ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN ALGORITMO PARA LA DETECCIÓN DE SOMNOLENCIA HUMANA, EN TIEMPO REAL BASADO EN VISIÓN ARTIFICIAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN ELECTRÓNICA Y TELECOMUNICACIONES

ADRIANA SOFÍA REVELO ÁLAVA

adriana.revelo@epn.edu.ec

DIRECTOR: Ph.D. ROBIN GERARDO ÁLVAREZ RUEDA

robin.alvarez@epn.edu.ec

Quito, octubre 2019

AVAL

Certifico que el presente trabajo fue desarrollado por Adriana Sofía Revelo Álava, bajo mi supervisión.

Ph.D. Robin Álvarez Rueda
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Adriana Sofía Revelo Álava, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamento y Normas vigentes.

Adriana Sofía Revelo Álava

DEDICATORIA

Dedicado a quienes se adelantaron y aun así siguen siendo el corazón de mi familia.

AGRADECIMIENTO

A Dios por su amor y gracia conmigo.

A mis padres Carmen (Nina) y Héctor por su apoyo y motivación para alcanzar cada una de mis metas.

A mi hermano Héctor Paúl y mi sobrina Sophie por darle vida a mis días.

A Erick por su bondad y paciencia. A mis amigos por el tiempo, alimento y momentos compartidos.

Un especial agradecimiento al Dr. Robin Álvarez por el tiempo invertido en guiar este Trabajo de Titulación.

A los profesores de la Escuela Politécnica Nacional que compartieron su conocimiento conmigo.

Sofía Revelo

ÍNDICE DE CONTENIDO

AVAL I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN VII
ABSTRACT VIII
1. INTRODUCCIÓN 1
1.1. OBJETIVOS 2
1.2. ALCANCE 2
1.3. MARCO TEÓRICO 4
1.3.1. ESTADO DEL ARTE DE ALGORITMOS Y TECNOLOGÍAS DE SOMNOLENCIA HUMANA4
2. METODOLOGÍA 9
2.1. DISEÑO DEL ALGORITMO DE DETECCIÓN DE SOMNOLENCIA 9
2.2. CONDICIONES DE FUNCIONAMIENTO 11
2.3. ETAPA 1: ADQUISICIÓN DE IMÁGENES 12
2.4. ETAPA 2: EXTRACCIÓN DE LA SECCIÓN DEL OJO 16
2.4.1. FASE 1: DETECCIÓN DE ROSTRO 16
2.4.2. FASE 2: RECORTE PROPORCIONAL DE OJO 22
2.5. ETAPA 3: PROCESAMIENTO DE IMAGEN DEL OJO 25
2.5.1. MÉTODO 1 26
2.5.2. MÉTODO 2 28
2.6. ETAPA 4: CLASIFICACIÓN DE OJO ABIERTO/CERRADO 31
2.6.1. MÉTODO 1 DE CLASIFICACIÓN DE OJO ABIERTO/CERRADO 31
2.6.2. MÉTODO 2 DE CLASIFICACIÓN DE OJO ABIERTO/CERRADO 42
2.7. ETAPA 5: DETECCIÓN DE SOMNOLENCIA 47
3. RESULTADOS Y DISCUSIÓN 51
3.1.1. ETAPA 1: ADQUISICIÓN DE IMÁGENES 52
3.1.2. ETAPA 2: OBTENCIÓN DE LA SECCIÓN DEL OJO 53
3.1.3. ETAPAS 3 Y 4: PROCESAMIENTO DE IMÁGENES Y CLASIFICACIÓN OJO ABIERTO/CERRADO 54
3.1.4. ETAPA 5: DETECCIÓN DE SOMNOLENCIA 56
4. CONCLUSIONES Y RECOMENDACIONES 65

4.1. CONCLUSIONES 65
4.2. RECOMENDACIONES 66
REFERENCIAS BIBLIOGRÁFICAS 67
ANEXOS 72

RESUMEN

Se presenta un algoritmo de detección de somnolencia humana en tiempo real usando visión artificial. La somnolencia es un estado cuyas consecuencias pueden ser muy peligrosas para conductores de vehículos, controladores aéreos, controladores de plantas nucleares, etc. En Ecuador en el año 2018 se reportaron 353 accidentes de tránsito por conducir en estado de somnolencia. Por este motivo es importante que los conductores se mantengan despiertos.

El algoritmo que se presenta obtiene imágenes frontales de una persona usando una cámara infrarroja, detecta automáticamente el rostro usando el algoritmo de Viola-Jones y extrae la porción del ojo. Posteriormente clasifica los ojos como abiertos o cerrados por medio de dos métodos: usando una red neuronal de tipo perceptrón multicapa (*Multi Layer Perceptron* o MLP) y extrayendo los máximos y mínimos horizontales y verticales de los bordes del ojo. Finalmente realiza la detección de somnolencia según el tiempo en que los ojos estuvieron cerrados dentro de un intervalo de tiempo.

Para la clasificación de ojos abiertos/cerrados usando MLP se obtuvo un 97% de precisión cuando las imágenes de entrenamiento y de prueba son de la misma persona y del 71% cuando las imágenes son de distintas personas. Para el segundo método se obtuvo el 84% de precisión.

Finalmente, este trabajo de titulación permitió realizar un paper que será publicado en IEEE Xplore previa a su presentación en IEEE ETCM 2019. Este paper se lo presenta en el ANEXO A, su copyright es: 978-1-7281-3764-3/19/$31.00 ©2019 IEEE.

Palabras clave: detección automática de somnolencia, detección de parpadeo, visión artificial.

ABSTRACT

This project presents a human drowsiness detection algorithm in real time using computer vision. Drowsiness is a state whose consequences can be very dangerous for vehicle drivers, air traffic controllers, nuclear plant controllers, etc. In 2018 in Ecuador, 353 traffic accidents were reported for driving while drowsy. For this is important that drivers keep alert while driving.

The algorithm that is presented in this project obtains frontal images of the driver using an infrared camera, then performs automatic face detection using the Viola-Jones algorithm and extracts the eye portion. Then, the classification between open and closed eye is done with two methods: a) using a multilayer perceptron (MLP) neural network and b) extracting the maximums and minimums of horizontal and vertical edges of the eye. Finally, it makes the detection of drowsiness considering the time the eyes were closed within a time interval.

For the open and close eye classification using the first method the algorithm obtains 97% of accuracy when the training and test images are from the same person and 71% when the training and test images are from different people. For the second method obtains 84% of accuracy.

Finally, this project allowed to write a paper that will be published in IEEE Xplore prior to its presentation in IEEE ETCM 2019. This paper is presented in ANEXO A, its copyright is: 978-1-7281-3764-3/19/$31.00 ©2019 IEEE.

Keywords: automatic drowsiness detection, blink detection, computer vision.

1. INTRODUCCIÓN

Los accidentes de tránsito debido la somnolencia y fatiga del conductor son un problema de interés mundial. En Ecuador, en el año 2018 se registraron 25.530 accidentes de tránsito, la causa principal fue por conducir desatento a las condiciones de tránsito con un 24,13% del total de accidentes, mientras que el 1,38% fue por conducir en estado de somnolencia o malas condiciones físicas [1]. Se ha determinado que una de las alteraciones más importantes producidas por la somnolencia es una menor concentración [2], disminuyendo el tiempo de reacción ante eventos que pueden conducir a un accidente inminente. Montoro, Soler y Tortosa exponen que el sueño altera negativamente las actividades psicofísicas que son requeridas para conducir [3], produciendo situaciones de peligro. Por este motivo para mejorar la seguridad vial, investigadores y empresas automotrices han implementado algoritmos y dispositivos que pretenden detectar dicha somnolencia humana. A estos algoritmos y dispositivos se los puede clasificar en tres grandes grupos: los que están basados en variables fisiológicas, los basados en el comportamiento ocular determinado mediante visión artificial y los basados en parámetros del vehículo. Los segundos son los más apreciados debido su exactitud y a que son mínimamente invasivos ya que no requieren la colocación de ningún dispositivo como es el caso de los del primer grupo.

El algoritmo pretende cumplir con tres objetivos: ser capaz de trabajar con distintos tipos de rostro, ser capaz de trabajar bajo diferentes condiciones de luminosidad y ser mínimamente invasivo, empleando dispositivos no sofisticados y de bajo costo. Adicionalmente, en el caso de un conductor de vehículo, dicho estado de somnolencia debe ser detectado incluso en condiciones de luminosidad pobre, por esta razón, en el presente trabajo se usa una cámara infrarroja que obtiene continuamente imágenes del rostro de una persona y si el algoritmo detecta somnolencia se enciende una alarma de alerta. Ya que en estado de somnolencia los parpadeos suelen ser más lentos y de mayor duración, el algoritmo determina este comportamiento.

El presente proyecto basado en visión artificial consta de dos métodos para la clasificación de ojos abiertos y cerrados:

- Método 1: mediante redes neuronales.
- Método 2: mediante la morfología del ojo.

Finalmente, el algoritmo puede ser adaptado a otros escenarios distintos al ambiente de conducción vehicular en donde se requiera detectar somnolencia humana; estos ambientes pueden ser para controladores aéreos, controladores de plantas nucleares, entre otros.

1.1. OBJETIVOS

El objetivo general de este Proyecto Técnico es: desarrollar un algoritmo para detectar la somnolencia humana, en tiempo real basado en inteligencia artificial.

Los objetivos específicos del Proyecto Técnico son:

i) Determinar el estado del arte en cuanto a algoritmos y tecnologías de detección de somnolencia humana.

ii) Adquirir varias imágenes, con una cámara infrarroja, de distintos sujetos de experimentación bajo diferentes niveles de iluminación que servirán de base para aplicar los distintos algoritmos de clasificación de ojos abiertos y cerrados.

iii) Estudiar algunas técnicas de detección de objetos, por ejemplo: Ada-boost, clasificadores en cascada, algoritmo Haar, entre otras.

iv) Implementar un algoritmo que permita binarizar imágenes donde se reconozca el ojo.

v) Realizar la detección automática de somnolencia, en tiempo real, empleando los siguientes métodos:

Método 1:
Por medio de una red neuronal de perceptrones multicapa (MLP) cuyas entradas serán entrenadas por las imágenes de ojos abiertos y cerrados que se obtuvieron en uno de los objetivos anteriores.

Método 2:
Mediante la morfología del ojo, extrayendo puntos de referencia extremos tanto horizontales como verticales.

vi) Analizar los resultados obtenidos con los dos métodos de clasificación anteriores y determinar el mejor de ellos para ser empleado en tiempo real.

1.2. ALCANCE

Para la adquisición, procesamiento de imágenes y la implementación de la interfaz gráfica de usuario se utilizará el software Matlab. De este software se hará uso de las siguientes

herramientas: Image Processing Toolbox, Computer Vision System Toolbox y Deep Learning Toolbox.

El algoritmo realizará la detección de rostro y la clasificación de ojo abierto/cerrado con lo que, basados en un umbral correspondiente a un comportamiento normal, se determinará si existe o no somnolencia en el usuario del sistema; en el caso de sobrepasar dicho umbral, se emitirá una alerta de modo que el conductor pueda tomar la decisión de seguir conduciendo, pero con mucho cuidado o tomar un descanso en un sitio apropiado de la carretera.

Para que el sistema funcione en condiciones de poca luz se usará una cámara infrarroja.

El algoritmo detectará el estado de somnolencia de una sola persona, la cual debe estar justo frente a la cámara, el algoritmo no identificará somnolencia con personas que se encuentren de perfil a la cámara. De existir varias personas frente a la cámara el detector seleccionará el rostro de la persona que esté más cerca de la cámara.

La detección se hará solamente en una persona, si existen varias personas frente a la cámara se hará la detección de quien esté en primer plano.

El algoritmo no será capaz de detectar somnolencia en una persona que esté usando gafas oscuras. Sin embargo, se realizaron pruebas en una persona usando diferentes tipos de gafas transparentes donde se encontró que los resultados son variables dependiendo del grosor del marco de los lentes y la existencia de reflejos en las lunas de los lentes, por este motivo este algoritmo no está adaptado para ser usado en personas que usan algún tipo de gafas.

El presente algoritmo funciona en un computador con la ayuda de una interfaz gráfica para mostrar los resultados, pero no se realiza el diseño ni implementación en algún dispositivo de hardware.

Los algoritmos serán probados en tiempo diferido por medio de imágenes previamente adquiridas y el mejor de ellos será implementado para la detección de somnolencia en tiempo real.

1.3. MARCO TEÓRICO

1.3.1. ESTADO DEL ARTE DE ALGORITMOS Y TECNOLOGÍAS DE SOMNOLENCIA HUMANA

Por varios años investigadores y empresas automotrices han trabajado en sistemas de detección de somnolencia humana. A estos sistemas se los puede clasificar en tres grandes grupos: sistemas invasivos, sistemas no invasivos y sistemas basados en parámetros del vehículo.

1.4.1.1. Detección de somnolencia invasiva

Estos sistemas se llaman invasivos ya que se usa dispositivos que estarán colocados directamente en el cuerpo de una persona para poder recibir señales y determinar si existe o no somnolencia en el individuo.

Existen varios indicadores biológicos para determinar si un individuo presenta síntomas somnolencia o no, entre los más conocidos está el monitoreo del cerebro por medio de electroencefalograma (EEG), el cual requiere electrodos ubicados en la cabeza para la detección de somnolencia. Otro sistema está basado en el monitoreo de los ojos con electrooculograma (EOG), el cual requiere de al menos dos electrodos ubicados en los bordes superior e inferior del ojo de la persona [4]. En la Figura 1.1 se muestra la ubicación de los electrodos tanto para EEG como para EOG.

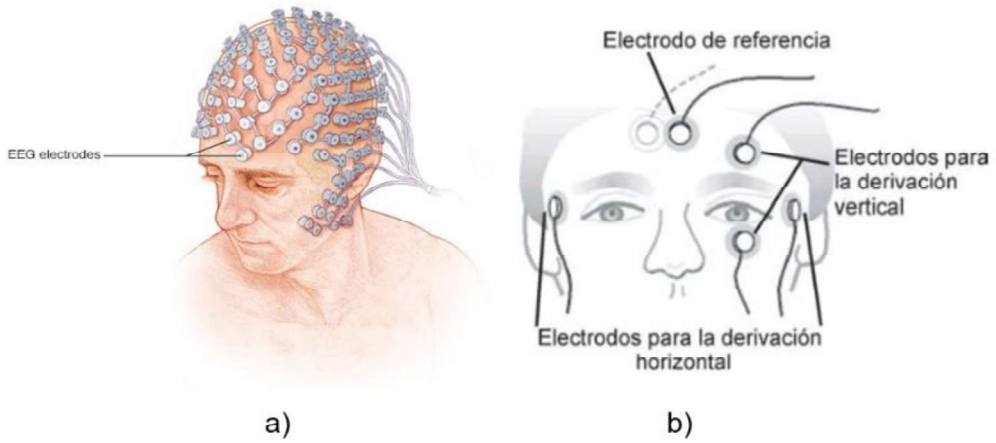


Figura 1.1 Ubicación de electrodos para métodos de somnolencia invasiva. a) Electrodos EEG [5], b) Electrodos EOG [6]

Otros bioseñales que podrían dar algún indicio sobre el estado de somnolencia son el electrocardiograma (ECG), electromiograma de superficie (SEMG) y la respuesta galvánica de la piel (GSR) [4].

A pesar de que estos sistemas de detección de somnolencia, especialmente el EEG y EOG, podrían dar resultados confiables, debido a su carácter invasivo no han logrado constituirse como sistemas de uso práctico y solo están limitados para ser usadas a nivel de laboratorio.

1.4.1.2. Detección de somnolencia no invasiva

A diferencia de los anteriores, estos no necesitan de sensores colocados de forma invasiva sobre el usuario, más bien aprovechan la existencia de indicadores físicos que pueden estar relacionados con su grado de somnolencia, estos indicadores son: posición de la cabeza, bostezos y actividad de los ojos. Estos comportamientos pueden ser monitoreados de forma no invasiva a través de visión artificial empleando una cámara de video.

Varias investigaciones incorporan algoritmos híbridos que involucran el seguimiento con cámaras de más de un aspecto físico, como la investigación de Qiang Ji y Xiaojie Yang [7] donde se hace el seguimiento de la posición de rostro, ojos y mirada.

La Tabla 1.1 realizada por A. Sahayadhas, K. Sundaraj y M. Murugappan en su investigación "Detecting Drive Droesiness Based on Sensors: A Review" presenta un resumen de trabajos anteriores para la detección de somnolencia [8].

Tabla 1.1 Resumen de trabajos sobre somnolencia. Adaptado de [8]

Sensor Usado	Medida de somnolencia	Técnicas de detección	Clasificación	Tasa de detección positiva
CCD micro cámara con iluminación infrarroja	Pupila	Ada-boost	Relación de alto y ancho de ojo	92%
Cámara e iluminación infrarroja	PERCLOS, duración de ojo cerrado, frecuencia de parpadeo y otros 3	Dos filtros Kalman para detección de pupila	Clasificador Fuzzy	Cerca del 100%
Cámara CCD	Bostezo	Modelo centro de gravedad y proyección gris	LDA (Linear Discriminant Analysis)	91,97%
Cámara de Video Digital	Acciones faciales	Filtro Gabor	SVM (Support Vector Machine)	96%
Cámara FireWire y cámara web	Duración y frecuencia de ojo cerrado	Transformada Hough	Clasificador neuronal	95%
Cámara	Características dinámicas multi escala	Filtro Gabor	Ada boost	98,33%
Cámara IR	Estado del ojo	Filtro Gabor	SVM (Support Vector Machine)	93%
Cámara simple	Parpadeo	Clasificadores en cascada para detección de rostro	Algoritmo de marca regional	98%
Cámara con iluminación IR	PERCLOS	Algoritmo Haar para detectar rostro	SVM (Support Vector Machine)	99%

Entre todos los indicadores listados, el más efectivo es el índice de porcentaje de ojo cerrado sobre la pupila en el tiempo (denominado PERCLOS del inglés *PERcentage of Eyelid CLOSure*), el cual fue propuesto en abril de 1999 [9].

1.4.1.2.1 Índice PERCLOS

Este índice está definido como el porcentaje de tiempo en el cual los párpados se encuentran cerrados de un 80% a 100% sobre un periodo de tiempo. Para calcular el índice PERCLOS hay que tomar en cuenta que se usa el cierre del párpado más no el parpadeo [10]. El cierre de párpado es mucho más lento que un parpadeo, ya que este considera la posición de los párpados antes que se cierren o abran por completo. Por ejemplo, en un intervalo de un minuto si los párpados cubren el ojo entre el 80% al 100% por 12 segundos, el índice PERCLOS sería 12/60 o 20%.

La Figura 1.2 muestra el porcentaje de ojo cerrado durante un cierre de párpados.

Figura 1.2 Porcentaje de ojo cerrado durante el cierre de párpados. Basado en [11]

PERCLOS presenta varios niveles de somnolencia dependiendo de su valor [10]:

- menor a 7,5% despierto
- del 7,5% al 15% cuestionable
- sobre 15% somnoliento

El índice PERCLOS es el más usado en la detección de somnolencia no invasiva, ya que posee los mejores resultados hasta el momento [12]; sin embargo, necesita de una tasa de adquisición alta para poder adquirir cada uno de los estados del ojo durante el cierre de párpados. Generalmente los algoritmos que usan el índice PERCLOS realizan la detección de pupila (Figura 1.3) para determinar el porcentaje de ojo cerrado [8], de esta manera este índice no dependerá de la forma de los ojos. La Figura 1.4 muestra que la pupila no se encuentra cubierta en unos ojos con rasgos asiáticos.

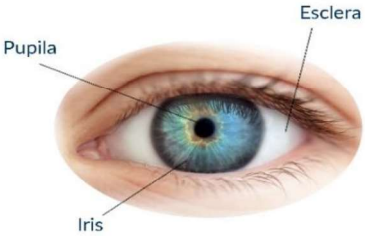


Figura 1.3 Partes externas del ojo [13]

Figura 1.4 Ojos de niña asiática [14]. La pupila no se cubre por los párpados.

1.4.1.2.2 Parámetros de somnolencia según la actividad de los ojos

La frecuencia de parpadeo y su duración pueden determinar la somnolencia en una persona. Individuos que presentan fatiga suelen tener un parpadeo más lento y por lo tanto su ojo estará cerrado por más tiempo.

Por lo general el parpadeo es involuntario y su frecuencia dependerá de varios factores como viento, deslumbramiento, cansancio, emociones, entre otros [15]. La tasa de parpadeo normal es de 15 a 20 veces por minuto, con una duración de parpadeo de 0,25 a 0,3 segundos [16], aunque estos valores pueden cambiar entre individuos. Se considera un estado de alerta cuando el ojo se encuentra cerrado entre el 0% al 30% del tiempo y un estado de somnolencia cuando el ojo se encuentra cerrado entre el 30% al 40% del tiempo [17].

1.4.1.3. Detección de somnolencia en base a parámetros vehiculares

Estos sistemas de detección usan indicadores del comportamiento del vehículo tales como la velocidad, aceleración, comportamiento del movimiento del volante y la posición del vehículo respecto a las señales viales. A pesar de obtener buenos indicios sobre el estado de alerta del conductor, son menos precisos y requieren cumplir ciertas condiciones como por ejemplo, que existan señales viales pintadas en la carretera. Adicionalmente, el costo de los vehículos se incrementa por añadir alguno de estos sistemas. A continuación, se mencionan algunas de las empresas que han trabajados en estos sistemas:

1.4.1.3.1. Volvo

Volvo en su modelo XC60 ha incorporado un sistema, llamado *Driver Control Alert*, el cual usa una cámara que visualiza el exterior, varios sensores y procesadores para determinar la distancia del auto con las líneas de la vía. En base a esto, se determina si la persona está cansada al evaluar los movimientos de auto [18].

1.4.1.3.2. Mercedes Benz

Mercedes Benz ha desarrollado un sistema llamado *Attention Assist*, que está disponible en varios de sus modelos, este sistema crea una base de datos de las características de manejo del conductor cuando se encuentra alerta, luego los compara con los datos que va adquiriendo; estas características pueden ser velocidad y movimientos del auto. Además, considera factores como el clima y tiempo continuo de conducción [19].

1.4.1.3.3. Ford

Su sistema *Drive Alert* usa una cámara para determinar la posición del auto respecto a las líneas viales, si el auto se aproxima a una de las líneas aplicará un torque y vibraciones para que el conductor se dé cuenta, el sistema no se activa si se usa direccionales [20].

1.4.1.3.4. Panasonic

Su sistema usa una cámara y varios sensores, los cuales analizan las expresiones del rostro del conductor, sensación térmica y temperatura del auto para determinar fatiga en la persona [21].

1.4.1.3.5. Bosch

Determina la fatiga del conductor por medio del ángulo de dirección, rapidez de giro y la posición del auto respeto a las líneas viales [22].

2. METODOLOGÍA

En este capítulo se presenta el diseño del algoritmo de detección de somnolencia. Este algoritmo está basado en la actividad de los ojos: las personas que se encuentran en un estado de somnolencia parpadean con menos frecuencia y el tiempo que los ojos permanecen cerrados durante el parpadeo es mayor [23]. La frecuencia normal de parpadeo es de 15 a 20 parpadeos por minuto, con una duración de parpadeo de 0.25 a 0.3 segundos. Al considerar el parpadeo más corto (0.25 segundos), una imagen debe ser adquirida máximo cada 0.25 segundos [16].

El algoritmo realizado en este proyecto es un sistema no invasivo que recopila y procesa imágenes usando visión artificial. La visión artificial es una disciplina que estudia e intenta construir modelos de las funcionalidades humanas con métodos computacionales. Esto quiere decir que el procesador luego de recibir una imagen puede reconocer el contenido de ésta automáticamente.

2.1. DISEÑO DEL ALGORITMO DE DETECCIÓN DE SOMNOLENCIA

El diseño del algoritmo se realizará en cinco etapas, como muestra la Figura 2.1:

1. Adquisición de imágenes
2. Extracción de la sección del ojo:
    a. Detección de rostro
    b. Recorte proporcional de ojo
3. Procesamiento de imagen (Método 1 y Método 2)
4. Clasificación de ojo abierto/cerrado (Método 1 y Método 2)
5. Detección de somnolencia

Cada una de estas etapas está formada por varias fases, en esta sección se detalla cada una de estas fases para su implementación en Matlab.

Para ambos métodos todas las etapas son iguales excepto las etapas de procesamiento y clasificación.

Para diseñar el algoritmo se dispone de los siguientes recursos:

- Matlab 2018b con Image Acquisition Toolbox, Image Processing Toolbox, Computer Vision System Toolbox, Deep Learning Toolbox y Matlab support Package for USB Webcams.
- Cámara infrarroja HD de parqueo para auto que se alimenta con 12 VDC.

- Capturador de video USB EasyCap.

La Figura 2.2 muestra el esquema de conexión de los recursos.

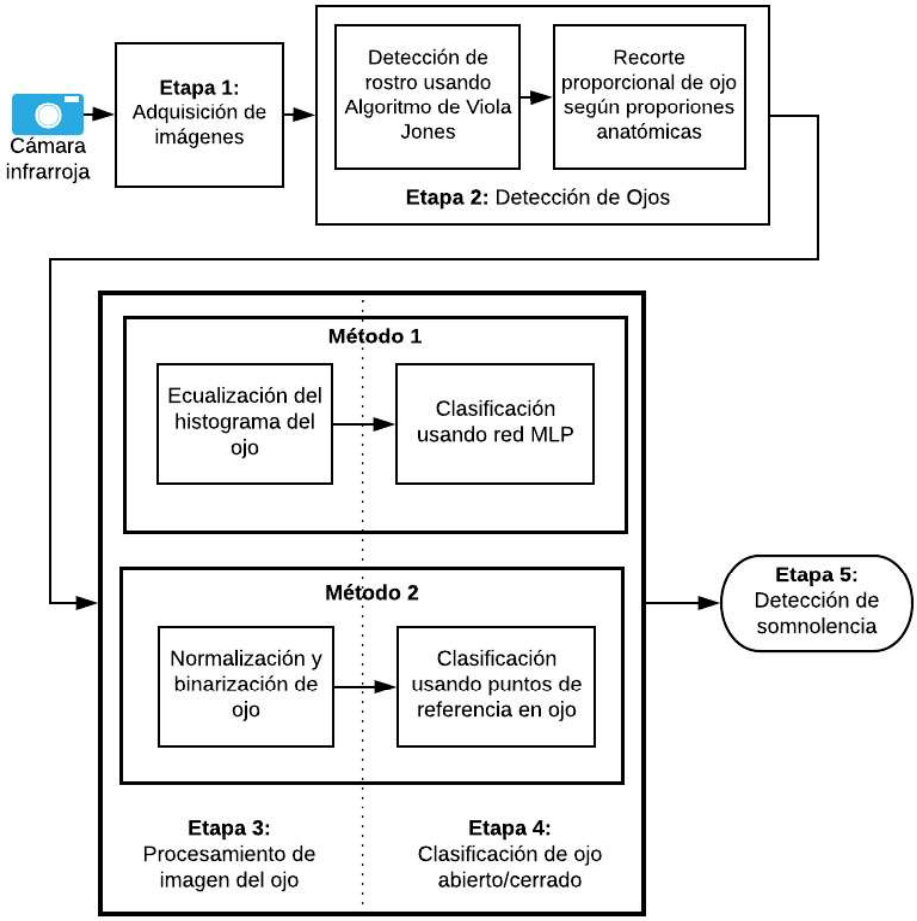


Figura 2.1 Esquema para la detección de somnolencia
El algoritmo propuesto está constituido por 5 etapas: Adquisición de imágenes, detección de ojos, procesamiento de la imagen del ojo, clasificación de ojo abierto/cerrado y detección de ojos.

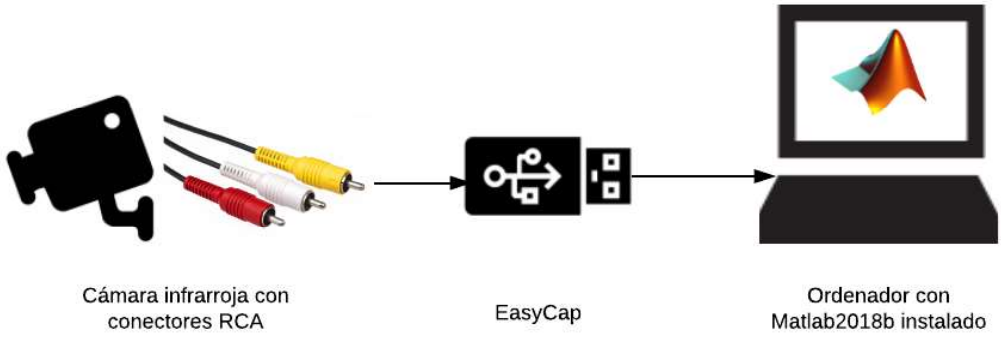


Figura 2.2 Esquema de conexión de los recursos
La cámara se conecta al capturador de video por medio de los conectores RCA y el capturador de video se conecta al ordenador por el puerto USB.

Se ha decidido usar Matlab debido a que posee varias herramientas para el procesamiento digital de imágenes y también porque posee una amplia documentación que explica estas

herramientas [24]. Sin embargo, hay que tomar en cuenta que Matlab requiere de licenciamiento, por lo que esto podría ser una desventaja si no se cuenta con una licencia.

2.2. CONDICIONES DE FUNCIONAMIENTO

Para realizar la detección de rostro se deben cumplir los siguientes requisitos:

- Estar de frente a la cámara.
- El usuario debe procurar mantener la cabeza recta. En las pruebas realizadas, para un correcto funcionamiento del sistema, dicha inclinación debió ser menor a 22° como se muestra en la Figura 2.3.
- Para realizar las pruebas, la cámara estuvo ubicada dos posiciones (Figura 2.4):
    - A 45° bajo la línea de vista de la persona y a 38cm de distancia horizontal de la persona.
    - A 10° bajo la línea de vista de la persona y a 45cm de distancia horizontal de la persona.
- La cámara no puede estar sobre la línea de vista de la persona ya que no existirá detección.
- El rostro completo debe estar dentro de cuadro de visión de la cámara.
- El usuario no debe estar usando gafas oscuras ni transparentes.
- No es necesario que el usuario dirija su mirada hacia la cámara.

La distancia de la cámara y su ubicación puede variar en valores cercanos a los mostrados en la Figura 30, las métricas son valores recomendados debido al ángulo de visión que posee la cámara por lo que si se usa otra cámara estas métricas pueden no dar buenos resultados al momento de la detección. Para saber la distancia correcta de una manera simple, la cabeza debe estar entre las líneas verdes del indicador de la cámara como se muestra en la Figura 2.5.

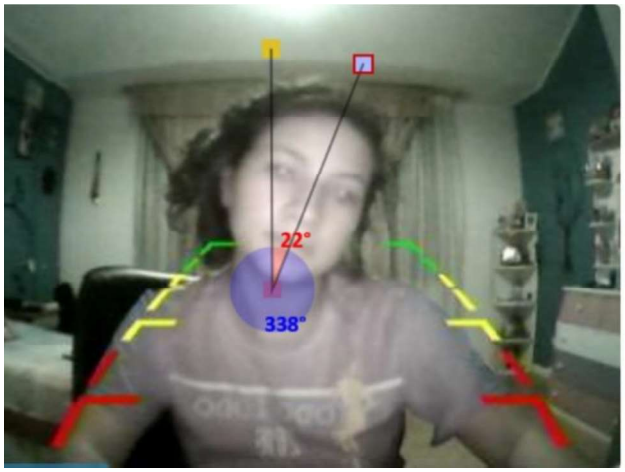


Figura 2.3 Límite de inclinación de cabeza para detección de rostro

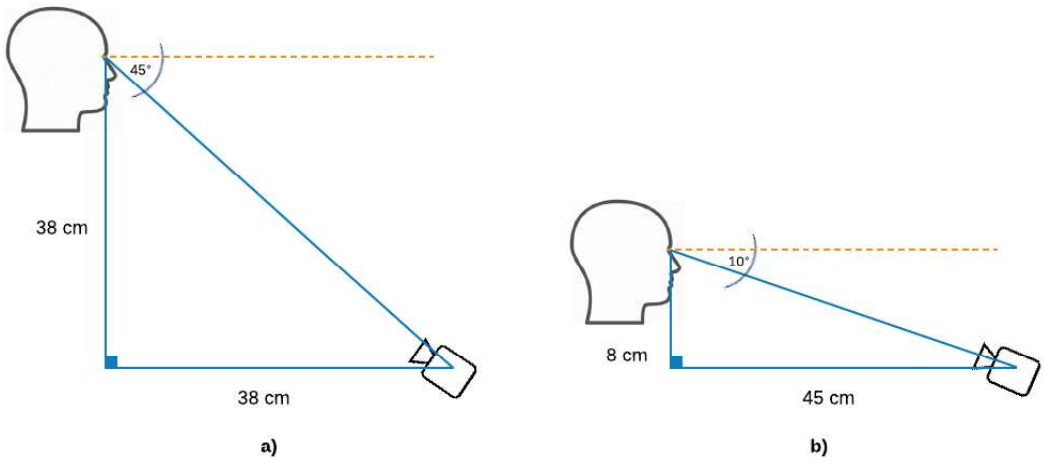


Figura 2.4 Posibles posiciones de la cámara a) cámara a 45° bajo la línea de vista, b) cámara a 10° bajo la línea de vista

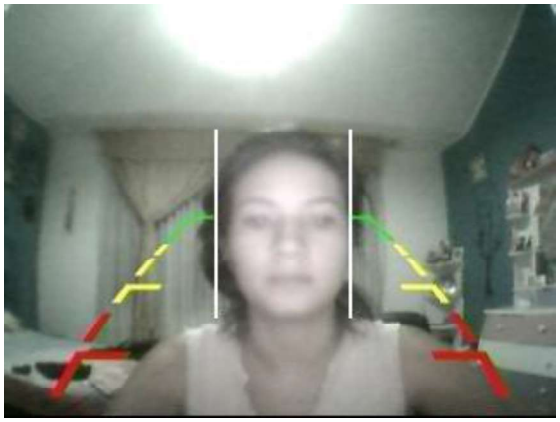


Figura 2.5 Ubicación del rostro entre la terminación de las líneas verdes de referencia de la cámara

2.3. ETAPA 1: ADQUISICIÓN DE IMÁGENES

La adquisición de imágenes consiste en capturar y digitalizar las imágenes por medio de una cámara. Para la adquisición de la imagen hay que tomar en cuenta la iluminación del lugar donde se realizará las capturas y también la posición de la fuente de luz respecto al objeto del que se obtendrá imágenes.

Los requisitos de iluminación también dependerán de la tecnología de la cámara a ser usada y sus características propias de adquisición de imágenes.

En el algoritmo presentado se realiza la adquisición de imágenes del rostro de una persona usando una cámara infrarroja para que el algoritmo pueda ser usado en escenarios con poca iluminación. La resolución usada es de 320x240, esta resolución tiene suficiente información para la detección de parpadeos que a su vez servirá para estimar el estado de somnolencia de la persona. La imagen adquirida usa el modelo RGB donde los colores son

representados en términos de sus componentes rojas (*Red*), verdes (*Green*) y azules (*Blue*), por este motivo se obtiene una matriz de tres dimensiones (240x320x3).

Las funciones en Matlab usadas para adquirir y presentar imágenes en este algoritmo son las siguientes:

- **webcam** crea un objeto de cámara web y se conecta con ella.
  Sintaxis: cam=webcam(*'cameraname'*).
- **snapshot** captura una imagen desde la cámara.
  Sintaxis: frame=snapshot(*cam*)
- **imshow** muestra una imagen y optimiza los ejes y propiedades de los objetos de imagen.
  Sintaxis: imshow(frame)

El diagrama de flujo de la Figura 2.6 muestra el proceso para realizar en Matlab la adquisición de imágenes. Primero se realiza la conexión con la cámara usando la función *webcam*, posteriormente se realiza la captura de la imagen con la función *snapshot*, a continuación, se presenta la imagen usando la función *imshow.* La adquisición y presentación se lo realiza en un bucle hasta que ya no se desee capturar más imágenes.

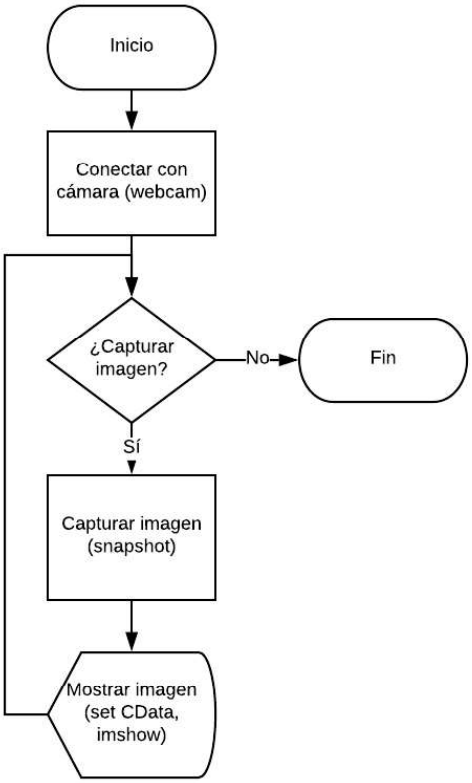


Figura 2.6 Diagrama de flujo de adquisición de imágenes
Se realiza la adquisición de imágenes hasta que ya no se desee capturar más imágenes.

2.3.1. CREACIÓN DE OBJETO Y ASIGNACIÓN DE RESOLUCIÓN

Se crea el objeto de video usando la función *webcam*, no se ingresa el nombre de la cámara como argumento de la función, ya que solamente existe conectada la cámara a ser usada, así que es la tomada por defecto. Para conocer los nombres de las cámaras disponibles se usa la función *webcamlist*.

Para el dispositivo usado existen las siguientes resoluciones disponibles:

- 720x576
- 720x480
- 640x480
- 320x240

Ya que el objetivo del algoritmo es trabajar en tiempo real, se ha seleccionado la menor resolución (320x240). Esta resolución representa una matriz de 76.800 elementos que serán procesados en las siguientes etapas. La resolución de 320x240 posee información suficiente para distinguir los ojos del rostro, primero se realizaron pruebas de adquisición con todas las resoluciones disponibles para elegir la resolución adecuada; estas pruebas y resultados se encuentran en el ANEXO B.

```
%% Creación de objeto y asignación de resolución
cam=webcam();  % Creación de objeto cam
cam.Resolution='320x240'; % Se establece la resolución de 320x240
```

2.3.2. INICIALIZACIÓN DE CÁMARA Y CAPTURADOR

Al momento de conectar la cámara con el capturador de video EasyCap existe un periodo pequeño de tiempo en el que las imágenes aparecen en negro, para evitar capturar estas imágenes en negro, se capturan 25 imágenes que no son procesadas, son tan solo para darle arranque al hardware. Este tiempo que requiere el dispositivo se debe a la conversión de video de analógico a digital que empieza a realizar.

```
%% Inicialización de cámara y capturador
for j=1:25
  snapshot(cam); % Se adquieren 25 imágenes que no se procesarán
end
```

2.3.3. CONTADOR DE TIEMPO

Se usan las funciones *tic* y *toc* para contar el tiempo que se demora en adquirir las imágenes y obtener los cuadros por segundo (*frames per second* o FPS) reales ya que la

tasa de adquisición teórica del dispositivo es de 30 FPS. En la sección 3.1.1 se muestra los resultados de las tasas de adquisición de imágenes para varias funciones de Matlab.

2.3.4. BUCLE DE ADQUISICIÓN

Se realiza la adquisición de imágenes hasta que el usuario cierre la ventana donde se presentan las imágenes adquiridas.

```
while size(findobj(h))>0 % Se adquiere imágenes hasta cerrar Objeto h
  I = snapshot(cam); % Se captura imagen
  set(handles.img(1),'CData',I); % Se presenta imagen
end
```

El código completo para la adquisición de imágenes en Matlab queda de la siguiente manera:

```
%% Creación de objeto y asignación de resolución
cam=webcam(); % Se crea objeto cam
cam.Resolution='320x240'; % Se establece la resolución de 320x240
%% Inicialización de cámara y capturador
for j=1:25
  snapshot(cam); % Se adquieren 25 imágenes que no se procesarán
end
%% Creación de objeto para salida
h = figure; % Se crea un objeto para mostrar imágenes
handles.img(1)=imshow(zeros(240,320)); % Se crea imagen de visualización
%% Obtención de imagen
tic % Inicia contador de tiempo
while size(findobj(h))>0 % Se adquiere imágenes hasta cerrar Objeto h
  I = snapshot(cam); % Se captura imagen
  set(handles.img(1),'CData',I); % Se presenta imagen
end
toc % Finaliza contador de tiempo
```

La Figura 2.7 muestra una imagen RGB obtenida usando el código de Matlab presentado anteriormente.

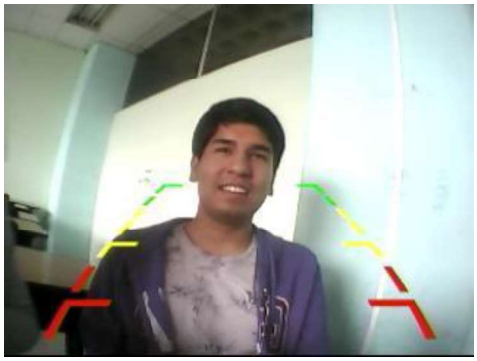


Figura 2.7 Imagen RGB con resolución de 320x240 adquirida con Matlab usando la cámara infrarroja.

2.4. ETAPA 2: EXTRACCIÓN DE LA SECCIÓN DEL OJO

La extracción de la sección de ojo se lo realiza en dos fases. Primero se hace la detección del rostro y posteriormente se hace un recorte proporcional de la sección del ojo (Figura 2.8).

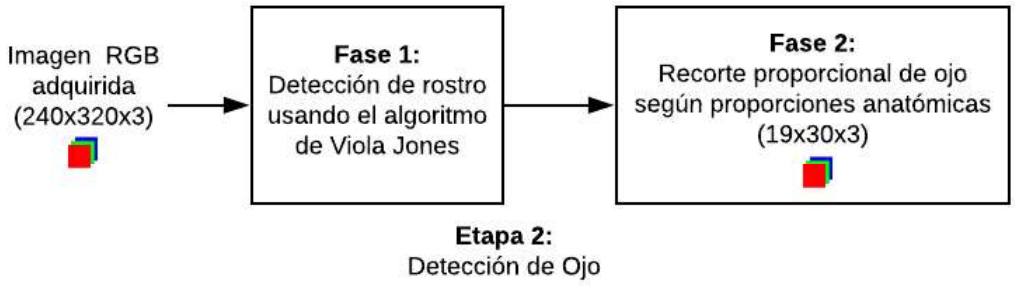


Figura 2.8 Esquema de Etapa 2: Detección de Ojos
En la Fase 1 se detecta el rostro y en la Fase 2 se recorta la sección del ojo.

2.4.1. FASE 1: DETECCIÓN DE ROSTRO

Para detectar el rostro de manera automática se usa la función de Matlab *vision.CascadeObjectDetector*, el cual usa el algoritmo de Viola Jones. Este detector de rostro usado en el algoritmo propuesto acepta una inclinación de cabeza máxima de 21°, si se supera este ángulo no se hará la detección, esto es esperado ya que el algoritmo de Viola Jones es sensible a la rotación [25]. Se usa el detector de Viola Jones ya que para la detección de rostros que miran hacia al frente posee mejores resultados comparado con otros métodos [26].

La sintaxis de la función es la siguiente: detector = vision.CascadeObjectDetector(*model*) Esta función permite realizar la detección de rostro, ojos, nariz y boca de personas: el parámetro *model* determina cuál de ellos se desea detectar, etc., según la Tabla 2.1.

Cada una de las opciones de *model* tiene distinto tiempo de procesamiento y de precisión, obteniendo resultados desde el 73% al 100% de presión según el modelo usado. En el algoritmo propuesto en este trabajo de usa "*FrontalFaceLBP*" para realizar la detección de rostro; este modelo posee el porcentaje de precisión más alta.

Además de la función *vision.CascadeObjectDetector*, existen otras funciones de Matlab como vision.PeopleDetector y peopleDetectorACF que permiten detectar personas, estos detectores se los detalla en el ANEXO C.

Tabla 2.1 Modelos de clasificación de *vision.CascadeObjectDetector*

Modelo de clasificación (parámetro *model*)	**Descripción**
'FrontalFaceCART'(Default)	Detecta rostros ubicados verticalmente y de frente. **Método para codificar las características:** características de Haar.
'FrontalFaceLBP'	Detecta rostros ubicados verticalmente y de frente. **Método para codificar las características:** patrones binarios locales LBP.
'UpperBody'	Detecta la región superior del cuerpo, esto incluye el área de la cabeza y de los hombros. **Método para codificar las características:** características de Haar.
'EyePairBig' 'EyePairSmall'	Detecta un par de ojos
'LeftEye' 'RightEye'	Detecta el ojo izquierdo y derecho de manera separada. **Método para codificar las características:** características de Haar.
'LeftEyeCART' 'RightEyeCART'	Detecta el ojo izquierdo y derecho de manera separada.
'ProfileFace'	Detecta rostros ubicados verticalmente. **Método para codificar las características:** características de Haar.
'Mouth'	Detecta bocas. **Método para codificar las características:** características de Haar.
'Nose'	Detecta narices. **Método para codificar las características:** características de Haar.

Las características de Haar que usan algunos de los modelos de la Tabla 2.1, se presentan en el ANEXO D.

2.4.1.1. Algoritmo de Viola Jones

El algoritmo de Viola Jones usado con la función *vision.CascadeObjectDetector* se realiza en 3 pasos:

Paso 1: Extracción de Características

Para la extracción de características el detector de usa el Patrón Binario Local (*Local Binary Pattern* o LBP).

LBP etiqueta cada pixel según el valor de pixeles vecinos. Como se muestra en la Figura 2.9, en un grupo de pixeles de 3x3, el valor central es el umbral, todos los números mayores o iguales al umbral tomarán el valor de 1 y los demás el cero. Seguidamente, desde el pixel ubicado a la izquierda del centro o umbral, se conforma el valor binario en sentido horario desde el valor menos significativo y luego se lo transforma a decimal. Si el pixel se encuentra en un borde, se puede completar los pixeles vecinos con un valor igual al del pixel o también se puede descartar los pixeles de los bordes. Posteriormente, a la imagen se la divide en bloques y se obtiene el histograma por cada bloque. Finalmente, los histogramas por cada bloque son concatenados [27], donde esta concatenación es el vector final de características (Figura 2.10).

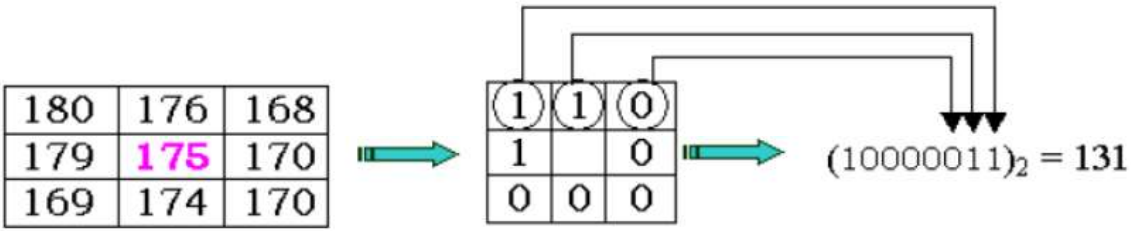


Figura 2.9 Ejemplo de patrón binario local con vecindario de 3x3 [27]
Los valores mayores o iguales a 175 son cambiados por 1 y los menores a 175 son cambiados a cero.

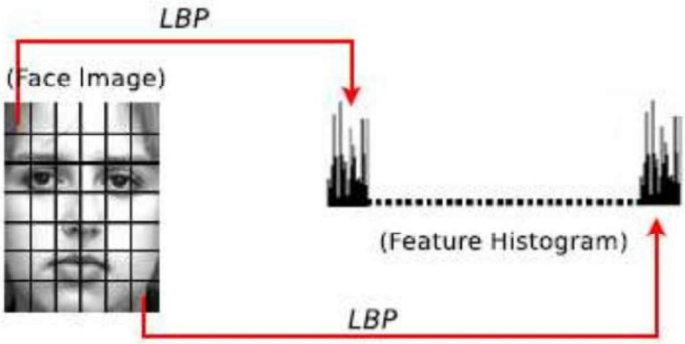


Figura 2.10 Ejemplo de histogramas de LBP en imagen dividida en 6x7 bloques [27]
Los histogramas por cada bloque son concatenados en un solo histograma.

Paso 2: Selección de características

Se usa Adaboost (*Adaptative Boosting*) para seleccionar un conjunto pequeño de las características y entrenar el clasificador. Su nombre, Adaboost, es debido a que este algoritmo de aprendizaje es usado para **aumentar (*boosting*)** el desempeño de clasificación de un algoritmo simple de aprendizaje, esto lo hace por medio de la combinación de clasificadores débiles para obtener un clasificador más fuerte. El

clasificador débil está diseñado para seleccionar una parte del histograma LBP que separe mejor los ejemplos positivos y negativos [27].

Para cada característica, el clasificador débil determina un umbral para la clasificación. Sea un clasificador débil $h_j(x)$ con una característica f_j, un umbral Θ_j y una pariedad p_j que indica la dirección de la desigualdad, así se tiene la Ecuación 2.1 [27]:

$$h_j(x) = \begin{cases} 1 & si\ p_j f_j(x) < p_j \theta_j \\ 0 & en\ otros\ casos \end{cases} \tag{2.1}$$

Proceso de aumento (*boosting*) [27]**:**

- Dadas imágenes de muestra (x_1,y_1), (x_2,y_2) . . . , (x_{n-1},y_{n-1}), (x_n,y_n) donde y_i= 0/1 para ejemplos negativos y positivos respectivamente (Figura 2.11). Las imágenes correspondientes a rostros son imágenes de muestras positivas y por lo tanto y_i es 1. Las imágenes no correspondientes a rostros son imágenes de muestra negativas por lo tanto y_i es 0.

Figura 2.11 Ejemplos de imágenes de muestras correspondientes a rostros y no rostros [28]. Los rostros al ser muestras positivas su valor y_i es 1 y las imágenes no correspondientes a rostros son muestras negativas y su valor y_i es 0.

- Se inicializan los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0|1$ respectivamente, donde m es el número total de muestras negativas y l el número total de muestras positivas.
- Para $t = 1, \dots, T$, donde T es el número de veces que se calcula los pesos:
    1. Se normaliza los pesos con la Ecuación 2.2,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \tag{2.2}$$

    Donde w_t es una probabilidad de distribución.

    2. Para cada característica j, se entrena el clasificador h_j, el cual está restringido para usar una sola característica. El error es evaluado con respecto a w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i - y_i) \right|$.

3. Se elige el clasificador h_t, con menor error ϵ_t.
4. Se actualizan los pesos con la Ecuación 2.3

$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i} \tag{2.3}$$

donde $e_i = 0$ si el ejemplo x_i es clasificado correctamente, $e_i = 0$ en otro caso y $\beta_t = \frac{e_t}{1-e_t}$.

- Al final el clasificador fuerte es el mostrado en la Ecuación 2.4:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t\, h_t(x) \geq \frac{1}{2}\sum_{t=1}^{T} \alpha_t \\ 0 & en\ otros\ casos \end{cases} \tag{2.2}$$

donde $\alpha_t = \log\left(\frac{1}{\beta_t}\right)$.

Paso 3: Clasificación en cascada

Se usa clasificadores en cascada para aumentar el desempeño de detección y reducir el tiempo computacional. Su disposición es en cascada porque realizan el procesamiento continuo de una imagen en una cadena de clasificadores. Los clasificadores más simples son usados para rechazar la mayoría de subventanas antes de llamar a clasificadores más complejos para reducir las tasas de falsos positivos: si un clasificador simple aplicado no es aprobado, es decir, no cumple con el umbral, ya no pasa al siguiente clasificador [29] tal como muestra la Figura 2.12. Los falsos positivos son la detección de rostro en donde no existe la presencia de uno.

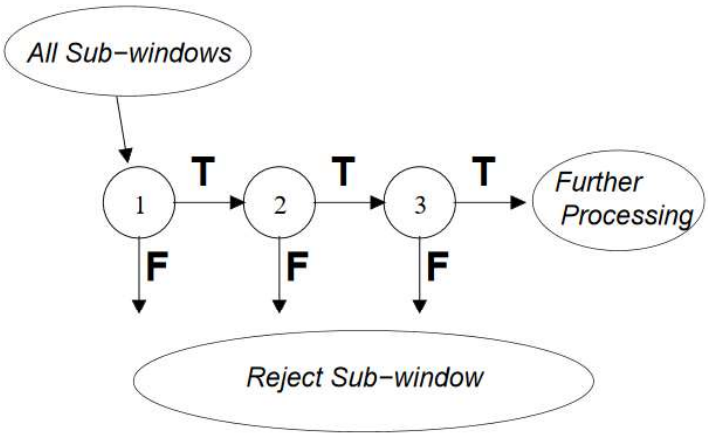


Figura 2.12 El resultado positivo de un clasificador simple pasa al siguiente clasificador y así sucesivamente. Si no, simplemente es rechazado [29]

2.4.1.2. Creación de detector

Para realizar la detección de rostro se crea el detector usando *FrontalFaceLBP*. *MergeThreshold* define el umbral con los criterios necesarios para declarar una detección

final en un área done hay múltiples detecciones. Los grupos que cumplen el umbral se combinan para producir un cuadro delimitador. El aumento de este umbral puede ayudar a suprimir detecciones falsas, pero si se elige un número excesivamente alto puede ya no existir detección. Se realizaron pruebas en escenarios variando la iluminación de una habitación y ocultando partes del rostro, para lo cual usando un umbral igual a 7 siempre existió detección del rostro; estas pruebas se encuentran en el ANEXO E. La Figura 2.13 muestra la detección de rostro con valores de *MergeThreshold* del 0 al 35, se evidencia que en valores muy pequeños la detección es errada y que con valores muy altos la detección no se realiza.

```
dcara=vision.CascadeObjectDetector('FrontalFaceLBP');    %detector    de
rostro
dcara.MergeThreshold = 7; % umbral de criterio para detección
```

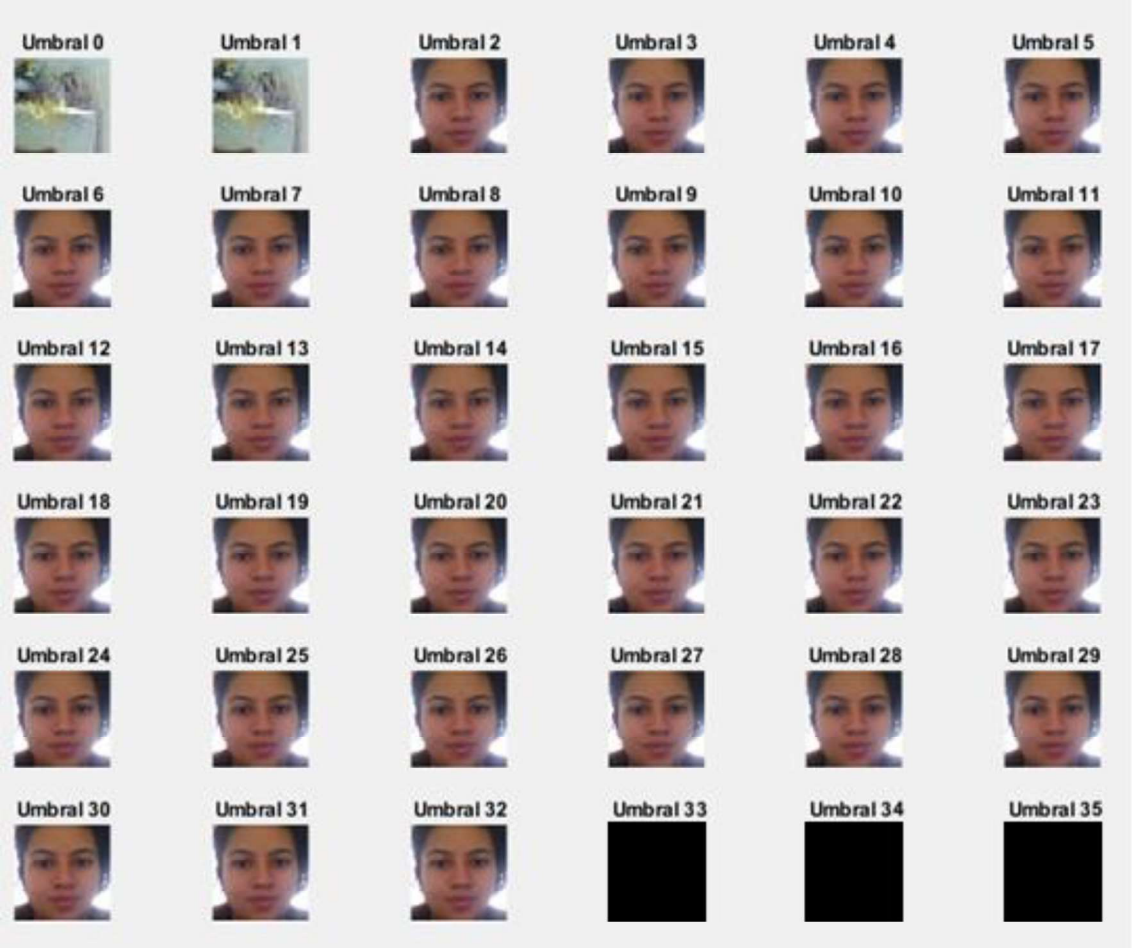


Figura 2.13 Resultados de probar un umbral del 0 al 35 para la detección de rostro
Para un umbral de 0 y 1 la detección es errada y para un umbral mayor a 33 la detección no se realiza. Se eligió el umbral de 7

2.4.1.3. Detección de rostro en imagen

Se usa el detector *dcara* creado anteriormente para obtener el rectángulo de ubicación (*xCara*) del rostro dentro de la imagen (*I*) obtenida en la Etapa 1. Este rectángulo posee cuatro puntos, el primero indica la coordenada en x, el segundo la coordenada en y, el tercero el ancho y el cuarto el alto. Pueden existir varios rostros detectados, por lo cual *xCara* tendrá varios rectángulos de ubicación y para elegir el más prominente se toma la primera fila de *xCara*.

```
%% Detección en imagen
xCara=dcara(I); % I:imagen adquirida en Etapa 1, dcara: detector creado
xCara=xCara(1,:);  %  xCara:rectángulo  de  ubicación  del  rostro  en  I.
Selección el primer rectángulo de ubicación
```

El código completo para la detección de rostro en Matlab queda de la siguiente manera:

```
% Luego de adquirir la Imagen I en la Etapa 1 se realiza:
%% Creación de detector
% Se usa el modelo FrontalFaceLBP para detectar rostros
dcara=vision.CascadeObjectDetector('FrontalFaceLBP');
dcara.MergeThreshold = 7; % umbral de criterio para detección
%% Detección en imagen
xCara=dcara(I); % I:imagen adquirida en Etapa 1, dcara: detector creado
xCara=xCara(1,:);  %  xCara:rectángulo  de  ubicación  del  rostro  en  I.
Selección el primer rectángulo de ubicación
```

2.4.2. FASE 2: RECORTE PROPORCIONAL DE OJO

Luego de detectar el rostro se realiza el recorte de la sección del ojo según proporciones anatómicas comunes [30], en la Figura 2.14 se pueden apreciar estas proporciones. Al dividir el ancho del rostro en 5 partes iguales (1/5), los ojos se encuentran en la segunda y cuarta división. Al dividir la longitud del rostro en tres partes iguales (1/3), los ojos se encuentran en la segunda división. Se realiza solamente el recorte de un ojo, ya que tiene el mismo comportamiento que el otro ojo y entonces es suficiente para investigar el estado de somnolencia del individuo. El usar un solo ojo se obtiene una matriz de menor valor y consecuentemente se reduce la carga computacional.

2.4.2.1. Obtención de rectángulo de recorte del ojo

Para obtener el rectángulo de recorte del ojo se usa el rectángulo de ubicación del rostro (*xCara*) y las proporciones de la Figura 2.14. *xCara* obtenido anteriormente corresponde al vector: $xCara = [x_0\ y_0\ w_0\ h_0]$. Se obtendrá el vector $xOjo = [x_1\ y_1\ \ w_1\ h_1]$ que corresponde al rectángulo de ubicación del ojo. Tomando en cuenta las proporciones anatómicas (Figura 2.14) se halla los puntos de la siguiente manera:

- x_1 es el un quinto del ancho del rostro más la coordenada en *x* donde inicia el rostro. La ecuación es la siguiente: $x_1 = x_0 + \frac{h_0}{5}$
- y_1 es el un tercio del alto del rostro más la coordenada en *y* donde inicia el rostro. La ecuación es la siguiente: $y_1 = y_0 + \frac{h_0}{3}$
- w_1 es el un quinto del ancho del rostro. La ecuación es la siguiente: $w_1 = \frac{w_0}{5}$
- h_1 es el un tercio del alto del rostro. La ecuación es la siguiente: $h_1 = \frac{h_0}{6}$

```
% xCara=[x0 y0 w0 h0]
% xCara(1)=x0, xCara(2)=y0, xCara(3)=w0, xCara(4)=h0
xOjo=[xCara(1)+xCara(3)/5 xCara(2)+xCara(4)/3 xCara(3)/5 xCara(4)/6];
```

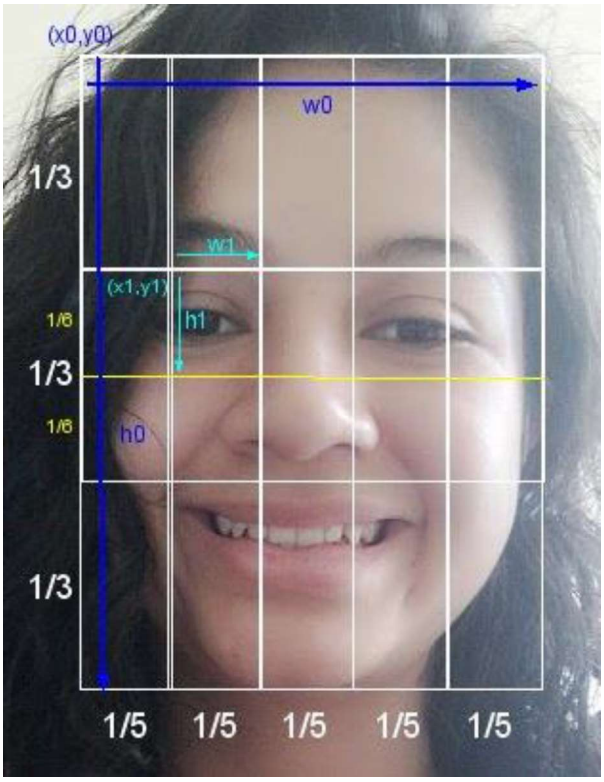


Figura 2.14 Proporciones anatómicas comunes de un rostro
Los ojos se encuentran a un tercio desde la parte superior del rostro y a un quinto desde alguno de los extremos del rostro.

2.4.2.2. Recorte del ojo

Para recortar la sección del ojo se usó la función *imcrop*, cuya sintaxis es Ojo=imcrop(I,*xOjo*). Donde *xOjo* es rectángulo de recorte del ojo obtenido anteriormente. Adicional al recorte del ojo se usa *imresize* para que al final todas las imágenes de ojos tengan un tamaño de 19x30. Se usa un tamaño de imagen de 19x30 debido a que el tamaño original de las imágenes de ojos está cerca de estos valores.

```
% Recorte de ojo
Ojo=imcrop(I,xCara); % I es imagen obtenida en Etapa 1
```

El código completo para el recorte del ojo queda de la siguiente manera:

```
% xCara=[x0 y0 w0 h0]
% xCara(1)=x0, xCara(2)=y0, xCara(3)=w0, xCara(4)=h0
xOjo=[xCara(1)+xCara(3)/5 xCara(2)+xCara(4)/3 xCara(3)/5 xCara(4)/6];
% Recorte de ojo
Ojo=imcrop(I,xCara); % I es imagen obtenida en Etapa 1
Ojo=imresize(Ojo,[19 30]); % cambio tamaño de matriz del ojo a 19x30
```

El diagrama de flujo de la Figura 2.15 muestra el proceso completo para codificar en Matlab la detección de ojo. Primero se realiza la detección de rostro usando *vision.CascadeObjectDetector*, posteriormente si existe un rostro en la imagen se realiza el recorte proporcional del ojo usando *imcrop*.

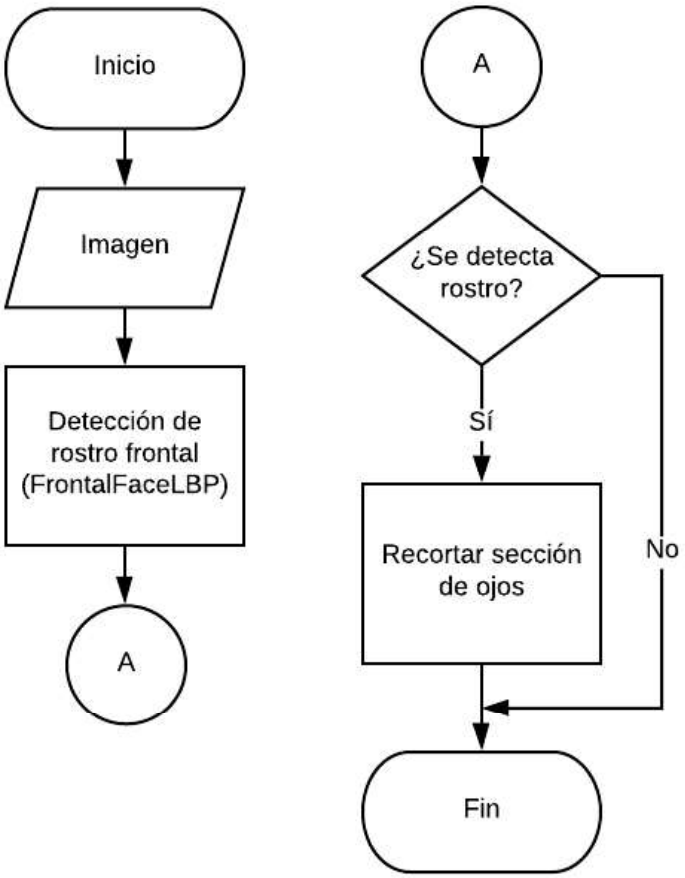


Figura 2.15 Diagrama de flujo para la extracción de la sección de ojo
Primero se realiza la detección de rostro y posteriormente se realiza el recorte de la sección del ojo en el rostro detectado.

El código completo de la función (*deteccion*) para la extracción de la sección del ojo en Matlab queda de la siguiente manera:

```
% I es la imagen obtenida en la Etapa 1
% Ojo es la imagen del ojo
function [Ojo] = deteccion( I )
%% Etapa 2 / Fase 1 Detección de Rostro
% Creación de detector, se usa el modelo FrontalFaceLBP para detectar
rostros
dcara=vision.CascadeObjectDetector('FrontalFaceLBP');
dcara.MergeThreshold = 7; % umbral de criterio para detección
% si no se detecta ni un rostro se devuelve Ojo=0
if isempty(xCara)
      Ojo=0;
else
%% Detección en imagen
      xCara=dcara(I); % I:imagen adquirida en Etapa 1, dcara: detector
      creado
      xCara=xCara(1,:); % xCara: rectángulo de ubicación del rostro en
      I. Selección el primer rectángulo de ubicación
%% Etapa 2 / Fase 2 Recorte proporcional del ojo
      % xCara=[x0 y0 w0 h0]
      % xCara(1)=x0, xCara(2)=y0, xCara(3)=w0, xCara(4)=h0
      xOjo=[xCara(1)+xCara(3)/5     xCara(2)+xCara(4)/3     xCara(3)/5
      xCara(4)/6];
      % Recorte de ojo
      Ojo=imcrop(I,xOjo); % Ojo es la imagen del ojo recortada de I
      Ojo=imresize(Ojo,[19 30]); % cambio tamaño de matriz del ojo a
      19x30
end
end
```

2.5. Etapa 3: Procesamiento de imagen del ojo

Para ambos métodos de clasificación se procesa de manera distinta las imágenes adquiridas (Figura 2.16). Para el Método 1 se realiza la ecualización del histograma de la imagen del ojo y para el Método 2 se normaliza y binariza la imagen del ojo.

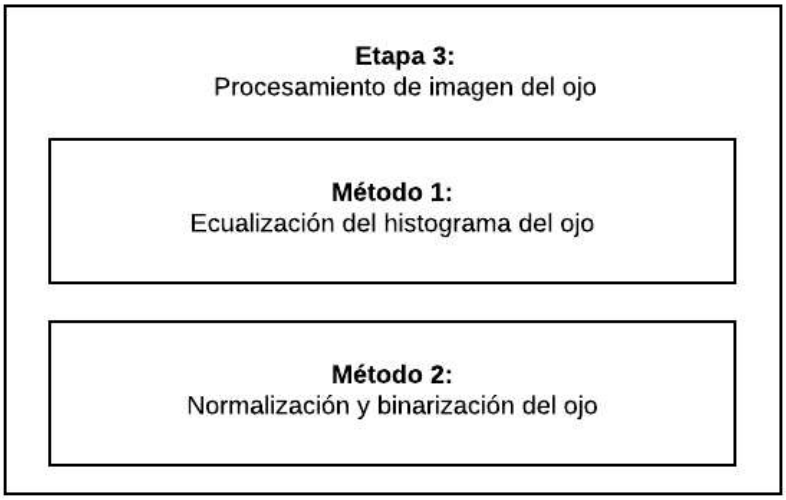


Figura 2.16 Procesamiento de imágenes para cada método de clasificación.
En el Método 1 se ecualiza la imagen del ojo. En el Método 2 se normaliza y binariza la imagen del ojo.

2.5.1. MÉTODO 1

A la imagen RGB obtenida se ecualiza su histograma. Se ecualiza la imagen del ojo ya que se realizaron varias pruebas de procesamiento para la entrada de la red neuronal y se encontró que el mejor resultado se presenta en imágenes ecualizadas, estos resultados se los presenta en la Tabla 3.2 de la Sección 3.1.3. Posteriormente se extrae el primer canal de la matriz y finalmente se convierte la matriz a un vector para que funcione como entrada a la red MLP (Figura 2.17).

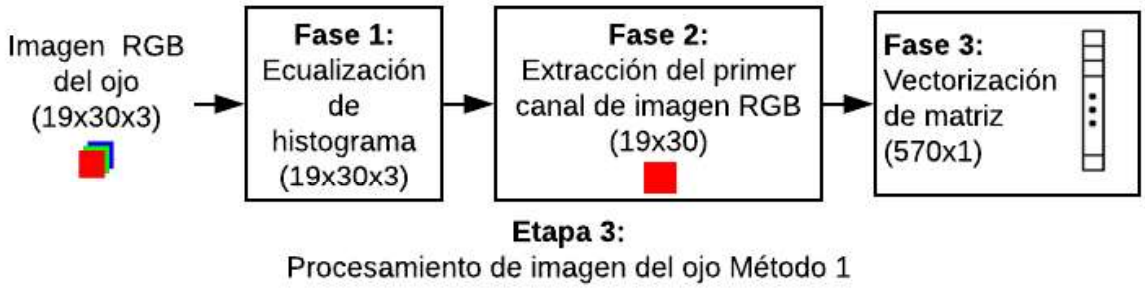


Figura 2.17 Esquema de procesamiento de imagen del ojo para el Método 1. En la Fase 1 se ecualiza el histograma de la imagen del ojo. En la Fase 2 se extrae el primer canal de la imagen ecualizada. En la Fase 3 se vectoriza la matriz de la imagen.

2.5.1.1. Fase 1: Ecualización de histograma

La ecualización de histograma permite mejorar el brillo y contraste de una imagen. El brillo representa la cantidad de luz y sombra en una imagen, cada pixel posee su propio valor, valores altos representan mayor brillo donde su máximo (255) sería el blanco y su mínimo el negro (0); por otro lado, el contraste representa la relación entre sombras y luces. Si el brillo de las luces es bajo y las sombras no son muy oscuras el contraste es débil, y si hay bastante diferencia en el rango de tonos claros y oscuros entre las luces y sombras el contraste es alto [31], la Figura 2.18 muestra ejemplos al combinar distintas cantidades de brillo y contraste.

Para que el histograma de la imagen tengo una distribución uniforme se procede a ecualizar el histograma de la imagen del ojo usando la función *histeq*, cuya sintaxis es J = histeq(I).

```
% Ojo es la imagen obtenida en la Etapa 2
Ojoeq=histeq(Ojo); % Ecualización del Ojo
```

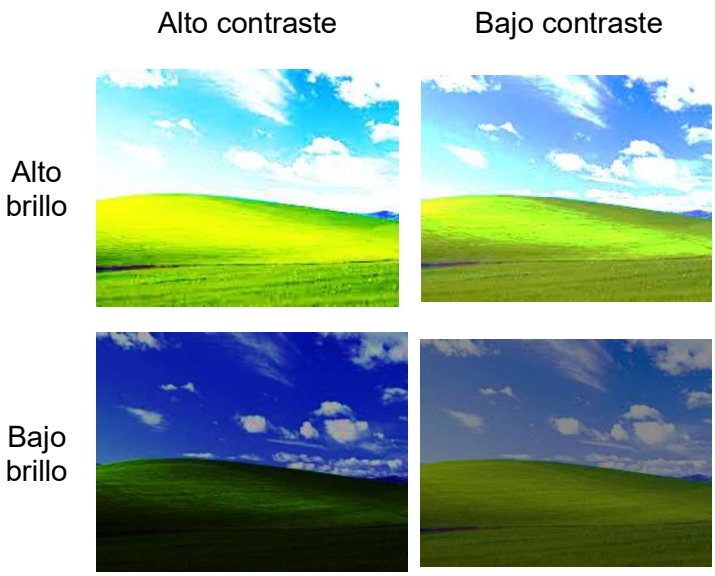


Figura 2.18 Ejemplo de una imagen cambiado sus valores de brillo y contraste.

En la Figura 2.19 se puede observar el resultado de ecualizar la imagen de un ojo cerrado y uno abierto. La imagen ecualizada resalta la sección del ojo, por lo que a la red neuronal se le facilita la clasificación.

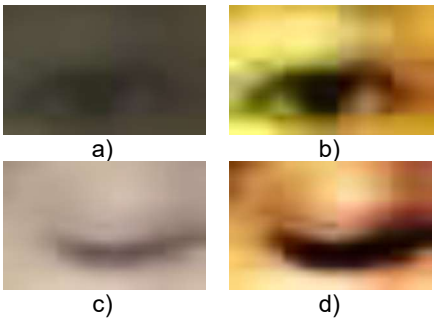


Figura 2.19 Resultado de ecualizar una imagen
a) Imagen original de ojo abierto. b) Imagen ecualizada de ojo abierto. c) Imagen original de ojo cerrado. d) Imagen ecualizada de ojo cerrado.

2.5.1.2. Fase 2: Extracción de primer canal de imagen RGB del ojo

Ya que una imagen RGB está compuesta de tres matrices (una denominada R, otra G y otra B), para poder pasar de matriz a vector, se necesita trabajar solamente con una de ellas. En nuestro caso, se extrae la primera matriz R de la imagen RGB del ojo.

```
Ojo1=double(Ojoeq(:,:,1)); % Extracción del primer canal RGB, se toma
la matriz R
```

2.5.1.3. Fase 3: Vectorización de matriz

Ya que la siguiente etapa de clasificación requiere como entrada un vector, se debe transformar la matriz R del ojo de 19x30 a vector en una sola columna de 570x1. Para la vectorización se usa la función *reshape*, cuya sintaxis es B = reshape(A, *size*).

```
Ojop =reshape(Ojop,[],1); % paso de matriz a vector para alimentar a la
red neuronal de la siguiente etapa
```

El código completo para el procesamiento de la imagen del ojo para el Método 1 queda de la siguiente manera:

```
% Ojo: imagen obtenida en Etapa 2
Ojoeq=histeq(Ojo); % Ecualización del Ojo
Ojop=double(Ojoeq(:,:,1)); % Extracción del primer canal RGB, se toma
la matriz R
Ojop =reshape(Ojop,[],1); % paso de matriz a vector para alimentar a la
red neuronal de la siguiente etapa
```

Otros métodos de procesamiento de imágenes como transformada FFT y DCT que se probó para entrenar la red neuronal se describen en el ANEXO F.

2.5.2. MÉTODO 2

La imagen RGB obtenida se la convierte a escala de grises para trabajar con una matriz de dos dimensiones y poder binarizar la imagen. Posteriormente para compensar el efecto de diferentes condiciones de iluminación, se normaliza la imagen en promedio y varianza [25], es decir que las imágenes tendrán una media igual a cero y una varianza igual a 1. Finalmente, se binariza la imagen usando un umbral τ_B. Se realiza esta binarización para en la Etapa 4 poder obtener los puntos de referencia del ojo. La Figura 2.20 muestra el esquema para el procesamiento de la imagen del ojo para el método 2.

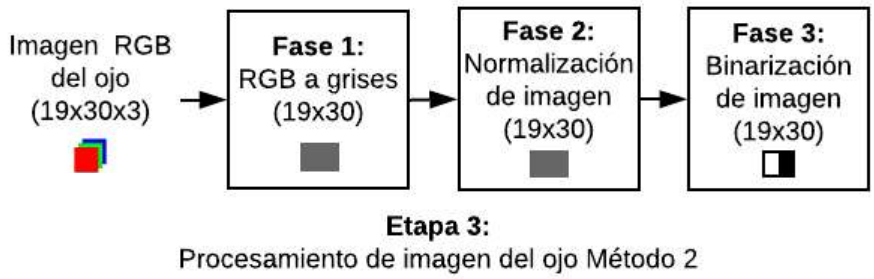


Figura 2.20 Esquema de procesamiento de imagen del ojo para el Método 2
En la Fase 1 se transforma a grises la imagen RGB. En la Fase 2 se normaliza la imagen. En la Fase 3 se binariza la imagen del ojo.

2.5.2.1. Fase 1: RGB a escala de grises

La imagen RGB del ojo se la transforma a escala de grises usando la función *rgb2gray*. El resultado es una matriz de 19x30.

```
Ig= rgb2gray(Ojo); % RGB a escala de grises para obtener una matriz de
dos dimensiones
```

2.5.2.2. Fase 2: Normalización de imagen

La imagen en escala de grises del ojo es normalizada. La Figura 2.21 muestra la imagen normalizada de un ojo abierto bajo distintas condiciones de iluminación: a) imagen en buenas condiciones de luz en una habitación iluminada con la fuente de luz ubicada frente a la persona, b) imagen en contra luz en una habitación iluminada con la fuente de luz ubicada detrás de la persona y c) imagen en muy malas condiciones de luz en una habitación sin iluminación.

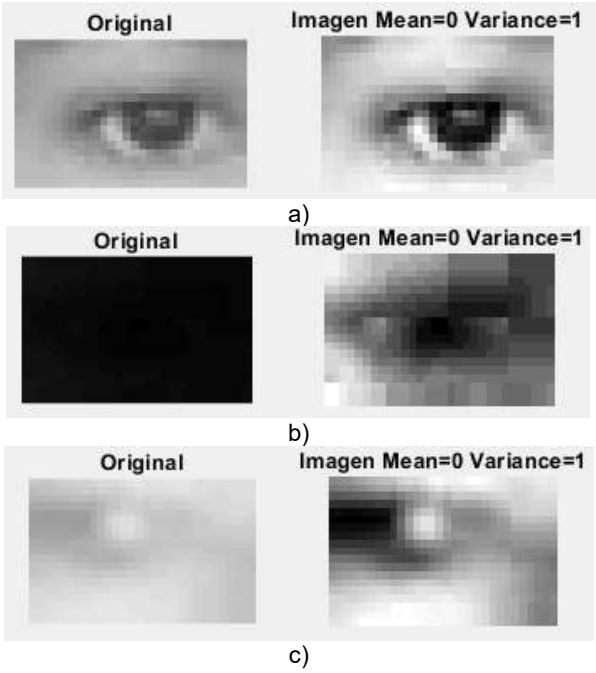


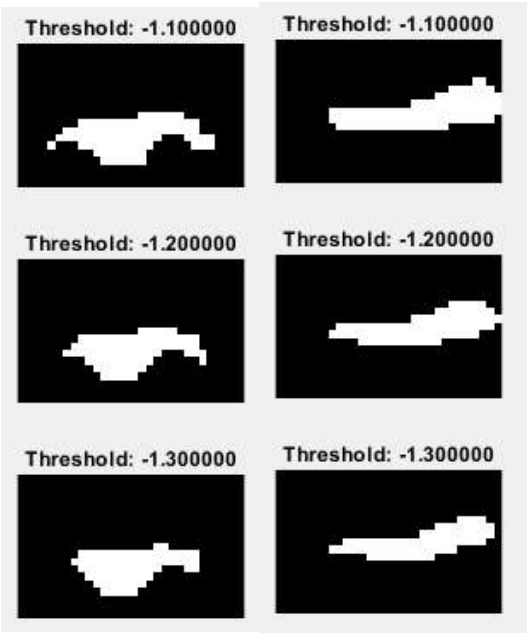
Figura 2.21 Imágenes de ojos abiertos con su imagen normalizada.
Al lado izquierdo se presentan las imágenes originales en escala de grises y a la derecha se presentan las mismas imágenes normalizadas. a) Imagen en buenas condiciones de luz, b) Imagen en contra luz, c) Imagen en muy malas condiciones de luz

Para normalizar la imagen se usa la siguiente ecuación: $J = \frac{I - mean2(I)}{std2(I)}$, donde J es la matriz normalizada e I es la matriz original. La función *mean2* obtiene el valor medio de los elementos de una matriz y la función *std2* obtiene la desviación estándar de los elementos de una matriz.

```
X=(Ig-mean2(Ig))/std2(Ig);  % Normalización de imagen del ojo para
compensar efecto de diferentes condiciones de luz
```

2.5.2.3. Fase 3: Binarización de imagen

El umbral τ_B, usado para binarizar la imagen, se ha obtenido realizando pruebas con distintos valores y comparando las imágenes binarizadas de un ojo abierto y cerrado para determinar un valor que funcione en ambos casos. Se han tomado 80 imágenes aleatorias de ojos abiertos y cerrados correspondientes a 8 personas distintas para obtener este valor para probar distintos valores τ_B y elegir un valor en el que el ojo se pueda diferenciar entre abierto y cerrado; las pruebas se detallan en el ANEXO G. La Figura 2.22 muestra la binarización de un ojo abierto y uno cerrado para diferentes valores umbral. El valor elegido es τ_B=-1.2. Si el valor del píxel evaluado es mayor que τ_B se cambia el valor a 0, caso contrario se cambia el valor a 1.



a) b)

Figura 2.22 Binarización del ojo con valores umbrales de -1.1, -1.2 y -1.3
a) Ojo abierto, b) ojo cerrado.

Finalmente, se realiza el cambio de formato a imagen binaria usando la función *im2bw*.

```
iOjo=X; % Se copia imagen normalizada a iOjo
iOjo(find(X>TB))=0; % Binarización, cambio a 0 TB=-1.2
iOjo (find(X<=TB))=1; % Binarización, cambio a 1 TB=-1.2
BW=im2bw(iOjo); % Cambio de formato a imagen binaria
```

El código completo para el procesamiento de la imagen del ojo para el Método 2 queda de la siguiente manera:

```
Ig= rgb2gray(Ojo); % RGB a escala de grises
X=(Ig-mean2(Ig))/std2(Ig); % Normalización de imagen del ojo
iOjo=X; % Se copia imagen normalizada a iOjo
iOjo(find(X>TB))=0; % Binarización, cambio a 0 TB=-1.2
iOjo (find(X<=TB))=1; % Binarización, cambio a 1 TB=-1.2
BW=im2bw(iOjo); % Cambio de formato a imagen binaria
```

2.6. ETAPA 4: CLASIFICACIÓN DE OJO ABIERTO/CERRADO

Para la clasificación de ojo abierto/cerrado se presentan los métodos (Figura 2.23):

- Método 1: usando una red neuronal MLP
- Método 2: usando marcadores sobre el ojo

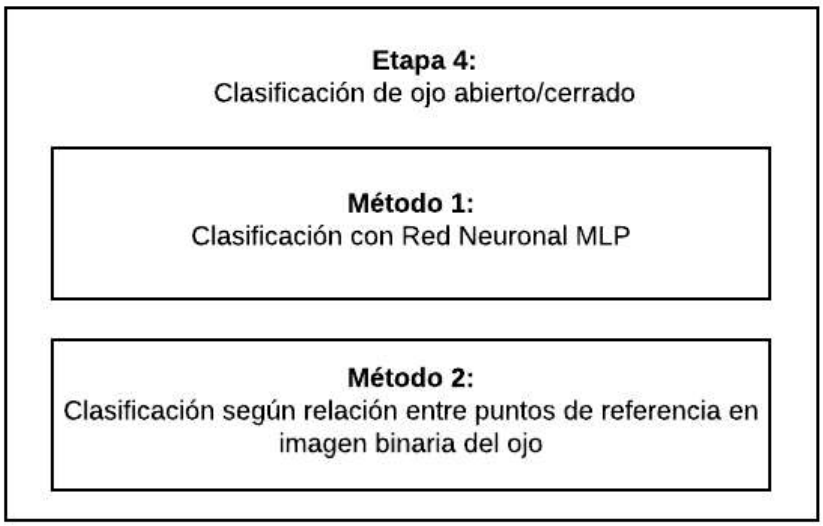


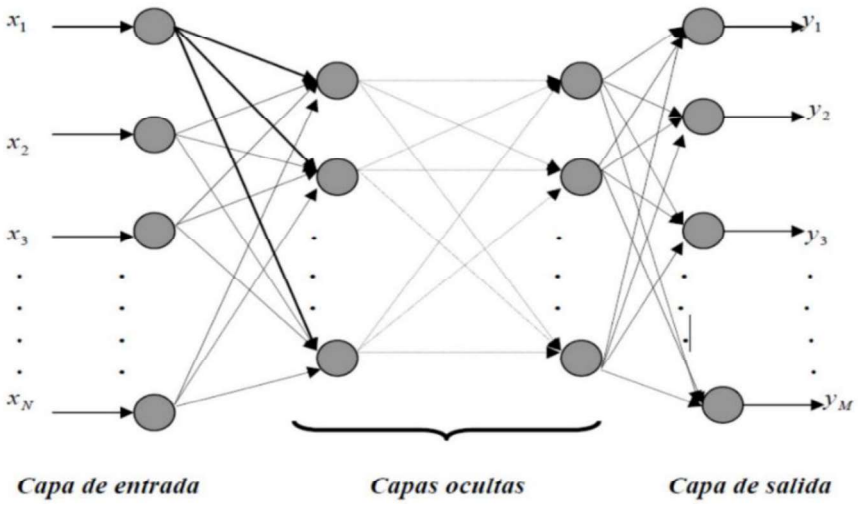
Figura 2.23 Métodos de clasificación de ojo abierto/cerrado

2.6.1. MÉTODO 1 DE CLASIFICACIÓN DE OJO ABIERTO/CERRADO

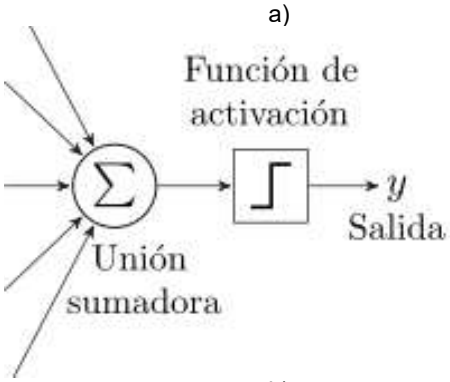
Para el método 1 se usa una red neuronal perceptrón multicapa (*Multilayer Perceptron* o MLP). Una MLP es una red que consiste de N entradas, j capas ocultas con k neuronas y M salidas (Figura 2.24a). Cada neurona posee una función de activación (Figura 2.24b). La función de activación más usada para cada neurona en redes MLP es la sigmoidal, sin embargo, puede usarse otra función [32]. En la Figura 2.25 se muestran varias funciones de activación usadas en redes MLP.

A continuación, se presentan las ecuaciones de la función de activación sigmoidal unipolar, o también conocida como función logística:

- Función sigmoidal unipolar: $\sigma(\mathrm{x}) = \frac{1}{1+\mathrm{e}^{-x}}$



a)



b)

Figura 2.24 a) MLP con N entradas y M salidas, las neuronas en la capa oculta son los círculos grises [33]. b) Constitución de neurona artificial [34].

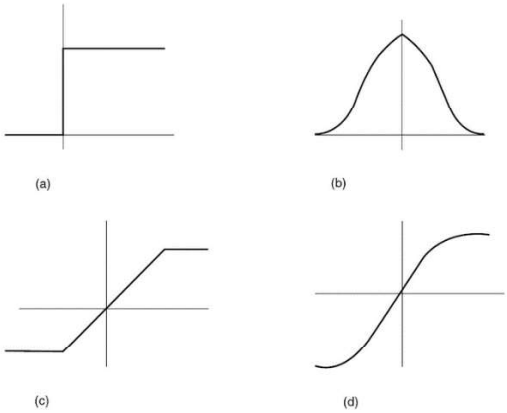


Figura 2.25 Ejemplos de funciones de activación usadas para neuronas artificiales [31]. (a) Función paso, (b) Gaussiana; (c) linear a trozos y (d) sigmoidal.

Una red MLP se deriva del procesamiento no lineal en las neuronas ocultas. Por este motivo se usa para obtener la forma de patrones que sean muy complejos y para realizar clasificación entre clases.

Entre las aplicaciones de las redes MLP está la clasificación de patrones, estimación de funciones y predicciones. Por este motivo en el algoritmo propuesto se usa una red MLP para clasificar entre ojos abiertos y cerrados.

En el algoritmo propuesto se usa una red MLP cuyas entradas son los pixeles de la imagen en escala de grises cuyo histograma ha sido ecualizado, en este caso son 570 pixeles ya que la dimensión de la imagen del ojo es de 19x30 pixeles y la salida de la red neuronal son los estados “abierto” o “cerrado”. Esta red neuronal tiene una capa oculta construida con 10 neuronas artificiales, función de transferencia sigmoidal en la capa oculta y una función de transferencia *softmax* en la capa de salida (Figura 2.26).

Cada entrada tiene un peso asociado *w*, este se va modificando al momento de entrenar la red neuronal [35]. Cada nodo (neurona artificial) de la capa oculta y de la capa de salida poseen una entrada extra llamada bias *b*, este es un valor constante que permite controlar el valor al que se activará la función de activación; por ejemplo si no se tiene el valor *bias* y debido a que la neurona artificial multiplica la entrada por el peso *w,* al tener una entrada igual a 0 no se podrá obtener una salida diferente de cero al no existir este valor constante [36].

Se usa la configuración con una sola capa oculta y función de activación sigmoidal ya que este tipo de red neuronal es capaz de aproximar cualquier función continua, por este motivo es llamado “aproximador universal” [37]. La elección de 10 neuronas se realizó luego de entrenar varias redes neuronales con distinto número de neuronas, los resultados de estos entrenamientos se detallan en la sección 3.1.3.1.

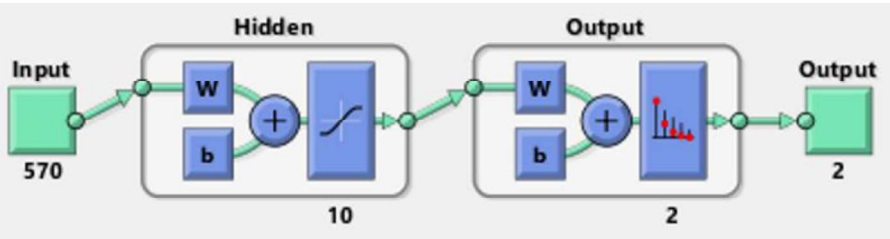


Figura 2.26 Red neuronal usada para la clasificación de ojo abierto/cerrado

La Figura 2.27 presenta el diagrama de red neuronal en función de las 570 entradas, 10 neuronas artificiales y 2 salidas.

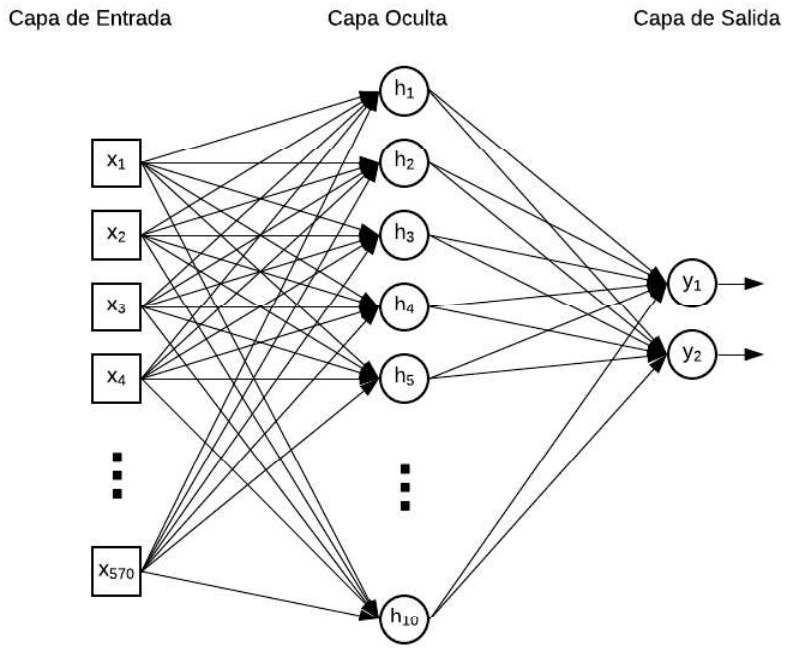


Figura 2.27 Diagrama de red neuronal usado para la clasificación de ojo abierto/cerrado

A los nodos de la capa oculta entran los valores de los pixeles de la imagen del ojo, en total 570 datos, estos valores son multiplicados por sus pesos y sumados, posteriormente pasan por la función de activación sigmoidal y entran a un nodo de la capa de salida (Figura 2.28a). En los nodos de la capa de salida también se cada entrada con su peso, en este caso la entrada es la salida de la capa oculta, estos valores son sumado y pasan por la función *softmax* (Figura 2.28b). La función softmax es la normalización de la función logística y es usada para comprimir un vector de datos: $F(x) = \frac{e^{x_i}}{\sum_{j=0}^{k}(e^{x_j})}$, $i = 1,2, \dots$.

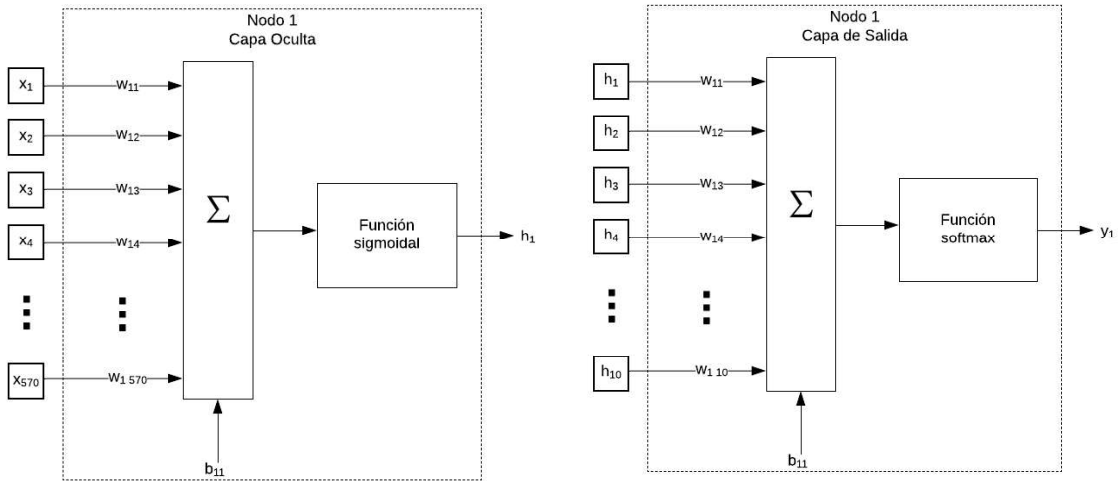


Figura 2.28 Estructura de un nodo de la capa oculta y de la capa de salida

Para el entrenamiento y prueba de la red neuronal se usó una base de datos formada de la siguiente manera:

- Imágenes de ojos abiertos y cerrados correspondientes a 8 personas diferentes.
- Tiene 300 imágenes por persona, en total 2400 imágenes.
- Las imágenes fueron adquiridas en ambientes interiores bajo condiciones de luz uniformes con una media de 300.8 luxes y desviación estándar de 17.29 luxes. Los valores de los luxes fueron medidos con el sensor de luz de un celular móvil.

La Figura 2.29 muestra algunas imágenes de muestra de ojos abiertos de la base de datos correspondientes a una sola persona.

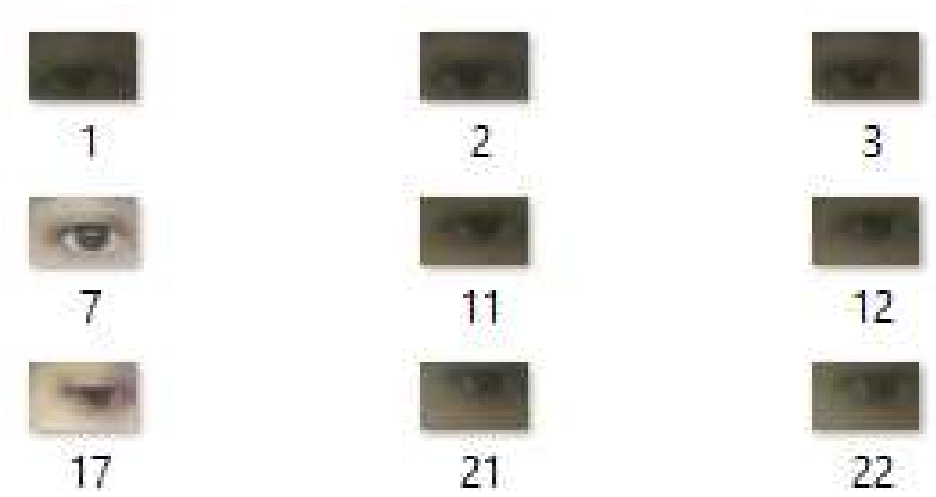


Figura 2.29 Muestras de la base de datos correspondientes a una sola persona.

La clasificación de ojo abierto/cerrado usando la red MLP se realiza en 5 fases. Primero a todas las imágenes de la base de datos que han sido vectorizadas se las concatena una junto la otra para formar la matriz de entrada de la red neuronal. Posteriormente se crea y configura la red neuronal. Usando la matriz de entrada y la red configurada se procede a entrenar la red neuronal. A continuación, cada imagen nueva que sea adquirida es evaluada con la red neuronal ya entrenada. Finalmente se clasifica el ojo como abierto o cerrado (Figura 2.30).

El diagrama de flujo de la Figura 2.31 muestra el proceso para clasificar la imagen de un ojo como abierto o cerrado usando una red MLP en Matlab. Primero se entrena la red neuronal, posteriormente se evalúa la imagen con la red y se realiza la clasificación de ojo abierto o cerrado.

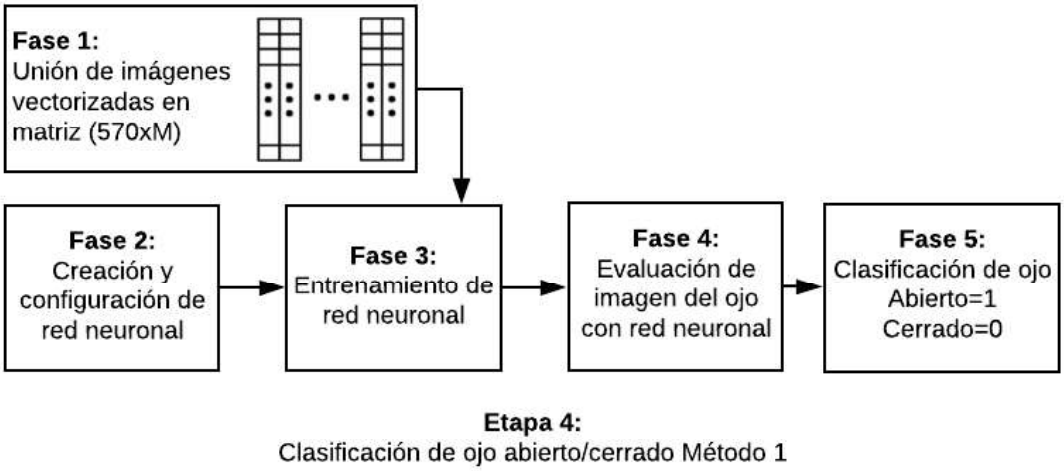


Figura 2.30 Esquema de clasificación de ojo abierto/cerrado para el Método 1
En la Fase 1 se concatenan las imágenes vectorizadas. En la Fase 2 se crea y configura la red neuronal. En la Fase 3 se entrena la red neuronal. En la Fase 4 se evalúa la imagen del ojo con la red neuronal entrenada. En la Fase 5 se realiza la clasificación de ojo como abierto (1) o cerrado (0).

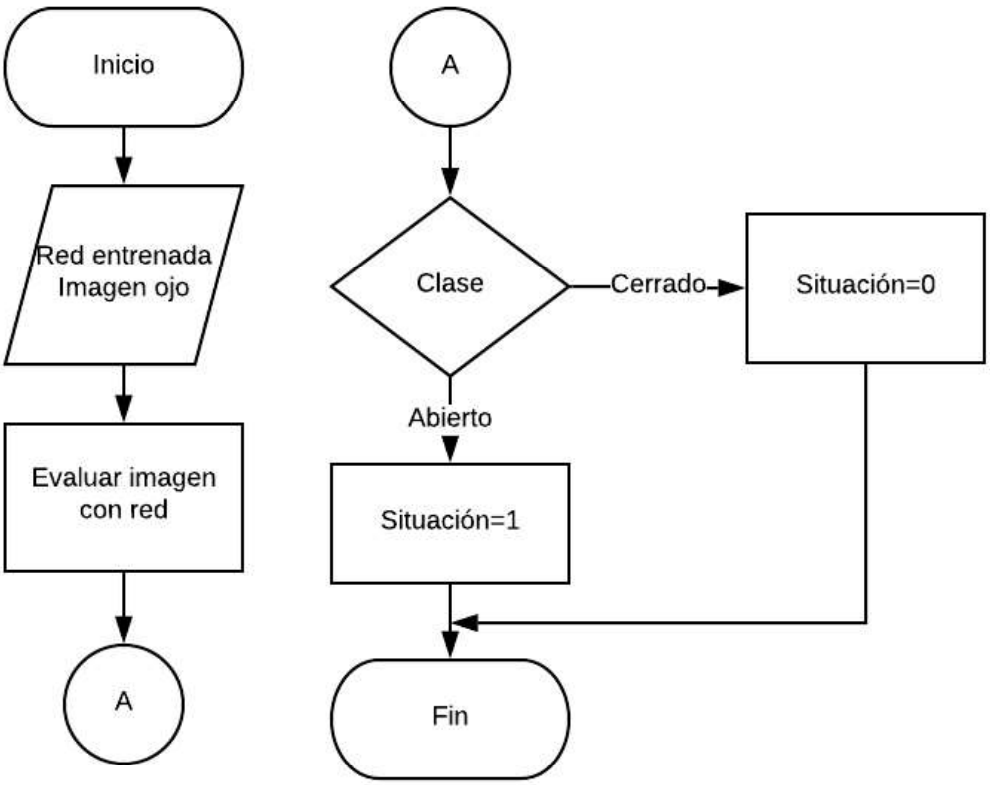


Figura 2.31 Diagrama de flujo de clasificación de ojo abierto y cerrado Método 1
Con la red entrenada se evalúa cada imagen nueva y le asigna una clase: Abierto(1) o Cerrado(0).

2.6.1.1. Fase 1: Concatenación de imágenes vectorizadas

Para entrenar la red neuronal se necesita una matriz *P* formada por las imágenes de ojos abiertos y cerrados de la base de datos, cada columna representa una imagen distinta. Para que la red pueda identificar si la imagen de entrenamiento corresponde a un ojo abierto o cerrado se crea la matriz *T*, la columna Ti especifica la clase correspondiente de la imagen Pi.

Debido a que las imágenes de ojos abierto y cerrados se encuentran almacenadas en formato RGB y en directorios diferentes (*rabierto* y *rcerrado*). Primero se debe realizar la lectura de cada una de las imágenes de ambos directorios y procesar las imágenes como lo indica la Etapa 3 para el Método 1, es decir hay que ecualizar la imagen, extraer el primer canal y vectorizar la matriz.

```
% rabierto: ruta de directorio donde están los ojos abiertos
% rcerrado: ruta de directorio donde están los ojos cerrados
% filename: nombre de archivo de ojo abierto
% filename2: nombre de archivo de ojo cerrado
filename=fullfile(rabierto,sprintf('%d.jpg',i)); % I número de imagen
filename2=fullfile(rcerrado,sprintf('%d.jpg',i)); % I número de imagen
%% Ojo abierto
% Procesamiento de imagen de ojo abierto. Etapa 3
x= imread(filename); % lectura de imagen ojo abierto
x=histeq(x); % ecualización de imagen ojo abierto
x=x(:,:,1); % extracción del primer canal
y=reshape(x,[],1); % vectorización de imagen ecualizada
%% Ojo cerrado
% Procesamiento de imagen de ojo cerrado. Etapa 3
x2= imread(filename2); % lectura de imagen ojo cerrado
x2=histeq(x2); % ecualización de imagen ojo cerrado
x2=x2(:,:,1); % extracción del primer canal
y2=reshape(x2,[],1); % vectorización de imagen ecualizada
```

Mientras se realiza la lectura y procesamiento de las imágenes, a los vectores resultantes (y para ojos abiertos y y2 para ojos cerrados) se los va concatenando en una sola matriz (A para ojos abiertos y A2 para ojos cerrados).

```
A(:,i)=y; % concatenación de imágenes de ojos abiertos
A2(:,i)=y2; % concatenación de imágenes de ojos cerrados
```

Al finalizar la lectura de cada una de las imágenes, se concatena la matriz de ojos abiertos (A) con la matriz de ojos cerrados ($\mathrm{A2}$), formando la matriz P anteriormente mencionada. Debido a que la clasificación será en dos clases, cada una de las clases se identificará por un vector de 2x1. El identificador de clase para el ojo abierto será $\mathrm{a} = [1\ 0]'$ y el identificador de clase para el ojo cerrado será $\mathrm{b} = [0\ 1]'$. La matriz T se formará de las columnas a o b

dependiendo de si la columna de la misma posición de la matriz P corresponde a un ojo abierto o cerrado. La matriz P por resultar de la concatenación de imágenes se encuentra en formato uint8, para que sirva de entrada a la red neuronal se transforma los datos de P a double.

```
a = [1 0]'; % identificador de clase de ojo abierto
b = [0 1]'; % identificador de clase de ojo cerrado
P = [A A2]; % concatenación de matrices de ojos abiertos y cerrados
P = double(P); % cambio de uint8 a double
aa=size(A); bb=size(A2);
T = [repmat(a,1,aa(2)) repmat(b,1,bb(2))]; % matriz de clases de P
```

El código completo de la función (*datos*) para crear las matrices P y T es el siguiente:

```
% rabierto: ruta de directorio donde están los ojos abiertos
% rcerrado: ruta de directorio donde están los ojos cerrados
function [P , T] = datos(rabierto,rcerrado)
% se obtiene la cantidad de imágenes que tiene cada directorio
info1=dir(rabierto); % información del directorio de ojos abiertos
para saber e número de elementos
info2=dir(rcerrado); % información del directorio de ojos abiertos
para saber e número de elementos
p=max([length(info1) length(info2)]); % se repite la lectura hasta
alcanzar p la mayor cantidad de elementos en uno de los directorios
z=570; % el tamaño de cada vector es de 19x30=570
for i=1:p-2 % el proceso se repite hasta acabar con todas las imágenes
      filename=fullfile(rabierto,sprintf('%d.jpg',i)); % imagen ojo
      abierto
      filename2=fullfile(rcerrado,sprintf('%d.jpg',i)); % imagen ojo
      cerrado
      %% Ojo abierto / Procesamiento Etapa 3
      x= imread(filename); % lectura de imagen ojo abierto
      x=histeq(x); % ecualización de imagen ojo abierto
      x=x(:,:,1); % extracción del primer canal
      y=reshape(x,[],1); % vectorización de imagen ecualizada
      A(:,i)=y; % concatenación de imágenes de ojos abiertos
      %% Ojo cerrado / Procesamiento Etapa 3
      x2= imread(filename2); % lectura de imagen ojo cerrado
      x2=histeq(x2); % ecualización de imagen ojo cerrado
      x2=x2(:,:,1); % extracción del primer canal
      y2=reshape(x2,[],1); % vectorización de imagen ecualizada
      A2(:,i)=y2; % concatenación de imágenes de ojos cerrados
end
%% Creación de matrices P y T
a = [1 0]'; % identificador de clase de ojo abierto
b = [0 1]'; % identificador de clase de ojo cerrado
P = [A A2]; % concatenación de matrices de ojos abiertos y cerrados
P = double(P); % cambio de uint8 a double
aa=size(A); bb=size(A2); % se ontiene el tamaño de matrices A y A2
T = [repmat(a,1,aa(2)) repmat(b,1,bb(2))]; % matriz de clases de P
end
```

2.6.1.2. Fase 2: Creación y configuración de red neuronal

Para crear y configurar la red neuronal en Matlab se usa el Deep Learning Toolbox, este Toolbox proporciona un conjunto de herramientas para diseñar e implementar redes neuronales superficiales y profundas.

Para ingresar al asistente se lo puede hacer por medio de *nnstar* y elegir la opción de reconocimiento de patrones y clasificación. También se puede ingresar directamente con *nprtool*. En la Figura 2.32 se observa el asistente de Matlab para crear redes neuronales.

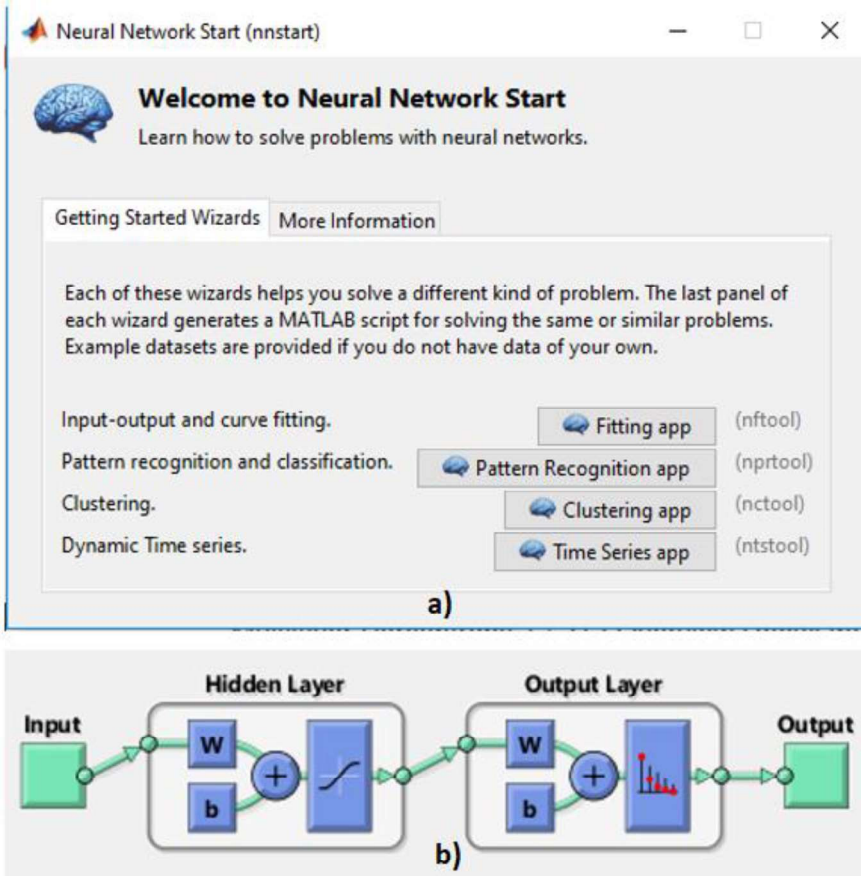


Figura 2.32 a) asistente para crear redes neuronales en Matlab, b) red neuronal para reconocimiento de patrones.

La red está formada por las entradas, una capa oculta con función de activación sigmoidal, una capa de salida con función de activación softmax y las salidas. La red se entrena con el método de gradiente conjugado escalado, el cual es el encargado actualizar los pesos.

```
trainFcn = 'trainscg';    % Función de entrenamiento: Scaled conjugated
gradient baackpropagation
```

Los parámetros por defecto de la red MLP se muestran en la Tabla 2.2.

Tabla 2.2 Parámetros por defecto de redes neuronales de clasificación en Matlab.

Parámetro	**Valor**	**Descripción**
net.trainParam.epochs	1000	Número máximo de épocas para entrenar.
net.trainParam.goal	0	Objetivo de desempeño. Es el error esperado.
net.trainParam.time	inf	Tiempo máximo para entrenar en segundos
net.trainParam.min_grad	1e-6	Gradiente de rendimiento mínimo
net.trainParam.max_fail	5	Máximo de fallos de validación

Un valor importante que determina el desempeño de la red neuronal es el tamaño de la capa oculta, por medio de pruebas se ha determinado que al usar un tamaño de 10 neuronas se tiene el mayor porcentaje de clasificación correcta en comparación al usar otra cantidad de neuronas, estos resultados están detallados en la sección 3.1.3.1.

```
hiddenLayerSize = 10; % Tamaño de la capa oculta es de 10 neuronas
net = patternnet(hiddenLayerSize, trainFcn); % creación de red de
reconocimiento de patrones
```

La red de reconocimiento de patrones se crea usando la función *patternnet*.

```
net = patternnet(hiddenLayerSize, trainFcn); % creación de red de
reconocimiento de patrones
```

También se debe configurar la división de los datos. Se usa el 70% de los datos para entrenar la red, el 15% de los datos para determinar cuando el entrenamiento ya no se puede mejorar y el 15% de los datos restantes se lo usa para probar el funcionamiento de la red neuronal. Esta división se la hace de manera aleatoria. Se asigna el 70% de imágenes para entrenar la red con el objetivo que al momento de realizar una prueba sea menos probable que tenga como entrada una imagen no conocida. El ANEXO H presenta resultados obtenidos al variar las proporciones de esta división de datos.

```
% División de datos para entrenamiento, evaluación y test
net.divideParam.trainRatio = 70/100; % 70% entrenamiento
net.divideParam.valRatio = 15/100; % 15% evaluación
net.divideParam.testRatio = 15/100; % 15% test
```

El código completo de la función (*redneu*) en donde se configura la red neuronal es el siguiente:

```
% P y T matrices obtenidas en la Etapa 4 / Fase 1
function [ net ] = redneu( P,T )
trainFcn = 'trainscg'; % Función por defecto para reconocimiento de
patrones
hiddenLayerSize = 10; % Tamaño de la capa oculta es de 10 neuronas
net = patternnet(hiddenLayerSize, trainFcn);
net.trainParam.epochs=1000; % Valor por defecto
% División de datos para entrenamiento, evaluación y test
net.divideParam.trainRatio = 70/100; % 70% entrenamiento
net.divideParam.valRatio = 15/100; % 15% evaluación
net.divideParam.testRatio = 15/100; % 15% test
[net,tr] = train(net,x,t); % Entrenamiento de red
end
```

2.6.1.3. Fase 3: Entrenamiento de la red neuronal

Para entrenar la red neuronal se usa las matrices *P* y *T* obtenidas en la Fase 1, la función *redneu* de la Fase 2. La red resultante se la guarda en el directorio con la función *save*. El código para entrenar la red neuronal es el siguiente:

```
% rabierto: ruta de directorio donde están los ojos abiertos
% rcerrado: ruta de directorio donde están los ojos cerrados
[P,T]=datos(rabierto,rcerrado); % función datos de Fase 1
P=double(P); % se cambia el formato de la matriz de uint8 a double
save net net % Almacenamiento de la red creada
```

2.6.1.4. Fase 4: Evaluación de la imagen del ojo

Para evaluar una nueva imagen fuera de la base de datos que sirvió para el entrenamiento, es necesario que esta imagen se encuentre ecualizada, vectorizada y en formato *double*.

```
% Ojoeq: Imagen ecualizada obtenida en la Etapa 3 / Fase 1
Ojoeq =double(Ojoeq (:,:,1)); % extracción de primer canal RGB y
cambio de formato a double
Ojoeq =reshape(Ojoeq,[],1); % vectorización de la matriz
```

Al cumplir con los requisitos anteriores se puede evaluar la imagen con la red obtenida en la Fase 3. El resultado de evaluar la imagen es un vector *output* de 2x1. Para encontrar la posición del valor máximo se usa la función *max*. Este valor permitirá realizar la clasificación en la Fase siguiente.

```
output=net(nueva); % evaluación con la red neuronal
[M,I]=max(output); % máximo de vector output
```

El código completo de la función (*evaluación*) para evaluar la imagen del ojo es el siguiente:

```
% Ojoeq: imagen obtenida en Etapa 3 / Fase 1
% net: red neuronal entrenada en Etapa 4 / Fase 3
function [ I ] = evaluacion( Ojoeq , net)
Ojoeq =double(Ojoeq (:,:,1)); % extracción de primer canal RGB
Ojoeq =reshape(Ojoeq,[],1); % vectorización de la matriz
output=net(nueva); % evaluación con la red neuronal
[M,I]=max(output); % máximo de vector output
end
```

2.6.1.5. Fase 5: Clasificación de ojo abierto/cerrado

Para realizar la clasificación de ojo abierto/cerrado se hace uso del valor *I* que se obtiene en la Fase 4. *I* indica la posición del valor máximo del vector resultante al evaluar la imagen. Si *I* es igual a 1, el ojo será clasificado como abierto, caso contrario se lo clasificará como cerrado. En la clasificación de ojo abierto/cerrado, el estado abierto está representado por el valor 1 y el estado de cerrado por el valor de 0. El código que realiza esta clasificación es el siguiente.

```
% I valor obtenido en la Fase 4
if(I==1) % decisión para la clasificación de ojo
      fprintf('Abierto'); % se imprime en pantalla el resultado
      situacion=1; % la situación del ojo es abierto (1)
else
      fprintf('Cerrado'); % se imprime en pantalla el resultado
      situacion=0; % la situación del ojo es cerrado (0)
end
```

2.6.2. MÉTODO 2 DE CLASIFICACIÓN DE OJO ABIERTO/CERRADO

La clasificación de ojo abierto/cerrado usando puntos de referencia sobre el ojo se realiza en 3 fases (Figura 2.33). Primero se obtienen los 4 puntos de referencia sobre la imagen binarizada del ojo. Posteriormente se calcula un valor usando los 4 puntos de referencia. Finalmente se usa este valor para clasificar el ojo como abierto o cerrado.

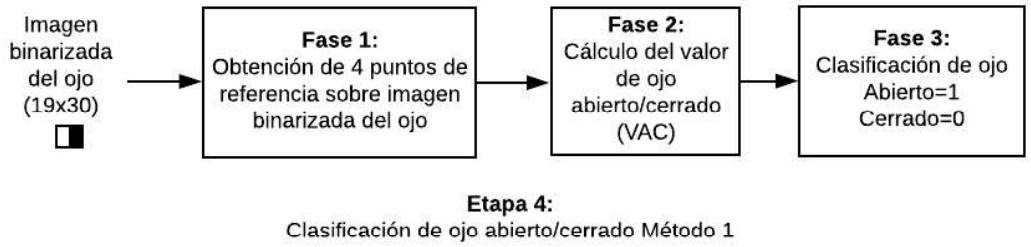


Figura 2.33 Esquema de clasificación de ojo abierto/cerrado para el Método 2
En la Fase 1 se obtienen los 4 puntos de referencia sobre la imagen binarizada del ojo. En la Fase 2 se calcula el valor de ojo abierto/cerrado (VAC). En la Fase 3 se realiza la clasificación de ojo como abierto (1) o cerrado (0).

2.6.2.1. Fase 1: Obtención de 4 puntos de referencia

En este método se han usado 4 puntos de referencia en la imagen binaria del ojo como se muestra en la Figura 2.34: uno del extremo superior $S(S_x, S_y)$, uno del extremo inferior $I(I_x, I_y)$, uno del extremo derecho $D(D_x, D_y)$ y otro del extremo izquierdo $Iz(Iz_x, Iz_y)$. Estos puntos están representados en coordenadas rectangulares (x, y), los cuales se obtienen de la siguiente manera:

- Marcador Izquierdo (Iz): Iz_x es la primera columna cuya suma es mayor a cero; Iz_y es el punto medio entre el primer y último valor de la columna Iz_x diferente de cero. El código para obtener el punto Iz es el siguiente:

```
izx= find(sum(Image)>0,1); % 1° columna con suma mayor a cero
izy1=find(Image(:,izx)>0,1); % 1° valor diferente a cero de
columna izx
izy2=find(Image(:,izx)>0,1,'last'); % último valor diferente a
cero de columna izx
izy=izy1+round(abs(izy1-izy2)/2); % punto medio entre izy1 e izy2
```

- Marcador Derecho (D): D_x es la última columna cuya suma es mayor a cero; D_y es el mismo valor de Iz_y. El código para obtener el punto D es el siguiente:

```
dx=find(sum(Image)>0,1,'last'); % última columna con suma mayor
a cero
dy=izy; % punto medio entre izy1 e izy2
```

- Marcador Inferior (I): I_x es el punto medio entre los puntos Iz_x y D_x; I_y es la última fila cuya suma es mayor a cero. El código para obtener el punto I es el siguiente:

```
iy=find(sum(Image(:,6:24),2)>0,1,'last'); % última fila con suma
mayor a cero
ix=izx+round(abs(dx-izx)/2); % punto medio entre los puntos izx
y dx
```

- Marcador Superior (S): S_x es el mismo valor de I_x; S_y es el primer punto mayor que cero en la columna S_x. El código para obtener el punto Sx es el siguiente:

```
sx=ix; % punto medio entre los puntos izx y dx
sy=find(sum(Image(:,sx),2)>0,1); % 1° punto mayor a cero en
columna sx
```

El diagrama de flujo de la Figura 2.35 muestra el proceso para clasificar la imagen de un ojo como abierto o cerrado usando puntos de referencia en una imagen binaria del ojo. Primero se binariza la imagen, posteriormente se encuentra los cuatro puntos de referencia sobre la imagen binaria del ojo. Luego se calcula y evalúa el valor VAC para finalmente realizar la clasificación de ojo abierto o cerrado.

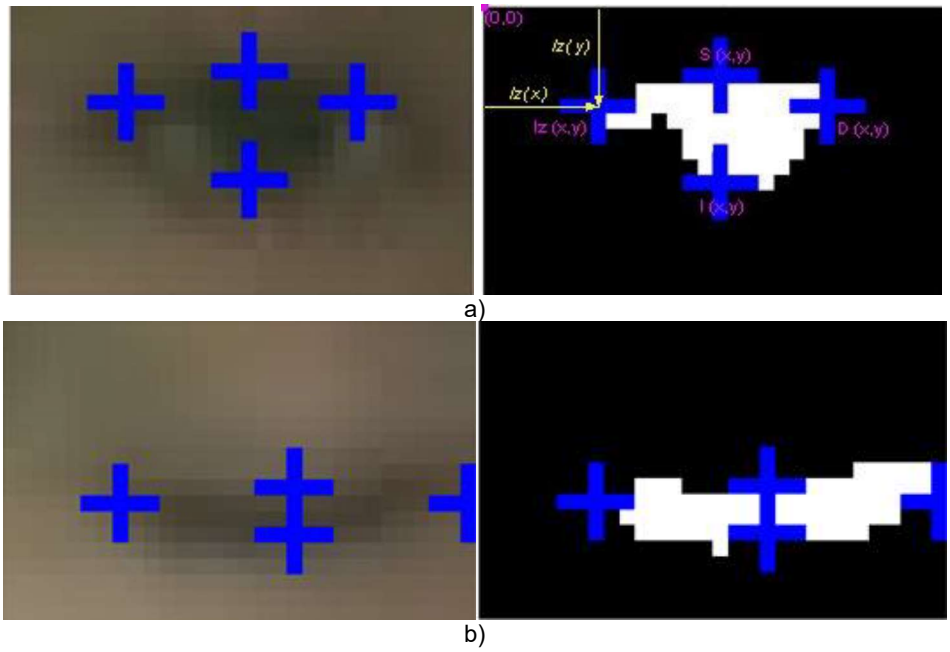


Figura 2.34 Cuatro puntos de referencia en imagen del ojo original e imagen binaria del ojo. a) Puntos de referencia en ojo abierto, b) puntos de referencia en ojo cerrado

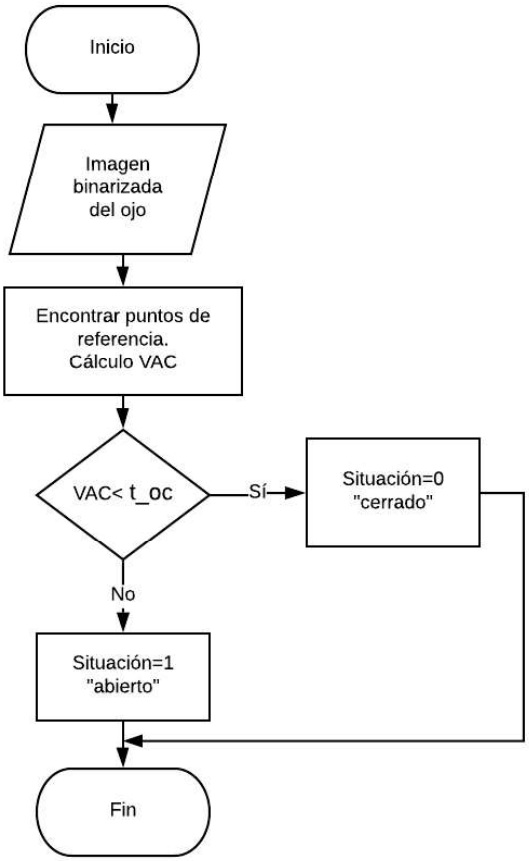


Figura 2.35 Diagrama de flujo de clasificación de ojo abierto y cerrado Método 2
En la imagen binarizada se encuentran los cuatro puntos de referencia, con estos puntos se calcula VAC. VAC permite clasificar si el ojo está abierto o cerrado.

El código completo de la función *marcadores* que sirve para encontrar la ubicación de los 4 puntos de referencia del ojo es el siguiente:

```
function [puntos] = marcadores(Image)
%% Punto Izquierdo (Iz) ----------------------------------------------
izx= find(sum(Image)>0,1); % 1° columna con suma mayor a cero
izy1=find(Image(:,izx)>0,1); % 1° valor diferente a cero en izx
izy2=find(Image(:,izx)>0,1,'last'); % último valor diferente a 0 en
izx
izy=izy1+round(abs(izy1-izy2)/2); % punto medio entre izy1 e izy2
%% Punto Derecho (D) --------------------------------------------------
dx=find(sum(Image)>0,1,'last'); % última columna con suma mayor a cero
dy=izy; % punto medio entre izy1 e izy2
%% Punto Inferior (I) --------------------------------------------------
iy=find(sum(Image(:,6:24),2)>0,1,'last'); % última fila con suma > 0
ix=izx+round(abs(dx-izx)/2); % punto medio entre los puntos izx y dx
%% Punto Superior (S)--------------------------------------------------
sx=ix; % punto medio entre los puntos izx y dx
sy=find(sum(Image(:,sx),2)>0,1); % 1° punto mayor a cero en columna sx
%% vectores posición ---------------------------------------------------
possup1=[supx,supy]; % coordenadas punto S
posinf1=[infx,infy]; % coordenadas punto I
posiz1=[izx,izy]; % coordenadas punto Iz
posder1=[derx,dery]; % coordenadas punto D
puntos=[possup1;posinf1;posiz1;posder1]; % coordenadas de todos los
puntos
```

2.6.2.2. Fase 2: Cálculo de valor de ojo abierto/cerrado (VAC)

Para determinar un límite entre un ojo abierto y uno cerrado se usa la ecuación: $VAC=difv-(difh/(difv))^3$, donde **VAC** representa el valor de ojo abierto o cerrado, ***difv*** es la diferencia absoluta entre los puntos verticales, ***difh*** es la diferencia absoluta entre los puntos horizontales. Se usa un umbral τ_{OC} para clasificar entre un ojo abierto o cerrado. Si VAC es menor que τ_{OC} entonces el ojo está cerrado, caso contrario está abierto. VAC está elevado a la tercera potencia para enfatizar la diferencia entre los estados del ojo abierto o cerrado.

La relación VAC fue obtenida probando diferentes relaciones entre los puntos, obteniendo con esta relación un 99% de precisión en 120 imágenes, las relaciones probadas se encuentran en la sección de Resultados y Discusión.

El valor umbral para la clasificación de ojo (τ_{OC}) se determinó observando el comportamiento de la gráfica de la relación VAC con respecto al estado real del ojo, en esta gráfica se evidenció que cada que el ojo está cerrado el valor VAC es menor -35, por lo tanto τ_{OC}=-35. La gráfica de la relación VAC se puede observar en la Figura 2.36.

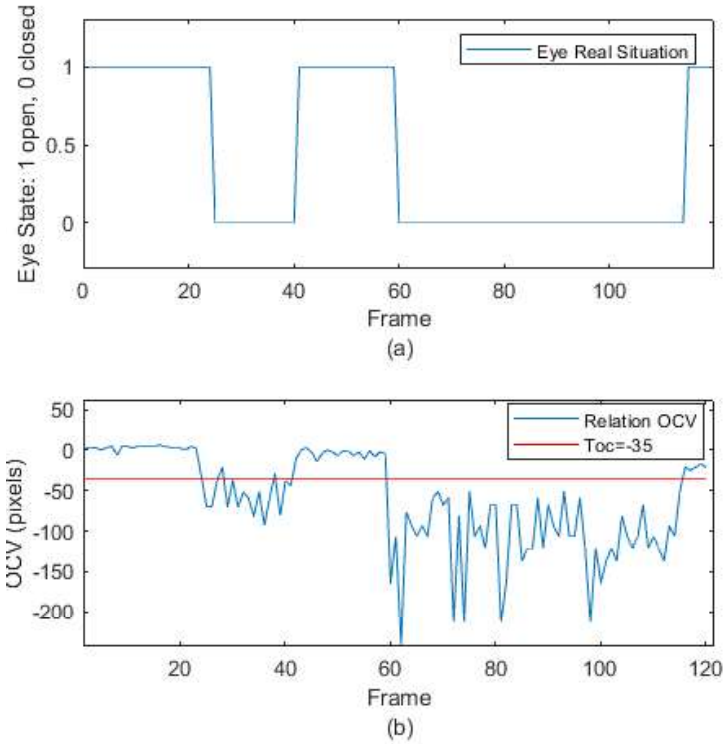


Figura 2.36 Gráfica de situación de ojo abierto/cerrado
a) Se presenta el estado real del ojo, donde 1 indica que el ojo está abierto y 0 cerrado. b) La línea azul es la relación VAC entre los puntos de referencia del ojo, la línea roja el valor umbral τ_{OC}=-35.

El código para encontrar el valor VAC es el siguiente:

```
[puntos]=marcadores(BW); % función de la Fase 1
difh=abs(puntos(3,1)-puntos(4,1)); % difh=izx-derx
difv=abs(puntos(1,2)-puntos(2,2)); % difh=supy-infy
VAC=difv-(difh/(difv+1))^3; % ecuación para hallar VAC
```

2.6.2.3. Fase 3: Clasificación de ojo abierto/cerrado

Para realizar la clasificación de ojo abierto/cerrado se hace uso del valor *VAC* que se obtiene en la Fase 2. Si *VAC* es menor a -35, el ojo será clasificado como cerrado, caso contrario se lo clasificará como abierto. En la clasificación de ojo abierto/cerrado, el estado abierto está representado por el valor 1 y el estado de cerrado por el valor de 0. El código que realiza esta clasificación es el siguiente:

```
if VAC<Toc % Toc=-35
   fprintf ': Cerrado'; % si VAC<Toc el ojo se clasifica como cerrado
   situacion=0; % la situación del ojo es cerrado (0)
else
   fprintf ': Abierto'; % si VAC>Toc el ojo se clasifica como abierto
   situacion=1; % la situación del ojo es abierto (1)
end
```

2.7. ETAPA 5: DETECCIÓN DE SOMNOLENCIA

La tasa de parpadeo normal es de 15 a 20 veces por minuto, con una duración de parpadeo de 0.25 a 0.3 segundos cada uno. Cuando se toma la mayor cantidad de parpadeos y la duración de parpadeo más larga (20 parpadeos por minuto y 0,3 segundos), la máxima cantidad de tiempo que el ojo estaría cerrado es de 6 segundos por minuto, así que bajo un estado de alerta el ojo estará aproximadamente cerrado el 10% del tiempo. De acuerdo con este porcentaje, Gang Li y Wan-Young Chung recomiendan determinar un estado de alerta entre el 0% al 30% y un estado de somnolencia entre el 30% al 40% [17].

Para detectar el estado de somnolencia se usa la ecuación: $D=t_{close}/T$, donde **D** representa la relación de tiempo en la cual el ojo ha estado cerrado, $\mathbf{t_{close}}$ es el tiempo total durante el cual el ojo estuvo cerrado, **T** es el tiempo total. Se usa un umbral $\boldsymbol{\tau_D}$ para determinar somnolencia. Si D es mayor que τ_D la persona está somnolienta, caso contrario no.

Para encontrar el umbral τ_D se escogió usar el valor del 40% (0.4) de acuerdo a [17], este valor evita la incertidumbre de usar el 30% donde bien se puede estar en estado de alerta como en estado de somnolencia. Cuando un ojo permanece cerrado más del 40% del tiempo total en una ventana de 20 cuadros, entonces existe somnolencia, por este motivo el valor umbral es τ_D =0.4 (40%). La Figura 2.37 muestra como la curva de la ecuación D aumenta cuando el ojo está cerrado.

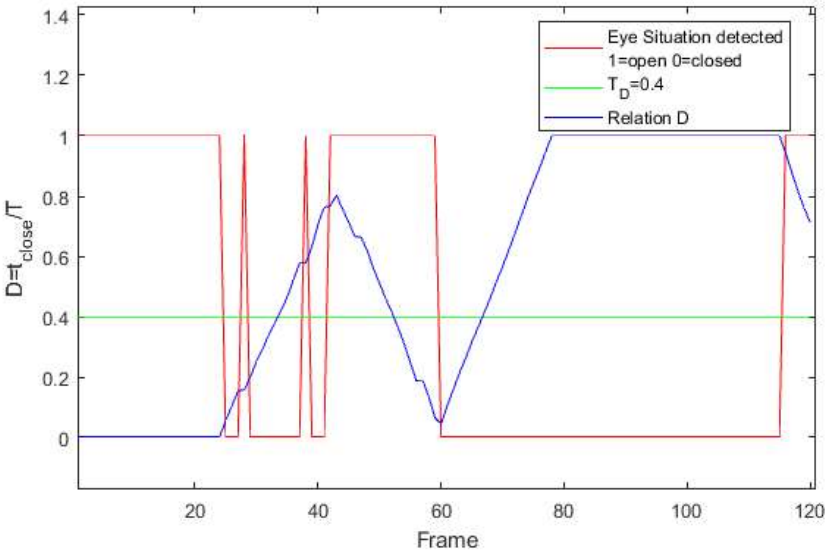


Figura 2.37 La línea roja representa la clasificación del ojo abierto(1)/cerrado(0). La línea verde es el umbral de somnolencia de 0.4(40%). La línea azul es el valor D de somnolencia detectado.

Debido a que las funciones para obtener el tiempo en Matlab están sujetas al procesamiento del equipo en el que se realiza la prueba, se usará como parámetro el número de imágenes en las cuales el ojo ha estado cerrado respecto al número total de imágenes en una ventana de 20 cuadros. La siguiente ecuación muestra la relación usada:

$$D = \frac{t_{close}}{T} = \frac{t_{close} * \frac{1}{fps}}{T * \frac{1}{fps}} = \frac{t_{close} * \frac{\#cuadros}{1seg}}{T * \frac{\#cuadros}{1seg}} = \frac{\#cuadros_{close}}{\#cuadros_{total}}$$

El diagrama de flujo de la Figura 2.38 muestra el proceso para la detección de somnolencia. Primero se evalúa la situación del ojo: abierto o cerrado. Si el ojo está cerrado el contador ojos cerrados (OC) se incrementa en uno. Posteriormente, si el contador OC es mayor al 40% de imágenes totales en la ventana deslizante se emite una alarma. Este proceso se realiza con cada imagen nueva que se adquiera.

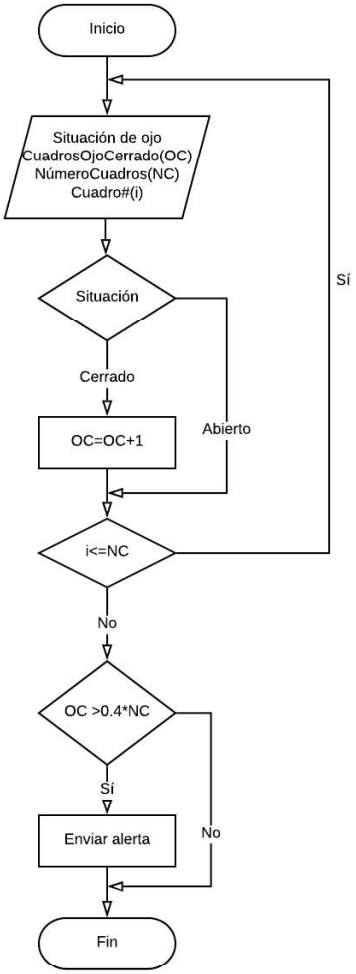


Figura 2.38 Diagrama de flujo de detección de somnolencia
Si el ojo es detectado como cerrado el contador OC aumenta en uno. Si el contador OC es mayor que el umbral se emite una señal audible de alerta.

Para realizar la detección de somnolencia se evalúan 20 cuadros continuos de imágenes. Por este motivo se tiene un buffer de 20 elementos donde en cada iteración se va cambiando un elemento de este buffer. Al completarse el buffer se vuelve a reiniciar el valor del contador para volver a completarlo.

```
ventana=20 % imágenes continuas que se evalúan
if j==ventana % si buffer alcanza el máximo
      j=1; % se reinicia el buffer
end
```

El buffer es un vector de 20x1 (*som*) que permite contar la cantidad de ojos cerrados que han existido en 20 cuadros continuos de imágenes. Si el ojo está cerrado este buffer cambia un elemento por 1 y si se encuentra abierto cambia un valor por 0.

```
% situacion: valor encontrado en Etapa 4 (1: abierto, 0: cerrado)
% j: ubicación actual en el buffer
if situacion==0 % si el ojo está cerrado
      som(j)=1; % se pone 1 en un elemento del buffer
else % caso contrario
      som(j)=0; % se pone 0 en un elemento del buffer
end
```

Para detectar la somnolencia se evalúa si la suma total de los ojos cerrados (*ssom*) corresponde a un valor mayor al 40% del total de elementos (*limite*), en el intervalo de los 20 cuadros consecutivos. Si se supera este valor se emite una alarma visual y auditiva. En la detección de somnolencia, los estados de alerta y somnolencia están representados por el valor 0 y 1 respectivamente en la variable *vsom*.

```
% Creación de alarma auditiva ----------------------------------------
WarnWave = [sin(1:.6:400), sin(1:.7:400), sin(1:.4:400)];
Audio = audioplayer(WarnWave, 22050);
% Detección de somnolencia -------------------------------------------
ssom=sum(som); % suma total de ojos cerrados
limite=0.4*ventana; % cálculo de límite del 40% ()
if ssom > limite % si cantidad de ojos > 0.4*ventana
      disp ' ALERTA' % se emite alarma visual
      play(Audio); % se emite alarma auditiva
      vsom=1 % existe somnolencia(1)
else
      vsom=0 % no existe somnolencia(0)
end
```

El código completo para la detección de somnolencia en Matlab queda de la siguiente manera:

```
ventana=20 % imágenes continuas que se evalúan
h = figure; % creación de objeto que permitirá terminar la ejecución
% j: ubicación actual en el buffer, se inicia en 1
% Creación de alarma auditiva ----------------------------------------
WarnWave = [sin(1:.6:400), sin(1:.7:400), sin(1:.4:400)];
Audio = audioplayer(WarnWave, 22050);
% Inicio de bucle de evaluación de somnolencia
while size(findobj(h))>0 % se repite hasta cerrar h
%% inicializador de ventana deslizante ------------------------------
if j==ventana % si buffer alcanza el máximo
      j=1; % se reinicia el buffer
end
%% contador de ojos cerrados ----------------------------------------
% situacion: valor encontrado en Etapa 4 (1: abierto, 0: cerrado)
if situacion==0 % si el ojo está cerrado
      som(j)=1; % se pone 1 en un elemento del buffer
else % caso contrario
      som(j)=0; % se pone 0 en un elemento del buffer
end
% Detección de somnolencia -----------------------------------------
ssom=sum(som); % suma total de ojos cerrados
limite=0.4*ventana; % cálculo de límite del 40% ()
if ssom > limite % si cantidad de ojos > 0.4*ventana
      disp ' ALERTA' % se emite alarma visual
      play(Audio); % se emite alarma auditiva
      vsom=1 % existe somnolencia(1)
else
      vsom=0 % no existe somnolencia(0)
end
      j=j+1; % contador de ubicación en el buffer
end % fin de bucle
```

3. RESULTADOS Y DISCUSIÓN

Con el fin de comprobar el funcionamiento del algoritmo propuesto en este trabajo, se han realizado pruebas para cada una de las etapas del algoritmo. Para las etapas de Obtención del ojo y clasificación de ojo abierto/cerrado se realizaron pruebas con una base de datos adquirida con la cámara infrarroja, para la etapa de detección de somnolencia se realizó pruebas en dos personas obteniendo un video cuando la persona estaba en estado de alerta y uno cuando estaba en estado de somnolencia.

Como prueba adicional se realizó la detección de somnolencia en una persona usando gafas transparentes.

Las características de la base de datos en las cuales se realizaron las pruebas son las siguientes:

Base de datos de imágenes

- Rostros de personas ecuatorianas
- Posee 300 imágenes por cada una de 8 personas, en total posee 2400 imágenes.
- 5 son mujeres y 3 hombres
- El rango de edades es entre 24 a 45 años
- Las imágenes fueron adquiridas con una cámara infrarroja

Base de datos de videos

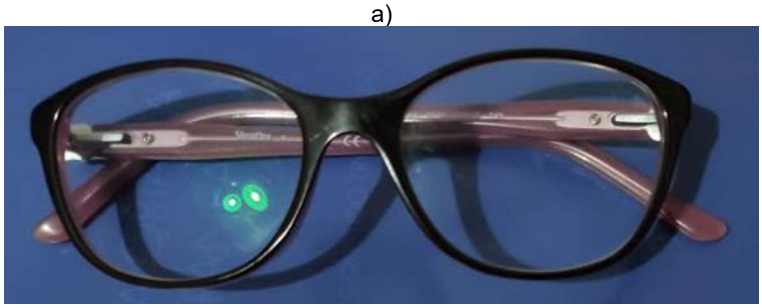
- 2 videos por cada una de 2 personas, total 4 videos.
- 1 video corresponde en estado de alerta y el otro en estado de somnolencia. Se llegó al estado de somnolencia reduciendo las horas de sueño del día anterior y aplazando la hora de dormir.
- La duración de cada video es de 100 segundos
- Los videos fueron adquiridos con una cámara infrarroja.

Base de datos de videos en persona con gafas transparentes

- 1 video en estado de alerta presentando reflejos en las gafas.
- 1 video en estado de alerta sin reflejos en las gafas.
- 1 video en estado de somnolencia sin reflejos en las gafas.
- 1 video en estado de alerta usando gafas con un marco más grueso.
- Las gafas transparentes para los 3 primeros videos poseen un marco de alrededor de 1 mm, y las gafas transparentes para el cuarto video poseen un marco de alrededor de 3 mm (Figura 3.1).

- La duración de cada video es de 100 segundos.
- Los videos fueron adquiridos con una cámara infrarroja.

a)



b)

Figura 3.1 a) Lentes transparentes con marco de 1mm. b) Lentes transparentes con un marco de 3mm

3.1.1. ETAPA 1: ADQUISICIÓN DE IMÁGENES

Para seleccionar la función adecuada se han efectuado varias pruebas con cada una de las funciones de video, de adquisición de imagen y de presentación (Sección 1.4.2.1). En la Tabla 3.1 se muestran los resultados obtenidos con cada función de adquisición y presentación de imágenes. El único parámetro tomado en cuenta son los cuadros por segundo obtenidos con cada una de las funciones ya que el algoritmo debe trabajar en tiempo real por lo que el tiempo de adquisición de imágenes es importante.

El propuesto usa las funciones que adquieren mayor cantidad de cuadros por segundo (FPS). Estas funciones son las siguientes: *webcam*, *snapshot* y actualización de ventana *imshow* con set *CData*. Con estas funciones se llega a obtener cerca de 29 FPS. Hay que tomar en cuenta que estas tasas de adquisición solamente representan la adquisición y visualización de las imágenes, mas no está considerado el tiempo que le toma al algoritmo realizar la detección de somnolencia.

Tabla 3.1 Cuadros por segundo para funciones de adquisición y presentación de imágenes.

Función de Video	Función de Adquisición	Función de Presentación	Imágenes adquiridas	Tiempo [s]	FPS
videoinput	getsnapshot	Image	500	57,83	8,76
		Imshow	500	124,44	4,03
		Imagesc	500	57,79	8,66
		Image, set Cdata	500	62,23	8,04
		Imagesc, set Cdata	500	58,12	8,62
		Imshow, set Cdata	500	58,61	8,56
		Vision player	500	58,12	8,63
videoDevice	step	Image	500	29,87	16,98
		Imagesc	500	26,63	18,79
		Imshow	500	84,47	5,92
		Image, set Cdata	500	21,09	23,92
		Imagesc, set Cdata	500	20,91	24,07
		Imshow, set Cdata	500	20,77	24,23
		Vision player	500	19,60	25,76
webcam	snapshot	Image	500	30,46	16,46
		Imagesc	500	30,28	16,61
		Image, set Cdata	500	19,59	25,73
		Imagesc, set Cdata	500	18,93	26,69
		Imshow, set Cdata	500	17,37	28,87
		Vision player	500	22,55	22,20

3.1.2. ETAPA 2: OBTENCIÓN DE LA SECCIÓN DEL OJO

La Figura 3.2 muestra el resultado al recortar el ojo proporcionalmente a las dimensiones del rostro.

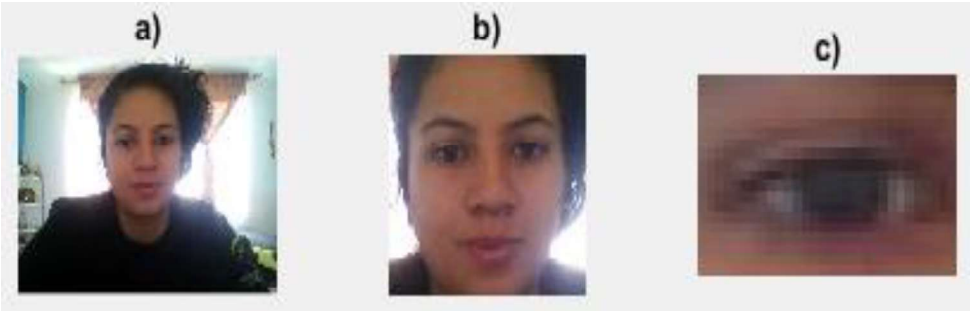


Figura 3.2 Resultado de la Obtención de la sección de ojo
a) Imagen original, b) detección de rostro y c) recorte proporcional de la sección del ojo

3.1.3. ETAPAS 3 Y 4: PROCESAMIENTO DE IMÁGENES Y CLASIFICACIÓN OJO ABIERTO/CERRADO

3.1.3.1. Método 1: Red MLP

La Tabla 3.2 presenta los porcentajes totales de clasificación correcta de la matriz de confusión para redes neuronales con 10, 15, 20, 25, 30 y 50 neuronas. También presenta pruebas hechas con distintos métodos de procesamiento de imágenes como entrada para la red neuronal.

Tabla 3.2 Porcentajes totales de clasificación de matriz de confusión

	Entrada a Red Nuronal		**Neuronas**						
	Procesamiento de imagen	**Transformada**	**10**	**15**	**20**	**25**	**30**	**50**	**Máx.**
Clasificación correcta [%]	-	-	99,60	99,53	99,74	99,70	99,45	99,74	**99,74**
		DC	99,67	99,76	99,64	99,74	99,63	99,67	**99,76**
		FFT magnitud	99,67	99,73	99,73	99,71	99,64	99,73	**99,73**
		FFT fase	97,69	96,33	96,66	96,58	97,08	96,93	**97,69**
	Ecualización	-	99,78	99,77	99,77	99,76	99,77	99,77	**99,78**
		DC	99,63	99,60	99,60	99,63	99,71	99,64	**99,71**
		FFT magnitud	99,74	99,70	99,73	99,74	99,77	99,70	**99,77**
		FFT fase	96,50	96,70	96,59	97,65	96,97	97,07	**97,65**
		Máximo	**99,78**	**99,77**	**99,77**	**99,76**	**99,77**	**99,77**	

En todas las pruebas se obtiene mayor porcentaje de clasificación correcta usando imágenes ecualizadas, por este motivo se decide usar este procesamiento de imagen. También se realizó una prueba más extensa con la cantidad de neuronas para entrenar la red neuronal, los resultados se pueden observar en el ANEXO I.

Para evaluar la clasificación de ojo abierto/cerrado usando la red MLP, se probaron dos diferentes condiciones etiquetadas: NET1 y NET2. NET1 se refiere al entrenamiento de una red por cada uno de los sujetos de prueba, es decir que se eligió el conjunto de entrenamiento de un sujeto y se probó la clasificación sobre el mismo sujeto. Hay que tomar en cuenta que para evitar contaminación de datos, las imágenes de entrenamiento y de prueba no son las mismas. En NET2, se usó la técnica *leave-one-out*, en donde se han usado las imágenes de 7 sujetos para el entrenamiento y se ha realizado la prueba en el 8° sujeto. La Tabla 3.3 muestra los resultados de clasificación usando NET1 y la Tabla 3.4 muestra los resultados usando NET2.

Los falsos positivos son imágenes de ojos abiertos que fueron clasificadas como cerrados y los falsos negativos son imágenes de ojos cerrados que fueron clasificadas como abiertas.

Tabla 3.3 Resultados de clasificación usando el Método 1 con NET1

Sujeto	Falsos Positivos [%]	Falsos Negativos [%]	Precisión [%]	Error [%]
1	0,00%	4,70%	95,30%	4,70%
2	2,20%	0,00%	97,80%	2,20%
3	0,00%	0,00%	100,00%	0,00%
4	0,00%	0,00%	100,00%	0,00%
5	2,20%	2,20%	95,60%	4,40%
6	2,30%	6,80%	90,90%	9,10%
7	0,00%	2,20%	97,80%	2,20%
8	3,40%	0,00%	96,60%	3,40%
Promedio			97,75%	3,25%
Desviación Estándar			2.95%	2,95%

Tabla 3.4 Resultados de clasificación usando el Método 1 con NET2

Sujeto	Falsos Positivos [%]	Falsos Negativos [%]	Precisión [%]	Error [%]
1	2,77%	56,06%	41,18%	58,82%
2	17,00%	30,00%	53,00%	47,00%
3	24,08%	2,68%	73,24%	26,76%
4	0,00%	19,93%	80,07%	19,93%
5	24,08%	2,68%	73,24%	26,76%
6	5,74%	23,31%	70,95%	29,05%
7	14,43%	0,34%	85,23%	14,77%
8	3,65%	1,30%	95,05%	4,95%
Promedio			71,50%	28,5%
Desviación Estándar			17,26%	17,26%

Se evidencia que la mayor precisión se da con la red entrenada para cada sujeto con sus propias imágenes. Por este motivo para la prueba de detección de somnolencia se hará usando NET1.

3.1.3.2. Método 2: Puntos de Referencia

La Tabla 3.5 muestra las diferentes relaciones tomadas en cuenta para realizar la clasificación de ojo abierto/cerrado. La primera relación es la que presenta mayor precisión (99%) al realizar la clasificación de ojo abierto/cerrado por lo cual es la elegida para ser usada en el algoritmo.

Tabla 3.5 Resultados de clasificación por relación

	Relación	Total imágenes	Imágenes erradas	Precisión [%]	Error [%]
1	difv-(difh/difv)^3	120	1	0,83%	99,17%
2	difv	120	2	1,67%	98,33%
3	difv/difh	120	2	1,67%	98,33%
4	difv-(difh/difv)	120	2	1,67%	98,33%
5	difh-(difv/difh)	120	2	1,67%	98,33%
6	difh	120	3	2,50%	97,50%
7	difh+(difv/difh)	120	3	2,50%	97,50%
8	Área total de imagen	120	14	11,67%	88,33%

La Tabla 3.6 muestra los resultados obtenidos para la clasificación de ojo para los 8 sujetos. El menor error es 2% y el mayor es 31%.

Tabla 3.6 Resultados de clasificación usando el Método 2

Sujeto	Total imágenes	Falsos positivos	Falsos negativos	Precisión [%]	Error [%]
1	100	1	1	98%	2%
2	100	8	4	88%	12%
3	100	1	1	98%	2%
4	100	4	0	96%	4%
5	100	14	9	78%	22%
6	100	31	0	69%	31%
7	100	15	0	85%	15%
8	100	39	0	61%	39%
Promedio				84%	16%
Desviación Estándar				14%	14%

3.1.4. ETAPA 5: DETECCIÓN DE SOMNOLENCIA

Para ambos métodos se estimó la carga computacional de ambos métodos (red MLP y puntos de referencia). El método 1 realiza alrededor de 5732 operaciones: 5710 entre la capa de entrada y la capa oculta, y 22 operaciones entre las salidas y la capa oculta. Mientras que el método 2 realiza alrededor de 613 operaciones: 570 para binarización, 39 para encontrar los puntos de referencia y 4 para la clasificación.

El método 1 al usar una red MLP es un algoritmo más complejo por el número de operaciones que realiza, esto repercute en la tasa de adquisición final del algoritmo ya que primero se debe procesar la imagen adquirida para proceder a adquirir esta imagen. Las tasas de adquisición incluyendo la detección de somnolencia se presentan en la Tabla 3.7. La tasa de adquisición también según lo que se visualice en pantalla ya que el algoritmo además de emitir una alarma al detectar somnolencia permite visualizar la imagen original, el ojo detectado y las gráficas de somnolencia. Por este motivo se ha seleccionado usar el Método 2 para la implementación en tiempo real.

Tabla 3.7 Resultados de tasas de adquisición del algoritmo de detección de somnolencia

	Número de imágenes	**Tiempo [s]**	**FPS**	**Actividades**
Método 1	265	54,89	5	• Detección de somnolencia • Visualización de imagen original y ojo • Visualización de gráficas de somnolencia y clasificación de ojo
	507	53,23	10	• Detección de somnolencia • Visualización de imagen original y ojo
	332	29,8	11	• Detección de somnolencia
Método 2	218	39,22	6	• Detección de somnolencia • Visualización de imagen original y ojo • Visualización de gráficas de somnolencia y clasificación de ojo
	555	43,3	13	• Detección de somnolencia • Visualización de imagen original y ojo
	468	29,46	16	• Detección de somnolencia

3.1.4.1. Pruebas piloto en dos sujetos

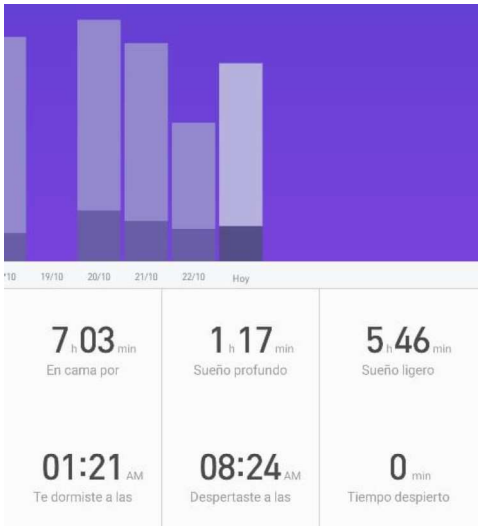
Para evaluar el funcionamiento del algoritmo, se realizaron dos pruebas piloto en dos personas usando el Método 2, la primera prueba se realizó cuando la persona estaba en un estado de alerta y la segunda cuando la persona estaba somnolienta. Al estado de somnolencia se llegó disminuyendo las horas de sueño del día anterior a la prueba y aplazando la hora en la que usualmente duermen los sujetos; la Figura 3.3 y Figura 3.4 muestran el monitoreo del sueño de los sujetos de prueba obtenidos usando una Pulsera Inteligente (Mi Band 4) y una aplicación móvil (Mi Fit).

a) Horas de sueño del día anterior a la prueba

b) Hora a la que se durmió el sujeto el día de la prueba

Figura 3.3 Monitoreo del sueño del sujeto 1 del día anterior a la prueba y del día de la prueba

a) Horas de sueño del día anterior a la prueba



b) Hora a la que se durmió el sujeto el día de la prueba

Figura 3.4 Monitoreo del sueño del Sujeto 2 del día anterior a la prueba y del día de la prueba

El Sujeto 1 es una mujer de 25 años, mientras que el sujeto 2 es un hombre de 24 años. A ambos sujetos antes de realizar la prueba de somnolencia se les pidió llenar un formulario, la Tabla 3.8 muestra las respuestas de ambos sujetos.

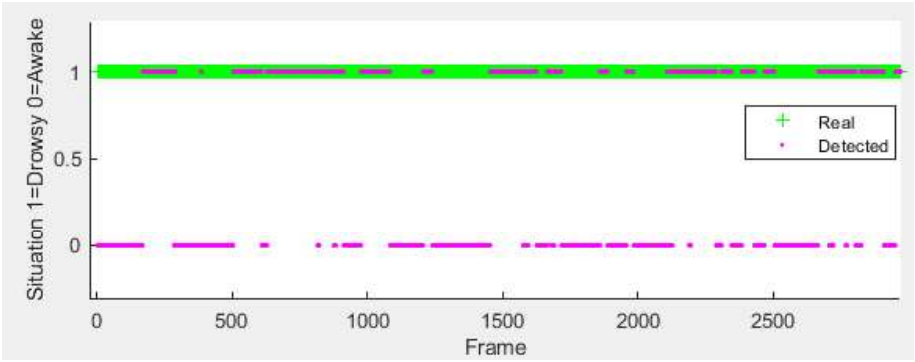
Tabla 3.8 Respuestas a encuesta realizada antes de la prueba de somnolencia

		Respuesta	
	Pregunta	**Sujeto 1**	**Sujeto 2**
Información del instante	Auto evaluación de somnolencia actual en escala del 0 al 10 (0 nada de somnolencia y 10 mucha somnolencia)	10	8
	Tiempo que podría seguir despierto en minutos	15	40
Información cotidiana	Hora a la que usualmente duerme	11pm	11pm
	Total de horas que usualmente duerme por la noche	8	8
	Horas de sueño que debe tener para mantenerse sin somnolencia en el día	8	8

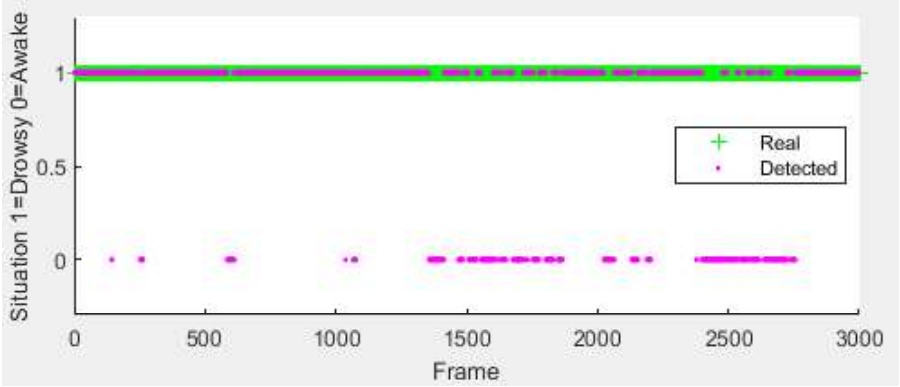
Observando los datos de la Tabla 3.8, de la Figura 3.3 y Figura 3.4 se identifica que para ambos sujetos las horas de sueño habituales se redujeron de 8 horas a 4:16 horas para el Sujeto 1 y de 8 horas a 4:56 horas para el Sujeto 2. Respecto a la hora habitual en que duermen para el Sujeto 1 cambió de 11pm a 2:34am y para el Sujeto 2 cambió de 11pm a 1:21am.

Al evaluar el algoritmo cuando los sujetos estaban en estado de somnolencia, la gráfica de resultado de somnolencia indicó que si hay somnolencia para ambos sujetos y la alarma fue emitida. En la Figura 3.5 se puede observar de color verde el estado real de la persona (somnolienta) y en color rosado las veces que la alarma fue emitida. A pesar de que la persona se encuentra en un estado de somnolencia, la alarma no es emitida todo el tiempo puesto que se evidenció que el comportamiento de los sujetos luego de recibir la alarma trata de volver a un estado de alerta.

También se realizó pruebas sobre los mismos sujetos, pero en un día en el que no presentaban síntomas de somnolencia. Cuando una persona se encuentra alerta, la detección de somnolencia es 0 como muestra la Figura 3.6, por lo cual no se emite la alarma.

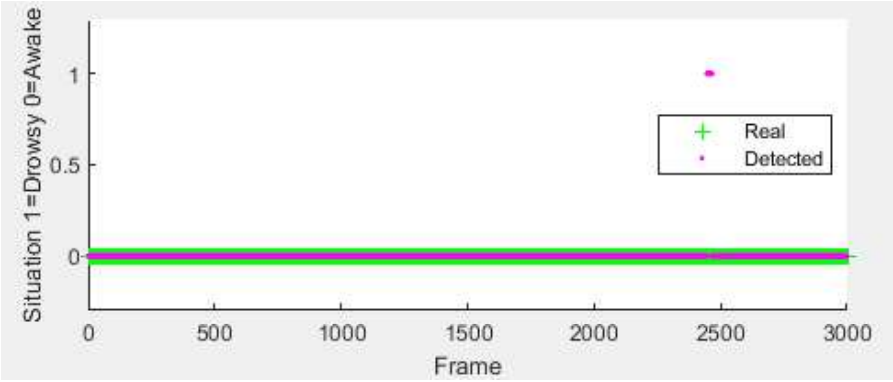


a)

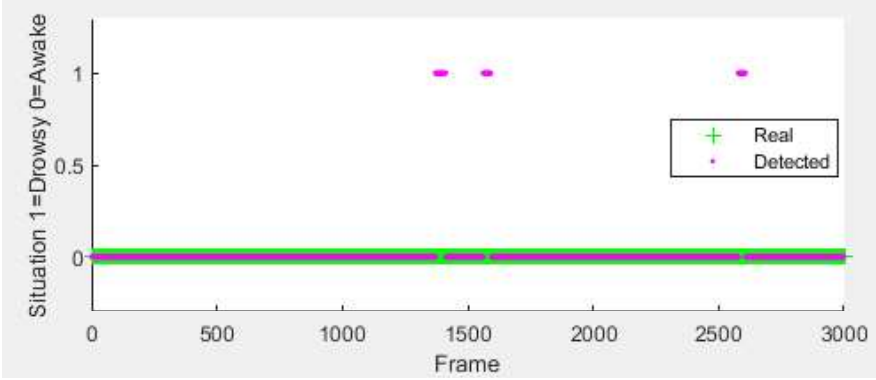


b)

Figura 3.5 Detección de somnolencia en estado de somnolencia usando el Método 2 a) Sujeto 1 b) Sujeto 2



a)



b)

Figura 3.6 Detección de somnolencia en estado de alerta usando el Método 2 a) Sujeto 1 b) Sujeto 2

En la Tabla 3.9 se tabulan los tiempos durante los cuales se emitió la alarma para ambos sujetos en estado de alerta y de somnolencia. En estado de alerta la alarma sonó menos de 4 segundos, mientras que en estado de somnolencia la alarma sonó más de 40 segundos.

Tabla 3.9 Tiempos en el cual sonó la alarma de somnolencia durante la detección

				Tiempo en el que se emite la alarma		**Tiempo en el que no se emite la alarma**	
Estado del Sujeto	**Sujeto**	**Total Imágenes**	**Tiempo total [s]**	**[s]**	**[%]**	**[s]**	**[%]**
Alerta	1	3000	100	0,63	0,63%	99,37	99,37%
	2	3000	100	3,30	3,30%	96,70	96,70%
Somnolencia	1	3000	100	47,53	47,53%	52,47	52,47%
	2	3000	100	77,33	77,33%	22,67	22,67%

3.1.4.2. Pruebas en un sujeto habitación sin iluminación

Para comprobar el funcionamiento del algoritmo en ambientes de poca iluminación, se realizó una prueba con el Sujeto 1 en estado de somnolencia. La Figura 3.7 muestra un comportamiento similar a la Figura 3.5, donde existe somnolencia, aunque la prueba se haya realizado en una habitación sin iluminación. La Figura 3.8 muestra una de las imágenes adquiridas para esta prueba.

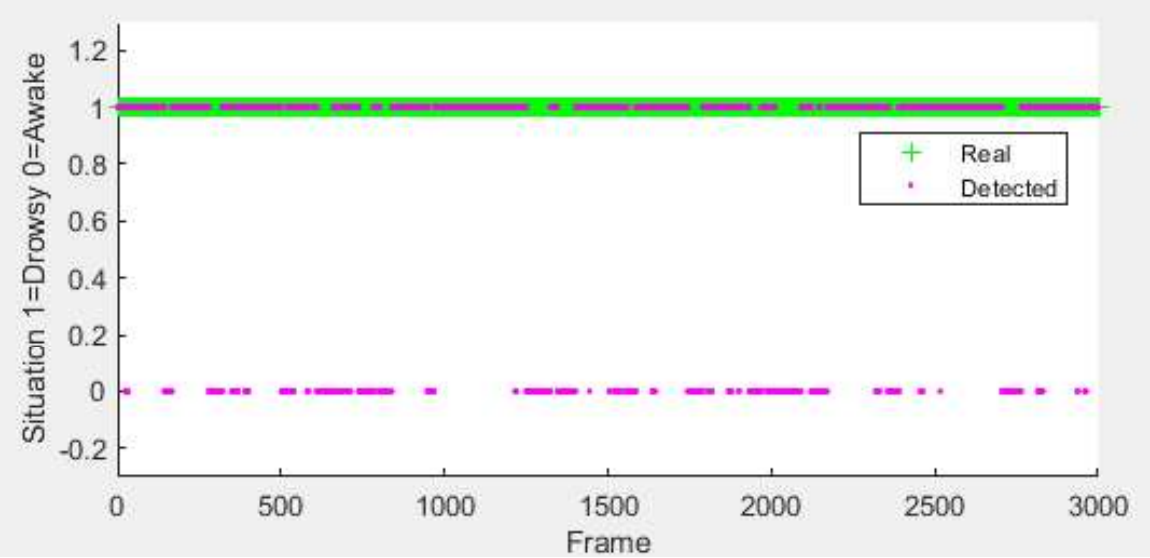


Figura 3.7 Detección de somnolencia en estado de somnolencia usando el Método 2 en una habitación sin iluminación.

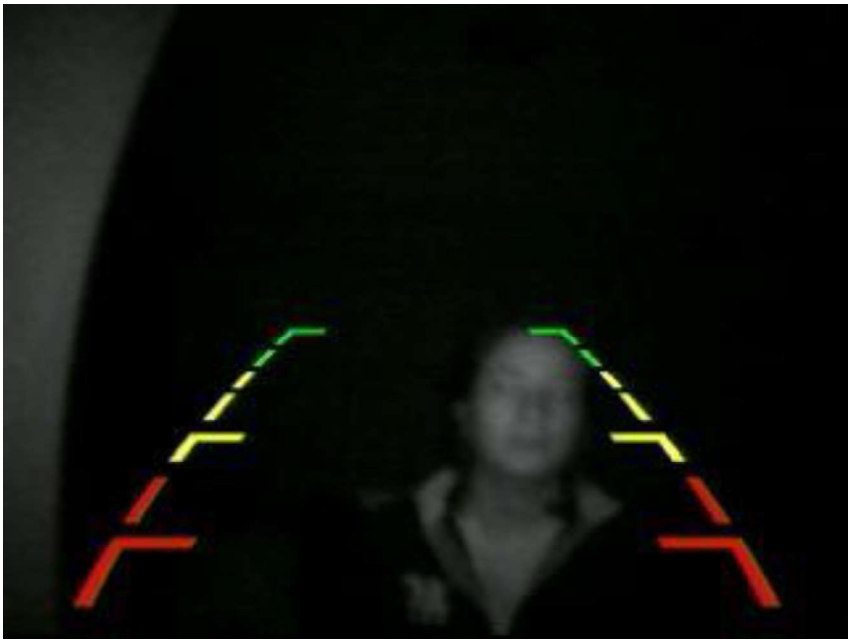


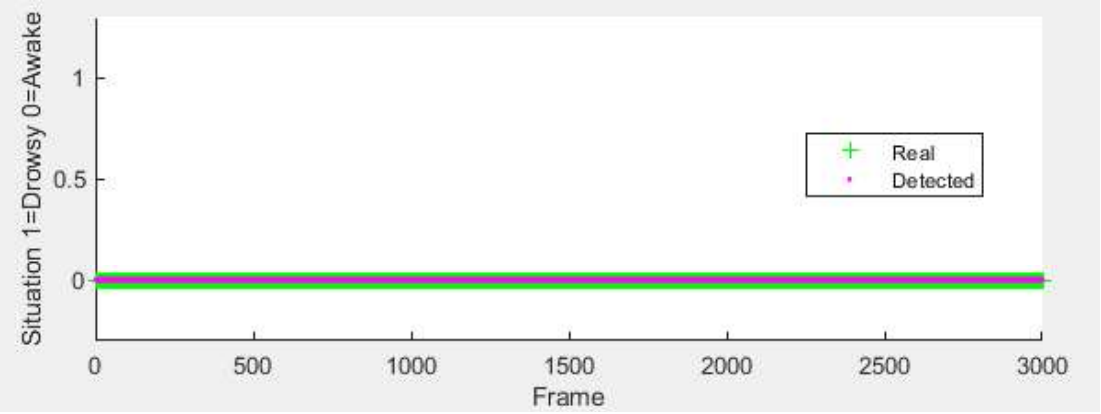
Figura 3.8 Imagen adquirida del Sujeto 1 en habitación sin iluminación

3.1.4.3. Pruebas en un sujeto usando gafas transparentes

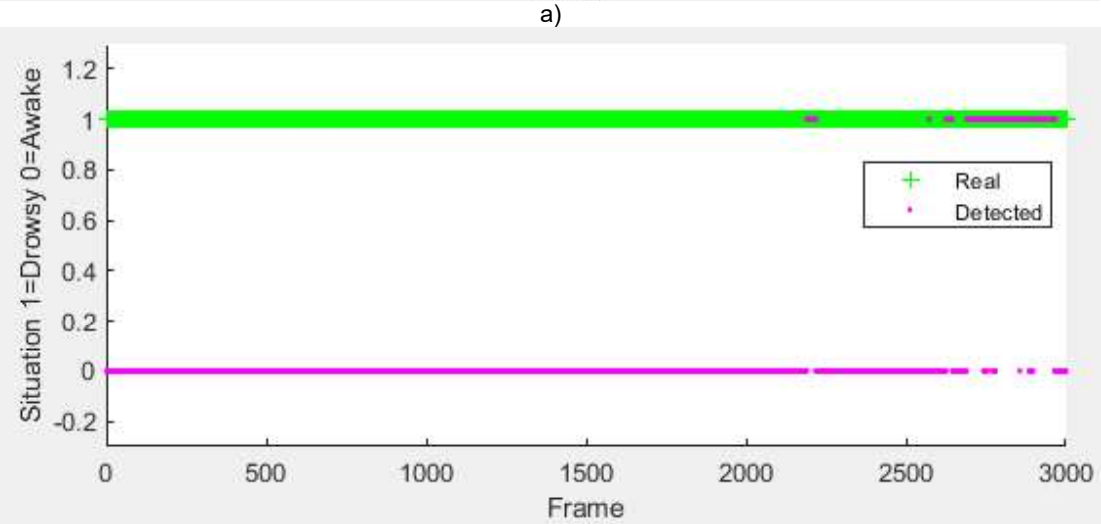
Se realizó pruebas de detección de somnolencia en una persona usando lentes transparentes. La Figura 3.10 presenta la detección de somnolencia para un estado de alerta y somnolencia en el Sujeto 1 usando lentes transparentes con marco de 1mm; hay que tomar en cuenta que en este video no existen reflejos en la sección de los lentes como muestra la Figura 3.9.

A pesar de que en estado de alerta no se emite la alarma de advertencia, al estar en estado de somnolencia no existe emisión de la alarma por más del 75%; esto evidencia que el algoritmo no está adaptado a funcionar con gafas transparentes, aunque no existan reflejos en los lentes.

Figura 3.9 Sujeto 1 usando lentes transparentes, en los cuales no se presenta reflejos. La ubicación de la cámara está a 45° bajo la vista del sujeto.



a)

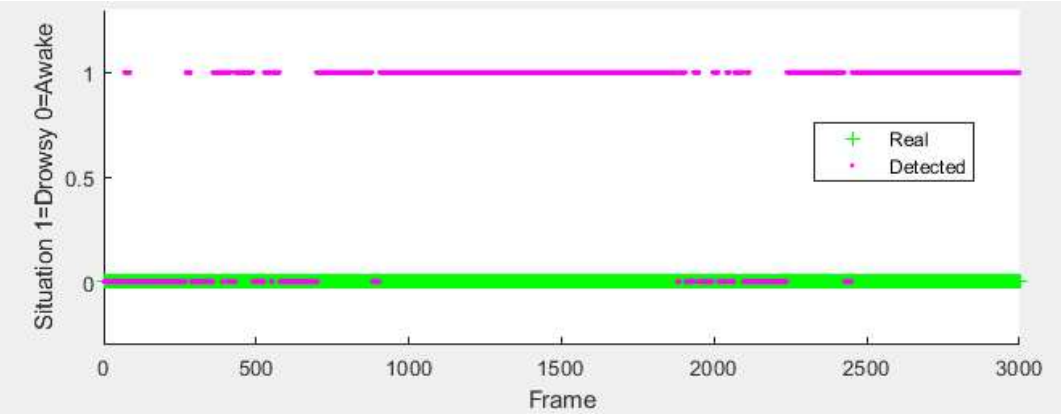


b)

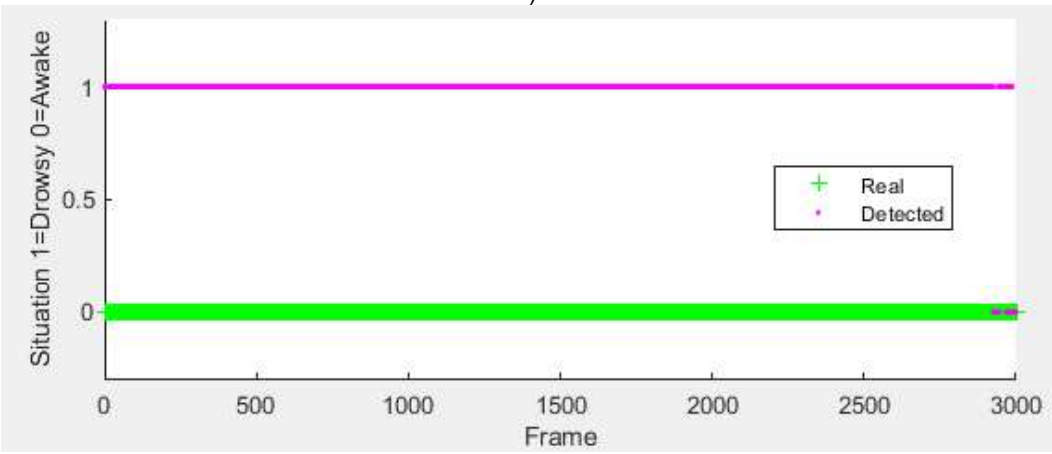
Figura 3.10 Detección de somnolencia en sujeto 1 usando lentes transparentes a) en estado alerta y b) en estado de somnolencia

En la Figura 3.12 se muestra los resultados de la detección de somnolencia en estado alerta usando dos gafas transparentes distintas. Para las gafas de 1mm se realiza la detección de rostro, pero al presentar reflejos en los lentes la detección de somnolencia no es la correcta, mientras que para las gafas de 3mm no se realiza la detección de rostro por lo cual la detección de somnolencia no es correcta. En la Figura 3.11 se muestra una imagen del rostro del sujeto 1 usando gafas transparentes y con reflejos en los lentes.

Figura 3.11 Sujeto 1 usando lentes transparentes, en los cuales se presenta reflejos. La ubicación de la cámara está a 45° bajo la vista del sujeto.



a)



b)

Figura 3.12 Detección de somnolencia en estado de alerta
a) Sujeto 1 usando gafas transparentes con marco de 1mm y con reflejos en el lente. b) Sujeto 2 usando gafas transparentes con marco de 3mm

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

- Se desarrollaron dos algoritmos para la detección de somnolencia humana, en el método 1 se usa una red neuronal MLP y en el método 2 se usa la morfología del ojo.
- Existen varias investigaciones para detectar la somnolencia humana, en la mayoría de estas investigaciones usan el índice PERCLOS. En el algoritmo desarrollado no se hace uso de este índice ya que este requiere tasas de adquisición de imágenes muy altas, por lo cual no se podría usar en dispositivos de bajo costo.
- La tasa de adquisición y procesamiento de imágenes difiere para cada método. Para el método 1 esta tasa es de 11 FPS, mientras que para el método 2 se tiene una tasa de 16 FPS; por lo que el uso del método 1 o método 2 dependerá de los recursos para realizar la detección de somnolencia.
- El algoritmo para la clasificación de ojo abierto/cerrado presenta una precisión del 97% para el método 1 y el 84% para el método 2. Esta precisión de clasificación de ojo es un indicador indirecto para evaluar la detección de somnolencia.
- El método 2 de clasificación puede ser usado para la detección de somnolencia, aunque posea un porcentaje de clasificación correcta menor respecto al otro método. Esto se debe a que el algoritmo propuesto no requiere de una clasificación perfecta por el uso de la ventana de 20 cuadros consecutivos, así que pocos errores cada 20 cuadros no afectará los resultados. El resultado de detección de somnolencia será afectado si hay ráfagas de errores.
- El método 2 de clasificación, es computacionalmente menos intensivo comparado con el primer método. Lo cual lo hace útil para ser usado con dispositivos poco sofisticados y de bajo costo.
- En estado de somnolencia la alarma suena más 40% del tiempo en el cual se está realizando la detección. Esto se debe a que la otra cantidad del tiempo el sujeto que se está sometiendo a la prueba intenta volver a un estado de alerta que le dura unos pocos segundos.
- En estado de alerta la alarma suena menos del 3% del tiempo, esto permite que no sea molesto para una persona que se encuentra alerta.
- El algoritmo funciona en ambientes con poca iluminación debido a que se usa una cámara infrarroja, la cual permite detectar el rostro de personas que se sometan a la detección.

- El algoritmo no funciona en personas usando gafas transparentes ya que los resultados varías respecto al grosor del marco de los lentes y a la existencia de reflejos de luz en los lentes.

4.2. RECOMENDACIONES

- Para un trabajo futuro, se recomienda probar el algoritmo en un grupo de personas más grande con una base de datos de imágenes tomadas bajo diferentes condiciones de luz también más grande. Además, las pruebas se las debería realizar en conjunto con un método invasivo para determinar si la persona está realmente alerta, somnolienta o si su comportamiento de parpadeo cambia al saber que está siendo observada.
- Se recomienda usar el método 1 de clasificación (usando una red MLP) cuando se posee dispositivos con altas prestaciones y sean capaces de realizar todo el procesamiento sin afectar los resultados en tiempo real. También se lo recomienda si no se requiere hacer la detección de somnolencia en tiempo real.
- Si se requiere un detector de somnolencia para un campo donde la persona tenga que rotar su rostro se debe usar un detector diferente al de Viola Jones, o modificar el algoritmo para que funcione con rostros que han rotado.

REFERENCIAS BIBLIOGRÁFICAS

[1] ANT, *Reporte Nacional de Siniestros Diciembre 2018,* Ecuador: Dirección de Estudios y Proyectos, 2018.

[2] Dirección General de Tráfico, *Otros Factores de Riesgo: El Sueño,* España, 2014.

[3] L. Montoro, J. Soler y F. Tortosa, «Psicología y Tráfico,» *Revista Tráfico,* vol. 35, 1988.

[4] I. Daza, L. Bergasa, S. Bronte, J. Yebes, J. Almazán y R. Arroyo, «Fusion of Optimized Indicators from Advanced Driver Assistance Systems (ADAS) for Driver Drowsiness Detection,» *Sensors,* 2014.

[5] Foundation for Medical Education & Research, «Electrodos de electroencefalografía,» Mayo. [En línea]. Available: https://www.mayoclinic.org/es-es/tests-procedures/eeg/multimedia/eeg-electrodes/img-20005916. [Último acceso: Enero 2019].

[6] O. Otero, G. Fuentes y F. López, «Diseño e Implementación de un Sistema de Control de Movimientos para una plataforma Móvil usando ElectroOculografía,» *Euro-American Association on Telematics and Information Systems,* 2010.

[7] Q. Ji y X. Yang, «Real-Time Eye, Gaze, and Face Pose Tracking for Monitoring Driver Vigilance,» *Real-Time Imaging,* vol. 8, 2002.

[8] A. Sahayadhas, K. Sundaraj y M. Murugappan, «Detecting Driver Drowsiness Based on Sensors: A Review,» *Sensors,* vol. 12, 2012.

[9] M. Cai, Y. Gu, H. Sun, J. Qi y B. Wang, «Driver Fatigue Detection System Based on DM3730,» *The Open Automation and Control Systems Journal,* vol. 7, 2015.

[10] R. Hammoud, Passive Eye Monitoring: Algorithms, Applications and Experiments, Springer, 2008, p. 308.

[11] B. Akrout y W. Mahdi, «A visual based approach for drowsiness detection,» *IEEE Intelligent Vehicles Symposium,* vol. 4, pp. 1324-1329, 2013.

[12] U. Trutschel, B. Sirois, D. Sommer, M. Golz y D. Edwards, «PERCLOS: An Alertness Measure of the Past,» *PROCEEDINGS of the Sixth International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design ,* pp. 172-179, 2017.

[13] P. Alegre, «Kaufer,» Clínica de Ojos, 2019. [En línea]. Available: https://www.kaufer.com/es/especialidades/cornea. [Último acceso: 17 Septiembre 2019].

[14] Y. Noticias, «Yahoo Noticias,» 30 Mayo 2013. [En línea]. Available: https://es.noticias.yahoo.com/fotos/es-la-%C3%BAnica-persona-de-su-pueblo-con-ojos-azules-1369912254-slideshow/. [Último acceso: 11 Septiembre 2019].

[15] M. Millodot, Dictionary of Optometry and Visual Science E-Book, ELSEVIER, 2009.

[16] L. ZiLin, G. Tan, Y. Pang, Y. TANG y K. Qian, «Driving Fatigue Detection based on Blink Frequency and Eyes Movement,» *SAE Technical Paper,* 2017.

[17] G. Li y W. Chung, «Detection of Driver Drowsiness Using Wavelet Analysis of Heart,» *Sensors,* vol. 13, 2013.

[18] Volvo Car USA, «Driver Alert Control,» Marzo 2017. [En línea]. Available: https://www.youtube.com/watch?v=sVDTnFeutOs. [Último acceso: Enero 2019].

[19] Mercedes-Benz of Temecula, «What is Mercedes-Benz ATTENTION ASSIST?,» Noviembre 2018. [En línea]. Available: https://www.mbtemecula.com/what-is-mercedes-benz-attention-assist/. [Último acceso: Enero 2019].

[20] T. H. Ford, «What are the Ford Driver Alert and Lane Keeping Aids?,» [En línea]. Available: https://www.tch.co.uk/about/why-choose-ford/lane-keeping-aid/. [Último acceso: Enero 2019].

[21] P. Bejerano, «Esta nueva tecnología detecta el nivel de somnolencia en conductores,» Blogthinkbig, Agosto 2017. [En línea]. Available: https://blogthinkbig.com/esta-nueva-tecnologia-detecta-el-nivel-de-somnolencia-en-conductores. [Último acceso: Enero 2019].

[22] Bosch, « Driver drowsiness detection,» [En línea]. Available: https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driver-assistance-systems/driver-drowsiness-detection/. [Último acceso: Enero 2019].

[23] B. Thorslund, Electrooculogram Analysis and Development of aSystem for Defining Stages of Drowsiness, Linköping: Linköping University, Dept. BiomedicalEngineering, 2004.

[24] S. Mallick, «OpenCV (C++ vs Python) vs MATLAB for Computer Vision,» 30 Octubre 2015. [En línea]. Available: https://www.learnopencv.com/opencv-c-vs-python-vs-matlab-for-computer-vision/. [Último acceso: 23 Octubre 2019].

[25] P. Viola y M. Jones, «Rapid Object Detection usinga Boosted Cascade of Simple Features,» *Computer Vision and Pattern Recognition,* 2001.

[26] A. Islam, A. Naeem y N. Hasan, «Comparison Between Viola-Jones and KLT Algorithms and Error Correction of Viola-Jones Algorithm,» *International Journal of Computer Engineering and Applications,* vol. VI, nº 5, pp. 87-95, 2017.

[27] J. Chang-yeon, «Face Detection using LBP features,» 2018.

[28] H. Zhu, S. Zhu y T. Koga, «Face detection based on AdaBoost algorithm with differential images,» *International Conference on Audio, Language and Image Processing,* pp. 718-722, 2008.

[29] D. Wyld, M. Wozniak, N. Chaki, N. Meghanathan y D. Nagamalai, Advances in Computing and Information Technology, Springer, 2011.

[30] A. Taejoo, «Approaches to Creating a Beautiful Face,» Magazine for Dermatologists & Plastic Surgeons, 29 Septiembre 2016. [En línea]. Available: http://idnps.com/clinical/approaches-to-creating-a-beautiful-face/2-1-considering-an-attractive-face/. [Último acceso: 15 Septiembre 2019].

[31] C. Botello, Adobe Photoshop CS5 Illustrated, Boston: Cengage Learning, 2011.

[32] Z. Zhang, Multivariate Time Series Analysis in Climate and Environmental Research, Springer, 2018.

[33] A. Murphy, Rosenblatt's Perceptron, Pearson, 2008.

[34] R. Rojas, Neural Networks: A Systematic Introduction, Berlin: Springer, 1996.

[35] J. M. Marin, «Introducción a las redes neuronales aplicadas,» [En línea]. Available: http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf. [Último acceso: 23 Octubre 2019].

[36] MissingLink, «Neural Network Bias: Bias Neuron, Overfitting and Underfitting,» [En línea]. Available: https://missinglink.ai/guides/neural-network-concepts/neural-network-bias-bias-neuron-overfitting-underfitting/. [Último acceso: 23 Octubre 2019].

[37] G. Cybenko, «Approximation by Superpositions of a Sigmoidal Function,» *Mathematics of Control, Signal, and Systems,* vol. 2, pp. 303-314, 1989.

[38] G. Zhai, J. Zhou y X. Yang, de *Digital TV and Wireless Multimedia Communication*, Shanghai, Springer, 2017.

[39] L. Koskela, Application of the new production philosophy to construction, Finland: VTT Building Technology, 1992, p. 13.

[40] A. Hofacker, Rapid lean construction - quality rating model, Manchester: s.n., 2008.

[41] «Foundation for Medical Education and Research,» [En línea]. Available: https://www.mayoclinic.org/es-es/tests-procedures/eeg/multimedia/eeg-electrodes/img-20005916. [Último acceso: 2019].

[42] D. López, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA DETERMINAR CONDICIONES DE FATIGA EN UNA PERSONA MEDIANTE EL ÍNDICE PERCLOS, UTILIZANDO OPEN CV Y UNA TARJETA RASPBERRY PI 3, Quito, 2017.

[43] E. R. Davies, Image Processing for the Food Industry, Worl Scientific, 2000.

[44] A. Manna, «Introducción al Procesamiento de Imágenes con Matlab1era Parte,» Buenos Aires, 2016.

[45] R. E. Farrell, RNA Methodologies: Laboratory Guide for Isolation and Characterization, ELSEVIER, 2010.

[46] M. R. Peres, The Focal Encyclopedia of Photography, Elsevier, 2007.

[47] S. Nagabhushana, Computer Vision and Image Processing, Delhi: New Age International, 2005.

[48] MathWorks, «vision.CascadeObjectDetector System object,» [En línea]. Available: https://la.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html#bs_kj2j. [Último acceso: Enero 2019].

[49] «Redes Neuronales Biológicas,» Noviembre 2012. [En línea]. Available: https://rneuronales.wordpress.com/tag/redes-neuronales-biologicas/. [Último acceso: Enero 2019].

[50] R. Grosse, Multilayer Perceptrons, Toronto, 2013.

[51] MathWorks, «integralImage,» 2019. [En línea]. Available: https://la.mathworks.com/help/vision/ref/integralimage.html. [Último acceso: 24 Agosto 2019].

[52] F. Nielsen y R. Bhatia, Matrix Information Geometry, Spreinger, 2013.

[53] S. Minz, S. Karmakar y L. Kharb, Information, Communication and Computing Technology.

[54] K. L. Priddy y P. E. Keller, Artificial Neural Networks: An Introduction, Washington: Spie Press, 2005.

[55] M. Fritz, B. Schiele y J. Piater, Computer Vision Systems, Bélgica: Springer, 2009.

[56] N. Triana, A. Jaramillo, R. Gutiérrez y C. Rodríguez, «Técnicas de umbralización para el procesamiento digital de imágenes de GEM-Foils,» *Scientia et Technic,* vol. 4, nº 21, pp. 352-359, 2016.

[57] N. Stanton, A. Hedge, K. Brookhuis, E. Salas y H. W. Hendrick, Handbook of Human Factors and Ergonomics Methods, CRC Press, 2006.

[58] S. Kolkur, D. Kalbande, P. Shimpi, C. Bapat y J. Jatakia, «Human Skin Detection Using RGB, HSV and YCbCr Color Models,» Agosto 2017. [En línea]. Available: https://arxiv.org/ftp/arxiv/papers/1708/1708.02694.pdf. [Último acceso: Marzo 2019].

[59] S. Minz, S. Karmakar y L. Kharb, Information, Communication and Computing Technology, Nueva Deli: Springer, 2018.

[60] C. Steinle, A First Level Trigger Approach for the CBM Experiment, disserta Verlag, 2012.

[61] J. D. Bossler, J. B. Campbell, R. B. McMaster y C. Rizos, Manual of Geospatial Science and Technology, CRC Press, 2010.

[62] Google trends, «Google Trends,» Google, Enero 2019. [En línea]. Available: https://trends.google.com/trends/explore?cat=5&date=2008-01-01%202019-01-15&q=%2Fm%2F0311pr.

[63] Y. Wang, «An Analysis of the Viola-Jones Face Detection Algorithm,» *Image Processing On Line,* nº 4, pp. 128-148, 2014.

[64] W. E. Snyder y H. Qi, Fundamentals of Computer Vision, Cambridge University Press, 2017.

ANEXOS

ANEXO A. Paper que se publicará en IEEE Xplore.

ANEXO B. Pruebas de adquisición de imágenes con distintas resoluciones.

ANEXO C. Funciones de Matlab para la detección de personas.

ANEXO D. Características de Haar.

ANEXO E. Pruebas para elegir el valor MergeThreshold.

ANEXO F. Transformadas de imágenes.

ANEXO G. Pruebas de umbral para binarización de imagen del ojo.

ANEXO H. Pruebas de división de base de datos para entrenamiento de red neuronal.

ANEXO I. Porcentajes de clasificación con distintas neuronas en la capa oculta.

ORDEN DE EMPASTADO