UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO

FACULTAD DE CIENCIAS QUÍMICAS, FÍSICAS Y MATEMÁTICAS

CARRERA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

"Reconocimiento de Somnolencia en Conductores bajo Condiciones Simuladas"

Tesis presentada por los Bachilleres:

RONDON CONDORI, LUIS ANGEL

PAUCARA NUÑEZ, FREDERICK JACINTO

Para optar al Título Profesional de:

INGENIERO INFORMÁTICO Y DE SISTEMAS

Asesor:

LIC. JUAN ANTONIO CRUZ TELLO

CUSCO – PERÚ

2013

TESIS AUSPICIADA POR EL CONSEJO DE INVESTIGACIÓN DE LA UNSAAC

RESUMEN

La somnolencia en conductores es una de las causas de accidentes de tránsito, por lo cual detectar el estado de somnolencia y advertir al conductor es una forma de solucionar este problema. Este proyecto tiene como objetivo principal la utilización de algoritmos de detección de objetos para reconocer el estado de somnolencia en conductores, por lo que debe trabajar con información visual obtenida del rostro del conductor. Para reconocer el estado de somnolencia se capturan fotogramas del conductor usando una webcam, cada fotograma es evaluado buscando primero detectar un rostro, si un rostro es detectado entonces se evalúa el estado de los ojos ("abiertos" o "cerrados"), con la información del estado de los ojos de los 10 últimos fotogramas se calcula el porcentaje de ojos cerrados o PERCLOS, para un PERCLOS mayor a 40% consideramos que el estado de somnolencia del conductor es peligroso y se muestra una señal de alarma. Las pruebas se realizaron bajo las condiciones simuladas siguientes: Luz natural diurna, una webcam de 640x480 píxeles ubicada a una distancia de 40 a 60cm del conductor y a la altura del volante de un auto sin movimiento, se obtuvo un error de hasta 8% en la clasificación de ojos "abiertos" o "cerrados", se utilizó el algoritmo de detección de objetos de Viola & Jones implementado en la librería OpenCV para ubicar el rostro y buscar ojos abiertos usando el lenguaje de programación C#, este algoritmo tiene buen desempeño en ojos con apertura de párpados mayores a 7mm, y con un índice PERCLOS > 40% la señal de alarma es mostrada en un tiempo promedio de 299ms desde que se detectaron los ojos cerrados.

Palabras Clave: Visión artificial, Detección de somnolencia, PERCLOS, detección de rostros, detección de ojos, OpenCV, accidentes de tránsito.

ABSTRACT

The drowsy on drivers is one of the causes of traffic accidents, detecting the state of drowsiness and warn the driver is one way to solve this problem. This project's main objective is the use of object detection algorithms to recognize the state of sleepiness in drivers, so it should work with visual information obtained from the driver's face. To recognize the state of drowsy driver frames are captured using a webcam, every frame is evaluated seeking first to detect a face, if a face is detected then assesses the state of the eyes ("open" or "closed"), with the status information from the eyes of the last 10 frames the percentage of eyes closed or PERCLOS is calculated, for a greater than 40% PERCLOS the driver drowsiness state is dangerous and shows an alarm signal. The tests were conducted under simulated conditions following: Natural light day, a webcam of 640x480 pixels located at a distance of 40 to 60cm and over driving of car without movement, it got an error of up to 8% in the classification eyes "open" or "closed", we used the detection algorithm objects Viola & Jones implemented in the OpenCV library to locate the face and searching eyes open using the programming language C #, this algorithm has good performance in eye with eyelid opening greater than 7mm, and with a PERCLOS index > 40% the alarm signal is shown in an average time of 299ms since they were detected eyes closed.

Keywords: Artificial vision, drowsiness detection, PERCLOS, face detection, eye detection, OpenCV, traffic accident.

AGRADECIMIENTOS

Son muchas las personas que nos han acompañado hasta este momento, tanto en el campo profesional como en nuestra propia vida. A ustedes nos gustaría agradecer su amistad, consejos, apoyo, ánimo y compañía. Algunas están aquí y otras en nuestros recuerdos, sin importar donde estén queremos agradecerles sinceramente por todo aquello que nos brindaron y por todas sus bendiciones.

Para ustedes: Muchas gracias y que Dios los bendiga.

"De nuestros miedos nacen nuestros corajes, y en nuestras dudas viven nuestras certezas. Los sueños anuncian otra realidad posible, y los delirios otra razón. En los extravíos nos esperan los hallazgos porque es preciso perderse para volver a encontrarse."

Eduardo Galeano

ÍNDICE

RESUMEN II

ABSTRACT III

AGRADECIMIENTOS IV

ÍNDICE VI

INTRODUCCIÓN XI

CAPÍTULO I 1

ASPECTOS GENERALES 1

1.1 PLANTEAMIENTO DEL PROBLEMA 1

1.1.1 DESCRIPCIÓN DEL PROBLEMA *1*

1.1.2 FORMULACIÓN DEL PROBLEMA *3*

1.2 OBJETIVOS DE LA TESIS 4

1.2.1 OBJETIVO GENERAL *4*

1.2.2 OBJETIVOS ESPECÍFICOS *4*

1.3 JUSTIFICACIÓN 4

1.4 METODOLOGÍA 5

1.5 LÍMITES 6

1.6 PLAN DE TRABAJO 6

CAPÍTULO II 9

ESTADO DE ARTE Y MARCO TEÓRICO 9

2.1 ESTADO DE ARTE 9

2.1.1 ACCIDENTES DE TRÁNSITO POR SOMNOLENCIA *9*

2.1.2 MÉTODOS USADOS PARA RECONOCIMIENTO DE SOMNOLENCIA *10*

2.1.3 SISTEMAS DE AYUDA A LA CONDUCCIÓN EXISTENTES *14*

2.2 MARCO TEÓRICO 18

2.2.1 SOMNOLENCIA *18*

2.2.2 OJOS HUMANOS *20*

2.2.3 CARACTERÍSTICAS VISUALES DE SOMNOLENCIA *22*

2.2.4 PORCENTAJE DE OJOS CERRADOS (PERCLOS) *22*

2.2.5 PROPORCIONES FACIALES *23*

2.2.6 VISIÓN ARTIFICIAL O VISIÓN POR COMPUTADORA *25*

2.2.7 CÁMARAS WEB *27*

2.2.8 SIMULACIÓN *30*

2.2.9 OPENCV *30*

2.2.10 DETECCIÓN DE ROSTROS *34*

2.2.11 DETECCIÓN DE ROSTROS CON OPENCV *34*

2.2.12 OTROS ALGORITMOS PARA LA DETECCIÓN DE OJOS *42*

CAPÍTULO III 46

RECONOCIMIENTO DE SOMNOLENCIA MEDIANTE LA DETECCIÓN DE OBJETOS 46

3.1 DESCRIPCIÓN DEL ALGORITMO 46

3.1.1 ADQUIRIR IMAGEN DE WEBCAM *50*
3.1.2 DETECTAR ROSTROS *51*
3.1.3 ESTABLECER REGIÓN DE INTERÉS *53*
3.1.4 BUSCAR OJOS ABIERTOS *55*
3.1.5 CALCULAR PERCLOS *57*
3.1.6 MÉTODO PRINCIPAL *61*
3.2 INTERFAZ DE CONFIGURACIÓN INICIAL 64
CAPITULO IV 65
RESULTADOS DE LA EXPERIMENTACIÓN 65
4.1 METODOLOGÍA PARA LA EXPERIMENTACIÓN 65
4.2 PRUEBA UNO CON LUZ DIURNA 66
4.2.1 ALGUNOS FOTOGRAMAS CLASIFICADOS COMO ABIERTOS *66*
4.2.2 ALGUNOS FOTOGRAMAS CLASIFICADOS COMO CERRADOS *67*
4.2.3 FOTOGRAMAS CLASIFICADOS COMO ABIERTOS INCORRECTAMENTE *68*
4.3 PRUEBA DOS CON LUZ DIURNA 69
4.3.1 ALGUNOS FOTOGRAMAS CLASIFICADOS COMO ABIERTOS *69*
4.3.2 ALGUNOS FOTOGRAMAS CLASIFICADOS COMO CERRADOS *70*
4.3.3 FOTOGRAMAS CLASIFICADOS COMO ABIERTOS INCORRECTAMENTE *71*
4.4 ANÁLISIS DE LOS RESULTADOS 72
4.5 LIMITACIONES 73
4.5.1 DEPENDENCIA DE LA ILUMINACIÓN *73*
4.5.2 DISTANCIA DEL CONDUCTOR A LA CÁMARA *74*
4.5.3 ORIENTACIÓN DEL ROSTRO *75*
4.5.4 POBRE DETECCIÓN CON LENTES *75*
4.5.5 APERTURA DE LOS PÁRPADOS *76*
CONCLUSIONES 78
RECOMENDACIONES 80
REFERENCIAS BIBLIOGRÁFICAS 82
REFERENCIAS WEB 84
ANEXOS 86
GLOSARIO DE TÉRMINOS 87

ÍNDICE DE FIGURAS

FIGURA 1: MORTALIDAD POR ACCIDENTES DE TRÁNSITO....2
FIGURA 2: MORTALIDAD POR ACCIDENTES DE TRÁNSITO DEBIDO AL CANSANCIO....2
FIGURA 3: NÚMERO DE ACCIDENTES INMINENTES O CONSUMADOS, SEGÚN LA HORA DEL DÍA, SUFRIDOS POR LOS CONDUCTORES DE ÓMNIBUS. LIMA, PERÚ....3
FIGURA 4: PLAN DE TRABAJO....8
FIGURA 5: SOMNOLENCIA DURANTE LA CONDUCCIÓN....9
FIGURA 6: EFECTOS DE SOMNOLENCIA EN LA CONDUCCIÓN....10
FIGURA 7: CLASIFICACIÓN DE LOS MÉTODOS PARA RECONOCIMIENTO DE SOMNOLENCIA....11
FIGURA 8: CLASIFICACIÓN DE MÉTODOS DE RECONOCIMIENTO DE SOMNOLENCIA COMO INTRUSIVOS Y NO INTRUSIVOS....12
FIGURA 9: EJEMPLO DE ANÁLISIS BASADO EN PATRONES DE LA CONDUCCIÓN MONITORIZANDO PRESIÓN AL VOLANTE....13
FIGURA 10: ANÁLISIS DE OJO PARA VALIDAR ESTADO DE SOMNOLENCIA....13
FIGURA 11: EJEMPLO DE MONITOREO DE SOMNOLENCIA CON SENSORES EN EL CONDUCTOR....14
FIGURA 12: SISTEMA DE DETECCIÓN DE SOMNOLENCIA DE BOSCH....15
FIGURA 13: SISTEMA DE AYUDA A LA CONDUCCIÓN DE VOLVO CARS....16
FIGURA 14: ASIENTO PARA DETECCIÓN DE SOMNOLENCIA DE DELTA TOOLING....17
FIGURA 15: ANATOMÍA DEL OJO HUMANO....20
FIGURA 16: PARTES EXTERNAS DEL OJO....21
FIGURA 17: CARACTERÍSTICAS DE UN PESTAÑEO REFLEJO....22
FIGURA 18: ANCHURA DEL ROSTRO SE DIVIDE EN 5 PARTES IGUALES....23
FIGURA 19: ALTURA DEL ROSTRO DIVIDIDO EN 3 PARTES IGUALES....24
FIGURA 20: EJEMPLO DE ROSTRO DIVIDIDO EN 5 PARTES IGUALES....24
FIGURA 21: EJEMPLO DE ROSTRO DIVIDIDO EN 3 PARTES IGUALES....24
FIGURA 22: EJEMPLO DE BINARIZACIÓN DE IMÁGENES....27
FIGURA 23: OBTENCIÓN DE IMÁGENES POR WEBCAM....28
FIGURA 24: IMAGEN CAPTURADA CON WEBCAM INFRARROJA....29
FIGURA 25: EJEMPLO DE RESOLUCIÓN DE IMÁGENES....29
FIGURA 26: VISTA GENERAL DE FUNCIONES DE OPENCV....32
FIGURA 27: EJEMPLO DE FALSO POSITIVO EN DETECCIÓN DE ROSTROS....33
FIGURA 28: REPRESENTACIÓN DE LA IMAGEN INTEGRAL....35
FIGURA 29: EJEMPLO DE IMAGEN INTEGRAL....35
FIGURA 30: EJEMPLO CÁLCULO EN IMÁGENES INTEGRALES....36
FIGURA 31: TIPOS DE FEATURES....37
FIGURA 32: EJEMPLO DE FEATURES USADOS EN DETECCIÓN DE ROSTROS....37
FIGURA 33: ESTRUCTURA DE CLASIFICACIÓN EN CASCADA....39
FIGURA 34: EJEMPLO DE CLASIFICACIÓN EN CASCADA PARA DETECTAR ROSTROS....40
FIGURA 35: VARIACIÓN EN EL NÚMERO DE VECINOS. IZQUIERDA 4, DERECHA 0....41
FIGURA 36: RESULTADO DE LA DETECCIÓN DE BORDES DE CANNY....43
FIGURA 37: ALGORITMO DE HOUGH: CADA PUNTO DEL CONTORNO DE LA IMAGEN SIRVE PARA TRAZAR UN CÍRCULO DE HOUGH....43
FIGURA 38: DETECCIÓN DE CÍRCULOS CON IMAGEN DE LA BASE DE DATOS CASIA....44
FIGURA 39: DETECCIÓN DE CÍRCULO CON IMAGEN DE LA WEBCAM....44
FIGURA 40: EJEMPLO DE REGIÓN DE INTERÉS PARA BUSCAR LOS OJOS....47
FIGURA 41: EJEMPLO DE FALSO POSITIVO AL BUSCAR UN OJO....47
FIGURA 42: PROPORCIONES DEL ROSTRO HUMANO....48
FIGURA 43: REGIÓN DE INTERÉS PARA LA BÚSQUEDA DE OJOS....48
FIGURA 44: DIAGRAMA DE ACTIVIDADES DEL ALGORITMO PROPUESTO....49
FIGURA 45: IDEA DEL ALGORITMO....50

FIGURA 46: RESULTADO DEL ALGORITMO 1 53
FIGURA 47: DISTANCIAS PARA EL CÁLCULO DE LA REGIÓN DE INTERÉS. 53
FIGURA 48: RESULTADO DEL ALGORITMO 2 54
FIGURA 49: RESULTADO DEL ALGORITMO 3 55
FIGURA 50: RESULTADO DEL ALGORITMO 4 57
FIGURA 51: COLOR MOSTRADO SEGÚN ESTADO DE SOMNOLENCIA DEL CONDUCTOR 61
FIGURA 52: DIAGRAMA DE SECUENCIA DEL ALGORITMO 62
FIGURA 53: MUESTRA DE LA INTERFAZ DEL MÓDULO PRINCIPAL 63
FIGURA 54: MUESTRA DE LA INTERFAZ DEL MÓDULO PRINCIPAL 63
FIGURA 55: VISTA DE LA CONFIGURACIÓN INICIAL PARA PROBAR LA HERRAMIENTA CONSTRUIDA 64
FIGURA 56: FOTOGRAMAS CLASIFICADOS COMO ABIERTOS EN PRUEBA 1 66
FIGURA 57: FOTOGRAMAS CLASIFICADOS COMO CERRADOS EN PRUEBA 1 67
FIGURA 58: FALSOS CERRADOS EN PRUEBA 1 68
FIGURA 59: FOTOGRAMAS CLASIFICADOS COMO ABIERTOS EN PRUEBA 2 69
FIGURA 60: FOTOGRAMAS CLASIFICADOS COMO CERRADOS EN PRUEBA 2 70
FIGURA 61: FALSOS ABIERTOS EN PRUEBA 2 71
FIGURA 62: ALGORITMO DE SEIFOORY ET. AL. [28] 72
FIGURA 63: IMAGEN OBTENIDA CON WEBCAM INFRARROJA 74
FIGURA 64: EN LA FIGURA: (A) ROSTRO MUY ALEJADO. (B) ROSTRO MUY CERCANO (C) ROSTRO A UNA DISTANCIA ADECUADA. 75
FIGURA 65: ROSTROS NO DETECTADOS POR INCLINACIÓN Y GIROS DEL ROSTRO 75
FIGURA 66: USO DE LENTES EN CONDUCTORES POR EDADES 76

ÍNDICE DE TABLAS

TABLA 1: COMPARACIÓN ENTRE LENGUAJE C Y C# 6
TABLA 2: APLICACIONES DE LA VISIÓN ARTIFICIAL 26
TABLA 3: DURACIÓN EN MILISEGUNDOS DE CLASIFICACIÓN DE OJOS CERRADOS EN 5 PRUEBAS 58
TABLA 4: RESULTADOS DE LA PRUEBA 1 66
TABLA 5: RESULTADOS DE LA PRUEBA 2 69
TABLA 6: ERRORES OBTENIDOS EN LAS PRUEBAS 1 Y 2 72
TABLA 7: RESULTADOS OBTENIDOS POR SEIFOORY ET. AL. [28] 73
TABLA 8: PRUEBA CON OJOS DE 7 A 9MM DE APERTURA DE PÁRPADOS 77
TABLA 9: TIEMPO DE EVALUACIÓN DE LOS FOTOGRAMAS 89

INTRODUCCIÓN

Hoy en día de una u otra forma dependemos del transporte en nuestra vida diaria, sin embargo, al haber un incremento constante del tráfico automovilístico también se incrementan los problemas sociales. Entre los más graves se encuentran los accidentes de tránsito, cada año mueren en las carreteras cientos de personas y muchas más resultan heridas consecuencia de los accidentes de tránsito. A pesar de que muchos de estos accidentes se dan por problemas mecánicos o factores de tiempo gran parte de ellos se deben a errores humanos. De hecho, de acuerdo con estudios recientes, la fatiga y somnolencia del conductor son algunas de las principales causas de los accidentes de tránsito. Por lo tanto, tenemos la necesidad de implementar sistemas de seguridad que contribuyan a aumentar la confianza en la conducción y minimizar las probabilidades de que ocurra un accidente de tránsito.

Esta es la principal motivación para el desarrollo del presente trabajo. Existen diversos enfoques orientados a la solución de este problema. Tenemos enfoques basados en el análisis de señales cerebrales, detección de la temperatura corporal, y otros enfoques basados en signos físicos relacionados con el sueño tales como cabeceo, bostezo, parpadeo o pestañeo prolongado además de la combinación de estos, utilizando en muchos de los casos técnicas de análisis y procesamiento de imágenes.

El objetivo del presente trabajo es utilizar los algoritmos de detección de objetos para reconocer el estado de somnolencia bajo condiciones simuladas, para esto nos basaremos en el enfoque basado en los signos relacionados con el sueño, en particular trabajaremos evaluando el pestañeo o parpadeo prolongado, para lo cual nos centraremos en el problema de la detección del rostro humano, y posteriormente buscar los ojos en el rostro detectado. Utilizaremos una cámara web como dispositivo sensor para monitorear al conductor, se calcula el índice PERCLOS, el cual es el porcentaje de ojos cerrados en un tiempo dado, a partir del valor del índice PERCLOS es que se decide si un conductor está entrando en un estado de somnolencia peligroso con lo cual se da la alarma al conductor.

El desarrollo del presente trabajo se dividió en los siguientes capítulos:

En el primer capítulo se mencionan los aspectos generales de la tesis: La formulación del problema, objetivos y metodología utilizada.

En el segundo capítulo presentamos el estado de arte y el marco teórico que nos ayudará en la comprensión del tema de estudio como son: La visión artificial, sistemas comerciales actuales, la somnolencia y sus características visibles, OpenCV y detección de rostros y ojos.

En el tercer capítulo planteamos el diseño del algoritmo para el reconocimiento de somnolencia usando información visual obtenida por webcam y analizando el estado de los ojos para calcular el índice PERCLOS como indicador de estados de somnolencia.

En el cuarto capítulo planteamos un método para probar la herramienta desarrollada y mostrando los resultados obtenidos.

En la parte final se incluyen las conclusiones a las que se han llegado en este trabajo además de las recomendaciones necesarias para futuras investigaciones relacionadas al tema de reconocimiento de somnolencia.

CAPÍTULO I

ASPECTOS GENERALES

1.1 PLANTEAMIENTO DEL PROBLEMA

1.1.1 DESCRIPCIÓN DEL PROBLEMA

Según una publicación hecha por el Ministerio de Salud: "Los accidentes de tránsito son uno de los principales problemas de salud pública y de desarrollo en el mundo, y afectan de forma desproporcionada a determinados grupos vulnerables de usuarios de la vía pública.

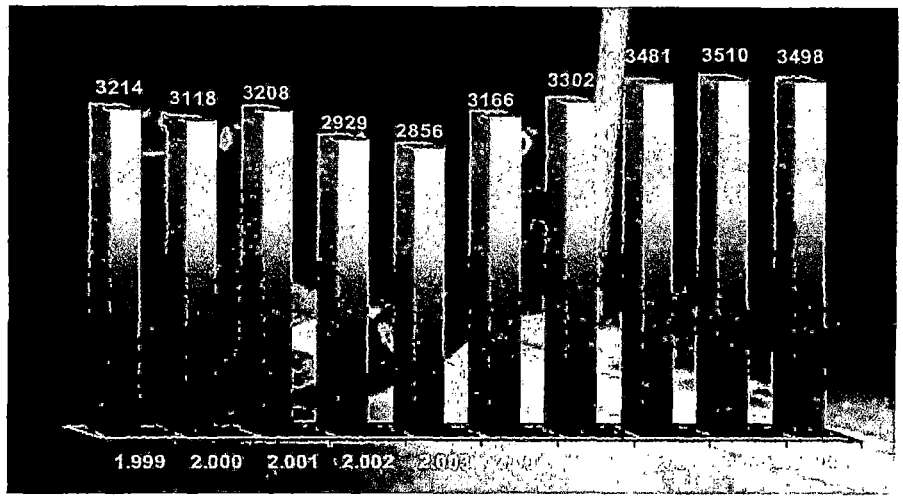
Se producen a consecuencia de una acción riesgosa, irresponsable o negligente de un conductor, pasajero o peatón, ya sea en las vías de una ciudad o en carretera. Se puede decir que gran parte de los accidentes de tránsito son predecibles y evitables, por eso es importante incidir en las campañas de prevención contra estos eventos. [1]"

Además en otra publicación hecha también por el Ministerio de Educación [2] se mencionan los factores causantes de accidentes de tránsito, como son:

- Alcohol
- Drogas
- Cansancio y Somnolencia
- Medicamentos
- Edad

Así mismo en [2] se indica que: "los accidentes de tránsito en nuestro país han ocasionado en el período 1998-2008 35605 víctimas mortales y 342766 lesionados, incrementándose las cifras año tras año."

Figura 1: Mortalidad por Accidentes de Tránsito

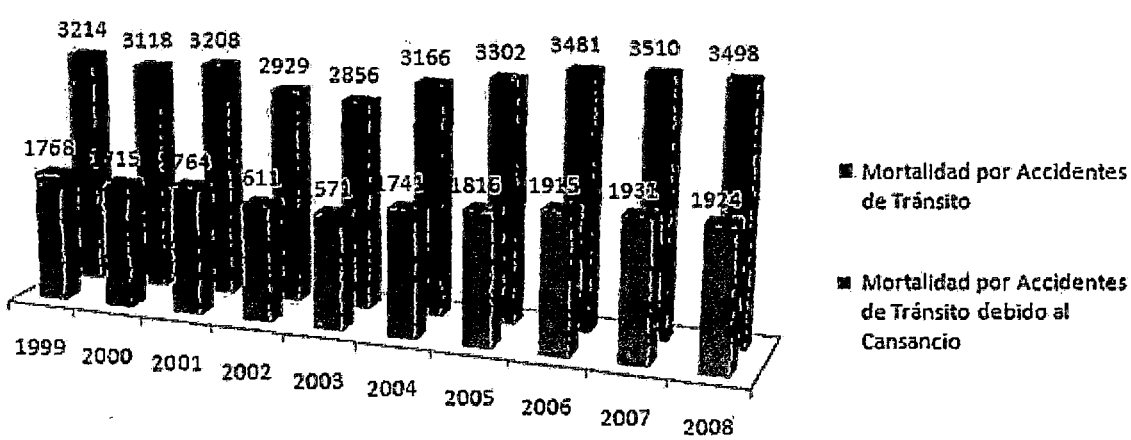


Fuente: Ministerio de Salud [2], p.12

Por otro lado en una encuesta hecha a 238 conductores de buses interprovinciales los autores señalan que "en la opinión de los encuestados, las causas más frecuentes de accidentes en las carreteras del Perú son: cansancio durante la conducción (55%), imprudencia del chofer (24%), exceso de velocidad (5%), ebriedad (4%), falla mecánica (3%), mal estado de las pistas(1%) y negligencia de las empresas de transportes (1%). [3]"

Tomando como referencia estos datos podríamos inferir que de las 35605 víctimas mortales, por accidentes de tránsito en el periodo de 1998-2008, 19583 son consecuencia del cansancio durante la conducción.

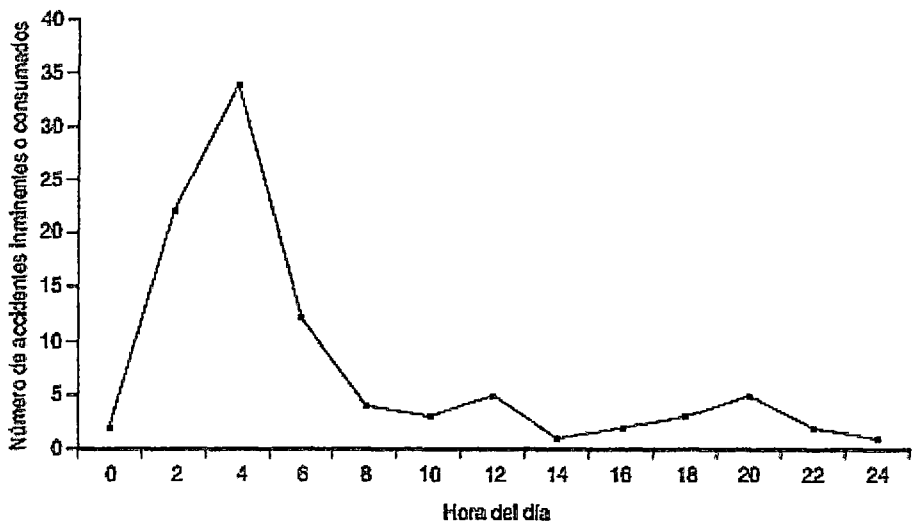
Figura 2: Mortalidad por Accidentes de Tránsito debido al cansancio



Fuente: Elaboración propia con los datos del Ministerio de Salud [2]

En el siguiente cuadro además podemos observar que proporcionalmente ocurren más accidentes de tránsito en horas de la madrugada lo que se puede deber a problemas de somnolencia.

Figura 3: Número de accidentes inminentes o consumados, según la hora del día, sufridos por los conductores de ómnibus. Lima, Perú



Fuente: Rey de Castro et al. [3]p.14

Siendo los accidentes de tránsito un problema que conlleva a la morbilidad de muchas personas y otras más lesionadas, y ya que la somnolencia durante la conducción es una de las causas de accidentes de tránsito se hace necesario disponer de herramientas que permitan monitorear el estado de somnolencia del conductor para reconocer su estado de somnolencia y así poder mostrar una señal de alarma para prevenir que se quede dormido.

1.1.2 FORMULACIÓN DEL PROBLEMA

Según lo antes expuesto los factores humanos y entre ellos la somnolencia en el conductor son causantes de accidentes de tránsito, para evitar muchos de estos accidentes se debería alertar al conductor cuando este entrando en un estado de somnolencia peligroso, esto nos lleva a formularnos el siguiente problema: ¿Se puede aplicar algoritmos de procesamiento de imágenes para reconocer el estado de somnolencia en conductores?

1.2 OBJETIVOS DE LA TESIS

1.2.1 OBJETIVO GENERAL

Utilizar algoritmos de detección de objetos para reconocer el estado de somnolencia en conductores bajo condiciones simuladas.

1.2.2 OBJETIVOS ESPECÍFICOS

1. Determinar aspectos físicos visuales que reflejen estado de somnolencia en una persona.
2. Utilizar un índice de somnolencia para determinar un estado de somnolencia peligroso.
3. Utilizar un algoritmo de detección de objetos que usando la información visual del rostro del conductor evalúe el estado de sus ojos.
4. Evaluar la eficacia del reconocimiento de somnolencia usando el lenguaje de programación C# y la librería OpenCV bajo condiciones simuladas, y comparando los resultados obtenidos con una clasificación manual del estado de los ojos.

1.3 JUSTIFICACIÓN

En nuestro país en el periodo de 1998-2008 por causas de accidentes de tránsito se tienen 35, 605 víctimas mortales y 342,766 lesionados según datos del Ministerio de Salud [2], además teniendo en cuenta que uno de los factores causantes de accidentes de tránsito es el cansancio y la somnolencia [2], consideramos que este proyecto es importante porque reconocer estados de somnolencia en conductores ayudará a reducir los accidentes de tránsito causados por somnolencia, y por ende contribuirá a la reducción de los índices de pérdidas personales y económicas.

1.4 METODOLOGÍA

El trabajo propuesto será de carácter descriptivo y aplicativo [4]. En la investigación se realiza un estudio descriptivo que permite poner de manifiesto los conocimientos teóricos que describen a nuestro problema. Y es aplicada ya que los conocimientos adquiridos serán usados para buscar la resolución del problema, es decir, se aplicarán técnicas y estrategias para enfrentar y solucionar el problema.
Para ello y con el fin de alcanzar nuestros objetivos se realizarán las siguientes actividades:

1. Se identifican las características visuales que permiten reconocer el estado de somnolencia en una persona, en particular trabajaremos con el parpadeo prolongado como característica de somnolencia.
2. Se establece un índice de somnolencia, el índice PERCLOS calcula el porcentaje de ojos cerrados en un tiempo dado y puede ser usado como indicador de somnolencia.
3. Se obtienen fotogramas del conductor usando una webcam y se detecta el rostro del conductor.
4. En el rostro detectado se clasifican el estado de los ojos como "abiertos" o "cerrados", esta información es usada para calcular el índice PERCLOS.
5. Con el valor del índice PERCLOS se evalúa el estado de somnolencia del conductor y se muestra una alarma si su estado de somnolencia es peligroso es decir si el valor del índice PERCLOS > 40%.
6. Se construye un prototipo del reconocimiento de somnolencia usando el lenguaje de programación C# por considerarla más conveniente principalmente por ya tener experiencia con ese lenguaje, y se usa la librería OpenCV como herramienta para la detección de rostros y evaluación del estado de los ojos el mismo que ya tiene implementados algoritmos de detección de objetos.

Tabla 1: Comparación entre lenguaje C y C#

	C	C#
Eficiencia	Excelente	Buena
Conocimiento del lenguaje	Ninguno	Amplio
OpenCV	Disponible	Disponible
Legibilidad	Buena	Excelente

Fuente: Elaboración propia en base a descripción vista en http://urriellu.net/es/articles-software/csharp-advantages.html

7. Se realizan pruebas de reconocimiento de somnolencia bajo condiciones simuladas, para esto se clasifican el estado de los ojos en los fotogramas de manera manual y se comparan con la clasificación hecha por el prototipo, de esta comparación se halla el error. Se utilizara una webcam con resolución VGA de 640x480 pixeles y 30fps ubicada a una distancia de entre 40 a 60cm del conductor, la computadora usada para las pruebas será una laptop HP de 8GB de RAM y procesador core I7 de segunda generación 2.2GHz con sistema operativo Windows 7 de 64 bits.

1.5 LÍMITES

- Todo el trabajo será probado en condiciones simuladas, por lo cual no se realizará la instalación de la herramienta en un auto en movimiento.
- Así mismo las condiciones de iluminación serán constantes por lo cual se trabajará con luz natural diurna.

1.6 PLAN DE TRABAJO

Buscando cumplir nuestros objetivos se plantea el siguiente plan de trabajo:

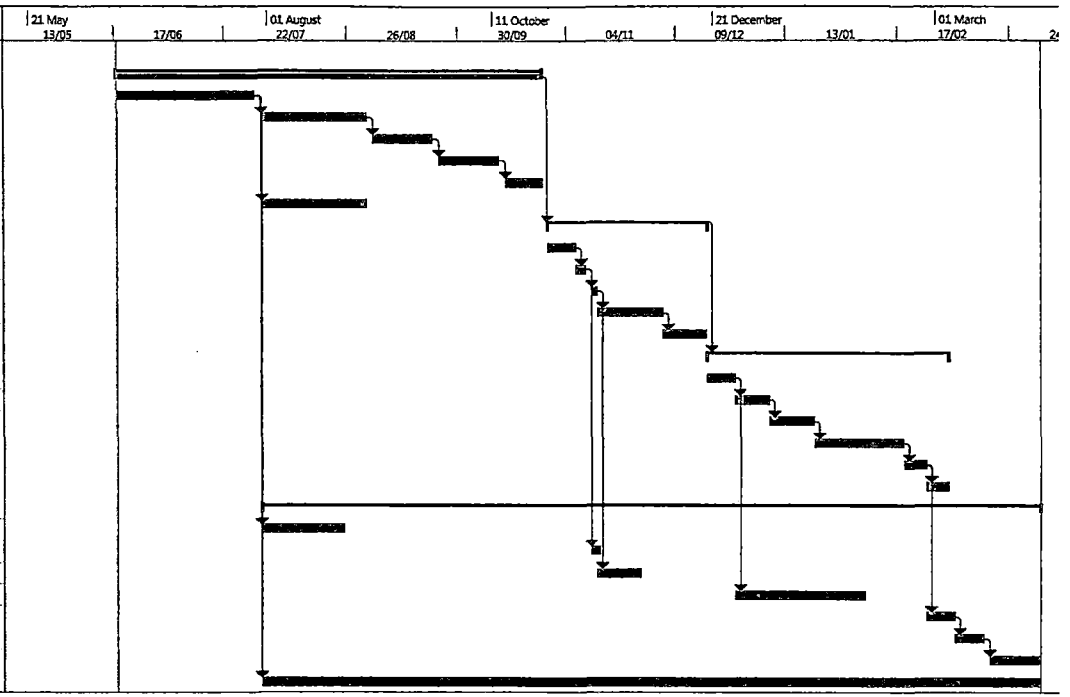
- Documentación: En esta parte se adquieren los conocimientos necesarios para empezar con nuestro trabajo. Consiste en lectura de documentación diversa

relacionadas al tema para poder entender el estado de arte y los métodos existentes de detección de somnolencia.

- Selección: Comienza tras el punto anterior y consiste en el entendimiento de las herramientas existentes, técnicas usadas, medidas del sueño y cuáles podemos utilizar para el desarrollo del presente trabajo.
- Desarrollo: Aplicaremos las técnicas y herramientas disponibles encontradas en la anterior etapa para evaluar el reconocimiento de somnolencia.
- Redacción: Aunque es el último paso del proyecto se ha llevado a cabo durante toda la duración del mismo, en esta parte redactamos la tesis en sí misma.

Figura 4: Plan de Trabajo

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1						
2		**Fase I: Documentación**	**95 days**	**Mon**	**Fri 26/10/12**	
3		Recolección de Información	30 days	Mon	Fri 27/07/12	
4		Lectura y Organización de la	25 days	Mon	Fri 31/08/12	3
5		Comprención del Estado de Arte	15 days	Mon	Fri 21/09/12	4
6		Familiarización con el	15 days	Mon	Fri 12/10/12	5
7		Familiarización con la Detección	10 days	Mon	Fri 26/10/12	6
8		Extracción de Información	25 days	Mon	Fri 31/08/12	3
9		**Fase II: Selección**	**37 days**	**Mon**	**Tue 18/12/12**	2
10		Recolección de librerias para	7 days	Mon	Tue 06/11/12	
11		Elección de Algoritmo para	3 days	Wed	Fri 09/11/12	10
12		Establecer Limites del Trabajo	2 days	Mon	Tue 13/11/12	11
13		Selección de las características	15 days	Wed	Tue 04/12/12	12
14		Planteamiento del	10 days	Wed	Tue 18/12/12	13
15		**Fase III: Desarrollo**	**54 days**	**Wed**	**Mon 04/03/13**	9
16		Analisis y Planteamiento de	7 days	Wed	Thu 27/12/12	
17		Planteamiento de Algoritmo	7 days	Fri 28/12/12	Mon 07/01/13	16
18		Construcción del Prototipo	10 days	Tue 08/01/13	Mon 21/01/13	17
19		Pruebas del algoritmo de	20 days	Tue 22/01/13	Mon 18/02/13	18
20		Optimización de Proceso	5 days	Tue 19/02/13	Mon 25/02/13	19
21		Pruebas de funcionamiento	5 days	Tue 26/02/13	Mon 04/03/13	20
22		**Fase IV: Redacción**	**177 days**	**Mon**	**Tue 02/04/13**	
23		Redacción de los Aspectos	20 days	Mon	Fri 24/08/12	3
24		Redacción del Estado del Arte	3 days	Mon	Wed 14/11/12	11
25		Redacción del Marco Teórico	10 days	Wed	Tue 27/11/12	12
26		Redacción del proceso de	29 days	Fri 28/12/12	Wed 06/02/13	16
27		Redacción de la	7 days	Tue 26/02/13	Wed 06/03/13	20
28		Redacción del Capitulo	7 days	Thu 07/03/13	Fri 15/03/13	27
29		Organización del Documento	12 days	Mon	Tue 02/04/13	28
30		Revisión y Corrección	177 days	Mon 30/07/12	Tue 02/04/13	3



Referencia: Elaboración propia

CAPÍTULO II

ESTADO DE ARTE Y MARCO TEÓRICO

2.1 ESTADO DE ARTE

2.1.1 ACCIDENTES DE TRÁNSITO POR SOMNOLENCIA

Muchos de los accidentes de tránsito se deben a errores humanos, entre ellos el cansancio y/o somnolencia del conductor, estas condiciones se derivan principalmente de la falta de sueño y periodos largos de conducción, tal como lo indica un estudio [3], en el cual se encuestaron 238 conductores de una población total de 400. Dicho estudio mostro que el 47% de los encuestados dormían menos de siete horas diarias y 40% menos de seis. Al momento de la encuesta hecha por este estudio, el 31% de los conductores habían dormido menos de seis horas de las últimas veinticuatro. Otra cosa que revelo el estudio fue que hasta 80% de los participantes reconocieron que solían conducir más de cinco horas sin detenerse. Además indican en el estudio que el 56% había presentado cansancio durante la conducción desde "alguna vez" hasta "siempre".

Figura 5: Somnolencia durante la Conducción

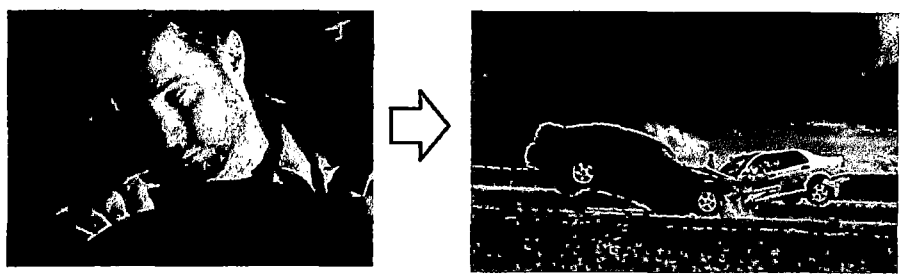
Fuente: Recuperado de http://noticias.coches.com/consejos/consejos-para-evitar-la-somnolencia-al-volante/3438

Último Acceso: 25 de mayo del 2013

Considerando lo anterior, podemos concluir que este factor de riesgo, la somnolencia, sirve de fuente para varios trabajos de investigación, por

ejemplo, por medio de la aplicación de diferentes técnicas en función al procesamiento de imágenes y sistemas de reconocimiento, para evaluar el nivel de somnolencia del conductor.

Figura 6: Efectos de Somnolencia en la Conducción



Fuente: Recuperado de http://sleepdisorders.dolyan.com/coping-with-sleepiness-fatigue-and-drowsy-driving/

Último Acceso: 25 de mayo del 2013

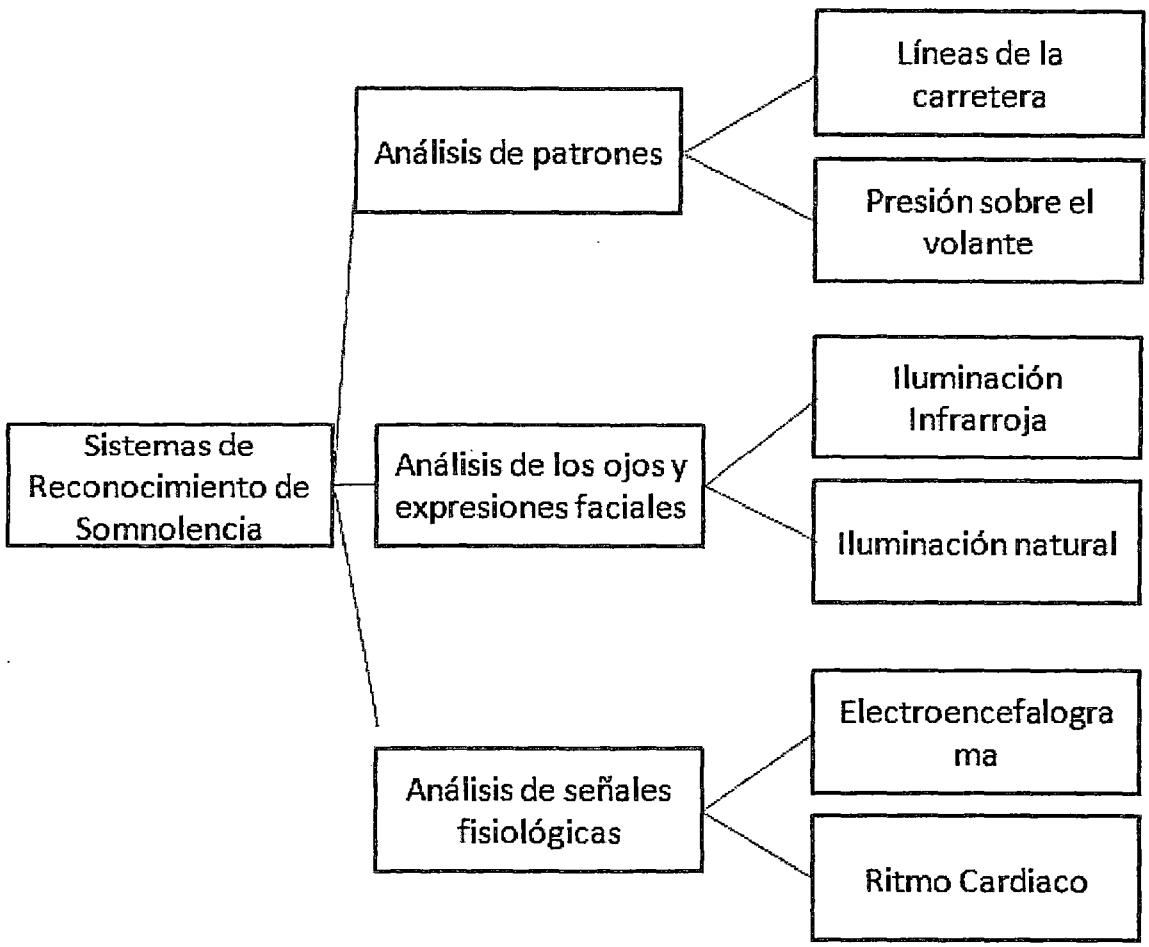
2.1.2 MÉTODOS USADOS PARA RECONOCIMIENTO DE SOMNOLENCIA

Para enfrentar el problema de reconocimiento de somnolencia en conductores se han planteado diferentes métodos y propuestas, según Flores [5]estos métodos se pueden clasificar en 3 categorías, los cuales son:

a) Análisis basados en patrones de conducción.
b) Análisis basados en cambios físicos de los ojos así como las expresiones faciales usando procesamiento de imágenes.
c) Análisis basados en el cambio de las medidas fisiológicas.

En la siguiente figura podemos observar mejor estos tres métodos:

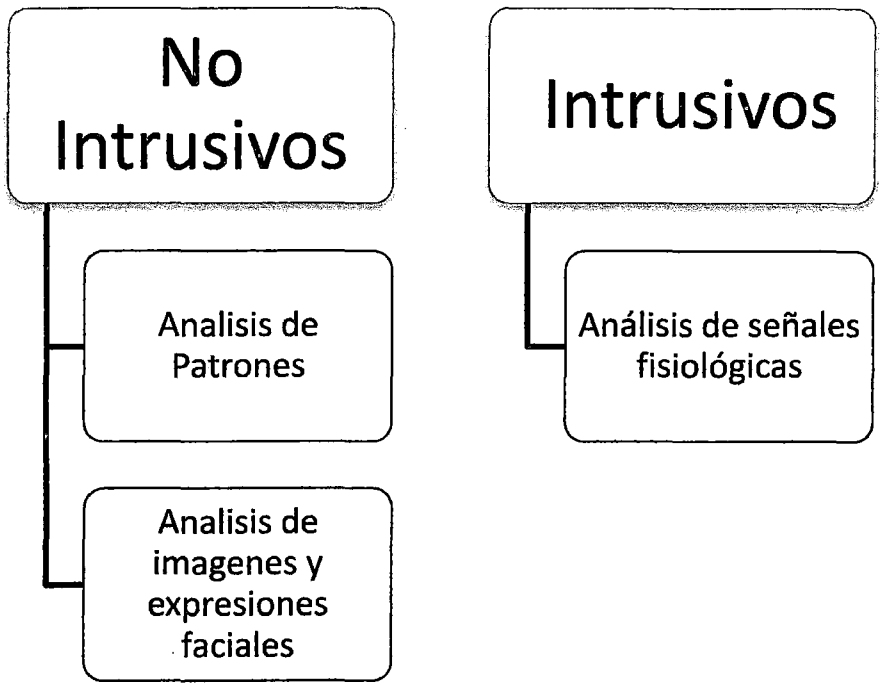
Figura 7: Clasificación de los métodos para reconocimiento de somnolencia



Fuente: Flores [5]. p.46

En un artículo sobre sistema avanzado de asistencia a la conducción [6] se habla de los métodos intrusivos o invasivos los cuales según los autores interfieren en la conducción por lo cual estos métodos para el reconocimiento de somnolencia pueden clasificarse en otros dos tipos: Métodos Intrusivos y métodos No Intrusivos. Los métodos intrusivos se denominan así porque afectan de alguna manera a la comodidad del conductor esto en casos como por ejemplo el uso de electrodos en la cabeza del conductor para medir diferentes señales fisiológicas, en cambio los métodos no intrusivos usan métodos que no interfieran en la forma de conducir minimizando las molestias.

Figura 8: Clasificación de métodos de Reconocimiento de Somnolencia como intrusivos y no intrusivos



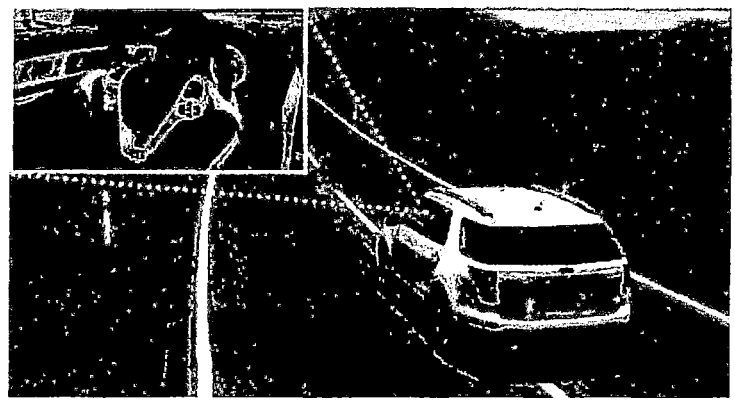
Fuente: Elaboración propia basados en descripción extraída de Flores et.al. [6] p.2

A continuación resumimos los métodos usados para reconocimiento de somnolencia descritos por Flores [5].

2.1.2.1 ANÁLISIS BASADOS EN PATRONES DE CONDUCCIÓN

En los métodos basados en patrones de conducción se construyen patrones generados a partir de diferentes experimentos. Este método es no intrusivo ya que no afecta al conductor, los patrones de conducción deben ser generados a partir de parámetros medibles y sobre características tanto del conductor y/o sobre el vehículo con su entorno, por ejemplo, se puede medir la presión que ejerce el conductor con sus manos sobre el volante del vehículo, también se podría tomar como parámetros indicadores del nivel de vigilia del conductor la posición lateral del auto respecto de la carretera u otros de este tipo. Este método no es fácil de modelar y puede ser necesario diseñar diferentes modelos para el mismo método dependiendo de las características del vehículo y/o conductor.

Figura 9: Ejemplo de Análisis basado en patrones de la conducción monitorizando presión al volante



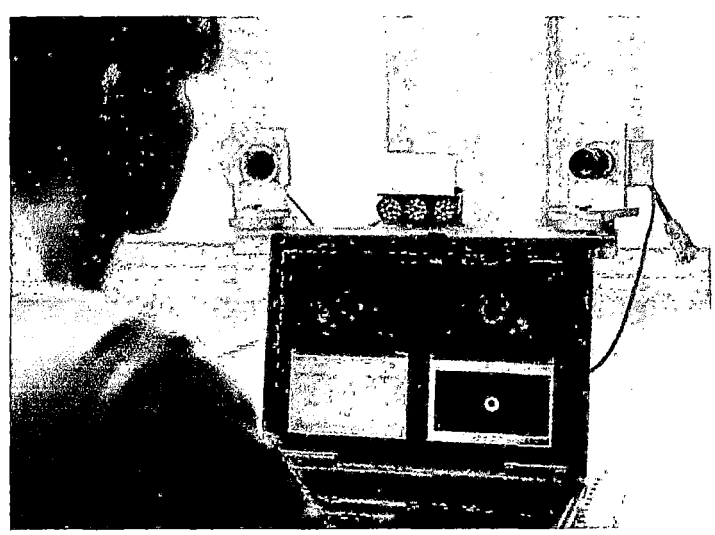
Fuente: Recuperado de http://carfanaticsforum.com/thread-10701.html

Último Acceso: 25 de mayo del 2013

2.1.2.2 ANÁLISIS BASADOS EN CAMBIOS FÍSICOS DE LOS OJOS ASÍ COMO LAS EXPRESIONES FACIALES USANDO PROCESAMIENTO DE IMÁGENES

Para este caso se utilizan técnicas de procesamiento de imágenes para reconocer los cambios físicos en los ojos y rostros del conductor durante la conducción y comparándolas con características propias de personas con sueño. Los métodos que basan su análisis en cambios físicos de los ojos y rostro son altamente fiables y tienen como ventaja el no ser invasivos o intrusivos, es decir, que no generan ni molestia ni incomodidad en el conductor ya que en este caso generalmente se usa una cámara que enfoca el rostro del conductor y se analiza el estado de sus ojos.

Figura 10: Análisis de ojo para validar estado de somnolencia



Fuente: Recuperado de http://phys.org/news/2010-10-driver-drowsiness-eyetracker.html

Último Acceso: 25 de mayo del 2013

2.1.2.3 ANÁLISIS BASADOS EN EL CAMBIO DE LAS MEDIDAS FISIOLÓGICAS

Para este método se mide los cambios fisiológicos que ocurren en el conductor tales como la variabilidad del ritmo cardiaco, y electroencefalograma, todos con el propósito de detectar los estados cognitivos humanos. Una clara ventaja de este método es la alta precisión en el diagnóstico del estado de somnolencia, mientras que, la mayor desventaja que tiene está en el uso de sensores conectados al cuerpo del conductor por lo que es un sistema altamente invasivo y puede generar incomodidad o molestias al conductor. Por ejemplo para el electroencefalograma se conectan varios electrodos a la cabeza durante la conducción y así poder registrar las ondas cerebrales.

Figura 11: Ejemplo de Monitoreo de Somnolencia con Sensores en el Conductor

Fuente: Recuperado de http://www.tecmovia.com/2011/08/24/reposacabezas-inteligentes-que-detectan-nuestra-somnolencia-de-la-mano-de-neurosky-y-gm/
Último Acceso: 25 de mayo del 2013

2.1.3 SISTEMAS DE AYUDA A LA CONDUCCIÓN EXISTENTES

Las principales empresas del sector automotriz están trabajando o participando directamente y/o a través de sus empresas fabricantes de componentes, en el diseño y construcción de sistemas de ayuda a la conducción. Entre estos sistemas se encuentran los sistemas de vigilancia del conductor las cuales utilizan distintas estrategias planteadas. Flores [5] recopila algunos de los sistemas comerciales existentes en la actualidad, los cuales según el autor trabajan mediante:

- Técnicas del análisis de los ojos y expresiones faciales
- Análisis de la carretera

- Información de las señales fisiológicas

Entre los sistemas comerciales que describe el autor tenemos:

2.1.3.1 Attention Technologies

Desarrollado por la empresa Attention Technologies Inc. Los cuales comercializan un sistema para la detección de somnolencia diseñados para vehículos de carga pesada, este sistema trabaja con el índice de somnolencia PERCLOS, funciona en tiempo real y tiene un sistema de visión artificial mediante iluminación infrarroja.

2.1.3.2 Seeing Machine

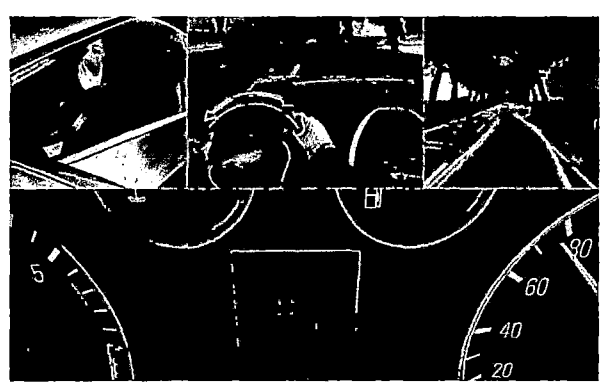
Seeing Machine es una empresa que desarrolla productos de alta tecnología, entre sus productos tiene *facelab* que mediante el uso de iluminación infrarroja puede ser usado para aplicaciones de biometría, seguimiento de ojos, y vigilancia de conductores.

2.1.3.3 Grupo Bosch

El Detector de Fatiga de Bosch puede detectar los primeros indicios de cansancio y somnolencia de los conductores a través de los movimientos del volante. Las informaciones necesarias provienen de un servo-dirección eléctrico o del sensor de ángulo de giro del volante.

El detector de Fatiga de Bosch es un claro ejemplo de los métodos basados en los patrones de la conducción.

Figura 12: Sistema de Detección de Somnolencia de Bosch



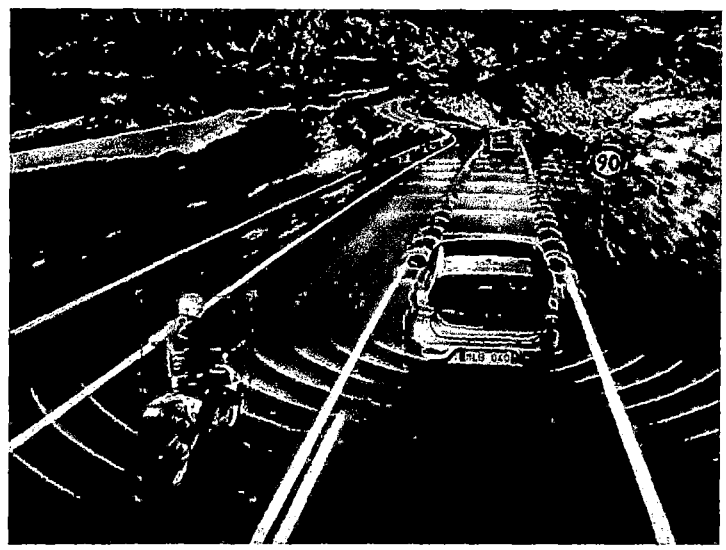
Fuente: Recuperado de http://telematicsnews.info/2012/03/30/bosch-supplies-drowsiness-detection-system-for-vw-passat-alltrack_m3301/

Último Acceso: 25 de mayo del 2013

2.1.3.4 Volvo Cars

El sistema de Volvo Cars utiliza una cámara instalada en la parte delantera del vehículo para calcular la dirección de la carretera respecto del vehículo, esto junto a los datos del volante ayudan a predecir un patrón de conducción anormal por lo que advierte al conductor con una señal acústica y también con un mensaje de alerta en el panel de instrumentos. También puede detectar señales de carretera y proximidad con otros vehículos.

Figura 13: Sistema de ayuda a la Conducción de Volvo Cars



Fuente: Recuperado de https://www.media.volvocars.com/global/enhanced/en-gb/Media/Preview.aspx?mediaid=42321
Último Acceso: 25 de mayo del 2013

2.1.3.5 Mercedes-Benz Anti-Fatigue Driver Alarm

El sistema experimental de Mercedes-Benz combina métodos intrusivos y no intrusivos, primero recurre a un sistema de visión infrarrojo que monitorea los ojos del conductor para evaluar el estado de somnolencia y al mismo tiempo mediante sensores colocados en la cabeza del conductor evalúa su actividad cerebral, con ambos resultados predice el estado de somnolencia del conductor.

2.1.3.6 Mitsubishi Motors Corporation

Esta empresa tiene un vehículo experimental para probar los diferentes desarrollos, uno de ellos evalúa el cambio involuntario de carril a través de una cámara que analiza la dirección de las líneas de carretera para deducir el estado de somnolencia del conductor.

2.1.3.7 AssistWare Technology

Esta empresa tiene un sistema que mediante una cámara analiza la carretera para calcular la posible trayectoria a seguir, según esto puede evaluar si la trayectoria actual es correcta o no, lo cual puede deberse a problemas de somnolencia o cambio involuntario de carril.

2.1.3.8 Delta Tooling

Desarrollado por la Universidad de Tokio, consiste en varios sensores ubicados en el asiento del conductor, los sensores miden las pulsaciones del conductor así como su respiración, con estas medidas pueden predecir la aparición del sueño. Según los investigadores este sistema será comercial en unos 5 años.

Figura 14: Asiento para detección de Somnolencia de Delta Tooling

Fuente: Recuperado de http://pinktentacle.com/2007/03/smart-car-seat-detects-drowsy-drivers/
Último Acceso: 25 de mayo del 2013

De los sistemas existentes mencionados anteriormente alguno de ellos son intrusivos, otros están en periodo de prueba y/o investigación, y su uso está limitado solo a ediciones de autos de fabricación reciente o vehículos para trabajos especializados, por lo cual no podrían usarse en gran parte del parque automotor disponible en nuestro país.

También podemos observar que algunos de los sistemas comerciales mencionados anteriormente utilizan técnicas de visión artificial y en particular el análisis de los ojos y el uso de la medida del PERCLOS para evaluar el estado de somnolencia, por lo cual es prueba que este método es factible de utilizar para la detección de somnolencia y de manera no intrusiva.

2.2 MARCO TEÓRICO

2.2.1 SOMNOLENCIA

Según la Real Academia Española [7], "*la somnolencia es la pesadez y torpeza de los sentidos motivadas por el sueño*", este estado es contrario a la recomendación hecha en el Manual de Manejo Defensivo [8] el cual indica que: "El conductor debe estar constantemente alerta ya que ninguna otra forma de transporte requiere tanta atención como la conducción de un vehículo automotor."

2.2.1.1 FACTORES DE SOMNOLENCIA Y SUS CONSECUENCIAS

El estado de somnolencia puede deberse a diversos factores entre los cuales podemos nombrar: Cansancio, mucho tiempo al volante, aburrimiento, consumo de alcohol y/o drogas, consumo de medicamentos que producen sueño, etc.

Según la Dirección General de Tráfico de España en una publicación sobre factores de riesgo "las alteraciones más importantes producidas por la somnolencia y que afectan a la conducción son las siguientes:

- **Incremento del tiempo de reacción:** La somnolencia aumenta sensiblemente el tiempo que tardas en reaccionar ante los estímulos en el tráfico. Por ejemplo, bajo su efecto son típicos los alcances traseros, que se producen cuando el vehículo que te precede frena y tú no eres capaz de reaccionar a tiempo para evitar la colisión.
- **Menor concentración y más distracciones:** La somnolencia hace más difícil mantener tu concentración en el tráfico. Por ello, las distracciones pueden aparecer con más facilidad. Esto sucede especialmente en entornos monótonos y en condiciones de poco tráfico.
- **Toma de decisiones más lenta y más errores:** La somnolencia hace que tardes más tiempo en procesar la información que recoges del ambiente y en reaccionar en consecuencia. Además, bajo su influencia, son más frecuentes las decisiones equivocadas, especialmente en situaciones complicadas y donde tengas que dar una respuesta rápida.

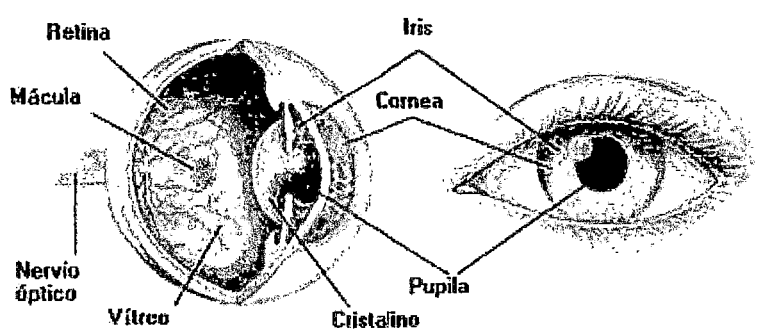
- **Alteraciones motoras:** Bajo los efectos de la somnolencia los músculos se relajan, por lo que tus movimientos serán más lentos y menos precisos. También pueden aparecer leves temblores en las manos o en otras partes del cuerpo.
- **Movimientos más automatizados:** Es importante destacar la tendencia a ejecutar los movimientos de forma automática bajo condiciones de somnolencia. Esto puede llevarte a realizar una maniobra basándote más en el hábito que en las necesidades de la situación. Por ejemplo, puedes llegar a rebasar una señal de stop, sin fijarte previamente si en ese momento venía otro vehículo.
- **Aparición de microsueños:** Los microsueños son periodos de apenas unos segundos de duración durante los que te quedas ligeramente dormido y permaneces ajeno a lo que ocurre en el tráfico. Son uno de los efectos más negativos de la somnolencia al volante y se relacionan con numerosos accidentes de extrema gravedad. El mayor problema es que estos microsueños suelen pasar completamente inadvertidos, por lo que no eres consciente de haberlos sufrido hasta que ya has salido de ellos.
- **Alteración de las funciones sensoriales:** En general, si estás somnoliento necesitarás que los estímulos sean más intensos (por ejemplo, luces más fuertes) para poder captarlos adecuadamente. Aunque el sueño afecta a todos los sentidos, repercute especialmente sobre la visión, que se deteriora considerablemente. Resulta más difícil enfocar la vista, produce visión borrosa y fatiga ocular, y favorece que se produzcan deslumbramientos.
- **Alteraciones en la percepción:** La somnolencia hace que tiendas a captar peor o de manera incorrecta las señales, las luces, los sonidos, etc. Con sueño identificas peor cualquier objeto del entorno de la vía. Finalmente, en casos de fuerte privación de sueño (48 o más horas sin dormir) puedes incluso llegar a padecer alucinaciones e ilusiones visuales.
- **Cambios en el comportamiento:** En ocasiones, con sueño al volante puedes sentirte en tensión, más nervioso e incluso más agresivo.

Además, es posible que tu comportamiento sea más arriesgado, especialmente cuando ya estés cerca de tu lugar de destino y tengas muchas ganas de llegar para dormir. También resulta curioso observar que muchos conductores con sueño tienen tendencia a ocupar el centro de la calzada o irse hacia la izquierda (quizá por el miedo a salirse por la derecha)" [9].

2.2.2 OJOS HUMANOS

"El ojo humano es un órgano que tiene una notable capacidad de adaptarse para ver objetos distantes o distinguir pequeños objetos, como también nos permite apreciar un amplio rango de colores. El funcionamiento del ojo se debe a que los rayos de luz que recibimos constantemente inciden, a través del cristalino, en la retina que se encuentra en la parte posterior del ojo; allí los rayos de luz se convierten en impulsos que viajan a través del nervio óptico hacia la corteza del cerebro relacionada con la visión, creándose las imágenes que vemos." [10]

Figura 15: Anatomía del Ojo Humano



Fuente: Morales [10] p.20

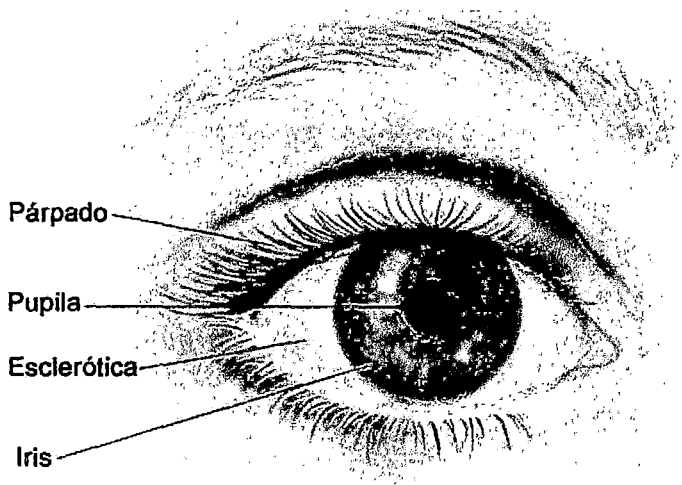
Entre las principales partes del ojo [11] se mencionan:

- Las cejas que evitan la caída del sudor en los ojos.
- Las pestañas, que impiden que el polvo entre en el ojo.
- Los párpados, que se cierran rápidamente ante cualquier roce o golpe en el ojo.
- La córnea es una membrana transparente que está en contacto con el exterior.
- El iris es la zona coloreada del ojo.

- La pupila se encuentra en el centro del iris.
- El cristalino es una lente convergente que enfoca las imágenes.
- La retina es una membrana sensible a la luz que está en el fondo del globo ocular. De ella sale el nervio óptico que da información al cerebro.

De ellas las partes externas son: El párpado, la pupila, esclerótica y el iris.

Figura 16: Partes externas del ojo



Referencia: Recopilado de
http://www.cancer.gov/espanol/pdq/tratamiento/melanomaintraocular/Patient/page1/AllPages/Print
Último Acceso: 01 de junio del 2013

2.2.2.1 EL PARPADEO O PESTAÑEO

"El parpadeo es la función de los párpados y con él se distribuye y renueva la película lagrimal. Hay dos tipos de parpadeo: El reflejo y el voluntario; ambos se llevan a cabo gracias a las funciones de los músculos palpebrales por estimulación de los pares craneales." [12]

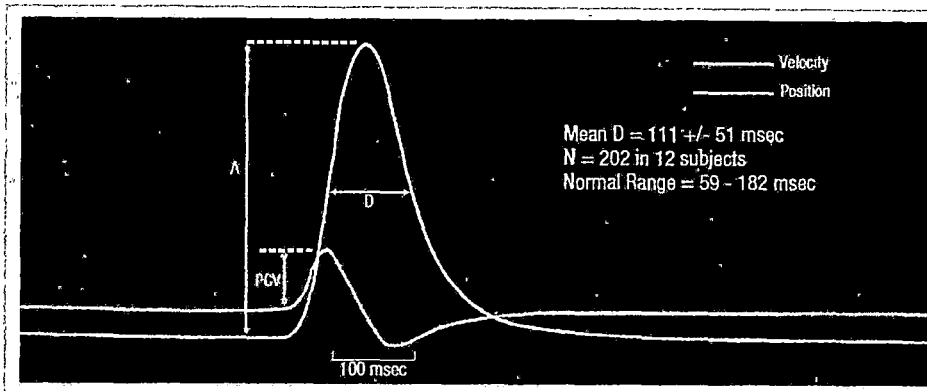
Según la Real Academia Española [7], "*el pestañeo es el movimiento rápido y repetido de los párpados.*" Por lo cual el término pestañeo o parpadeo vendrían a significar lo mismo.

- **Frecuencia del parpadeo o pestañeo en Adultos:** Según [12] en forma espontánea se parpadea de cuatro a seis veces por minuto, y de manera refleja cuando existe algún estímulo sobre las pestañas o la superficie ocular.
- **Duración del parpadeo o pestañeo:** En un estudio [13] sobre velocidad promedio de pestañeos para monitoreo de somnolencia, para el cual se trabajó con 12 personas, 7 mujeres y 5 varones,

de entre 19 y 64 años, a los cuales se les monitoreo los ojos y se extrajeron 202 pestañeos aleatorios de cada uno, con lo cual se obtuvo que en promedio la duración normal de un pestañeo reflejo es de 111+-51 msec.

Figura 17: Características de un pestañeo reflejo

The characteristics of a normal blink are shown in Fig 1 in terms of its amplitude (A), peak closing velocity (PCV) and duration. It is difficult to be sure when a blink ends and the eyelids are finally open. The duration of blinks is more accurately measured by their duration (D) at half their amplitude. The mean D for alert subjects here was 111 +/- 51 msec, and the normal range was 59-182 msec.



The amplitude (A), duration at half amplitude (D) and peak closing velocity (PCV) of a normal blink

Referencia: Extraído de Johns [13], p.2

2.2.3 CARACTERÍSTICAS VISUALES DE SOMNOLENCIA

Las personas con estado de somnolencia presentan ciertas características visibles en el rostro que pueden permitirnos reconocer estados de somnolencia, es así que, el rostro, los ojos y la boca brindan información visual para reconocer cuando un conductor presenta somnolencia.

Entre las características visibles de la somnolencia se pueden observar: "La aparición de bostezos o de un parpadeo más frecuente, o bien la caída del tono muscular de los músculos de la nuca y, con él, de la cabeza." [14]

2.2.4 PORCENTAJE DE OJOS CERRADOS (PERCLOS)

PERCLOS [15] son las siglas de Percentaje of the Time Eyelids are Closed, o en español porcentaje de ojos cerrados y es mayormente utilizado para la medición del estado de somnolencia basado en el estado de los ojos (Abiertos o Cerrados), para calcular este índice se mide el porcentaje de cierre de los ojos sobre un intervalo de tiempo.

Como el índice del PERCLOS se calcula en base al estado de los ojos, la información de los ojos es obtenida generalmente por cámaras, donde en un tiempo determinado se capturan "n" fotogramas, en los cuales algunos

corresponderán a ojos cerrados y los restantes a ojos abiertos, luego para calcular el índice PERCLOS se usa la siguiente fórmula:

$$PERCLOS = \frac{Cantidad\ de\ Fotogramas\ "cerrados"}{Cantidad\ de\ Fotogramas\ evaluados}\ x\ 100\%$$

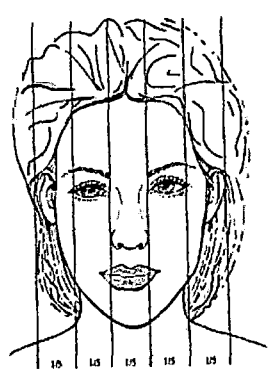
"Un video resulta de la exposición imágenes o fotogramas uno detrás de otro." [16] Es decir que los videos están formados a partir de imágenes o fotogramas, estos fotogramas son extraídos y analizados para calcular el índice PERCLOS.

La principal ventaja de usar PERCLOS como indicador de somnolencia es que puede ser usado por sistema de detección no intrusivos y en tiempo real.

2.2.5 PROPORCIONES FACIALES

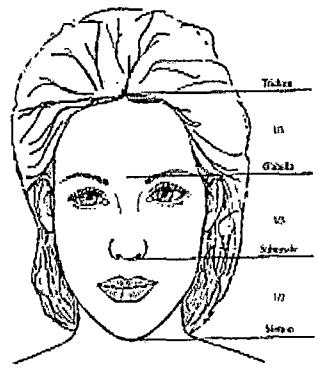
El rostro humano presenta ciertas proporciones faciales que se cumplen en la mayoría de las personas, Burgué [17] señala que el rostro humano en ancho puede dividirse en 5 partes iguales donde cada parte correspondería al ancho de los ojos, así mismo señala que en altura el rostro puede ser dividido en tres partes iguales pasando estas líneas divisorias por el mentón, la base de la nariz, y la grabela lo cual podemos observar en las siguientes figuras.

Figura 18: Anchura del rostro se divide en 5 partes iguales



Referencia: Burgué [17], p.6

Figura 19: Altura del rostro dividido en 3 partes iguales

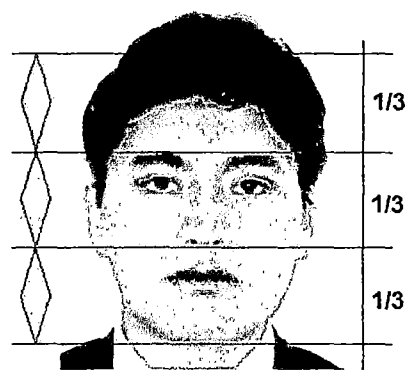


Referencia: Burgué [17], p.6

Figura 20: Ejemplo de Rostro dividido en 5 partes iguales

Referencia: Elaboración propia en base a descripción de Burgué [17]

Figura 21: Ejemplo de Rostro dividido en 3 partes iguales



Referencia: Elaboración propia en base a descripción de Burgué [17]

2.2.6 VISIÓN ARTIFICIAL O VISIÓN POR COMPUTADORA

"El término visión por computadora dentro del campo de la Inteligencia Artificial puede considerarse como el conjunto de todas aquellas técnicas y modelos que nos permitan el procesamiento, análisis y explicación de cualquier tipo de información obtenida a través de imágenes digitales. Desde sus inicios, los desarrollos de la visión por computadora han estado inspirados en el estudio del sistema visual humano, el cual sugiere la existencia de diferentes tipos de tratamiento de la información visual dependiendo de metas u objetivos específicos, es decir, la información percibida es procesada en distintas formas con base en las características particulares de la tarea a realizar; así como en psicología se estudian y desarrollan teorías sobre la percepción visual, la visión por computadora propone varias técnicas y teorías que permiten obtener una representación del mundo a partir del análisis de imágenes obtenidas desde cámaras de video.

Debido a que la información visual es una de las principales fuentes de datos del mundo real, resulta útil el proveer a una computadora digital del sentido de la vista, que junto con otros mecanismos como el aprendizaje hagan de esta una herramienta capaz de detectar y ubicar objetos en el espacio.

La meta de la visión por computadora es modelar y automatizar el proceso de reconocimiento visual, esto es, *distinguir entre objetos con importantes diferencias entre ellos*, como diferenciar un automóvil y una bicicleta en una fotografía así como separar aves en vuelo del fondo en un video, o seguir la trayectoria de objetos en imágenes aéreas." [18]

Son varias las aplicaciones de la visión artificial las cuales se pueden observar en la siguiente tabla extraída de [19].

Tabla 2: Aplicaciones de la visión artificial

Área de producción	Aplicación		
Control de calidad	Inspección de productos (papel, aluminio, acero,...)	Astronomía	Exploración del Espacio
	Identificación de piezas	Reconocimiento de caracteres	Control de cheques, inspección de textos, ...
	Etiquetados (fechas de caducidad,...)	Control de tráfico	Matrículas de coches
	Inspección de circuitos impresos		Tráfico viario
	Control de calidad de los alimentos (naranjas,...)	Meteorología	Predicción del tiempo
Robótica	Control de soldaduras	Agricultura	Interpretación de fotografías aéreas
	Guiado de robots (vehículos no tripulador)		Control de plantaciones
Biomédicas	Análisis de imágenes de microscopía (virus, células, proteínas)	Militares	Seguimiento de objetivos
	Resonancias magnéticas, tomografías, genoma humano		Vigilancia por satélites

Referencia: Platero [19], p.31

Platero [19] en su trabajo sobre visión artificial menciona que la visión artificial pretende capturar la información visual del entorno físico para extraer características relevantes visuales, utilizando procedimientos automáticos. Asimismo, dicho autor afirma que los dos pilares del sistema físico de visión artificial son:

- El sistema de formación de las imágenes y
- El sistema de procesamiento de éstas.

A continuación resumimos estos dos conceptos en base a lo descrito en el trabajo de Platero [19]:

2.2.6.1 FORMACIÓN DE IMÁGENES

La formación de imágenes es la primera etapa de la visión artificial, constituido por la captación de la imagen de interés, su iluminación y la adquisición de la señal en el computador.

La representación de una imagen en el computador es una matriz cuyos valores corresponden a la intensidad de color de cada píxel de la imagen capturada. Cuando se trabaja con imágenes a color, se obtiene una matriz

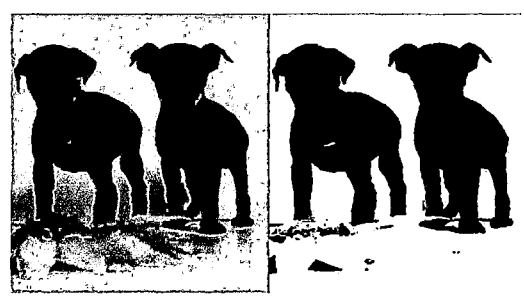
de matrices, donde las matrices internas representan los tres colores básicos reconocidos por un computador que son rojo, verde y azul. Por otro lado, cuando se trabaja con imágenes en escala de grises, el tratamiento resulta mucho más fácil ya que lo que se obtiene es una sola matriz sencilla con valores internos que oscilan entre 0 y 255,dependiendo de la intensidad del gris de cada píxel.

2.2.6.2 PROCESAMIENTO DE IMÁGENES

Una vez obtenidas las imágenes a través de la cámara se procede a procesar la misma, de manera que se pueda obtener áreas con significado relativas al estudio que se necesite realizar. Las técnicas usadas para el procesamiento de imágenes son:

- Segmentación de imágenes: Permite segmentar una imagen en áreas con significado, una técnica muy usada para esto es la binarización de imágenes.
- Morfología: Busca realzar la forma de las imágenes.
- Procesamiento de histograma: Permite mejorar la calidad de una imagen.

Figura 22: Ejemplo de Binarización de Imágenes



Referencia: Recopilado de http://multitouchvisual.blogspot.com/2010/09/tratando-imagenes.html

Ultimo Acceso: 28 de junio del 2013

2.2.7 CÁMARAS WEB

En el trabajo de Zambrano se menciona que: "Una cámara web o webcam es una pequeña cámara digital conectada a una computadora, la cual puede capturar imágenes y transmitirlas a través de Internet, ya sea a una página web u otras computadoras de forma privada.

Las webcams necesitan una computadora para transmitir las imágenes. Sin embargo, existen otras cámaras autónomas que tan sólo necesitan un

punto de acceso a la red informática, bien sea Ethernet o inalámbrico. Para diferenciarlas de la webcam o cámaras de web se las denomina net cam o cámaras de red.

También son muy utilizadas en mensajería instantánea y chat como el MSN Messenger, Yahoo! Messenger, Ekiga, Skype etc. En el caso del MSN Messenger aparece un icono indicando que la otra persona tiene webcam. Por lo general puede transmitir imágenes en vivo, pero también puede capturar imágenes o pequeños vídeos (dependiendo del programa de la webcam) que pueden ser grabados y transmitidos por internet. Este dispositivo se clasifica como de entrada, ya que por medio de él podemos transmitir imágenes hacia la computadora." [20]

Figura 23: Obtención de Imágenes por Webcam

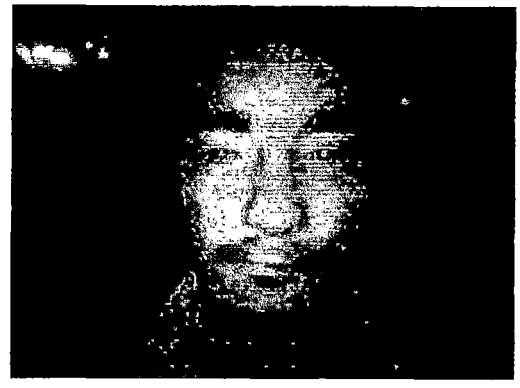
Referencia: Recopilado de http://eslonlinegrad.blogspot.com/p/online-learning-tools.html
Último Acceso: 20/04/2013

Cazares [21] menciona dos criterios que se deben tener en cuenta para la selección de una cámara, los cuales resumiremos a continuación:

- Sensibilidad: Se refiere a la cantidad real de luz visible o infrarroja necesaria para producir una imagen de calidad.

  Con cámaras web sensibles a luz infrarroja podemos observar el rostro del conductor en la oscuridad, es decir en la conducción durante la noche donde se presentan mayormente los estados de sueño.

Figura 24: Imagen capturada con webcam Infrarroja

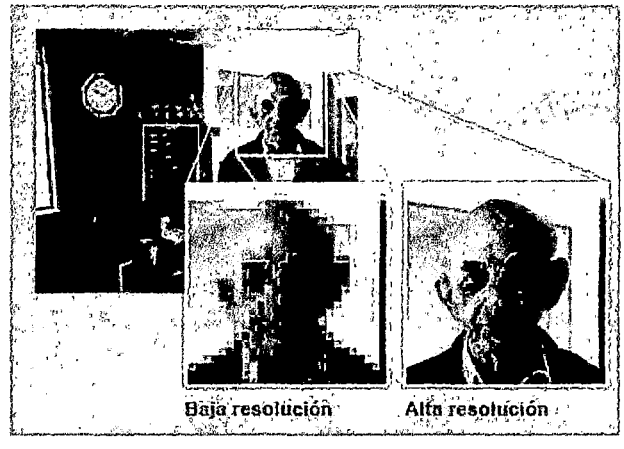


Referencia: Flores [5], p.49

- Resolución: Define la calidad de la imagen. A mayor resolución es mayor el número de pixeles de la imagen, y puede hacerse zoom en ellas sin afectar mucho la calidad de las imágenes.

  Por ejemplo en la siguiente figura se trata de hacer un zoom del rostro de una persona, ocurren dos escenarios diferentes según la resolución de la imagen. En el caso de baja resolución es difícil observar si los ojos de la persona están abiertos o cerrados, en el caso de alta resolución sucede lo contrario.

Figura 25: Ejemplo de Resolución de Imágenes



Referencia: Recopilado de http://www.hp.com/latam/pe/pyme/ipg/color/design/0909-hagalo.html

Último Acceso: 10/04/2013

Ya que las webcam suelen tener bajas resoluciones en este trabajo evitaremos el uso del zoom para procesar las imágenes por lo cual el rostro del conductor debe estar lo más cerca posible a la webcam pero sin llegar a ser intrusivo.

2.2.8 SIMULACIÓN

"La simulación implica crear un modelo que aproxima cierto aspecto de un sistema del mundo real que permite predecir cierto aspecto del comportamiento de un sistema de interés así como en diseñar y realizar experimentos y extraer conclusiones para apoyar a la toma de decisiones [22]".

2.2.8.1 CONDICIÓN DE MANEJO SIMULADO

De acuerdo al concepto anterior podemos decir que una condición de manejo simulado es representar una situación de manejo tomando solo algunos aspectos de interés necesarios para realizar experimentos para un problema dado.

Las variables que se quieren representar en el presente trabajo son la somnolencia y la luz, la velocidad del auto no será tomada en cuenta por la cual se usa un auto sin movimiento.

Para el presente trabajo esta condición de manejo simulado recoge las siguientes características:

- Una persona en el asiento del conductor con la mirada al frente y las manos al volante, cada cierto tiempo la persona debe fingir estados de somnolencia mediante pestañeos o parpadeos prolongados.
- Un auto con una webcam instalada en el volante, el auto debe estar detenido.
- Iluminación natural con luz diurna.

2.2.9 OPENCV

"OpenCV (Open Source Computer Vision Library) es una librería desarrollada por Intel. Es una colección de funciones en C y C++ que implementan algunos de los algoritmos más populares en el procesamiento digital de imágenes. Fue originalmente desarrollada para proveer una infraestructura libre y abierta donde los esfuerzos distribuidos de la

comunidad de visión por computadora pudieran ser consolidados y optimizados. Algunas áreas de aplicación son interacción computadora-humano (HCI), identificación de objetos, segmentación y reconocimiento, rastreo de movimientos, reconocimiento de rostros, reconocimiento de gestos, monitoreo, entendimiento de movimiento, biométricas, robots móviles, etc. Es principalmente una librería de alto nivel que implementa algoritmos como técnicas de calibración de cámaras, detección de características y rastreo, análisis de figuras, análisis de movimiento, reconstrucción en 3D, segmentación y reconocimiento de objetos.

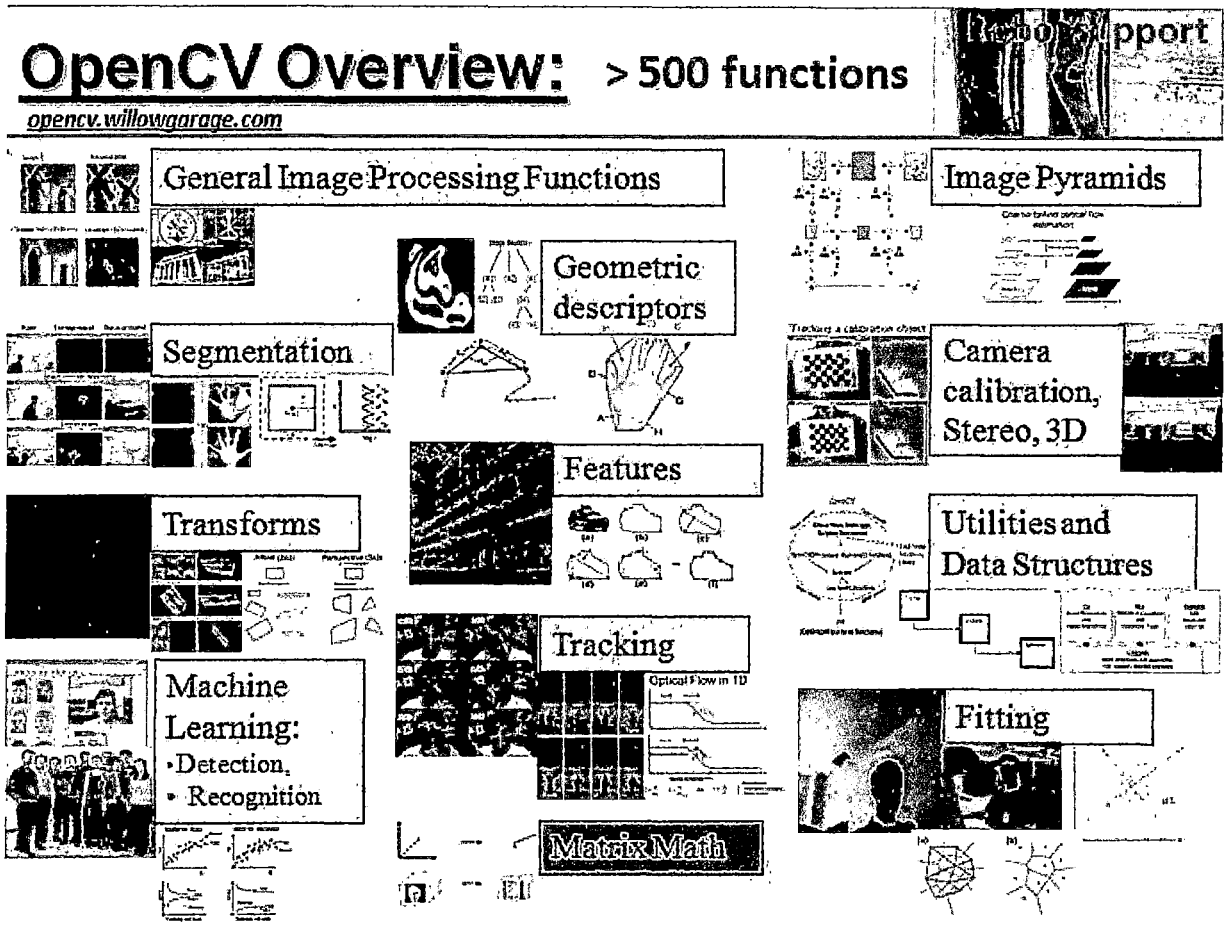
La principal característica de la librería además de su funcionalidad es el desempeño. Los algoritmos están basados en estructuras de datos dinámicas y más de la mitad de las funciones fueron optimizadas a nivel de ensamblador para tomar ventaja de la arquitectura de los procesadores Intel.

Para poder utilizar muchas de las funciones de OpenCV es necesario tener la imagen en una estructura tipo IplImage. Ésta estructura contiene la siguiente información:

- Ancho y altura de la imagen en píxeles
- La profundidad del píxel en bits. Es decir, con cuantos bits se representa un píxel. Los valores aceptados son 8,16 y 32 bits con signo y sin signo.
- El número de canales. El número de dimensiones en el espacio de color. RGB tiene 3 canales mientras que una imagen de grises tiene un solo canal.
- Origen. Indica en donde empieza la imagen, puede ser la esquina superior izquierda o la esquina inferior derecha.
- El orden de los datos. Indica si los componentes de una imagen de color vienen entrelazados o separados.
- Un apuntador a los datos de la imagen.

Las funciones que contiene OpenCV son muy variadas, existen desde operaciones sencillas como dibujar líneas y círculos hasta funciones que encuentran regiones en la imagen donde se encuentra una mano humana y reconocen el gesto que está realizando." [23]

Figura 26: Vista general de funciones de OpenCV



Referencia: Recopilado de http://opencv.willowgarage.com/wiki/

Último Acceso: 10 de abril del 2013

2.2.9.1 RECONOCIMIENTO DE PATRONES CON OPENCV

"OpenCV tiene una serie de funciones orientadas al reconocimiento de patrones, y de entre estos patrones, el más característico es el relacionado con el reconocimiento facial. Esta mecánica de reconocimiento está basada en las características tipo Haar de un objeto.

Un proceso de reconocimiento puede ser mucho más eficiente si está basado en la detección de características más significativas del tipo de objeto que debe ser detectado. Este es el caso de las características tipo Haar, que reseñan la existencia de orientación del contraste entre regiones dentro de una imagen. Un conjunto de estas características puede ser utilizado para determinar el contraste exhibido por el rostro humano y su interrelación espacial". [24]

2.2.9.2 DETECCIÓN DE OBJETOS USANDO CLASIFICADORES CON OPENCV

"La detección de objetos de OpenCV fue propuesta por Paul Viola. Primero un clasificador es entrenado con unos pocos centenares de imágenes de ejemplo de un objeto en particular (Por ejemplo una colección de rostros o una colección de ojos), generando lo que se conoce como ejemplos positivos; los cuales son escalados todos al mismo tamaño, así como ejemplos negativos - imágenes arbitrarias del mismo tamaño.

Una vez que el clasificador ha sido entrenado, puede ser aplicado a posteriori a una región de interés sobre una imagen de entrada. El clasificador devuelve un "1" si considera que hay algún parecido entre el hipotético objeto del área de interés (Detecto una cara o detecto un ojo por ejemplo) y un "0" en caso contrario. Para buscar el objeto por toda la imagen, se puede mover la ventana de búsqueda a lo largo de la imagen y comprobar cada área usando el clasificador.

La detección de objetos basados en un clasificador no es cien por ciento fiable pero con un buen entrenamiento del clasificador puede lograr menor ocurrencias de *falsos positivos*.

Un falso positivo ocurre cuando el clasificador indica que una región de interés coincide con el objeto buscado cuando en realidad no lo es, abajo podemos observar una imagen con un *falso positivo* en la detección de rostros." [24]

Figura 27: Ejemplo de FALSO POSITIVO en detección de rostros

Referencia: Recopilado de http://note.sonots.com/SciSoftware/haartraining.html
Último Acceso: 23 de mayo del 2013

2.2.10 DETECCIÓN DE ROSTROS

"La detección de rostros es actualmente utilizada en diversos tipos de aplicaciones. Por ejemplo, muchas cámaras fotográficas digitales, antes de tomar una fotografía, detectan los rostros presentes en la escena para mejorar el enfoque y la nitidez. También se utiliza la detección de rostros en aplicaciones de seguridad y seguimiento de personas, entre otras. En un sistema reconocedor de rostros es imprescindible, en primera instancia, detectar los rostros en una imagen para luego poder determinar la identidad de cada uno.

La detección de rostros determina la ubicación, tamaño y orientación de las caras que se encuentran presentes en una imagen. Cuando hablamos de ubicación, nos referimos a la posición de las caras en relación a la imagen. Además las caras pueden presentarse con diversos tamaños, no sólo por las diferentes contexturas físicas de diferentes personas, sino también por la variedad de distancias de los rostros respecto a la cámara al momento de tomar la fotografía.

Existen diversos métodos para la detección de rostros. Sin embargo, muchos de ellos no son aplicables cuando se busca hacerlo en tiempo real. El método propuesto por Viola y Jones fue el primero en ofrecer detección de rostros robusta y en tiempo real." [25]

2.2.11 DETECCIÓN DE ROSTROS CON OPENCV

A continuación resumiremos el algoritmo usado por la librería OpenCV para la detección de rostros para lo cual resumiremos parte de la información presentada por Roldan [26]en su tesis Visión por Computador en IPhone4.

Según Roldan, OpenCV contempla un método de detección de objetos basado en el algoritmo de Viola Jones. Este algoritmo consigue unos tiempos de detección tan bajos que puede ser usado incluso en sistemas de tiempo real. Para conseguir tal rendimiento, este algoritmo usa una representación alternativa de la imagen que facilita los cálculos necesarios para la detección, junto a unos clasificadores aplicados en cascada y basados en features o características.

2.2.11.1 REPRESENTACIÓN DE LA IMAGEN

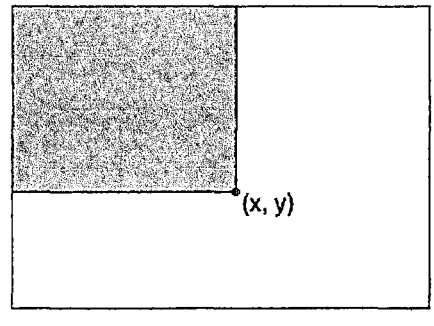
Para obtener el alto rendimiento que el algoritmo de Viola Jones alcanza, se utiliza una representación alternativa de la imagen conocida como imagen integral, que permite realizar complejos cálculos de manera muy eficiente.

En este formato cada píxel de la imagen contiene el resultado de la suma del valor de los píxeles por encima y a la izquierda de dicho píxel en la imagen.

Entonces para cada pixel en la imagen:

$$P_{(x,y)} = \sum_{i=0;\, j=0}^{x-1;\, y-1} P_{(i,j)}$$

Figura 28: Representación de la imagen integral



Referencia: Roldan [26], p.9

A modo de ejemplo, en el caso trivial en el que todos los píxeles de la imagen tuvieran valor 1, la imagen integral equivalente sería la siguiente.

Figura 29: Ejemplo de Imagen Integral

1	1	1
1	1	1
1	1	1

Imagen Original

1	2	3
2	4	6
3	6	9

Imagen Integral

Referencia: Roldan [26], p.10

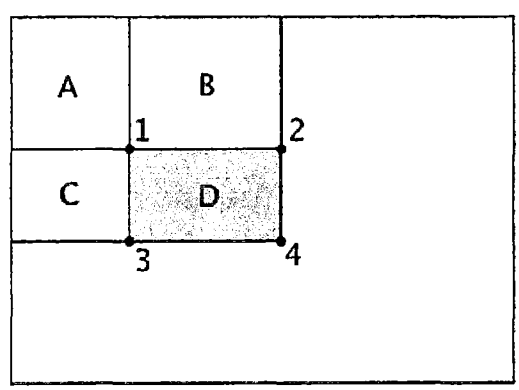
Esta representación de la imagen permite realizar operaciones de manera sencilla, rápida y eficiente. Por ejemplo, la suma de cualquier rectángulo puede ser calculada a partir de 4 referencias a valores de la imagen.

En la imagen posterior si necesitamos calcular el valor para el rectángulo D:

- La suma de los píxeles del rectángulo A es el valor de la imagen integral en el punto 1.
- El valor del punto 2 se corresponderá con A + B.
- El valor del punto 3 se corresponde con la suma de las regiones A + C.
- El valor del punto 4 es A + B + C + D.

Este tipo de cálculo es de gran utilidad para calcular el valor de las características o features en los que el algoritmo de Viola Jones está basado.

Figura 30: Ejemplo Cálculo en imágenes integrales



Referencia: Roldan [26], p.10

2.2.11.2 CARACTERÍSTICAS O FEATURES

El algoritmo de detección clasifica las imágenes candidatas en función del valor de determinadas features. Se usan 3 tipos de features: De 2 rectángulos, de 3 rectángulos y de 4 rectángulos.

Figura 31: Tipos de Features

Feature de 2 rectángulos *Feature de 3 rectángulos* *Feature de 4 rectángulos*

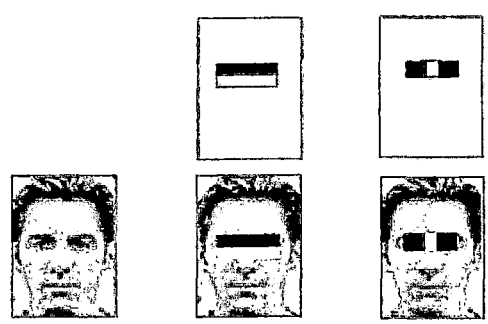
Referencia: Roldan [26], p.11

El valor de cada feature se calcula como la resta de la suma de los píxeles de los rectángulos grises menos la suma de los píxeles de los rectángulos blancos. Estos cálculos, tal y como se ha visto en el apartado anterior, resultan muy efectivos y fáciles de realizar cuando la imagen se encuentra representada como una imagen integral.

2.2.11.3 CLASIFICADORES

El principio sobre el que se fundamente este algoritmo consiste en que el valor de algunas de estas features será significativamente más alto cuando se encuentren situadas sobre una determinada región del patrón que se desea detectar.

Figura 32: Ejemplo de Features usados en detección de rostros



Referencia: Roldan [26], p.11

Puesto que el número de posibles features presentes en una imagen llega a ser incluso superior al número de píxeles de la misma, se usa una modificación del algoritmo de aprendizaje AdaBoost para construir una serie de clasificadores a partir de la selección de las features que mayor información aportan sobre la identificación de un determinado patrón.

AdaBoost considera cada una de las features existentes como un posible clasificador débil - Un clasificador débil es aquel cuya tasa de aciertos en la clasificación es tan solo superior a la mitad de los casos - y estudia los resultados que obtiene cada uno de ellos en un conjunto de imágenes de

entrenamiento de las que ya se conocen, a priori, las dimensiones y coordenadas en las que aparece el patrón a detectar.

De esta manera, uno de estos clasificadores débiles calculará el valor de una determinada feature en una determinada posición de la imagen, si su valor supera un determinado valor umbral, el clasificador considerará la imagen sobre la que ha sido aplicado como un positivo, y si no lo supera, como un negativo.

A partir de la información de estos resultados, AdaBoost construye unos clasificadores fuertes mediante la combinación ponderada de varios clasificadores débiles, previamente seleccionados en función de los resultados obtenidos por cada uno de ellos en el conjunto de imágenes de entrenamiento.

2.2.11.4 EL DETECTOR DE CLASIFICADORES EN CASCADA

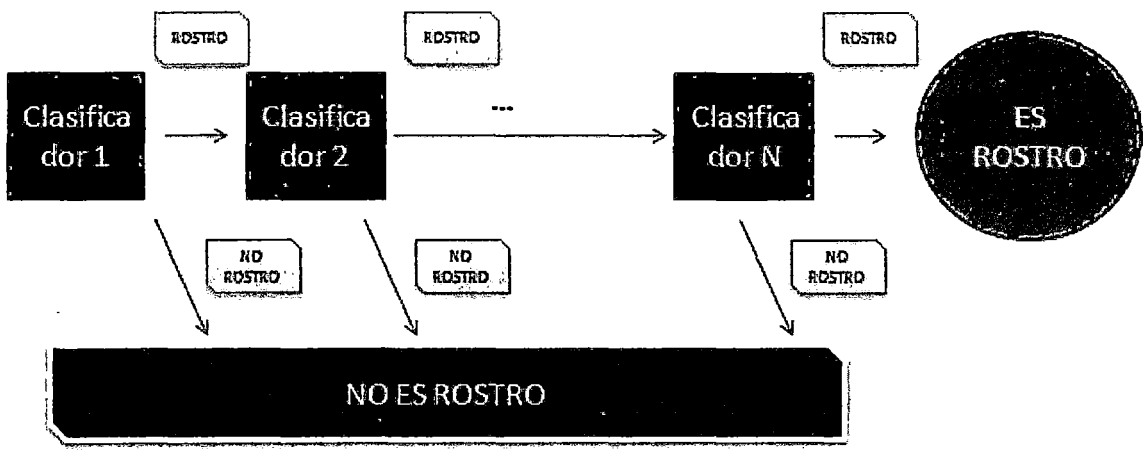
El principio básico del algoritmo de detección facial de Viola y Jones consiste en escanear el detector muchas veces a través de una misma imagen, en diferentes posiciones y a distintas escalas. Incluso si una imagen contiene muchas caras está claro que la mayoría de las sub-ventanas que se escaneen no contendrán ningún rostro. Esto lleva a una nueva manera de ver el problema: *En vez de encontrar rostros, el algoritmo debería descartar "no-rostros"*.

La idea subyacente se basa en que es más fácil descartar una imagen que no *contenga* un rostro que encontrar un rostro en una imagen. El tiempo de clasificación será constante sin importarnos la entrada del clasificador, ya que el clasificador tiene que evaluar todas las características que lo forman. Aumentar la velocidad de clasificación generalmente implica que el error de clasificación aumentará inevitablemente, ya que para disminuir el tiempo de clasificación se debería disminuir el número de clasificadores simples que se utilizan. Para evitar esto Viola y Jones proponen un método para reducir el tiempo de clasificación manteniendo los requerimientos de rendimiento del clasificador.

Este método *consiste* en el uso de una cascada de clasificadores fuertes. El trabajo en cada etapa del clasificador consiste en determinar si la sub-ventana que se analiza es definitivamente un no-rostro o podría ser un

rostro. Cuando una sub-ventana es clasificada como no-rostro en alguna de las etapas del detector, se descarta inmediatamente. En caso contrario, si se clasifica como un posible rostro, pasa a la siguiente etapa del clasificador. Se identificará una sub-ventana como contenedora de un rostro si y sólo si pasa a través de todas las etapas del detector de forma positiva.

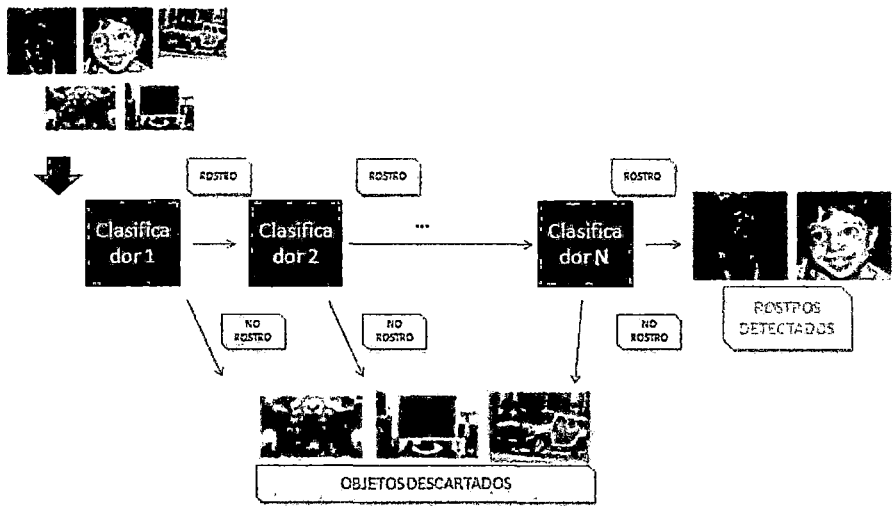
Figura 33: Estructura de clasificación en cascada



Referencia: Elaboración Propia en base a descripción del autor

Si se utilizase un clasificador que tuviese un único estado, normalmente habría que aceptar los falsos negativos para reducir la tasa de falsos positivos. Sin embargo, para las primeras etapas del clasificador en cascada se acepta una alta tasa de falsos positivos esperando que las etapas posteriores puedan encargarse de reducir esta tasa mediante clasificadores más especializados. Con esto se pretende también reducir la tasa de falsos negativos en el clasificador final, ya que una sub-ventana será clasificada como rostro sólo en el caso de que haya pasado por todas las etapas del clasificador.

Figura 34: Ejemplo de Clasificación en cascada para detectar Rostros



Referencia: Elaboración propia en base a descripción del autor e imágenes de la web

2.2.11.5 MÉTODO DETECTHAARCASCADE DE OPENCV

Este método de la librería OpenCV usa el clasificador para buscar rostros u ojos en una imagen, tiene varios parámetros que resultan importantes para minimizar el error en la detección. Estos parámetros están descritos en un tutorial de fewTutorials [27] el cual traduciremos y describiremos a continuación:

El método DetectHaarCascade del OpenCV tiene los siguientes parámetros:

```
DetectHaarCascade(HaarCascadehaarObj,
    doublescaleFactor,intnNeighbors,
    HAAR_DETECTION_TYPEflag,SizeminSize).
```

- **Haar Cascade**: Es un archivo XML el cual es el clasificador de rostros, este archivo debió ser entrenado para detectar el objeto que deseamos encontrar, en nuestro caso rostros. Se puede entrenar un clasificador en cascada para cualquier objeto sin embargo para objetos comunes como rostros y ojos es recomendable usar los disponibles en la librería OpenCV por la calidad de su entrenamiento.
- **Factor de Escala:** Para buscar rostros o cualquier otro objeto OpenCV toma varias secciones pequeñas de la imagen y verifica si esta es el objeto buscado, luego toma una sección un poco más grande para la siguiente búsqueda, el factor de escala permite

especificar el porcentaje de incremento de estas secciones, mientras mayor sea el factor de escala el detector será más rápido pero puede fallar al no detectar objetos que pudieron estar en secciones obviadas. Por defecto OpenCV considera un factor de escala de 1.1 es decir que la siguiente sección de imagen será un 10% más grande. Este parámetro debe tener valores de 1.1, 1.2, 1.3 o 1.4.

- **Número de vecinos Mínimos:** Al buscar el rostro o el objeto deseado, OpenCV clasificará varias secciones como el objeto buscado, sin embargo podrían serlo o no, además estas secciones muchas veces pertenecen en realidad al mismo objeto entonces este parámetro nos permite establecer el número de vecinos mínimos que el objeto debe tener para así rechazar o no la sección dependiendo si tiene o no el mínimo de vecinos requeridos. Este parámetro varía de 0 a 4. En caso de que se use 0 como número de vecinos mínimos cualquier sección clasificada como el objeto buscado será aceptado como tal sin importar el número de vecinos que tenga.

Figura 35: Variación en el número de vecinos. Izquierda 4, derecha 0.

Referencia: Recopilado de http://fewtutorials.bravesites.com/entries/emgu-cv-c/level-3c---how-to-improve-face-detection

Último Acceso: 15 de febrero del 2013

- **Poda de Canny:** Este parámetro solo acepta dos valores 0 y *DO_CANNY_PRUNING*, si la opción de Poda de Fanny es seleccionada entonces para buscar rostros u otro objeto deseado

OpenCV descarta regiones de la imagen que probablemente no contengan un rostro, esto reduce trabajo computacional, para esto las regiones son procesadas con la ayuda de detección de bordes de Canny antes de intentar la detección de rostros. Aunque hacer esto puede aumentar la velocidad de búsqueda de imágenes también podrían perderse algunos rostros.

- **Escala Mínima de Detección:** Este parámetro indica el tamaño del rostro más pequeño para empezar a buscar en la imagen, su valor por defecto puede ser encontrado en el archivo XML del clasificador, por ejemplo en el clasificador que usamos es 20x20 (<size>20 20</size>). Para minimizar el número de búsquedas de rostros u otros objetos este valor debe ser lo más pequeño posible dependiendo de la imagen y del objeto buscado.

2.2.11.6 DETECCIÓN DE OJOS CON OPENCV

Forma parte de la detección de rostros, sobre todo en los enfoques basados en los rasgos faciales, para la detección de ojos se usan los patrones referencia de los ojos, el patrón de ojo se compone de los ojos centrados con el centro del iris del usuario. Utilizando los patrones de referencia de los ojos, se busca en la imagen la localización de los ojos en el rostro.

2.2.12 OTROS ALGORITMOS PARA LA DETECCIÓN DE OJOS

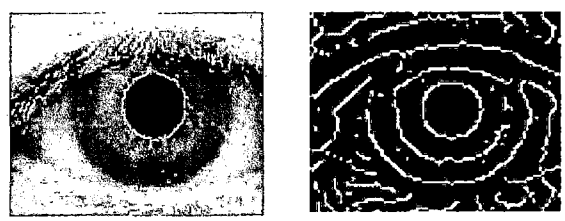
2.2.12.1 DETECCIÓN DE CÍRCULOS DE HOUGH

Gonzáles en su tesis sobre reconocimiento de Iris [28] nos da una descripción sobre el algoritmo de detección de círculos de Hough donde afirma que: "La transformada de Hough es un algoritmo estándar en la visión por computador, usado para determinar los parámetros espaciales de objetos de diversa forma que estén incluidos dentro de una imagen. Esta transformada se puede utilizar para detectar automáticamente las coordenadas y el radio del iris y de la pupila."

Este algoritmo puede ser usado para detectar el iris y la pupila cuando el ojo está abierto, si un iris no es detectado entonces se presume que el ojo está cerrado.

El proceso básico del detector de Hough empieza partiendo de la imagen completa del ojo, sobre la que se aplica inicialmente algún algoritmo de detección de contornos, como por ejemplo el detector de bordes de Canny. En la siguiente figura se muestra un ejemplo de la capacidad de este detector, incluido en la librería OpenCV:

Figura 36: Resultado de la detección de bordes de Canny



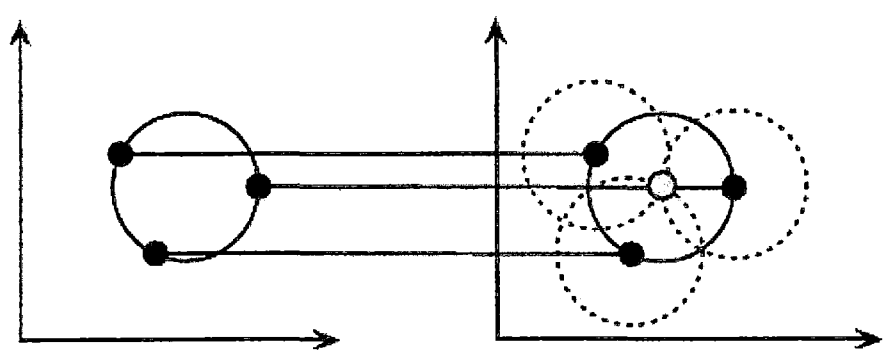
Referencia: Gonzáles [28], p31.

A partir de la imagen de contornos, se procede a crear el llamado espacio de Hough, que contiene círculos que pasan sobre cada punto de contorno. Estos círculos se definen con la ecuación general:

$$x^2 + y^2 - r^2 = 0$$

Siendo x,y las coordenadas del centro del círculo, y r el radio. Una vez creados todos los círculos posibles, el máximo dentro del espacio de Hough proporciona el círculo mejor definido.

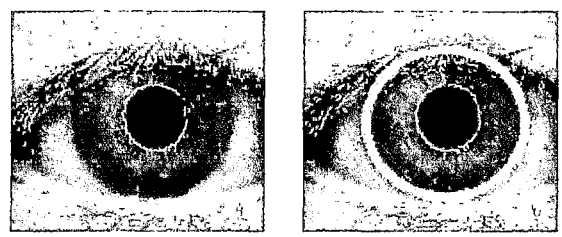
Figura 37: Algoritmo de Hough: Cada punto del contorno de la imagen sirve para trazar un círculo de Hough



Referencia: Gonzáles [28], p17.

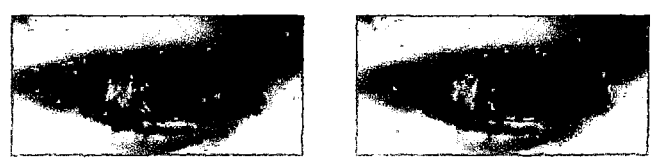
Al tratar de trabajar con el algoritmo de reconocimiento de Iris de Hough para la detección de ojos se obtuvo buenos resultados con imágenes de alta resolución extraídas de la base de datos CASIA, sin embargo al probarla con imágenes extraídas de la webcam (Figura 39) la detección de círculos fue nula por lo cual este algoritmo no fue usado para el presente trabajo.

Figura 38: Detección de círculos con imagen de la base de datos CASIA



Referencia: Elaboración propia

Figura 39: Detección de círculo con imagen de la webcam



Referencia: Elaboración Propia

Como se ha visto hay tres técnicas principales utilizadas para analizar y reconocer la somnolencia del conductor, métodos basados en patrones de conducción, análisis de los ojos y expresiones faciales utilizando visión por computador y análisis de las señales fisiológicas. Cada una de ellas abarca distintas metodologías para abordar el problema y han servido de base para el diseño de distintos sistemas de ayuda en la conducción. Las industrias vinculadas al sector automotriz también están dirigiendo su interés a la construcción de estos sistemas, sin embargo, tienen costos elevados que quizás no se puedan asumir en nuestro país. Esto nos motiva a construir un modelo de reconocimiento de somnolencia que pueda ser factible de usar en nuestro medio, los sistemas de análisis de ojos y expresiones faciales son sistemas no intrusivos y con posibilidad de aplicación con herramientas baratas como lo es el uso de una webcam, preferimos este método

para nuestro trabajo el cual nos servirá además para mostrar que el uso de la tecnología puede contribuir en la reducción de los accidentes de tránsito producto de los errores humanos, tales como los producidos por somnolencia del conductor. También se ha visto que mediante la visión artificial y usando los métodos de clasificación de Viola y Jones ya implementados por la librería OpenCV nos puede servir para detección de rostro y ojos lo cual sirve de base al trabajo realizado en este proyecto, para medir el estado de somnolencia basados en el estado de los ojos podemos usar la medida del índice PERCLOS.

CAPÍTULO III

RECONOCIMIENTO DE SOMNOLENCIA MEDIANTE LA DETECCIÓN DE OBJETOS

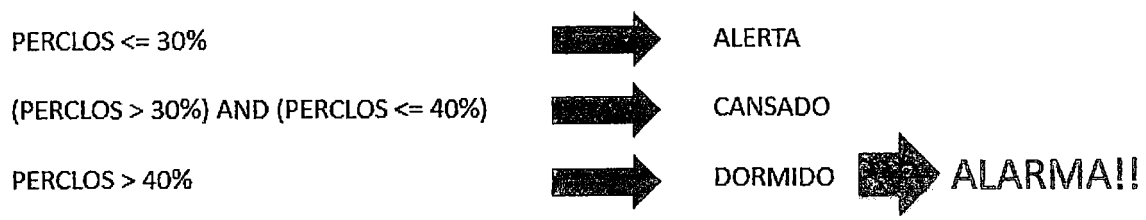
Estudiadas las herramientas y definiciones del capítulo anterior es posible, mediante la visión computacional y particularmente mediante el análisis de imágenes, la construcción de una herramienta de reconocimiento de somnolencia usando la librería OpenCV y basándonos en las características propias del rostro humano y de los estados de somnolencia utilizando la medida del PERCLOS.

3.1 DESCRIPCIÓN DEL ALGORITMO

Para la detección de somnolencia nos basaremos en la medida del PERCLOS ya descrita en el segundo capítulo, podemos medir el estado de somnolencia de una persona calculando el porcentaje de ojos cerrados en los 10 últimos fotogramas. Se irá almacenando el estado de los ojos en los últimos 10 fotogramas, se puede clasificar al conductor en tres tipos de estados dependiendo su estado de somnolencia.

- Alerta o despierto
- Cansado
- Dormido

Consideraremos los siguientes porcentajes de PERCLOS para evaluar el estado de somnolencia del conductor.



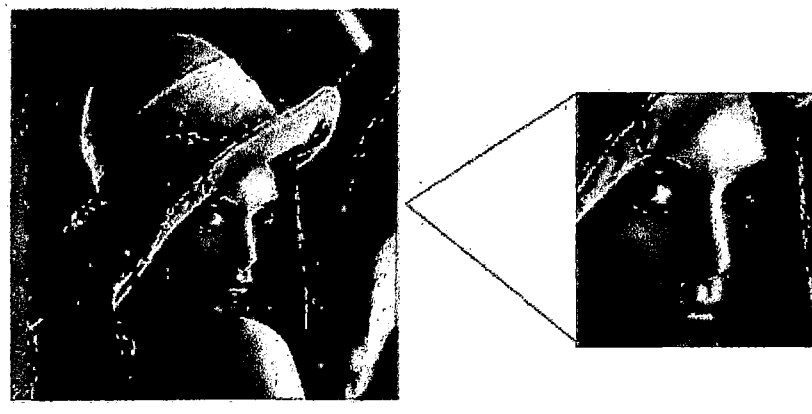
En nuestro algoritmo, se obtiene la primera imagen a través de la webcam. Luego usamos el clasificador en cascada *haarcascade_frontalface_alt_tree.xml,* el cual es

un clasificador ya entrenado para detectar rostros, con el cual OpenCV busca rostros en la imagen.

Si un rostro no es detectado entonces debemos considerar que en la imagen actual no se encontró ningún ojo abierto lo cual es lo mismo a decir que el ojo está cerrado (es mejor que la herramienta de una falsa alarma de somnolencia a que no de la alarma puesto que con el clasificador los rostros no son detectados cuando la cara está muy inclinada cosa que sucede por ejemplo en los cabeceos de sueño).

En caso de que un rostro sea detectado se extrae esta región de interés para proceder a buscar ojos abiertos en esta sub-imagen.

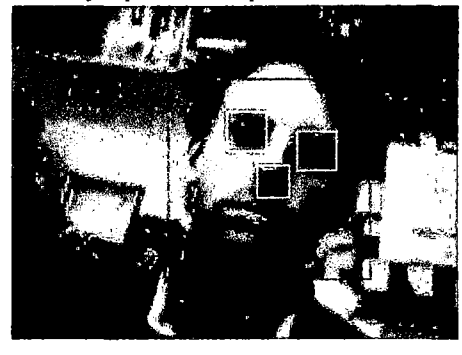
Figura 40: Ejemplo de Región de Interés para buscar los ojos



Referencia: Elaboración propia con imagen recopilada de http://www.johnloomis.org/eop513/notes/lenna/html/lenna.html

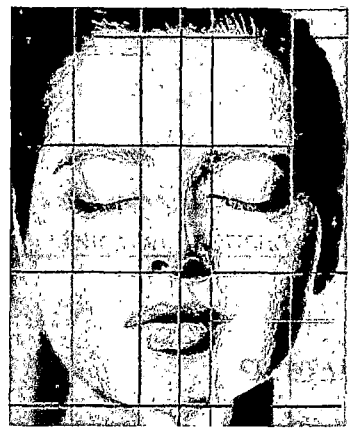
Sin embargo basándonos en características de rostros y para disminuir la ocurrencia de FALSOS POSITIVOS de ojos abiertos o cerrados podemos reducir aún más esta Región de Interés enfocándonos solo en la parte superior del rostro para buscar los ojos.

Figura 41: Ejemplo de Falso positivo al buscar un ojo



Referencia: Elaboración propia en base a imagen obtenida de la web

Figura 42: Proporciones del rostro humano

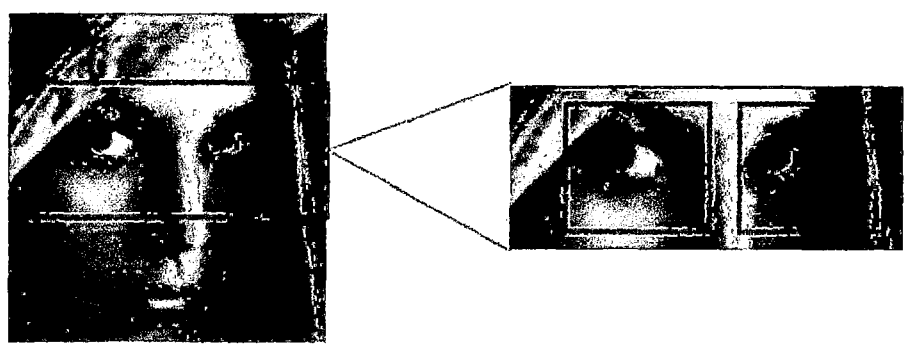


Referencia: Recopilado de http://www.rinoplastia.eu/105_la_nariz_ideal.htm

Último Acceso: 01 de junio del 2013

Para la obtención de la nueva Región de Interés a partir del rostro, obviamos la mitad inferior del rostro y la parte de la frente, de esta manera reducimos la probabilidad de falsos positivos en otras partes del rostro y nos enfocamos solamente en buscar ojos abiertos en una Región de Interés más relevante, para esto nos basaremos en las proporciones del rostro, además al reducir la Región de Interés se reduce el proceso computacional.

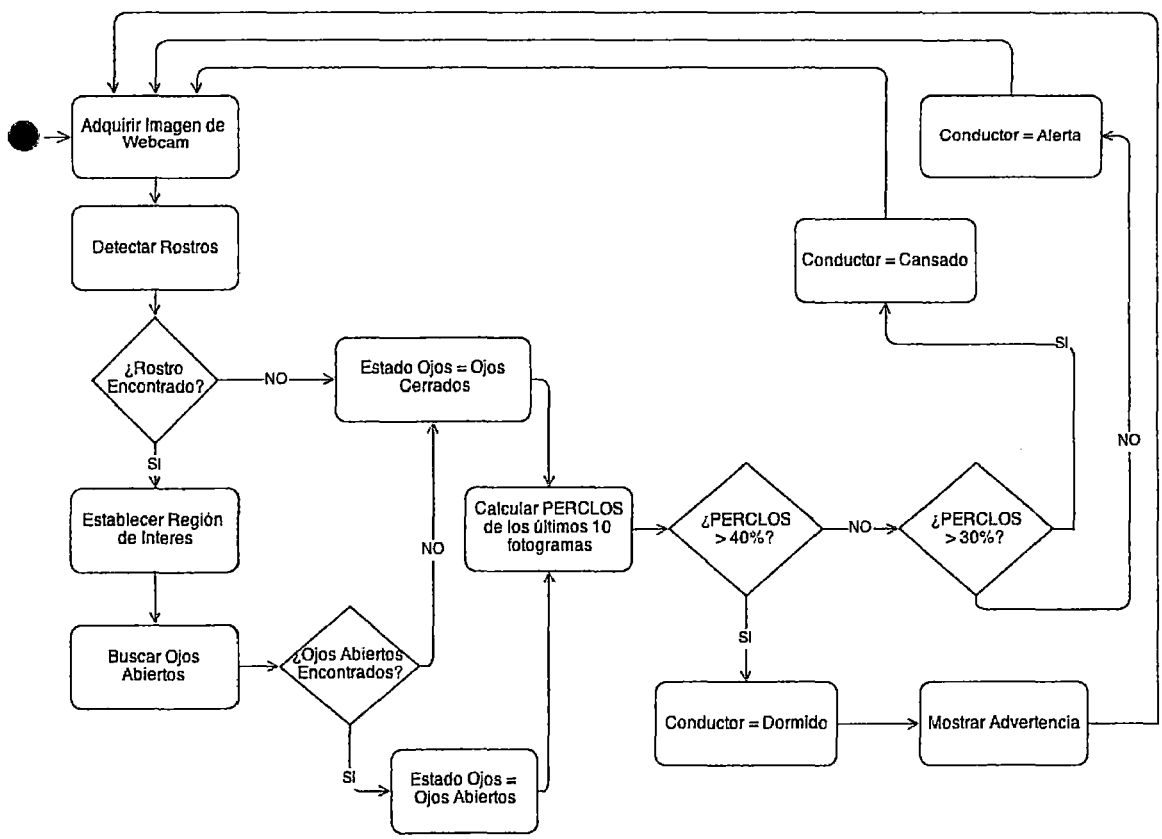
Figura 43: Región de Interés para la búsqueda de Ojos



Referencia: Elaboración propia con imagen recopilada de
http://www.johnloomis.org/eop513/notes/lenna/html/lenna.html

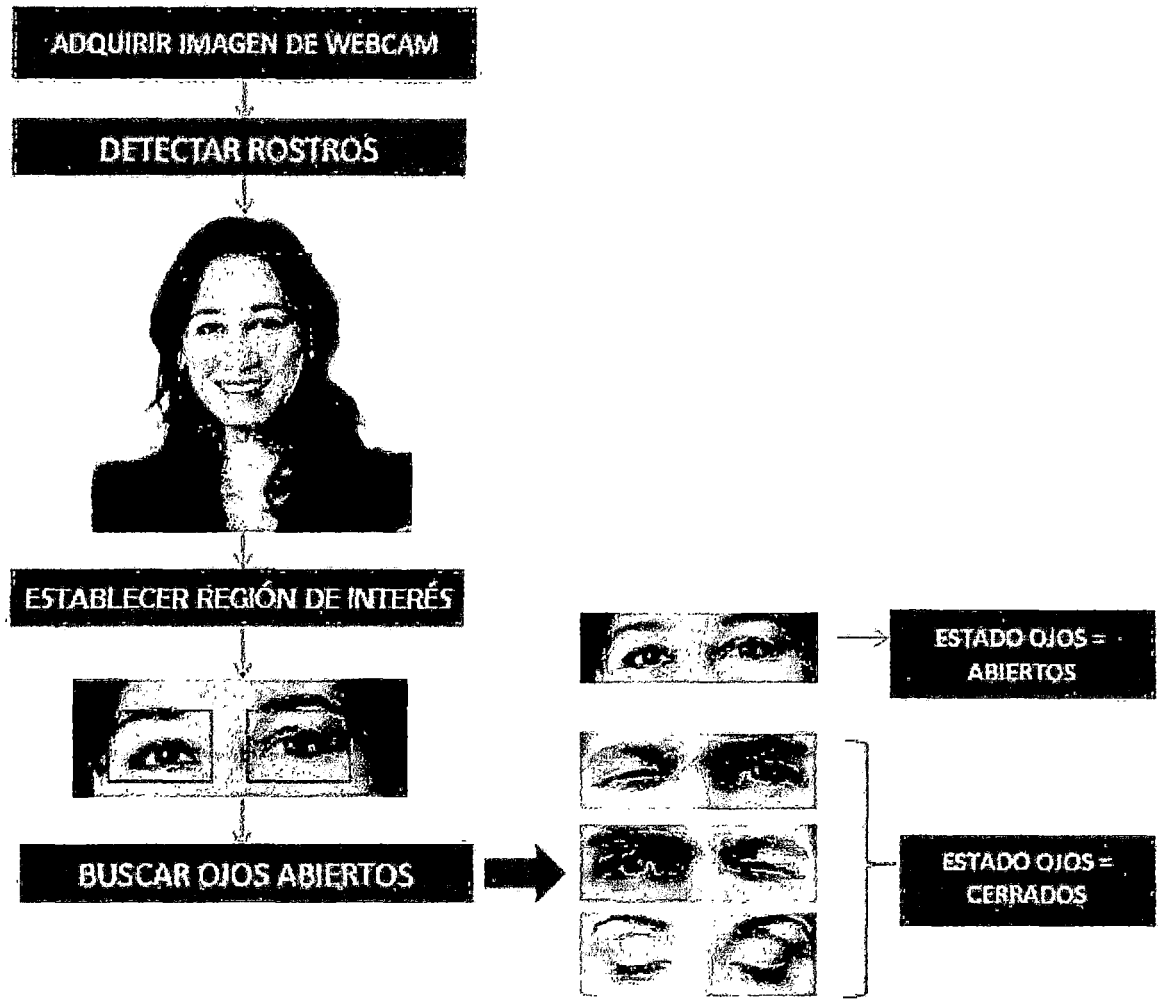
Obtenida esta última región de interés buscamos si están los ojos abiertos para lo cual usamos el clasificador en cascada *haarcascade_eye_2.xml*, este clasificador está entrenado para buscar ojos. Si no encontramos 2 ojos abiertos, uno en el lado izquierdo del rostro y otro en el lado derecho del rostro, se considera los ojos como cerrados caso contrario consideramos los ojos como abiertos.

Figura 44: Diagrama de actividades del Algoritmo propuesto



Referencia: Elaboración propia

Figura 45: Idea del Algoritmo



Referencia: Elaboración propia en base a imagen obtenida de la web

3.1.1 ADQUIRIR IMAGEN DE WEBCAM

La clase *CCamara* nos permite capturar imágenes obtenidas de una webcam, para esto utiliza la clase *Capture* de la librería OpenCV, se han definido tres métodos principales.

- IniciarCapturaCamara
- CapturarImagenCamara
- DesconectarCamara

3.1.1.1 IniciarCapturaCamara

Este método inicializa a_*CapturaCamara* el cual es un objeto de la clase Capture de OpenCV, se le pasa una variable numérica *a_CamaraIndex* para indicar el índice de la Webcam de la que deseamos obtener las imágenes, por defecto será 0 y dependerá de cuantas cámaras se tengan instaladas en el computador. El método retorna true o false dependiendo si se pudo o no inicializar *a_CapturaCamara*. Este método también puede ser usado para usar un archivo de video como medio de entrada.

3.1.1.2 CapturarImagenCamara

Este método captura una imagen de la webcam o del archivo de video y la devuelve, para esto hace uso del método *QueryFrame* de la clase *Capture* de la librería OpenCV, el cual toma una imagen de la webcam o de un archivo de video, lo descomprime y la devuelve. Si no se inicializo *a_CapturaCamara* devuelve nulo.Este método es el que nos permitirá capturar las imágenes de la webcam para poder procesarlas en nuestra herramienta.

3.1.1.3 DesconectarCamara

Este método libera los recursos usados por a_CapturaCamara y por lo tanto detiene la obtención de imágenes de la webcam.

3.1.2 DETECTAR ROSTROS

Una vez obtenida la imagen de la webcam el siguiente paso es detectar el rostro del conductor, para esto hacemos uso de la clase *CDetectorRostro* el cual también hace uso de elementos de la librería OpenCV, se definen dos métodos principales.

- CargarClasificadorRostros
- DetectarRostros

3.1.2.1 CargarClasificadorRostros

Este método nos permite establecer el archivo del clasificador que usaremos para detectar los rostros en la imagen. Utilizamos el clasificador *haarcascade_frontalface_alt_tree.xml* construido por el Prof. Dr. Rainer Lienhart de Multimedia Computing Lab.

3.1.2.2 DetectarRostros

Este método nos permite detectar un rostro en una imagen usando el clasificador anteriormente cargado, hace uso del método *DetectHaarCascade* de la librería OpenCV. Devuelve un arreglo de *Bitmaps* si encuentra algún rostro caso contrario retornara *null.*

El siguiente *algoritmo* describe todo el proceso de la detección de rostros.

Algoritmo 1: Detecta rostros en una imagen, y actualiza los vectores X e Y con los valores de la posición de los rostros hallados en la imagen.

Variables de Entrada:

I: Una imagen capturada de un video o de la webcam

X: Un vector de reales, almacena la componente X de la posición en I de los rostros detectados

Y: Un vector de reales, almacena la componente Y de la posición en I de los rostros detectados

Variables de Salida:

S: Un vector de imágenes en donde se almacenarán los rostros detectados en la imagen

---

```
DetectarRostros(I,X,Y)
01: R ←DetectHaarCascade(I)//Detecta los rostros con ayuda del clasificador de rostros
02: si longitud(R)> 0 entonces
03:    para  k← 1 hasta longitud(R) hacer
04:       S[k] ←Imagen(ancho(R[k]), alto(R[k])) //Extrae la imagen del rostro detectado
05:       X[k] ← R[k].X  //Guarda la posición X en la imagen del rostro detectado
06:       Y[k] ← R[k].Y  //Guarda la posición Y en la imagen del rostro detectado
07:    fin para
08:    retornar S
09: caso contrario
10:    retornar nulo
11: fin si
```

Figura 46: Resultado del Algoritmo 1

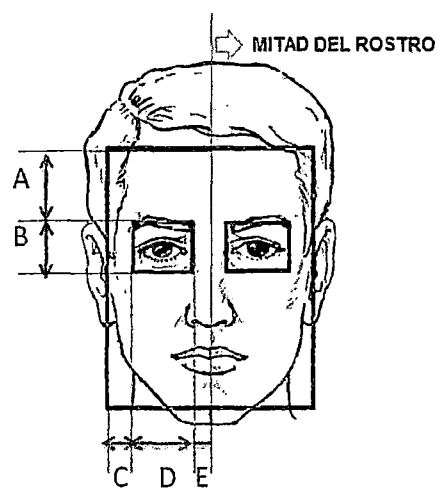
Referencia: Elaboración propia

3.1.3 ESTABLECER REGIÓN DE INTERÉS

Una vez localizado el rostro en la imagen el siguiente paso es encontrar los ojos sin embargo para eliminar el mayor número de FALSOS POSITIVOS nos basamos en las características propias del rostro humano y tomamos dos regiones de interés para buscar los ojos en la imagen del rostro. En la siguiente figura observamos el cálculo de la Región de Interés para el ojo del lado izquierdo, para la Región de Interés del lado derecho aplicamos las mismas distancias pero invertidas en el caso de C, D y E.

Como no todos los rostros detectados tendrán el mismo tamaño los valores de A, B, C, D y E deben estar expresados como porcentajes, donde los valores de A,B,C y E serán 30%, 28%, 5% y 3% respectivamente.

Figura 47: Distancias para el cálculo de la región de interés.



Referencia: Elaboración propia en base a imagen recopilada de http://shadwofdemon.blogspot.com/2011/03/rostro-de-un-hombre.html

- Cálculo de la región de interés del lado izquierdo

Algoritmo 2: Calcula la Región de Interés para el ojo izquierdo

Variables de Entrada:

I: Una imagen de un rostro detectado

A: Entero que representa la distancia en porcentaje desde el lado superior de **I** hasta la Región de Interés

B: Entero que representa el porcentaje de la altura de la Región de Interés

C: Entero que representa la distancia en porcentaje desde el lado izquierdo de **I** hasta la Región de Interés

E:Entero que representa la distancia en porcentaje desde la mitad horizontal de **I** hasta la Región de Interés

Variables de Salida:

D': Un real que guarda el ancho de la región de interés para el ojo izquierdo

B': Un real que guarda el alto de la región de interés para el ojo izquierdo

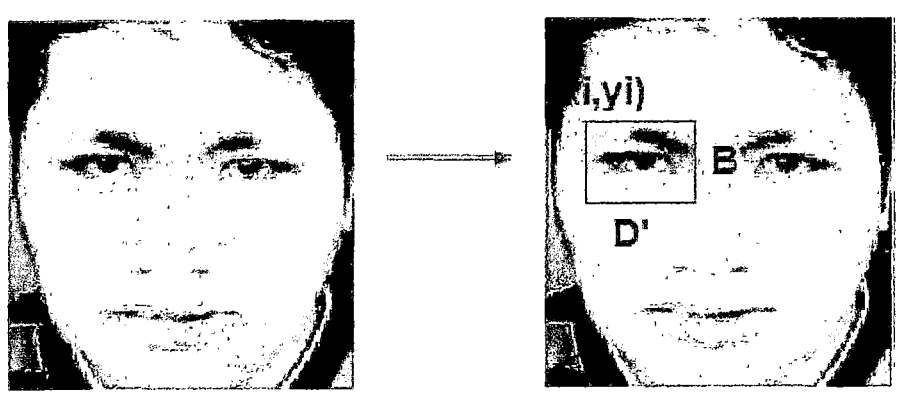
xi: Un real con la componente X de la posición de la esquina superior izquierda de la Región de Interés

yi: Un real con la componente Y de la posición de la esquina superior izquierda de la Región de Interés

```
RegionDeInteresIzquierdo(I)
01: D'←ancho(I)/2 - (ancho(I) x C/100) - (ancho(I) x E/100)
02: B'← alto(I)x B/100
03: xi←ancho(I) x C/100 //Coordenada X de la esquina superior izquierda de la ROI
04: yi ←alto(I) x A/100  //Coordenada Y de la esquina superior izquierda de la ROI
```

La región de interés para el ojo izquierdo es el rectángulo de ancho D' y alto B' extraído del rostro a partir de las coordenadas (xi,yi).

Figura 48: Resultado del Algoritmo 2



Fuente: Elaboración propia

- Cálculo de la región de interés del lado derecho: Usamos las mismas medidas de la región de interés del lado izquierdo pero invertidos hacia la otra mitad del rostro.

Algoritmo 3: Calcula la Región de Interés para el ojo derecho

Variables de Entrada:

I: Una imagen de un rostro detectado

A: Entero que representa la distancia en porcentaje desde el lado superior de I hasta la Región de Interés

B: Entero que representa el porcentaje de la altura de la Región de Interés

C: Entero que representa la distancia en porcentaje desde el lado izquierdo de I hasta la Región de Interés

E:Entero que representa la distancia en porcentaje desde la mitad horizontal de I hasta la Región de Interés

Variables de Salida:

D': Un real que guarda el ancho de la región de interés para el ojo izquierdo

B': Un real que guarda el alto de la región de interés para el ojo izquierdo

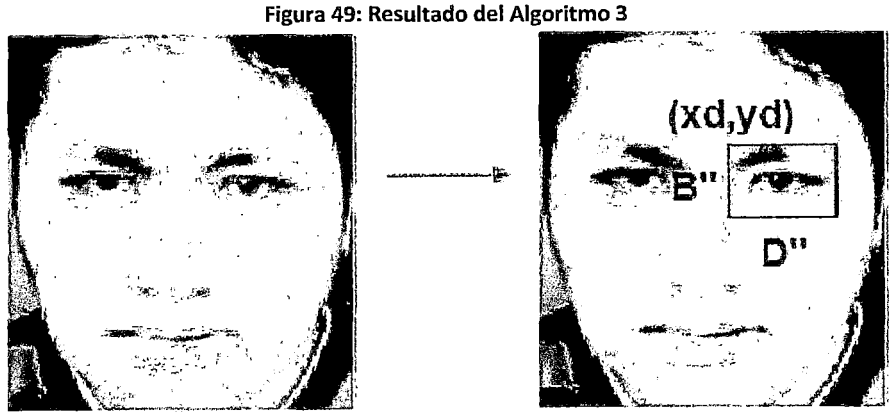
xi: Un real con la componente X de la posición de la esquina superior izquierda de la Región de Interés

yi: Un real con la componente Y de la posición de la esquina superior izquierda de la Región de Interés

```
RegionDeInteresDerecho(I)
01: D"←ancho(I)/2 - (ancho(I) x C/100) - (ancho(I) x E/100)
02: B"← alto(I) x B/100
03: xd ←ancho(I)/2 - ancho(I) x E/100
04: yd ←alto(I) x A/100
```

La región de interés para el ojo derecho es el rectángulo de ancho D" y alto B" extraído del rostro a partir de las coordenadas (xd,yd).

Figura 49: Resultado del Algoritmo 3



Fuente: Elaboración propia

3.1.4 BUSCAR OJOS ABIERTOS

Establecidas las dos regiones de interés procedemos a buscar los ojos abiertos, primero se localiza el ojo izquierdo si este es encontrado entonces posteriormente se localiza el ojo derecho. Se hace uso de la clase

CDetectorOjos, la cual funciona de forma similar a la clase *CDetectorRostro* explicado en la etapa de detectar rostros. Se cuenta con dos métodos principales:

- CargarClasificadorOjos
- DetectarOjos

3.1.4.1 CargarClasificadorOjos

Este método nos permite establecer el archivo del clasificador que usaremos para detectar los ojos en las regiones de interés calculadas. Utilizamos el clasificador *haarcascade_eye_2.xml,* este clasificador nos ayuda a detectar ojos abiertos y está disponible para la librería OpenCV.

3.1.4.2 DetectarOjos

Este método nos permite detectar los ojos en el rostro detectado utilizando el clasificador de ojos abiertos y al igual que el método de *DetectarRostros* explicado anteriormente utiliza el método *DetectHaarCascade* de la librería OpenCV. Devuelve el número de ojos encontrados, dos si se encontraron ambos ojos abiertos y cero si no se encontró ningún ojo abierto. Primero se busca un ojo abierto en la región de interés del lado izquierdo solo si este es hallado entonces se procede a buscar un ojo en la región de interés del lado derecho. En este método está incluido el cálculo de las regiones de interés anteriormente descritas.

Algoritmo 4: Detectar ojos abiertos en una imagen

Variables de Entrada:

I: Una imagen de un rostro detectado

Variables Auxiliares:

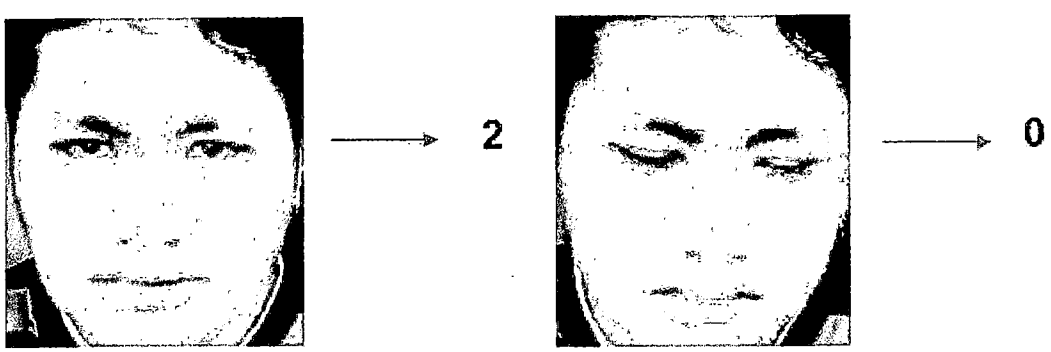
D', B', xi, yi, D", B", xd, yd: Almacenan los valores para las regiones de interés

Variables de Salida:

S: Un entero de valor 2 cuando se detectan ambos ojos abiertos y 0 si hay menos de dos ojos abiertos.

```
DetectarOjos(I,x,y)
01: RegionDeInteresIzquierdo(I)
02: S ← 0
03: ROI(I, D', B', xi, yi) // Establece la región de interés para buscar el ojo izquierdo
04: O ←DetectHaarCascade(I)//Busca ojos abiertos en I en la región de interés izquierda
05: si longitud(O) > 0 entonces//Se encontró por lo menos un ojo izquierdo abierto
06:     RegionDeInteresDerecho(I)
07:     ROI(I, D", B", xd, yd) // Establece la región de interés para buscar el ojo derecho
08:     O ←DetectHaarCascade(I) //Busca ojos abiertos en I en la región de interés derecha
09:     si longitud(O) > 0 entonces //Se encontró por lo menos un ojo derecho abierto
10:         S ←2
11:     fin si
12: fin si
13: retornar S
```

Figura 50: Resultado del Algoritmo 4



Referencia: Elaboración propia

3.1.5 CALCULAR PERCLOS

Como se indicó en el anterior capitulo para el cálculo de estado de somnolencia nos basamos en la medida del PERCLOS que es el porcentaje de ojos cerrados. En el caso de que en la parte de detección de rostros no

se haya encontrado ningún rostro entonces el fotograma actual es considerado como si los ojos estuvieran cerrados, si se detecta un rostro se hace la búsqueda de los ojos, si ambos ojos son encontrados entonces se considera el fotograma como ojos abiertos caso contrario diremos que los ojos están cerrados.

El valor del PERCLOS se actualiza en cada fotograma, para esto se considera el estado de los ojos en los últimos 10 fotogramas y contando el número de ojos cerrados en esos 10 fotogramas calculamos el valor del PERCLOS.

Si tenemos 10 fotogramas ó imágenes "f" y tenemos una función C(x) que devuelve **0** si en la imagen hay un rostro con ambos ojos abiertos y **1** si en el fotograma los ojos están cerrados entonces el valor de PERCLOS sería igual a:

$$PERCLOS = \frac{\sum_{i=1}^{n} C(f_i)}{n} \; x \; 100\%$$

En el presente trabajo el índice PERCLOS a partir del cual se considera que el estado de somnolencia es peligroso es para un PERCLOS>40% valor que se ha obtenido con base a la medición del tiempo de procesamiento de 5 pestañeos prolongados, para cada fotograma en que los ojos están cerrados se mide el tiempo y se obtuvo el promedio, con lo cual se obtuvo la siguiente tabla:

Tabla 3: Duración en milisegundos de clasificación de ojos cerrados en 5 pruebas

Estado ojos	Prueba 1 (ms)	Prueba 2 (ms)	Prueba 3 (ms)	Prueba 4 (ms)	Prueba 5 (ms)	Promedio (ms)
Cerrados	49	53	51	51	39	48.6
Cerrados	99	118	110	133	130	118
Cerrados	186	179	163	169	165	172.4
Cerrados	237	245	250	235	234	240.2
Cerrados	275	287	294	331	309	299.2
Cerrados	362	361	346	367	362	359.6
cerrados	396	400	438	420	431	417
cerrados	451	482	482	472	477	472.8
cerrados	500	543	531	539	567	536

Fuente: Elaboración propia

Sabiendo que un pestañeo reflejo dura como máximo 182ms podemos deducir que tres fotogramas "*cerrados*" seguidos corresponden a un pestañeo reflejo ya que su tiempo de procesamiento corresponde en promedio a 172ms, el cuarto fotograma *cerrado* es evaluado a los 240ms, entonces un pestañeo prolongado y por lo mismo un estado de somnolencia peligroso ocurriría a partir del cuarto fotograma cerrado detectado.
Se evalúan el estado de los ojos de los últimos 10 fotogramas para obtener el índice PERCLOS y en base a este índice hemos definido tres estados de somnolencia:

- ALERTA o DESPIERTO: Cuando el valor de PERCLOS es menor o igual a 30%.
- CANSADO: Este estado ocurre cuando el valor de PERCLOS supera al 30% pero es menor o igual al 40%.
- DORMIDO: Este estado ocurre cuando el valor de PERCLOS supera al 40%, y es en este estado en el que se debe dar alguna señal de alarma al conductor.

Esta etapa de calcular el valor de PERCLOS incluye dos métodos:

- actualizarPERCLOS
- DibujarEstadoActual

3.1.5.1 actualizarPERCLOS

Este método cuenta el número de ojos cerrados en los "n" fotogramas anteriores, en nuestro caso 10, y calcula el porcentaje de ojos cerrados y también guarda estos valores PERCLOS para mostrar su evolución en un gráfico.

Algoritmo 5: Calcula el valor del PERCLOS actual y actualiza H

Variables de Entrada:

b: Un booleano que indica si se encontraron o no ambos ojos abiertos en el fotograma o imagen actual

H: Un arreglo de booleanos que guarda el estado de los ojos de los "n" fotogramas anteriores

Variables Auxiliares:

n: El número de fotogramas que se considerarán para el cálculo del PERCLOS

c:Un entero el cual servirá como contador de los "ojos cerrados" en H

Variables de Salida:

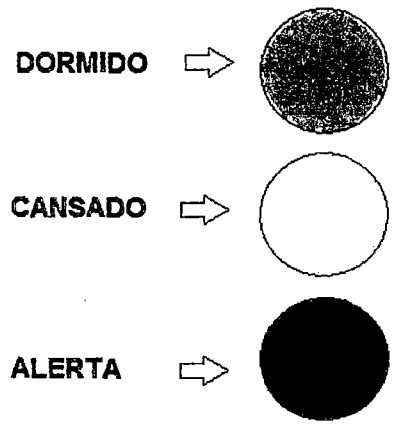
PERCLOS: Guarda el valor del índice PERCLOS, calculado a partir de los valores almacenados en H

```
actualizarPERCLOS(b,H)
01: c ← 0
02: para i←1hasta n-1 hacer
03:    H[i] ← H[i+1]
04:    si H[i]<>TRUE entonces  // En el fotograma el ojo está cerrado
05:       c ← c + 1
06:    fin si
07: fin para
08: H[n] ←b
09: si H[n]<>TRUE entonces
10:    c ← c + 1
11: fin si
12:PERCLOS ←c/n x 100
```

3.1.5.2 DibujarEstadoActual

Este método sirve para detectar el estado de somnolencia actual en el conductor, hace uso del valor del PERCLOS y de los valores definidos para cada estado (ALERTA <= 30%, CANSADO entre 30% y 40% y DORMIDO superior al 40%). Para mostrar estos valores en pantalla se ha hecho una imagen tipo semáforo en la cual el VERDE representa al estado ALERTA, el color AMARILLO representa al estado CANSADO y el ROJO al estado DORMIDO, en el estado DORMIDO es cuando se debería lanzar la alarma de somnolencia.

Figura 51: Color mostrado según estado de somnolencia del conductor



Referencia: Elaboración propia

3.1.6 MÉTODO PRINCIPAL

El siguiente método engloba todos los métodos descritos anteriormente y se observa la secuencia de los mismos.

Como se observa luego de capturar una imagen de la cámara se busca rostros, si un rostro es hallado se procede a buscar los ojos caso contrario se actualizara el valor del PERCLOS considerando que el fotograma actual están los OJOS CERRADOS, si se encuentra un rostro se busca los ojos, si dos ojos son encontrados uno para el lado izquierdo y otro para el lado derecho del rostro se actualiza el valor del PERCLOS considerando que en el fotograma actual se tiene los OJOS ABIERTOS caso contrario se considerara que en el fotograma se tienen los OJOS CERRADOS.

Algoritmo 6: Evalúa el estado de somnolencia del conductor en los fotogramas obtenidos de la webcam o el archivo de video

Variables Auxiliares:

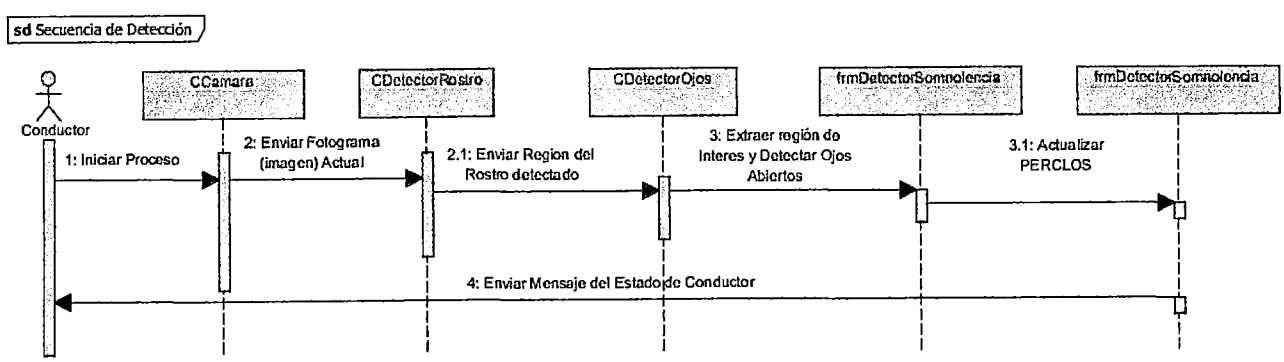
I: Una imagen obtenida de un video o de una webcam

R: Un vector que almacena los rostros detectados en I

S: Un entero de valor 2 si se encontraron ambos ojos abiertos, 0 si algún ojo está cerrado

```
DetectarEstadoSomnolencia()
01: I ←CapturarImagenCamara()
02: R ←detectarRostros(I)
03: si R=nulo entonces//No se detectó rostros
04:     actualizarPERCLOS(false) // Se considera que tiene los ojos cerrados
05: caso contrario
06:     S ←detectarOjos(R[0]) // Se busca ojos abiertos solo en el primer rostro detectado
07:     si S=2 entonces // Se encontraron ambos ojos abiertos
08:         actualizarPERCLOS(true)
09:     caso contrario // El conductor tiene por lo menos un ojo cerrado
10:         actualizarPERCLOS(false)
11: fin si
```

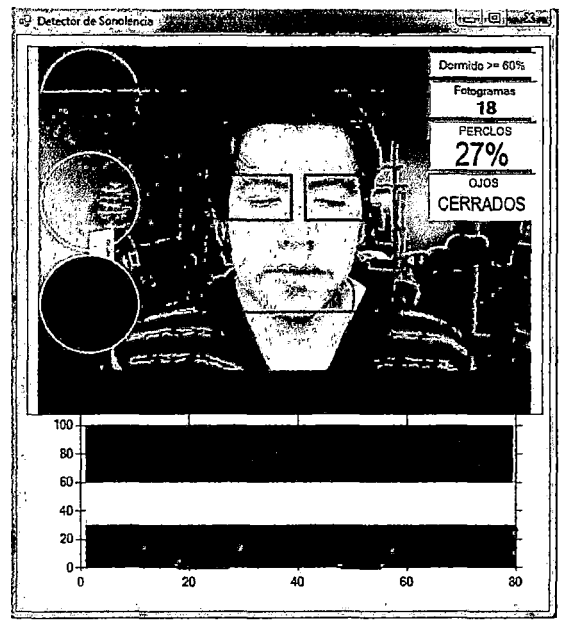
Figura 52: Diagrama de secuencia del algoritmo



Fuente: Elaboración propia

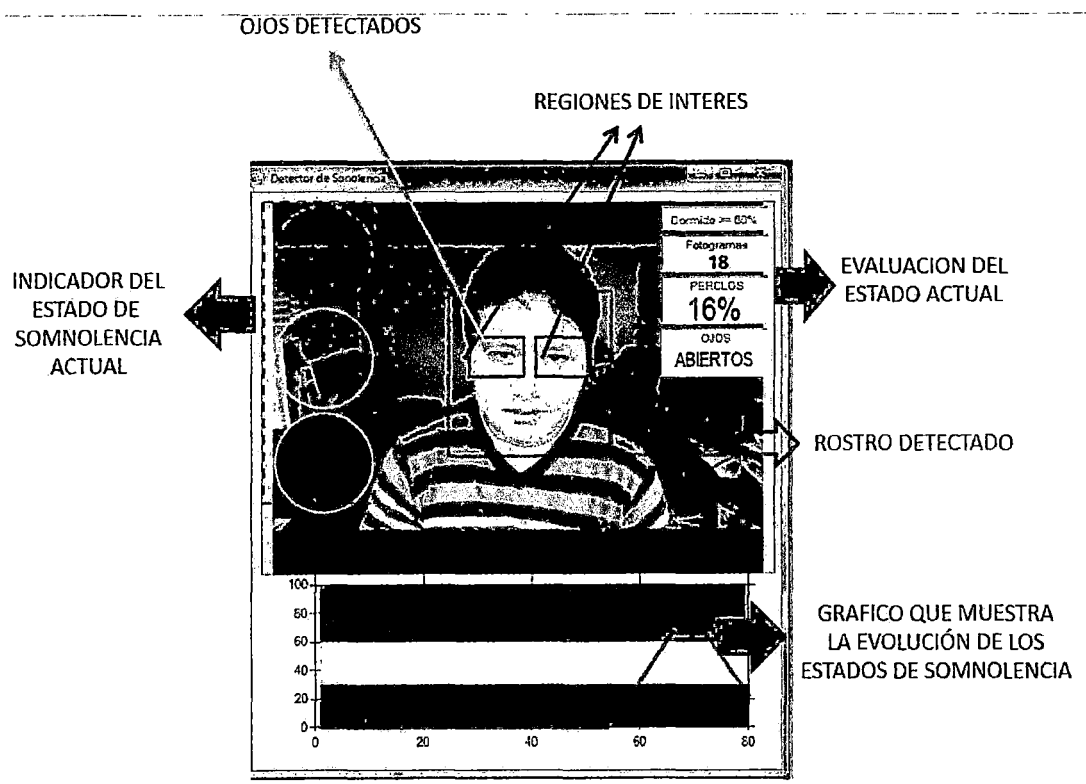
En las siguientes imágenes se muestran la interfaz del módulo principal y se describe cada una de las partes mostrada.

Figura 53: Muestra de la Interfaz del módulo principal



Fuente: Elaboración propia

Figura 54: Muestra de la interfaz del módulo principal

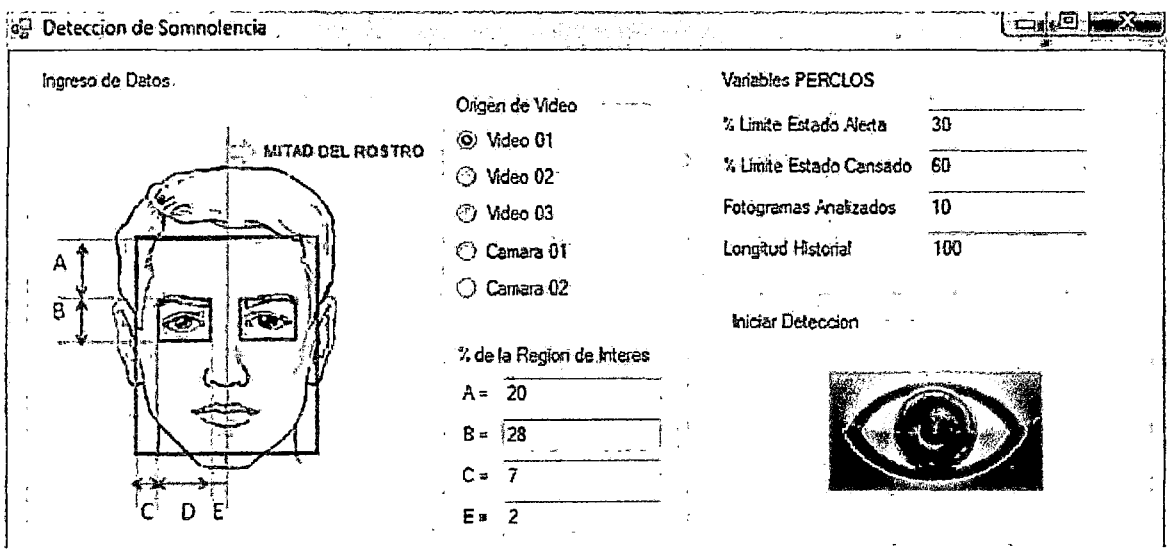


Fuente: Elaboración propia

3.2 INTERFAZ DE CONFIGURACIÓN INICIAL

Para facilitar la prueba de la herramienta construida se ha implementado una interfaz inicial en donde se indica los parámetros que se usaran para la detección de la somnolencia, tales como fuente de video, parámetros para el cálculo de la región de interés de los ojos y parámetros para el cálculo de la medida de somnolencia PERCLOS.

Figura 55: Vista de la Configuración inicial para probar la herramienta construida



Fuente: Elaboración propia

CAPITULO IV

RESULTADOS DE LA EXPERIMENTACIÓN

Para probar el algoritmo de reconocimiento de somnolencia usamos tres fuentes de video como medios de entrada, un video obtenido de la web en donde la conductora está manejando su auto durante la noche y el cual ha sido grabado usando una webcam con luz infrarroja y dos videos los cuales grabamos durante luz de día y usando una cámara sin luz infrarroja.

4.1 METODOLOGÍA PARA LA EXPERIMENTACIÓN

Para poder evaluar los resultados obtenidos con el prototipo de reconocimiento de somnolencia lo que hemos hecho es capturar imágenes de una persona con la mirada al frente durante aproximadamente un minuto, a esta persona se le pide que simule efectos de somnolencia (es decir pestañear como cuando se tiene sueño) en pequeños intervalos de tiempo, entonces se evalúa cada fotograma con el prototipo y cada fotograma evaluado es guardado en dos carpetas diferentes, si el fotograma fue clasificado como **ojos abiertos** lo guarda en una carpeta ABIERTOS y si fue clasificado como **ojos cerrados** lo guarda en una carpeta CERRADOS, luego contamos el total de fotogramas que se evaluaron **T**, y mediante verificación manual contamos la cantidad de fotogramas que tienen los ojos abiertos **A**, la cantidad de fotogramas que tienen los ojos cerrados **C** y también contamos cuantos fotogramas fueron clasificados erróneamente como abiertos **Fa** y cuantos fotogramas fueron clasificados erróneamente como cerrados **Fc.** Los valores de **Fa** y **Fc** corresponden a los FALSOS POSITIVOS.

Luego el error más relevante para evaluar la herramienta es el error producto de clasificar ojos cerrados erróneamente como ojos abiertos, este error en porcentajes sería:

$$E = \frac{Fa}{C} x\ 100\%$$

4.2 PRUEBA UNO CON LUZ DIURNA

Tabla 4: Resultados de la prueba 1

Número de Fotogramas	**3766**
Fotogramas con ojos abiertos	2966
Fotogramas con ojos cerrados	800
Falsos ojos abiertos	55
Falsos ojos cerrados	16

4.2.1 ALGUNOS FOTOGRAMAS CLASIFICADOS COMO ABIERTOS

Figura 56: Fotogramas clasificados como abiertos en prueba 1

Referencia: Elaboración propia

4.2.2 ALGUNOS FOTOGRAMAS CLASIFICADOS COMO CERRADOS

Figura 57: Fotogramas clasificados como cerrados en prueba 1

Referencia: Elaboración propia

4.2.3 FOTOGRAMAS CLASIFICADOS COMO ABIERTOS INCORRECTAMENTE

Figura 58: Falsos Cerrados en prueba 1

Referencia: Elaboración propia

4.3 PRUEBA DOS CON LUZ DIURNA

Tabla 5: Resultados de la prueba 2

Número de Fotogramas	**3567**
Fotogramas con ojos abiertos	2866
Fotogramas con ojos cerrados	701
Falsos ojos abiertos	64
Falsos ojos cerrados	0

4.3.1 ALGUNOS FOTOGRAMAS CLASIFICADOS COMO ABIERTOS

Figura 59: Fotogramas clasificados como abiertos en prueba 2

Referencia: Elaboración propia

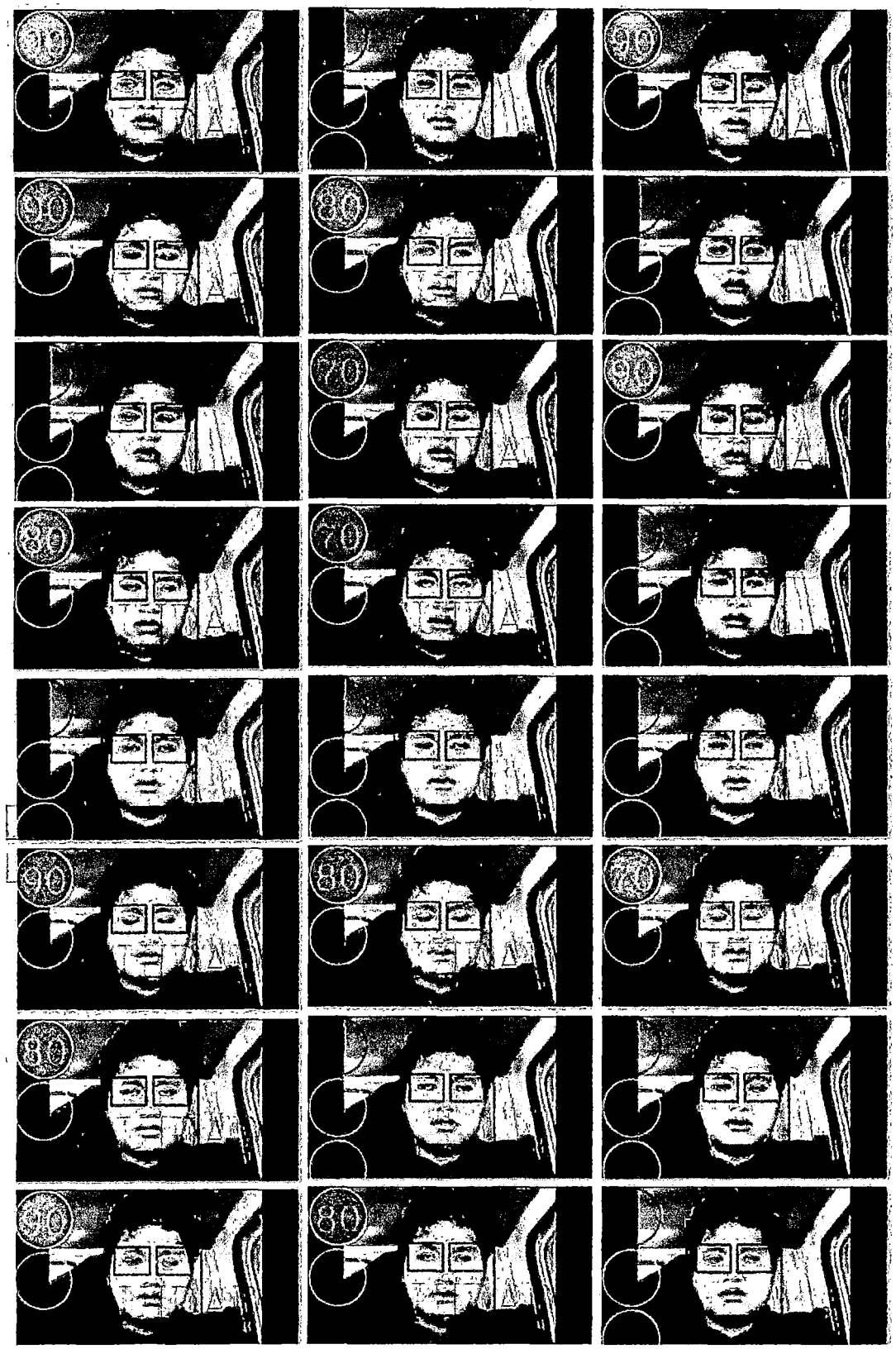
4.3.2 ALGUNOS FOTOGRAMAS CLASIFICADOS COMO CERRADOS

Figura 60: Fotogramas clasificados como cerrados en prueba 2

Referencia: Elaboración propia

4.3.3 FOTOGRAMAS CLASIFICADOS COMO ABIERTOS INCORRECTAMENTE

Figura 61: Falsos abiertos en prueba 2



Referencia: Elaboración propia

4.4 ANÁLISIS DE LOS RESULTADOS

En la siguiente tabla se resumen los resultados obtenidos, así como, los porcentajes de falsos positivos que se han obtenido para cada caso.

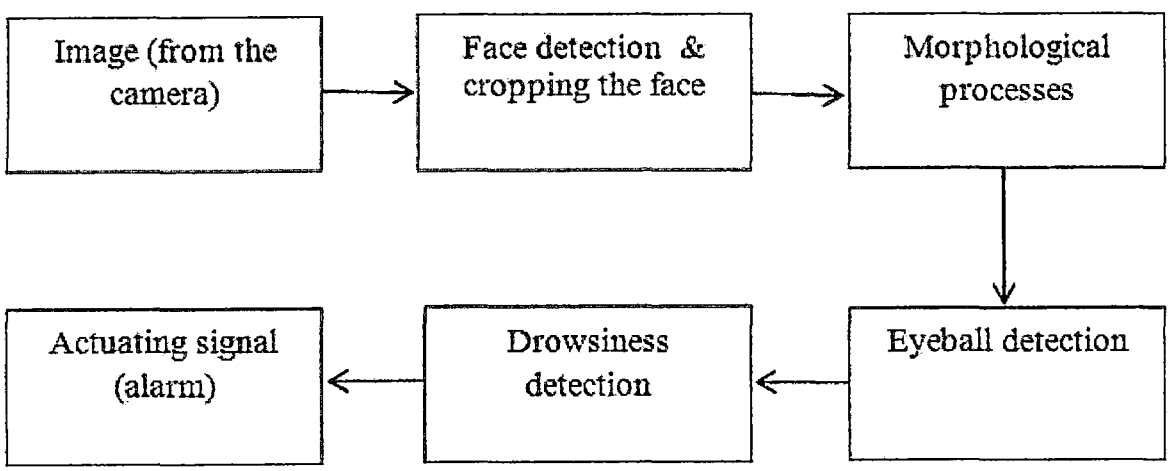
Tabla 6: Errores obtenidos en las pruebas 1 y 2

Fuente de prueba	Total Fotogramas (T)	Fotogramas Abiertos (A)	Fotogramas Cerrados (C)	Falsos Cerrados (Fc)	Falsos Abiertos (Fa)	% Falsos	% Falsos Cerrados	% Falsos Abiertos (E)
Video Frederick	3766	2966	800	16	55	**1.89%**	**0.54%**	**6.88%**
Video Angel	3567	2866	701	0	64	**1.79%**	**0.00%**	**9.13%**
					Promedio:	**1.84%**	**0.27%**	**8.00%**

Aunque en promedio el porcentaje de FALSOS POSITIVOS es bajo (1.84%), individualmente el porcentaje de falsos cerrados en promedio es de 0.27% y el porcentaje de falsos abiertos en promedio es 8.00%, como nos interesa más evitar los FALSOS ABIERTOS podemos decir que el error obtenido es de 8% en promedio.

Seifoory et.al. en su trabajo sobre detección de somnolencia [28] muestran los resultados que obtuvieron en pruebas realizadas con una webcam de resolución de 640x480, con iluminación estable y a una distancia aproximada de entre 20 y 40cm, el algoritmo que utilizan trabaja evaluando cada 10 fotogramas, se obtiene una imagen de la webcam, se ubica el rostro, y mediante un análisis morfológico del rostro tratan de ubicar los círculos del iris, con esta información calculan el estado de somnolencia y si es necesario accionan una alarma.

Figura 62: Algoritmo de Seifoory et. al. [28]



Referencia: Seifoory et. al. [29], p.52

Para validar sus resultados Seifoory et. al. [28] presentan también resultados obtenidos en otros trabajos, los cuales mostramos en la siguiente tabla:

Tabla 7: Resultados obtenidos por Seifoory et. al. [28]

Estado de los Ojos	Algoritmo basado en Varianza	Algoritmo basado en distancia entre parpados	Algoritmo basado en brillo	Algoritmo de Seifoory et.al. sin Reconocimiento Facial	Algoritmo de Seifoory et.al. con Reconocimiento Facial	**Nuestro Algoritmo**
	Precisión (%)	Precisión (%)	Precisión (%)	Precisión (%)	Precisión (%)	**Precisión (%)**
Abiertos	96.4	100	96.4	95.5	97	**99.73**
Cerrados	67.5	72.5	94.7	96	98	**92.00**

Referencia: Elaboración propia en base a tabla presentada por Seifoory et.al. [28] p.53

Estos resultados muestran que el uso de algoritmo de detección de objetos aplicado al reconocimiento de somnolencia nos da una precisión cercana a los obtenidos por el algoritmo basado en brillo, aunque la precisión es inferior al algoritmo propuesto por Seiffoory et. al. es superior a los resultados de los algoritmos basado en varianza y distancia entre parpados.

4.5 LIMITACIONES

Existen algunas limitaciones del reconocimiento de somnolencia, algunas como condiciones de iluminación, distancia del rostro frente a la cámara, entre otras pueden afectar a la calidad de la imagen obtenida y por lo tanto a su procesamiento.

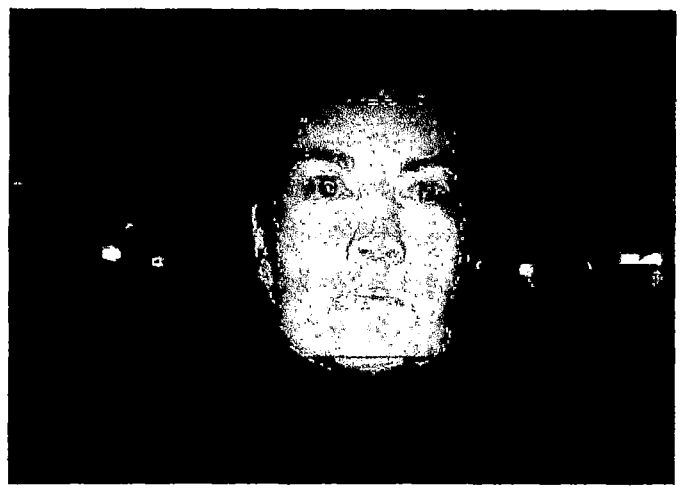
4.5.1 DEPENDENCIA DE LA ILUMINACIÓN

Inclusive con pésimas condiciones de iluminación se puede detectar rostros sin embargo cuando se producen sombras en los ojos puede hacer que algunas veces no se detecten los ojos lo cual hace que se considere erróneamente un estado de OJOS CERRADOS. Esto se soluciona teniendo condiciones adecuadas de luz, en el presente trabajo se considero solamente el problema de reconocimiento de somnolencia limitado a iluminación natural diurna y no se trabajo con luz de noche ya que fue difícil conseguir webcams con luz infrarroja en el mercado nacional, el problema

de la iluminación durante la noche, en donde los conductores son más propensos en entrar a un estado de somnolencia, se puede enfrentar usando webcams con luces infrarrojas puesto que permiten recoger suficiente información visual incluso en la oscuridad.

En la siguiente imagen se observa como con una webcam con luces infrarrojas muestra claramente el rostro del conductor durante la noche.

Figura 63: Imagen obtenida con webcam Infrarroja



Referencia: Extraído de video obtenido de https://www.youtube.com/watch?v=74Wp75I83gA
Último Acceso: 04 de abril del 2013

4.5.2 DISTANCIA DEL CONDUCTOR A LA CÁMARA

Cuando la distancia entre el rostro del conductor y la webcam no se encuentra en un rango optimo puede haber errores en la detección de los estados de somnolencia, si la webcam está muy lejos del rostro del conductor bajará la calidad de la imagen del rostro lo cual hará que falle en la detección de los ojos, en cambio si la cámara está muy cerca del rostro del conductor no se detectará el rostro por lo cual tampoco se detectarán los ojos. Por lo tanto es recomendable para el buen monitoreo del estado de somnolencia del conductor que la webcam no esté ni muy cerca ni muy lejos del rostro del conductor, en general esto ocurre cuando el rostro del conductor está a una distancia de entre 40cm a 60cm de la webcam, o cuando en la imagen el rostro ocupa las 3/4 partes de la altura de la imagen aproximadamente.

Figura 64: En la figura: (a) Rostro muy alejado. (b) Rostro muy cercano (c) Rostro a una distancia adecuada.

(a) (b) (c)

Referencia: Elaboración propia

4.5.3 ORIENTACIÓN DEL ROSTRO

Se puede detectar rostros y ojos cuando el conductor está mirando más o menos de frente hacia la cámara sin embargo se falla en la detección de rostros y ojos si el rostro del conductor está inclinado o girado hacia los lados, a pesar de esto ya que el objetivo es verificar el estado de somnolencia e interesa que el conductor tenga la mirada hacia el frente el considerar que el conductor no está ALERTA al tener el rostro girado no es un problema relevante ya que incluso la inclinación del rostro podría deberse a cabeceos y en general representa distracción del conductor.

Figura 65: Rostros no detectados por inclinación y giros del rostro

Referencia: Elaboración propia

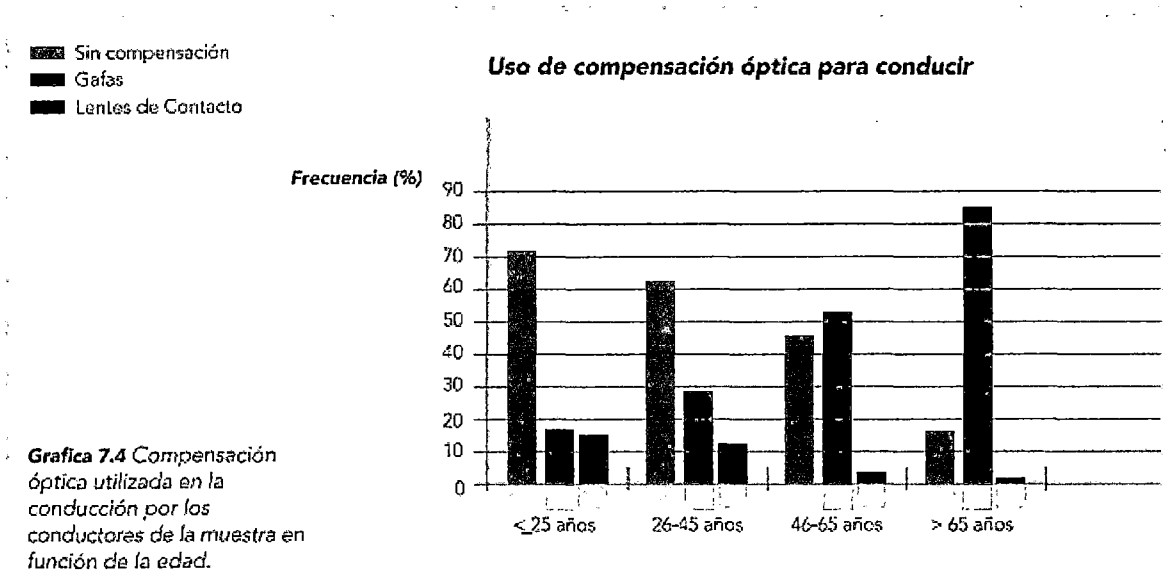
4.5.4 POBRE DETECCIÓN CON LENTES

En general cualquier objeto que interfiera los ojos del conductor pueden hacer que la herramienta falle en detectar los ojos, objetos como gafas oscuras para el sol, gorras, cascos y otros que imposibiliten que el rostro y/o los ojos aparezcan claramente en la imagen harán que la herramienta considere al conductor como en estado de OJOS CERRADOS. Este problema aún no ha sido solucionado por lo que la herramienta puede no

trabajar correctamente con personas con gafas, sin embargo con el uso de lentes con lunas y marcos transparentes este problema puede ser solucionado así mismo se puede reemplazar las gafas por lentes de contacto lo cual eliminaría totalmente este problema.

En la siguiente figura podemos observar el porcentaje de conductores que usan lentes para conducir, esta figura fue extraída de los resultados de un estudio sobre visión y conducción hecha en España por la Universidad Politécnica de Cataluña [30], de la cual podemos inferir que 50% de los conductores < 65 años usan lentes, por lo cual se puede inferir que el presente trabajo se puede aplicar a 50% de los conductores.

Figura 66: Uso de lentes en conductores por edades



***Grafica 7.4** Compensación óptica utilizada en la conducción por los conductores de la muestra en función de la edad.*

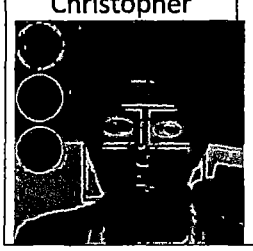
Referencia: Torrents et. al. [30], p46.

4.5.5 APERTURA DE LOS PÁRPADOS

La apertura de los parpados en los ojos afecta a la eficacia del algoritmo, en general mientras los ojos tengan mayor apertura de parpados mejor será la detección de los ojos, caso contrario mientras menor sea la apertura de los párpados la herramienta clasificará a los ojos como "cerrados" a pesar de encontrarse abiertos.

Para encontrar cual es la distancia mínima entre los párpados con la cual trabajamos en el presente trabajo hemos realizado pruebas con cuatro personas de lo cual se obtiene la siguiente tabla:

Tabla 8: Prueba con ojos de 7 a 9mm de apertura de párpados

Fuente de Prueba	Apertura de Párpados	Total Fotogramas (T)	Fotogramas Abiertos (A)	Fotogramas Cerrados (C)	Falsos Cerrados (Fc)	Falsos Abiertos (Fa)	% Falsos	% Falsos Cerrados	% Falsos Abiertos
Video María 	9mm	1942	1616	326	29	1	1.54	1.49	0.05
Video Vargas 	8mm	1715	1323	392	16	11	1.57	0.93	0.64
Video Betsy 	7mm	805	514	291	42	87	16.02	5.22	10.81
Video Christopher 	6mm	1204	957	247	347	0	28.82	28.82	0.00

Fuente: Elaboración propia

De la tabla anterior podemos ver que a partir de una distancia de 7mm para abajo el error supera al 16% por lo cual para el presente trabajo solo consideraremos ojos con apertura de párpados mayores a 7mm.

CONCLUSIONES

1. El parpadeo o pestañeo prolongado es síntoma de somnolencia en una persona, sin embargo de acuerdo a la tabla 8 se puede inferir que clasificar los ojos como "abiertos" o "cerrados" es posible solamente cuando se tiene una apertura de párpados amplia, en la tabla 8 observamos que se obtienen buenos resultados cuando la apertura de los párpados es superior a 7mm, sin embargo, para apertura de párpados menores o iguales a 7mm el error es alto donde el porcentaje de falsos detectados se acerca al 30%.
2. Se sabe de los resultados teóricos que la duración de un pestañeo reflejo es de un máximo de 182ms [13], por lo que, cualquier pestañeo de una duración mayor a 182ms evidenciaría un estado de somnolencia peligroso, de acuerdo a la tabla 3 después de evaluar cuatro fotogramas como "cerrados" se supera este tiempo máximo y teniendo en cuenta que evaluamos el estado de los ojos de los últimos 10 fotogramas el índice PERCLOS > 40% representa un estado de somnolencia peligroso con lo cual la alarma es mostrada en un tiempo promedio de 299ms desde que el conductor se quedó dormido.
3. Se utiliza el algoritmo de detección de objetos de Viola & Jones en primer lugar para ubicar el rostro del conductor en la imagen, posteriormente se establecen regiones de interés para buscar ojos abiertos mediante la detección de objetos y con esta información calcular el índice PERCLOS asumiendo que si no se detectó ningún ojo abierto entonces estos deben estar cerrados, cada fotograma es evaluado en un tiempo promedio de 85ms de acuerdo a la tabla 9.
4. Usando una webcam de 640x480 pixeles ubicada a una distancia media de entre 40 a 60cm y con luz natural diurna se obtuvo fotogramas del conductor y se evaluó el estado de los ojos usando el algoritmo de detección de objetos de Viola & Jones ya implementado en la librería OpenCV, los resultados de esta evaluación comparados con una clasificación manual nos dio un error de FALSOS ABIERTOS promedio de 8% para ojos con apertura de párpados mayores a 8mm, estos resultados son próximos a los errores obtenidos en otros trabajos.

Bajo las condiciones simuladas siguientes: Luz natural diurna, una webcam ubicada a una distancia de 40 a 60cm del conductor y a la altura del volante de un auto sin movimiento, obtenemos un error de hasta 8% en la clasificación de ojos "abiertos" o "cerrados" usando el algoritmo de detección de objetos de Viola & Jones en ojos con apertura de párpados mayores a 7mm, el índice PERCLOS > 40% representa un estado de somnolencia peligroso y se llega a este valor en un tiempo promedio de 299ms luego del cual se muestra la alarma al conductor.

RECOMENDACIONES

Los resultados obtenidos en este proyecto pueden servir de base para trabajos futuros sobre todo en sistemas de ayuda a la conducción.

1. Se recomienda utilizar la herramienta desarrollada para implementar un sistema de ayuda a la conducción con alerta de sueño al llegar a un estado de somnolencia peligroso.
2. También se debería estudiar la posibilidad de usar cámaras con mejor resolución para eliminar al máximo los errores en la detección de ojos, para disminuir el problema de la apertura de los párpados se podría usar cámaras con mejor resolución que una vez ubicado el ojo evaluarlo con otros algoritmos tal como la detección de círculos de Hough para detectar iris.
3. Para poder detectar el estado de somnolencia incluso cuando el conductor tiene el rostro girado se plantear usar más cámaras, con 3 cámaras instaladas una al centro y dos a los costados derecho e izquierdo del conductor se podría reconocer el estado de los ojos del conductor cuando este tenga la mirada a izquierda o derecha.
4. Además de evaluar el pestañeo prolongado se recomienda ampliar el presente trabajo para evaluar otras características del sueño como lo son el bostezo y el cabeceo.
5. El presente trabajo se limita a condiciones de luz diurna, se recomienda evaluar el funcionamiento de este trabajo en condiciones nocturnas para lo cual se debe plantear el uso de cámaras infrarrojas.
6. Sabiendo que el tiempo de procesamiento y por lo tanto el tiempo de respuesta del algoritmo es importante para poder dar la alarma de estado de somnolencia peligroso en el menor tiempo posible se recomienda utilizar lenguajes de programación más rápidos en un entorno real, tales como, el C o C++; así como recomendamos también que dichos lenguajes sean enseñados en la Carrera Profesional de Ingeniería Informática y de Sistemas.
7. Las horas de conducción afectan en el cansancio y la somnolencia del conductor, recomendamos también que en futuros trabajos se implementen

mecanismos para reconocer el tiempo que un conductor está frente al volante para lo cual se podría usar algoritmos de reconocimiento facial.

REFERENCIAS BIBLIOGRÁFICAS

[3] J. Rey de Castro, J. Gallo y H. Loureiro, «Cansancio y somnolencia en conductores de ómnibus y accidentes de carretera en el Perú: estudio cuantitativo,» *Revista Panamericana de Salud Pública,* vol. 16, nº 1, pp. 11-18, 2004.

[5] M. J. Flores, «Sistema avanzado de asistencia a la conducción mediante visión por computador para la detección de somnolencia,» [Tesis doctoral] Departamento de Ingeniería de Sistemas y Automática, Universidad Carlos III de Madrid. Leganés, 2009.

[6] M. J. Flores, J. M. Armingol y A. de la Escalera, «Sistema Avanzado de Asistencia a la Conduccion para la Detección de la Somnolencia,» *Revista Iberoamericana de Automática e Informática industrial,* nº 8, pp. 216-228, 2011.

[10] J. D. Morales, «Diseño de prótesis ocular electrónica,» [Tesis de grado] Universidad de San Carlos de Guatemala. Guatemala, 2009.

[14] B. Mwengue y D. Rodenstein, «Evaluación de la Somnolencia,» *Archivos de Bronconeumología,* vol. 45, nº 7, pp. 349-351, 03 2009.

[18] M. Chirinos, «Visión Artificial: Percepción de profundidad,» [Tesis Profesional] Instituto tecnológico de Mérida. Mérida, 2004.

[19] C. Platero, «Apuntes de Visión Artificial,» Universidad Politécnica de Madrid. Madrid, 2009.

[20] O. E. Zambrano y A. X. Toala, «Implementación de un sistema de vigilancia utilizando una Web Cam, Asterix y telefonos Grandstream,» [Tesis de Grado] Escuela Superior Politécnica del Litoral. Guayaquil, 2009.

[21] P. S. Cazares, «Implementación de una tarjeta controladora de video porteros mediante PIC´s,» [Tesis de Grado] Escuela Politécnica Nacional. Quito, 2007.

[23] L. A. Martinez, «Sistema de visión para el equipo de robots autónomos del ITAM,» [Tesis de Grado] Instituto Tecnológico Autónomo de México. México D.F., 2004.

[25] A. Morelli y S. Padovani, «Detección y Reconocimiento de Caras,» [Tesis de Licenciatura] Universidad de Buenos Aires. Buenos Aires, 2011.

[26] M. Gonzales, «Reconocimiento de Iris» [Trabajo de fin de grado] Universidad de Barcelona.

[28] P. Roldán, «Visión por Computador en Iphone 4» [Tesi de Grado] Universitat Oberta de Catalunya.

[29] H. Seifoory, D. Taherkhani, B. Arzhang, Z. Eftekhari y H. Memari, «An Accurate Morphological Drowsy Detection» *International Proceedings of Computer Science and Information Technology,* vol. 21, nº 2011, pp. 51-54, 2011.

[30] A. Torrents, J. Escofet, B. Arzhang, Z. Eftekhari y H. Memari, «Estado de la visión de los conductores españoles» Universidad Politécnica de Cataluña, 2011.

REFERENCIAS WEB

[1] Ministerio de Salud, «Semana de Transito Seguro y Saludable,» 2012. [En línea]. Available: http://www.minsa.gob.pe/portada/est_san/transito/SEMANA_TRANSITO_SEGURO_SALUDABLE.pdf. [Último acceso: 20 04 2013].

[2] Ministerio de Salud, «Factores Causantes de Accidentes de Tránsito,» 2012. [En línea]. Available: http://ditoe.minedu.gob.pe/Materiales%20DITOE/P12.pdf. [Último acceso: 03 05 2013].

[4] Universidad Nacional José Faustino Sánchez Carrión, «Metodología de la Investigación,» Octubre 2011. [En línea]. Available: https://www5.uva.es/guia_docente/uploads/2012/475/46179/1/Documento4.pdf. [Último acceso: 13 12 2011].

[7] Real Academia Española, «Diccionario de la lengua española,» 2001. [En línea]. Available: http://www.rae.es/drae/ [Último acceso: 01 05 2013].

[8] DEPARTAMENTO DE PREVENCION DE RIESGOS DEL TRABAJO, «Manual de Manejo Defensivo,» 2005. [En línea]. Available: http://www.ingenieroambiental.com/4000/Manual-Manejo-Defensivo.pdf. [Último acceso: 01 02 2013].

[9] J. Valcarcel, «Otros factores de riesgo: El sueño,» 09 2010. [En línea]. Available: http://aplch.dgt.es/PEVI//documentos/catalogo_recursos/didacticos/did_adultas/suenio.pdf. [Último acceso: 2013 02 1].

[11] A. Ramo, «Los sentidos,» 1999. [En línea]. Available: http://www.aplicaciones.info/naturales/natura21.htm. [Último acceso: 2013 05 27].

[12] E. Graue, «El ojo: Estructura y función,» 2009. [En línea]. Available: http://highered.mcgraw-hill.com/sites/dl/free/9701069293/786420/Graue_oftalmologia_3e_cap_muestra.pdf.

[13] M. W. Johns, «The Amplitude-Velocity Ratio of Blinks: A New Method for Monitoring Drowsiness,» Epworth Sleep Centre, [En línea]. Available: http://www.mwjohns.com/wp-content/uploads/2009/murray_papers/apss_2003_06_03_the_amplitude_velocity_ratio_of_blinks_a_new_method_of_monitoring_drowsiness_poster_69.pdf. [Último acceso: 2013 01 04].

[15] FEDERAL HIGHWAY ADMINISTRATION, «PERCLOS: A Valid Psychophysiological Measure of Alertness As Assessed by Psychomotor Vigilance,» 1998. [En línea]. Available: http://www.fmcsa.dot.gov/documents/tb98-006.pdf. [Último acceso: 2012 09 07].

[16] Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado, «Formatos de imagen, audio y video,» 2012. [En línea]. Available: http://www.ite.educacion.es/formacion/materiales/155/cd/pdf/04_video.pdf. [Último acceso: 01 06 2013].

[17] J. Burgue, «La cara, sus proporciones estéticas,» [En línea]. Available: http://www.sld.cu/galerias/pdf/sitios/protesis/la_cara,_sus_proporciones_esteticas.pdf. [Último acceso: 05 11 2012].

[24] Universidad Politécnica de Valencia, «Trabajando con OpenCV,» [En línea]. Available: http://web-sisop.disca.upv.es/imd/cursosAnteriors/2k8-2k9/videos/trabajandoConOpenCV/secciones/Interaccion.html. [Último acceso: 24 03 2013].

[27] Few Tutorials, «How To Improve Face Detection,» 01 2012. [En línea]. Available: http://fewtutorials.bravesites.com/entries/emgu-cv-c/level-3c---how-to-improve-face-detection. [Último acceso: 10 08 2012].

ANEXOS

GLOSARIO DE TÉRMINOS

Binarización:	La binarización de imágenes es una técnica del procesamiento de imágenes que consiste en un proceso de reducción de la información de una imagen digital a dos valores: 0 (negro) y 255(blanco).
Conductor:	El conductor o chofer, persona encargada de conducir un vehículo de motor para transportar a personas.
Falso abierto:	Cuando en un fotograma los ojos están cerrados y es clasificado incorrectamente como si tuviera los ojos abiertos.
Falso cerrado:	Cuando en un fotograma los ojos están abiertos y es clasificado incorrectamente como si tuviera los ojos cerrados.
Falso Positivo:	En el presente trabajo un falso positivo es cuando un fotograma con ojo cerrado es clasificado como "abierto" o cuando un fotograma abierto es clasificado como "cerrado".
Fotograma:	Por extensión también se llama de ese modo a cada una de las imágenes individuales captadas por cámaras de video y registradas analógica o digitalmente.
Luz Infrarroja:	La luz infrarroja nos brinda información especial que no podemos obtener de la luz visible. Nos muestra cuánto calor tiene alguna cosa y nos da información sobre la temperatura de un objeto. Se usa generalmente para poder ver objetos en la oscuridad.
OpenCV:	OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel, se usa en aplicaciones de reconocimiento facial.
Parpadeo:	Movimiento rápido de abrir y cerrar los ojos.
PERCLOS:	Índice que calcula el porcentaje de ojos cerrados en una línea de tiempo.
Pestañeo:	Movimiento rápido y repetido de los párpados.
Pestañeo Reflejo:	En el presente trabajo un pestañeo reflejo es aquel pestañeo cuya función es lubricar los ojos, ocurre de manera involuntaria o refleja y su duración máxima es de 182ms.
Pestañeo Prolongado:	Es un pestañeo que dura más que un pestañeo reflejo, generalmente es síntoma de que la persona se ha quedado dormida.
Somnolencia:	La somnolencia se refiere a sentirse cansado o adormilado, o no poder mantener los ojos abiertos.

Visión Artificial:	Es un subcampo de la inteligencia artificial, el propósito de la visión artificial es programar un computador para que "entienda" una escena o las características de una imagen.
Webcam:	Es una pequeña cámara digital conectada a una computadora la cual puede capturar imágenes

TIEMPO PROMEDIO DE EVALUACIÓN DEL ESTADO DE LOS OJOS EN UN FOTOGRAMA

Tabla 9: Tiempo de evaluación de los fotogramas

Fotogramas Evaluados	Tiempo de Evaluación (seg)	Duración de la evaluación (seg)
Fotograma 1	0.884	---
Fotograma 2	0.914	0.030
Fotograma 3	0.957	0.043
Fotograma 4	1.125	0.168
Fotograma 5	1.174	0.049
Fotograma 6	1.246	0.072
Fotograma 7	1.346	0.100
Fotograma 8	1.438	0.092
Fotograma 9	1.57	0.132
Fotograma 10	1.629	0.059
Fotograma 11	1.731	0.102
Fotograma 12	1.791	0.060
Fotograma 13	1.913	0.122
Fotograma 14	1.972	0.059
Fotograma 15	2.099	0.127
Fotograma 16	2.153	0.054
Fotograma 17	2.228	0.075
Fotograma 18	2.316	0.088
Fotograma 19	2.464	0.148
Fotograma 20	2.517	0.053
Fotograma 21	2.569	0.052
Fotograma 22	2.68	0.111
Fotograma 23	2.743	0.063
Fotograma 24	2.849	0.106
Fotograma 25	2.9	0.051
Fotograma 26	3.006	0.106
Fotograma 27	3.06	0.054
Fotograma 28	3.17	0.110
Fotograma 29	3.271	0.101
Fotograma 30	3.323	0.052
Fotograma 31	3.433	0.110
Fotograma 32	3.504	0.071
Fotograma 33	3.624	0.120
Fotograma 34	3.684	0.060
Fotograma 35	3.83	0.146
Fotograma 36	3.882	0.052
Fotograma 37	3.933	0.051
Fotograma 38	4.052	0.119

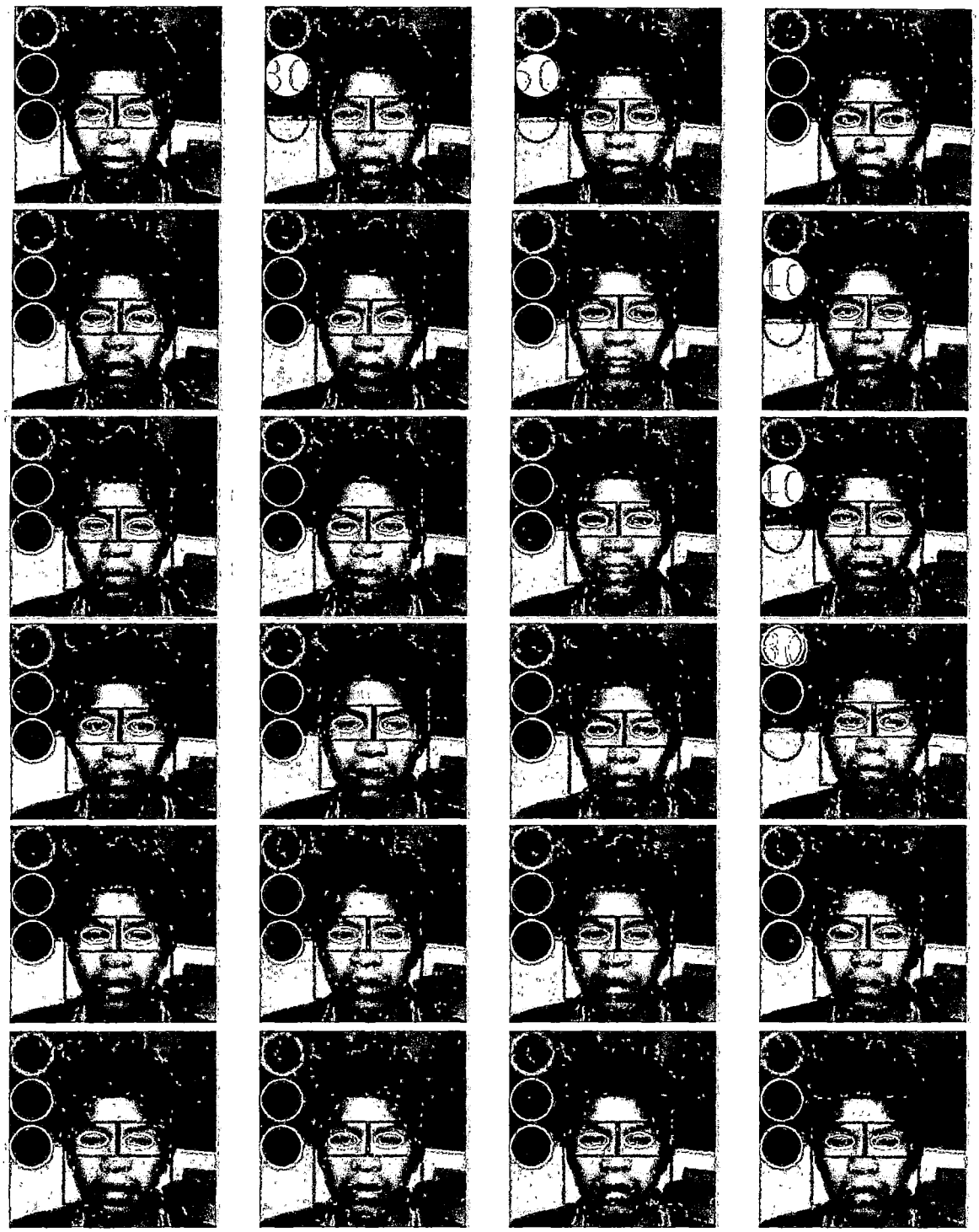
Fotograma 39	4.112	0.060
Fotograma 40	4.196	0.084
Fotograma 41	4.25	0.054
Fotograma 42	4.379	0.129
Fotograma 43	4.426	0.047
Fotograma 44	4.541	0.115
Fotograma 45	4.597	0.056
Fotograma 46	4.695	0.098
Fotograma 47	4.756	0.061
Fotograma 48	4.903	0.147
Fotograma 49	4.955	0.052
Fotograma 50	5.044	0.089
	Promedio:	**0.085**

De acuerdo a la tabla anterior el tiempo promedio en que un fotograma es evaluado para conocer el estado de los ojos es de 85ms.

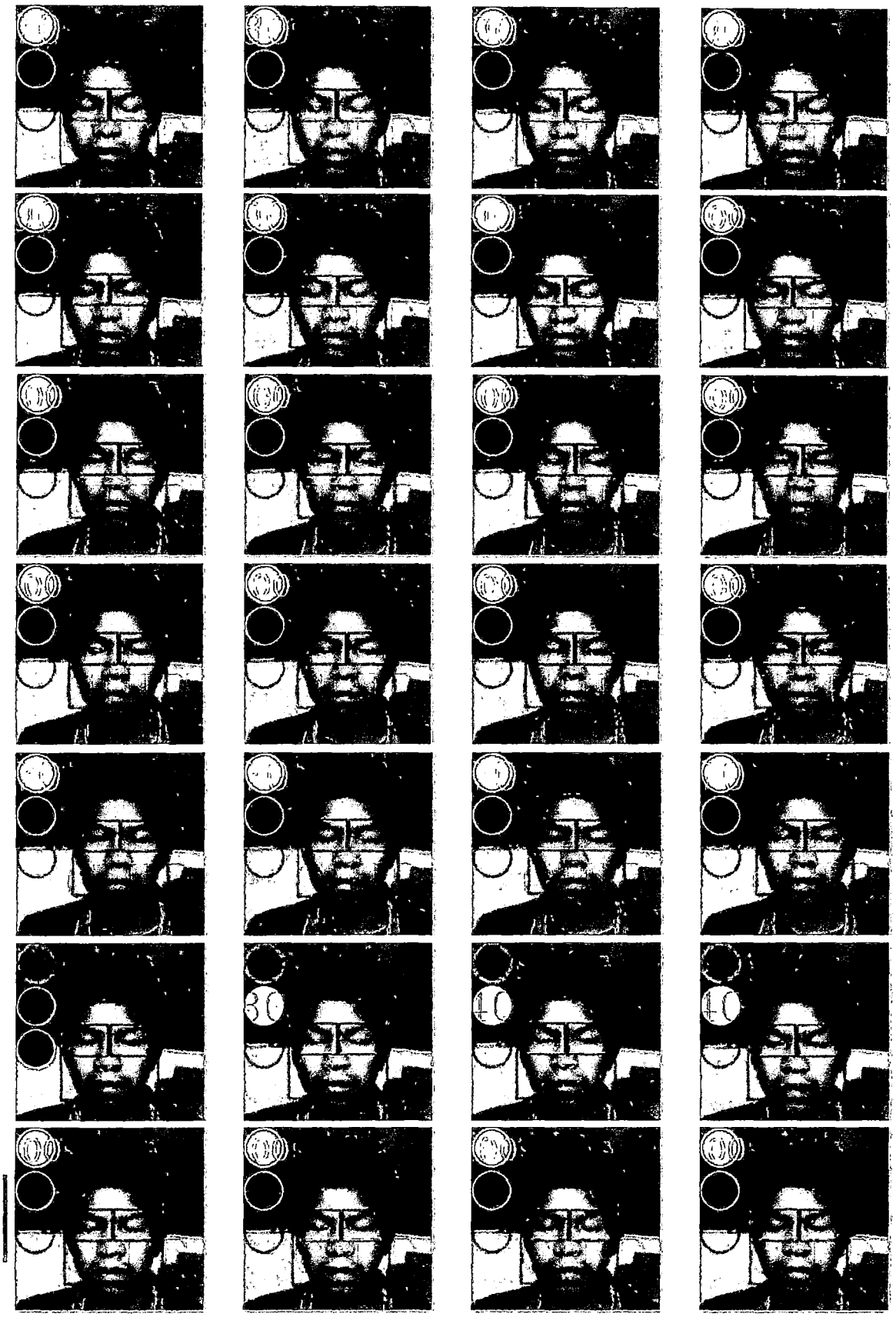
FOTOGRAMAS EVALUADOS EN LAS PRUEBAS REALIZADAS CON APERTURA DE PÁRPADOS DE 6,7,8 y 9mm

PRUEBAS REALIZADAS EN OJOS CON 6mm

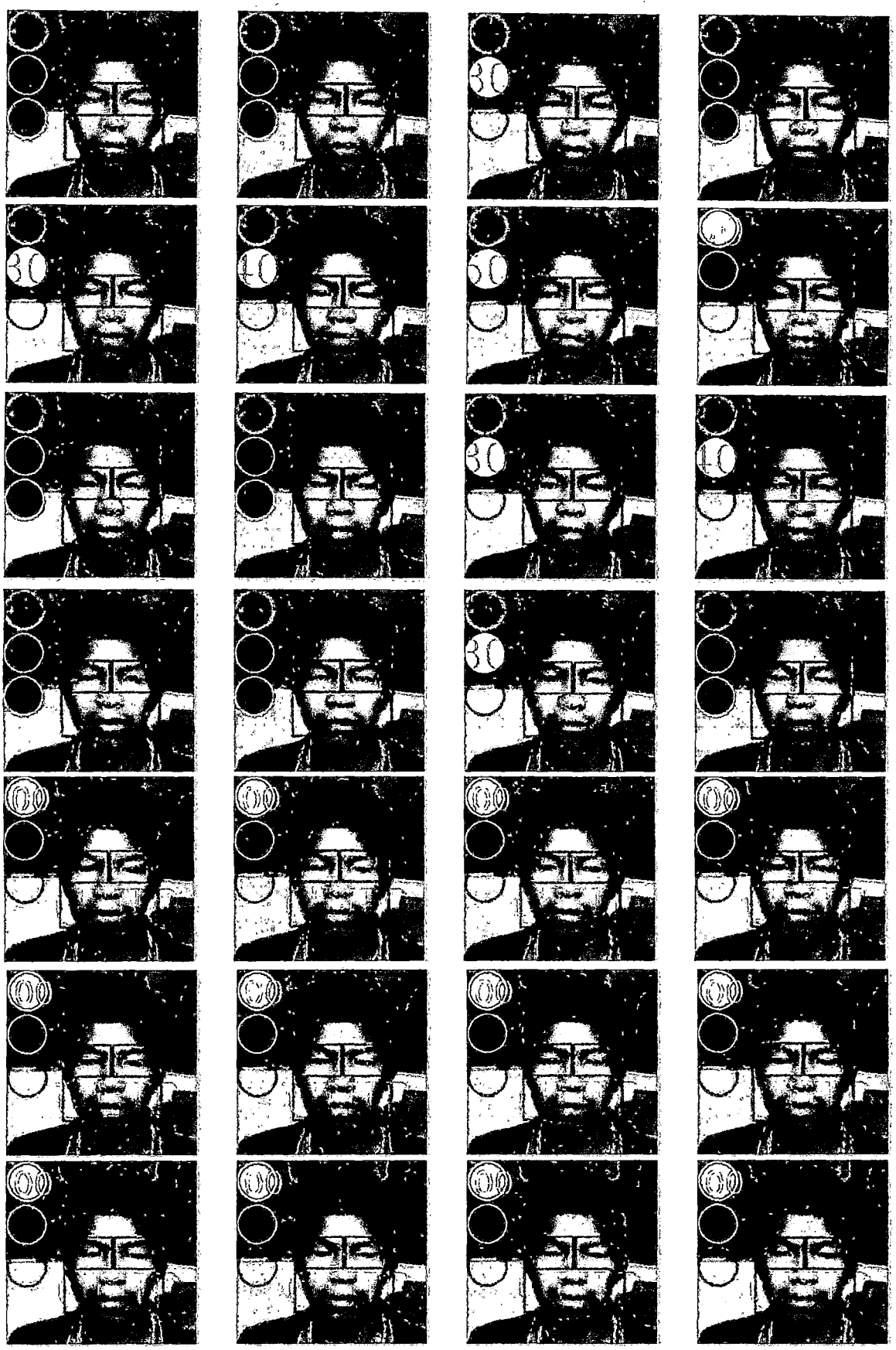
- **Algunos fotogramas clasificados como abiertos**



- **Algunos fotogramas clasificados como cerrados**



- **Algunos fotogramas clasificados falsamente como abiertos**



PRUEBAS REALIZADAS EN OJOS CON 7mm

- **Algunos fotogramas clasificados como abiertos**

- **Algunos fotogramas clasificados como cerrados**

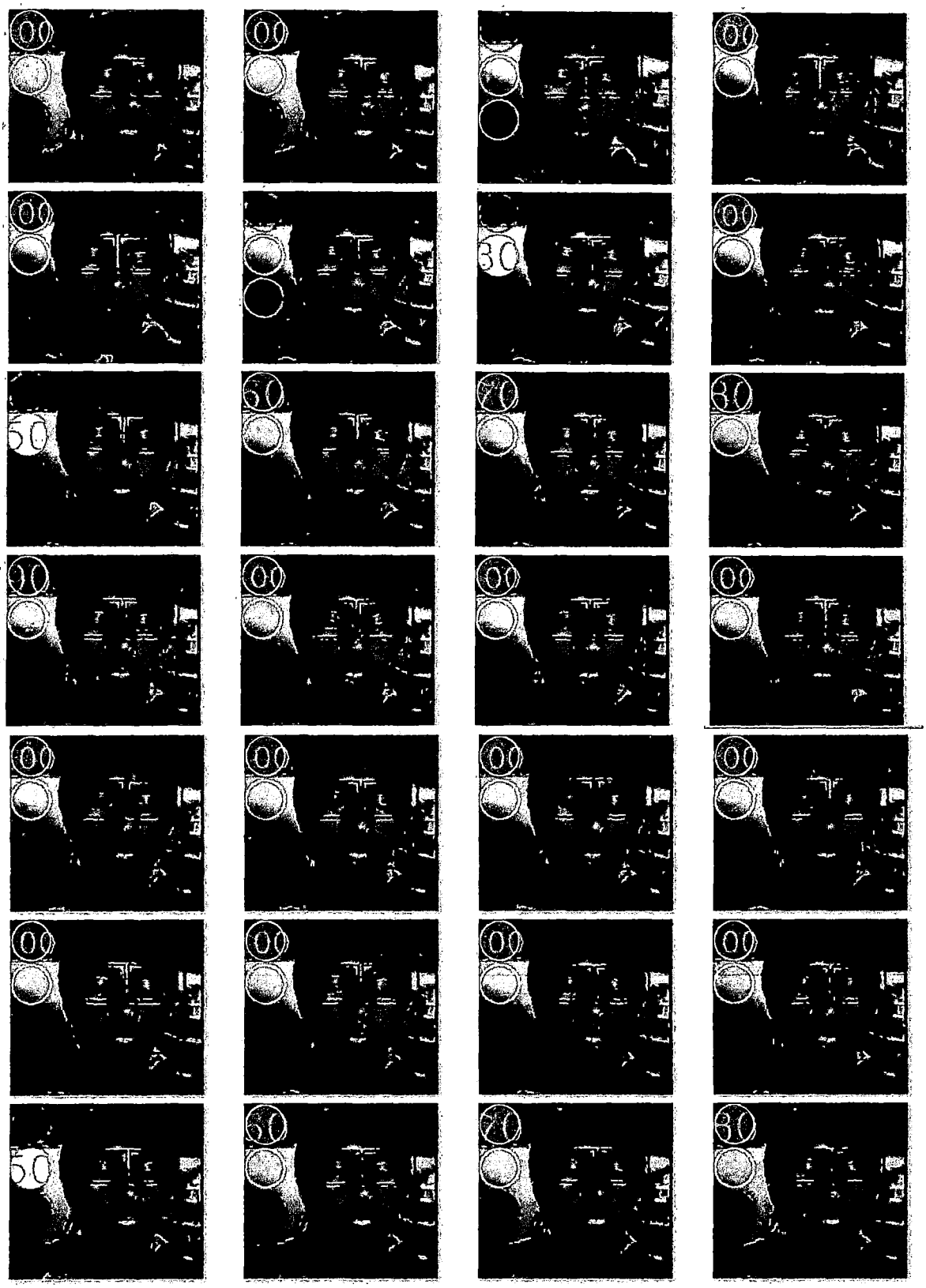
- **Algunos fotogramas clasificados falsamente como abiertos**

- **Algunos fotogramas clasificados falsamente como cerrados**

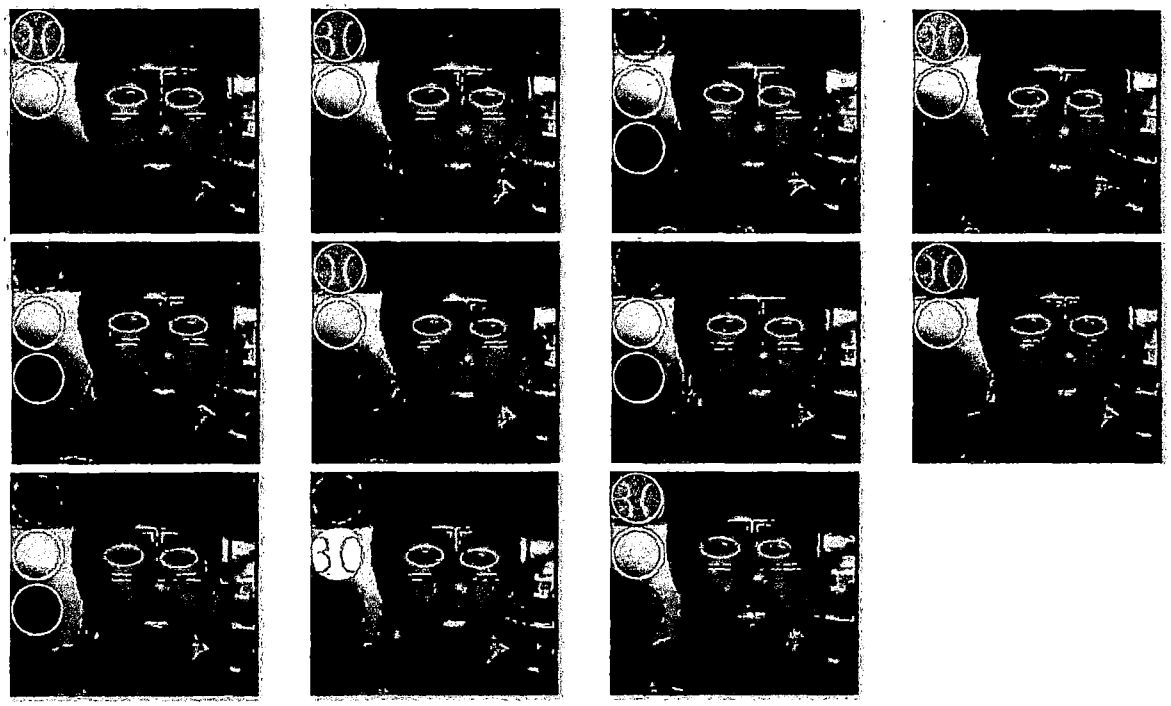
PRUEBAS REALIZADAS EN OJOS CON 8mm

- **Algunos fotogramas clasificados como abiertos**

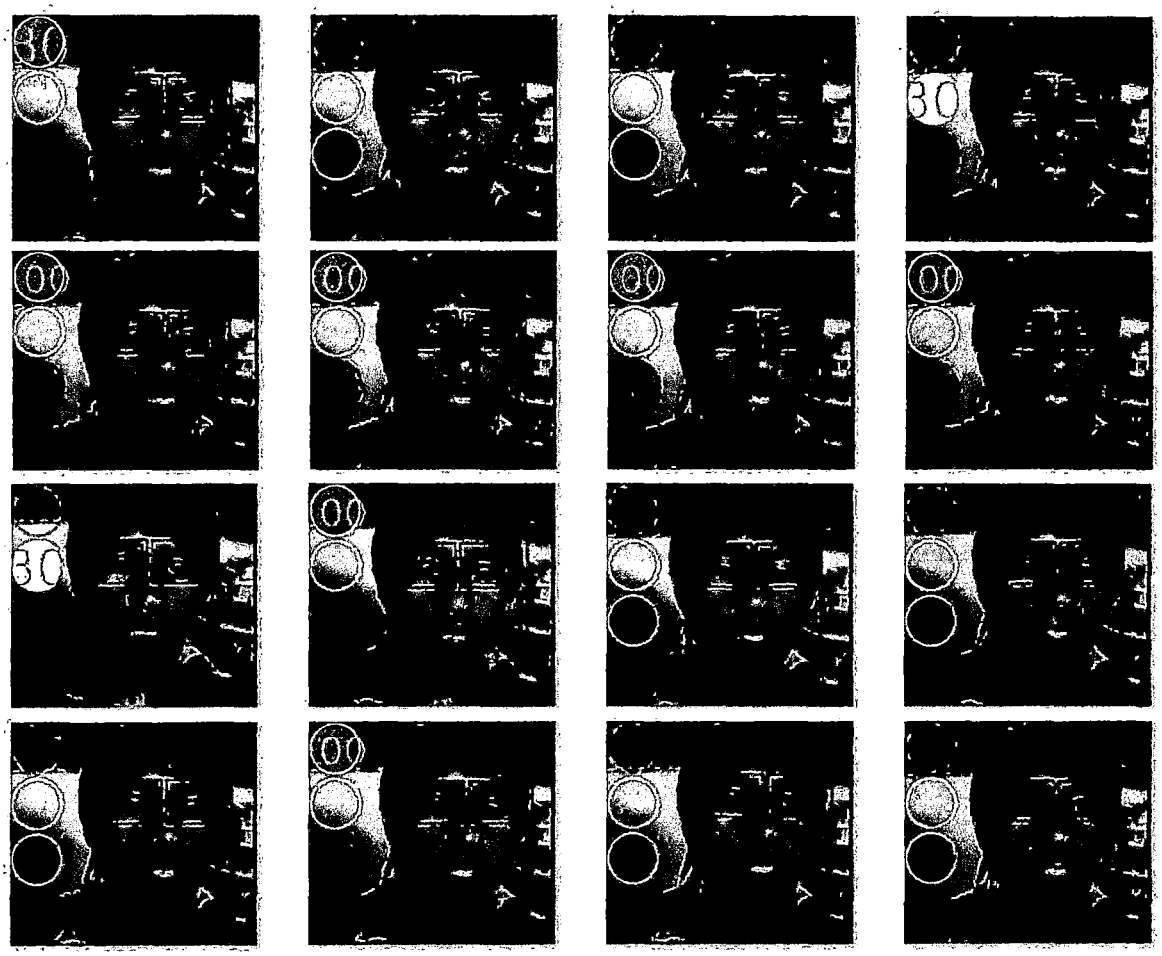
- **Algunos fotogramas clasificados como cerrados**



- **Algunos fotogramas clasificados falsamente como abiertos**

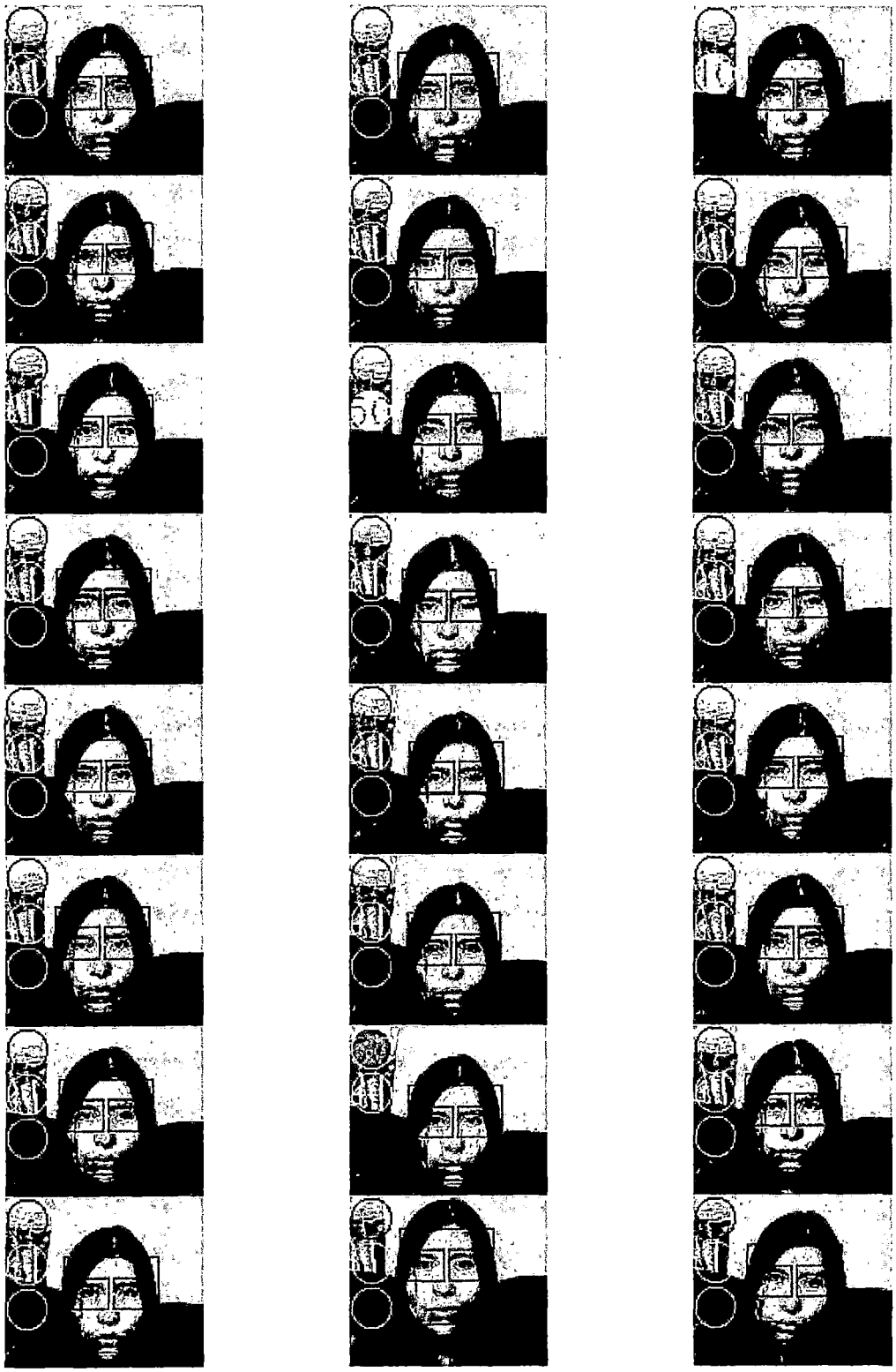


- **Algunos fotogramas falsamente clasificados como cerrados**

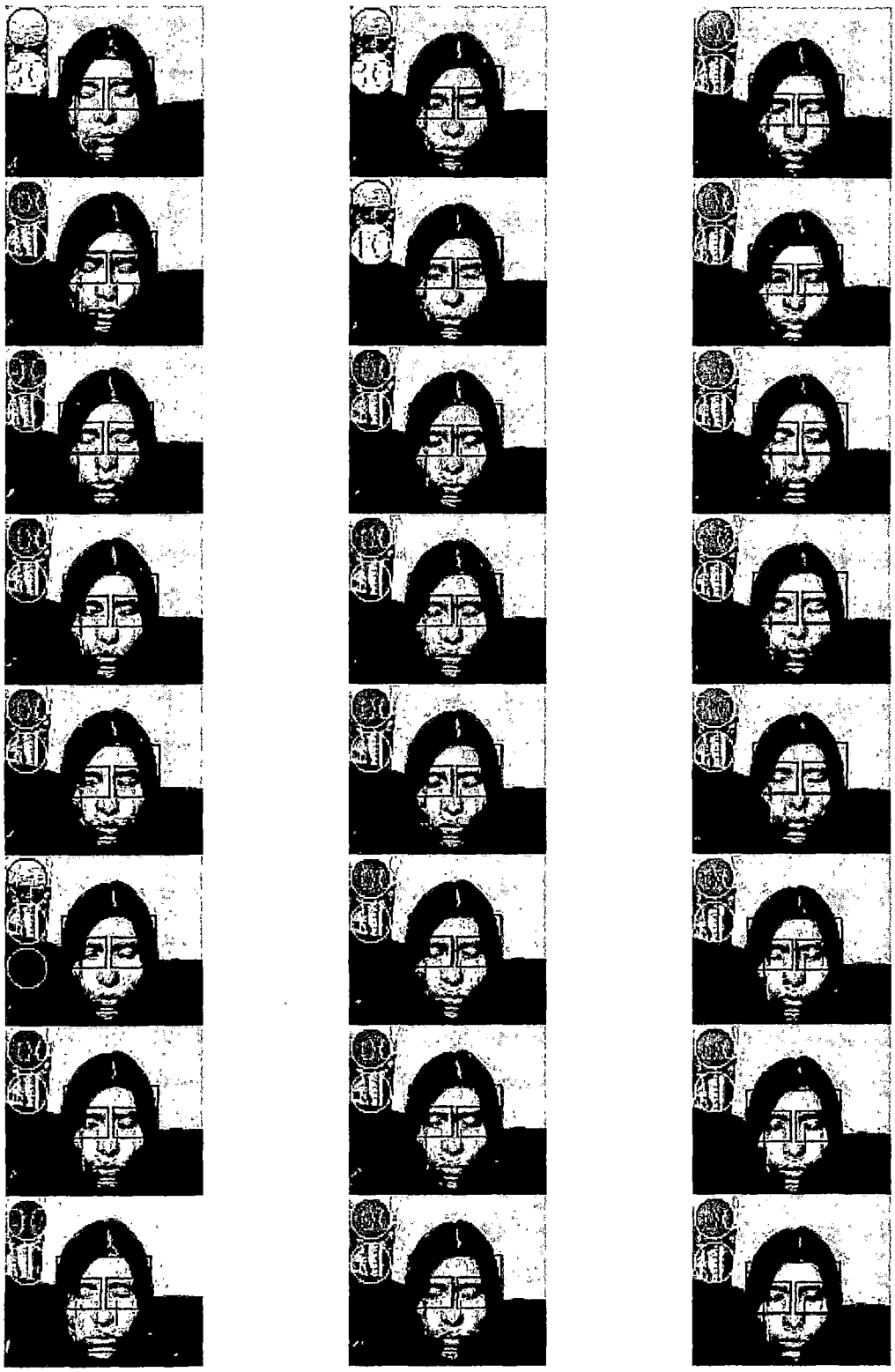


PRUEBAS REALIZADAS EN OJOS CON 9mm

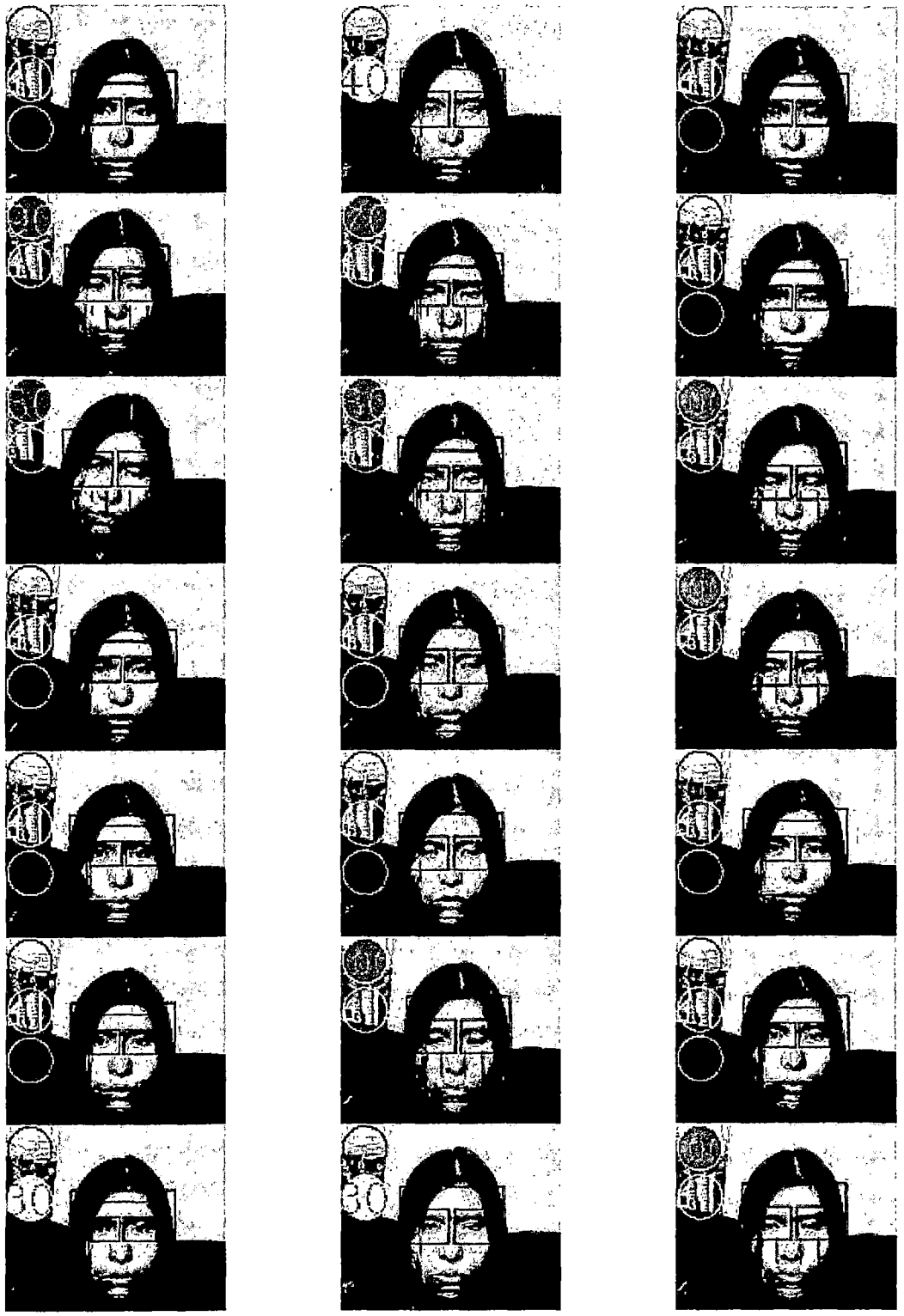
- **Algunos fotogramas clasificados como abiertos**



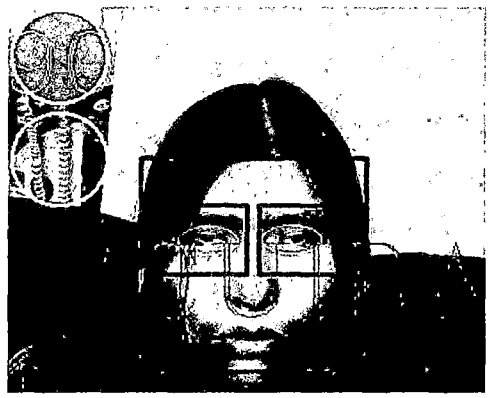
- **Algunos fotogramas clasificados como cerrados**



- **Algunos fotogramas clasificados falsamente como cerrados**



- **Algunos fotogramas clasificados falsamente como abiertos**



CCAMARA.CS

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Linq;
using System.Text;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.Util;
using Emgu.CV.CvEnum;
using System.IO;

namespace Librerias
{
publicclassCCamara
    {

//************************************************************************
//                              VARIABLES GLOBALES
//************************************************************************
privateCapture a_CapturaCamara;        //Toma imagenes de la camara como image
frames
int a_CamaraIndex;          // Indice de la camara fuente de las imagenes
string a_FileName;
bool iscamera;
Image<Bgr, Byte> a_ImagenFrame;     // Guarda una imagen capturada

public CCamara()
        {
            a_CamaraIndex = 2;
        }

public CCamara(object pCamaraIndex)
        {
try
            {
                a_CamaraIndex = int.Parse(pCamaraIndex.ToString());
                iscamera = true;
            }
catch
            {
                a_FileName = pCamaraIndex.ToString();
                iscamera = false;
            }
        }


publicbool IniciarCapturaCamara()
        {
//Inicializar la Camara Seleccionada si no ha sido iniciada aun
if (a_CapturaCamara == null)
            {
try
                {
if (iscamera) // capturar imagenes desde camara
                        a_CapturaCamara = newCapture(a_CamaraIndex);
else// capturar imagenes desde video
```

```
a_CapturaCamara = newCapture(a_FileName);
returntrue;
                }
catch (NullReferenceException excpt)
                {
returnfalse;
                }
            }
else
returntrue;
        }

publicImage<Bgr, Byte> CapturarImagenCamara()
{
if (a_CapturaCamara != null)
            {
//coger una imagen de la camara web
                a_ImagenFrame = a_CapturaCamara.QueryFrame();

//retornar la imagen capturada
return a_ImagenFrame;
            }
else
returnnull;
        }

publicvoid DesconectarCamara()
        {
if (a_CapturaCamara != null)
            {
                a_CapturaCamara.Dispose();
                a_CapturaCamara = null;
}
        }
    }
}
```

CDETECTAROJOS.CS

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.Drawing;
using System.ComponentModel;
using System.Data;
using System.Linq;
using System.Text;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.Util;
using Emgu.CV.CvEnum;
using System.IO;

namespace Librerias
{
publicclassCDetectorOjos
    {

//*************************************************************************
//                          VARIABLES GLOBALES
//*************************************************************************
privateHaarCascade haarOjos;//El Clasificador basado en algoritmo viola-jones
(detector)

//Parametros para la deteccion de ojos
privateint a_Tamanio = 10;
privatedouble a_FactorEscala = 1.1;
privateint a_VecinosMinimos = 15;
privateint a_PorcentajeAlturaInteres = 30;//20   B
privateint a_PorcentajeAlturaInteres_Superior = 20;//27   A
privateint a_PorcentajeAnchoInteres = 0;//13;  C
privateint a_PorcentajeAnchoInteres_Der = 0;//5;  E


public CDetectorOjos()
        {
            a_Tamanio = 25;
            a_FactorEscala = 1.1;
            a_VecinosMinimos = 4;
            CargarClasificadorOjos();
        }

public CDetectorOjos(object[] pParametrosRegion)
        {
            a_Tamanio = 25;
            a_FactorEscala = 1.1;
            a_VecinosMinimos = 4;

// Actualizar Region de Interes
            a_PorcentajeAlturaInteres_Superior =
int.Parse(pParametrosRegion[0].ToString());//27   A
            a_PorcentajeAlturaInteres =
int.Parse(pParametrosRegion[1].ToString());//20   B
            a_PorcentajeAnchoInteres =
int.Parse(pParametrosRegion[2].ToString());//13;  C
            a_PorcentajeAnchoInteres_Der =
int.Parse(pParametrosRegion[3].ToString());//5;  E
```

```
            CargarClasificadorOjos();
        }


publicvoid CargarClasificadorOjos()
        {
// Cargar el clasificador de ojos
            haarOjos = newHaarCascade("haarcascade_eye_2.xml");
        }

publicint DetectarOjos(Image<Bgr, Byte> p_ImagenRostro, Image<Bgr, Byte>
p_ImagenContinente, int XPosRostro,int YPosRostro)
        {
Image<Gray, byte> grayframe = p_ImagenRostro.Convert<Gray, byte>();
int mitadrostro = grayframe.Width / 2;
int desplazar = (int)(grayframe.Height * a_PorcentajeAlturaInteres_Superior / 100);
int desplazarX = (int)(grayframe.Width * a_PorcentajeAnchoInteres / 100);
int desplazarX_Der = (int)(grayframe.Width * a_PorcentajeAnchoInteres_Der / 100);


// CALCULAMOS LA REGION DE INTERES PARA EL OJO IZQUIERDO
int anchoROI = mitadrostro - (int)(grayframe.Width * a_PorcentajeAnchoInteres /
100) - (int)(grayframe.Width * a_PorcentajeAnchoInteres_Der / 100);
int altoROI = (int)(grayframe.Height * a_PorcentajeAlturaInteres / 100);
int xROI = (int)((grayframe.Width * a_PorcentajeAnchoInteres / 100));
int yROI = (int)(grayframe.Height * a_PorcentajeAlturaInteres_Superior / 100);

// DIBUJAMOS UN RECTANGULO EN LA REGION DE INTERES DEL LADO IZQUIERDO
Rectangle regionDeInteresIzq = newRectangle(xROI,yROI,anchoROI,altoROI);
            p_ImagenRostro.Draw(regionDeInteresIzq, newBgr(Color.Blue), 1);
CProcesarImagen.DibujarRectangulos(p_ImagenContinente, regionDeInteresIzq,
XPosRostro, YPosRostro, Color.Blue, 2);

// CALCULAMOS LA REGION DE INTERES PARA EL OJO DERECHO
            xROI = mitadrostro + ((int)(grayframe.Width *
a_PorcentajeAnchoInteres_Der / 100));
Rectangle regionDeInteresDer = newRectangle(xROI, yROI, anchoROI, altoROI);
            p_ImagenRostro.Draw(regionDeInteresDer, newBgr(Color.Blue), 1);

// DIBUJAMOS UN RECTANGULO EN LA REGION DE INTERES DEL LADO DERECHO
CProcesarImagen.DibujarRectangulos(p_ImagenContinente, regionDeInteresDer,
XPosRostro, YPosRostro, Color.Blue, 2);

//DETECTAR OJOS EN LA REGION DE INTERES DEL LADO IZQUIERDO Y GUARDARLA EN UN ARRAY
'MCvAvgComp[]'
            grayframe.ROI = regionDeInteresIzq;
var ojos = grayframe.DetectHaarCascade(haarOjos, a_FactorEscala, a_VecinosMinimos,
HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
newSize(a_Tamanio, a_Tamanio))[0];


int OjosDetectados = 0;
// SE DETECTO ALGUN OJO EN EL LADO IZQUIERDO
if (ojos.Length > 0)
            {
                OjosDetectados++;
// AHORA BUSCAMOS OJOS EN EL LADO DERECHO
                grayframe.ROI = regionDeInteresDer;
//DETECTAR OJOS EN LA REGION DE INTERES DEL LADO IZQUIERDO Y GUARDARLA EN UN ARRAY
'MCvAvgComp[]'
```

```
var ojosDer = grayframe.DetectHaarCascade(haarOjos, a_FactorEscala,
a_VecinosMinimos,
HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
newSize(a_Tamanio, a_Tamanio))[0];
if (ojosDer.Length > 0)
{
                    OjosDetectados++;

Rectangle rct;
PointF centro;
SizeF tamanio;
Ellipse elipse;
int ojoizquierInd = 0;
int ojoderechInd = 0;
int altura = 0;
int indice = 0;

// DIBUJAMOS EL OJO IZQUIERDO DETECTADO
foreach (var ojoizq in ojos)
                    {
//var ojoizq = ojos[0];
rct = ojoizq.rect;
if (altura < rct.Height)
{
altura = rct.Height;
ojoizquierInd = indice;
                        }
                        indice++;
                    }

                    rct = ojos[ojoizquierInd].rect;
                    rct.X = rct.X + desplazarX;
                    rct.Y = rct.Y + desplazar;
                    centro = newPointF(rct.X + (rct.Width / 2) + XPosRostro, rct.Y
+ (rct.Height / 2) + YPosRostro);
tamanio = newSizeF((float)(rct.Height * 0.6), (float)(rct.Width * 1.2));
                    elipse = newEllipse(centro, tamanio, (float)((0 * Math.PI) /
180));
                    p_ImagenContinente.Draw(elipse, newBgr(Color.Snow), 1);


// DIBUJAMOS EL OJO DERECHO DETECTADO
                    indice = 0;
                    altura = 0;
foreach (var ojoder in ojosDer)
                    {
//var ojoder = ojosDer[0];
rct = ojoder.rect;
if (altura < rct.Height)
{
                            altura = rct.Height;
ojoderechInd = indice;
                        }
indice++;
                    }

                    rct = ojosDer[ojoderechInd].rect;
                    rct.X = rct.X + desplazarX_Der + mitadrostro;
                    rct.Y = rct.Y + desplazar;
                    centro = newPointF(rct.X + (rct.Width / 2) + XPosRostro, rct.Y
+ (rct.Height / 2) + YPosRostro);
```

```
tamanio = newSizeF((float)(rct.Height * 0.6), (float)(rct.Width * 1.2));
                    elipse = newEllipse(centro, tamanio, (float)((0 * Math.PI) /
180));
                    p_ImagenContinente.Draw(elipse, newBgr(Color.Snow), 1);
}
            }
return OjosDetectados;
        }
    }
}
```

CDETECTARROSTROS.CS

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.ComponentModel;
using System.Data;
using System.Linq;
using System.Text;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.Util;
using Emgu.CV.CvEnum;
using System.IO;

namespace Librerias
{
publicclassCDetectorRostro
    {

//*************************************************************************
//                          VARIABLES GLOBALES
//*************************************************************************
privateHaarCascade haarRostros;//El Clasificador basado en algoritmo viola-jones
(detector)

//Parametros para la deteccion del rostro
privateint a_Tamanio = 25;
privatedouble a_FactorEscala = 1.3;
privateint a_VecinosMinimos = 3;

public CDetectorRostro()
        {
            a_Tamanio = 25;
            a_FactorEscala = 1.1;
            a_VecinosMinimos = 4;
            CargarClasificadorRostros();
        }

publicvoid CargarClasificadorRostros()
        {
// Cargar el clasificador de rostros
            haarRostros = newHaarCascade("haarcascade_frontalface_alt_tree.xml");
        }

publicBitmap[] DetectarRostros(Image<Bgr, Byte> p_Imagen, refint[] XPosRostro,
refint[] YPosRostro)
{
Image<Gray, byte> grayframe = p_Imagen.Convert<Gray, byte>();
a_Tamanio = (int)( grayframe.Height / 2.5);

//Detectar rostros de la imagen y guardarla en un array de 'MCvAvgComp[]'
var rostros = grayframe.DetectHaarCascade(haarRostros, a_FactorEscala,
a_VecinosMinimos,
HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
newSize(a_Tamanio, a_Tamanio))[0];

// Si se detecto algun rostro
if (rostros.Length > 0)
            {
```

```
Bitmap BmpInput = grayframe.ToBitmap();
Bitmap RostroDetectado;
Graphics FaceCanvas;

Bitmap[] RostrosDetectados = newBitmap[rostros.Length];
                XPosRostro = newint[rostros.Length];
                YPosRostro = newint[rostros.Length];
int NroRostro = 0;

foreach (var rostro in rostros)
                {
//Dibujar un rectangulo verde en los rostros detectados
p_Imagen.Draw(rostro.rect, newBgr(Color.Green), 2);
XPosRostro[NroRostro] = rostro.rect.X;
                    YPosRostro[NroRostro] = rostro.rect.Y;

//Recoger un rostro detectado
RostroDetectado = newBitmap(rostro.rect.Width, rostro.rect.Height);
FaceCanvas = Graphics.FromImage(RostroDetectado);
                    FaceCanvas.DrawImage(BmpInput, 0, 0, rostro.rect,
GraphicsUnit.Pixel);

RostrosDetectados[NroRostro] = RostroDetectado;
                    NroRostro++;
                }

return RostrosDetectados;
            }
else
// NO SE DETECTO NINGUN ROSTRO
returnnull;
        }
    }
}
```

FRMDETECTORSOMNOLENCIA.CS

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.Util;
using Emgu.CV.CvEnum;
using System.IO;
using Librerias;

namespace DeteccionSomnolencia
{
publicpartialclassfrmDetectorSomnolencia : Form
    {
CCamara a_Camara;
CDetectorRostro a_DetectorRostros;
CDetectorOjos a_DetectorOjos;
int a_NroFotogramas = 8;
int a_LongitudHistorial = 150;
ArrayList a_EstadoOjos = newArrayList();
ArrayList a_HistorialSomnolencia = newArrayList();

float aPorcentajeMaxVerde = 35;
float aPorcentajeMaxAmarillo = 70;

object[] aParametrosRegion;
object aCamaraIndex;

public frmDetectorSomnolencia()
        {
            InitializeComponent();
InicializarGrafica();
            aaction();
        }

public frmDetectorSomnolencia(object[] pParametrosRegion, object[]
pParametrosPERCLOS, object pCamaraIndex)
        {
// Iniciar Componentes
            InitializeComponent();

// Actualizar Datos
            aPorcentajeMaxVerde = int.Parse(pParametrosPERCLOS[0].ToString());
            aPorcentajeMaxAmarillo = int.Parse(pParametrosPERCLOS[1].ToString());
            a_NroFotogramas = int.Parse(pParametrosPERCLOS[2].ToString());
            a_LongitudHistorial = int.Parse(pParametrosPERCLOS[3].ToString());
            aParametrosRegion = pParametrosRegion;
            aCamaraIndex = pCamaraIndex;
            lbNroFotogramas.Text = a_NroFotogramas.ToString();
            lbDormido.Text = "Dormido >= " + aPorcentajeMaxAmarillo.ToString() +
"%";
```

```
// Procesar
            InicializarGrafica();
            aaction();
        }

publicvoid aaction()
        {
            a_Camara = newCCamara(aCamaraIndex);
            a_DetectorRostros = newCDetectorRostro();
            a_DetectorOjos = newCDetectorOjos(aParametrosRegion);
            a_Camara.IniciarCapturaCamara();
Application.Idle += DetectarEstadoSomnolencia;
}

privatevoid InicializarGrafica()
        {
for (int pointIndex = 0; pointIndex < a_LongitudHistorial; pointIndex++)
{
                chFlujoSomnolencia.Series[3].Points.AddY(aPorcentajeMaxVerde);
                chFlujoSomnolencia.Series[2].Points.AddY(aPorcentajeMaxAmarillo);
                chFlujoSomnolencia.Series[0].Points.AddY(100);
            }
        }

privatevoid DetectarEstadoSomnolencia(object sender, EventArgs arg)
        {
// INICIALIZAR CAPTURA DE IMAGENES
Image<Bgr, Byte> imagenCamara = a_Camara.CapturarImagenCamara();
if (imagenCamara != null)
            {
int[] XPosRostro = newint[] { };
int[] YPosRostro = newint[] { };
// BUSCAR EL ROSTRO EN LA IMAGEN CAPTURADA
Bitmap[] rostrosEncontrados = a_DetectorRostros.DetectarRostros(imagenCamara, ref
XPosRostro, ref YPosRostro);

if (rostrosEncontrados == null)// ROSTRO NO HALLADO ENTONCES OJOS CERRADOS
                    actualizarPERCLOS(false, imagenCamara);
else
if (rostrosEncontrados.Length > 0)
                    {
Image<Bgr, Byte> imagenRostro = newImage<Bgr, byte>(rostrosEncontrados[0]);
// BUSCAR LOS OJOS ABIERTOS
int ojosEncontrados = a_DetectorOjos.DetectarOjos(imagenRostro, imagenCamara,
XPosRostro[0], YPosRostro[0]);

if (ojosEncontrados > 1)  // AMBOS OJOS ENCONTRADOS ENTONCES OJOS ABIERTOS
                            actualizarPERCLOS(true, imagenCamara);
else// OJOS HALLADOS MENORES A 2 ENTONCES OJOS CERRADOS
                            actualizarPERCLOS(false, imagenCamara);
                    }
else// ROSTRO NO HALLADO ENTONCES OJOS CERRADOS
                        actualizarPERCLOS(false, imagenCamara);

// MOSTRAR LA IMAGEN CAPTURADA
                ib_CamImagen.Image = imagenCamara;
            }
        }
```

```
privatevoid actualizarPERCLOS(bool SeDetectoOjoAbierto, Image<Bgr, Byte>
p_imagenCamara)
        {
// Agregamos el estado del conductor en el fotograma actual
if (a_EstadoOjos.Count < a_NroFotogramas)
                a_EstadoOjos.Add(SeDetectoOjoAbierto);
else
            {
if (a_EstadoOjos.Count>0)
a_EstadoOjos.RemoveAt(0);
                a_EstadoOjos.Add(SeDetectoOjoAbierto);
            }

// Calculamos el porcentaje PERCLOS
int OjosCerrados = 0;
for (int i = 0; i < a_EstadoOjos.Count; i++)
{
if (((bool)a_EstadoOjos[i]) == false)
                    OjosCerrados++;
            }
float p_Perclos = (OjosCerrados * 100) / a_EstadoOjos.Count;

// Guardamos el PERCLOS actual para mostrar un historial de los estados de
somnolencia del conductor
if (a_HistorialSomnolencia.Count < a_LongitudHistorial)
                a_HistorialSomnolencia.Add(p_Perclos);
else
            {
if (a_HistorialSomnolencia.Count>0)
a_HistorialSomnolencia.RemoveAt(0);
a_HistorialSomnolencia.Add(p_Perclos);
            }

// Mostramos el historial de somnolencias del conductor en un chart
chFlujoSomnolencia.Series[1].Points.Clear();
for (int i = 0; i < a_HistorialSomnolencia.Count; i++)
            {

chFlujoSomnolencia.Series[1].Points.AddY(a_HistorialSomnolencia[i]);
}
// Se dibuja el estado actual en un indicador tipo semaforo
            DibujarEstadoActual(p_imagenCamara, p_Perclos);

// Guardar las imagenes para pruebas



            lbEstadoActual.ForeColor = Color.Red;
            lbEstadoActual.Text = "CERRADOS";
if (SeDetectoOjoAbierto)
            {
                lbEstadoActual.Text = "ABIERTOS";
                lbEstadoActual.ForeColor = Color.Green;
            }


/* pictureBox1.Image = CProcesarImagen.ImagenTipoBitmaps(p_imagenCamara);
            string rutaimagen = DateTime.Now.Minute.ToString() +
DateTime.Now.Second.ToString() + DateTime.Now.Millisecond.ToString() + ".jpg";

            if (SeDetectoOjoAbierto)
```

```
                pictureBox1.Image.Save("D:/abierto/"+ rutaimagen);
else
                pictureBox1.Image.Save("D:/cerrado/" + rutaimagen);*/
}

privatevoid DibujarEstadoActual(Image<Bgr, Byte> p_imagenCamara, float
perclosActual)
        {
float CentroX = (p_imagenCamara.Width/10) + 2;
float CentroY = (p_imagenCamara.Width / 10) + 2;
float Radio = p_imagenCamara.Width / 10;

PointF puntoFRojo = newPointF(CentroX,CentroY);
PointF puntoFAmarillo = newPointF(CentroX, CentroY * 3 + 3);
PointF puntoFVerde = newPointF(CentroX, CentroY*5 + 6);

CircleF circuloRojo = newCircleF(puntoFRojo,Radio);
CircleF circuloAmarillo = newCircleF(puntoFAmarillo, Radio);
CircleF circuloVerde = newCircleF(puntoFVerde, Radio);

            p_imagenCamara.Draw(circuloRojo, newBgr(Color.Red), 2);
            p_imagenCamara.Draw(circuloAmarillo, newBgr(Color.Yellow), 2);
            p_imagenCamara.Draw(circuloVerde, newBgr(Color.Green), 2);


lbPERCLOS.Text = perclosActual.ToString() + "%";
if (perclosActual < aPorcentajeMaxVerde)
            {
// EL ESTADO ACTUAL ES ALERTA O DESPIERTO
p_imagenCamara.Draw(circuloVerde, newBgr(Color.Green), 0);
                p_imagenCamara.Draw(circuloVerde, newBgr(Color.White), 2);
MCvFont fuente = newMCvFont(FONT.CV_FONT_HERSHEY_COMPLEX, 1.5, 2.5);
Point puntoFRojoTexto = newPoint((int)(CentroX - 43), (int)(CentroY * 5 + 28));
if(perclosActual<10)
                    puntoFRojoTexto = newPoint((int)(CentroX - 20), (int)(CentroY *
5 + 28));

                p_imagenCamara.Draw(perclosActual.ToString(), ref fuente,
puntoFRojoTexto, newBgr(Color.Red));
                lbPERCLOS.ForeColor = Color.Green;
}
else
            {
if (perclosActual < aPorcentajeMaxAmarillo)
                {
// EL ESTADO ACTUAL ES CANSADO
                    p_imagenCamara.Draw(circuloAmarillo, newBgr(Color.Yellow), 0);
                    p_imagenCamara.Draw(circuloAmarillo, newBgr(Color.White), 2);
MCvFont fuente = newMCvFont(FONT.CV_FONT_HERSHEY_COMPLEX, 1.5, 2.5);
Point puntoFRojoTexto = newPoint((int) (CentroX - 43), (int)(CentroY * 3 + 25));
                    p_imagenCamara.Draw(perclosActual.ToString(), ref fuente,
puntoFRojoTexto, newBgr(Color.Red));
                    lbPERCLOS.ForeColor = Color.Chocolate;
}
else
                {
// EL ESTADO ACTUAL ES DORMIDO
                    lbPERCLOS.ForeColor = Color.Red;
                    p_imagenCamara.Draw(circuloRojo, newBgr(Color.Red), 0);
                    p_imagenCamara.Draw(circuloRojo, newBgr(Color.White), 2);
MCvFont fuente = newMCvFont(FONT.CV_FONT_HERSHEY_COMPLEX, 2, 3);
```

```
Point puntoFRojoTexto = newPoint(p_imagenCamara.Width / 4,(int)(
p_imagenCamara.Height / 1.25));
p_imagenCamara.Draw("CUIDADO!!!", ref fuente, puntoFRojoTexto, newBgr(Color.Red));

fuente = newMCvFont(FONT.CV_FONT_HERSHEY_COMPLEX, 1.5, 2.5);
                    puntoFRojoTexto = newPoint((int)(CentroX - 40), (int)(CentroY +
25));
if (perclosActual > 99)
                    {
                        fuente = newMCvFont(FONT.CV_FONT_HERSHEY_COMPLEX, 1, 2);
                        puntoFRojoTexto = newPoint((int)(CentroX - 50),
(int)(CentroY + 18));
                    }
                    p_imagenCamara.Draw(perclosActual.ToString(), ref fuente,
puntoFRojoTexto, newBgr(Color.White));
                }
            }
        }

privatevoid DetenerCaptura()
        {
Application.Idle -= DetectarEstadoSomnolencia;
            a_Camara.DesconectarCamara();
        }

privatevoid frmDetector_FormClosing(object sender, FormClosingEventArgs e)
{
            DetenerCaptura();
        }
    }
}
```
