ojos mayor a 91% con ángulos entre -10° y 20° .

Un promedio de detección de Boca mayor a 88.5% con ángulos entre -10° y 20° .

5.2. RESULTADOS DEL SISTEMA FINAL PARA LA DETECCIÓN DE FATIGA

La prueba del sistema final para la detección de fatiga se llevó a cabo usando un algoritmo dentro de la aplicación para simular el estado de “fatiga” y “no fatiga” a través de indicadores de luz, en la pantalla del dispositivo.

Para las pruebas de detección de fatiga, sobre el sistema final, se integró a la aplicación, una simulación de dos estados de la persona (“fatiga” y “no fatiga”), con indicadores de luz en la pantalla del dispositivo móvil, destinados a señalar el momento en el cual la persona debe realizar un parpadeo o un bostezo.

En el estado de “no fatiga” solo activa un indicador de luz de color amarillo el cual se muestra 15 veces a lo largo de 1 minuto, simulando un parpadeo de una persona que no presenta fatiga.

En el estado de “fatiga” se activan dos indicadores de luz, el primero cuenta con dos destellos (amarillo y rojo) y se activa al rededor de 16 a 22 veces por minuto. El color amarillo indica al usuario, realizar parpadeos de duración “normal” (duracion entre 100 y 200 milisegundos), mientras que el destello rojo indica parpadeos de duración “larga” (duración superior a 200 milisegundos). El segundo indicador corresponde a una luz verde, señal para que el usuario realice un bostezo, se activa 1 o 2 veces por minuto.

Esta prueba se llevó a cabo con 5 personas en el entorno controlado en términos de iluminación y ángulos del dispositivo, usando los valores encontrados en las pruebas de la sección 5.1, con una duración de 7 minutos, los 3 primeros minutos activando la simulación de “no fatiga” y los últimos 4 minutos, activando la simulación de “fatiga”. Las tablas subsiguientes registran los resultados obtenidos en cada minuto, a partir de los parpadeos y bostezos indicados por el aplicativo (Ind.), realizados por la persona (Realiz.), y detectados por el sistema (Det.).

Tabla 4. Resultados de las pruebas con la simulación de fatiga (sujeto 1).

Tiempo (Min)	Parpadeos Duración Normal			Parpadeos de Duración Larga			Bostezos			Estado
	Ind.	Realiz.	Det.	Ind.	Realiz.	Det.	Ind.	Realiz.	Det.	
1	15	14	14	0	1	0	0	0	0	no fatiga
2	15	15	15	0	0	0	0	0	0	no fatiga
3	15	16	14	0	2	1	0	0	0	no fatiga
4	8	8	7	8	7	7	1	1	0	posible fatiga
5	8	8	8	10	10	10	1	1	1	fatiga
6	8	8	7	9	9	9	2	2	1	fatiga
7	8	7	7	9	9	8	1	1	0	posible fatiga

Fuente: Elaboración Propia.

Tabla 5. Resultados de las pruebas con la simulación de fatiga (sujeto 2).

Tiempo (Min)	Parpadeos Duración Normal			Parpadeos de Duración Larga			Bostezos			Estado
	Ind.	Realiz.	Det.	Ind.	Realiz.	Det.	Ind.	Realiz.	Det.	
1	15	15	15	0	0	0	0	0	0	no fatiga
2	15	15	14	0	0	0	0	0	0	no fatiga
3	15	15	12	0	0	2	0	0	0	no fatiga
4	7	7	7	9	9	9	1	1	1	fatiga
5	7	7	7	8	8	7	2	2	2	posible fatiga
6	8	8	8	10	9	9	1	1	1	fatiga
7	7	7	7	10	10	9	2	2	1	fatiga

Fuente: Elaboración Propia.

Tabla 6. Resultados de las pruebas con la simulación de fatiga (sujeto 3)

Tiempo (Min)	Parpadeos Duración Normal			Parpadeos de Duración Larga			Bostezos			Estado
	Ind.	Realiz.	Det.	Ind.	Realiz.	Det.	Ind.	Realiz.	Det.	
1	15	15	14	0	0	1	0	0	0	no fatiga
2	15	15	15	0	0	0	0	0	0	no fatiga
3	16	16	14	0	0	1	0	0	0	posible fatiga
4	7	6	7	10	10	9	2	2	2	fatiga
5	6	6	6	11	10	10	2	2	1	fatiga
6	5	5	5	12	12	12	1	1	1	fatiga
7	7	7	7	8	8	8	1	1	1	fatiga

Fuente: Elaboración Propia.

Tabla 7. Resultados de las pruebas con la simulación de fatiga (sujeto 4).

Tiempo (Min)	Parpadeos Duración Normal			Parpadeos de Duración Larga			Bostezos			Estado
	Ind.	Realiz.	Det.	Ind.	Realiz.	Det.	Ind.	Realiz.	Det.	
1	15	14	13	0	0	1	0	0	0	no fatiga
2	15	15	15	0	0	0	0	0	0	no fatiga
3	15	15	15	0	0	0	0	0	0	no fatiga
4	9	8	8	8	8	8	2	2	0	posible fatiga
5	8	9	8	10	10	9	1	1	1	fatiga
6	8	8	8	11	10	10	1	1	1	fatiga
7	6	6	6	10	10	9	2	2	0	posible fatiga

Fuente: Elaboración Propia.

Tabla 8. Resultados de las pruebas con la simulación de fatiga (sujeto 5).

Tiempo (Min)	Parpadeos Duración Normal			Parpadeos de Duración Larga			Bostezos			Estado
	Ind.	Realiz.	Det.	Ind.	Realiz.	Det.	Ind.	Realiz.	Det.	
1	15	15	14	0	0	1	0	0	0	no fatiga
2	16	16	15	0	0	0	0	0	0	no fatiga
3	15	15	13	0	0	1	0	0	0	no fatiga
4	8	8	8	9	9	9	2	2	1	fatiga
5	7	8	8	11	11	11	2	2	2	fatiga
6	10	10	9	9	9	8	1	1	0	posible fatiga
7	7	7	7	11	11	10	1	1	1	fatiga

Fuente: Elaboración Propia.

Se calcularon la exactitud y la sensibilidad del sistema para evaluar su capacidad de clasificar correctamente los estados de fatiga. La exactitud mide el rendimiento general del sistema al determinar cuántas de las clasificaciones realizadas son correctas en comparación con el total, mientras que la sensibilidad se enfoca específicamente en la habilidad del sistema para detectar correctamente los casos de fatiga. Esta última es crucial, ya que el objetivo principal del sistema es alertar sobre la fatiga de manera efectiva. Estos cálculos se llevaron a cabo utilizando las fórmulas (6) y (7). Obteniendo un 85.7% de exactitud y una sensibilidad de 82.5%.

$$Exactitud = \frac{(VP + VN)}{(VP + VN + FP + FN)} \quad (6)$$

$$Sensibilidad = \frac{(VP)}{(VP + FN)} \quad (7)$$

Según los registros obtenidos durante la prueba, en los primeros 3 minutos, el estado de la persona fue 'no fatiga' en todos los casos, excepto uno, donde el sistema falló, generando un parpadeo adicional. Esto probablemente se debió a que el algoritmo simula parpadeos de manera 'natural', con intervalos de tiempo con variación de 1 o 2

segundos. Como resultado, se produjeron más de indicaciones de parpadeo en el sujeto 3 durante el tercer minuto.

Durante los 4 minutos restantes, en los que se simuló 'fatiga', el sistema logró detectar este estado en la mayoría de los casos. El resultado mínimo obtenido fue 'posible fatiga'. Las fallas ocurrieron principalmente por dos razones. La primera y más importante fue la imprecisión en la detección de bostezos, probablemente debido a que el clasificador no es lo suficientemente preciso. Como ya se ha mencionado anteriormente, una posibilidad a esto se debe a que el número de muestras positivas y negativas usadas para entrenarlo es limitado y sin una base de datos amplia, es difícil construir un clasificador que iguale el rendimiento de los pre-entrenados de OpenCV.

La segunda razón es que, aunque se instruyó a los participantes a realizar parpadeos más lentos cuando se encendiera la luz roja del indicador, la detección no siempre fue precisa. Los parpadeos duran apenas unos milisegundos, lo que dificulta que las personas calculen su duración exacta. Como resultado, algunos parpadeos largos indicados por el sistema fueron detectados como cortos, y viceversa. Esto afecta la capacidad del sistema para emitir un resultado correcto, ya que la estrategia de detección de fatiga requiere que la mayoría de los parpadeos en un minuto tengan una duración superior a 200 milisegundos.

6. CONCLUSIONES

Se logró implementar un conjunto de algoritmos capaces de realizar la detección de las manifestaciones de fatiga presentes en el rostro humano en el espectro visible, mediante uso de herramientas para visión artificial y con la facilidad para ejecutarlos dentro de una aplicación móvil.

Se integraron los algoritmos con una estrategia de detección de fatiga, a partir de parámetros extraídos de las manifestaciones presentes en los usuarios, frecuencia de parpadeo, duración de parpadeo y presencia de bostezos.

Se desarrolló un protocolo de pruebas exhaustivo para determinar el desempeño del aplicativo tomando en cuenta variaciones de iluminación, inclinación de cámara y cambio de fondos, usando un entorno semi controlado para definir alcances y limitaciones del sistema. Los datos obtenidos por estas pruebas revelaron que el dispositivo debe encontrarse en un entorno en el cual, la parte frontal del rostro se vea

iluminada, en donde el sensor de luz registre un valor entre 60 y 180 lux, evitando fuentes de luz que provengan de la parte posterior del usuario, estos valores están dentro del rango de iluminación que presentan lugares como habitaciones, cocinas, salas, oficinas. Las variaciones de ángulo del dispositivo no deben sobrepasar los 20 grados de inclinación vertical y 10 grados de inclinación horizontal, con respecto al rostro frontal del usuario.

La aplicación móvil fue desarrollada para el sistema operativo Android, en un dispositivo de gama media (motorola G60 con procesador Snapdragon 732G Octa-Core), lo que abarca la mayoría de los celulares del mercado actual, dando accesibilidad para un amplio público. Se utilizó el lenguaje de programación Java y la biblioteca OpenCV, especializada en visión artificial, garantizando que la aplicación sea lo suficientemente liviana para un procesamiento rápido y una interfaz ágil.

El sistema de detección de fatiga desarrollado mostró una exactitud del 85.7%, y sensibilidad del 82.5%, demostrando ser capaz de identificar correctamente el estado de fatiga en las personas evaluadas, dando una alerta temprana a pesar de las grandes limitaciones que enfrenta el sistema inherentes a las capacidades de procesamiento así como a variables del entorno. Esta solución tiene el potencial de ser aplicada en personas que trabajan en salud ocupacional, trabajos de oficina y la productividad, contribuyendo a la prevención de accidentes y a la mejora del bienestar general de los usuarios.

La implementación de los algoritmos usados en el desarrollo de esta aplicación, demuestra la viabilidad de utilizar herramientas de visión artificial en dispositivos móviles, y abre la puerta a futuras investigaciones de aplicaciones en diversos campos. Adicionalmente la estructura modular del código, facilita la integración de nuevas funcionalidades e incluso ser reutilizado y modificado para detección de otro tipo de aplicaciones como detección de objetos o emociones humanas.

7. TRABAJOS A FUTURO

Mantener un registro a largo plazo del estado de una persona es fundamental para comprender y abordar los riesgos asociados a la fatiga recurrente, la cual no solo afecta el rendimiento diario, sino que también puede tener graves consecuencias para la salud mental y física, incluyendo el desarrollo de trastornos como ansiedad y depresión. Incorporar una funcionalidad dentro del aplicativo para crear una base de datos dentro del teléfono y almacenar así, el historial del usuario a lo largo del tiempo ayudarían a anticipar y prevenir situaciones de fatiga crónica. Por otro lado, también permitiría brindarle al usuario la posibilidad de adecuar la aplicación de una forma personalizada modificando parámetros como la frecuencia de parpadeos, que ayudarán a darle más precisión al sistema.

Actualmente, hay una escasez de conjuntos de datos amplios y diversificados que contengan registros del estado de fatiga, por lo que crear una base de datos en entornos regulados, puede brindar datos más confiables y consistentes para el desarrollo y la validación de algoritmos para su detección. Los ambientes controlados permiten manipular variables, lo que facilita la observación y el registro de los efectos de estas variables. Esto puede ayudar a mejorar la comprensión de los factores que contribuyen a la fatiga y a optimizar los métodos de detección.

Investigar algoritmos con redes neuronales ofrecidos por la biblioteca Tensor Flow Lite para detección de los signos de fatiga, esta librería ofrece capacidades avanzadas de aprendizaje automático y diseñada específicamente para dispositivos con recursos limitados, lo que puede resultar ser muy efectivo para detectar patrones complejos como la fatiga, incorporando más indicadores sin que haya repercusiones en el rendimiento del dispositivo.

8. BIBLIOGRAFÍA

- [1] Universidad Complutense de Madrid, “FATIGA LABORAL: Conceptos y Prevención”, Servicio de Prevención de Riesgos Laborales y Medicina del Trabajo, España, 2016. [En línea]. Disponible en: <https://www.lineaprevencion.com/recursos/fatiga-laboral-conceptos-y-prevencion> [Accedido: 21-sep-2024]
- [2] O. Pineda et al, “Factores asociados a la fatiga laboral en el personal administrativo de los Hospitales: Vicente Corral Moscoso y Homero Castanier Creso”, Universidad del Azuay, Ecuador, 2020. [En línea]. Disponible en: <https://dspace.uazuay.edu.ec/handle/datos/9702> [Accedido: 17-sep-2024]
- [3] A. Hervás y F. Tortosa, “Un estudio piloto sobre el efecto diferencial de la fatiga por conducción en personas mayores”, Universitas Psychologica, 2011 [En línea]. Disponible en: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1657-92672011000300020 [Accedido: 13-may-2024]
- [4] B. P-Perozzi, “Fatiga: historia, neuroanatomía y características psicopatológicas. Una revisión de la Literatura”, Neuropsiquiatr, vol. 81, no. 3, pp. 174–182, 2018.
- [5] J. P. Arias, “Detección de signos de fatiga en humanos utilizando imágenes del rostro”, Trabajo de Grado, Universidad del Valle, Cali, 2020.
- [6] R. J. Moreno, “Sistema de detección de nivel de cansancio en conductores mediante técnicas de visión por computador”, M.S. Tesis, Universidad Nacional de Colombia, Bogotá, 2011.
- [7] M. Palma y Yeferson T. “Aplicación móvil para la detección de somnolencia de un conductor aplicando visión artificial”, Universidad Nacional de Loja, Ecuador, 2016.
- [8] B. A. Cardozo y B. A. Torres, "Aplicación para teléfono móvil con sistema operativo Android que permita detectar somnolencia y emitir una alarma sonora a conductores de automóvil mediante procesamiento de imágenes", Trabajo de grado, Universidad Distrital “Francisco José de Caldas”, Bogotá, 2019.
- [9] O. Berkati and M. N. Srifi, "Predict Driver Fatigue Using Facial Features," International Symposium on Advanced Electrical and Communication Technologies (ISAECT), Rabat, Morocco, 2018, pp. 1-5, doi: 10.1109/ISAECT.2018.8618740.

- [10] M. Zhang et al., "Research on tiredness detection method based on fusion of face image and EEG information," IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC), Greenville, SC, USA, 2021, pp. 396-400, doi: 10.1109/ICFTIC54370.2021.9647277.
- [11] E. Essel, F. Lacy, W. Elmedany, F. Albalooshi and Y. Ismail, "Driver Drowsiness Detection Using Fixed and Dynamic Thresholding," International Conference on Data Analytics for Business and Industry (ICDABI), Sakhr, Bahrain, 2022, pp. 552-557, doi: 10.1109/ICDABI56818.2022.10041670.
- [12] A. Shome, A. Jain, V. Choudhary, P. Garg, R. Chopra and D. Vii, "Driver Drowsiness Detection System using DLib," 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2022, pp. 193-197, doi: 10.1109/ICACITE53722.2022.9823848.
- [13] S. Gaikwad, U. Patil and M. Subhedar, "Driver Assistance Systems with Driver Drowsiness Detection Using Haar-Cascade Algorithm," 3rd International Conference on Advancement in Electronics & Communication Engineering (AECE), GHAZIABAD, India, 2023, pp. 5-10, doi: 10.1109/AECE59614.2023.10428159.
- [14] P. Sudarshan, V. Bhardwaj and Virender, "Real-time Driver Drowsiness Detection and Assistance System using Machine Learning and IoT," 8th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2023, pp. 1128-1132, doi: 10.1109/ICCES57224.2023.10192831.
- [15] A. Ritesh, N. Jagatia and P. Deshmukh, "Driver Drowsiness Detection Using Machine Learning," 6th International Conference on Advances in Science and Technology (ICAST), Mumbai, India, 2023, pp. 100-105, doi: 10.1109/ICAST59062.2023.10455018.
- [16] CESVI COLOMBIA, "Fatiga Al Volante: Una De Las Principales Causas De Accidentalidad Vial", Comunicado, Colombia, 2022. [En línea]. Disponible en: <https://www.cesvicolombia.com/fatiga-al-volante-una-de-las-principales-causas-de-accidentalidad-vial/> [Accedido: 18-sep-2024]
- [17] "Sueño y fatiga, y su influencia al volante", *Fundación MAPFRE*. [En línea]. Disponible en: <https://www.fundacionmapfre.org/educacion-divulgacion/seguridad-vial/movilidad-segura-salud/temas-conduccion-segura/enfermedades-neurologicas/sueno-fatiga/> [Accedido: 19-msep-2024]

- [18] “Lesiones y accidentes laborales más comunes”. *Bodytrak*. [En línea]. Disponible en:
<https://bodytrak.co/es/news/guia-de-las-lesiones-y-accidentes-laborales-mas-comunes/#Conclusión> [Accedido: 16-sep-2024]
- [19] B. Carton, J. Mongardini y Y. Li, " Smartphones Drive New Global Tech Cycle, but Is Demand Peaking?", 2018. [En línea] Disponible en:
<https://blogs.imf.org/2018/02/08/smartphones-drive-new-global-tech-cycle-but-is-demand-peaking/> [Accedido: 05-mar-2022]
- [20] M. Shanahan y K. Bahia, “The State of Mobile Internet Connectivity 2023”, GSMA, Reino Unido, 2023. [En línea]. Disponible en:
https://www.gsma.com/r/wp-content/uploads/2023/10/The-State-of-Mobile-Internet-Connectivity-Report-2023.pdf?utm_source=website&utm_medium=button&utm_campaign=somic23 [Accedido: 19-sep-2024]
- [21] L. S. Aaronson *et al.*, “Defining and measuring fatigue”, *Image J Nurs Sch*, vol. 31, pp. 50, 1999.
- [22] Biblioteca Nacional de Medicina, “Fatiga”, 2018. [En línea]. Disponible en:
<https://medlineplus.gov/spanish/ency/article/003088.htm#:~:text=La%20fatiga%20es%20una%20falta,o%20a%20la%20falta%20de%20sue%C3%B1o> [Accedido: 21-sep-2024]
- [23] M. R. Wasserman, “Fatiga”, 2017. [En línea]. Disponible en
<https://www.msdmanuals.com/es-co/hogar/temas-especiales/s%C3%ADntomas-inepec%C3%ADficos/fatiga> [Accedido: 16-sep-2024]
- [24] P. P. Caffier, U. Erdmann y P. Ullsperger, “Experimental evaluation of eye-blink parameters as a drowsiness measure,” *Eur. J. Appl. Physiol.*, vol. 89, no. 3, pp. 319–325, 2003.
- [25] G Barbato *et al*, “Diurnal variation in spontaneous eye-blink rate,” *Psychiatry Research*, 2000, doi: 10.1016/s0165-1781(00)00108-6.
- [26] Stern, J A *et al*. “The endogenous eyeblink,” *Psychophysiology* vol. 21,1 (1984): 22-33. doi:10.1111/j.1469-8986.1984.tb02312.x
- [27] A. Carpenter, “The rate of blinking during prolonged visual search,” *Journal of Experimental Psychology*, Vol 38(5), Oct 1948.

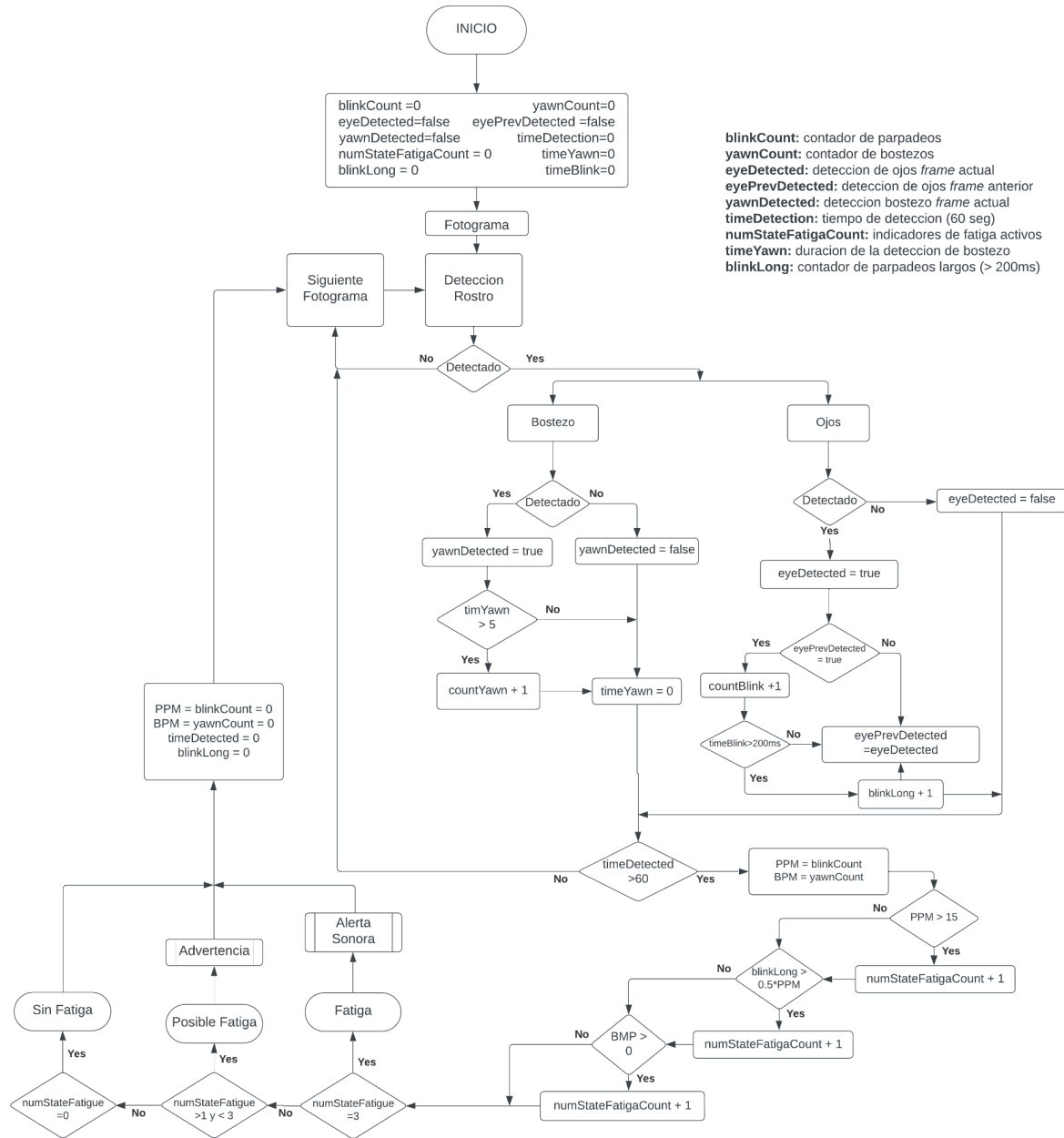
- [28] C. Hoffman, "Luckiesh and Moss on reading illumination". *Journal of Applied Psychology*, 1947. doi: <https://doi.org/10.1037/h0057842>
- [29] D. M. Noboa *et al.*, "Frecuencia de signos y síntomas de fatiga física y mental en médicos de un Hospital de segundo nivel de la ciudad de Quito – Ecuador", CD, vol. 3, n.º 3.2, pp. 277-289, ago. 2019. Doi: <https://doi.org/10.33262/cienciadigital.v3i3.2.756>
- [30] S. Abtahi, B. Hariri and S. Shirmohammadi, "Driver drowsiness monitoring based on yawning detection," IEEE International Instrumentation and Measurement Technology Conference, Hangzhou, China, 2011, pp. 1-4, doi: 10.1109/IMTC.2011.5944101.
- [31] A. E. Sulavko and S. S. Zhumazhanova, "Human psychophysiological state recognition based on analysis of thermograms of face and neck regions," 2017 Dynamics of Systems, Mechanisms and Machines (Dynamics), Omsk, Russia, 2017, pp. 1-7, doi: 10.1109/Dynamics.2017.8239515.
- [32] A. Hassoumi *et al.*, "Thermal Imaging of the Face: Mental Workload Detection in Flight Simulator," In: Neuroergonomics and Cognitive Engineering, AHFE International, vol 42, 2022. Doi: <http://doi.org/10.54941/ahfe1001822>
- [33] G. Sikander and S. Anwar, "A Novel Machine Vision-Based 3D Facial Action Unit Identification for Fatigue Detection," in IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 5, pp. 2730-2740, May 2021, doi: 10.1109/TITS.2020.2974263
- [34] K. Cen, "Study of Viola-Jones Real Time Face Detector," 2016. [En línea] Disponible en: <https://www.semanticscholar.org/paper/Study-of-Viola-Jones-Real-Time-Face-Dete>
- [35] S. Lee. "Understanding Face Detection with the Viola-Jones Object Detection Framework," Medium. [En línea]. Disponible en : <https://towardsdatascience.com/understanding-face-detection-with-the-viola-jones-object-detection-framework-c55cc2a9da14> [Accedido: 20-nov-2022]
- [36] C. J. Pardo, "Detección de Rostros basado en Filtros Haar + Adaboost", [En línea]. Disponible en: <https://carlosjuliopardoblog.wordpress.com/2017/05/12/filtros-haar-deteccion-de-rostros/>. [Accedido: 28-nov-2022]

- [37] Ghosh, M. et. al, "Face Detection and Extraction Using Viola–Jones Algorithm," Computational Advancement in Communication, Circuits and Systems. 2021. [En línea]. Disponible en: https://link-springer-com.bd.univalle.edu.co/chapter/10.1007/978-981-16-4035-3_9?fromPaywallRec=true [Accedido: 19-sep-2024]
- [38] A. Cardona y F. Pineda, "Reconocimiento de rostros en tiempo real sobre dispositivos móviles de bajo costo", *Lámpsakos*, vol. 1, n.º 20, pp. 30–39, 2019.
- [39] M Palaniappan, K. R. Sowmia and S. Aravindkumar, "Real Time Fatigue Detection Using Shape Predictor 68 Face Landmarks Algorithm," *International Conference on Innovative Trends in Information Technology (ICITIIT)*, Kottayam, India, 2022, pp. 1-5, doi: 10.1109/ICITIIT54346.2022.9744142.
- [40] T. Soukupová, and C. Jan, "Real-Time Eye Blink Detection using Facial Landmarks," Czech technical university in Prague, Rimske Toplice, Slovenia, Feb 2016.
- [41] i-bug , "i-bug, intelligent behaviour understanding group," 2013. [En línea]. Disponible en: <https://ibug.doc.ic.ac.uk/resources/300-W/> [Accedido: 19-sep-2024]
- [42] M. Antúnez, "Algoritmos de detección de objetos para la detección y seguimiento de ojos", Trabajo de grado, Universidad Politécnica de Catalunya, Departament de Teoria del Senyal i Comunicacions (2019).
- [43] Tutor de programación, "Detección de líneas y círculos usando la transformada de Hough con OpenCV", 2017 [En línea]. Disponible en: <https://acodigo.blogspot.com/2017/09/deteccion-de-lineas-y-circulos-usando.html> [Accedido: 17-sep-2024]
- [44] M. Wang, F. Zheng and J. Lu, "Face Detection with Improved Faster-R-CNN," *4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, Hamburg, Germany, 2022, pp. 84-91, doi: 10.1109/AIAM57466.2022.00024.
- [45] O. Cakiroglu, C. Ozer and B. Gunsel, "Design of a Deep Face Detector by Mask R-CNN," *27th Signal Processing and Communications Applications Conference (SIU)*, Sivas, Turkey, 2019. doi: 10.1109/SIU.2019.8806447.
- [46] S. Moraleda, "Análisis del comportamiento de la mirada frente a un ordenador", E.T.S. de Ingenieros Informáticos, Universidad politécnica de Madrid, 2018. [En

- [línea]. Disponible en: https://oa.upm.es/51633/1/TFM_SARA_MORALEDA_MORENO.pdf [Accedido: 18-sep-2024]
- [47] U. Trutschel et al., "PERCLOS: An Alertness Measure of the Past," in Proceedings of the 6th International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design: driving assessment 2011, 2011, pp. 172–179.
- [48] M F. Melo, "El mapa mundial de Android e iOS", Statista, Industria y consumo mundial de smartphones, 2024 [En línea]. Disponible en: <https://es.statista.com/grafico/29620/sistema-operativo-movil-con-la-mayor-cuota-de-mercado-por-pais/#:~:text=Seg%C3%BAAn%20cifras%20de%20junio%20de,representa%20el%2027%2C19%25> [Accedido: 19-sep-2024]
- [49] OPENCV, "OPENCV, Open Source Computer Vision Library", 2022. [En línea]. Disponible en: <https://opencv.org/about/>
- [50] Dlib, "Dlib C++ Library", 2022. [En línea]. Disponible en: <http://dlib.net/>
- [51] Shabnam Abtahi, Mona Omidyeganeh, Shervin Shirmohammadi, Behnoosh Hariri, August 1, 2020, "YawDD: Yawning Detection Dataset", IEEE Dataport, doi: <https://dx.doi.org/10.21227/e1qm-hb90>
- [52] K. Oakes, "Hechos Sascinantes Sobre Tus Ojos", BuzzFeed, 2015 [En línea] Disponible en: <https://difundir.org/2015/10/18/13-hechos-fascinantes-sobre-tus-ojos-que-te-haran-sentir-una-mayor-admiracion-por-ellos/> [Accedido: 20-sep-2024]
- [53] Shutterstock "Repertorio de Fotografías" [En línea] Disponible en: <https://www.shutterstock.com/es/image-photo/image-unshaved-man-30s-bristle-casual-1351222946> [Accedido: 21-sep-2024]
- [54] J. Fraile Vilarrasa y A. Gago Calderon, "Iluminacion con tecnologia led", Paraninfo S.A, pp 97-115, 2012.
- [55] Reza Ghoddoosian, Marnim Galib, Vassilis Athitsos, "A Realistic Dataset and Baseline Temporal Model for Early Drowsiness Detection". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019. Disponible en: <https://sites.google.com/view/utarlidd/home>

ANEXOS

Anexo A. Diagrama del algoritmo simplificado para la detección de fatiga.



Fuente: Elaboración propia

Anexo B. Casos de Uso Real para el diseño de la aplicación.

Título: Iniciar la detección de fatiga	Código: CUR - 01	Página: 1 de 1
---	-----------------------------------	---------------------------------

REVISIÓN HISTÓRICA			
Rev.	Descripción del Cambio	Autor	Fecha
001	Construcción del caso 1	Victor Villota Quintero	2023-02-26

INFORMACIÓN GENERAL	
Actores:	Usuario, Aplicación móvil
Propósito:	La aplicación monitorea los parpadeos y bostezos del usuario utilizando la cámara del dispositivo y los clasificadores de OpenCV
Resumen:	La aplicación detecta el rostro, boca y ojos de la persona identificando parpadeos y bostezos, llevando un conteo de estos parámetros, cada 60 segundos, y mostrándose en la interfaz.
Tipo:	Real

Curso Normal de los Eventos	
Acción de los Actores – Usuario	Acción de los Actores - Aplicación móvil
1. El usuario presiona el botón "Iniciar" en la interfaz.	
	2. La aplicación activa los detectores de rostros, ojos y boca.
	3. La aplicación comienza a analizar el flujo de vídeo en busca de signos de fatiga (parpadeos y bostezos).
	4. Los contadores de parpadeos y bostezos se actualizan en tiempo real en la interfaz.
	5. La aplicación móvil presenta el resultado del estado de la persona, pasado 1 minuto de iniciar el proceso de detección de fatiga.
	6. Los contadores de cada parámetro analizado se reinician.

Título: Detener la detección de fatiga	Código: CUR – 02	Página: 1 de 1
---	-----------------------------------	---------------------------------

REVISIÓN HISTÓRICA			
Rev.	Descripción del Cambio	Autor	Fecha
001	Construcción del caso 2	Victor Villota Quintero	2023-02-26

INFORMACIÓN GENERAL	
Actores:	Usuario, Aplicación móvil
Propósito:	Detener la funcionalidad del aplicativo en cualquier momento que el usuario lo desee.
Resumen:	La aplicación detiene la captura de fotogramas y por consiguiente, la detección de rostro, ojos y boca en el momento en el que el usuario lo ordene.
Tipo:	Real

Curso Normal de los Eventos	
Acción de los Actores – Usuario	Acción de los Actores - Aplicación móvil
1. El usuario presiona el botón "Detener" en la interfaz.	
	2. La aplicación detiene las capturas de la cámara.
	3. Se detiene la detección de rostro, ojos y boca por parte de los clasificadores de OpenCV.
	3. Se restablecen los contadores de parpadeos y bostezos.
	5. La aplicación móvil queda en espera de que se vuelva a activar la detección de fatiga.

Título: Alerta de Fatiga	Código: CUR – 03	Página: 1 de 1
---	-----------------------------------	---------------------------------

REVISIÓN HISTÓRICA			
Rev.	Descripción del Cambio	Autor	Fecha
001	Construcción del caso 3	Victor Villota Quintero	2023-03-11
INFORMACIÓN GENERAL			
Actores:	Aplicación móvil		
Propósito:	Cuando se detecta un estado de fatiga, la aplicación emite una alerta sonora y notifica al usuario.		
Resumen:	Luego de iniciar la detección de fatiga y el sistema realiza la monitorización de los parpadeos y bostezos, la aplicación determina el estado de la persona y procede a notificar al usuario este estado cada minuto desde el inicio de la detección.		
Tipo:	Real		

Curso Normal de los Eventos	
	Acción de los Actores - Aplicación móvil
	1. La aplicación monitorea continuamente los parpadeos y bostezos del usuario.
	2. Se verifican los contadores de parpadeos y bostezos.
	3. Se determina que se cumplen las condiciones de fatiga (alta frecuencia de parpadeos, duración prolongada de parpadeos, presencia de bostezos)
	4. Se emite una alerta visual de estado de fatiga.
	5. Se emite una alerta sonora para notificar al usuario, el estado de fatiga que presenta.
	6. Se restablecen los contadores de parpadeos y bostezos y se reinicia el monitoreo del usuario.
Curso Alternativo de los Eventos	
	Acción de los Actores - Aplicación móvil
	1. La aplicación monitorea continuamente los parpadeos y bostezos del usuario.
	2. Se verifican los contadores de parpadeos y bostezos.
	3. Se determina que no se cumplen las condiciones de fatiga.
	4. La aplicación muestra un mensaje en la pantalla de "no fatiga" o "posible fatiga" dependiendo de las manifestaciones de fatiga encontradas.
	6. Se restablecen los contadores de parpadeos y bostezos y se reinicia el monitoreo del usuario.

Título: Activar/Desactivar Simulación de Estado de Fatiga	Código: CUR – 04	Página: 1 de 1
--	-----------------------------------	---------------------------------

REVISIÓN HISTÓRICA			
Rev.	Descripción del Cambio	Autor	Fecha
001	Construcción del caso 4	Victor Villota Quintero	2023-11-14

INFORMACIÓN GENERAL	
Actores:	Usuario, Aplicación móvil
Propósito:	Activar indicadores de parpadeos y bostezos generados por la aplicación para simular las manifestaciones de fatiga referentes a los parpadeos y bostezos de la persona, para realizar las pruebas del sistema final.
Resumen:	La aplicación permite activar y desactivar la simulación de estado de fatiga a partir de una "checkbox", usado para la realización de pruebas del sistema final, en el cual, la aplicación emite destellos de luz para indicar los momentos en los cuales el usuario debe realizar parpadeos normales, parpadeos de larga duración y bostezos.
Tipo:	Real

Curso Normal de los Eventos	
Acción de los Actores – Usuario	Acción de los Actores - Aplicación móvil
1. El usuario marca la casilla "Simulación de Fatiga" en la interfaz.	
	2. La aplicación ajusta sus parámetros internos para simular condiciones de fatiga.
	3. La aplicación móvil queda en espera de que se active la detección de fatiga para iniciar los indicadores de luz que simulan la fatiga.
Curso Alternativo de los Eventos	
Acción de los Actores – Usuario	Acción de los Actores - Aplicación móvil
1. El usuario desmarca la casilla "Simulación de Fatiga" en la interfaz.	
	2. La aplicación ajusta sus parámetros internos para simular condiciones de no fatiga.
	3. La aplicación móvil queda en espera de que se active la detección de fatiga para iniciar los indicadores de luz que simulan la no fatiga.

Anexo C. Tablas de porcentaje de detección con Variaciones de Iluminación.

Tabla C.1. Detección del Rostro con variaciones en la luz principal

Detección del Rostro con variaciones en la luz principal						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	65	94.8	99.3	99.1	94.8	98.8
325	100	99.9	99.8	98.9	96.1	99.1
500	121	99.9	99.9	98.6	99.6	99.8
700	167	99.8	99.7	99.7	99.1	99.4
900	185	99.5	99.6	99.3	98.8	98.3
1150	203	99.7	99.3	99.4	97.7	98.7
Luz de fondo y de relleno constantes (400 lm).						

Fuente: Elaboración propia.

Tabla C.2. Detección del Rostro con variaciones en la luz de fondo

Detección del Rostro con variaciones en la luz de fondo						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	75	87.9	76.8	78.1	81.7	84.8
325	120	88.3	76.3	79.6	83.8	81.9
500	134	77.8	75.4	85.9	77.1	77.1
700	172	68.3	74.6	73.1	74.8	75.3
900	194	60.4	71.5	63.2	70.9	74.6
1150	212	62.5	70.6	61.5	71.4	71.3
Luz principal y de relleno constantes (400 lm).						

Fuente: Elaboración Propia.

Tabla C.3. Detección del Rostro con variaciones en la luz de fondo

Detección del Rostro con variaciones en la luz de relleno						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	52	98.7	94.8	99.7	97.1	93.1
325	82	99.1	98.9	98.8	93.7	94.8
500	115	98.7	99.2	99.1	99.4	97.6
700	152	98.6	98.9	99.4	98.9	98.9
900	172	97.3	99.6	98.6	99.3	99.1
1150	199	99.8	99.7	99.3	99.6	99.5
Luz principal y de fondo constantes (400 lm).						

Fuente: Elaboración propia.

Tabla C.4. Detección de Ojos con variaciones en la luz Principal

Detección del Ojos con variaciones en la luz principal						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	65	86.7	90.5	84.3	73.5	73.9
325	100	81.5	94.2	84.8	94.6	95.7
500	121	98.7	96.7	94.4	94.3	94.6
700	167	99.1	97.8	97.6	98.2	97.3
900	185	96.8	96.3	97.3	97.9	95.8
1150	203	97.7	95.2	98.7	97.3	95.1

Luz de fondo y de relleno constantes (400 lm).

Fuente: Elaboración propia.

Tabla C.5. Detección de Ojos con variaciones en la luz de fondo

Detección del Ojos con variaciones en la luz de fondo						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	75	75.9	76.1	70.7	75.8	84.1
325	120	70.6	74.2	73.4	80.3	71.9
500	134	65.8	75.1	71.8	71.4	75.3
700	172	55.7	71.3	64.8	72.6	74
900	194	54.3	65.7	57.3	68.7	73.2
1150	212	50.1	68.6	50.1	63.3	70.2

Luz principal y de relleno constantes (400 lm).

Fuente: Elaboración propia.

Tabla C.6. Detección de Ojos con variaciones en la luz de relleno

Detección del Ojos con variaciones en la luz de relleno						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	52	83.6	84.2	91.8	97.4	92.8
325	82	88.7	96.3	93.2	94.7	94.8
500	115	84.5	97.4	94.6	93.1	94.3
700	152	97.7	98.3	95.4	96.8	97.8
900	172	98.1	97.6	97.2	97.2	97.9
1150	199	96.8	98.6	96.3	98.3	98.1

Luz principal y de fondo constantes (400 lm).

Fuente: Elaboración propia.

Tabla C.7. Detección de Boca con variaciones en la luz Principal

Detección del Boca con variaciones en la luz principal						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	65	64.5	68	80.2	90.7	86.3
325	100	71.3	97	96.1	91.3	87.1
500	121	86.7	96.1	90.7	93.5	85.6
700	167	89.2	89.1	86.8	85	79.3
900	185	95.5	95.7	78.3	84.7	68.5
1150	203	87.1	70.3	64.6	83.2	60.5
Luz de fondo y de relleno constantes (400 lm).						

Fuente: Elaboración propia.

Tabla C.8. Detección de Boca con variaciones en la luz de fondo

Detección del Boca con variaciones en la luz de fondo						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	75	67.2	61.7	70.4	76.4	83.2
325	120	75.7	63.8	72.5	73.9	69.3
500	134	60.8	60.5	60.8	71.6	80.5
700	172	58.7	54.7	57.6	67.1	66.3
900	194	45.9	58.3	47.9	63.5	67.8
1150	212	40.2	50.2	48.3	57.3	61.2
Luz principal y de relleno constantes (400 lm).						

Fuente: Elaboración propia.

Tabla C.9. Detección de Boca con variaciones en la luz de relleno

Detección del Boca con variaciones en la luz de relleno						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	52	87.3	88.6	94.1	87.1	96.3
325	82	83.7	90.2	96.6	95.8	95.5
500	115	94.2	93.3	89.7	95.2	90.6
700	152	79.3	83.1	72.8	72	79.3
900	172	77.2	73.7	69.3	70.8	84.2
1150	199	74.2	69.3	65.4	76.3	74.7
Luz principal y de fondo constantes (400 lm).						

Fuente: Elaboración propia.

Tabla C.10. Detección del Rostro con variaciones en la luz principal (sin luz de fondo)

Detección del Rostro con variaciones en la luz principal (sin luz de fondo)						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	54	99.4	98.8	94.2	97.9	99.7
325	67	99.5	99.6	98.8	99.5	99.9
500	110	98.9	99.4	99.1	99.6	99.6
700	135	99.8	99.8	98.9	99.7	99.5
900	174	99.6	99.7	99.5	99.5	99.7
1150	191	99.9	99.6	99.4	99.8	99.6
Luz de relleno constante (400 lm).						

Fuente: Elaboración propia.

Tabla C.11. Detección de Ojos con variaciones en la luz principal (sin luz de fondo)

Detección de Ojos con variaciones en la luz principal (sin luz de fondo)						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	54	95.4	95.1	96.4	72.8	74.1
325	67	94.9	97.3	97.6	94.8	87.2
500	110	97.8	97.8	97.2	97.3	94.8
700	135	97.3	96.4	97.1	97.9	97.3
900	174	98.4	99.7	97.3	98.1	98.3
1150	191	98.7	99.1	98.6	98.6	98.8
Luz relleno constante (400 lm).						

Fuente: Elaboración propia.

Tabla C.12. Detección de Boca con variaciones en la luz principal (sin luz de fondo)

Detección de Boca con variaciones en la luz principal (sin luz de fondo)						
Luz emitida aprox. (lm)	Sensor de luz (lx)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
175	54	91.4	93.9	84.1	90.9	93.7
325	67	93.8	96.4	95.4	94.3	93.1
500	110	97.1	96.7	96.3	90.3	95.7
700	135	96.3	95.1	92.8	85.5	93.8
900	174	87.6	89.7	73.5	70.4	74.1
1150	191	83.3	88.2	71.9	75.3	73.3
Luz principal constante (400 lm).						

Fuente: Elaboración propia.

Anexo D. Tablas de porcentaje de detección con Variaciones de Inclinación de Cámara.

Tabla D.1. Detección del rostro con inclinación en el eje horizontal.

Deteccion del Rostro con (Inclinacion en el eje horizontal)					
angulos (°)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
-30	78.9	96.3	95.3	93.6	93.6
-20	96.8	95.8	70.6	96.8	94.9
-10	97.9	98.1	98.7	97.4	98.7
0	99.5	99.4	96.7	99.1	99.5
10	94.8	99.9	94.8	99.8	99.9
20	96.4	97.9	87.4	99.6	97.4
30	84.9	94.7	85.6	97.3	96.4
Luz principal (500 lm) y luz de relleno (400 lm)					

Fuente: Elaboración propia.

Tabla D.2. Detección de ojos con inclinación en el eje horizontal.

Deteccion del Ojos con (Inclinacion en el eje horizontal)					
angulos (°)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
-30	40.3	76.8	76.3	88.5	91.1
-20	83.7	92.1	95.8	78.9	92.6
-10	95.6	95.4	96.1	96.3	93.8
0	96.3	96.5	98.3	89.6	97.6
10	97.8	97.6	97.7	97.4	96.3
20	93.8	97.3	75.9	91.6	94.9
30	58.9	83.1	75.2	86.8	78.7
Luz principal (500 lm) y luz de relleno (400 lm)					

Fuente: Elaboración propia.

Tabla D.3. Detección de Boca con inclinación en el eje horizontal.

Deteccion del Boca con (Inclinacion en el eje horizontal)					
angulos (°)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
-30	36.7	50.3	40.3	52.5	53.6
-20	40.5	81.3	52.9	76.8	64.9
-10	93.6	86.1	84.3	80.9	76.3
0	96.1	96.8	94.6	84.1	74.1
10	98.3	93.3	83.7	84.6	93.6
20	94.7	91.8	68.3	64.3	91.7
30	75.3	61.8	64.1	40.8	65.3
Luz principal (500 lm) y luz de relleno (400 lm)					

Fuente: Elaboración propia.

Tabla D.4. Detección del rostro con inclinación en el eje vertical.

Deteccion del Rostro con (Inclinacion en el eje vertical)					
angulos (°)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
-30	83.8	84.1	60.3	61.4	77.2
-20	93.8	96.7	71.1	93.1	96.7
-10	98.9	98.7	98.6	98.7	98.9
0	99.5	99.6	99.2	99.5	99.6
10	98.9	99.7	99.3	99.8	99.8
20	96.3	98.9	98.7	94.9	99.4
30	81.7	92.7	91.2	71.1	86.1
Luz principal (500 lm) y luz de relleno (400 lm)					

Fuente: Elaboración propia.

Tabla D.5. Detección de ojos con inclinación en el eje vertical.

Deteccion del Ojos con (Inclinacion en el eje vertical)					
angulos (°)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
-30	58.1	40.3	78.9	58	45.9
-20	80.7	51.2	81.2	61.3	57.4
-10	83.1	94.5	94.5	87.6	89.3
0	94.3	98.1	96.8	98.3	97.3
10	95.3	97.6	93.2	97.6	96.8
20	83.9	95.7	83.3	95.8	97.1
30	63.1	77.1	76	77.9	64.7
Luz principal (500 lm) y luz de relleno (400 lm)					

Fuente: Elaboración propia.

Tabla D.6. Detección de Boca con inclinación en el eje vertical.

Deteccion del Boca con (Inclinacion en el eje vertical)					
angulos (°)	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5
-30	15.3	42.8	26.3	40.3	60.1
-20	37.1	63.9	45.1	76.2	75.3
-10	50.3	81.3	59.9	72.9	78.9
0	93.9	98.8	82.2	83.6	84.4
10	95.6	97.6	93.6	89.1	92.1
20	96.8	88.7	71.8	96.3	90.8
30	75.8	95.1	68.9	70.8	74.1
Luz principal (500 lm) y luz de relleno (400 lm)					

Fuente: Elaboración propia.

Anexo E. Código de la clase principal.

```
public class MainActivity extends CameraActivity {

    private SensorManager sensorManager;
    private Sensor lightSensor;
    private SensorEventListener lightEventListener;
    private double valueSensorLigth;
    CameraBridgeViewBase cameraBridgeViewBase;
    CascadeClassifier faceCascade,eyeCascade,yawnCascade;
    private TextView blinkCountTextView,blinkTimeTextView,    yawnCountTextView,timeTextView,stateTextView,
    sensorTextView;
    private Button startButton,stopButton;
    private boolean start = false, stop = false, simFat;
    private View indBlink,indYawn;
    private CheckBox fatCheck;
    private Handler handler;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        blinkCountTextView = findViewById(R.id.blinkCountTextView);
        yawnCountTextView = findViewById(R.id.yawnCountTextView);
        cameraBridgeViewBase = findViewById(R.id.cameraView);
        blinkTimeTextView = findViewById(R.id.blinkTimeTextView);
        timeTextView = findViewById(R.id.timeTextView);
        sensorTextView = findViewById(R.id.sensorTextView);
        stateTextView = findViewById(R.id.stateTextView);
        startButton = findViewById(R.id.startButton);
        stopButton = findViewById(R.id.stopButton);
        fatCheck = findViewById(R.id.simFat);
        indBlink = findViewById(R.id.indicator_light_view);
        indYawn = findViewById(R.id.indicator_light_view2);
        handler = new Handler();

        getPermissionFunction();
        lightSensorFunction();
        stateSelection();
        cameraBridgeViewBase.setCameraIndex(1); // Configuración de la
        Detection d = new Detection(yawnCountTextView, blinkCountTextView, blinkTimeTextView,
            timeTextView, sensorTextView, stateTextView) {

            @Override
            public void showTitle(TextView textView, String text, long count) {runOnUiThread(() -> textView.setText(text+ count));}

            @Override
            public void showState(TextView textView, String text) { runOnUiThread(() -> textView.setText(text));}

            @Override
            public void alertFatiga() {
                Vibrator vibrator = (Vibrator) getSystemService(VIBRATOR_SERVICE);
                if (vibrator != null) { vibrator.vibrate(500);}

                Toast.makeText(MainActivity.this, "¡ALERTA!", Toast.LENGTH_SHORT).show();
                MediaPlayer mediaPlayer = MediaPlayer.create(MainActivity.this, R.raw.alertasound);
                mediaPlayer.start();
            }
        };
    }
}
```

```

    }
};
cameraBridgeViewBase.setCvCameraViewListener(d);
/* Se verifica la carga de la libreria OpenCV */
if (OpenCVLoader.initDebug()) {
    Log.d("LOADED", "success");
    cameraBridgeViewBase.enableView();
} else {Log.d("LOADED", "error");
}

        faceCascade = CascadeClassifierLoader.loadCascade(this, R.raw.haarcascade_frontalface_alt,
"haarcascade_frontalface_alt.xml");
        eyeCascade = CascadeClassifierLoader.loadCascade(this, R.raw.haarcascade_eye, "haarcascade_eye.xml");
        yawnCascade = CascadeClassifierLoader.loadCascade(this, R.raw.haarcascade_yawn, "haarcascade_yawn.xml");
        d.faceCascade = faceCascade; d.eyeCascade = eyeCascade; d.yawnCascade = yawnCascade;
        pressButtons(d);
    }

private void getPermissionFunction() {
    if (checkSelfPermission(Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
        requestPermissions(new String[]{Manifest.permission.CAMERA}, 101);
    }
}

private void pressButtons(Detection d){
    StateSimulation state = new StateSimulation(indBlink, indYawn, handler);
    startButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            start = true; Toast.makeText(MainActivity.this, "Deteccion Fatiga Iniciada ", Toast.LENGTH_SHORT).show();
            d.start = start;
            state.setSimFat(simFat);
            state.setSimState(true);
            state.startDelayBlink();
            if(simFat){state.startDelayYawn();}
        }
    });
    stopButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            stop = true; // Establecer la variable en true
            Toast.makeText(MainActivity.this, "Deteccion Fatiga Finalizada", Toast.LENGTH_SHORT).show();
            d.stop = stop;
            state.setSimState(false);
        }
    });
}

private void lightSensorFunction(){

    sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    lightSensor = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);

    if (lightSensor == null) {
        Toast.makeText(this, "El dispositivo no tiene sensor de luz!", Toast.LENGTH_SHORT).show();
        valueSensorLigth = 40;
    }
}

```

```

        runOnUiThread() -> sensorTextView.setText(""+valueSensorLigth));
    }else{
        lightEventListener = new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent sensorEvent) {
                runOnUiThread() -> sensorTextView.setText(""+Math.round(sensorEvent.values[0]));}
            @Override
            public void onAccuracyChanged(Sensor sensor, int i) {
            }
        };
        sensorManager.registerListener(lightEventListener, lightSensor, SensorManager.SENSOR_DELAY_NORMAL);
    }
}

private void stateSelection(){
    fatCheck.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (isChecked) {
                Toast.makeText(getApplicationContext(), "Simulacion Activada", Toast.LENGTH_SHORT).show();
                simFat = true;
            } else {
                Toast.makeText(getApplicationContext(), "Simulacion Desactivada", Toast.LENGTH_SHORT).show();
                simFat = false;
            }
        }
    });
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (grantResults.length > 0 && grantResults[0] != PackageManager.PERMISSION_GRANTED) {
        getPermissionFunction();
    }
}

@Override
protected List<? extends CameraBridgeViewBase> getCameraViewList() {
    return Collections.singletonList(cameraBridgeViewBase);
}
}

```

Anexo F. Código de la clase para la detección de fatiga.

```
public abstract class Detection implements CameraBridgeViewBase.CvCameraViewListener2 {

    private static final int TIME_OF_DETECTION =60;
    private Mat grayMat, rotatedFrame, grayImg, filtImg, outImg;
    public CascadeClassifier faceCascade = null, eyeCascade = null, yawnCascade = null;
    public boolean start = false, stop = false, stateDetection = false, eyesDetected = false, previousEyesDetected = false,
    yawnDetected = false, durationBlink = false, presenceYawn = false, frequencyBlink = false;
    private int blinkCount = -1, longBlinkCount = 0, yawCount =0, fatigaStatesCount = 0;
    private TextView blinkCountTextView, blinkTimeTextView, yawnCountTextView, timeTextView, sensorTextView,
    stateTextView;
    private long numYawFra = 0, timDetAct = 0;
    Time yawDuration = new Time();
    Time timeDet = new Time();
    Time blinkDuration = new Time();
    double numfraTot = 0, numFramEyeClo = 0, sensorValue;

    public Detection(TextView yawnCountTextView, TextView blinkCountTextView, TextView blinkTimeTextView,
        TextView timeTextView, TextView sensorTextView, TextView stateTextView) {
        this.yawnCountTextView = yawnCountTextView;
        this.blinkCountTextView = blinkCountTextView;
        this.blinkTimeTextView = blinkTimeTextView;
        this.timeTextView = timeTextView;
        this.sensorTextView = sensorTextView;
        this.stateTextView = stateTextView;
    }

    @Override
    public void onCameraViewStarted(int width, int height) {
        grayMat = new Mat();
        rotatedFrame = new Mat();
    }

    @Override
    public void onCameraViewStopped() {
        if (grayMat != null) {
            grayMat.release();
            rotatedFrame.release();
        }
    }

    @Override
    public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
        sensorValue = Double.parseDouble(sensorTextView.getText().toString());
        TimeFunction();
        rotatedFrame = RotatedFrameFunction(inputFrame.rgba());
        grayMat = processsingImagenFunction(rotatedFrame);
        MatOfRect faces = new MatOfRect();
        if (faceCascade != null) {
            faceCascade.detectMultiScale(grayMat, faces, 1.4, 8, 2, new Size(270, 270));
        }

        Rect[] facesArray = faces.toArray();
        for (Rect face : facesArray) {
```

```

    Imgproc.rectangle(rotatedFrame, face.tl(), face.br(), new Scalar(255, 0, 0, 255), 3);
    Imgproc.rectangle(grayMat, face.tl(), face.br(), new Scalar(255, 0, 0, 255), 3);
    int faceX1 = face.x;
    int faceY1 = face.y;
    int faceX2 = face.x + face.width;
    int faceY2 = face.y + face.height;

    /** Region de los ojos. */
    int eyesY2 = faceY1 + (faceY2 - faceY1)/2;
    int eyesY1 = faceY1 + (eyesY2 - faceY1)/3;
    Mat eyeRegion = grayMat.submat(new Rect(faceX1, eyesY1, faceX2 - faceX1, eyesY2 - faceY1));
    int mouthY1 = faceY1 + (faceY2 - faceY1)/2;
    int mouthY2 = faceY2 + 30;
    Mat mouthRegion = grayMat.submat(new Rect(faceX1, mouthY1, faceX2 - faceX1, mouthY2 - mouthY1));
    MatOfRect eyeMatRec = new MatOfRect();
    if (eyeCascade != null) {
        eyeCascade.detectMultiScale(eyeRegion, eyeMatRec, 1.1, 12, 2, new Size(50, 50), new Size(80, 80));
    }

    Rect[] eyesArray = eyeMatRec.toArray();
    for (Rect eye : eyesArray) {

        Point eyeTl = new Point(eye.tl().x + faceX1, eye.tl().y + eyesY1);
        Point eyeBr = new Point(eye.br().x + faceX1, eye.br().y + eyesY1);

        double aspectRatio = 0.8; // Puedes ajustar este valor según tus necesidades
        double semiMajorAxis = eye.width / 2;
        double semiMinorAxis = semiMajorAxis * aspectRatio;
        Point eyeCenter = new Point((eyeTl.x + eyeBr.x) / 2, (eyeTl.y + eyeBr.y) / 2);

        Imgproc.ellipse(rotatedFrame, eyeCenter, new Size(semiMajorAxis, semiMinorAxis), 0, 0, 360, new Scalar(0, 255, 0,
255), 3);

    }

    MatOfRect yawMatRec = new MatOfRect();
    if (yawnCascade != null) {
        yawnCascade.detectMultiScale(mouthRegion, yawMatRec, 1.3, 10, 2, new Size(70, 70));
    }
    Rect[] yawnArray = yawMatRec.toArray();
    for (Rect yawRect : yawnArray) {
        Point yawnTl = new Point(yawRect.tl().x + faceX1, yawRect.tl().y + mouthY1);
        Point yawnBr = new Point(yawRect.br().x + faceX1, yawRect.br().y + mouthY1);

        Imgproc.rectangle(rotatedFrame, yawnTl, yawnBr, new Scalar(0, 255, 0, 255), 3);
        Imgproc.rectangle(grayMat, yawnTl, yawnBr, new Scalar(0, 255, 0, 255), 3);
    }

    if(stateDetection == true && sensorValue > 15 && sensorValue <215) {
        numfraTot++;
        blinkDetectFunction(eyeMatRec); // contador de parpadeos
        yawnDetectFunction(yawMatRec); // contador bostezos
    }
}
return rotatedFrame;
}

```

```

public void TimeFunction(){
    if(start == true){
        timeDet.setTimIni(System.currentTimeMillis()/1000);
        start = false;
        blinkCount = 0;
        yawCount = 0;
        showTitle(yawnCountTextView, "Bostezos: ",0);
        showTitle(blinkCountTextView,"Parpadeos: ",0);
        showTitle(blinkTimeTextView,"Tiempo Parpadeo: ",0);
        stateDetection = true;
    }
    if(stop == true){
        stop = false;
        showTitle(yawnCountTextView, "Bostezos: ",0);
        showTitle(blinkCountTextView,"Parpadeos: ",0);
        showTitle(blinkTimeTextView,"Tiempo Parpadeo: ",0);

        showTitle(timeTextView, "Tiempo: ",0);
        showState(stateTextView,"ESTADO: ");

        stateDetection = false;
    }

    if( stateDetection == true){
        timeDet.setTimFin(System.currentTimeMillis()/1000);
        timDetAct = timeDet.getTimAct();
        showTitle(timeTextView, "Tiempo: ", timDetAct);
        if(timDetAct < TIME_OF_DETECTION) {
            timeDet.setTimFin(System.currentTimeMillis() / 1000);
            timDetAct = timeDet.getTimAct();
            showTitle(timeTextView, "Tiempo: ", timDetAct);
        }else if(timDetAct == TIME_OF_DETECTION){
            fatigaDetectionFunction();
            timDetAct =0;
            timeDet.setTimIni(System.currentTimeMillis()/1000);

            showTitle(yawnCountTextView, "Bostezos: ", 0);
            showTitle(blinkCountTextView,"Parpadeos: ",0);
        }
    }else {
        stop = false;
    }
}

private void blinkDetectFunction(MatOfRect eyeMatRec) {
    if (!eyeMatRec.empty()) {
        eyesDetected = true;
        blinkDuration.setTimFin(System.currentTimeMillis());

    } else {
        if (blinkDuration.getTimIni() == 0) {
            blinkDuration.setTimIni(System.currentTimeMillis());
        }
        eyesDetected = false;
        numFramEyeClo++;
    }
}

```

```

    }
    if (eyesDetected && !previousEyesDetected && !yawnDetected) {
        blinkCount++;
        showTitle(blinkCountTextView,"Parpadeos: ",blinkCount);
        if(blinkCount>0 && blinkDuration.getTimAct()<400){
            showTitle(blinkTimeTextView,"Tiempo Parpadeo: ",blinkDuration.getTimAct());
            if(blinkDuration.getTimAct()>200 && blinkDuration.getTimAct()<400){
                longBlinkCount++;
            }
        }
        blinkDuration.setTimIni(0);
        blinkDuration.setTimFin(0);
    }
    previousEyesDetected = eyesDetected;
}

private void yawnDetectFunction(MatOfRect yawMatRec){
    if (!yawMatRec.empty()) {
        yawnDetected = true;
        if(yawDuration.getTimIni()==0) {
            yawDuration.setTimIni(System.currentTimeMillis() / 1000);
        }
        numYawFra++;
        showTitle(yawnCountTextView, "Bostezos: ", yawCount);
    }else{
        yawDuration.setTimFin(System.currentTimeMillis()/1000);
        if(yawnDetected == true && yawDuration.getTimAct() > 5){
            if(numYawFra>30){
                yawCount++;
                showTitle(yawnCountTextView, "Bostezos: ", yawCount);
            }else{
                showTitle(yawnCountTextView, "Bostezos: ", yawCount);
            }
        }
        yawDuration.setTimAct(0);
        yawDuration.setTimIni(0);
        yawDuration.setTimFin(0);
        numYawFra=0;
        yawnDetected = false;
    }
}

private Mat processingImagenFunction(Mat intImage) {

    grayImg = new Mat();
    filtImg = new Mat();
    outImg = new Mat();

    Imgproc.cvtColor(intImage, grayImg, Imgproc.COLOR_RGBA2GRAY);
    Imgproc.GaussianBlur(grayImg, filtImg, new Size(5, 5), 0);
    Imgproc.equalizeHist(filtImg,outImg);
    return outImg;
}

private Mat RotatedFrameFunction(Mat inImg){
    Mat outImg = new Mat();

```

```

    double angle = 90; // angulo de rotacion.
    int width = inImg.cols();
    int height = inImg.rows();
    Point center = new Point(width / 2, height / 2);
    Mat rotationMatrix = Imgproc.getRotationMatrix2D(center, angle, 1);
    Imgproc.warpAffine(inImg, outImg, rotationMatrix, inImg.size());
    Core.flip(outImg, outImg, 1);
    return outImg;
}

public void fatigaDetectionFunction(){
    double PMPM=15;
    double PPM = blinkCount;
    double BPM = yawCount;

    if(PPM>= PMPM){
        frecuenciaBlink = true;
        fatigaStatesCount++;
        Log.d("FRECUENCIA DEL PARPADEO: ", ""+PPM +"por minuto");
    }
    if(0<BPM){
        presenceYawn = true;
        fatigaStatesCount++;
        Log.d("BOSTEZOS: ", ""+BPM +"por minuto");
    }
    if(longBlinkCount>7){
        durationBlink = true;
        fatigaStatesCount++;
        Log.d("DURACION DEL PARPADEO", ""+ longBlinkCount +" veces en 1 minuto");
    }

    if(fatigaStatesCount>=2){
        showState(stateTextView,"ESTADO: FATIGA ");
        alertFatiga();
    }else if(fatigaStatesCount==1){
        showState(stateTextView,"ESTADO: POSIBLE FATIGA ");
    }else{
        showState(stateTextView,"ESTADO: SIN FATIGA ");
    }
    frecuenciaBlink = false;
    durationBlink = false;
    presenceYawn = false;
    numFramEyeClo = 0;
    numfraTot = 0;
    blinkCount = 0;
    yawCount = 0;
    longBlinkCount =0;
    fatigaStatesCount =0;
}

public abstract void showTitle (TextView textView, String text, long count);
public abstract void showState(TextView textView,String text);
public abstract void alertFatiga();
}

```

Anexo G. Código de la clase para la carga de clasificadores.

```
public class CascadeClassifierLoader {

    public static CascadeClassifier loadCascade(Context context, int resId, String outFilNam) {
        CascadeClassifier cascadeClassifier = new CascadeClassifier();

        try {
            InputStream is = context.getResources().openRawResource(resId);
            File cascadeDir = context.getDir("cascade", Context.MODE_PRIVATE);
            File cascadeFile = new File(cascadeDir, outFilNam);
            FileOutputStream os = new FileOutputStream(cascadeFile);

            byte[] buffer = new byte[4096];
            int bytesRead;
            while ((bytesRead = is.read(buffer)) != -1) {
                os.write(buffer, 0, bytesRead);
            }
            is.close();
            os.close();
            cascadeClassifier.load(cascadeFile.getAbsolutePath());
            if (cascadeClassifier.empty()) {
                Log.e("CASCADE", "Failed to load cascade classifier");
            } else {
                Log.d("CASCADE", "Loaded cascade classifier");
            }
        } catch (IOException e) {
            e.printStackTrace();
            Log.e("CASCADE", "Error copying cascade classifier");
        }
        return cascadeClassifier;
    }
}
```

Anexo H. Código de la clase para la simulación de estado.

```
public class StateSimulation {

    private static final int FLASH_BLINK = 400, FLASH_YAWN = 5000;
    private Handler handler;
    private boolean simState = true, simFat;
    private View indParpadeos, indBostezos;
    private int delayBlink = 2500, delayYawn = 30000, flashColor;

    public StateSimulation(View indParpadeos, View indBostezos, Handler handler) {
        this.indParpadeos = indParpadeos;
        this.indBostezos = indBostezos;
        this.handler = handler;
    }

    public void startDelayBlink() {
        Random randomBlink = new Random();
        Random randomColor = new Random();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                if (simState) {
                    if (simFat) {
                        delayBlink = (19 + randomBlink.nextInt(19)) * 100;
                        Log.d("delayBlink: ", "" + delayBlink);
                        flashColor = randomColor.nextInt(2);
                        Log.d("color: ", "" + flashColor);

                        if (flashColor == 1) {
                            indParpadeos.setBackgroundColor(Color.YELLOW);
                        } else {
                            indParpadeos.setBackgroundColor(Color.RED);
                        }
                    } else {
                        indParpadeos.setBackgroundColor(Color.YELLOW);
                        delayBlink = (25 + randomBlink.nextInt(25)) * 100;
                        Log.d("delayBlink: ", "" + delayBlink);
                    }
                }
                handler.postDelayed(new Runnable() {
                    @Override
                    public void run() {
                        if (simState) {
                            indParpadeos.setBackgroundColor(Color.GRAY);
                            startDelayBlink();
                        }
                    }
                }, FLASH_BLINK);
            }
        }, delayBlink);
    }

    public void startDelayYawn() {
        Random randomYawn = new Random();
```

```

handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        if (simState) {
            indBostezos.setBackgroundColor(Color.GREEN);
            delayYawn = (30 + randomYawn.nextInt(10))*1000;
            Log.d("IntervaloBostezo: ", "" + delayYawn);
            handler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    if (simState) {
                        indBostezos.setBackgroundColor(Color.GRAY);
                        startDelayYawn();
                    }
                }
            }, FLASH_YAWN);
        }
    }, delayYawn);
}

public void setSimFat(boolean simFat) { this.simFat = simFat; }
public void setSimState(boolean simState) { this.simState = simState; }
}

```

Anexo I. Código de la clase para manejo de tiempos.

```
public class Time {  
    private long timIni=0, timFin=0, timAct =0;  
  
    public void setTimIni(long timIni) {  
this.timIni = timIni;  
    }  
    public void setTimFin(long timFin) {  
this.timFin = timFin;  
    }  
    public long getTimIni() {  
return timIni;  
    }  
    public void setTimAct(long timAct) {  
this.timAct = timAct;  
    }  
    public long getTimAct() {  
        timAct = timFin - timIni;  
        return timAct;  
    }  
}
```