UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación

PROTOTIPO DE SISTEMA DE VISIÓN ARTIFICIAL PARA DETECTAR CONDUCTORES DESCUIDADOS

Reporte Técnico de Investigación presentado por:

Leonardo Alejandro Villanueva Betancour 132304

Requisito para la obtención del título de:

INGENIERO EN SISTEMAS COMPUTACIONALES

Dr. Javitt Higmar Nahitt Padilla Franco

Ciudad Juárez, Chihuahua 27 de noviembre de 2019

Ciudad Juárez, Chihuahua, a 7 de Noviembre de 2019

Asunto: Liberación de Asesoría

Mtro. Ismael Canales Valdiviezo
Jefe del Departamento de Ingeniería
Eléctrica y Computación
Presente.-

Por medio de la presente me permito comunicarle que, después de haber realizado las asesorías correspondientes al reporte técnico Prototipo de sistema de visión artificial para detectar conductores descuidados, del alumno Leonardo Alejandro Villanueva Betancour, de la Licenciatura en Ingeniería en Sistemas Computacionales, considero que lo ha concluido satisfactoriamente, por lo que puede continuar con los trámites de titulación intracurricular.

Sin más por el momento, reciba un cordial saludo.

Atentamente

Javitt Higmar Nahitt Padilla Franco

Profesor Investigador IIT

UACJ
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y COMPUTACIÓN

Ccp:
Coordinador del Programa de Sistemas Computacionales
Leonardo Alejandro Villanueva Betancour
Archivo

Ciudad Juárez, Chihuahua, a 6 de Noviembre de 2019

Asunto: Autorización de Publicación

C. Leonardo Alejandro Villanueva Betancour

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la impresión del documento de Prototipo de sistema de visión artificial para detectar conductores descuidados, para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Licenciado en Ingeniería en Sistemas Computacionales.

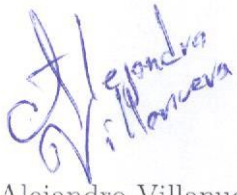
Sin otro particular, reciba un cordial saludo.

Mtra. Ivonne Haydee Robledo Portillo

Profesor Titular de Seminario de Titulación II

Declaración de Originalidad

Yo, Leonardo Alejandro Villanueva Betancour, declaro que el material contenido en esta publicación fue elaborado con la revisión de los documentos que se mencionan en el capítulo de Bibliografía, y que la solución obtenida es original y no ha sido copiada de ninguna otra fuente, ni ha sido usada para obtener otro título o reconocimiento en otra institución de educación superior.



Leonardo Alejandro Villanueva Betancour

Agradecimientos

Agradezco a Voluntarias Vicentinas de Ciudad Juárez por su paciencia, amabilidad y servicio, por darme la oportunidad de seguir estudiando, por sus platicas espirituales, por obligarme a acercarme a mi familia pues sin esta ayuda no podría haber logrado mí desarrollo personal y profesional.

Agradezco a mi madre Lucia Betancour Hernández por tenerme la paciencia y comprensión que solo una madre tiene por su hijo, por apoyarme en las cosas inapreciables, por ofrecerme una calidad de vida con su esfuerzo, gracias por darme la vida madre.

Agradezco a mi padre Sergio Villanueva Chávez por ser su hijo y mostrarme las maravillas de la vida, por su alegría y generosidad. Te agradezco por darme una madre fuerte y decidida, por darme un hogar y una familia.

Agradezco a todas las personas que me ayudaron de una forma u otra, a mis amigos, maestros y familiares que me invitaron a no desistir en esta carrera, por ser sostén al darme ánimos y al creer en mí, gracias a todos.

Dedicatoria

Dedico el presente proyecto de investigación a mi querido padre Sergio Villanueva Chávez por instruirme en el estudio de las ciencias y artes, contribuyendo en mi formación académica y personal en todo momento, apoyándome incondicionalmente en vida en cada meta y objetivo propuestos, esto y más te los debo a ti.

Dedico este proyecto de investigación a todas las personas que creyeron en mí y me brindaron su ayuda ya sea económica o moralmente.

Índice general

1. Planteamiento del Problema **4**

1.1. Antecedentes 4

1.2. Definición del problema 9

1.3. Objetivos 10

1.3.1. Objetivos específicos 10

1.4. Justificación 10

1.5. Alcances y limitaciones 12

1.5.1. Alcances 12

1.5.2. Limitaciones 12

2. Marco teórico **14**

2.1. Visión artificial 14

2.1.1. Adquisición de imágenes para un sistema de visión artificial 15

2.1.2. La óptica 17

2.1.3. Software para visión artificial 18

2.1.4. Áreas de aplicación de la visión artificial 20

3. Desarrollo del Proyecto **21**

3.1. Producto propuesto 22

3.2. Análisis de requisitos 23

3.2.1. Software utilizado 23

3.2.2. Hardware y sus configuraciones 25

3.2.3. Definición de descuido y modelos en Keras 29

3.3. Diseño del prototipo del sistema 31

3.3.1. Obtención de vídeos 32

3.3.2. Creación del set de datos 33

3.3.3. Compilación del modelo 35

3.3.4. Pruebas de exactitud 37

3.3.5. Implementación del sistema de prototipo 37

3.4. Diseño del prototipo del programa 40

3.4.1. Obtención de vídeo de prueba 41

3.4.2. Generar Set de datos 43

3.4.3. Compilar un modelo 44

3.4.4. Pruebas del modelo 45

3.4.5. Implementación del modelo 47

3.5. Codificación del prototipo de sistema 49

3.5.1. Obtención de vídeo de prueba 51

3.5.2. Generar Set de datos 53

3.5.3. Compilar un modelo 54
3.5.4. Pruebas del modelo 58
3.5.5. Implementación del modelo 64
3.6. Resultados de la codificación 67
3.6.1. Obtención de vídeo de prueba 67
3.6.2. Generar Set de datos 69
3.6.3. Compilar un modelo 74

4. Resultados y Discusiones **77**
4.0.1. Pruebas del modelo 77
4.1. Implementación del prototipo de sistema 80
4.1.1. Evaluación de conductores descuidados 81
4.1.2. Evaluación de conductores cuidadosos 82
4.1.3. Resultados finales 83
4.2. Modelo final 86

5. Conclusiones **87**
5.1. Con respecto al objetivo de la investigación 87
5.2. Recomendaciones para futuras investigaciones 89

Bibliografía **90**

Índice de figuras

2.1. Distancia focal y los ángulos de visión [1]. 18

3.1. Desarrollo en cascada. 22

3.2. Configuración de la cámara web C920 en el automóvil. 27

3.3. Visión dentro del automóvil. 28

3.4. Clasificación binaria de imágenes. 29

3.5. Diseño del prototipo del sistema. 31

3.6. Estructura de archivos. 32

3.7. Diseño de una red neuronal convolucional. 36

3.8. Áreas de trabajo. 38

3.9. Alarma visual. 39

3.10. Conjunto de datos: Numero 0. 70

3.11. Conjunto de datos: Numero 1. 71

3.12. Conjunto de datos: Numero 2. 71

3.13. Conjunto de datos: Numero 3. 72

Índice de tablas

3.1. Resultados de obtención de vídeo de prueba 67

3.2. Resultados de generar set de datos 69

3.3. Clasificación y filtrado de la información 73

3.4. Resultados de la compilación 76

4.1. Test 0: Resultados de las pruebas 78

4.2. Test 1:Área de trabajo: 78

4.3. Test 1: Resultados de las pruebas 79

4.4. Test 1: Fotogramas clasificados como conductores descuidados 79

4.5. Área de trabajo: Implementación 80

4.6. Resultados de la implementación 81

4.7. Evaluación de conductores descuidados 81

4.8. Evaluación de conductores cuidadosos 82

4.9. Resultados finales: Promedio de modelos 83

4.10. Resultados finales: Promedio exactitud respecto a los vídeos 84

Resumen

En la actualidad, los accidentes de tráfico son muy frecuentes, para contrarrestarlos se crean reglamentos de tránsito y vialidad, mientras que los fabricantes de vehículos emplean sus desarrollos tecnológicos en mantener a salvo los individuos dentro del automóvil. Aun así, los accidentes viales siguen ocurriendo, la razón es que mayoría de los accidentes viales son a causa de los errores del conductor al manejar un vehículo, siendo más precisos la conducción descuidada puede desencadenar eventos fatídicos. Este proyecto de investigación se ha desarrollado un prototipo de sistema que por medio de la visión artificial evalué las acciones del conductor. Este prototipo consta de diferentes partes usando la librería de Python-OpenCV se efectuó una configuración mono ocular de visión artificial, de manera subsecuente, por medio de redes neuronales convolucionales se crearon modelos de clasificación de imágenes, detectando cuando un conductor presta atención al entorno fuera del vehículo o dentro de este. Emitiendo alarmas visuales, por medio de OpenCV, y auditivas al momento de detectar a un conductor descuidado. Finalizando con el proyecto se ejecutaron los modelos generados, para obtener una conclusión positiva en los resultados.

Palabras clave: Redes neuronales convolucionales, aprendizaje profundo, clasificador de imágenes, conducción de vehículos.

Abstract

Currently, traffic accidents are very frequent, to counteract them are created traffic and road regulations, while vehicle manufacturers use their technological developments to keep individuals inside the car safe. Still, road accidents continue to occur, the reason is that most road accidents are due to driver errors when driving a vehicle being more precise, careless driving can trigger fateful events. This research project has been developed a prototype system that through artificial vision i evaluated the actions of the driver. This prototype consists of different parts using the Python-OpenCV library a mono ocular vision configuration was made, subsequently, through convolutional neural networks, image classification models were created, detecting when a driver pays attention to the environment outside or inside the vehicle. Emitting visual alarms, through OpenCV, and auditory when detecting a careless driver. Finishing with the project the generated models were executed, to obtain a positive conclusion in the results.

Keywords: Convolutionary neural networks, deep learning, image classifier, vehicle driving.

Introducción

En la actualidad, los vehículos terrestres implementan múltiples tecnologías que proporcionan seguridad, y permiten conservar la integridad física de los conductores y pasajeros en siniestros viales, evitando las lesiones o hasta la muerte. Aun así, los daños a la vía pública, a los vehículos y los gastos médicos, son pérdidas económicas que pueden llegar a ser muy altas, si no se cuenta con un seguro adecuado.

Para evitar siniestros viales se toman medidas como: un reglamento de vialidad y tránsito respecto a la localidad, se mantienen las vías de transporte terrestre (calles, avenidas, carreteras, entre otros.) en óptimas condiciones; las organizaciones relacionadas con el transporte asumen compromisos con la seguridad vial de sus trabajadores y pasajeros o carga; las compañías de automóviles implementan sistemas de seguridad que pueden llegar a manipular el vehículo en situaciones específicas. Aun así, los accidentes de tráfico siguen siendo muy frecuentes.

La mayoría de los accidentes viales son los errores que el conductor comete al manejar, un descuido de segundos puede desencadenar un siniestro vial, por lo tanto, las organizaciones de transporte capacitan a sus trabajadores y les imponen reglas estrictas para evitar la imprudencia frente al volante, mientras que, los fabricantes de automóviles desarrollan vehículos para reducir el error del conductor.

Entre las tecnologías desarrolladas que se enfocan en verificación y control de calidad, la visión artificial es una herramienta innovadora, en la cual, se obtiene información del entorno

de manera eficiente por medio de una cámara y se realizan procesos con software especializado, estas tareas de verificación pueden llegar a ser exhaustivos para una persona cuando se desempeña por varias horas. El uso de visión artificial es muy versátil para los sistemas que requieren precisión, repetitividad y constancia, el software puede reconocer objetos específicos con los que el conductor interactúa y además identificar los gestos corporales del conductor, con los cuales, se llegarían a evaluar por medio de sus gestos corporales, posición de cabeza, brazos, piernas, entre otros. y deducir el estado del conductor (descuidado o atento) al manejar el vehículo.

En el desarrollo de este proyecto se basa en crear un sistema de visión artificial que detecte al conductor que se pueden traducir en atento o descuidado al manejar su automóvil, por ejemplo, al tomar el celular mientras se maneja, emitir inmediatamente alertas sonoras y visuales para corregir su conducta. Para llegar a concluir el proyecto en los siguientes capítulos se describen el proceso llevado a cabo.

En el capitulo 1 llamado, "Planteamiento del problema", como su nombre lo indica se presentan antecedentes, definición del problema, objetivos generales y específicos del proyecto, justificación y finalizando con los alcances y limitaciones que se enfrenta el proyecto.

Siguiendo con el capitulo 2, "Marco teórico" en el que describirán las tecnologías y software principal a usar en este proyecto, así como sus limitaciones, características y para finalizar ejemplos de su uso.

Continuando el capitulo 3, "Desarrollo del proyecto", definiendo la metodología de trabajo como cascada y sus procesos de desarrollo en etapas, como: Análisis de requisitos, diseño del prototipo del sistema y programa, codificación del prototipo del sistema y para finalizar los objetos resultantes de las etapas previas.

Después de eso, el capitulo 4 "Resultados y Discusiones", los modelos generados anteriormente se ejecutan en pruebas de evaluación del modelo, dando resultados de efectividad en sus pruebas e implementación.

Para finalizar con el capitulo 5, "Conclusiones", culminando el ciclo de vida de este proyecto de investigación, se mencionan las partes mas relevantes del funcionamiento del sistema, se determina si el objetivo general fue cumplido y para terminar se da recomendaciones para los próximos investigadores.

Capítulo 1
Planteamiento del Problema

Conforme una persona adquiere experiencia al manejar ya sea por trabajo o por transporte personal, esta genera confianza en sí mismo y en el comportamiento de otros conductores, formando un margen de error, por ejemplo: Cuando el conductor se detiene en un semáforo tiene que esperar, mientras tanto, puede escuchar la radio, revisar documentos, leer y enviar mensajes, contestar llamadas, maquillarse, buscar objetos, entre otros., por lo cual se genera una desconexión del individuo y su entorno, entonces cuando llegue el momento de avanzar el conductor puede simplemente acelerar sin observar detalladamente propiciando un accidente; Otro ejemplo, al entrar a un estacionamiento público, el conductor (mientras el vehículo sigue en movimiento) toma su celular envía y responde mensajes, entonces entre la distracción puede llegar a golpear a un vehículo. Por lo tanto, es importante que el conductor este centrado en el manejo, desde que se enciende hasta que se apaga el automóvil.

1.1. Antecedentes

En las últimas décadas, se ha dado un impulso notable en la seguridad de las personas ya sea dentro o fuera del automóvil, esto se debe al constante cambio en los estándares de seguridad, el programa de Evaluación de Vehículos Nuevos para América Latina y el Caribe (Acrónimo del inglés, Latin NCAP) califica de cero a cinco estrellas la protección que proporciona el

vehículo al conductor y los pasajeros, siendo la más alta mejor, entonces un auto que en el año 2015 obtuvo cinco estrellas probablemente en este año 2018 no lo haría, pues a partir del año 2016 los protocolos de evaluación se expandieron y son más exigentes [2].

En el año 2018, en Ecuador se realizó un estudio a la población de conductores de vehículo liviano y cuyo tipo de licencia no es profesional [3], entre el problema se descubre que gran porcentaje de personas desconocen si sus vehículos tienen alguna tecnología que ayude a disminuir el número de accidentes, tales como, Sistema antibloqueo (ABS), Sensor eléctrico de estabilidad (EPS), Sensor de fatiga, Sensor de freno de emergencia, Sensor adaptativo de velocidad, Sensor de alcoholemia, entre otros. Al finalizar el estudio se concluyó que la incorporación de sensores en vehículos particulares ayuda de una forma segura y confiable a evitar y disminuir los accidentes de tránsito.

En México, en [4] el INEGI presenta un proyecto llamado "Estadística de accidentes de tránsito terrestre en zonas urbanas y suburbanas (ATUS)", donde muestra de manera cuantitativa la incidencia de percances viales, con un total de más de trescientos setenta mil accidentes de tránsito en el territorio mexicano, lo cual provocó cuatro mil seiscientos un víctimas fatales, solamente en el año 2015. El objetivo de ATUS (desde 1997 hasta la actualidad) es generar información cada año sobre los siniestros del transporte de tipo terrestre a nivel nacional, así como contribuir con la planeación y organización del transporte.

En [5], la Comisión Nacional de Seguridad (CNS) proporciona un artículo sobre las causas de un accidente, en las carreteras del territorio mexicano en el 2015, las cuales son las siguientes: del 80 % de las incidencias corresponden al conductor, seguido del 7 % al vehículo, 9 % a los sucesos de agentes naturales y solo el 4 % de casos al camino.

Los factores humanos son la causa del mayor porcentaje de accidentes de tránsito [6], siendo el primer factor que interviene, al ser la persona la que toma las decisiones sobre el movimiento del vehículo, al mismo tiempo, es el conductor el responsable de comprarse un vehículo, decidir conducirlo, cuándo revisar su funcionamiento, e incluso desplazarse con el

mismo u optar por otras vías de transporte. Entonces, las principales causas en las que el conductor puede ocasionar un accidente vial debido al factor humano son las siguientes [5]: Conducir a exceso de velocidad, salud física del conductor, conducir con cansancio o sueño, conducir bajo los efectos de cualquier sustancia que inhiba al conductor de sus capacidades, efectuar movimientos imprudentes y de omisión por el conductor.

Entre los sistemas para resolver el factor humano respecto al conductor, se encuentra la monitorización de la fatiga, en [7] la Universidad de Alcalá, se realizó un proyecto para detectar el nivel de fatiga del conductor, por medio de variables biológicas, en concreto en la variabilidad del ritmo cardiaco (HRV) que refleja las interacciones del sistema nervioso. Se desarrolló un hardware para la adquisición de datos, en los cálculos del HRV, se tuvo en cuenta la presión de la mano cuando se agarra el volante, la temperatura de la cabina y el exterior.

Para monitorizar la fatiga por medios visuales, en [8], se desarrolló en la Universidad Nacional de Loja, una aplicación Android para la detección de adormecimiento de un conductor utilizando visión artificial, está aplicación es desarrollada con el kit de desarrollo nativo (acrónimo del inglés, NDK) de este modo se podía acceder directamente a todos los recursos de un dispositivo móvil en tiempo real. El proceso de detección de somnolencia consta de la detección del rostro del conductor, seguido de detectar los ojos y finalmente obtener el estado del conductor por medio de perclos, relación de tiempo en que los ojos están cubiertos por el párpado, el cual es noventa por ciento más eficaz en comparación a otros métodos de obtención del indicio de adormecimiento por medio de visión artificial.

Siguiendo con la monitorización de la fatiga por medios visuales se tiene en [9], el Instituto Nacional de Astrofísica, Óptica y Electrónica Tonantzintla, Puebla, México, se desarrolló un algoritmo para la segmentación de las regiones de piel. El objetivo de este proyecto era el estudio y el análisis de los diferentes espacios de color (RGB, YCrCb, HSI y LUX), para aplicarlo a un sistema de visión artificial que detecte la fatiga percibiendo el pestañeo

prolongado en los ojos del conductor, con la cámara detecta el rostro, y el procesado de imagen del algoritmo, la unión del dos planos Cr del espacio YCrCb y el plano U' del espacio LUX, se detecta claramente el área de los ojos. En este trabajo se concluye que la iluminación fue un factor que limitaba el algoritmo, en lugares con muy poca o nula luz dejaba de funcionar.

En [10], en la ciudad de Madrid, se desarrolló un sistema para detectar peatones por medio de visión artificial con el uso de una cámara estéreo en ambientes complejos. El objetivo de este proyecto consiste en detectar a los peatones de forma rápida y eficaz para el uso en tiempo real, y evitar lesiones por atropello. En el desarrollo del proyecto la cámara estéreo que capta el entorno frente al automóvil, por medio de los histogramas de gradientes orientados (HOG) se localiza el peatón en tiempo real, aunque dependiendo de la escena, está podía ser más complicada y por ende tardada al procesar. Al finalizar el desarrollo en el periodo de pruebas, el sistema puede detectar múltiples personas a una distancia de treinta y dos punto cinco metros, lo cual, da suficiente tiempo para medidas preventivas.

El fabricante de vehículos Volvo *Car Corporation* ha estado desarrollando un sistema de seguridad *intellisafe* [11], el cual es un sistema con sensores de proximidad, de inclinación, de radar, cámaras frontales de visión artificial para detectar (peatones, ciclistas, autos, entre otros.) y saber su distancia. Los vehículos de nuevo modelo y de gama alta implementan está tecnología, entre sus funciones, protección de peatones y ciclistas: detecta si hay personas o ciclistas en dirección al vehículo, ya sea de manera frontal o lateral-frontal. Al detectarlos se activa con una luz en el parabrisas para ayudar a evitar accidentes, e incluso frenando automáticamente para proteger al peatón de ser atropellado.

La visión artificial es una manera de obtener información del entorno en grandes cantidades, en las últimas décadas se ha ido popularizando el uso de sistemas con visión artificial, esto se debe a que el ser humano puede tardar varios segundos en realizar la identificación de un objeto, en cambio para una tarea relativamente simple y repetitiva una computadora lo haría de manera muy rápida, de tal manera que se optimizarían los procesos. Un claro

ejemplo esta en [12], en este documento se explica el desarrollo de un sistema para la reconocimiento del nivel del líquido y colocación de las botellas, los objetivos del proyecto eran el reconocimiento mediante medidas pre determinados, y a su vez, con el uso de un operador robótico, colocar las botellas previamente aprobadas por el sistema hasta situarlas en su respectiva caja de transporte.

Las tareas simples y repetitivas son estresantes para una persona y se puede llegar a tener un bajo desempeño conforme se desarrollan las labores, la visión artificial es una gran ayuda para la comprensión del entorno. En la carrera de lanzar un vehículo guiado independiente muchos fabricantes han implementado la visión artificial y otros sensores para crear nuevas tecnologías en sus vehículos. Un pionero y líder en sistemas de asistencia al conductor basados en la visión por computadora es Volvo *Car Corporation* en asociación con *Delphi Corporation* en el año 2007, presentan los sistemas novedosos *Driver Alert Control* (DAC) y Mobileye [13].

En los siguientes párrafos se describirán los sistemas desarrollados e implementados por Volvo *Car Corporation* [13] que posteriormente se nombraría *intellisafe* [11] para el público en general, las funciones son las siguientes: *Driver Alert Control* (DAC), monitorea una combinación de vehículos, camino y parámetros de manejo y evalúa si el vehículo está siendo manejado de manera controlada o no controlada (conductor soñoliento); *Mobileye*, es un sistema de visión artificial colocando una cámara en visor central con vista hacia fuera del automóvil, este sistema emboca las siguientes tecnologías:

- *Lane Departure Warning* (LDW), este sistema que detecta si el conductor se está desviando de su carril lentamente, por medio de visión artificial detecta las líneas blancas de las calles, alerta con vibraciones en el volante y suavemente maniobra para volver al carril.

- *Collision Warning with Auto Brake* (CWAB), es un sistema de alerta de colisión con freno automático, la cual, tiene otras funciones en *intellisafe*, como por ejemplo, *Adap-*

tive Cruise Control que mantiene la distancia de seguridad con el coche que le precede.

Aunque el desarrollo de *intellisafe* [11] por parte de Volvo *Car Corporation* no es conducción autónoma real, estos sistemas pueden llegar detectar su entorno, como vehículos, ciclistas, peatones, distancias entre el vehículo y el objeto o lugar, entre otros. y "manejar" un automóvil de manera controlada y en situaciones específicas, un ejemplo es *Park Assist Pilot*, el cual es un sistema automático para estacionar el vehículo, con el único requerimiento situar en paralelo al aparcamiento, activar el sistema y soltar el volante, el automóvil se ocupará de la maniobra automáticamente, solo avisando cuando frenar.

Un grupo de investigadores ponen a prueba un vehículo autónomo en un entorno real, en [14] se ejecutó una prueba desafiante para las tareas de detección de objetos estereoscópicos. El objetivo del proyecto es reducir este sesgo proporcionando puntos de referencia desafiantes con dificultades novedosas para la comunidad de visión artificial y en conclusión cumplen con su objetivo de proporcionar más información que la compilada en los puntos de referencia hasta el año 2012, en la detección de objetos y su ubicación.

1.2. Definición del problema

Hoy en día las personas que conducen un vehículo son descuidados en el desarrollo de esta tarea, esto se debe a la confianza que genera al conducir diariamente por las mismas calles. Las empresas de transporte en sus unidades terrestres optan por colorar cámaras de vigilancia dentro y fuera del de los vehículos con el único fin de proporcionar seguridad: al personal, a los pasajeros, al vehículo y mercancía, pero estas cámaras en el caso de un accidente vial, estos dispositivos se utilizan con propósitos de recopilar evidencia, mediante las grabaciones, se buscan principalmente a los responsables para enjuiciarlos, generalmente se pone en duda al conductor si sus acciones fueron las correctas al manejar la situación. Por lo tanto, estas cámaras y ningún desarrollo tecnológico ayudan a un conductor a prevenir sus descuidos para

tener un mejor desempeño en la realización de tareas como el manejo.

1.3. Objetivos

Desarrollar un sistema en el que se detecte por medio de visión artificial, los gestos del cuerpo que se pueden traducir en un descuido por parte del individuo, y se activen alarmas visuales (por medio de OpenCV) y sonoras inmediatamente se incurra en el acto de descuido.

1.3.1. Objetivos específicos

1. Desarrollar un sistema de visión artificial que reconozca el rostro del conductor.
2. Desarrollar funciones de reconocimiento de los gestos corporales a estar frente al volante en un vehículo.
3. Desarrollar funciones de evaluación de los gestos corporales de descuido.
4. Desarrollar e implementar la emisión de alertas sonoras y visuales cuando ocurra el descuido.
5. Implementar el sistema en un vehículo y evaluar los resultados generados por medio de pruebas.

1.4. Justificación

La seguridad que proporcione un vehículo es muy importante para el conductor, pasajeros y peatones, las empresas fabricantes de automóviles han adoptado múltiples formas para proteger a los usuarios dentro y fuera de sus vehículos, tanto así que Volvo *Car Corporation* propone que en el año 2020 las personas no resulten lesionadas ni heridas de gravedad al

volante de su reciente producción de vehículos; Estos automóviles nuevos tienen que ser de la gama alta o premium, su costo genera una barrera para la gran mayoría de conductores de países en vías de desarrollo, como lo es México, dejando prácticamente el uso exclusivo de estos vehículos a países de primer mundo.

En México, en [4] las estadísticas que genera el INEGI concluyen que el principal factor en accidentes viales es el humano, principalmente el conductor pues él decide sobre su automóvil (acelerar, frenar, uso de luces direccionales, mantenimiento, entre otros.), entonces se entiende que, para eliminar este factor se tiene que excluir la interacción humana con el vehículo, la solución sería la conducción autónoma.

En la carrera de las compañías fabricantes de automóviles por conseguir el vehículo con conducción totalmente autónoma se tienen grandes avances, aunque, no accesibles para todas las personas por cuestiones económicas, por lo tanto, este proyecto pretende dar un enfoque directo a las acciones del conductor y persuadirlo de sus distracciones al conducir, teniendo como efecto reducir el factor humano.

La realización de este proyecto pretende generar nueva tecnología, la cual tendría un impacto significativo en tareas como la conducción de vehículos terrestres, ya sean, transporte de pasajeros, de carga, urbano, interurbano, público, privado o escolares; También teniendo un impacto en la conservación de la integridad física y psicológica de las personas, además del valor económico de los vehículos y vías de transporte como lo son calles, avenidas, carreteras, entre otros.

1.5. Alcances y limitaciones

1.5.1. Alcances

El proyecto en desarrollo tiene como alcance crear un sistema de visión artificial que detecte conductores descuidados, este sistema se podrá utilizar en cualquier medio de transporte terrestre, porque se evalúa al conductor y la manera de desarrollarse en el manejo de los vehículos.

1. Los aspectos relacionados a la visión artificial se realizarán y documentarán, por ejemplo: Obtención de la secuencia de imágenes, descripción de la cámara y su respectivos parámetros de imagen, procesamiento de imágenes para el óptimo desempeño del sistema y ubicación del dispositivo de grabación.

2. El reconocimiento del cuerpo del conductor y objetos, así como la interpretación de sus acciones para poder generar alertas visuales, mostradas por medio de Python-OpenCV y sonoras, no serán contemplados los factores externos al entorno del conductor.

3. Las definiciones de las acciones del conductor, como conducción cuidadosa, descuidada y su representación en el sistema de alarmas.

1.5.2. Limitaciones

Las siguientes limitaciones restringirán el avance del proyecto:

1. Característica técnica de los elementos de hardware: El uso de una cámara de vídeo de bajo perfil puede generar distintos problemas como, sistema de reducción de vibraciones precarios o nulos produciendo ruido en la imagen, problemas generales con la óptica normalmente la iluminación de la escena (imágenes sobreexpuestas o subexpuestas); La

capacidad computo insuficiente para el preprocesamiento y procesamiento de imágenes para el reconocimiento de gestos en tiempo real.

2. Características de la muestra: Los conductores que se evalúen pueden ser personas no relacionadas con un manejo profesional, ejemplo: personas de experiencia en el manejo de 5 años o menos; personas con más de 20 años de experiencia. Teniendo a individuos de pruebas con resultados heterogéneos, debido a la manera de comportarse totalmente distinta frente al volante, sin incurrir en un acto de descuido real; Los vehículos disponibles para las pruebas se limitan a transmisión automática y dejando la palanca de cambios en un modo pasivo.

3. Disposición por parte de los conductores en ser observados y evaluados, y disposición de tiempo por parte de ellos por sus ocupaciones escolares, laborales, familiares, entre otros.

Capítulo 2
Marco teórico

En la siguiente sección se abordarán los conceptos básicos que servirán para el desarrollo de este proyecto. Los siguientes conceptos que se definirán a continuación son: La visión artificial, su definición , los elementos que la componen , el factor de multiplicación de la distancia focal, el software especializado y las áreas de aplicación.

2.1. Visión artificial

La visión artificial es una disciplina científica como un subcampo de la inteligencia artificial, que mediante la implementación de métodos, técnicas y modelos que permiten adquirir imágenes, o una secuencia de imágenes, procesa por medio de filtros para generar información numérica o simbólica útil para una computadora, analizando la información proporcionada comparándose con criterios previamente definidos y comprendiendo estas imágenes del mundo real, esto con el fin de que una computadora interprete la escena, simulando la visión humana [12], [15], [16], el propósito de la visión artificial es programar una computadora para que 'entienda' lo que 've' y poder actuar en determinadas situaciones.

2.1.1. Adquisición de imágenes para un sistema de visión artificial

La clave principal para el correcto funcionamiento del sistema se basa en, que un objeto tiene que tener luz para poder ser observado, lo ideal es tener un ambiente con iluminación controlada y diferencias lumínicas mínimas para obtener una imagen útil para el. Hay distintos tipos de iluminación [12], [15], [16]:

- Iluminación direccional: Consiste en orientar directamente con un haz de luz al objeto.
- Iluminación difusa: Consiste en lograr que los haces de luz incidan desde diferentes direcciones.
- Iluminación a contraluz: Consiste en iluminar la parte posterior del objeto, contraria a la cámara de manera que esté alineado con el lente.
- Iluminación oblicua: Tiene como objetivo crear sombras para aumentar el contraste de las partes tridimensionales, con el fin de destacar detalles pequeños.

El método de iluminación óptimo se escoge según las características que se desean resaltar, al igual que la fuente de luz, para que no afecte la calidad de la imagen capturada o el lente de la cámara, existen múltiples fuentes de luz:

- Luz fluorescente de alta frecuencia. Cantidad generada de luz media, desgaste rápido y bajo costo.
- Luz halógena. Cantidad generada de luz alta, desgaste rápido y alto costo.
- Luz de xeon. Cantidad generada de luz muy alta, desgaste rápido y muy alto costo.
- Luz LED. Cantidad generada de luz desde baja hasta alta es muy configurable, desgaste lento y alto costo.

La cámara desempeña la función más importante en el proceso de un sistema de visión artificial, en los proyectos es necesario determinar el hardware que cumpla con las necesidades básicas y objetivos propuestos en el proyecto [12], [15], [16], existen 2 tipos de cámaras:

1. Cámaras analógicas. Es hardware que posee una salida analógica de video, la señal de video es limitada por el ancho de banda y el ruido análogo que puede llegar a proporcionar el cable.

2. Cámaras digitales. Son dispositivos de actualidad, poseen sensores sensibles a la luz y los componentes electrónicos brindan un desempeño y calidad de imagen mayor, se dividen en dos sensores los cuales son:

   a) Dispositivo de carga acoplada o acrónimo del inglés CCD: Se realiza una lectura de cada valor en cada una de las celdas, para obtener la información y un convertidor traduce de analógico al digital para formar datos, es necesario un chip dedicado para manejar la información, con lo cual es necesario equipos grandes y un mayor gasto.

   b) Semiconductor complementario de óxido metálico o acrónimo del inglés CMOS: Con celdas independientes, la digitalización de los pixeles se realiza en los transistores de cada celda, por lo que no se hace necesario un chip dedicado, si no que se realiza dentro del sensor, obteniendo equipo más pequeño y barato.

En el campo de la visión artificial los dispositivos más utilizados son los que implementan interfaces como Gigabit ethernet, USB2, USB3, Cameralink, Cameralink HS, Firewire. En los sistemas de visión artificial, las cámaras requieren características específicas siendo más sofisticadas que las convencionales, ofreciendo un control de tiempos y señales, de la velocidad de obturación, de la sensibilidad y de otros factores fundamentales a la hora de la implementación en el sistema de visión artificial.

En conclusión, a la hora de seleccionar una cámara hay que tener en cuenta una serie de características, y entre las principales se destacan:

- Resolución: número de pixeles que conforman la imagen capturada.
- Sensibilidad: nivel mínimo de iluminación que puede capturar el sensor.
- Rango dinámico: margen de la intensidad luminosa (luz) que puede capturar el sensor.
- Señal/ruido: influencia entre pixeles.
- Velocidad: velocidad máxima a la que puede capturar imágenes, siendo su medida en fotogramas por segundo (FPS).

2.1.2. La óptica

El factor de multiplicación de la distancia focal es un concepto antiguo y que se puede usar a favor en las cámaras digitales [15], este factor numérico resulta de multiplicar la distancia focal de un objetivo y para determinar el formato de referencia. En pocas palabras, una cámara con un sensor de imagen es más pequeño que el formato completo de la luz captada, esto tiene un efecto de recorte y obteniendo la parte central de la imagen, lo que provoca que con una misma distancia focal se obtenga un ángulo de visión menor.
Lo antes mencionado se puede observar en la Figura 2.1, el rectángulo de color rojo muestra lo que un sensor de 24 X 36 mm ve, el rectángulo color azul lo que vería un sensor de 15 X 23 mm y el círculo muestra la imagen real de la mayoría de las lentes de formato 35 mm. En conclusión, diferentes tamaños de lentes pueden captar menos o más información respecto a su sensor.

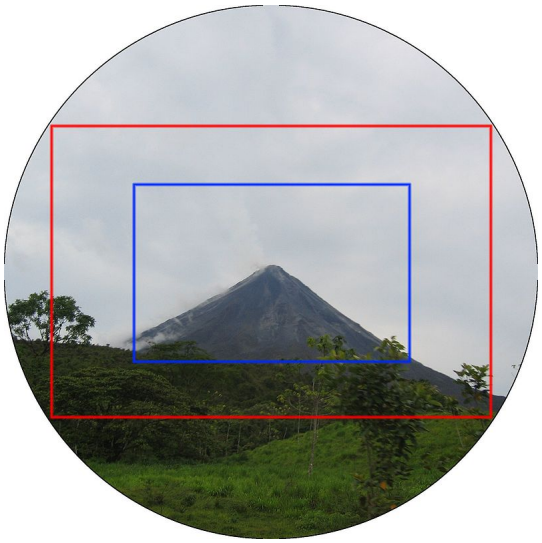


Figura 2.1. Distancia focal y los ángulos de visión [1].

2.1.3. Software para visión artificial

La rápida evolución de los procesadores y puertos de alta velocidad en las últimas décadas ha generado que los sistemas de visión artificial visualicen y procesen imágenes en tiempo real, y se implementen en aplicaciones de visión en entornos científicos e industriales. La evolución del hardware ha permitido el desarrollo de bibliotecas de visión que funcionan en entornos predeterminados. La plataforma Windows fue predominante en los softwares especializados, pero hoy en día las empresas desarrollan sus sistemas para soportar la mayoría de las plataformas como LINUX y MAC OS, las bibliotecas son las siguientes:

1. *The Open Computer Vision Library (OpenCV)* :
   Desarrollado por Intel® Corporation, la librería que proporciona un marco de trabajo de nivel medio-alto que ayuda al personal docente e investigador a desarrollar nuevas formas de interactuar con las computadoras, características de [17], [18]:

   - Su uso es libre tanto para su uso comercial como no comercial.

- Cuenta con 300 funciones y estructuras escritas en lenguaje C.
- No utiliza bibliotecas numéricas externas, aunque puede hacer uso de alguna de ellas, si están disponibles, en tiempo de ejecución.
- Dispone de interfaces para algunos otros lenguajes y entornos:
  - EiC - intérprete ANSI C escrito por Ed Breen. Hawk y CvEnv son entornos interactivos que utilizan el intérprete EiC.
  - Ch - intérprete ANSI C/C++ creado y soportado por la compañía SoftIntegration; Matlab® - gran entorno para el cálculo numérico y simbólico creado por Mathworks.

2. *JavaVis: Una librería para visión artificial en Java*:
   Desarrollado por el departamento de ciencia de la computación e inteligencia artificial de la Universidad de Alicante, que principalmente ayuda en el aprendizaje de los alumnos en las diversas materias de ingeniería informática, técnicas y ampliación de inteligencia artificial por nombrar algunas. Esta biblioteca presenta una serie de ventajas:

   - Portabilidad. Este software está escrito en lenguaje Java, y al estar orientado a objetos es posible una programación multiplataforma.
   - Interfaz gráfica. Con el propósito de solo desarrollar el código Java y olvidar los comandos de manejo de entorno, implementa una interfaz gráfica desarrollada.
   - Uso desde el entorno gráfico, línea de comandos o llamadas desde otro algoritmo.
   - Manejo de tipo especial de ficheros e imágenes. Este formato de imagen permite almacenar secuencias de imágenes y aprovechar la visión estéreo en cada instante.

2.1.4. Áreas de aplicación de la visión artificial

La visión artificial tiene diversas aplicaciones, en el sector industrial, seguridad electrónica, medicina, entretenimiento, entre otros. Entre las áreas de aplicación más comunes se menciona las siguientes aplicaciones:

- Inspección industrial y control de calidad : Esto permite automatizar procesos, que al ser humano le lleva más tiempo y energía para poder clasificar las piezas en defectuosas y no defectuosas, ejemplo en [12].
- Vigilancia y seguridad : Aplicados a los sistemas de videovigilancia, que permite detectar, reconocer rostros, también advierte de situaciones sospechosas, ejemplo en [19].
- Reconocimiento de caras : Utilizado en sistemas de detección y reconocimiento facial en este caso, ejemplo en [17] y [20].
- Vehículos autónomos: Los sistemas de aparcado automático, la detección de coches y peatones hasta ciclistas, el reconocimiento de la señalización o las alarmas de cambio involuntario de carril, ejemplo en [11], [13] y [14].
- Detección de patrones de sueño o somnolencia: Estos sistemas permiten detectar la somnolencia por medio de visión artificial detectando los ojos, ejemplo en [8] y [9].

Capítulo 3
Desarrollo del Proyecto

Este proyecto de software tuvo un ciclo de desarrollo y consta de una serie de tareas que finalizan en diferentes tiempos, semejante al método de cascada, se descartó la fase de mantenimiento, para adaptarse al modelo de entrega del proyecto. Con la característica de regresar a la fase anterior para solucionar errores, mejorar el diseño o falta de planificación del software, como se muestra en la siguiente Figura 3.1, esto presenta una ventaja, ya que se puede regresar hasta el diseño del sistema o programa y realizar ajustes, y al utilizar una plataforma tan experimental como lo es el aprendizaje de máquina puede fallar completamente un modelo y en una iteración sería imposible obtener un desarrollo óptimo del prototipo de sistema, lo ideal es que se puedan solucionar los problemas en el diseño del programa pues más arriba se tiene el diseño del sistema lo cual modificaría el objetivo general del proyecto.

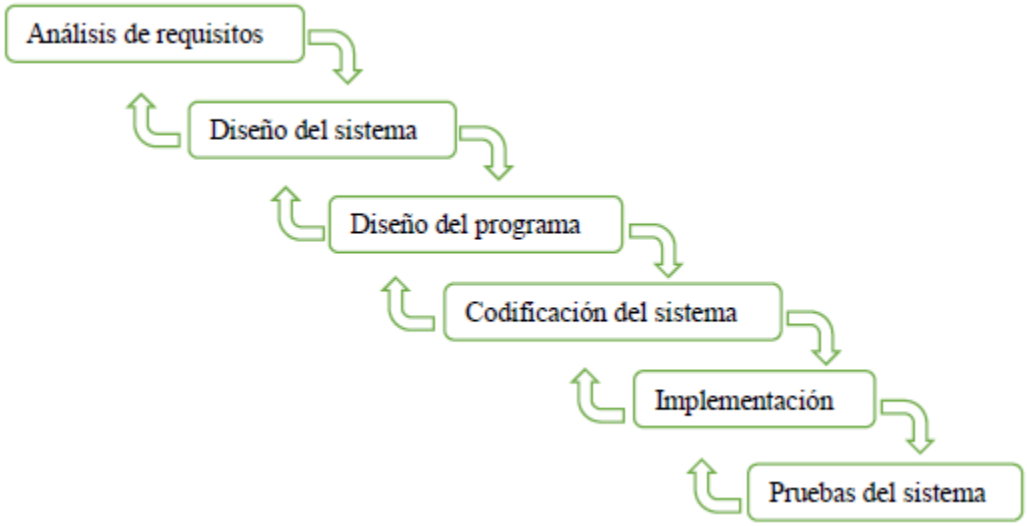


Figura 3.1. Desarrollo en cascada.

Las ventajas de utilizar la metodología de desarrollo de cascada son las siguientes:

- Modelo fácil de aplicar.
- Orientado a generar documentos.
- Es utilizado con frecuencia, por su efectividad.
- Se genera un análisis del software antes de iniciar a codificar.
- Buen funcionamiento en equipos de perfil bajo.

3.1. Producto propuesto

Se propuso un sistema el cual sea capaz de detectar los descuidos del conductor, para ser más específicos se desarrolló un sistema de visión artificial el cual detecta cuando el individuo está conduciendo con su atención al entorno fuera del automóvil, por lo tanto, otras acciones

como comer, maquillarse, quitar ambas manos del volante, entre otros., está penalizado por el sistema y se procede a emitir alertas visuales y auditivas.

3.2. Análisis de requisitos

Este proyecto se desarrolló un sistema de visión artificial mono ocular, el cual, fue montado en un automóvil con el propósito de poner a prueba el sistema en situaciones reales. Se analizaron diversas librerías referentes a la visión artificial como son, los modelos de detección de objetos, seguimiento de objetos, clasificación de imágenes, entre otros., gracias a esta investigación se llegó a la conclusión de utilizar las siguientes interfaces y plataformas que serían un entorno de desarrollo integral, a continuación, se listan las tecnologías utilizadas en este proyecto:

3.2.1. Software utilizado

Python3

En este proyecto se utilizará Python versión 3, un lenguaje de programación multiplataforma que es orientado a objetos y es software libre. Su sintaxis es sencilla y clara, además se reducen de manera significativa el ejecutar funciones o algoritmos, siendo este un lenguaje interpretado es bastante potente, la instalación de bibliotecas y módulos externos es sencilla a comparación de otros lenguajes, añadiéndole el gestor de paquetes y un sistema de gestión de entornos como lo es Anaconda que ofrece simplificar la administración de software muy eficiente.

OpenCV

Este software, utilizará la biblioteca de OpenCV la cual fue construida para proporcionar la infraestructura en sistemas de visión artificial, además de tener licencia libre, cuenta con 2500 algoritmos, es sencillo de utilizar y modificar su código. Esta biblioteca cuenta con funciones para visión artificial y aprendizaje automático, las cuales, se utilizan para detectar y reconocer diferentes objetos, clasificar las acciones humanas, extraer modelos de objetos en 3D, se puede configurar para cámaras estéreo en los sistemas de visión, manejo de imágenes y vídeos en general. OpenCV cuenta con interfaces en múltiples lenguajes de programación Python, MATLAB, C++, C y Java, y es compatible con los sistemas operativos de la actualidad: Windows, Android, MacOS y Linux.

Keras

Esta biblioteca de redes neuronales está desarrollada para ofrecer la capacidad de experimentar con redes de aprendizaje profundo (DNN por sus siglas en inglés) pues su interfaz de programación de aplicaciones, por sus siglas en inglés API, de alto nivel, es capaz de ser utilizada por usuarios de poca experiencia, Además de ejecutarse sobre "TensorFlow", Therano y CNTK. Asimismo de redes neuronales estándar ofrece soporte para redes neuronales convolucionales y recurrentes, es compatible con Python versiones 2.4 a 3.6.

TensorFlow

Esta plataforma proporciona el aprendizaje de máquina (por sus siglas en inglés "ML"), siendo un ecosistema flexible con librerías propias y agregadas por la comunidad, la plataforma pretende impulsar a los desarrolladores a desarrollar ML de manera fácil y proporcionar aplicaciones de grandes capacidades, puede ser utilizada con API de alto nivel como Keras.

3.2.2. Hardware y sus configuraciones

Configuración de pruebas y entorno de implementación, Automóvil:

- Marca: Pontiac
- Modelo: Grand AM
- Generación: Grand AM (H)
- Tipo de carrocería: Berlina

Laptop Asus Transformer Book T100HA:

- Sistema operativo: Windows 10 1803 LTS
- Memoria RAM: 4Gb
- Procesador: 1.44GHz Intel Atom x5-Z8500

Detalles para mencionar: Se instaló la versión de Python 3.7.3 pues su único propósito es el de recopilar información por medio de la biblioteca OpenCV-Python y el dispositivo de cámara web C920.

Laptop Asus N550JV:

- Sistema operativo: Windows 10 1903
- Memoria RAM: 8Gb
- Procesador: Intel Core i7 4700HQ 2.4GHZ

Detalles para mencionar: Se instaló todo el software antes mencionado en el análisis por medio de Anaconda3, utilizando como medio principal de desarrollo la interfaz interactiva

Jupyter Notebook, se creó el principal entorno de desarrollo llamado "cv", se instalaron los siguientes paquetes las siguientes versiones disponibles para el gestor de paquetes Conda:

- Python: 3.7.3
- OpenCV: 4.1.1
- Numpy: 1.16.4
- MathPlot: 3.1.0
- Keras: 2.2.5
- TensorFlow: 1.14.0

Logitech C920 HD Pro-Webcam:

- Resolución máxima: 1080 x 1920 píxeles / 30 FPS
- Tipo de enfoque: automático
- Micrófono integrado: estéreo

Detalles para mencionar: Las opciones que ofrece la cámara web se dejaron por defecto, por ejemplo, zum óptico, enfoque de lente, entre otros.

Configuración de la visión artificial

Se implementó el sistema de visión artificial dentro del automóvil en una posición intuitiva para tener una imagen clara conductor, por desgracia el entorno del automóvil es muy difícil de manejar para una cámara web por lo siguiente, el ángulo de visión que proporciona la cámara web es decente pudiendo visualizar el rostro del individuo y a su vez solo en ciertos momentos se puede visualizar el brazo derecho completamente, en cambio un punto totalmente muerto es el brazo izquierdo, el torso y las piernas del conductor. Esto es muy limitante para los objetivos del proyecto pues no se cuenta con suficientes recursos para obtener una cámara de 180 grados y un soporte de retrovisor, para tener mejor visión dentro del automóvil. A continuación, en las siguientes Figuras 3.2 y 3.3 se puede ver como se dispone de la configuración.

Figura 3.2. Configuración de la cámara web C920 en el automóvil.

Visión dentro del automóvil por medio de la cámara web C920, debidamente configurada en OpenCV, el ancho de fotograma es 1920 píxeles, alto de fotograma 1080 píxeles, véase Figura 3.3.

Figura 3.3. Visión dentro del automóvil.

Una vez ya definido el entorno de desarrollo que se utilizaron en esta sección se presenta los componentes de hardware y su configuración, también se presenta las interacciones de componentes y uso en el prototipo del sistema, como lo son: la configuración de visión artificial, definición de modelos en Keras, detalles de la creación de modelos, en general.

3.2.3. Definición de descuido y modelos en Keras

En el área de redes neuronales convolucionales (acrónimo en inglés "CNN") se pueden crear infinidad de modelos para resolver un solo problema, esto se debe a que no existe un estándar ideal a seguir, pues la cantidad de variables que modifican los resultados puede llegar a ser extenuante, por eso Keras en conjunto con TensorFlow proponen una interfaz amigable al usuario y este solo se dedique a experimentar con las CNN. En el área de clasificación de imágenes existen dos vertientes de modelos, binarios o categóricos, que por su nombre lo dicen todo, se basan en una simple neurona o varias en la última capa de la red neuronal.

Por motivos de tiempo del desarrollo se implementó en el prototipo de sistema para detectar conductores descuidados una clasificación binaria, estado descuidado y cuidadoso para ser más exacto, véase la Figura 3.4, la definición de descuido que se maneja es la siguiente manera:

- No tiene la vista dirigida al exterior del automóvil.
- Hace movimientos muy exagerados para ver a los laterales del vehículo.
- Objetos que interfieren con la captura de datos de la visión artificial.

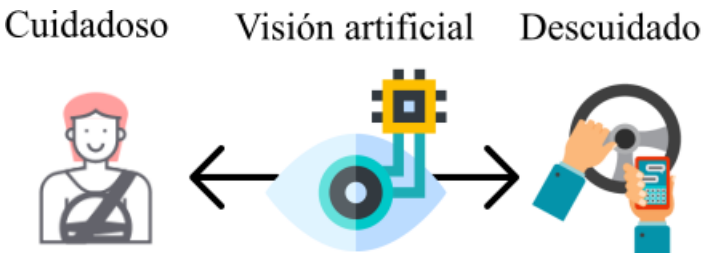


Figura 3.4. Clasificación binaria de imágenes.

Estas reglas resultaran muy obvias para muchos, pero no existe una definición exacta de un descuido al conducir, esto se deja más a la experiencia de cada individuo o políticas de

empresas de transporte, pudiendo agregar y quitar reglas a la definición. Para dejar en claro la idea que se manejará, la atención del individuo debe de estar al frente del vehículo a donde se desea dirigir.

3.3. Diseño del prototipo del sistema

Ya realizado el análisis del prototipo del sistema, se diseñaron los siguientes pasos que se siguieron para el desarrollo del sistema, véase la Figura 3.5:

1. Obtención de vídeos de prueba
2. Generar Set de datos
3. Compilar un modelo
4. Pruebas del modelo
5. Implementación del modelo

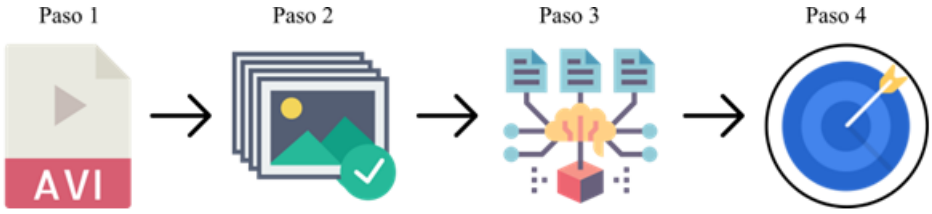


Figura 3.5. Diseño del prototipo del sistema.

Se aplico una estructura de archivos para tener un orden para el fácil acceso y desarrollo, la carpeta raíz contendrá los archivos principales de Jupyter Notebook, tales como los componentes de los pasos del 1 al 4, después se creó la carpeta "dataset" donde se guardaron los resultados del paso 1 y 2. Para el paso 3 se crearon carpetas llamadas "M" más el número del modelo agregando guión bajo más el Conjunto de datos utilizado en la compilación del modelo, por ejemplo: M0_DSn0, M1_DSn1, entre otros., en estas carpetas se guardaron los modelos resultantes del paso 3 y respectivamente sus pruebas de exactitud. A continuación, en la Figura 3.6 se muestra como es la jerarquía de una manera más visual.

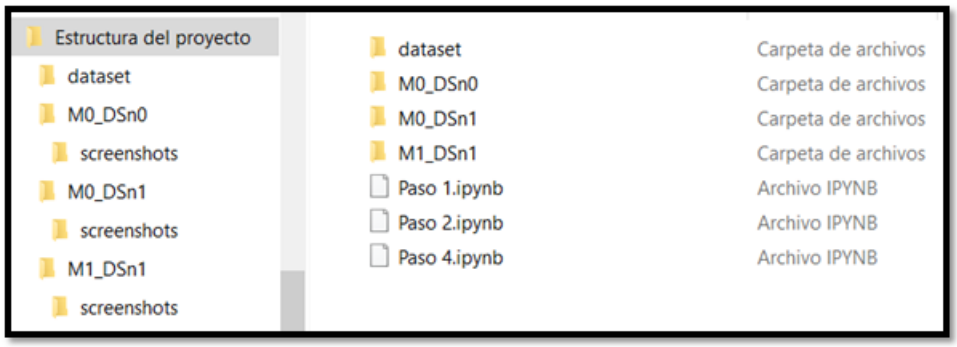


Figura 3.6. Estructura de archivos.

3.3.1. Obtención de vídeos

Es de esperar las redes neuronales necesitan grandes cantidades de datos para ser nutridas, para solucionar este problema de información, este componente de obtención de vídeos se encarga de grabar el material para su posterior uso en otros componentes, como lo son generar configuraciones de datos y pruebas de modelos. Por suerte al estar en un entorno tan variable de iluminación y en conjunto con las vibraciones del automóvil que generan al conducir un vídeo puede contener gran cantidad de información útil.

Después de obtener cada vídeo es necesario seleccionar los fotogramas los cuales serán clasificados como 'conductor descuidado', para este paso fue necesario crear una lista de los fotogramas, en un archivo de texto, los cuales sera utilizados en las pruebas e implementación del modelo.

3.3.2. Creación del set de datos

De manera subsecuente al obtener los vídeos del componente anterior, se necesitó crear un conjunto de datos para utilizarlos como entrenamiento, validación y pruebas del modelo.

Definición de información útil y reutilizable

Para generar el set de datos es necesario entregar la imagen digital lo más cercano al modelo de predicción que se quiere lograr, para esto se eligió un solo sitio donde implementar el sistema de visión artificial, como se describió en el análisis del sistema, por lo tanto, después de obtener un conjunto de datos y querer combinarlo con otro deben de tener un conjunto de características similares, como las siguientes:

- Área de trabajo: Este sistema al ser dedicado a la detección de descuidos es importante obtener la mayor información del estado del individuo al manejar, por este motivo es necesario seleccionar las áreas de interés de los datos, la configuración de visión artificial se puede obtener un plano muy bueno del rostro del conductor, véase Figura 3.3, para concluir nuestra área de interés seria la cabeza del conductor y nuestro punto central el rostro.
- Modelo de color: Al manejar imágenes es necesario especificar qué tipos de imágenes se van a manejar, en este caso se optó por un modelo RGB de tres capas por ser el más utilizado en OpenCV.
- Filtrado: Es necesario repartir la información generada en diferentes partes, entrenamiento con un 75 %, validación con un 20 %, y para finalizar los datos sin ver con un 5 %, para su posterior clasificación de categorías.
- Clasificación de categorías: Para obtener una muestra homogénea y eficaz, es necesario eliminar las imágenes inútiles donde no se puedan ofrecer un beneficio al conjunto de

datos o imágenes totalmente negras o blancas por culpa de la iluminación, a la par, los razonamientos para obtener información son las siguientes:

1. "driving-fixation": Categoría de "estatus: cuidadoso" se rige por las características de las acciones del conductor, por ejemplo, en la categoría "estatus: cuidadoso" se clasificarían todas las muestras donde el individuo este concentrado en la tarea de conducir, como ejemplo, el mirar al frente, frente – derecha, retrovisor izquierdo y derecho, retrovisor central, entre otros.
2. "other-fixation": Categoría de "estatus: descuidado" sucede lo contrario, toda acción fuera de la actividad de conducir queda considerada como descuidada.

3.3.3. Compilación del modelo

Este componente se configuró con las muestras preprocesadas para obtener un modelo que pueda interpretar los estados al conducir: descuidado y cuidadoso. Para adquirir este modelo es necesario definir cuantas capas de convolución, capas de agrupación y capas de neuronas se desean implementar para obtener un modelo lo suficientemente robusto, véase la Figura 3.7. A continuación, un listado de los parámetros a configurar:

- Capas de convolución y máxima agrupación (del inglés "Convolution layers", "max polling" y sus respectivos acrónimos "CL" y "POL" utilizados en este documento): Para mantener coherencia y obtener una imagen debidamente reducida se decidió aplicar un agrupamiento de imagen lineal, por ejemplo, 256, 128, 64, 32, 16, 8, entre otros.

- Capa de neuronas densamente conectadas (acrónimo en inglés "DL" utilizado en este documento): No existe un estándar para este paso, aunque, es necesario crear una conexión de CL a DL 1 a 1, o sea, que la proporción sea lineal y ningún valor se pierda al momento de que la información de la imagen agrupada (procesos previos CL y POL) sea recibida por las capas densamente conectadas (DL). En estas capas DL se definió como máximo 2048 neuronas en la primera capa y un mínimo de salida de una neurona.

- Épocas (del acrónimo en inglés "epochs"): Para llegar a un modelo exacto (acrónimo en inglés "acc") y sin pérdida (del inglés "loss") es necesario entrenar el modelo en repetidas ocasiones para que "aprenda", debido al tiempo limitado en el desarrollo del proyecto, se definieron 10 épocas como mínimo y 50 épocas como máximo.

- Resultado Exactitud y Pérdida: Cada época de entrenamiento arroja un valor que representa cuanto ha aprendido la red neuronal de las entradas del conjunto de datos, entre mas alto sea el valor de exactitud mejor funcionará y por el contrario entre menos valor de pérdida obtenga será mejor el desempeño.

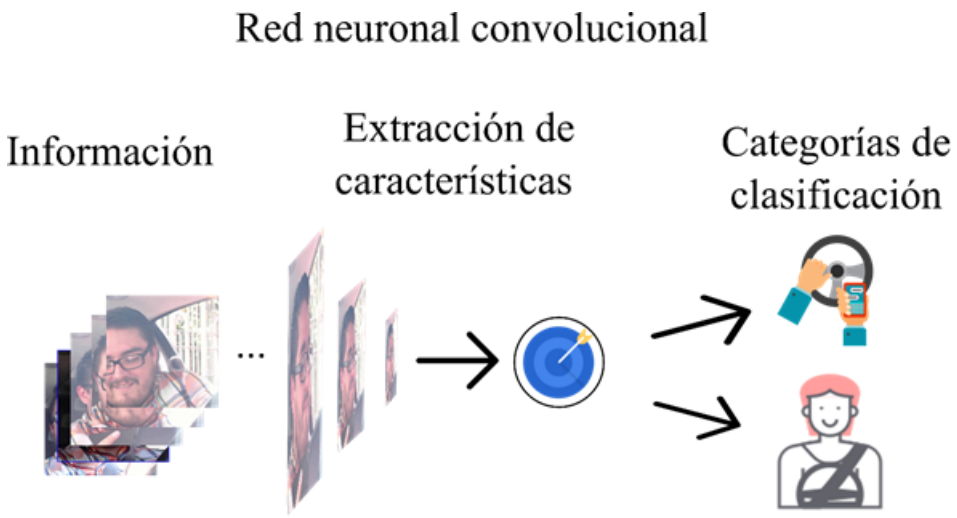


Figura 3.7. Diseño de una red neuronal convolucional.

3.3.4. Pruebas de exactitud

Después de generar el modelo se idearon una serie de cuatro pruebas con la finalidad de asegurar que la red neuronal convolucional, funciona debidamente y pueda ser implementado en cualquier entorno:

- Prueba de exactitud 0, prueba de generador: Se apartó un 5 % de datos que no fueron vistos por el modelo de entrenamiento, por lo tanto, al poner a prueba la entrada y la salida de datos se puede llegar a la conclusión de qué tan eficiente es el modelo.
- Prueba de exactitud 1, prueba Blanca: El modelo se entrenó y validó con una serie de fotogramas los cuales pertenecen a un recorte del vídeo por consiguiente debería de presentar como respuesta lo que se espera del modelo, una detección correcta del descuido, en el entorno de pruebas más fácil. Para validar este resultado se utilizará la lista de fotogramas de conductores descuidados tener un punto de comparación.

3.3.5. Implementación del sistema de prototipo

En el área de visión artificial limitar el área de trabajo es importante pues el modelo puede interpretar de diferentes maneras una misma imagen, véase la Figura 3.8 del lado izquierdo, la cual es la misma pero en un encuadre diferente. Para evitar el sesgo que esto genera esta implementación del modelo utiliza áreas de trabajo mas grandes, pudiendo encuadrar completamente el área de interés como lo es la cabeza del conductor, véase la Figura 3.8 del lado derecho.

Figura 3.8. Áreas de trabajo.

Esta herramienta de implementación cuenta con las siguientes funciones:

- Alarma auditiva: el cual reproducirá un pitido corto al generarse un descuido por corto tiempo.
- Alarma visual: se representará en 2 textos de colores en la interfaz desarrollada en OpenCV, siendo el mensaje "Estado: descuidado" (traducido al inglés "Status: careless") de color amarillo y su contra parte "Estado: cuidadoso" (traducido al inglés "Status: careful") en color azul cían, véase la Figura 3.9.
- Cambiar el área de trabajo: Agrandar el área de trabajo así como su posición en el fotograma, véase la Figura 3.9.
- Accesos directos: para poder reproducir diferentes partes del vídeo, su velocidad de presentación, así como capturar fotogramas específicos.
- Punto de comparación: Anteriormente se creó la lista de fotogramas clasificados como conductores descuidados, de manera que al comparar el resultado del modelo y la

clasificación de los fotogramas se obtendrá un porcentaje de error de modelo, en otras palabras este porcentaje representa si el modelo cumple con el objetivo general.

Figura 3.9. Alarma visual.

3.4. Diseño del prototipo del programa

Como se dijo en la metodología, se pueden reiterar en múltiples ocasiones para obtener un prototipo del sistema de detección que cumpla con los objetivos generales. Se desarrollaron los componentes de manera separada, aunque todos comparten casi las mismas líneas de código o funciones, utilizan las bibliotecas de OpenCV-Python, Keras, Numpy y reutilizan la misma estructura del código, a continuación, el pseudocódigo del sistema de visión artificial:

```
Importar bibliotecas necesarias
Conectar el sistema con el vídeo o cámara
Si se conectó correctamente Aplicar:
    Configurar la cámara
    Inicializar variables
Mientras sea verdad:
    Obtener fotogramas del objeto de vídeo
    ...
    Modificar fotogramas respecto al componente
    Mostrar el resultado
    Agregar funciones a la aplicación (Opcional)
Fin
```

En esta sección se presentan los seudocódigos de los componentes principales del programa, que consta de 5 pasos:

1. Obtención de vídeos de prueba
2. Generar Set de datos
3. Compilar un modelo

4. Pruebas del modelo

5. Implementación del modelo

3.4.1. Obtención de vídeo de prueba

El siguiente pseudocódigo es para el componente del paso 1, para mantener íntegros los datos, es necesario guardar directamente los fotogramas sin ningún filtro o cambio a la imagen.

```
Importar bibliotecas necesarias
Conectar el sistema con el vídeo o cámara
...
Si se conectó correctamente Aplicar:
Configurar la cámara
    Ancho de fotograma = 1920
    Alto de fotograma = 1080
    Fotograma por segundo = 30
    Códec de compresión de vídeo = H264
Mientras sea verdad:
Obtener fotogramas del objeto de vídeo
...
Guardar fotogramas
Fin
```

Para la reproducción del vídeo posteriormente de su grabación, agregando la función de crear el archivo de texto para la lista de fotogramas de conductores descuidados, seria el siguiente pseudocódigo:

```
Importar bibliotecas necesarias
```

```
Conectar el sistema con el vídeo o cámara
Crear un archivo de texto
...
Si se conectó correctamente Aplicar:
    Configurar Velocidad de reproducción
Mientras sea verdad:
Obtener fotogramas del objeto de vídeo
...
Si presionó una tecla:
        Tecla y: Guardar el numero del fotograma
        Tecla -: Modificar la velocidad de reproducción
        Tecla p: Ir al fotograma
Fin
```

3.4.2. Generar Set de datos

El siguiente pseudocódigo es para el componente del paso 2, para transformar los datos en información útil para el modelo es necesario delimitar el área de trabajo como se maneja Python se puede segmentar un fotograma fácilmente agregando una tupla de valores iniciales y finales respecto al plano X – Y, aplicar un modelo de color y recortar los fotogramas respecto al conductor, también para obtener un set de datos se debe de aplicar una frecuencia de fotogramas para abarcar todo el vídeo. A continuación, el pseudocódigo:

```
Importar bibliotecas necesarias
Conectar el sistema con el vídeo o cámara
...
Si se conectó correctamente Aplicar:
    Configurar la cámara
    Inicialización de Variables de coordenadas Yi:Xf y Xi:Xf de recorte
    Inicialización de variables para la frecuencia de frames
Mientras sea verdad:
    Cargar frames respecto a la frecuencia
    Obtener fotogramas del objeto de vídeo
    ...
    Recortar respecto al fotograma Yi:Xf , Xi:Xf
    Guardar fotogramas
Fin
```

3.4.3. Compilar un modelo

El siguiente pseudocódigo es para el componente del paso 3, fue necesario el resultado del componente 2 para utilizarlo en los generadores de entrenamiento y validación, así como su normalización y clasificación. A continuación, el pseudocódigo:

```
Importar bibliotecas necesarias
Configuración de variables para el modelo
Crear objeto del modelo
Agregar capas para la red neuronal convolucional
    Agregar filtros de agrupamiento
    Agregar capa que evita el sobre entrenamiento
Agregar capas para la red neuronal
Crear objeto generador de imágenes
Crear objeto generador de entrenamiento
Crear objeto generador de validación
Inicial el ajuste con los generadores de entrenamiento y validación
Guardad el modelo ajustado
Fin
```

3.4.4. Pruebas del modelo

Al finalizar la compilación del modelo se procede a aplicarle una serie de pruebas al modelo generado, con la finalidad de obtener un modelo apto para presentar:

Prueba de exactitud 0

Esta evaluación del modelo se aplica al finalizar el ajuste del modelo, para evaluar su desempeño de entradas y salidas del modelo, a continuación, el pseudocódigo:

```
Importar bibliotecas necesarias
Cargar el modelo
Crear objeto de generador de imágenes
Crear objeto de datos sin ver
Evaluar modelo con el objeto de datos sin ver
Mostrar resultados
Fin
```

Prueba de exactitud 1

Esta evaluación del modelo se ejecutó con una herramienta en OpenCV consiste en recortar el área de trabajo y enviarla para generar una predicción del modelo, en las pruebas de exactitud blanca se envía la misma área de trabajo que se utilizó para el conjunto de datos, a continuación, el pseudocódigo:

```
Importar bibliotecas necesarias
Conectar el sistema con el vídeo o cámara
Cargar el modelo
Inicialización de variables
```

```
Si se conectó correctamente Aplicar:
    Inicialización de Variables de coordenadas Yi:Xf y Xi:Xf de recorte
Mientras sea verdad:
    Obtener fotogramas del objeto de vídeo
    Modificar el fotograma de las coordenadas Yi:Xf , Xi:Xf
    Llamar a la función de predicción
    Llamar a la función de evaluación del modelo
    Si es un descuido:
        Envía señales visuales y auditivas
        Aumentar la variable de descuidos
    Si no lo es:
        Aumentar la variable de cuidadoso

Guardar los resultados
Fin
```

3.4.5. Implementación del modelo

Este componente se desarrolló en la biblioteca de OpenCV con el propósito de ser más versátil al momento de probar los modelos, además de teclas especiales para obtener capturas del sistema en ejecución, la reproducción del vídeo, obtener una recopilación de datos correctos para la evaluación del modelo, entre otras funciones, a continuación, se mostrará el pseudocódigo con todos los módulos dichos anteriormente en el diseño en esta herramienta:

```
Importar bibliotecas necesarias
Cargar el modelo
Inicialización de variables
Conectar el sistema con el vídeo o cámara
Funciones para mover el área de trabajo en X, Y, Área.
Si se conectó correctamente Aplicar:
    Inicialización de Variables de coordenadas Yi:Xf y Xi:Xf de recorte
    Inicialización de variables para la reproducción de vídeo
    Crear controles deslizantes para las funciones X,Y, Área
Mientras sea verdad:
    Obtener fotogramas del objeto de vídeo
    Modificar el fotograma de las coordenadas Yi:Xf , Xi:Xf
    Llamar a la función de predicción
    Llamar a la función de evaluación del modelo
    Si es un descuido:
        Emita señales visuales amarillo
        Emita señales auditivas
        Aumentar la variable de descuidos
    Si no lo es: Pasar
        Emita señales visuales  azul cían
```

```
        Disminuir la variable de descuidos
        Aumentar la variable de cuidadoso
    Mostrar fotograma con su respectiva información
    Si presionó una tecla:
        Tecla S: Capturar fotogramas
        Tecla J: Guardar numero del fotograma
        Tecla F: Guardar Fotograma
        Tecla numérica: Reproducción del vídeo
        Tecla P: Configurar la velocidad del vídeo
        Tecla Q: Salir

Mostrar resultados en la terminal
Fin
```

3.5. Codificación del prototipo de sistema

En esta sección se codificaron los componentes del sistema en el lenguaje de Python versión 3 añadiendo las librerías de OpenCV-Python para ejecutarlo en el sistema operativo Windows 10 x64, utilizando IDE Anaconda3 y el principal medio de compilación de Jupyter Notebook la lista de tareas que se codificaron son las siguientes:

1. Obtención de vídeos de prueba
2. Generar Set de datos
3. Compilar un modelo
4. Pruebas del modelo
5. Implementación del modelo

Se codificó el esqueleto de la visión artificial para los siguientes componentes 1, 2 y 4. Esta estructura es para no saturar el documento con código repetitivo y solo describir las líneas de código que sean de vital importancia para el componente, a continuación, el código que se utilizó y sus módulos, véase la Lista 3.1:

```
import cv2 as cv2
import numpy as np
from PIL import Image
import os
#-----Importar librerías-----#
#-----Crear Objeto de video-----#
if (video.isOpened() == False):
    print("Unable to read video feed")
else:
#-----Inicialización de variables para el objeto-----#
While True:
#-----Configuración de la frecuencia del tiempo-----#
    ret,frame=video.read()
  if not ret:
    break
#-----Modificación del fotograma capturado-----#
#-----Guardar el fotograma-----#
cv2.imshow("Nombre",frame)
if k==ord("q"):
  break#Presionar la tecla q para finalizar la captura
#-----Opciones-----#
#-----Final del código-----#
video.release()
cv2.destroyAllWindows()
```

Listado 3.1. Estructura principal

3.5.1. Obtención de vídeo de prueba

Este módulo se codificó para obtener vídeos por medio de la configuración de visión artificial mostrada el análisis del requisitos, véase la Figura 3.2, a continuación, se muestran las líneas de código, véase el Listado 3.2, 3.3, 3.4 y 3.5:

```
camNum = 1 #Numero de la cámara que se utilizará
video = cv2.VideoCapture(camNum, cv2.CAP_DSHOW)
```

Listado 3.2. Crear Objeto de vídeo

```
iWidthX = 1920; iHeightY = 1080; iFps = 30
video = cv2.VideoCapture(camNum, cv2.CAP_DSHOW)
video.set(cv2.CAP_PROP_FRAME_WIDTH, iWidthX)
video.set(cv2.CAP_PROP_FRAME_HEIGHT, iHeightY)
video.set(cv2.CAP_PROP_FPS, iFps)
fourcc = cv2.VideoWriter_fourcc("H","2","6","4")
strNameVideo="datasets/"+"NombreDelVideo"+".mp4"
out = cv2.VideoWriter(strNameVideo,fourcc,iFps,(iWidthX,iHeightY))
```

Listado 3.3. Inicialización de variables para el objeto

```
out.write(frame)
Opciones:
k = cv2.waitKey(33) & 0xff
if key ==ord("f"):
  cv2.imwrite("F-0" + str(sec) + ".jpg", frame)
```

Listado 3.4. Guardar el fotograma

```
out.release()
```

Listado 3.5. Final del código

Para continuar, el código de la reproducción del vídeo anteriormente grabado y la función de crear la lista de fotogramas de descuidos y la tecla especial para escribir en el archivo de texto el fotograma, véase el Listado 3.6, 3.7 y 3.8:

```
vtest = "datasets/video.mp4"
textFileOutputName = vtest[:-4] + "video_CarelessFramesCount.txt"
textFileOutput = open(textFileOutputName, "w")
video = cv2.VideoCapture(vtest)
```

Listado 3.6. Crear Objeto de vídeo

```
if key == ord("y"):
        textFileOutput.write( str(intframe) + "\n")
```

Listado 3.7. Opciones

```
textFileOutput.release()
```

Listado 3.8. Final del código

3.5.2. Generar Set de datos

Como en la introducción del capítulo 3 en este apartado se utilizará la estructura propuesta en la Lista 3.1, a continuación, véase el Listado 3.9, 3.10, 3.11, 3.12 y 3.13 para visualizar el código, las variables y recursos utilizados son "vtest0.mp4", en el área de trabajo X:320, Y:1050 a 10 fotogramas por segundo.

```
vtest = "datasets/vtest0.mp4"
iWidthX = 1080; iHeightY = 1920; IMG_SIZE = 256
cap = cv2.VideoCapture(vtest)
```

Listado 3.9. Crear Objeto de vídeo

```
stY = 320 ; stX = 1050
enY = stY + IMG_SIZE; enX = stX + IMG_SIZE
sec = 0; frameRate = 0.10  # For dataset
```

Listado 3.10. Inicialización de variables para el objeto

```
    sec = sec + framerate;        sec = round(sec, 2)
    cap.set(cv2.CAP_PROP_POS_MSEC, sec*1000)
```

Listado 3.11. Configuración de la frecuencia del tiempo

```
im = Image.fromarray(frame[stY:enY,stX:enX], "RGB")
```

```
im = im.resize((256,256))
im.save(os.path.join("datasets/DSn1/", str(sec)+".jpg"), "JPEG")
```

Listado 3.12. Modificación del fotograma capturado

```
if key == ord("f"):
        cv2.imwrite("F-"+str(sec)+".jpg",frame)
        cv2.imwrite("FS-"+str(sec)+".jpg",frame[stY:enY,stX:enX])
```

Listado 3.13. Opciones

3.5.3. Compilar un modelo

El siguiente componente se codificó linealmente, no utiliza la estructura propuesta en la Lista 3.1, en consecuencia porque es secuencial el proceso, aunque se detalla por módulos para ser mas comprensible el código, a continuación, véase el Listado 3.14, 3.15, 3.16, 3.17, 3.18 y 3.19 el código para compilar un modelo en Keras que corresponde al Modelo M0 en conjunto con el conjunto de datos Dns1 (M0_DSn1):

```
from keras.models import Sequential
from keras import models
from keras import layers
from keras import optimizers
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K
```

Listado 3.14. Importar librerías

```
epochs = 10; batch_size = 32
train_path_dir = "train"; validation_path_dir = "test"
nb_train_samples = 667; nb_validation_samples = 166
conv1 = 256 ; conv2 = 64;
conv3 = 32; conv4 = 16;
convSet2 = (2, 2); convSet3 = (3, 3)
dense0 = 1; dense1 = 64; dense2 =1024; #dense3 =;
```

Listado 3.15. Crear variables

```
model = models.Sequential()
model.add(layers.Conv2D(
    conv1, convSet3, activation="relu",
    input_shape=(256,256,3)))
model.add(layers.MaxPooling2D(3,3))
#------ more CL for conv2, 3, 4 ------#
model.add(layers.Flatten()); model.add(layers.Dropout(0.5))
#------ more DL for dense2, 3, 4 ------#
model.add(layers.Dense(dense1, activation="relu"))
model.add(layers.Dense(dense0, activation="sigmoid"))
model.compile(
    loss="binary_crossentropy",
    optimizer="adam",
```

```
    metrics=["acc"])
```

Listado 3.16. Agregar capas en secuencia

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

validation_datagen = ImageDataGenerator(rescale=1.255)

train_generator = train_datagen.flow_from_directory(
    train_path_dir,
    target_size=(size,size),
    batch_size = batch_size,
    class_mode="binary")

validation_generator = validation_datagen.flow_from_directory(
    validation_path_dir,
    target_size=(size,size),
    batch_size = batch_size,
    class_mode="binary")
```

Listado 3.17. Configurar el conjunto de datos de entrenamiento y validación

```
history = model.fit_generator(
    train_generator,
    epochs = epochs,
    steps_per_epoch = nb_train_samples // batch_size,
    validation_steps = nb_validation_samples // batch_size,
    validation_data=validation_generator,
    workers=4)
```

Listado 3.18. Iniciar el ajuste del modelo

```
strSaveWeightsJSON = "M0_DSn1/model_load_createModel.JSON"
strSaveWeightsH5 = "M0_DSn1/model_load_weights.H5"
model_json = model.to_json()
with open(strSaveWeightsJSON, "w") as json_file:
    json_file.write(model_json)
model.save_weights(strSaveWeightsH5)
```

Listado 3.19. Guardar modelo

3.5.4. Pruebas del modelo

Como en la introducción del capítulo 3 en este apartado se utilizará la estructura propuesta en la Lista 3.1, al finalizar la compilación del modelo se procedió a aplicarle una serie de pruebas al modelo generado, con la finalidad de obtener un modelo apto para presentar, en la prueba de exactitud 1 se pueden desechar los modelos por los siguientes ejemplos: se mantiene estático el modelo, genera resultados no deseados, comportamiento anormal, entre otros., por estas causas no cumplirían con el objetivo del prototipo del proyecto, a continuación, las pruebas de exactitud:

Prueba de exactitud 0

En el diseño del prototipo del proyecto se mencionó que la prueba de exactitud 0 se puede ejecutar inmediatamente después de crear el modelo, aunque también se incluye el código para ejecutarlo desde otra instancia de Jupyter Notebook, véase el Listado 3.20 y 3.21, a continuación el código:

```
from keras import models
from keras.models import model_from_json
from keras.preprocessing.image import ImageDataGenerator
mtest = "M0_DSn1/"
dir_model_json  = mtest + "model_load_createModel.JSON"
dir_model_h5    = mtest + "model_load_weights.H5"
test_dir = "datasets/DSn1/unseen"

json_file = open(dir_model_json, "r")
loaded_model_json = json_file.read()
```

```
json_file.close()
loaded_model = model_from_json(loaded_model_json)

loaded_model.load_weights(dir_model_h5)
print("Loaded model from disk: ", strSaveWeightsH5)
loaded_model.compile(
    loss="binary_crossentropy",
    optimizer="adam",
    metrics=["accuracy"])
```

Listado 3.20. Cargar el modelo desde el directorio

```
test_datagen = ImageDataGenerator(rescale=1. / 255)
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size = (size , size),
    batch_size = 32,
    class_mode = "binary",)
test_loss , test_acc = loaded_model.evaluate_generator(
    test_generator ,
    steps = 50 )
print("test acc:  ", test_acc)
print("test_loss: ", test_loss)
```

Listado 3.21. Test 0: Configurar los conjuntos de datos sin ver

Prueba de exactitud 1

Como se dijo en el diseño del programa, en esta prueba de exactitud 1 se utilizó el modelo de Keras previamente compilado, la configuración de generar set de datos (incluye los parámetros como: área de trabajo, medidas, modelo de capas de color) utilizado para nutrir el conjunto de datos, para que de este modo el modelo se ejecute en un entorno amigable, por lo cual se puede reutilizar todo el código de 'Generar set de datos', codificado anteriormente, véase el Listado 3.9, 3.10, 3.11, 3.12 y 3.13. Agregando el módulo 'Definición de funciones necesarias', de la Lista 3.22 en 'Importar Librerías', y en 'Modificación del fotograma capturado', se modifica por el siguiente código de la Lista 3.23. Para finalizar se agrega todo el módulo 'Cargar el modelo desde el directorio', presentado en la prueba de exactitud 0, véase la Lista 3.20.

En este punto del código es necesario eliminar la instrucción del código 'Guardar fotograma', pues ahora es necesario transformar esa variable en un arreglo para posteriormente utilizarla en una predicción del modelo. Además de declarar las funciones necesarias para cargar el archivo de texto de la lista de fotogramas descuidados, seguido de la función de comparar las predicciones del modelo contra las respuestas correctas, a continuación, véase los listados 3.22, 3.23 y 3.24 para visualizar el código.

En la función "existInList()" al retornar "True" o "False", se puede dar cabida a crear vídeos que contengan solo lo los fotogramas que incluye la lista de conductores descuidados o contrario, que no contenga ninguno. Utilizando de tal manera el Listado 3.3 y 3.4 en la instrucción 'Si detecta este fotograma en la lista guardar en este vídeo o si no en este otro'.

```
textFileInput = vtest[:-4] + "_CarelessFramesCount_v1.txt"


def read_InputText(filename):
```

```
    with open(filename) as f:
        return [int(x) for x in f]

def existInList(intframe,datasetCarelessFrames):
    return intframe in datasetCarelessFrames

def testingFrame(intframe,
                 modelError,
                 datasetCarelessFrames,
                 prediction):
    if prediction == 1:
        if existInList(intframe,datasetCarelessFrames):
            modelError[0] = int(modelError[0]) + 1
            return modelError , setColorUI_V1(prediction)
        if not existInList(intframe,datasetCarelessFrames):
            modelError[1] = int(modelError[1]) + 1
            return modelError , setColorUI_V1(prediction)

    if prediction == 0:
        if existInList(intframe,datasetCarelessFrames):
            modelError[1] = int(modelError[1]) + 1
            return modelError , setColorUI_V1(prediction)
        if not existInList(intframe,datasetCarelessFrames):
            modelError[0] = int(modelError[0]) + 1
            return modelError , setColorUI_V1(prediction)

def setColorUI_V1(prediction):
```

```
    if prediction == 1:
        return (0,255,255) #Yellow
    if prediction == 0:
        return (255,255,0) #BlueCian
datasetCarelessFrames = read_InputText(textFileInput)
```

Listado 3.22. Definición de funciones necesarias

```
#DELETE—— im.save(os.path.....#
img_array = np.array(im)
img_array = np.expand_dims(img_array, axis=0)
prediction = int(loaded_model.predict(img_array)[0][0])
modelError, colorInterface = testingFrame(, , , , )
if prediction == 1: # Si el conductor esta descuidado
  iDrivingStatus_Bad = iDrivingStatus_Bad + 1
  frame = cv2.putText(frame,"Careless",,,,
                                colorInterface,,)
if prediction == 0:
  frame = cv2.putText(frame,"Carefull",,,,
                                colorInterface,,)
```

Listado 3.23. Modificación del fotograma capturado versión 1

```
print ("\n\n———————————————————————————————————————")
print (mtest +" "+ vtest +" "+str(defaultSet))
print ("Count Model careless:\t", iDrivingStatus_Bad)
print ("Count Model careful :\t", iDrivingStatus_Good)
```

```
print ("Error Model    Correct:\t" +
        str(modelError[0]*100/lengthTotalFrames))

print ("Error Model Incorrect:\t" +
        str(modelError[1]*100/lengthTotalFrames))
```

Listado 3.24. Final del código

3.5.5. Implementación del modelo

En esta parte de la codificación solo se agregan funciones especificas de Python-OpenCV, pues se reutilizará el código anterior de prueba de exactitud 1 con ligeros cambios y adición de funciones tales como:

- Alertas visuales (por medio de Python-OpenCV)
- Alertas sonoras

En esta lista de códigos se mejora la la configuración de color en la interfaz, pudiendo mostrar colores acordes a, si la predicción del modelo es correcta o incorrecta, lo cual los colores cambiaran a verde o rojo respectivamente, además de cuando el modelo detecta un descuido reproduce un sonido, a continuación, véase el Listado 3.25 y 3.26 para visualizar el código:

```
import winsound

def aAlertFunc(freq, length):
    winsound.Beep(int(min(max(freq,28),32766)),int(length))

def testingFrame(,,,):
    if prediction == 1:
        if existInList(,):...
            return , setColorUI_V2(prediction,True)
        if not existInList(,):...
            return , setColorUI_V2(prediction,False)
```

```
    if prediction == 0:
        if existInList(,):...
            return   , setColorUI_V2(prediction,False)
        if not existInList(,):...
            return   , setColorUI_V2(prediction,True)

def setColorUI_V2(prediction,testbin):
    if prediction == 1 and testbin == True or
        prediction == 0 and testbin == True:
        return (0,255,0) #Green
    if prediction == 1 and testbin == False or
        prediction == 0 and testbin == False:
        return (0,0,255) #Red
```

Listado 3.25. Definición de funciones necesarias

```
    #Visualización de numero de fotogramas y nivel de alarma
    frame = cv2.putText(frame,"F: " + str(intframe),,,,,,)
    frame = cv2.putText(frame,"A: " + str(iDriCount - 37),,,,,,)

    if prediction==1:
        frame = cv2.rectangle(frame,
            (stX, stY), (enX, enY),
                colorInterface,)
        frame = cv2.putText(frame,"State: Careless",
            (10,iHeightY - 10),
                font, 4, colorInterface,2,cv2.LINE_AA)
```

```
        iDrivingStatus_Bad = iDrivingStatus_Bad + 1
            iDriCount = iDriCount + 1
        aAlertFunc( 493, iDriCount)

    if prediction==0:

        frame = cv2.rectangle(frame,
            (stX, stY), (enX, enY),
                colorInterface, )
        frame = cv2.putText(frame,"State: Carefull",
            (10,iHeightY - 10),
                font, 4, colorInterface,2,cv2.LINE_AA)
        iDrivingStatus_Good = iDrivingStatus_Good + 1
        if iDriCount == 37:
            pass
        else:
            iDriCount = iDriCount - 1
```

Listado 3.26. Modificación del fotograma capturado versión 2

3.6. Resultados de la codificación

En esta sección del capítulo se presenta los objetos generados de la codificación: vídeos, conjuntos de datos y modelos. Para comenzar se presenta una descripción del objeto y sus configuraciones, respecto a cada componente.

3.6.1. Obtención de vídeo de prueba

Durante el transcurso del proyecto se crearon alrededor de 20 vídeos de pruebas, por motivos de vibraciones del automóvil, calles en mal estado, posición de la cámara comprometida, datos recurrentes, entre otros. Lo cual, provocó que muchos de estos vídeos fueron descartados, en la Tabla 3.1 se pueden ver los vídeos recopilados y documentados para su uso. Acrónimos utilizados en las columnas, fotogramas por segundo (FPS), ancho y alto del fotograma, total de fotogramas (tFrames), fotogramas descuidados (cFrames).

Tabla 3.1. Resultados de obtención de vídeo de prueba

Nombre	Duración	FPS	Ancho	Alto	tFrames	cFrames
vtest0	86	30.0	1920	1080	2591	209
vtest1	160	30.0	1920	1080	4808	546
v121258	496	30.0	1920	1080	14894	1770
v132600	146	30.0	1920	1080	4394	665
v144800	238	30.0	1920	1080	7152	2427
v173415	231	30.0	1920	1080	6924	727

Al realizar la captura de vídeos con objeto de proponer un sistema funcione en un escenario lo mas parecido a la realidad, cabe mencionar que los vídeos están acelerados teniendo de espera un milisegundo entre fotograma y fotograma. A continuación, se describen los 6 vídeos utilizados en este proyecto, en los cuales, se da una breve explicación de la acción del descuido y el minuto de reproducción en el vídeo:

- vtest0: En este vídeo al iniciar la reproducción se muestra al conductor, acomodando objetos varios dentro del automóvil, luego de eso en el minuto 0:50 revisa su teléfono celular , después de eso conduce hasta estacionarse y tomar varios objetos del lado del copiloto en los últimos 2 segundos.

- vtest1: En este vídeo se presenta con los primeros 16 segundos de acciones varias del conductor, como por ejemplo, abrocharse el cinturón, utilizar el teléfono celular para tomar fotos, acomodar los objetos que se encuentran a su alrededor, entre otros., Después de ello utiliza el teléfono celular en el minuto 1:16.

- v121258: En este vídeo, durante los primeros 3 segundos se presenta al conductor utilizando una computadora portátil y tomando agua, seguido del minuto 1:03 se muestra al conductor leer documentos y utilizar su teléfono celular, desde este punto continuamente toma su teléfono en los minutos 1:29, 2:06 y 2:28, hasta llegar a un estacionamiento y pausar la reproducción del vídeo, siguiendo conduciendo hasta tomar de nuevo el teléfono celular en minuto 4:10 y 6:53.

- v121258: En este vídeo se presenta al conductor utilizando una computadora portátil en el segundo 25 y en el minuto 2:20 hasta finalizar el vídeo.

- v144800: En este vídeo se observa al conductor comiendo en repetidas ocasiones durante toda la reproducción, además de utilizar el celular.

- v173415: En este vídeo se observa al conductor utilizar su celular en los minutos 0:17, 2:15 y 3:00.

La acción de manejar en un entorno real es muy exigente, en consecuencia no se obtiene grandes cantidades de información sobre un "conductor descuidado". Si se observan los vídeos muchos de los descuidos documentados anteriormente se realizan mientras se espera en un semáforo o el vehículo esta estacionado. Lo ideal sería obtener vídeos en un entorno artificial donde se pueda obtener información sin poner en riesgo la vida del conductor ni la de otros.

3.6.2. Generar Set de datos

Rápidamente después de obtener los vídeos del componente anterior en seguida se procedió a extraer los datos y transformarlos en información útil para los conjuntos de datos, el primer vídeo utilizado fue "vtest0.mp4" seguido de "vtest1.mp4", extrayendo la información de manera uniforme respecto a la velocidad de fotogramas (FPS) se proporciona mas información a cada iteración del set de datos. A continuación, una tabla donde se muestran las características de los resultados, véase la Tabla 3.2, al terminar de obtener el conjunto de datos se procede a clasificar la información en sus categorías, véase la Tabla 3.3.

Tabla 3.2. Resultados de generar set de datos

Nombre	Vídeo	Área de trabajo	Medidas	FPS	Total
DSn0	vtest0	Y: 135 X: 840	600x540	2	166
DSn1	vtest0	Y: 160 X: 920	512x512*	10	860
DSn2	vtest0	Y: 320 X: 1050	256x256	30	2591
DSn3	vtest1	Y: 240 X: 1000	256x256	DSn2 + 10 FPS	3868

Análisis de la Tabla 3.2: Con el fin de obtener un set de datos adecuado, se logro generar varios conjuntos los cuales se describen a continuación:

- Conjunto de datos DSn0: Utilizando los vídeos vtest0 se logró extraer el primer conjunto de datos "DSn0" con fotogramas de medidas de 600x540 píxeles, una velocidad de fotogramas de 2 por segundo, con un total de 166 fotogramas a clasificar, al intentar utilizar

este set en el siguiente componente, compilación del modelo, resultaron ser demasiado grandes las imágenes para las especificaciones técnicas del hardware disponible, esto ocasionó tiempos de espera de excesivos y que el sistema se congelara constantemente por falta de memoria RAM, debido a esto en los próximos modelos se utilizaron medidas de 256x256 píxeles como máximo. Véase la Figura 3.10, para visualizar los datos que conforman las categorías, siendo las imágenes de la izquierda con fondo amarillo "other-fixation" y en la derecha con fondo azul "driving-fixation".

Figura 3.10. Conjunto de datos: Numero 0.

- Conjunto de datos DSn1: Inmediatamente después de descartar el set "DSn0", se definió un área de 512x512 píxeles, los cuales, fueron escalados a la mitad resultando en un fotograma de medidas 256x256 píxeles, esta medida resulto ser apto para los recursos del hardware disponible del proyecto, una velocidad de fotogramas de 10 por segundo, con un total de 860 fotogramas. Véase la siguiente Figura 3.11, para visualizar los datos que conforman las categorías, siendo las imágenes de la izquierda con fondo amarillo "other-fixation" y en la derecha con fondo azul "driving-fixation".

Figura 3.11. Conjunto de datos: Numero 1.

- Conjunto de datos DSn2: Intentando mejorar los resultados del modelo M0 compilado con el "DSn1" se creó este set de datos, esta utilizando un área de trabajo del 256x256 píxeles, centrando el área en el rostro del conductor, y una velocidad de fotogramas de 30 por segundo, en pocas palabras se utilizó cada fotograma del vídeo, con un total de 2591 fotogramas resultantes. Véase la siguiente Figura 3.12, para visualizar los datos que conforman las categorías, siendo las imágenes de la izquierda con fondo amarillo "other-fixation" y en la derecha con fondo azul "driving-fixation".

Figura 3.12. Conjunto de datos: Numero 2.

- Conjunto de datos DSn3: Después de compilar los modelos M2 y M3, al ser inferiores en el porcentaje de exactitud al modelo M1, se creó este set de datos intentando mejorar los resultados del modelo M1, se planteo juntar el anterior conjunto de datos "DSn2" y con la siguiente configuración del set: Utilizando el vídeo "vtest1" con una velocidad de fotogramas de 10 por segundo en un área de 256x256, igual que el set anterior centrándose en el rostro del conductor. Añadiendo 1277 nuevos fotogramas al conjunto de datos, resultando en total 3868 fotogramas. Véase la siguiente Figura 3.13, para visualizar los datos que conforman las categorías, siendo las imágenes de la izquierda con fondo amarillo "other-fixation" y en la derecha con fondo azul "driving-fixation".

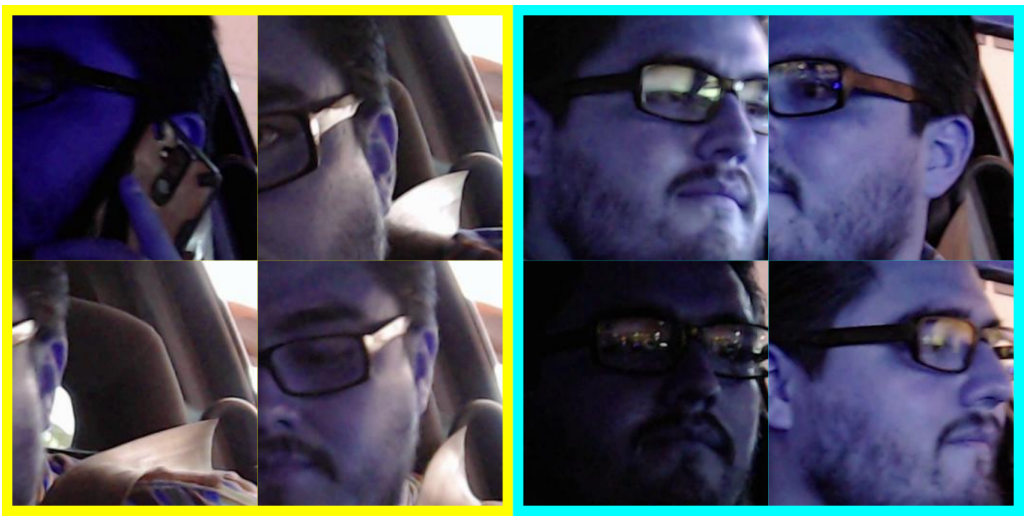


Figura 3.13. Conjunto de datos: Numero 3.

Después de crear set de datos, se procedió a filtrar la información en las carpetas de filtrado: Entrenamiento, validación y datos sin ver con un 75 %, 20 % y 5 % de datos disponibles respectivamente. Al mismo tiempo dentro de estas carpetas separar en clasificaciones binarias anteriormente nombradas como "driving-fixation" y "other-fixation". La selección de fotogramas se rige principalmente por la definición de descuido especificada en el análisis de requisitos. Véase la Tabla 3.3.

Tabla 3.3. Clasificación y filtrado de la información

Nombre	Entrenamiento		Validación		Datos sin ver		Total	
Clasificación	DF	OF	DF	OF	DF	OF	DF	OF
DSn0	92	10	37	6	21	2	150	18
DSn1	590	67	150	16	34	3	784	86
DSn2	1914	204	297	51	125	0	2433	255
DSn3	2595	345	657	80	191	0	3443	425

Como se puede observar en la Tabla 3.3, se puede nota una gran cantidad de datos de parte de la categoría "driving-fixation" teniendo aproximadamente el 10 % del total de los datos, como ejemplo: El set "DSn3" contiene 3868 fotogramas en total, los cuales, se dividen en 3443 en la categoría "driving-fixation", mientras que "other-fixation" solo contiene 425.

3.6.3. Compilar un modelo

Una vez obtenidos el conjunto de datos del componente anterior, lo siguiente es definir una arquitectura que se utilizará para compilar el modelo. Como se mencionó en el diseño del prototipo de sistema, se compone de ciertos números de capas de convolución y filtrado, siguiendo de capas de neuronas densamente conectadas y épocas de aprendizaje.

En el caso del set de datos "DSn0" no se logro compilar un modelo, pues se compone de datos que miden 600x540 píxeles, por consecuencia, el hardware disponible no pudo ser suficiente para manejar un área de trabajo grande, presentando que el sistema se congelara repetidamente y tiempos de compilación largos, por lo que se optó por generar un conjunto de datos diferentes.

A continuación, se hace un breve resumen de la arquitectura utilizada para la compilación de los modelos. Acrónimos utilizados constantemente: Capa de convolución (CL), capa densamente conectada (DL), parámetros entrenables (PE), parámetros (Acc) y pérdida (Loss):

- Tamaño del lote: Se decidió utilizar un lote de 32 unidades, por ser el mas común al entrenar CNN.
- Capas de entrada: Respecto al set de datos, como ejemplo: En "DSn1" siendo 256x256 píxeles y 3 capas por el modelo de color RGB.
- Capas de convolución: En 4 capas serían suficientes para concretar un modelo decente, reduciendo las características de entrada 256x256 a una salida de 16x16 de medida de forma, utilizando un filtro de agrupamiento de características de 2x2, utilizando la función de activación "ReLU", pues el objetivo es extraer las características dominantes de la imagen, acelerando la convergencia de la CNN.
- Capa densamente conectadas: Utilizando de 1 a 3 capas se obtendría un aprendizaje apropiado para presentar un modelo de calidad.

- Perceptrón: La ultima capa incluye la función de activación “sigmoid”.
- Optimizador: Se utiliza el algoritmo “Adam” por ser el mas rápido en converger en épocas tempranas, siendo mas rápido que “SGD” o “RMSprop”, dejando por defecto en 0.001 la taza de aprendizaje.

A continuación, una descripción de los modelos generados con éxito. Incluyendo el set de datos utilizado, el tamaño de las capas convolucionales y sus filtros de agrupamiento, seguido de las capas densamente conectadas, posterior sus épocas de entrenamiento. Seguido de la Tabla 3.4 donde se muestra las descripciones de una manera mas visible, adicionalmente, incluyen los parámetros a entrenar y resultados de exactitud y pérdida final:

- Modelo M0: Este modelo se compiló utilizando el set “DSn1”, cada capa de convolución son las siguientes en orden: 128, 128, 64 y 32. Utilizando filtros de 3x3 por cada CL, añadiendo la primera capa de 512 neuronas y finalizando el perceptrón, con 25 épocas de aprendizaje.
- Modelo M1: Este modelo se cambia el set de datos al “DSn2”, adicionalmente se cambian las capas de convolución a 256, 64, 32 y 16 con filtros de 2x2 por cada CL, a excepción de CL 1 que utiliza un filtro de 3x3 debido a que el hardware se congelaba constantemente, con 10 épocas de aprendizaje.
- Modelo M2: Respecto al Modelo M1 se hacen cambios a la parte de red neuronal, se duplica la primera capa de neuronas densamente conectadas de 1024 a 2048 y la segunda DL de 64 a 512.
- Modelo M3: Respecto al modelo M2 se cambia la segunda capa de convolución de 64 a 128.
- Modelo M4: Se diseñaron los Modelos M4-0, M4-1 y M4-2 con el propósito de comparar modelos para ver su progreso de aprendizaje y poder concretar el número de épocas

ideales para el entrenamiento y poder calcular el ajuste del modelo, siendo 10, 25 y 50 épocas de entrenamiento respectivamente. Se compiló utilizando el set "DSn3" con diferencias de agregar una cuarta capa de neuronas densamente conectadas: DL 2 de 512 a 1024, DL 3 de 1 a 512 y para finalizar el perceptrón en DL 4.

- Modelo M5: Como ultimo intento se agranda la tercera capa de neuronas densamente conectadas de 512 a 1024, con 25 épocas de entrenamiento.

Tabla 3.4. Resultados de la compilación

Nombre	M0	M1	M2	M3	M4-0	M4-1	M4-2	M5
Dataset	DSn1	DSn2	DSn2	DSn2	DSn3	DSn3	DSn3	DSn3
CL1	128(3)	256(3)	256(3)	256(3)	256(3)	256(3)	256(3)	256(3)
CL2	128(3)	64(2)	64(2)	128(2)	128(2)	128(2)	128(2)	128(2)
CL3	64(3)	32(2)	32(2)	32(2)	32(2)	32(2)	32(2)	32(2)
CL4	32(3)	16 (2)	16(2)	16(2)	16(2)	16(2)	16(2)	16(2)
DL1	512	1024	2048	2048	2048	2048	2048	2048
DL2	1	64	512	512	1024	1024	1024	1024
DL3	N/A	1	1	1	512	512	512	1024
DL4	N/A	N/A	N/A	N/A	1	1	1	1
Épocas	25	10	10	10	10	25	50	25
PE	3455713	1476849	3788913	3862705	5436593	5436593	5436593	5961905
Acc(%)	90.47	93.27	91.84	92.22	92.78	94.46	95.77	94.88
Loss(%)	20.09	18.67	18.56	19.51	20.23	14.06	11.72	13.09

Un error recurrente de los modelos previamente descritos esta en la primera capa convolucional, en esta se realiza una capa de agrupamiento y filtro de 3x3, reduciendo el "frame", para ejemplificar, en el modelo M1 tiene una entrada de medidas 256x256, después de aplicar el filtro de agrupación de características de 3x3, dejaría el "frame" de tamaño 84x84 píxeles, lo cual, lo ideal seria que la siguiente capa de convolución fuera de este tamaño, al contrario de lo que se específico en el análisis del programa, al utilizar las capas en una reducción lineal de filtro 2x2 siendo: 256, 128, 64, 32 y 16 píxeles. Aunque parece que no afecta a los resultados de test e implementación.

Capítulo 4
Resultados y Discusiones

En la sección anterior se grabaron 6 vídeos, se crearon 4 conjuntos de datos, se compilaron 8 modelos de clasificación binaria en Keras, con el objetivo de desarrollar un sistema de visión artificial, el cual, detecte los gestos del cuerpo que pueden traducirse como un descuido.

Utilizando los vídeos y modelos creados anteriormente, se ejecutarán las pruebas: test 0, test 1 e implementación del modelo. Incluyendo como verificación la lista de conductores para tener una noción de exactitud, a continuación, se describen los procesos y se muestran los resultados.

4.0.1. Pruebas del modelo

Test 0

Como se mencionó en el diseño del prototipo del sistema en el apartado de compilación del modelo esta prueba de exactitud 0, se ejecutó al momento de compilar el modelo, los resultados son los siguientes, véase la Tabla 4.1.

Tabla 4.1. Test 0: Resultados de las pruebas

Nombre	M0(%)	M1(%)	M2(%)	M3(%)	M4-0(%)	M4-1(%)	M4-2(%)	M5(%)
Acc	91.89	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Loss	16.36	02.39	00.53	01.68	04.58	02.91	00.65	01.38

Análisis de la Tabla 4.1: En el Modelo M0 presentó la mayor perdida con 16.36 % y ganando por mas del 10 %, además de, presentar una exactitud del 91.89 % en comparación de otros modelos compilados que ofrecen el 100 %. Aunque estos valores solo evalúan la entrada y al salida de parámetros, no son útiles al evaluar un modelo.

Test 1

De manera subsecuente se presentarán los resultados de exactitud, a fin de evaluar el desempeño del prototipo de sistema, aceptando los resultados y decidiendo si iterar en la metodología o dar por cumplido el objetivo general.

Esta prueba de exactitud 1 se ejecutó con los valores de área por defecto del conjunto de datos DSn2, véase la Tabla 4.2, y en conjunto con la "lista de conductores descuidados" se puede obtener una referencia de la exactitud del modelo, posteriormente, se presentan los resultados del test 1, véase la Tabla 4.3.

Tabla 4.2. Test 1:Área de trabajo:

Nombre	Área de trabajo	Medida
vtest0	Y:320 X:1050	256x256
vtest1	Y:240 X:1000	256x256

Análisis de la Tabla 4.3: En primer instancia el M0 resulta ser demasiado prometedor con un valor de exactitud del 91.93 % pero el modelo resultó ser un total fracaso, no logró

Tabla 4.3. Test 1: Resultados de las pruebas

Nombre	M0(%)	M1(%)	M2(%)	M3(%)	M4-0(%)	M4-1(%)	M4-2(%)	M5(%)
vtest0	91.93	97.14	94.05	96.21	95.13	96.52	97.80	96.79
vtest1	91.86	95.77	94.57	95.88	95.77	94.67	93.73	95.23

clasificar ningún fotograma como descuidado, véase la Tabla 4.4. Por esta razón se crearon mas modelos siendo mas selectivos en el área de trabajo y mas generosos en el apartado de cantidad de datos del siguiente "DSn", además de agregar una tercera capa de neuronas densamente conectadas.

Tabla 4.4. Test 1: Fotogramas clasificados como conductores descuidados

Nombre	M0(f)	M1(f)	M2(f)	M3(f)	M4-0(f)	M4-1(f)	M4-2(f)	M5(f)
vtest0	0	199	57	125	85	121	160	130
vtest1	0	452	140	413	264	199	244	222

4.1. Implementación del prototipo de sistema

En resumen de la prueba de exactitud 1, los ocho modelos en su test 1 pudieron obtener resultados correctos de 90.80 % hasta el 97.80 %, ya dependiendo del modelo y vídeo utilizado para la prueba. En esta sección se presenta la implementación de los modelos en nuevos vídeos, esperando resultados similares en los vídeos de "vtest0" y 'vtest1".

Como se dijo en el diseño del prototipo del sistema se pondrán a prueba los modelos en entornos y vídeos diferentes. A continuación, se presentan las áreas de trabajo de cada vídeo utilizadas, en las cuales se visualiza perfectamente el rostro del conductor pudiendo cubrir la cabeza en todo momento de la reproducción del vídeo, véase la Tabla 4.5.

Tabla 4.5. Área de trabajo: Implementación

Nombre	Área de trabajo (píxeles)	Medida (píxeles)
vtest0	Y:230 X:990	384x384
vtest1	Y:180 X:900	384x384
v121258	Y:210 X:950	384x384
v132600	Y:400 X:850	384x384
v144800	Y:185 X:1000	384x384
v173415	Y:165 X:850	384x384

Utilizando la lista de conductores descuidados de los vídeos obtenidos, se logró evaluar con mas exactitud el comportamiento de los modelos, téngase en cuenta que son resultados aplicando el modelo al vídeo completo. A continuación, véase los resultados en la siguiente Tabla 4.6. Como se menciono en la prueba de exactitud 1, el Modelo M0 aunque muestre un promedio muy alto, en los resultados de detección de descuidos sus resultados son cero y queda completamente descartado por arrojar predicciones totalmente estáticas, por ende excluido de las siguientes implementaciones.

Tabla 4.6. Resultados de la implementación

Nombre	M1(%)	M2(%)	M3(%)	M4-0(%)	M4-1(%)	M4-2(%)	M5(%)
vtest0	93.55	93.82	94.44	94.75	95.94	96.41	96.60
vtest1	93.44	92.55	94.28	94.55	93.59	92.34	94.69
v121258	70.04	92.36	78.94	91.23	79.91	79.49	79.58
v132600	83.73	93.61	94.88	94.38	95.51	95.35	95.58
v144800	67.64	67.44	83.82	76.13	78.00	81.69	82.85
v173415	98.23	95.01	98.38	97.71	97.68	97.90	97.97

4.1.1. Evaluación de conductores descuidados

Para evitar el sesgo que proporciona los resultados anteriores, a continuación, solo se evaluaran los fotogramas que se encuentran en la lista de conductores descuidados en vez de los fotogramas totales, en pocas palabras evaluar los fotogramas que se consideran "descuidados", con el fin de proporcionar un resultado sólido y poder dar información para los siguientes modelos a compilar, véase los resultados en la siguiente Tabla 4.7:

Tabla 4.7. Evaluación de conductores descuidados

Nombre	M1(%)	M2(%)	M3(%)	M4-0(%)	M4-1(%)	M4-2(%)	M5(%)
vtest0	88.03	23.92	46.88	35.88	54.06	81.33	62.20
vtest1	64.19	25.83	64.19	42.71	49.61	42.45	50.12
v121258	73.22	47.06	68.98	54.35	66.21	67.90	67.62
v132600	61.35	11.12	61.05	24.81	50.37	64.51	41.05
v144800	80.52	5.78	56.94	31.26	38.25	56.28	55.12
v173415	92.43	54.88	92.02	79.22	91.74	94.63	94.08

El Modelo M1 compilado con el conjunto de datos DSn2, resulto ser el mejor en detectar conductores descuidados en los siguientes vídeos: "vtest0", "vtest1", "v144800" y "v121258". Seguido del modelo M4-2 compilado con el conjunto de datos DSn3 durante 50 épocas, que fue el mejor en los vídeos: "v132600" y "v173415".

Cabe resaltar que entre el modelo M1 y M2 existe una abismal diferencia de resultados

del 50.12 % en promedio y las diferencias entre estos modelos radica en las capas de neuronas densamente conectadas: Duplicando la capa DL1 de 1024 a 2048 y la capa DL2 de 64 a 512.

Se diseñaron los Modelos M4-0, M4-1 y M4-2 con el propósito de comparar modelos para ver su progreso de aprendizaje del modelo. Teniendo los resultados se puede concluir que la tendencia va en aumento, presentando porcentajes de 42.22 % hasta un 65.53 %, ganando un 23.31 % en los promedios de todos los vídeos, en 40 épocas de diferencia, en comparación con el modelo M1 presentaría una diferencia de arquitectura: en la capa CL 2 y DL 3, DL 1, DL 3. Que tiene como consecuencia que aumenten los parámetros entrenables en 3959744, por lo que aumentar las épocas de entrenamiento resultó ser la opción ideal para aumentar el porcentaje de exactitud.

4.1.2. Evaluación de conductores cuidadosos

Para evitar el sesgo que proporciona los resultados anteriores, a continuación, solo se evaluaran los fotogramas que no se encuentran en la lista de conductores descuidados en vez de los fotogramas totales, en pocas palabras evaluar los fotogramas elegidos como "cuidadoso", con el fin de proporcionar un resultado sólido y poder dar información para los siguientes modelos a compilar, véase los resultados en la siguiente Tabla 4.8.

Tabla 4.8. Evaluación de conductores cuidadosos

Nombre	M1(%)	M2(%)	M3(%)	M4-0(%)	M4-1(%)	M4-2(%)	M5(%)
vtest0	94.20	99.95	98.61	99.95	99.83	97.69	99.70
vtest1	92.07	90.87	91.75	91.30	91.30	88.18	91.39
v121258	85.95	99.91	98.39	99.81	99.33	99.03	99.35
v132600	98.98	99.89	99.43	99.91	99.81	97.50	99.14
v144800	62.03	99.59	97.43	99.40	99.19	95.32	96.99
v173415	98.90	99.93	99.17	99.80	98.07	98.14	98.46

4.1.3. Resultados finales

Para concluir con las pruebas de implementación, en la siguiente Tabla 4.9 se presentan los promedios de los modelos ordenándolos por su desempeño al clasificar fotogramas descuidados y cuidadosos, presentándolos en porcentajes.

Tabla 4.9. Resultados finales: Promedio de modelos

Nombre	Descuidados (%)	Cuidadosos (%)	Promedio(%)
M1	76.62 %	88.69 %	82.66 %
M4-2	67.85 %	95.98 %	81.91 %
M3	65.01 %	97.46 %	81.24 %
M5	61.70 %	97.51 %	79.60 %
M4-1	58.37 %	97.92 %	78.15 %
M4-0	44.71 %	98.36 %	71.53 %
M2	28.10 %	98.36 %	63.23 %

Anteriormente en la tabla de evaluación de conductores descuidados, véase la Tabla 4.7, se observaron que los modelos M1 y M4-2 eran los más aptos para clasificar fotogramas "descuidados", por el contrario los modelos M2 y M4-0 eran los menos indicados. Entre modelos los resultados de implementación son muy notables las diferencias teniendo resultados de 8.77 % hasta el 48.53 % de promedio.

Mientras que la tabla de evaluación de conductores cuidadosos, véase la Tabla 4.8, contrario a las evaluaciones anteriores, se muestran que los modelos aptos para clasificar fotogramas "cuidadosos", teniendo a M2 y M4-0 como los mejores modelos, mientras que los modelos M1 y M4-2 eran los menos indicados. Entre modelos los resultados de implementación, a diferencia de la anterior evaluación, los resultados son mas pequeños, al comparar las evaluaciones entre los modelos teniendo resultados de 0.44 % hasta el 9.67 % de promedio.

Los vídeos utilizados en la implementación pueden ser factor decisivo al elegir los modelos mas aptos, del mismo modo que los modelos, se requiere comparar los resultados para evaluar la realización del objetivo general. En este caso se presenta la "dificultad" para el modelo al

clasificar el fotograma, como ejemplo, En la siguiente Tabla 4.10 se resumen los promedios de clasificaciones exitosas respecto a los vídeos:

Tabla 4.10. Resultados finales: Promedio exactitud respecto a los vídeos

Nombre	Descuidados(%)	Cuidadosos(%)	Promedio(%)
v173415	85.57	98.92	92.25
v121258	63.62	97.40	80.51
vtest0	56.04	98.56	77.30
v132600	44.89	99.24	72.07
vtest1	48.44	90.98	69.71
v144800	46.31	92.85	69.58

En la tabla anterior se puede notar que el vídeo vtest1 es la implementación con mas fallos de exactitud , mientras que en v173415, por el contrario es la implementación con una exactitud del 92.25 %, de tal manera que entre los vídeos tienen una diferencia de entre el 11.51 % hasta el 28.11 % de promedio.

Se puede concluir que la "dificultad" de clasificación resulta de los siguientes áreas de fallos:

- Lista de conductores descuidados: Al utilizar un recurso como una base de datos se implica la idea de que sus datos sean totalmente acertados, respecto al análisis del descuido, aunque se estima que exista entre el 1-3 % de resultados erróneos. Es mas probable que otras áreas de la implementación fallen.
- Set de datos: Como se muestra en la Tabla 3.3, la diferencia de resultados en la evaluación de clasificación, puede deberse a no tener suficiente ejemplos sobre un conductor "descuidado".
- Predicciones y fotogramas:

    1. Fotogramas borrosos: Se ha observado que fallan las predicciones al hacer movimientos muy rápidos, en que los fotogramas del rostro del conductor resultan

borrosos, como ejemplo, al caer en un bache o conducir en una calle irregular.

2. Fotogramas falsos: Existen fotogramas que se clasifican y se validad erróneamente, como falsos positivos o positivos falsos. Esto se puede evaluar respecto a la lista de conductores descuidados o dependiendo del caso, estimar si el modelo debería clasificar correctamente el fotograma y es necesario ejemplos en el set de datos, mas entrenamiento, una red neuronal mas robusta, entre otros.

- Área de implementación: Utilizando un área de trabajo que no permita visualizar el rostro del conductor o que aparezca un elemento externo que interfiera en la visión artificial, por consecuencia, se evalué de manera incorrecta.

4.2. Modelo final

En conclusión, comparando los modelos y sus resultados en vídeos en los anteriores párrafos y tablas, se presentan un modelo apto para cumplir los objetivos generales, y es el modelo M4-2. Aunque el modelo M1 tenga mejores resultados respecto a clasificación de descuidados, por un lado, en clasificación de conductores cuidadosos deja mucho de desear, como consecuencia merma el promedio de la evaluación. Lo que se presenta es el modelo M4-2, el cual puede dar cabida a otro tipo de implementaciones en el área de visión artificial.

Aunque no se pudo aumentar el rendimiento de los modelos, la mayoría de estos no consiguen llegar al 70 % de fotogramas descuidados clasificados correctamente, se puede obtener un resultado satisfactorio evaluando los vídeos en la implementación, en [21] se puede observar el comportamiento del modelo M4-2 en los seis vídeos anteriormente documentados.

A partir de este momento es recomendable analizar los fotogramas clasificados incorrectamente, luego de eso, proponer una solución y aplicarla, como ejemplo: Lo ideal sería obtener los fotogramas de los vídeos que no formaron parte del set de datos, específicamente los que pertenecen a "lista de conductores descuidados" y agregarlos a un set de datos "DSn4", seguido de utilizar las estructuras a los modelos M1 o M4-2 que mostraron mejores resultados, por al menos 25 o mas épocas de entrenamiento. Con la finalidad de compilar un modelo mas grande, que pueda sobrepasar los resultados de los modelos anteriores.

Capítulo 5
Conclusiones

En este capítulo se mencionan las partes mas relevantes en la implementación de prototipo de sistema creado anteriormente.

5.1. Con respecto al objetivo de la investigación

Recapitulando en el capítulo uno, en el apartado de objetivo general, esta investigación consistió en desarrollar un prototipo de sistema que detecte por medio de visión artificial los gestos del cuerpo que se pueden traducir en un descuido por parte del individuo y se activen alarmas visuales y sonoras inmediatamente se incurra en el acto de descuido.

- Los resultados del modelo están definidos por el ángulo del rostro del conductor, pudiendo clasificar como conductor descuidado, cuando el individuo quita la vista fuera del camino o algunas ocasiones cuando se le cambia sutilmente a la radio del automóvil.
- El comportamiento del modelo es perceptible a la proximidad del rostro del conductor, pudiendo detectar como conductor cuidadoso cuando esta bien posicionado en el asiento del piloto. Por el contrario si el rostro se aproxima a la cámara, se expresa como conductor descuidado, pues las acciones pueden considerarse descuidadas aunque se observe al frente.

- El modelo es capaz de clasificar como un conductor descuidado, la acción de sostener cualquier objeto cerca de la barbilla, a excepción de las manos, que tiene un comportamiento mas perceptible.

- Al utilizar una cámara web común en configuración de la visión artificial, se requiere de un escenario debidamente iluminado, lo ideal sería utilizar cámaras con sensores infrarrojos para ambientes con baja o nula iluminación.

- El desempeño en la implementación del modelo no presento ralentizaciones en su uso, pudiendo manejar 30 fotogramas por segundo utilizándose como configuración en la cámara web o los vídeos documentados.

- Los modelos compilados y la configuración de visión artificial pueden ser utilizados en otras plataformas que soporten el software usados en este proyecto: Python-OpenCV, TensorFlow y Keras.

5.2. Recomendaciones para futuras investigaciones

Para desarrollar este proyecto fue fundamental crear conjuntos de datos, compilar modelos y evaluarlos, resulta un proceso delicado y complejo, pues hasta un mínimo detalle puede cambiar el comportamiento previsto de los modelos compilados. En este prototipo de proyecto se demostró que utilizando el campo de clasificación de imágenes que proporciona la visión artificial, para ser mas específico, entrenando una red neuronal convolucional en una clasificación binaria fue posible llegar a los objetivos.

Para llegar a la solución completa del problema planteado es necesario seguir con el desarrollo e implementación, como por ejemplo:

- Desarrollar un sistema de visión artificial clasificador de imágenes por categorías como: Vista frontal, frontal derecha, lateral izquierdo, lateral derecho y vista en el espejo retrovisor.
- Utilizar una configuración de visión artificial estéreo, que permitan obtener mas información del conductor.
- Utilizar dos o mas modelos de clasificación de imágenes para deducir el estado del conductor.
- Utilizar este modelo desde el arranque del vehículo, implementando en una computadora de placa única tipo Raspberry.
- Utilizar otros campos de la visión artificial para detectar objetos, como ejemplo, celulares, auriculares, la posición exacta del rostro y mirada del conductor.

Bibliografía

[1] Ravedave, "Crop factor", Accedido en 2018-10-15 a Commons.wikimedia.org, 2005.

[2] Latin-NCAP, "About US", Accedido en 2018-09-09 a www.latinncap.com, 2015.

[3] J. L. Gámez Núñez , *Estudio de los sistemas inteligentes (sensores) en los vehículos que permitan evitar y disminuir el índice de accidentes de transito*, PhD thesis, Universidad de Guayaquil, 2018.

[4] Instituto Nacional de Estadística y Geografía (INEGI), "Síntesis Metodológica de la Estadística de Accidentes de Tránsito Terrestre en Zonas Urbanas y Suburbanas", 2009.

[5] Comisión nacional de seguridad, "Accidentes y sus Factores", Accedido en 2018-09-09 a cns.gob.mx, 2015.

[6] Fundación MAPFRE, "El factor humano en la Seguridad Vial", Accedido en 2018-09-09 a www.fundacionmapfre.org, 2018.

[7] J. L. García, E. Rogado, R. Barea, L. M. Bergasa, E. López, M. Ocaña, and D. Schleicher, "Sistema detector de fatiga en la conducción", p. 6, 2009.

[8] M. Yeferson and B. Torres, "Aplicación móvil para la detección de somnolencia de un conductor aplicando visión artificial Aplicación móvil para la detección de somnolencia de un conductor aplicando visión artificial", , no. June, 2018.

[9] I. J. García Enríquez, M. N. Ibarra Bonilla, and J. M. Ramírez Cortés, *Segmentación de rostro por color de la piel aplicado a detección de somnolencia en el conductor*, PhD thesis, Instituto Nacional de Astrofísica, Óptica y Electrónica, 2009.

[10] N. M. Cruz, *Desarrollo de un sistema avanzado de asistencia a la conducción en tiempo real para la detección de peatones en entornos urbanos complejos*, PhD thesis, Universidad Carlos III de Madrid, 2013.

[11] Volvo Car Corporation, "Intellisafe, Un nuevo concepto de seguridad.", Accedido en 2018-09-09 a www.volvocars.com/mx, 2018.

[12] G. Lozano and J. Orduz, *Diseño de un sistema de visión artificial para la revisión del nivel de llenado de bebidas embotelladas*, PhD thesis, Universidad autónoma del Caribe, 2015.

[13] Volvo Car Corporation, " Mobileye® advanded vehicle technologies power Volvo car's driver alert control (DAC) system", 2007.

[14] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite", *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, 2012.

[15] Centro Integrado Politécnico ETI, "Visión Artificial", p. 10, 2015.

[16] J. C. Paredes, *Teoría y Aplicación de la Informática 2 Visión artificial*, PhD thesis, Universidad Católica Nuestra Señora de la Asunción, 2015.

[17] V. M. Arévalo, J. González, and G. Ambrosio, "La Librería De Visión Artificial OpenCV Aplicación a La Docencia E Investigación", pp. 1–6, 2002.

[18] A. López, M. Prieto, and Á Ramírez, "Enseñanza del Procesamiento de Imáganes en Ingeniería usando Python", vol. 3, 2015.

[19] H. Espinosa, *Diseño E Implementación De Un Sistema De Seguridad Y Alerta Para Vehículos, Basado En Reconocimiento Facial Y Localización GPS, En Una Raspberry Pi B Plus.*, PhD thesis, Escuela Politécnica Nacional, 2016.

[20] E. R. Barriga Caballero, *Aplicación práctica de la visión artificial para el reconocimiento de rostros en una imagen, utilizando redes neuronales y algoritmos de reconocimiento de objetos de la biblioteca OpenCV*, PhD thesis, Universidad distrital Francisco José de Caldas, 2017.

[21] L. A. Villanueva Betancour, "Detect driver careless - youtube", Accedido en 2019-11-21 a www.youtube.com/mx, 2019.