IMPLEMENTACIÓN DE UN SISTEMA PARA LA DETECCIÓN DE PELIGROS POTENCIALES PARA MOTOCICLISTAS BASADO EN INFORMACIÓN VISUAL

ALEXANDER BERMUDEZ CASTAÑEDA

UNIVERSIDAD SANTO TOMÁS

PROGRAMA DE INGENIERÍA ELECTRÓNICA

BOGOTÁ D.C.

2021

IMPLEMENTACIÓN DE UN SISTEMA PARA LA DETECCIÓN DE PELIGROS POTENCIALES PARA MOTOCICLISTAS BASADO EN INFORMACIÓN VISUAL

ALEXANDER BERMUDEZ CASTAÑEDA

Proyecto de grado presentado como requisito para optar al título de INGENIERO ELECTRÓNICO

Directora

Ing. SINDY AMAYA, MSc.

Codirector

Ing. JUAN MANUEL CALDERÓN, Ph.D.

UNIVERSIDAD SANTO TOMÁS

PROGRAMA DE INGENIERÍA ELECTRÓNICA

BOGOTÁ D.C.

2021

Nota de Aceptación

Firma del jurado 1 - Evaluador

Firma del jurado 2 - Evaluador

Firma del jurado 3 - directora

Bogotá D.C., 23 de abril de 2021

Dedico este trabajo a Dios quien me brindó la gran oportunidad de estudiar y seguir mis sueños, a mis padres Darío y Nelly, porque a pesar de la distancia me han brindado su apoyo y motivación incondicional durante mi proceso de formación, a mi hermano Hernán y a mi familia por sus consejos que me permitieron crecer como persona y profesionalmente.

A mi buen amigo Jorge Ivan Nieto, con quien compartí buenos momentos y quien desafortunadamente ya no está con nosotros.

AGRADECIMIENTOS

Agradecimientos a mi directora Sindy Paola Amaya y codirector Juan Manuel Calderón por su apoyo y enseñanzas en el desarrollo de este proyecto.

Agradecimiento a la Universidad Santo Tomás por brindarme todos los materiales necesarios para el desarrollo de este proyecto.

GLOSARIO

1. **BN:** La normalización por lotes consiste en añadir un paso extra habitualmente entre las neuronas y la función de activación, con el objetivo de normalizar las activaciones de salida.

2. **COCO:** Es un conjunto de datos de detección, segmentación y subtítulos de objetos a gran escala.

3. **CPU:** La unidad central de procesamiento es el hardware dentro de un ordenador u otros dispositivos programables cuyo trabajo es interpretar las instrucciones de un programa informático mediante la realización de operaciones aritméticas y lógicas.

4. **CUDA:** Son las siglas de arquitectura unificada de dispositivos de cómputo que hace referencia a una plataforma de computación en paralelo.

5. **GPS:** El sistema de posicionamiento global es un sistema de radio navegación de los Estados Unidos de América, que proporciona servicios de posicionamiento, navegación y cronometría gratuita a usuarios de todo el mundo.

6. **GPU:** La unidad de procesamiento lógico es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante.

7. **IoT:** El internet de las cosas se puede definir como la agrupación e interconexión de dispositivos y objetos a través de una red.

8. **ReLU:** Es una función de activación lineal rectificada, se puede decir que es una función lineal por partes que genera la entrada directa si es positiva, de lo contrario, generará cero.

9. **TPU:** Es la unidad de procesamiento tensorial el cual es un circuito integrado de aplicación específica y acelerador de inteligencia artificial desarrollado por Google para el aprendizaje automático con redes neuronales artificiales.

CONTENIDO

1. INTRODUCCIÓN 14

2. PROBLEMA 15

3. ANTECEDENTES 17

3.1 INTELIGENCIA ARTIFICIAL 17
3.2 VISIÓN ARTIFICIAL 17
3.2.1 ETAPAS EN UN PROCESO DE VISIÓN ARTIFICIAL 17
3.3 TRABAJOS PREVIOS 18
3.3.1 HELMY 19
3.3.2 SISTEMA DE SEGURIDAD PARA MOTOCICLISTA MEDIANTE EL USO DEL INTERNET DE LAS COSAS 19
3.3.3 IMPLEMENTACIÓN Y ANÁLISIS DE CASCO INTELIGENTE 19
3.3.4 SISTEMA DE TRANSPORTE INTELIGENTE PARA PREVENCIÓN Y DETECCIÓN DE ACCIDENTES 19
3.3.5 SISTEMA DE IDENTIFICACIÓN DE ACCIDENTES Y NAVEGACIÓN 20
3.3.6 PROCESAMIENTO DE VIDEO PARA DETECCIÓN DE AUTOMÓVILES DE LA PARTE TRASERA DE UNA MOTOCICLETA Y PREVENCIÓN DE ACCIDENTES. 21
3.3.7 SISTEMA DE VISIÓN PARA CASCOS DE MOTOCICLETA 24

4. JUSTIFICACIÓN 25

5. OBJETIVOS 26

5.1 OBJETIVO GENERAL 26
5.2 OBJETIVOS ESPECÍFICOS 26

6. MARCO TEÓRICO 27

6.1 SOFTWARE 27
6.1.1 APRENDIZAJE DE MÁQUINA 27
6.1.2 REDES NEURONALES ARTIFICIALES 27
6.1.3 CONVOLUTIONAL NEURAL NETWORK (CNN) 28
6.1.4 TENSORFLOW LITE 28
6.1.5 MOBILNETS 28
6.1.6 TEXT TO SPEECH 30
6.2 HARDWARE 30
6.2.1 DEV BOARD 30
6.2.2 T.P.U. 31
6.2.3 COMPRENSIÓN DE LA ESTRUCTURA Y FUNCIONAMIENTO DE LA TARJETA CORAL 31
6.2.4 JETSON NANO 31
6.2.5 REALSENSE 32
6.2.6 TRIANGULACIÓN MEDIANTE VISIÓN ESTÉREO 33

7. DISEÑO Y EJECUCIÓN DEL PROYECTO 38

7.1 ALGORITMO DE RECONOCIMIENTO DE OBJETOS 38
7.1.1 GENERACIÓN DEL MODELO DE INFERENCIA DE OBJETOS QUE SE EJECUTARÁ 38
7.1.2 SEGMENTACIÓN SEMÁNTICA 49
7.2 DISTANCIA DE LOS VEHÍCULOS 51
7.2.1 RADAR HW-MS03 51
7.2.2 RADAR DE MICROONDAS HB100 52

7.2.3 TRIANGULACIÓN MEDIANTE VISIÓN ESTEREO *53*
7.2.4 USO DE REALSENSE D435I *55*
7.2.4 TIEMPOS DE RESPUESTA DE LOS SENSORES *58*
7.3 CÁLCULO VELOCIDAD RELATIVA DE LOS VEHÍCULOS 58
7.4 SISTEMA DE ASISTENCIA 59
7.4.1 RESPUESTA DEL SISTEMA DE ASISTENCIA *60*
8. RESULTADOS DEL PROYECTO 62
9. CONCLUSIONES Y TRABAJO FUTURO 82
10. BIBLIOGRAFÍA 83
11. ANEXOS 86
ANEXO 1: 86
ANEXO 2: 87
ANEXO 3: 88

LISTA DE FIGURAS

Figura 1. Etapas de un sistema de visión artificial [12]. 18
Figura 2. Diagrama de bloques de las 4 partes en que se divide el sistema propuesto [17]. 21
Figura 3. Ubicación final de la cámara (sobre el casco del motociclista) [18]. 22
Figura 4. Imagen procesada dividida en regiones [18]. 22
Figura 5. Diagrama de bloques del algoritmo usado [18]. 23
Figura 6. Prototipo de un casco para motocicletas con un sistema de visión que permite ver 360 grados al mismo tiempo [19]. 24
Figura 7. Estructura general de una neurona biológica [25]. 27
Figura 8. Red neuronal artificial [26]. 28
Figura 9 Convolución estándar (izquierda), convolución separable en profundidad (derecha) con BN y ReLU [29]. 29
Figura 10. Coral Dev Board [32]. 30
Figura 11. Compute primitive [33]. 31
Figura 12. Jetson Nano. Figura [35] 32
Figura 13. Cámara Intel RealSense [36]. 32
Figura 14. Pasos detección de la profundidad [37]. 33
Figura 15. Representación del triángulo formado por las cámaras y el punto X, fuente autor 34
Figura 16. Representación del ángulo de visión de la cámara, fuente autor 35
Figura 17. Representación de un ángulo diferente de teta medios, fuente autor. 36
Figura 18. Descripción general del desarrollo del diseño y la ejecución del proyecto, fuente autor. 38
Figura 19. Flujo de trabajo para crear un modelo para Edge TPU [34]. 39
Figura 20. Cámara GoPro ubicada en la parte superior del casco, fuente autor 40
Figura 21. Ejemplo de cuadro tomado por la cámara, fuente autor 41
Figura 22. Uso de la herramienta LabelImg, fuente autor 41
Figura 23. Diagrama del algoritmo utilizado para descargar imágenes y etiquetas de COCO, fuente autor 42
Figura 24. Estructura general de la carpeta que contiene los conjuntos de datos, fuente autor 43
Figura 25. Muestra del conjunto de datos generado, fuente autor. 44
Figura 26. Diagrama del algoritmo utilizado para crear el conjunto de datos de prueba y entrenamiento, fuente autor 45
Figura 27. Diagrama del algoritmo utilizado para transferir la información necesaria de los archivos XML a un CSV, fuente autor 45
Figura 28. Mapa de etiquetas, fuente autor 46
Figura 29. Flujo de configuraciones realizadas al archivo de configuración, fuente autor. 47
Figura 30. Pérdida total entrenamiento de la red MobileNet SDD v1 y conjunto de datos híbrido, fuente autor 48
Figura 31. Pérdida total entrenamiento de la red MobileNet SDD v2 y conjunto de datos híbrido, fuente autor 48

Figura 32. Pérdida total entrenamiento de la red MobileNet SDD v2 y conjunto de datos etiquetado manualmente, fuente autor 49
Figura 33. Flujo de instalación de paquetes necesarios para ejecutar la segmentación de imágenes, fuente autor 49
Figura 34. Diagrama del algoritmo utilizado para realizar la segmentación de imágenes, fuente autor. 50
Figura 35. Imagen de entrada al algoritmo de segmentación, fuente autor 50
Figura 36. Resultado de la segmentación de imágenes, fuente autor 51
Figura 37. Radar HW - MS03, fuente autor 51
Figura 38. Circuito de salida del sensor, fuente autor 52
Figura 39. Radar HB100, fuente autor 52
Figura 40. Circuito amplificador de señal, fuente autor 53
Figura 41. Montaje inicial de las dos cámaras, fuente autor 54
Figura 42. Diagrama del algoritmo utilizado para calcular la profundidad a partir de visión estéreo, fuente autor 55
Figura 43. Sistema operativo descargado para el desarrollo del proyecto, fuente autor 56
Figura 44. Sistema operativo de la Jetson iniciado, fuente autor 56
Figura 45. Visualización de la profundidad en la aplicación de la realsense, fuente autor 57
Figura 46. Diagrama del algoritmo para detección de objetos y profundidad, fuente autor. 58
Figura 47. Tarjeta Jetson con adaptador de audio, fuente autor. 59
Figura 48. Diagrama del algoritmo utilizado para el uso del API text to speech, fuente autor. 60
Figura 49. Estructura de la propuesta final, fuente autor. 61
Figura 50. Montaje Jetson y Realsense, fuente autor. 62
Figura 51. Montaje detección de teléfono, fuente autor. 63
Figura 52. Resultados de detección del teléfono, fuente autor. 63
Figura 53. Resultados de la detección de varios objetos, prueba uno, fuente autor. 64
Figura 54. Resultados de la detección varios objetos, prueba dos, fuente autor. 64
Figura 55. Resultados de la distancia y velocidad de un objeto inmóvil, fuente autor. 65
Figura 56. Resultados de la distancia y velocidad de objetos en movimiento, fuente autor. 65
Figura 57. Resultados de la detección de vehículos, prueba uno, fuente autor. 66
Figura 58. Resultados de la detección de vehículos, prueba dos, fuente autor. 67
Figura 59. Resultados de la detección de vehículos, prueba tres, fuente autor. 68
Figura 60. Resultados de la detección de vehículos, prueba cuatro, fuente autor. 68
Figura 61. Flujo de trabajo para el entrenamiento de los modelos, fuente autor. 69
Figura 62. Flujo general de la propuesta final, fuente autor 69
Figura 63. Resultados del mAP del entrenamiento del modelo ssd_mobilenet_v2_coco, fuente autor 70
Figura 64. Matriz de confusión para ssd_mobilenet_v1_coco, fuente autor 72
Figura 65. Métricas de desempeño del modelo ssd_mobilenet_v1_coco, fuente autor 73
Figura 66. Matriz de confusión para ssd_mobilenet_v2_coco, fuente autor 74
Figura 67. Métricas de desempeño del modelo ssd_mobilenet_v2_coco, fuente autor 75
Figura 68. Matriz de confusión para ssd_mobilenet_v2_coco entrenamiento con conjunto de datos hibrido, fuente autor 76
Figura 69. Métricas de desempeño del modelo ssd_mobilenet_v2_coco entrenamiento con conjunto de datos hibrido, fuente autor 77
Figura 70. Prototipo final, fuente autor 78

Figura 71. Cámara RealSense en el casco del motociclista, fuente autor 79
Figura 72. Motociclista usando el prototipo final, fuente autor.....79
Figura 73. Detecciones realizadas con el sistema en funcionamiento, fuente autor 80
Figura 74. Diagrama de bloques de los componentes de la placa base (Dev Board) [36]. 86
Figura 75. Diagrama de bloque de los componentes de SOM (Dev Board) [36]. 87
Figura 76. Arquitectura de la cámara Intel RealSense [37].....88

LISTA DE TABLAS

Tabla 1. Arquitectura de red completa para MobileNet [29]. 29
Tabla 2. Comparación de tiempos de respuesta obtenidos. 58
Tabla 3. Velocidades y presiones de los modelos entrenados. 69
Tabla 4. Comparación de sensores utilizados. 78
Tabla 5. Resultados pruebas de medición distancias con los métodos utilizados. 80
Tabla 6. Comparación de costos de los sensores y métodos utilizados. 81

RESUMEN

Este proyecto plantea varios métodos de detección de objetos como también su profundidad relativa, haciendo énfasis en el uso de sistemas embebidos. La detección de objetos se realiza mediante el uso de cámaras para posteriormente obtener una respuesta por parte del modelo de inteligencia artificial.

Se evalúa la funcionalidad de diferentes tecnologías con el fin de encontrar un método de detección de profundidad fácil de implementar y con gran exactitud.

1. INTRODUCCIÓN

El uso de motocicletas ha estado en crecimiento a lo largo de las últimas décadas, es así que para diciembre del 2018 el 57% del parque automotor colombiano lo conformaban las motos, siendo Bogotá la ciudad que encabeza la lista del mayor número registrado con un total de 480,404 [1]. Sin embargo, la visibilidad trasera que tiene un motociclista es inferior a la que tiene un coche especialmente cuando se conduce a altas velocidades o condiciones climáticas adversas [2], por consiguiente, obtener mejor vista trasera y proporcionarle asistencia al conductor durante su viaje ayudaría a prevenir y reducir los accidentes de vehículos de dos ruedas teniendo en cuenta que algunos de los siniestros viales ocurren por descuido y distracción de los pilotos.

Para el desarrollo de este proyecto se implementó un prototipo capaz de detectar vehículos que representen peligro potencial al motociclista principalmente cuando el piloto tenga la necesidad de hacer un giro o cambiar de carril, brindándole información como distancia y velocidad aproximada de estos. Para ello se utilizó la visión artificial que cuenta con inteligencia artificial la cual en los últimos años ha estado en auge principalmente por tres razones: aumento de la capacidad de cómputo, facilidad de acceso a grandes cantidades de información en mayor parte debido a la llegada de la era informática, y finalmente el desarrollo de algoritmos más potentes que representan disminución del tiempo de entrenamiento. El prototipo a realizar es móvil, y para este fin fue necesario ejecutar una red neuronal en un dispositivo portátil, como un smartphone, el problema de ello es que los celulares de precio asequible no ejecutan los modelos de forma apropiada, una alternativa para este problema es implementar el algoritmo en una tarjeta de desarrollo Coral Dev Board, la cual contiene un procesador *Edge TPU* que ayuda a realizar una rápida inferencia para modelos de aprendizaje profundo [3].

En el desarrollo de este trabajo se presentará una descripción de trabajos previos donde se mostrarán algunas de las formas utilizadas para mejorar la seguridad de los conductores de motocicleta, posteriormente se abordarán algunos conceptos que son necesarios para la comprensión del desarrollo del proyecto, finalmente se presentan algunos métodos que se usan para lograr detectar vehículos y su distancia aproximada al motociclista.

2. PROBLEMA

En los últimos años la población mundial ha aumentado a un ritmo acelerado. En 1999 llegó a 6 mil millones de personas y se predice que alcanzará los 9 mil millones para el año 2050 [1]. Con este rápido incremento llegan otros problemas como el desempleo y una mayor congestión en sistemas de transporte, para el 2017 Bogotá contaba con un estimado de 8 millones de habitantes [2] y aunque la administración del Distrito cada vez se preocupa más por movilizar a sus habitantes de una manera cómoda y segura, el sistema de transporte público no alcanza a proveer vehículos suficientes para transportar a toda la comunidad. En consecuencia, surge la necesidad de tener un sistema de transporte propio el cual sea de bajo costo, bajo consumo de combustible y con la capacidad de liberarse fácilmente de la congestión vehicular, reduciendo así los tiempos de recorrido de las personas. Es por ello que una de las alternativas es la adquisición de motocicletas [3], no obstante, pese a que presenta dichas ventajas es muy riesgoso conducir vehículos de dos ruedas, un informe consolidado del Observatorio Nacional de Seguridad Vial (ONSV) registró que, de las 7.158 muertes ocurridas en el 2016 por accidentes de tránsito, 3.759 (52 por ciento) fueron motociclistas lo cual indica que cada día murieron 10 en promedio [4].

Según Cesvi (centro de experimentación y seguridad vial), algunas de la principales causas de accidentalidad de motociclistas en Colombia se debe a: invasión de carril 34.7%, pérdida de control del vehículo por causa de deficiencias de la maniobrabilidad del conductor 23.8%, no respetar la prelación (cruce de pare) 14.2%, exceso de velocidad 10.7%, circular sin ocupar el carril 5.7%, no mantener la distancia de seguridad 4.9% y por tratarse de un vehículo que no cuenta con carrocería y que generalmente choca con vehículos de mayor tamaño, los accidentes resultan con lesiones graves [5].

Otra de las dificultades que los motociclistas presentan a la hora de conducir es la falta de accesorios tecnológicos que le permitan una mayor seguridad y que ayuden al conductor a evitar accidentes, como es el caso de algunos automóviles (Tesla motors) que cuentan con sistemas de control de velocidad adaptativo, sensores ultrasónicos que están analizando continuamente el entorno para hacer cambios de carril, cámaras que detectan peatones y señales de tránsito, todo esto para que sean autónomos y reducir los índices de accidentalidad [6]. No obstante, los motociclistas también poseen otros problemas como la falta de visión trasera la cual está parcialmente oculta por el cuerpo del motociclista o se crea un punto ciego por los lados del casco lo cual hace que algunas veces el conductor tenga que girar la cabeza para ver el tráfico que se encuentra por la parte trasera de la motocicleta. Además, los espejos retrovisores convencionales que poseen las motocicletas permiten una visibilidad limitada al cubrir un pequeño campo de visión, adicional a esto son inestables al tener una tendencia a moverse o agitarse a altas velocidades y fuertes vientos en contra [7].

A partir del planteamiento del problema se llega a la siguiente pregunta de investigación:

¿Cómo implementar un asistente de viaje para un conductor de motocicleta que le brinde información sobre peligros potenciales en los puntos ciegos existentes?

3. ANTECEDENTES

3.1 INTELIGENCIA ARTIFICIAL

Es la combinación de algoritmos con el propósito de crear máquinas “inteligentes”, idealmente es un agente racional capaz de tomar decisiones que aumenten sus probabilidades de éxito siendo capaz de tener las mismas capacidades que el ser humano. Actualmente la IA (Inteligencia artificial) está presente en los asistentes virtuales como SIRI, en los dispositivos de reconocimiento facial, aplicaciones móviles, videojuegos, entre otras; en general es usada con el propósito de facilitarle la vida las personas [8].

3.2 VISIÓN ARTIFICIAL

A partir de la inteligencia artificial surge la visión computacional o procesamiento de imágenes la cual utiliza captación y procesamiento de imágenes para lograr que el sistema de procesamiento entienda, pueda describir y lograr resultados [9].

Existen distintos propósitos por los cuales se procesan y manipulan imágenes uno de ellos es poder observar objetos que no son distinguibles por el ojo humano, esto puede suceder porque no había suficiente luz, razones climáticas, entre otras; en general la visión artificial es programar una computadora para que entienda una escena o características de una imagen [10]. En otros casos el objeto puede ser distinguible pero no se puede ver claramente para rescatar sus características, se aplican técnicas de procesamiento de imágenes para generar una imagen más visible y nítida [11].

3.2.1 ETAPAS EN UN PROCESO DE VISIÓN ARTIFICIAL

Partiendo de la escena el primer paso es adquirir una imagen digital a través de algún tipo de dispositivo que permita digitalizar las señales captadas del entorno, el siguiente paso es procesar la imagen captada, con el fin de mejorar sus características y que el resultado final tenga mayores posibilidades de éxito, luego se pasa al proceso de segmentación el cual se realiza para dividir la imagen en los diferentes objetos que la forman, este es un paso crítico del proceso debido a que una buena segmentación autónoma facilita la solución del problema, por el contrario un mal desarrollo de la misma conducirá al fallo. A la salida se obtiene una imagen con datos que contienen las fronteras de la región deseada o los puntos de ella misma. A continuación, se muestra en un diagrama de bloques las etapas de la visión artificial [12].

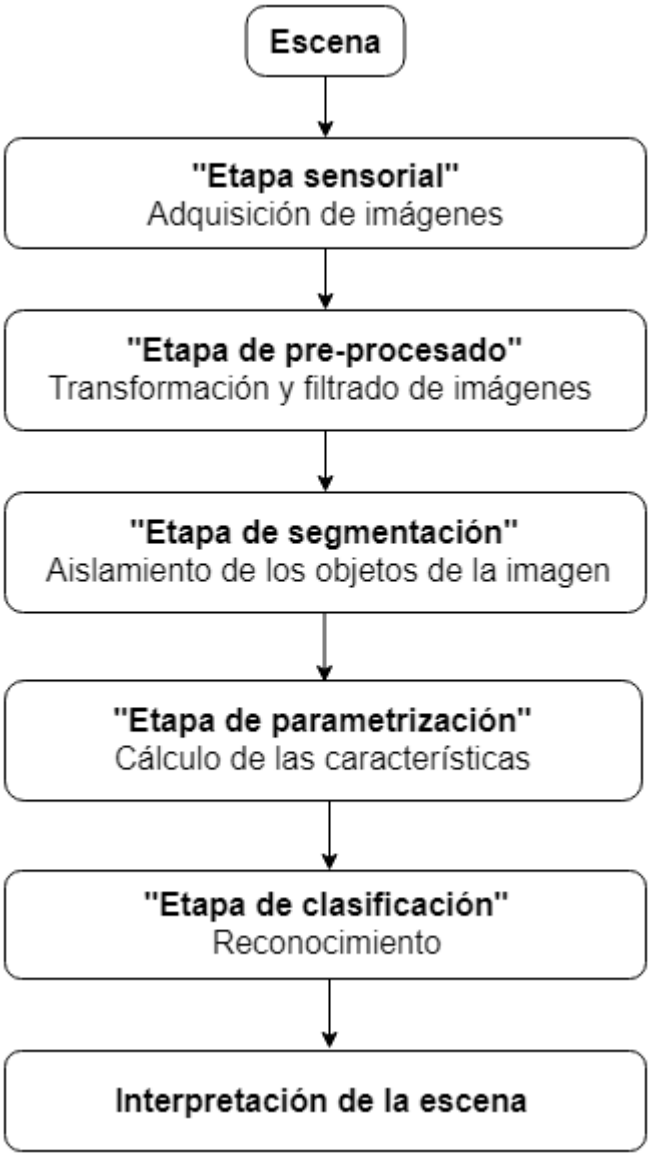

Figura 1. Etapas de un sistema de visión artificial [12].

3.3 TRABAJOS PREVIOS

En la actualidad existen muchos estudios sobre cómo prevenir y reducir los índices de accidentalidad no obstante existen muy poca innovación tecnológica en los vehículos de dos ruedas que disminuya los índices de accidentalidad, algunas de las investigaciones realizadas recientemente intentan brindar una mayor seguridad a los conductores de motocicleta como los que se mencionan a continuación.

3.3.1 HELMY

Consta de un sistema de dispositivos electrónicos que utiliza tecnologías de *blockchain* y *big data,* el cual busca proteger la vida y seguridad de los motociclistas, desarrollado por la Universidad de los Andes en conjunto con la empresa *Protraffic*, compañía colombiana de ingeniería y tecnología que trabaja en el área de seguridad vial. El prototipo cuenta con un sistema de llamada telefónica de emergencia en caso de un accidente, un sistema antirrobo el cual permite encender la motocicleta únicamente cuando el conductor se autentique mediante su teléfono celular y finalmente un sistema que impide el encendido de la moto si el motociclista no tiene puesto el casco de manera correcta [13].

3.3.2 SISTEMA DE SEGURIDAD PARA MOTOCICLISTA MEDIANTE EL USO DEL INTERNET DE LAS COSAS

En diversos países asiáticos las motocicletas son ampliamente utilizadas, no obstante los conductores de motocicletas no le dan la importancia necesaria al uso correcto del casco, es por ello que el documento presenta una solución al riesgo de conducir sin protecciones craneales. Cuenta con un módulo de procesamiento el cual va anclado a la motocicleta y a través de sensores capacitivos detecta si el casco se encuentra a una distancia prudente para habilitar el encendido o por el contrario apagar la motocicleta si el casco se aleja de ella, también implementan un sistema de bocinas y micrófono dentro del casco para que los usuarios puedan contestar llamadas sin tener que usar el teléfono directamente mientras conducen [14].

3.3.3 IMPLEMENTACIÓN Y ANÁLISIS DE CASCO INTELIGENTE

En el documento [15] se presenta un sistema que proporciona seguridad al conductor a través de la implementación de un sistema inteligente compuesto por dos módulos uno en el casco del motociclista y otro en la moto conectados entre sí por radiofrecuencia capaces de detectar si el piloto lleva el casco puesto, además cuenta con un sensor piezoeléctrico capaz de detectar el exceso de velocidad, un acelerómetro capaz de notificar vía GSM a un contacto personal y las autoridades en cuestión si ha ocurrido algún accidente con el automotor y finalmente un sensor de alcohol integrado en el casco del motociclista habilita el encendido de la motocicleta; si el sensor MQ-6 detecta índices de alcohol en el aire exhalado por el conductor no permite que se encienda el motor del vehículo evitando así accidentes por imprudencia de las personas y haciendo las vías más seguras [15].

3.3.4 SISTEMA DE TRANSPORTE INTELIGENTE PARA PREVENCIÓN Y DETECCIÓN DE ACCIDENTES

Teniendo en cuenta que en países en vía de desarrollo se han incrementado en gran medida el uso de vehículos de dos ruedas como una alternativa fácil y económica;

también crecen rápidamente los índices de accidentalidad de estos, es por ello que desarrollan un sistema que verifica los índices de alcohol del piloto y si tiene el casco puesto antes de encender la motocicleta, para la detección del casco se usa un interruptor que se activa al colocarse el casco, usan también un sensor de alcohol MQ-3 para analizar continuamente la respiración del conductor y detectar si existen índices de alcohol y en la motocicleta un dispositivo de procesamiento (microcontrolador) capaz de monitorear las variables captadas de los sensores y procesarlas; además de ello el motor del vehículo está anclado a la activación de un relé el cual se activa solo si el usuario no presenta índices de alcohol y tiene el casco puesto. Se proporciona una solución para el problema de la alta accidentalidad de las motocicletas, detectando el accidente tan pronto ocurra a través de un acelerómetro e informando a las autoridades encargadas [16].

3.3.5 SISTEMA DE IDENTIFICACIÓN DE ACCIDENTES Y NAVEGACIÓN

Debido al incremento de accidentalidad en motociclistas en la India y los bajos índices de uso del casco como herramienta protectora, se desarrolla un sistema capaz de detectar accidentes y ayudar al conductor en su recorrido, este prototipo se divide en cuatro partes las cuales se pueden observar en la figura 2. Para la primera parte la cual es identificación de accidentes y módulo de alerta usan un Arduino, un sensor de presión y por último un módulo GSM con el fin de que cuando el sensor exceda un límite de presión (posible accidente) se da la alerta respectiva a las autoridades competentes mediante el uso combinado del módulo GSM y el Arduino. La segunda parte es el sistema de navegación en el cual se idearon una forma de rastrear vía *GPS* el automotor con el fin de que cuando ocurra un accidente inmediatamente se pueda localizar la motocicleta y enviar vía GSM a los organismos de socorro un mensaje con la ubicación donde ocurrió el suceso aumentando la rapidez con que se ayuda al accidentado y disminuyendo las posibles víctimas mortales. En la tercera parte la cual se titula servicio de llamadas se implementa un reconocimiento de voz que está conectado al teléfono móvil y en caso de que el piloto reciba una llamada puede contestar con solo decir la palabra clave *"ANSWER*" sin necesidad de manipular el celular directamente, esto con el fin de evitar que el conductor use el teléfono mientras conduce y disminuir la posibilidad de distracción en la vía, la última parte cuenta de un sistema de paneles solares como una alternativa para alimentar el sistema [17].

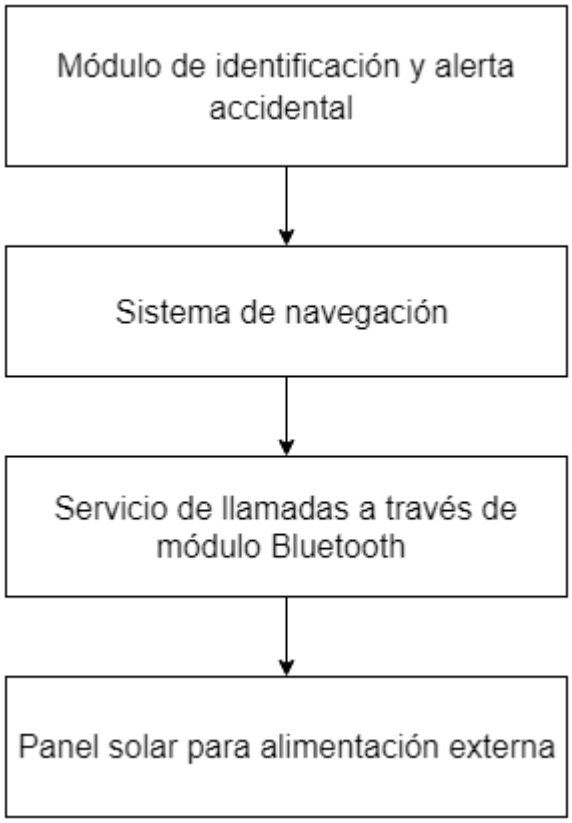

Figura 2. Diagrama de bloques de las 4 partes en que se divide el sistema propuesto [17].

.

3.3.6 PROCESAMIENTO DE VIDEO PARA DETECCIÓN DE AUTOMÓVILES DE LA PARTE TRASERA DE UNA MOTOCICLETA Y PREVENCIÓN DE ACCIDENTES.

En este documento [18] se presenta una metodología para monitorear el tráfico detrás del conductor de una motocicleta, también contiene un sistema para alertar al conductor en caso de peligro. A través de procesamiento de video se ofrecen instrucciones al conductor en tiempo real sobre los vehículos que se aproximan y para ello plantean un algoritmo que se ejecuta en un procesador de video montado dentro del casco.
Para lograr obtener fotografías con las cuales fuera fácil trabajar se ubicó la cámara en un lugar estratégico, en un comienzo planeaban colocar la cámara cerca a la placa del vehículo pero no funcionó correctamente debido a que no tiene la altura necesaria para que se pueda monitorear de forma adecuada el tráfico, luego de ello intentaron ubicar la cámara en la parte trasera del casco, no obstante en esta ubicación hubo otros problemas como el copiloto, que obstaculizaba la visión trasera, es por ello que finalmente se decidió colocarla centrada sobre el casco como se observa en la figura 3 [18].

Figura 3. Ubicación final de la cámara (sobre el casco del motociclista) [18].

La cámara captura el video en tiempo real del escenario trasero de la motocicleta y lo envía al procesador donde un algoritmo (figura 5), lo procesa para posteriormente indicarle al conductor mediante una alerta sonora la proximidad de un vehículo y si el coche está demasiado cerca mediante una voz pregrabada se advierte al piloto. A continuación, se presenta en diagrama de bloques del algoritmo de procesamiento de imágenes utilizado [18].

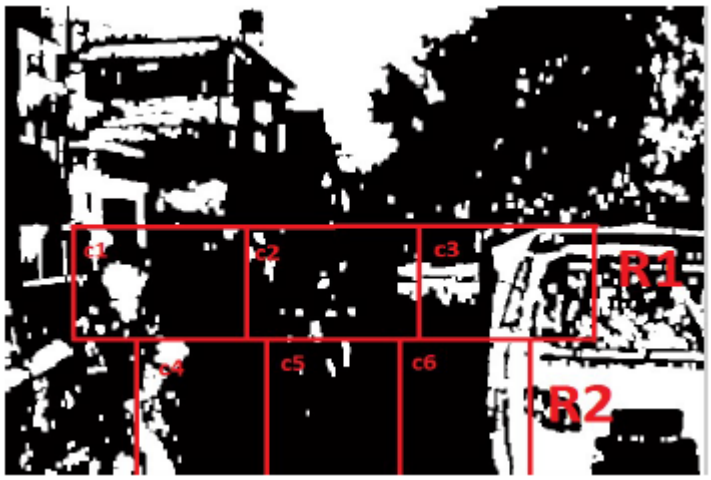

Figura 4. Imagen procesada dividida en regiones [18].

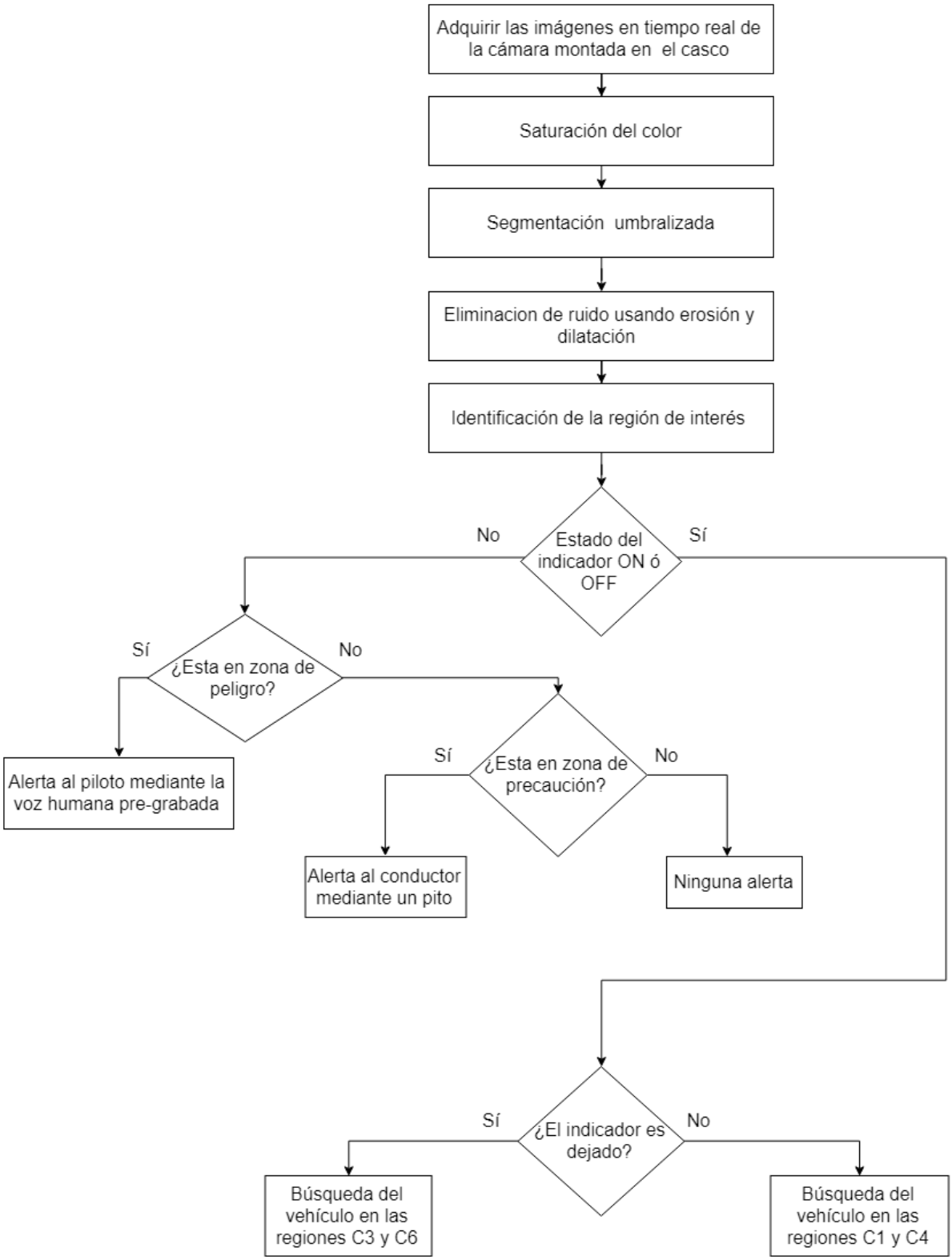

Figura 5. Diagrama de bloques del algoritmo usado [18].

3.3.7 SISTEMA DE VISIÓN PARA CASCOS DE MOTOCICLETA

Se presenta el desarrollo de un prototipo de un sistema que facilite la visión al motociclista y funcione como espejo retrovisor alterno capaz de mostrar un conjunto de imágenes provenientes de la parte posterior de una motocicleta, evitando que el conductor tenga que girar la cabeza para ver el estado de tráfico detrás de él, ya que a altas velocidades esto sería una distracción que puede provocar accidentes; el proyecto mencionado en el documento [19] tiene como propósito reducir los accidentes de tráfico producidos por los puntos ciegos que tiene este automotor. Se crea un visor óptico teniendo en cuenta la capacidad visual del ser humano la cual en promedio puede enfocar adecuadamente a una distancia de 10 cm, el resultado de este prototipo se puede observar en la figura 6 [19].

Figura 6. Prototipo de un casco para motocicletas con un sistema de visión que permite ver 360 grados al mismo tiempo [19].

4. JUSTIFICACIÓN

En Colombia el crecimiento acelerado de motociclistas implica un aumento en la accidentalidad de los mismos, en América latina y el Caribe los siniestros de tránsito en motocicletas representan la principal causa de muerte entre niños de 5 a 14 años y la segunda entre la población de 15 a 44 años [20]; es por ello que diversas organizaciones han decidido crear planes de acción para reducir los índices de lesiones y muertes en usuarios de las motocicletas; la implementación del uso del casco, de medidas de visibilidad, control de comportamientos de riesgo y la aplicación de leyes donde se restringe el consumo de alcohol son algunas de las principales prácticas que se deben aplicar para la reducción de los índices de accidentalidad [21], el casco por su parte es una herramienta protectora para el cráneo capaz de disminuir el riesgo y la gravedad de los traumatismos en alrededor del 72%, disminuyendo hasta en 39% las probabilidades de muerte [20].

Según una investigación realizada por el instituto de gestión y transporte (Virginia EE UU) cerca del 80 % de las colisiones y 65 % de las casi-colisiones, contienen de alguna forma la falta de atención en los 3 segundos previos al evento [22]; es por ello que con la implementación del sistema de ayuda que contará con una cámara que mediante un sistema de visión artificial pueda detectar los peligros que puedan convertirse en posibles colisiones y avisarle al conductor de la motocicleta hasta 2.2 segundos antes del suceso mediante alarmas visuales y sonoras; según un estudio realizado en el 2009 por la empresa AXA determinó que en un tiempo de 1.5 Segundos de alerta previa puede prevenir hasta el 90% de las colisiones traseras y con un tiempo de 2.0 se pueden prevenir casi todos los accidentes [23].

Con la implementación de un sistema capaz de detectar los posibles vehículos acercándose a la motocicleta cuando el piloto realice maniobras para adelantar o girar se puede aumentar la visión del motociclista e informarlo de los posibles riesgos de choque, disminuyendo así los índices de distracción del conductor y a su vez evitando puntos muertos en la visión; para el desarrollo del sistema se tuvo en cuenta la distancia y velocidad aproximada con la cual se acercan los vehículos a la motocicleta, esto con el fin de activar las posibles alertas de peligro permitiendo al conductor reaccionar oportunamente disminuyendo las probabilidades de un accidente de tránsito.

5. OBJETIVOS

5.1 OBJETIVO GENERAL

Diseñar e implementar un sistema prototipo de reconocimiento de objetos que permita cubrir los puntos ciegos existentes, para ser usado como asistente de viaje por el conductor de una motocicleta.

5.2 OBJETIVOS ESPECÍFICOS

- Investigar acerca del tipo de sensores y las características técnicas de los mismos que ayuden a determinar la velocidad y distancia aproximada de automóviles.
- Seleccionar una técnica de procesamiento de imágenes capaz de procesar los datos adquiridos que se ajuste al tiempo de respuesta del sistema a desarrollarse.
- Desarrollar un sistema de asistencia que le informe al motociclista acerca de los riesgos que han sido identificados por del sistema.
- Validar el sistema en un entorno controlado, teniendo en cuenta los factores que podrían afectar el funcionamiento del mismo cuando esté a campo abierto.

6. MARCO TEÓRICO

Para el desarrollo de este proyecto fue necesario conceptualizar algunos términos que brindarán una visión más amplia del tema y ayudarán a entender el propósito de este trabajo.

6.1 SOFTWARE

6.1.1 APRENDIZAJE DE MÁQUINA

Es una disciplina derivada de la Inteligencia artificial y cuyo objetivo es predecir comportamientos futuros a través del análisis de millones de datos mejorando su aprendizaje a lo largo del tiempo y sin intervención humana [24].

6.1.2 REDES NEURONALES ARTIFICIALES

El funcionamiento de las redes neuronales artificiales está inspirado en las redes neuronales biológicas (figura 7) cuyo funcionamiento se podría resumir en lo siguiente: las dendritas y el cuerpo celular reciben impulsos químicos y eléctricos provenientes de neuronas vecinas excitándose y generando una respuesta que es transportada por el axón a los terminales axónicos que a su vez se conectan con miles de neuronas más [25].

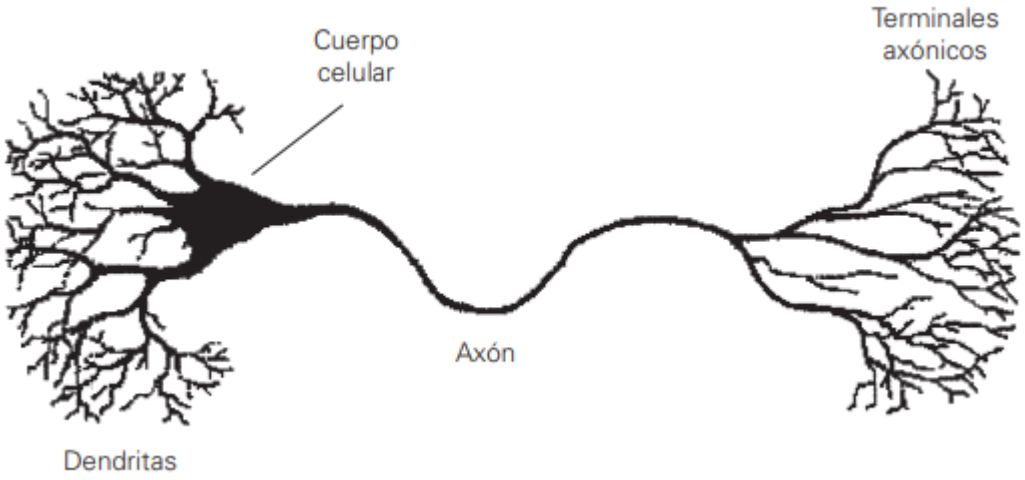

Figura 7. Estructura general de una neurona biológica [25].

Las ANN (*Artificial Neural Networks*) simulan las neuronas biológicas del cerebro humano las cuales procesan información y son capaces de aprender a través de la experiencia y abstraen características relevantes de nuevos datos [26], las neuronas individuales se conectan entre si mediante enlaces que tienen asociado un peso, la suma ponderada de los enlaces “señales” suministra la entrada total o neta de la neurona y mediante la

aplicación de una función matemática sobre la entrada se estima el valor de la salida [25] que será enviada a otras neuronas que están organizadas en niveles o capas formando así una red neuronal artificial. En la figura 8 se puede observar la representación de una red neuronal artificial.

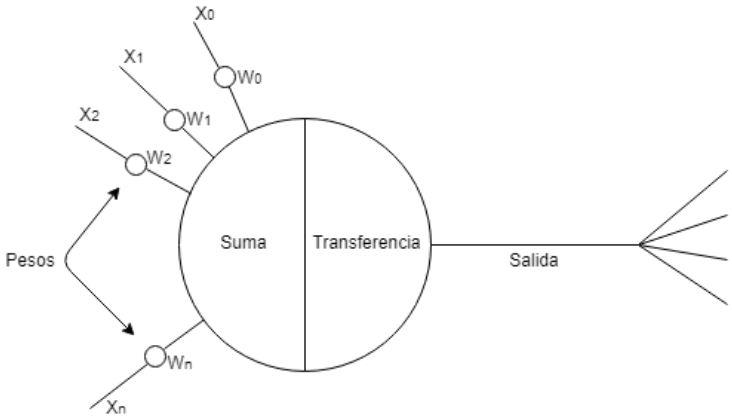

Figura 8. Red neuronal artificial [26].

6.1.3 *CONVOLUTIONAL NEURAL NETWORK (CNN)*

Es una red neuronal artificial multicapa jerárquica que extrae características de sus capas ocultas, su nombre se debe a su arquitectura y funcionamiento ya que se conforman de 5 a 7 niveles las cuales se organizan en capas convolucionales y de submuestreo [27]. Por su estructura es ampliamente utilizada en el procesamiento de imágenes detectando características u objetos que el desarrollador considere pertinentes para lograr su objetivo.

6.1.4 *TENSORFLOW LITE*

Es un conjunto de herramientas que ayudan a los desarrolladores a ejecutar modelos TensorFlow en dispositivos móviles, integrados y de IoT, permitiendo la inferencia del aprendizaje automático en el dispositivo con baja latencia y un pequeño tamaño binario [28].

6.1.5 *MOBILNETS*

Son modelos eficientes para aplicaciones de visión móviles e integradas, se basan en una arquitectura optimizada que utiliza convoluciones separables en profundidad para construir redes neuronales profundas y livianas. La arquitectura completa de MobilNet se puede visualizar en la tabla 1 [29].

Tipo / Paso	Forma de filtro	Tamaño de entrada
Conv / s2	3 x 3 x 3 x 32	224 x 224 x 3
Conv dw / s1	3 x 3 x 32 dw	112 x 112 x 32
Conv / s1	1 x 1 x 32 x 64	112 x 112 x 32
Conv dw / s2	3 x 3 x 64 dw	112 x 112 x 64
Conv / s1	1 x 1 x 64 x 128	56 x 56 x 64
Conv dw / s1	3 x 3 x 128 dw	56 x 56 x 128
Conv / s1	1 x 1 x 128 x 128	56 x 56 x 128
Conv dw / s2	3 x 3 x 128 dw	56 x 56 x 128
Conv / s1	1 x 1 x 128 x 256	28 x 28 x 128
Conv dw / s1	3 x 3 x 256 dw	28 x 28 x 256
Conv / s1	1 x 1 x 256 x 256	28 x 28 x 256
Conv dw / s2	3 x 3 x 256 dw	28 x 28 x 256
Conv / s1	1 x 1 x 256 x 512	14 x 14 x 256
5 x Conv dw / s1 Conv / s1	3 x 3 x 512 dw 1 x 1 x 512 x 512	14 x 14 x 512 14 x 14 x 512
Conv dw / s2	3 x 3 x 512 dw	14 x 14 x 512
Conv / s1	1 x 1 x 512 x 1024	7 x 7 x 512
Conv dw / s2	3 x 3 x 1024 dw	7 x 7 x 1024
Conv / s1	1 x 1 x 1024 x 1024	7 x 7 x 1024
Avg Pool / s1	Pool 7 x 7	7 x 7 x 1024
FC / s1	1024x1000	1 x 1 x 1024
Softmax / s1	Clasificador	1 x 1 x 1000

Tabla 1. Arquitectura de red completa para MobileNet [29].

Cabe resaltar que la normalización por lotes (BN) y ReLU se aplican después de cada convolución (figura 9).

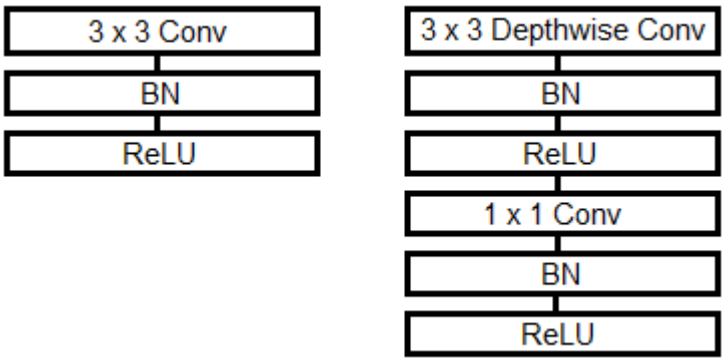

Figura 9 Convolución estándar (izquierda), convolución separable en profundidad (derecha) con BN y ReLU [29].

6.1.6 *TEXT TO SPEECH*

Según la documentación oficial de Google es una API [30] que convierte texto en voz que suena natural basada en tecnologías de inteligencia artificial, se pueden elegir más de 220 tipos de voces en 40 idiomas y variedades lingüísticas. Cuenta con la opción de entrenar un modelo de síntesis de voz personalizado con grabaciones propias de audio con el fin de crear una voz única, cuenta con personalizaciones de tono, ajuste de velocidad de locución, volumen y la tasa de muestreo en hercios.
El proceso de convertir entradas de texto en datos de audio se llama síntesis y el resultado de una síntesis es llamada voz sintética. [31]

6.2 HARDWARE

6.2.1 DEV BOARD

Para el desarrollo del proyecto se requiere correr modelos de aprendizaje de máquina, por lo cual se probó la tarjeta Dev Board (figura 10) la cual fue desarrollada por Google y contiene un coprocesador *Edge TPU* que hace posible realizar inferencias de aprendizaje de máquina de alto rendimiento con bajo costo de energía [32]. En los anexos uno y dos se ilustran los componentes principales de la tarjeta.

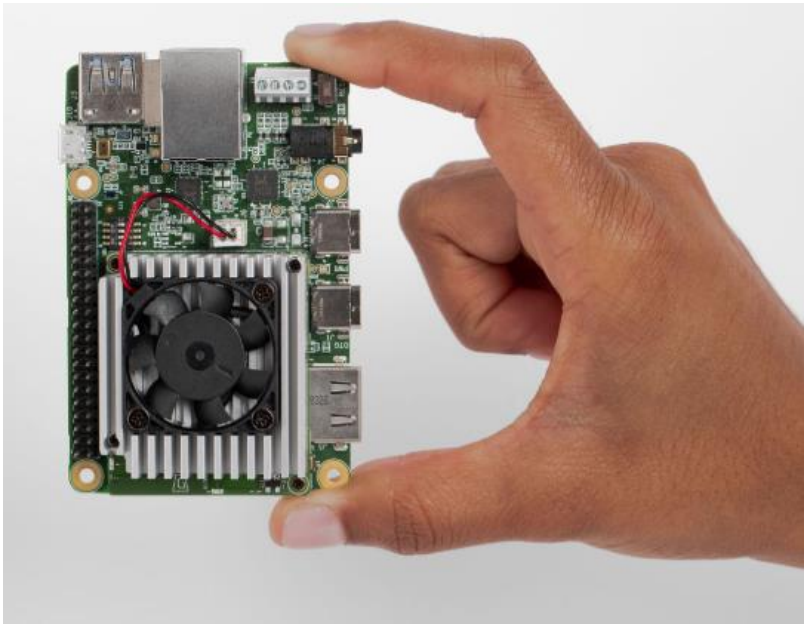

Figura 10. Coral Dev Board [32].

6.2.2 T.P.U.

Es un hardware especializado para el procesamiento de tensores (matrices), diseñado por Google para acelerar el desarrollo de tareas de aprendizaje profundo utilizando TensorFlow el cual es un framework de programación; la figura 11 compara la unidad más pequeña de cómputo (compute primitive) en una CPU, GPU y TPU [33].

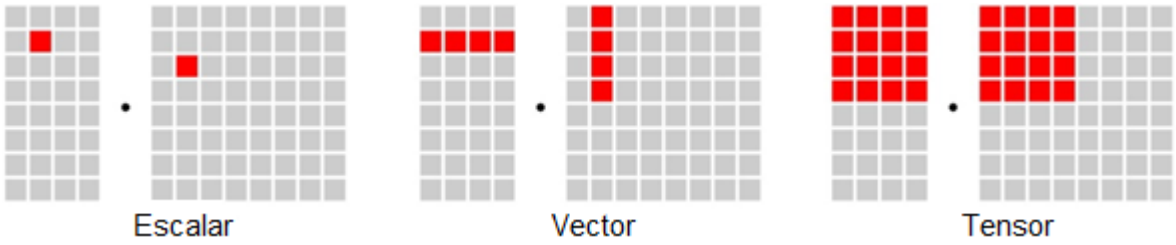

Figura 11. Compute primitive [33].

6.2.3 COMPRENSIÓN DE LA ESTRUCTURA Y FUNCIONAMIENTO DE LA TARJETA CORAL

La principal característica es su unidad de procesamiento tensorial TPU la cual permite mejorar el rendimiento de los modelos de aprendizaje de máquina, el sistema operativo que ejecuta la tarjeta es una distribución de Linux llamada Mendel la cual fue diseñada especialmente para manejar el hardware de la misma. Edge TPU es capaz de ejecutar redes neuronales de alineación profunda como redes neuronales convolucionales (CNN) y solo admite modelos TensorFlow Lite que estén totalmente cuantificados en 8 bits y luego compilados específicamente para Edge TPU, cuando se quiera crear un modelo TensorFlow que aproveche al máximo la tarjeta en tiempo de ejecución se deben tener en consideración las siguientes cosas, los parámetros del tensor se cuantifican (números de punto fijo de 8 bits; int8 o uint8), los tamaños de tensor son constantes en tiempo de compilación (sin tamaños dinámicos), los parámetros del modelo (como los tensores de polarización) son constantes en tiempo de compilación, los tensores son de 1, 2 o 3 dimensiones si un tensor tiene más de 3 dimensiones solo las tres dimensiones más internas pueden tener tamaño mayor que uno [34].

6.2.4 JETSON NANO

Es una computadora para aplicaciones integradas de inteligencia artificial e IoT, incluye un entorno Linux con compatibilidad de aplicaciones nativas de la nube y cuenta con una CPU ARM Cortex-A57 MpCore de cuatro núcleos y una GPU Nvidia Maxwell con 128 núcleos CUDA [35].

Figura 12. Jetson Nano. Figura [35]

6.2.5 REALSENSE

Este tipo de tecnologías utiliza tres escáneres ópticos separados: Una cámara óptica, una cámara infrarroja y un proyector laser para leer el entorno. Al trabajar en conjunto los tres lectores gráficos pueden detectar profundidad, movimiento humano y escanear objetos en 3D. [36]

Figura 13. Cámara Intel RealSense [36].

La cámara de profundidad Intel RealSense utiliza visión estéreo para calcular profundidad. La implementación de la visión estéreo consta de un capturador de imágenes izquierdo, un capturador de imágenes derecho y un proyector de infrarrojos opcional. El proyector de infrarrojos proyecta un patrón infrarrojo (IR) estático no visible para mejorar la precisión de la profundidad en escenas con poca textura. Los lectores de imágenes izquierdo y

derecho capturan la escena y envían los datos del generador de imágenes al procesador de imágenes de profundidad, que calcula los valores de profundidad para cada píxel de la imagen mediante la correlación de puntos en la imagen de la izquierda a la imagen de la derecha y mediante el cambio entre un punto en la imagen de la izquierda y la Imagen de la derecha. Los valores de los píxeles de profundidad se procesan para generar un cuadro de profundidad. Los fotogramas de profundidad posteriores crean una secuencia de video en profundidad [37].

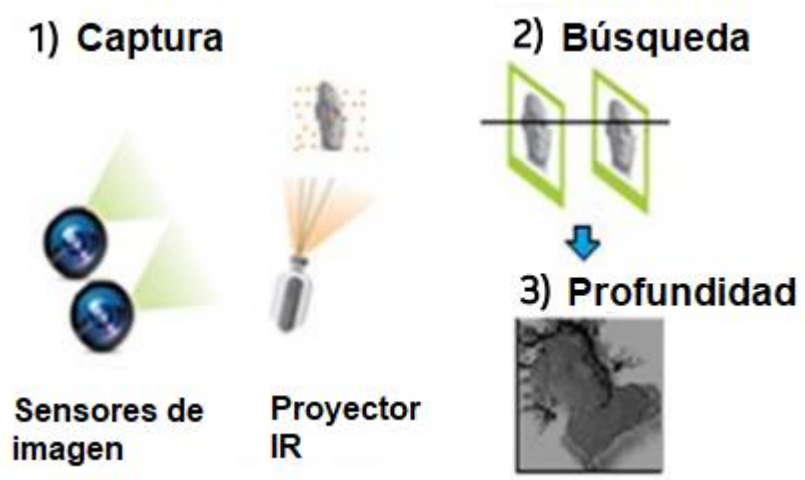

Figura 14. Pasos detección de la profundidad [37].

El sistema de cámaras tiene dos componentes principales, el procesador *Vision D4* y el módulo *Depth*. El procesador *Vision D4* está en la placa base del procesador o en una placa discreta con conexión USB2.0/USB 3.1. El módulo *Depth* incorpora las imágenes izquierda y derecha para visión estéreo con el proyector *IR* opcional y el sensor de color RGB. Los datos del sensor de color RGB se envían al procesador de visión D4 a través del procesador de señal de imagen de color *(ISP)* en la placa madre del procesador, en el anexo tres se presenta el diagrama de la arquitectura mencionada [37].

6.2.6 TRIANGULACIÓN MEDIANTE VISIÓN ESTÉREO

La triangulación es el proceso mediante el cual se usa la trigonometría básica para determinar la distancia de un punto teniendo en cuenta los ángulos desde puntos conocidos al extremo del objeto. A continuación, se explicará paso a paso este proceso.
Para este procedimiento se debe tener visión estero, lo que se resume en dos cámaras de características similares con ángulos de visión similares, luego se ubican las cámaras de forma paralela en la cual se conoce la distancia que existe entre cada una, en este caso la distancia es de 9.7 cm, esto se puede ver representado en la figura 15 donde B será igual a la distancia de separación, X el punto al que queremos medir la profundidad o distancia representada con d, cL la cámara izquierda, cR la cámara derecha. Como se puede

observar entre las cámaras y el punto X se crea un triángulo con el que se puede aplicar trigonometría para calcular d ya que se tiene el valor del lado B y se puede calcular los ángulos alfa y beta.

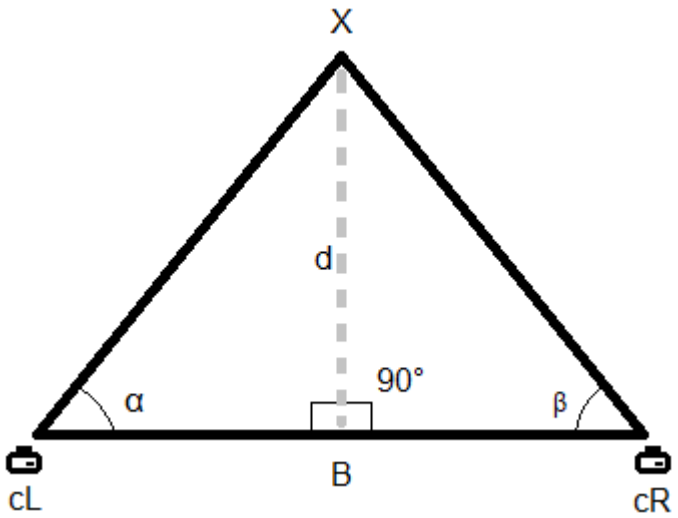

Figura 15. Representación del triángulo formado por las cámaras y el punto X, fuente autor

Para encontrar la distancia (d) se tiene que recordar que:

$$\tan = \frac{cateto\ opuesto}{cateto\ adyacente} \quad (1)$$

Por lo tanto, el triángulo que se forma se puede partir a la mitad lo que nos da como resultado dos triángulos rectángulos con los cuales se puede calcular las incógnitas, obteniendo:

$$\tan(\alpha) = \frac{d}{B/2} \quad (2)$$

$$\tan(\beta) = \frac{d}{B/2} \quad (3)$$

Como B es una variable conocida, entonces:

$$B/2 = \frac{d}{\tan(\alpha)} \quad (4)$$

$$\mathrm{B}/2 = \frac{d}{\tan(\beta)} \tag{5}$$

Donde B/2 es la mitad del triángulo acutángulo, por lo tanto:

$$\mathrm{B} = \frac{d}{\tan(\alpha)} + \frac{d}{\tan(\beta)} \tag{6}$$

Despejando el valor de la distancia (d) se obtiene:

$$\mathrm{d} = \frac{B}{\cot(\alpha) + \cot(\beta)} \tag{7}$$

Como se puede observar en la ecuación 7 la distancia está representada en función de los ángulos alfa y beta, para el cálculo de los ángulos se realiza el siguiente procedimiento:
Teniendo en cuenta que la resolución de imagen será de 640px por 480px se puede hacer un estimado de los ángulos considerando los grados de visibilidad de la cámara en este caso teta tomará el valor de 60 grados.

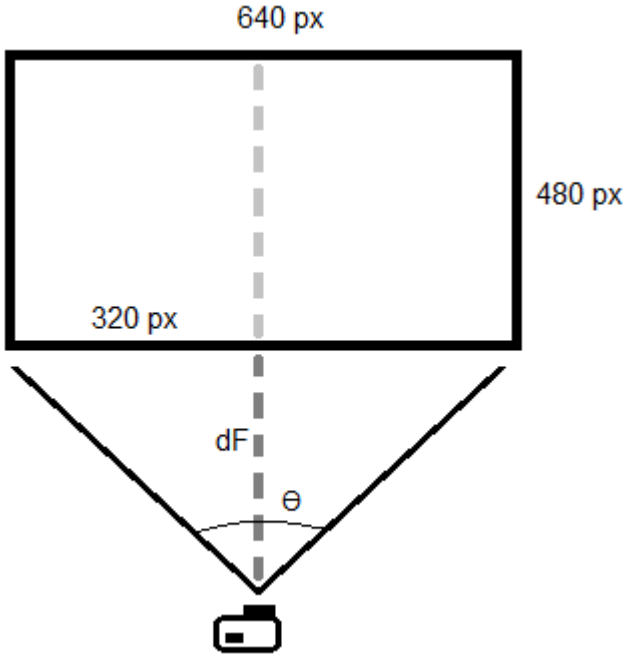

Figura 16. Representación del ángulo de visión de la cámara, fuente autor

Los ángulos alfa y beta que se necesitan para dar el estimado de profundad se pueden calcular recurriendo nuevamente a la fórmula de la tangente, cabe resaltar que en el caso que el punto X se encuentre en la mitad del cuadro (imagen) el ángulo tomara el valor de teta medio, no obstante, en la mayoría de los casos no ocurre de ese modo es por ello que partiendo de la figura 16 de donde a partir de los triángulos rectángulos podemos abstraer la siguiente información para ángulos diferentes al punto medio:

$$\tan\left(\frac{\Theta}{2}\right) = \frac{320}{dF} \tag{8}$$

Donde teta son los grados de visión de la cámara y dF la distancia al cuadro en pixeles, por lo tanto, despejando la incógnita se obtiene:

$$\mathrm{dF} = \frac{320}{\tan\left(\frac{\Theta}{2}\right)} \tag{9}$$

Teniendo el valor de la distancia al cuadro (dF) se puede reemplazar el 320 por el número de pixeles que el punto se separa de la mitad del cuadro, dicho número se puede calcular a partir del modelo de detección de objetos ya que retorna las coordenadas del objeto detectado en la imagen, la variación de pixeles se puede ver representada en la figura 17.

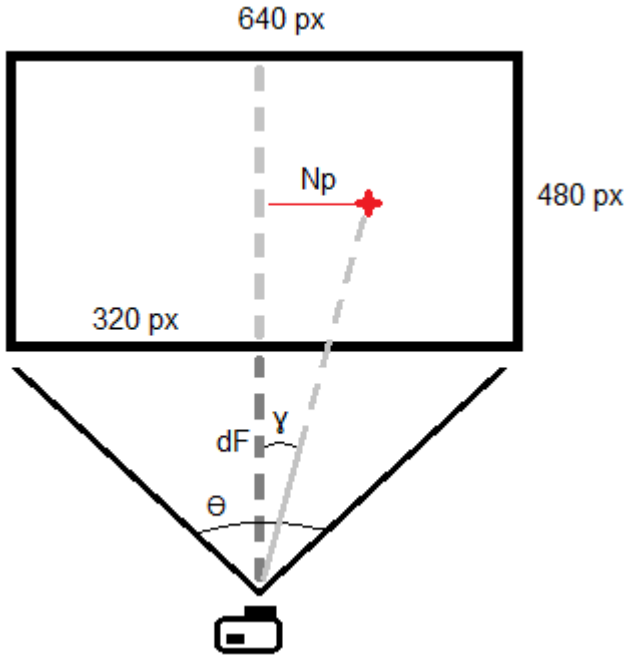

Figura 17. Representación de un ángulo diferente de teta medios, fuente autor.

Reemplazando en la ecuación 9 el valor del número de pixeles que el objeto X se aleja del punto medio se obtiene:

$$\mathrm{dF} = \frac{Np}{\tan(\gamma)} \tag{10}$$

Por lo tanto, para obtener el ángulo gamma el cual es la incógnita se realiza lo siguiente:

$$\gamma = \tan^{-1}\left(\frac{Np}{dF}\right) \tag{11}$$

Obteniendo los ángulos faltantes podemos reemplazar todas las incógnitas de la ecuación 7 y así obtendremos la distancia que existe de la cámara al objeto deseado.

Para que este método funcione correctamente es recomendable que las dos cámaras tengan las mismas características de diseño, además en el montaje se evidencio que el procedimiento tenía un índice de error elevado debido a que si una de las dos cámaras se desalinea o se mueve deja de funcionar apropiadamente.

7. DISEÑO Y EJECUCIÓN DEL PROYECTO

A continuación, se presenta cada uno de los procedimientos necesarios para el desarrollo del proyecto y el cumplimiento del objetivo principal.

Figura 18. Descripción general del desarrollo del diseño y la ejecución del proyecto, fuente autor.

7.1 ALGORITMO DE RECONOCIMIENTO DE OBJETOS

Para el algoritmo de reconocimiento de objetos se realizan pruebas sobre dos tarjetas de desarrollo: Coral Dev board y Jetson nano.

7.1.1 GENERACIÓN DEL MODELO DE INFERENCIA DE OBJETOS QUE SE EJECUTARÁ

En este ítem existen dos opciones una es diseñar la arquitectura de un modelo desde cero teniendo en cuenta una estructura que sea compatible con la tarjeta de desarrollo y siguiendo un flujo de trabajo adecuado para la ejecución del modelo en la tarjeta a probar, (en la figura 19 se ve representado el flujo de trabajo para la tarjeta Dev Board), no obstante esto podría traer problemas cuando se despliegue el algoritmo, por este motivo se opta por usar un modelo preentrenado de TensorFlow compatible con la placa y se pueda ejecutar con TensorFlow lite, según la documentación oficial de coral los modelos que se sirven para dicho propósito serían dos MobileNet SDD v1 y MobileNet SDD v2, teniendo en consideración que las dos redes tienen una velocidad de inferencia teórica de 29 ms se elige la red MobileNet SDD v1 para pruebas generales, no obstante también se prueba la red MobileNet SDD v2.

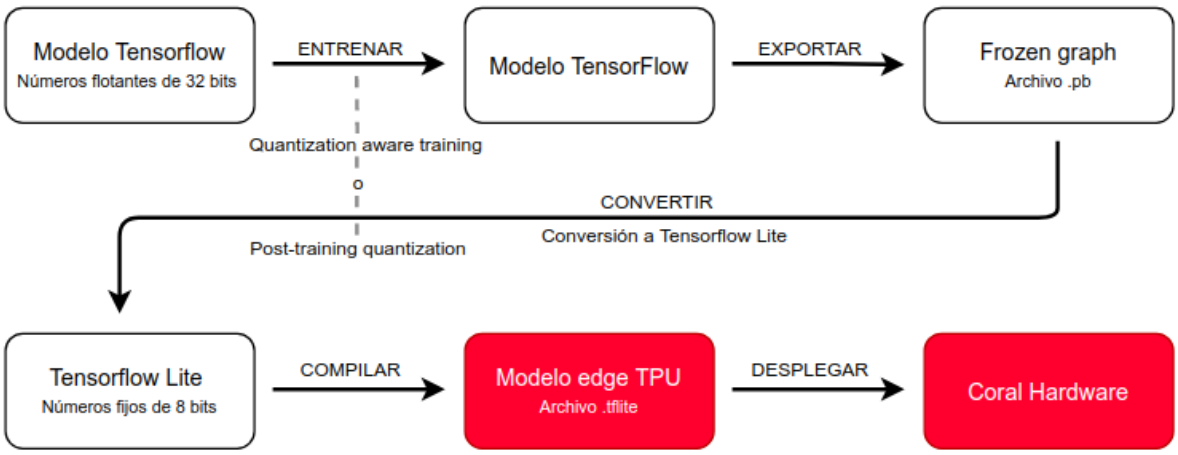

Figura 19. Flujo de trabajo para crear un modelo para Edge TPU [34].

Al reentrenar un modelo existente de TensorFlow se debe cumplir un prerrequisito principal el cual es tener un conjunto de datos de entrenamiento y otro de prueba en formato record y en una proporción de cantidad de muestras de 70% - 30% o 80% - 20% respectivamente; para lograr esto a lo largo de una semana se hacen grabaciones de video en las vías de la ciudad de Bogotá y alrededores, desde una motocicleta Yamaha Fazer 2.0 con ayuda de una cámara GoPro, la cámara se ubica en la parte superior del casco (figura 20) para evitar obstaculizar la visión de la misma cuando el motociclista lleve copiloto, cabe resaltar que el lente de la cámara está apuntando hacia atrás.

Los videos se grabaron con diferentes condiciones climáticas, de luz y en distintos días, lo que garantiza que el conjunto de datos disponga de un amplio número de perspectivas. Al analizar los videos se concluye que los objetos que representan peligro relevante para el motociclista son los demás vehículos; por lo tanto, las clases que reconocerá el sistema se dividen en cinco: bicicletas, buses, camiones, carros y motocicletas.

Figura 20. Cámara GoPro ubicada en la parte superior del casco, fuente autor

Al elegir las clases a detectar se etiqueta el conjunto de datos que se usará para entrenar el modelo para ello se seleccionan manualmente los mejores cuadros de cada video (figura 21) y seguido a esto se realiza el proceso manual de etiquetado de cada imagen usando LabelImg la cual es una herramienta de anotación gráfica de imágenes; cómo se puede observar en la figura 22 el usuario selecciona la ubicación del objeto a detectar y luego se le asigna un nombre de acuerdo a su clase para finalmente guardarla en un archivo XML donde muestra el nombre de la imagen, su ubicación, el tamaño de la misma y las coordenadas (xmin, ymin, xmax, ymax) de cada una de las selecciones que se han hecho de los objetos, deben existir la misma cantidad de archivos XML que de imágenes.

Figura 21. Ejemplo de cuadro tomado por la cámara, fuente autor

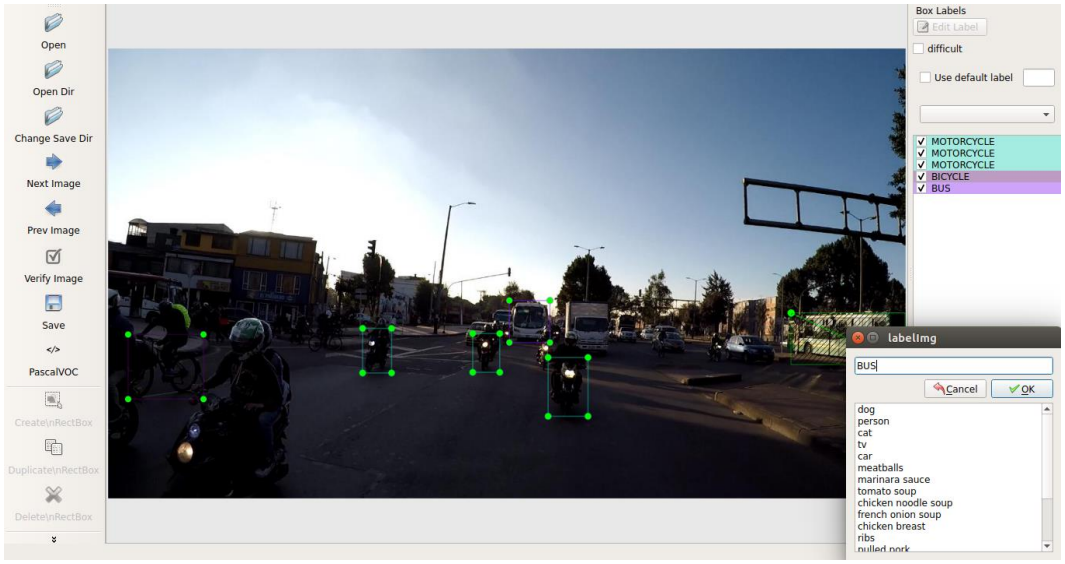

Figura 22. Uso de la herramienta LabelImg, fuente autor

Teniendo en mente la premisa que entre mayor sea la cantidad de datos mejor serán los resultados obtenidos y el número de etiquetas hechas manualmente solo alcanzó la cifra de 820 imágenes y 3.200 etiquetas por lo que se opta por crear un conjunto de datos hibrido, aumentando su tamaño mediante el uso del API de COCO el cual es un repositorio de detección, segmentación y subtitulación de objetos a gran escala, con esto se logran descargar 29,084 imágenes con un total de 75.748 etiquetas, en la figura 23 se

puede observar el diagrama de bloques del algoritmo desarrollado en Python para este propósito.

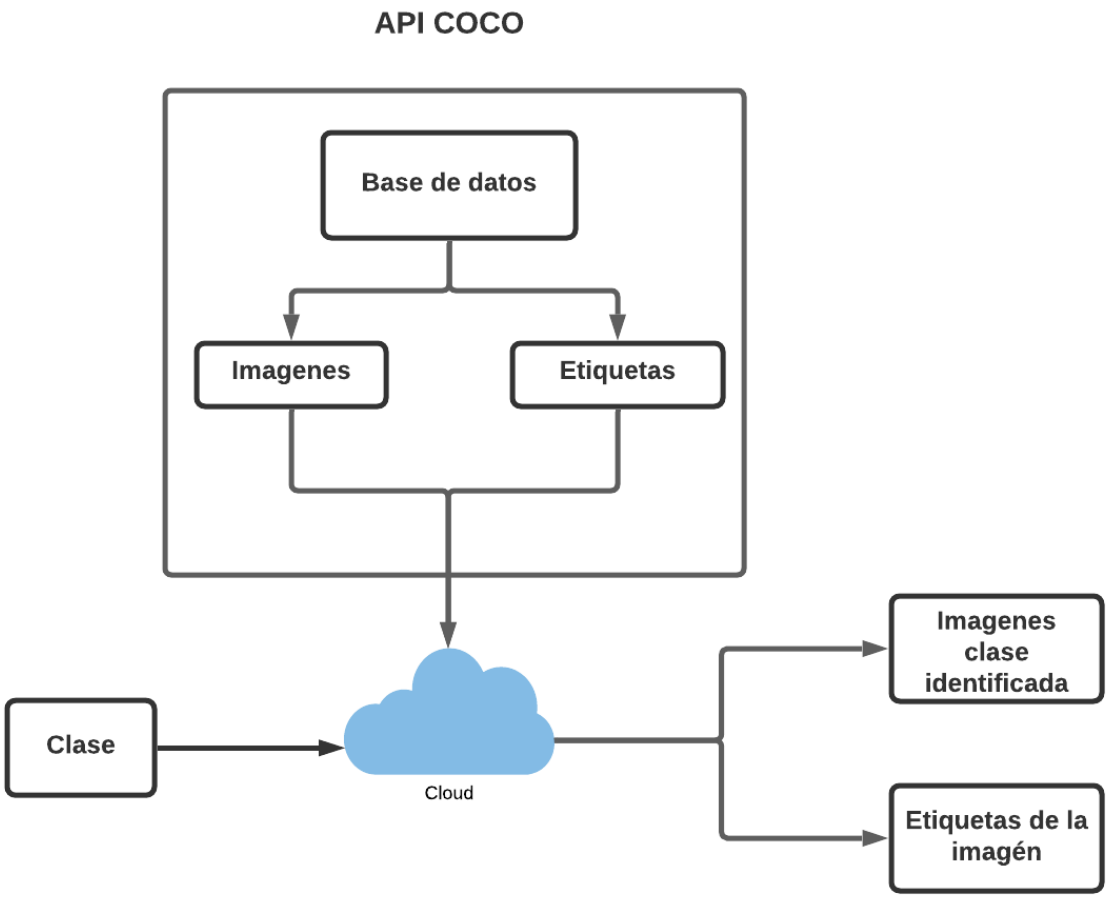

Figura 23. Diagrama del algoritmo utilizado para descargar imágenes y etiquetas de COCO, fuente autor

En la figura 25 se presenta una muestra del conjunto de datos hibrido con el que se realizan algunos de los entrenamientos, las primeras siete imágenes pertenecen a una fracción de los datos descargados de COCO y las últimas dos imágenes corresponden a los cuadros de los videos captados desde la motocicleta, si el lector desea acceder a los conjuntos de datos de prueba y entrenamiento utilizados en el desarrollo de este proyecto lo puede hacer dando clic aquí.

La estructura general de la carpeta se muestra en la figura 24, dentro de las subcarpetas se encontrarán los archivos en formato record los cuales se utilizaron para el entrenamiento como también dos carpetas con el conjunto de datos en su forma base una carpeta contendrá las imágenes y la otra las respectivas etiquetas, también encontrará los códigos hechos en Python para crear los archivos de entrenamiento.

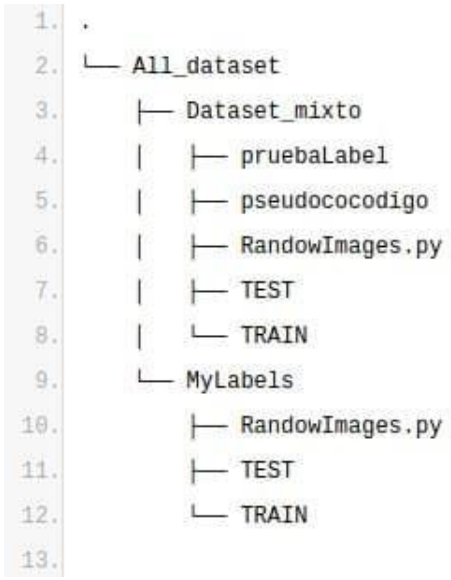

Figura 24. Estructura general de la carpeta que contiene los conjuntos de datos, fuente autor

Figura 25. Muestra del conjunto de datos generado, fuente autor.

Cuando se crea el conjunto de datos se desarrolla un algoritmo que organiza los datos para tener los conjuntos de prueba y entrenamiento con una equivalencia de 20% y 80% respectivamente (el diagrama del algoritmo utilizado se puede ver en la figura 26). Luego se debe transferir la información de los archivos XML correspondiente a cada una de las imágenes a un archivo Excel para ello se desarrolla un script en Python el cual se puede ver representado en la figura 27 y servirá como parámetro de entrada al script desarrollado por un usuario de github que genera el archivo con formato record.

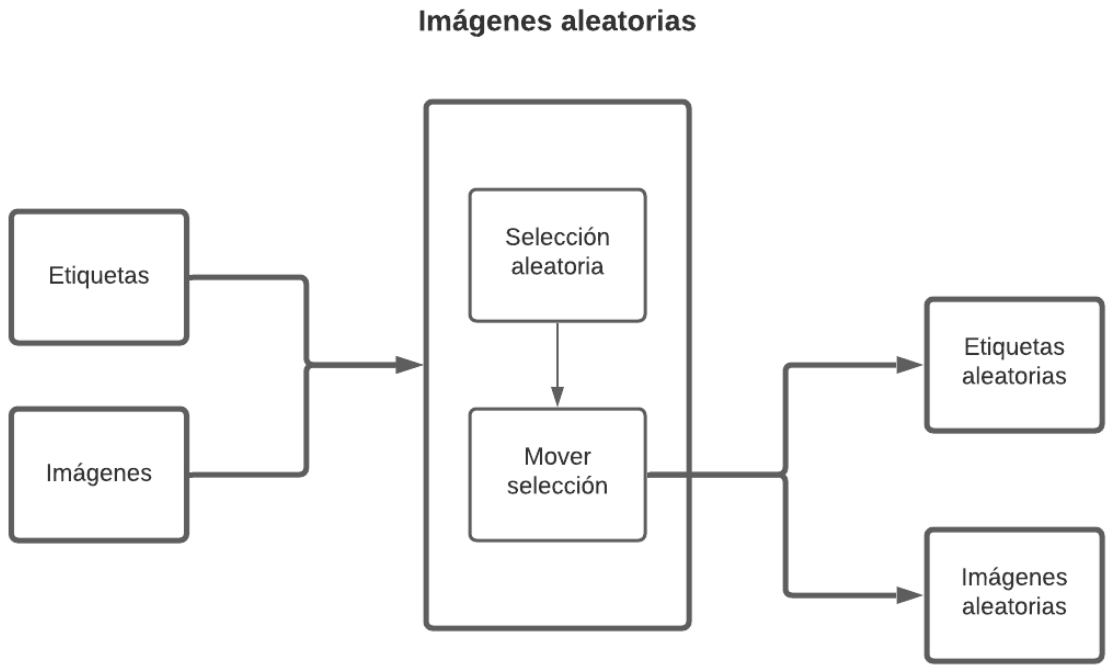

Figura 26. Diagrama del algoritmo utilizado para crear el conjunto de datos de prueba y entrenamiento, fuente autor

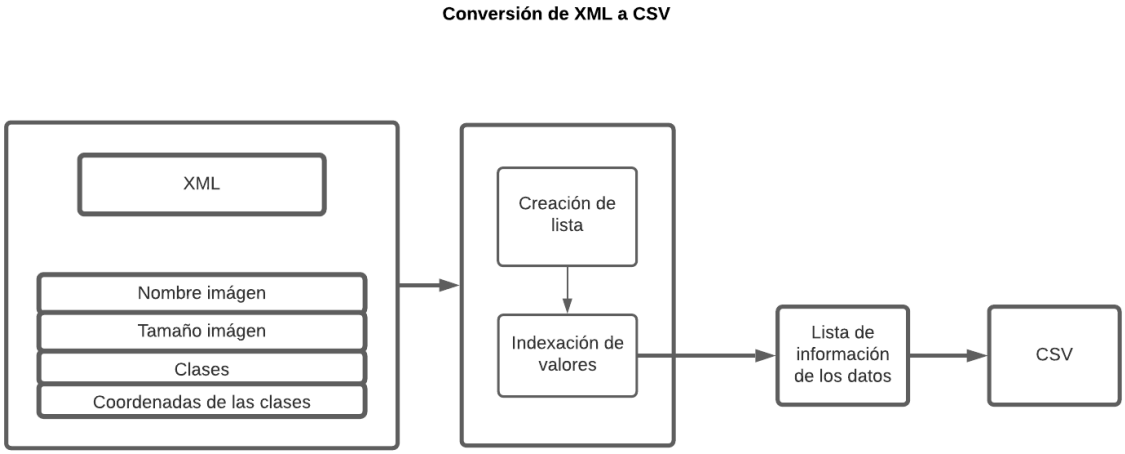

Figura 27. Diagrama del algoritmo utilizado para transferir la información necesaria de los archivos XML a un CSV, fuente autor

Con ayuda de los script que proporciona la biblioteca de tensorflow se inicia el proceso de reentrenamiento de la red previamente entrenada para una función similar de reconocimiento de objetos, teniendo como prerrequisito los archivos de entrenamiento y prueba en formato record, un archivo en formato pbtxt el cual es un mapa de etiquetas donde se le asigna un id a cada una de las clases existentes (figura 28) en este caso son

cinco una para cada tipo de vehículo y finalmente el modelo pre entrenado que se ha seleccionado (se debe adaptar la configuración del archivo llamado pipeline.confing según las necesidades teniendo en cuenta el número de etiquetas, la tasa de aprendizaje, la ubicación de los archivos previamente nombrados entre otras cosas; este archivo está incluido en la carpeta del modelo previamente entrenado el flujo de configuraciones realizadas a este archivo, el esquema de lo mencionado anteriormente se puede ver en la figura 29); durante este proceso se utiliza la CPU a un 98% de su capacidad así como 13GB de RAM, un tiempo de entrenamiento de aproximadamente 3 días, usando la versión 1.15 de TensorFlow.

```
item {
  id: 1
  name: 'BICYCLE'
}

item {
  id: 2
  name: 'BUS'
}

item {
  id: 3
  name: 'CAR'
}

item {
  id: 4
  name: 'MOTORCYCLE'
}

item {
  id: 5
  name: 'TRUCK'
}
```

Figura 28. Mapa de etiquetas, fuente autor

La figura 29 muestra los parámetros establecidos en el archivo de configuración con el que se ejecutaron los entrenamientos de los modelos.

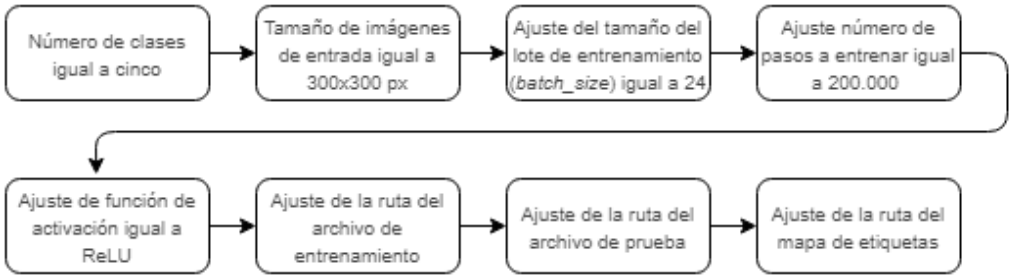

Figura 29. Flujo de configuraciones realizadas al archivo de configuración, fuente autor.

Al realizar el proceso de entrenamiento el algoritmo genera un archivo que sirve como punto de control y proporciona una forma de protección en caso de fallos eléctricos, de computadora u otras fallas externas que impidan culminar el proceso de entrenamiento, este archivo que se genera proporciona un punto de partida sin tener que iniciar el entrenamiento desde cero, no obstante al terminar el entrenamiento es necesario transformar el archivo en algo que sea capaz de hacer inferencia este proceso se conoce como “frozen graph”, posteriormente se convierte el gráfico en un archivo de búfer plano Tensorflow Lite y finalmente se compila el modelo para Edge TPU (Pruebas en la Coral), así se pueden hacer pruebas funcionales de inferencia directamente sobre la tarjeta de desarrollo todo esto se lleva a cabo con ayuda de los scripts proporcionados por Google.

Al realizar los entrenamientos de las redes se pueden obtener las gráficas de error total en donde se estabiliza alrededor de 5 en cada entrenamiento sin importar el conjunto de datos que tenga de entrada.

En la figura 30 se ve representada la pérdida total del entrenamiento usando la red MobileNet SDD v1 y el conjunto de datos híbrido, se entrena hasta las cien mil iteraciones donde se estabiliza en 5.4, se puede evidenciar que el entrenamiento presenta ruido lo cual se esperaba debido a que se entrenó con un conjunto de datos híbrido y las etiquetas varían un poco en los dos conjuntos.

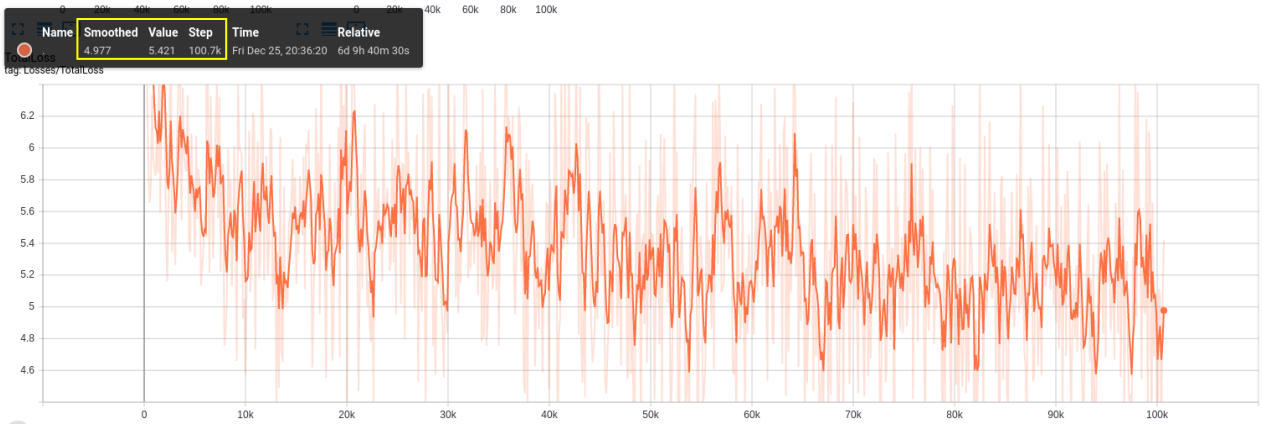

Figura 30. Pérdida total entrenamiento de la red MobileNet SDD v1 y conjunto de datos híbrido, fuente autor

Otro entrenamiento que se realizó fue con la red con la red MobileNet SDD v2 y el mismo conjunto de datos híbrido dando como resultado un entrenamiento que se estabiliza en 5.7, presentando un comportamiento similar a la anterior red.

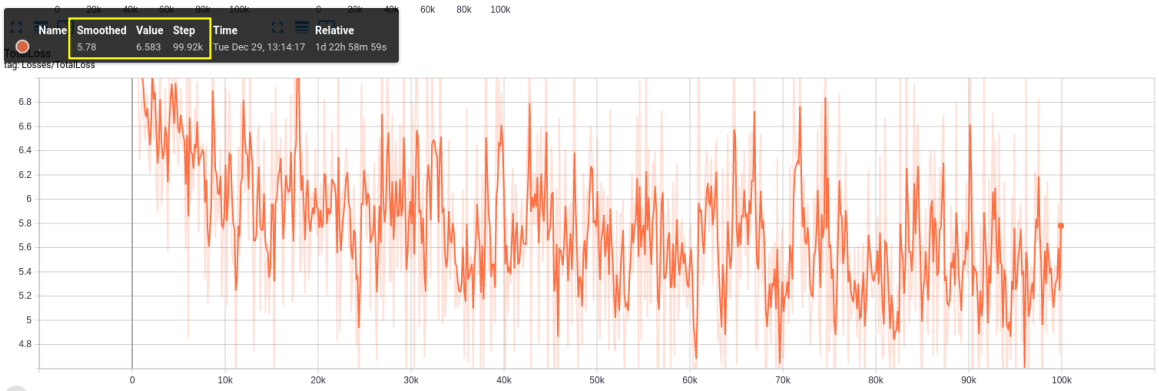

Figura 31. Pérdida total entrenamiento de la red MobileNet SDD v2 y conjunto de datos híbrido, fuente autor

Uno de los entrenamientos que cabe resaltar es un entrenamiento usando la red MobileNet SDD v2 pero esta vez usando únicamente el *dataset* que se etiquetó manualmente, el cual tiene alrededor de 3700 etiquetas y se usa un porcentaje de datos de prueba y entrenamiento del 20% y 80% respectivamente, en este caso no se tiene tanto ruido como en los anteriores entrenamientos lo que puede deberse a la diferencia en las formas en que están etiquetados los datos, se estabiliza en 5.2 dejándolo entrenar hasta las 32000 iteraciones.

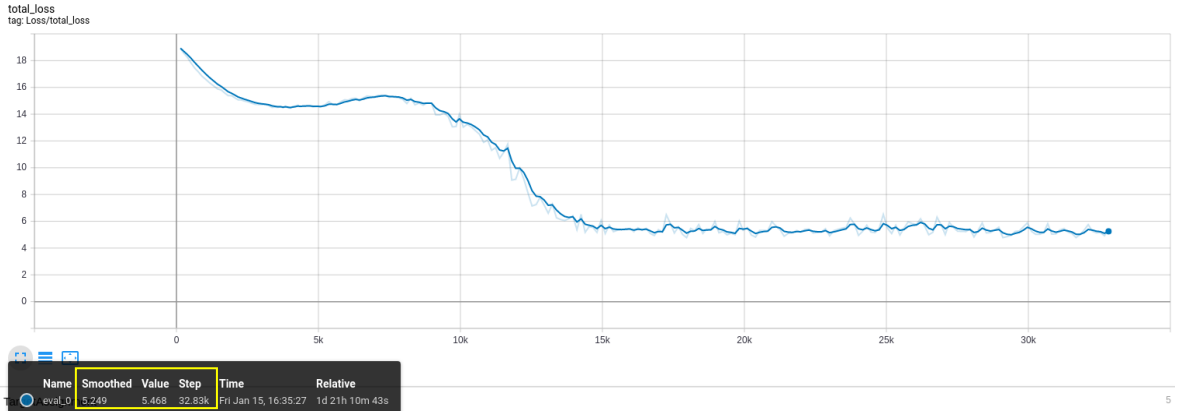

Figura 32. Pérdida total entrenamiento de la red MobileNet SDD v2 y conjunto de datos etiquetado manualmente, fuente autor

7.1.2 SEGMENTACIÓN SEMÁNTICA

La necesidad de una técnica eficaz para resolver algunos problemas de visión por computadora da lugar a la invención de la segmentación de imágenes, lo cual es la tarea de clasificar los pixeles de una imagen. Se pueden identificar dos tipos principales de segmentación de imágenes los cuales son: la segmentación semántica, en la que se identifican los mismos objetos segmentándose con los mismos mapas de colores y la segmentación de instancias, la cual se diferencia de la anterior porque diferentes instancias del mismo objeto se segmentan con diferentes mapas de colores [38].
Al indagar en internet se encuentra la librería PixelLib la cual fue creada para la fácil segmentación de imágenes en la vida real, los pasos que se llevan a cabo con el fin de utilizar la librería, se presentan en la figura 33.

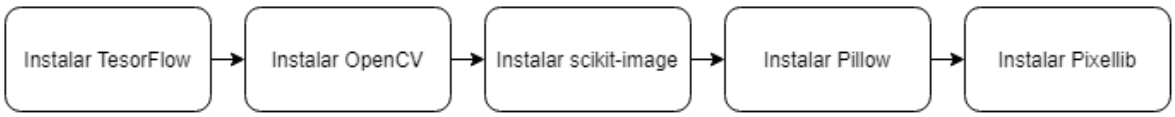

Figura 33. Flujo de instalación de paquetes necesarios para ejecutar la segmentación de imágenes, fuente autor

Luego de instalar los requerimientos se realiza un código en Python para realizar la segmentación, en la figura 34 se puede ver el diagrama del script desarrollado.

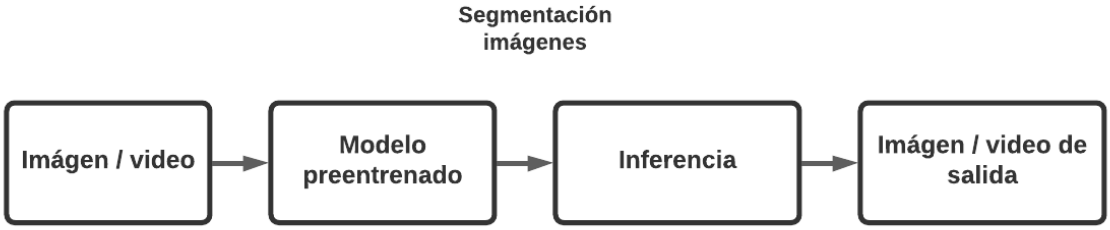

Figura 34. Diagrama del algoritmo utilizado para realizar la segmentación de imágenes, fuente autor.

Cabe aclarar que el modelo que se está cargando (*Mask R_CNN*) en uno proporcionado por la librería, el cual está entrenado con el conjunto de datos de COCO y puede detectar hasta ochenta clases. A continuación, se presenta el resultado de la segmentación de un cuadro del video grabado desde la motocicleta.

La imagen de entrada se puede visualizar en la figura 35.

Figura 35. Imagen de entrada al algoritmo de segmentación, fuente autor

Y el resultado obtenido a partir de la segmentación puede observarse en la figura 36.

Figura 36. Resultado de la segmentación de imágenes, fuente autor

El modelo retorna las coordenadas de la clase detectada a partir de la segmentación que realiza.

7.2 DISTANCIA DE LOS VEHÍCULOS

Para encontrar la distancia aproximada de los vehículos que se aproximan al piloto por la parte trasera de la motocicleta se plantean las siguientes opciones:

7.2.1 RADAR HW-MS03

Para determinar la proximidad de los vehículos se plantea el uso de sensores ultrasónicos (radar), en este caso el módulo HW-MS03 (figura 37) el cual funciona sin importar las condiciones climáticas como humedad, temperatura, etc, trabaja a una frecuencia 2,4 Ghz, con salida digital de 3 V y distancia de medición por default de 10 m en interiores.

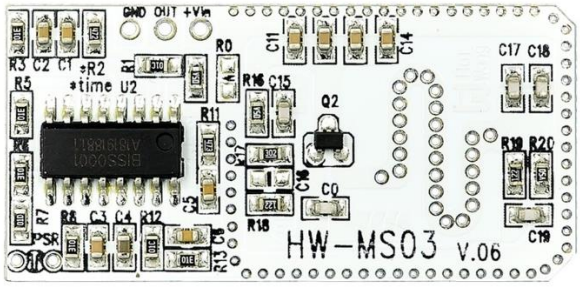

Figura 37. Radar HW - MS03, fuente autor

Se realizan pruebas de laboratorio y se determinar que la distancia práctica detectada usando el montaje de la figura 38 es de 4 m con velocidad de respuesta de alrededor de los 2.1 s.

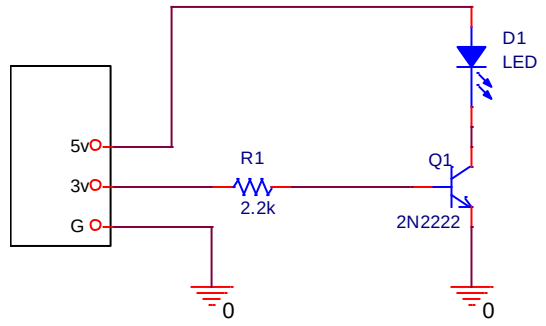

Figura 38. Circuito de salida del sensor, fuente autor

7.2.2 RADAR DE MICROONDAS HB100

El segundo sensor que se toma en consideración es el radar HB100 (figura 39) el cual usa el efecto Doppler para medir la velocidad de un objeto en movimiento, el principio de funcionamiento se basa en su capacidad de medir el desplazamiento de frecuencia que un cuerpo en movimiento produce al momento en que la señal de radiofrecuencia que emite el radar lo golpea. Trabaja por pulsos con un ancho 20u s, un ciclo útil del 1 % a una frecuencia de 10.525 Ghz, con una distancia de medición de alrededor de 20 m y su funcionamiento no se ve altero por factores externos como humedad, luz, ruido, flujo de aire, etc.

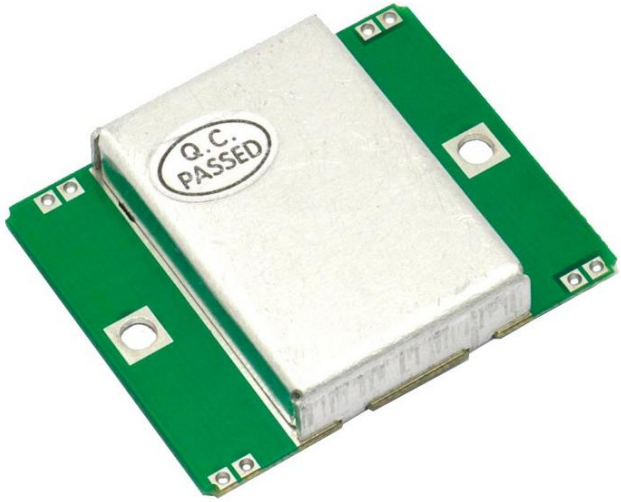

Figura 39. Radar HB100, fuente autor

Para las pruebas de laboratorio fue necesario amplificar la señal de salida del radar con el circuito que se puede visualizar en la figura 40, comparándolo con el radar HW-MS03 este alcanzo una mayor distancia de detección alrededor de 6.5 m en pruebas de laboratorio y

una de las ventajas que tiene es su facilidad para medir la velocidad del objeto, esto se puede lograr gracias a la variación de la frecuencia de la señal senoidal que se tiene en la salida del amplificador la cual es directamente proporcional con la velocidad, aunque el funcionamiento de los sensores utilizados es bueno no se logra conseguir una distancia de medición prudente para lograr informar al conductor de motocicleta, también existe el problema que los sensores solo detectarán el objeto más cercano y se puede dar la posibilidad que se acerquen a la vez mas de un vehículo, es por esto que se migra a la opción de triangulación usando visión estero que es el uso simultáneo de dos cámaras.

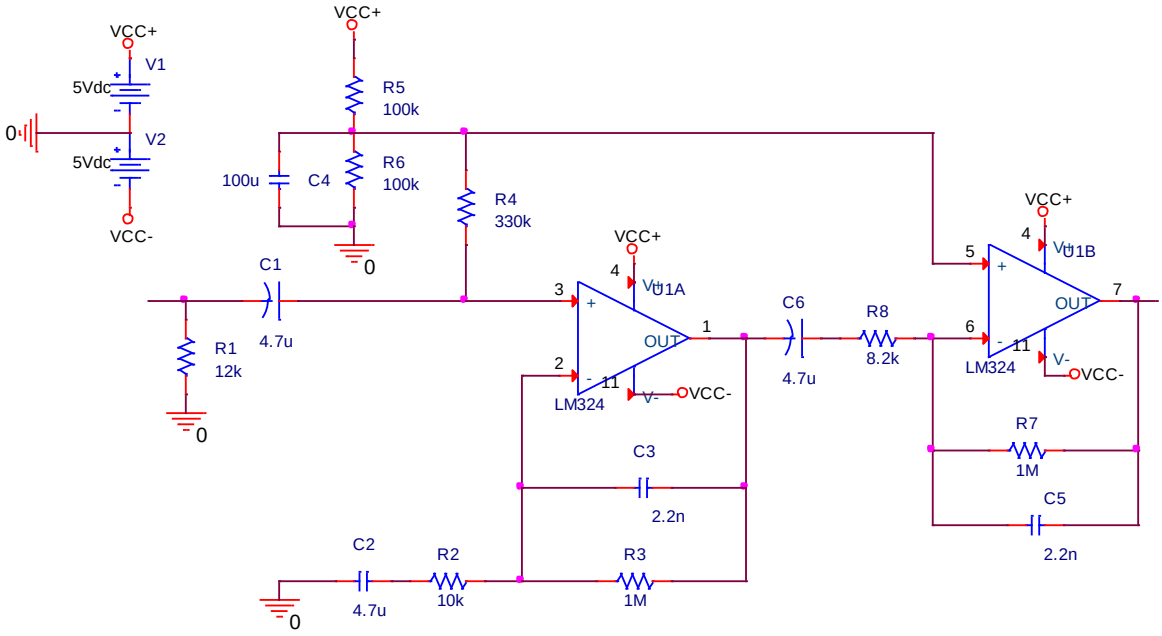

Figura 40. Circuito amplificador de señal, fuente autor

7.2.3 TRIANGULACIÓN MEDIANTE VISIÓN ESTEREO

Teniendo en cuenta los cálculos de la sección 6.2.6 se procede a realizar el código para la detección de profundidad mediante visión estéreo, las pruebas iniciales se hicieron tomando como referencia la pantalla del computador, pero al realizar las pruebas variaba la precisión es por ello que se pasan las dos cámaras Logitech a una base fija, con el fin de disminuir el error presentado, en la figura 41 se evidencian las pruebas iniciales que se realizaron para este método.

Figura 41. Montaje inicial de las dos cámaras, fuente autor

Posteriormente cuando se colocan ambas sobre la base fija se consigue un resultado aceptable no obstante se presentan algunas imprecisiones ya que las cámaras aun tenían la posibilidad de mover el lente y si un factor externo como fuertes ráfagas de viento o si una persona lo logra mover, se perderá la exactitud de los cálculos del código ya que estos dependen de que las dos cámaras se encuentren en paralelo y en el mismo ángulo de visión, en la figura 42 se muestra el diagrama del código utilizado es las pruebas mencionadas.

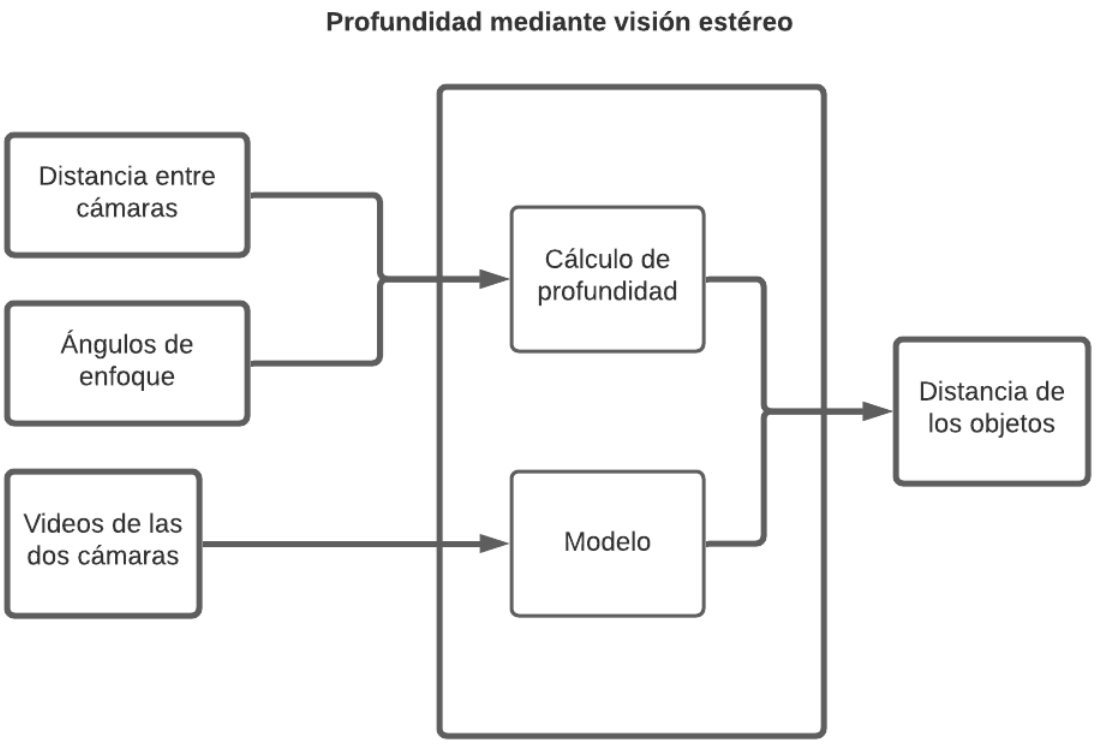

Figura 42. Diagrama del algoritmo utilizado para calcular la profundidad a partir de visión estéreo, fuente autor

7.2.4 USO DE REALSENSE D435I

En este caso se usó una Realsense la cual usa tecnologías de detección de imágenes estereoscópicas utilizando dos cámaras para detectar la profundidad y permiten que los dispositivos vean y comprendan su entorno. Al intentar acoplar el sistema de la Dev Board y la cámara se presentaron varios problemas, los principales fueron incompatibilidad de dispositivos, el sistema operativo que maneja la Coral es limitado debido a que se diseñó específicamente para el uso en la tarjeta y falta de soporte para usar tecnologías como la realsense en ella, para solucionar estos inconvenientes se usó la tarjeta Jetson nano que permite una compatibilidad flexible con varios tipos de dispositivos además que cuenta con una distribución de Linux (Ubuntu 18.4.2 LTS).

La versión del sistema operativo que se usó en la tarjeta fue Jetpak 4.2 (ver figura 43), la vista previa del sistema se puede visualizar en la figura 44, la ventaja de este sistema operativo es su interface gráfica que permite trabajar directamente sobre su sistema en comparación de la tarjeta Coral Dev Board la cual permitía únicamente trabajar a través de conexión SSH.

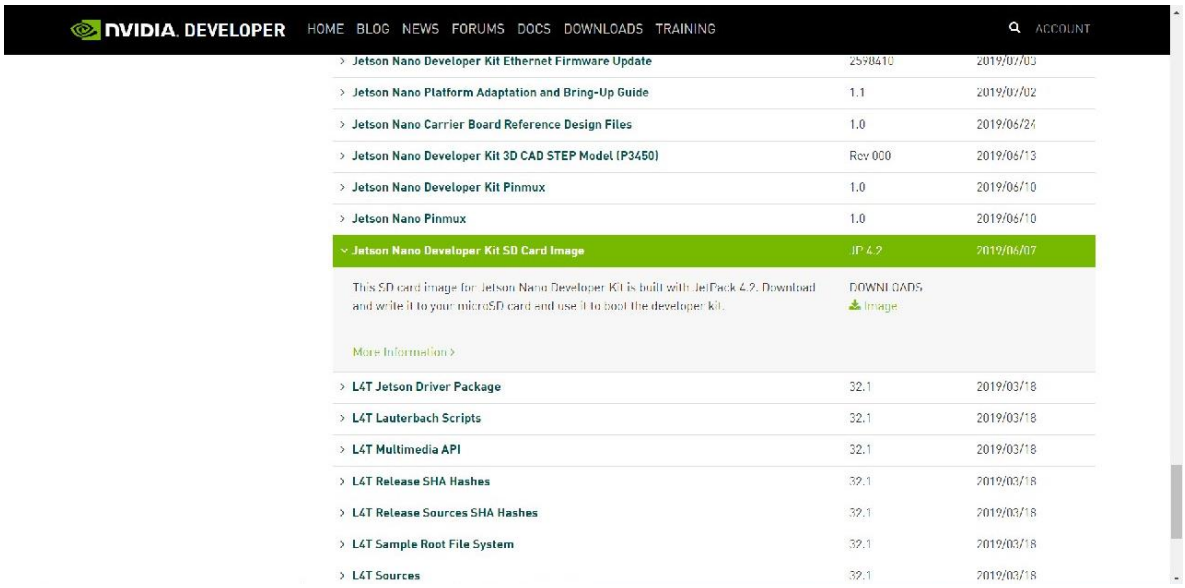

Figura 43. Sistema operativo descargado para el desarrollo del proyecto, fuente autor

Figura 44. Sistema operativo de la Jetson iniciado, fuente autor

La tarjeta tiene dos modos de funcionamiento el modo MANX el cual permite utilizar todas las capacidades del dispositivo y el modo 5W que reduce la potencia y es ideal para proyectos que no sean tan exigentes en el uso de sus recursos. En este caso como la alimentación se realiza por USB con un adaptador de cinco voltios y dos amperios se opta por usar el último modo mencionado para evitar que se apague o reinicie durante la ejecución de los scripts. En documentación oficial se recomienda que al usar el modo MANX se utilice el conector de barril que trae la tarjeta con un adaptador de cinco voltios y

cuatro amperios.

Al realizar las configuraciones iniciales de la tarjeta se realiza la instalación del SDK de la aplicación de la cámara realsense la cual nos permitirá dar una previsualización de los mapas de profundidad generados por la misma; cuando se conecta la cámara a la tarjeta a través del puerto USB el programa la detecta y comienza a generar la previsualización del entorno en una gráfica 3D, en caso de que no se ejecute es recomendable validar las conexiones, a continuación se muestra una imagen captada de la aplicación ejecutándose en la tarjeta Jetson Nano.

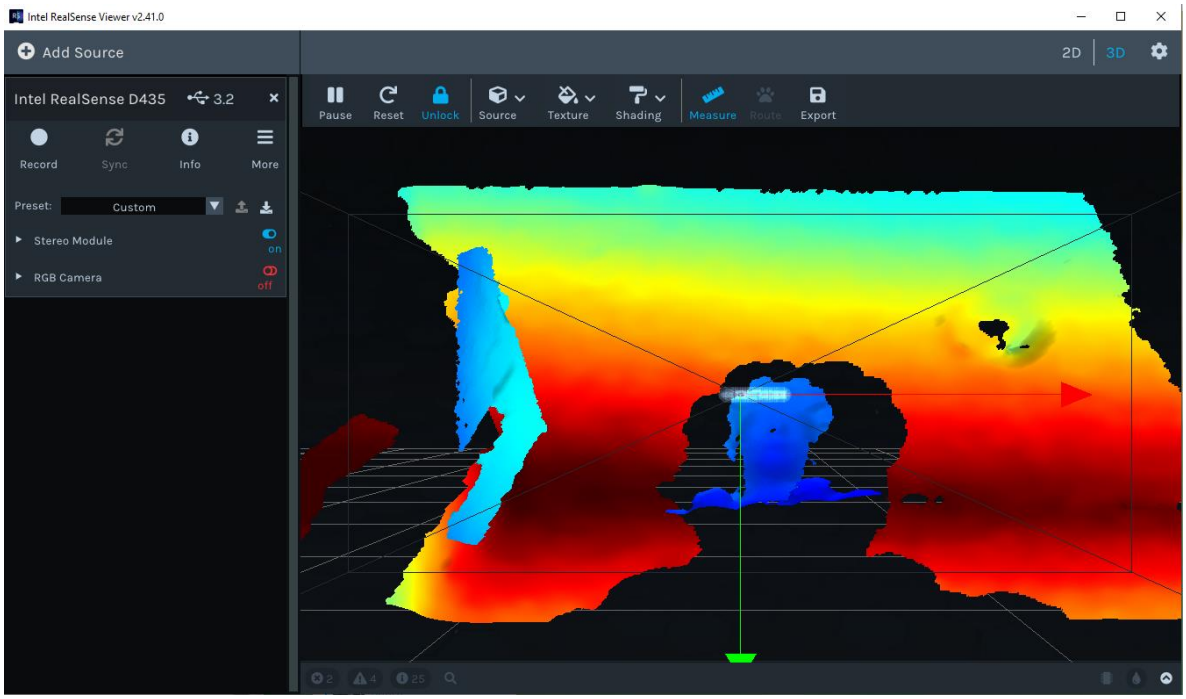

Figura 45. Visualización de la profundidad en la aplicación de la realsense, fuente autor

Para la detección de objetos y profundidad se desarrolló un script de Python que toma como entrada el modelo previamente entrenado detecta los objetos y a partir de las coordenadas de los objetos que retorna el modelo se estima la profundidad del objeto, en la figura 46 se presenta el diagrama del algoritmo utilizado para tal fin.

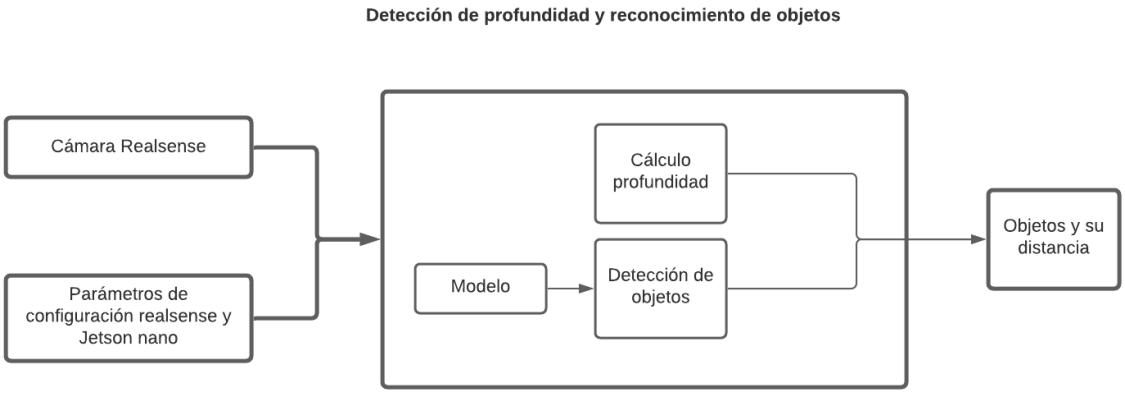

Figura 46. Diagrama del algoritmo para detección de objetos y profundidad, fuente autor.

7.2.4 TIEMPOS DE RESPUESTA DE LOS SENSORES

Teniendo en cuenta que un tiempo de 1.5 Segundos de alerta previa puede prevenir hasta el 90% de las colisiones traseras y con un tiempo de 2.0 se pueden prevenir casi todos los accidentes [23], se presenta la comparación de los tiempos de respuesta de cada método utilizado en la tabla número 2.

Sensor	Tiempo de respuesta (s)
Radar HW-MS03	2.1
Radar de microondas HB100	2.5
Cámaras estéreo	1.9
Cámara RealSense	0.045

Tabla 2. Comparación de tiempos de respuesta obtenidos.

Como se puede evidenciar en la tabla 2 la mayor velocidad de respuesta que se logra obtener en el desarrollo de este proyecto fue mediante el uso de la Cámara RealSense la cual nos proporciona respuesta con alrededor de 22 FPS y teniendo en cuenta esta velocidad y las demás ventajas de los métodos descritos en el documento, tales como distancia detectada, limitaciones del montaje, entre otros, se tuvo como candidato número uno al prototipo que usa para la detección de objetos la Cámara RealSense, el cual cumple con los requerimientos de tiempos descritos del sistema.

7.3 CÁLCULO VELOCIDAD RELATIVA DE LOS VEHÍCULOS

Para el cálculo de la velocidad relativa de los vehículos con respecto al motociclista y teniendo en cuenta que velocidad está asociada al cambio de posición de un objeto a lo largo del tiempo se plantea el uso de la formula estándar de esta para su cálculo.

$$v = \frac{x1 - x0}{t1 - t0} \quad (12)$$

Donde x es la distancia del objeto la cual se logró obtener a partir del uso de la cámara RealSense y t es el tiempo en el que tarda el video de cambiar de cuadro y detectar el objeto nuevamente. Teniendo esta información se logra obtener un estimado de la velocidad de los vehículos con respecto al motociclista.

7.4 SISTEMA DE ASISTENCIA

El sistema de asistencia planteado en el desarrollo de este proyecto es uno que le permita al usuario saber los riesgos que pueda tener mientras conduce, en las validaciones previas se opta por reconocer como peligro potencial para el motociclista los otros vehículos que están en la vía, es por ello que se le suministra al conductor la información obtenida a través del procesamiento de la información del entorno previamente descrito.

Para la solución de este problema se opta por usar un método poco invasivo el cual es el uso de audio para informarle al usuario cada vez que un auto este muy próximo; para esto se usó el API de Google "*Text to speech*" la cual toma como parámetro de entrada un texto y mediante el uso de la inteligencia artificial de Google lo transforma en audio, el problema que se tenía era que la tarjeta Jetson no contaba con salida de audio por lo que una solución sencilla fue usar una adaptador de audio USB figura 48 con lo que solucionó dicho problema, en la figura 47 se puede observar el diagrama de la codificación en Python usada para tal fin.

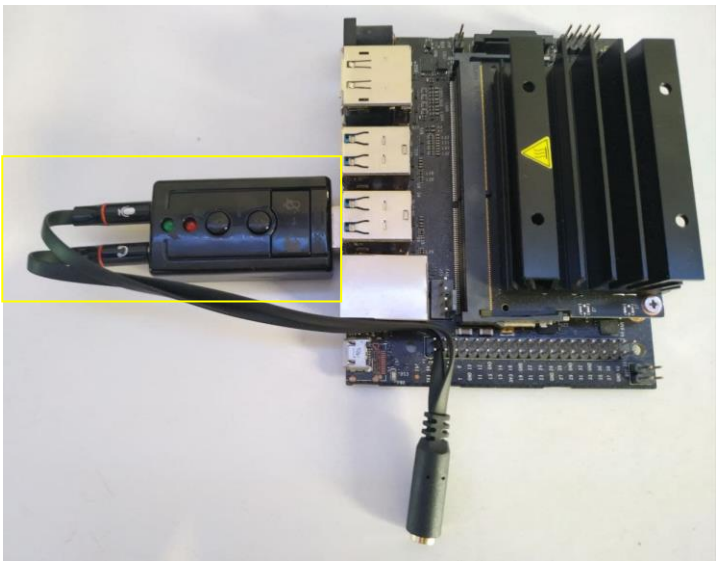

Figura 47. Tarjeta Jetson con adaptador de audio, fuente autor.

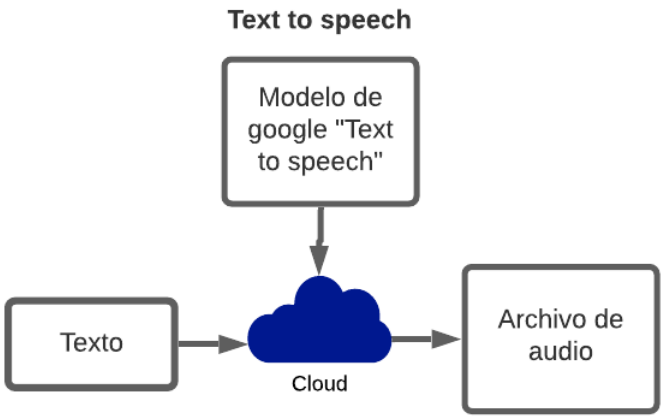

Figura 48. Diagrama del algoritmo utilizado para el uso del API text to speech, fuente autor.

7.4.1 RESPUESTA DEL SISTEMA DE ASISTENCIA

Google proporciona el servicio de "*Text to speech*", el cual crea datos de audio sin procesar del habla humana natural, es decir que crea un audio que suena como una persona hablando y tiene un límite de 1 millón de peticiones mensualmente, al exceder este límite tiene un costo de 16 USD al mes, en el desarrollo del actual proyecto no tuvo costo alguno devino a que no se superó este tope. Ofrece una amplia variedad de voces en diferentes idiomas y permite ajustar la velocidad de locución y el tono de la misma, en este caso se eligió una voz en español masculina para algunas pruebas la cual fue la *es-ES-Wavenet-B* y utiliza la misma tecnología que usan para el asistente y traductor de Google, esta tecnología genera un habla más natural que otros sistemas de conversión de texto a voz [39].

Teniendo en cuenta el artículo 108 del código nacional de tránsito terrestre el cual específica la distancia de seguridad entre vehículos y considerando que el proyecto está planteado para una ciudad se opta por realizar advertencias al conductor cuando se detectan vehículos a menos de 10 metros; el sistema le da una advertencia al conductor cuando detecta vehículos a menos de la distancia mencionada, le informa que tipo de transporte detecta y su distancia como se puede evidenciar en esta muestra de audios generados por el sistema mediante el uso de la biblioteca de Google. El archivo de audio llamado "Audio_01" hace referencia a la respuesta del sistema al detectar el teléfono celular de la figura 52 y los audios que tienen por nombre "Audio_02" y "Audio_03" hacen referencia a la respuesta plateada del sistema al detectar 1 o *n* cantidad de vehículos.

En la figura 49 se presenta el diagrama de la estructura completa del desarrollo final.

Figura 49. Estructura de la propuesta final, fuente autor.

8. RESULTADOS DEL PROYECTO

Para realizar las pruebas funcionales de lo planteado en este documento se utilizan los videos que sé pre grabaron para la creación del conjunto de datos de entrenamiento de los modelos, también se hicieron pruebas en tiempo real en una habitación que tiene diversos objetos para comprobar la correcta medición de profundidad. En este enlace se encuentra algunos ejemplos de los videos utilizados para los análisis que se encuentran en esta sección, así como un video corto donde se presenta una prueba en la habitación la cual detecta los objetos que se encuentran delante de la cámara en este caso detecta un teléfono celular, un teléfono y una persona.

Al realizar pruebas de funcionalidad acoplado la tecnología de RealSense y la detección de objetos se pueden observar los siguientes resultados:

En la figura 50 se puede evidenciar el montaje que se realizó con el fin de determinar si el código realizado (figura 46) para detectar los vehículos y su profundidad funciona correctamente, para las pruebas en laboratorio se usó el modelo preentrenado de COCO para detectar objetos.

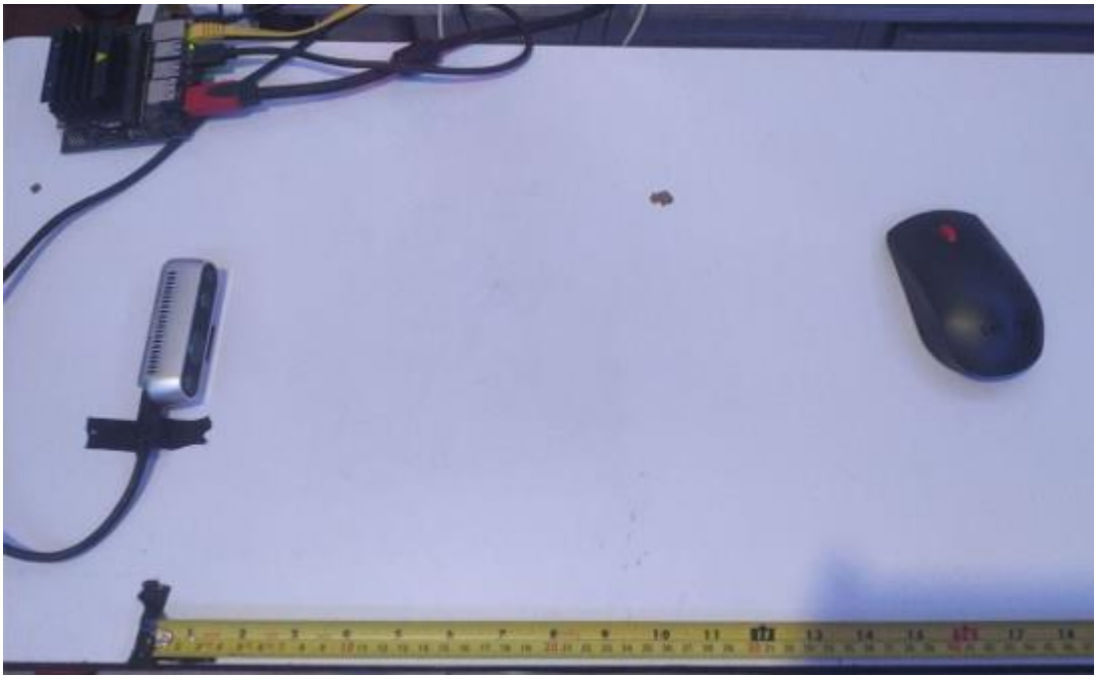

Figura 50. Montaje Jetson y Realsense, fuente autor.

A continuación, se presenta un ejemplo de detección de un objeto como también su profundidad, para la prueba se utiliza un teléfono. Para validar la que la distancia retornada por el programa sea la distacia real se utilizó un metro y se hicieron divisiones

sobre la mesa cada 10 centimetros.

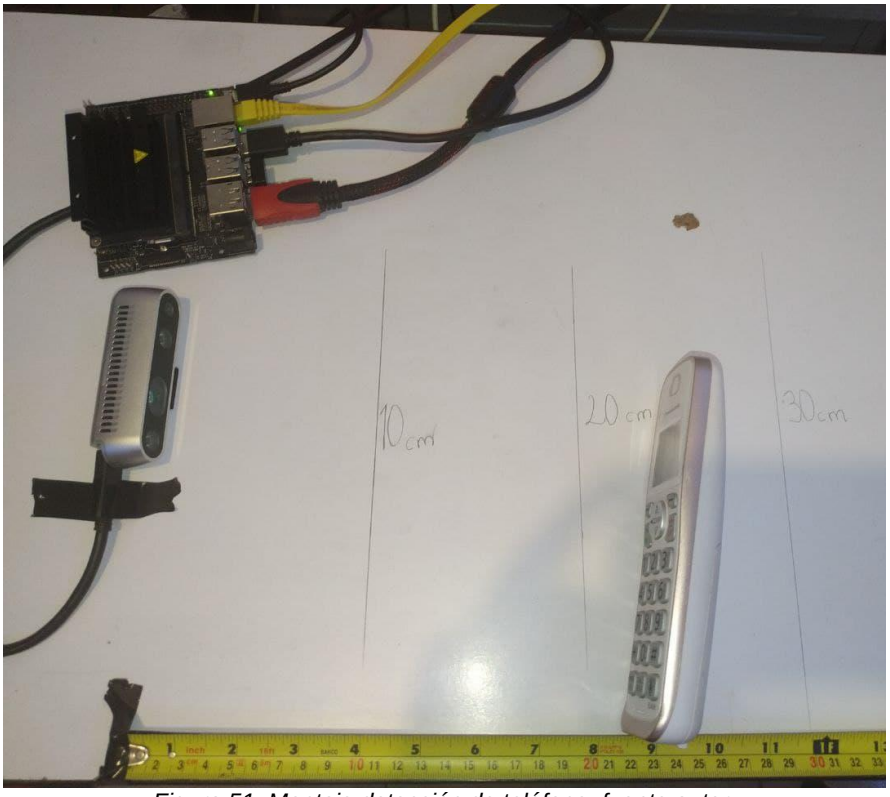

Figura 51. Montaje detección de teléfono, fuente autor.

Los resultados obtenidos por el programa son muy cercanos a la realidad, el código retorna la inferencia del objeto y una profundidad aproximada de 0.219 metros (figura 51) lo que equivale a 22 sentimetros y como se evidencia en la figura 52 esa es la distancia a la que se encuentra el teléfono.

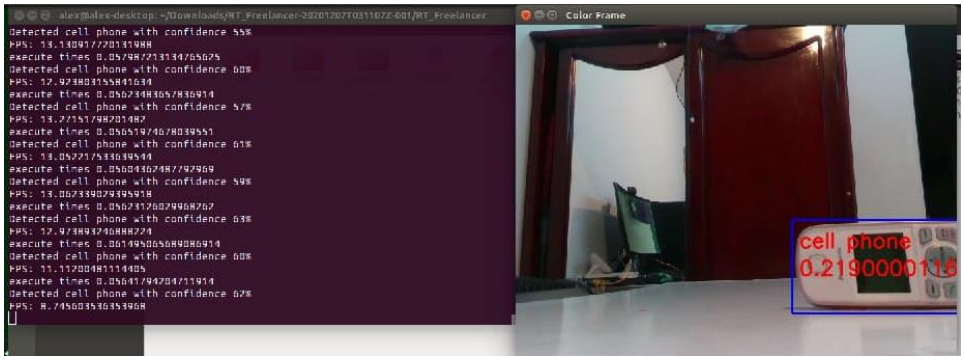

Figura 52. Resultados de detección del teléfono, fuente autor.

Posteriormente al comprobar que el código funcionaba correctamente se pasó a la siguiente fase la cual es verificar que el script desarrollado (figura 46) funcionase para n

cantidad de objetos detectados lo cual se presenta a continuación.

Se agrega una mayor cantidad de objetos para que el modelo pueda hacer las inferencias y como resultado se logra evidenciar que el modelo funciona adecuadamente para calcular la profundidad de cualquier cantidad de objetos detectados.

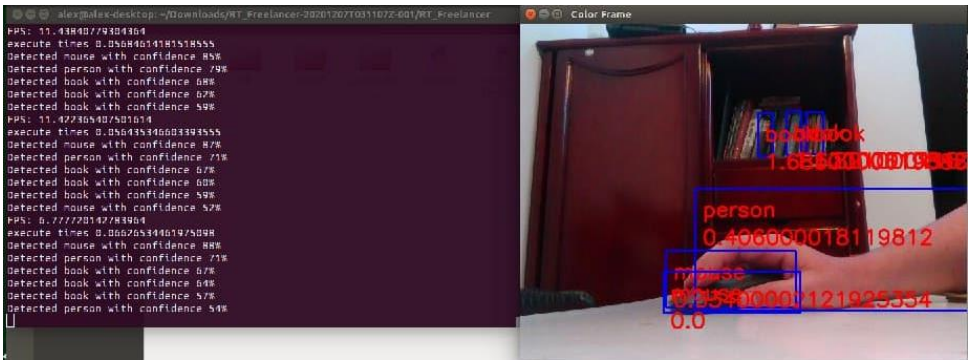

Figura 53. Resultados de la detección de varios objetos, prueba uno, fuente autor.

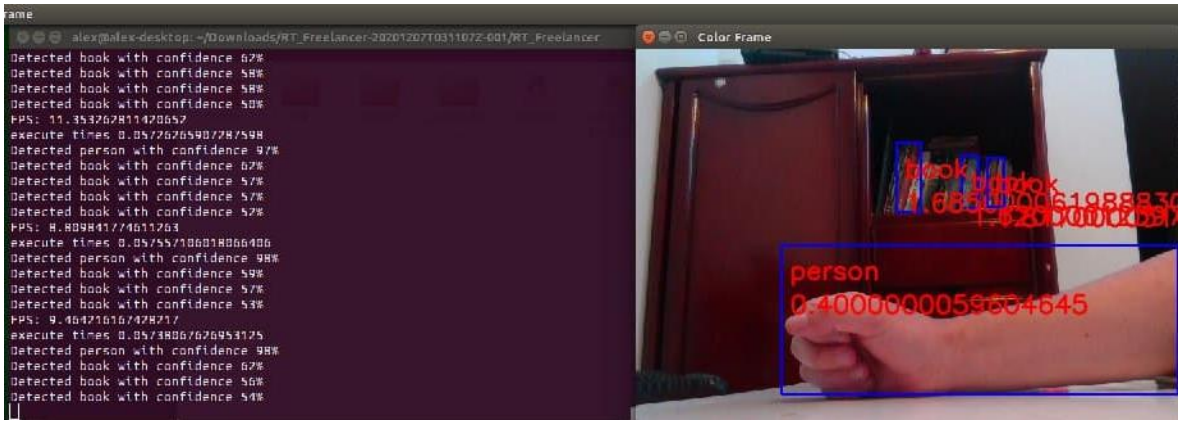

Figura 54. Resultados de la detección varios objetos, prueba dos, fuente autor.

A partir de los cálculos para la obtención de la profundidad y teniendo en cuenta la ecuación número 12, se puede estimar la velocidad relativa de los vehículos que se acercan al motociclista por la parte trasera, para el desarrollo del algoritmo se hizo uso del cálculo de la distancia entre puntos (ecuación 13) para determinar si el objeto del frame anterior es el mismo objeto detectado en el frame actual.

$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2} \quad (13)$$

Teniendo en cuenta lo anterior en la figura 55 se presentan los resultados de la detección de un objeto con su distancia y velocidad relativas, como se puede observar la velocidad está en cero debido a que no se encuentra en movimiento.

Figura 55. Resultados de la distancia y velocidad de un objeto inmóvil, fuente autor.

En la figura 56 se evidencia los resultados de la detección de objetos con su distancia y velocidad relativas cuando están en movimiento, el reloj detectado cambia su velocidad de 0 a aproximadamente 0.348 m/s.

Figura 56. Resultados de la distancia y velocidad de objetos en movimiento, fuente autor.

A continuación (figuras: 57, 58, 59 y 60), se presentan los resultados de los modelos entrenados para la deteccion de vehiculos, se puede evidenciar que en general se lograron resultados aceptables con los dos tipos de redes, no obstante aun se pueden mejorar las detecciones ya que en algunos frames de los videos de grabados desde la motocicleta no detecta el vehículo, sin embargo en el frame posterior a la no detección lo reconoce apropiadamente.

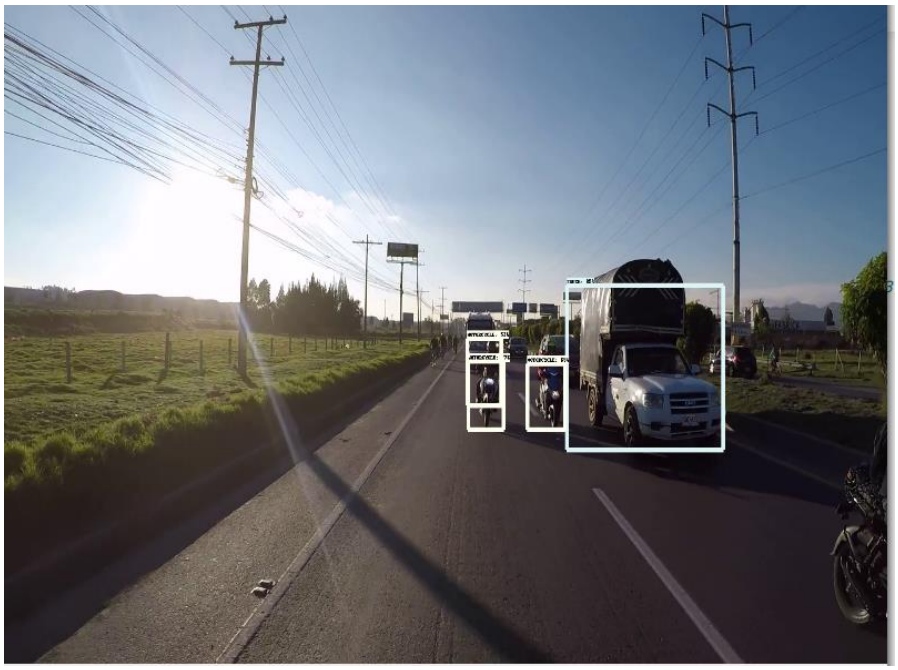

Figura 57. Resultados de la detección de vehículos, prueba uno, fuente autor.

En la figura 58 se ve un ejemplo de un *frame* en donde no reconoció el vehículo de la izquierda, esto ocurre debido a algunos factores como la luz del sol que esta encima del auto.

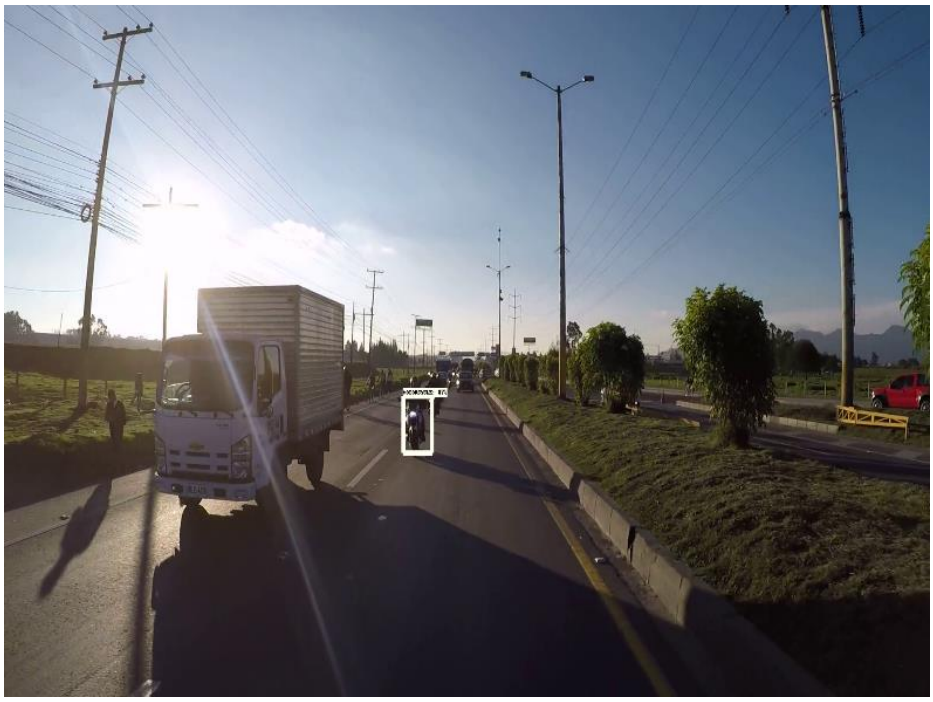

Figura 58. Resultados de la detección de vehículos, prueba dos, fuente autor.

En la figura 59, se muestra un ejemplo donde reconoce apropiadamente las motocicletas del cuadro del video.

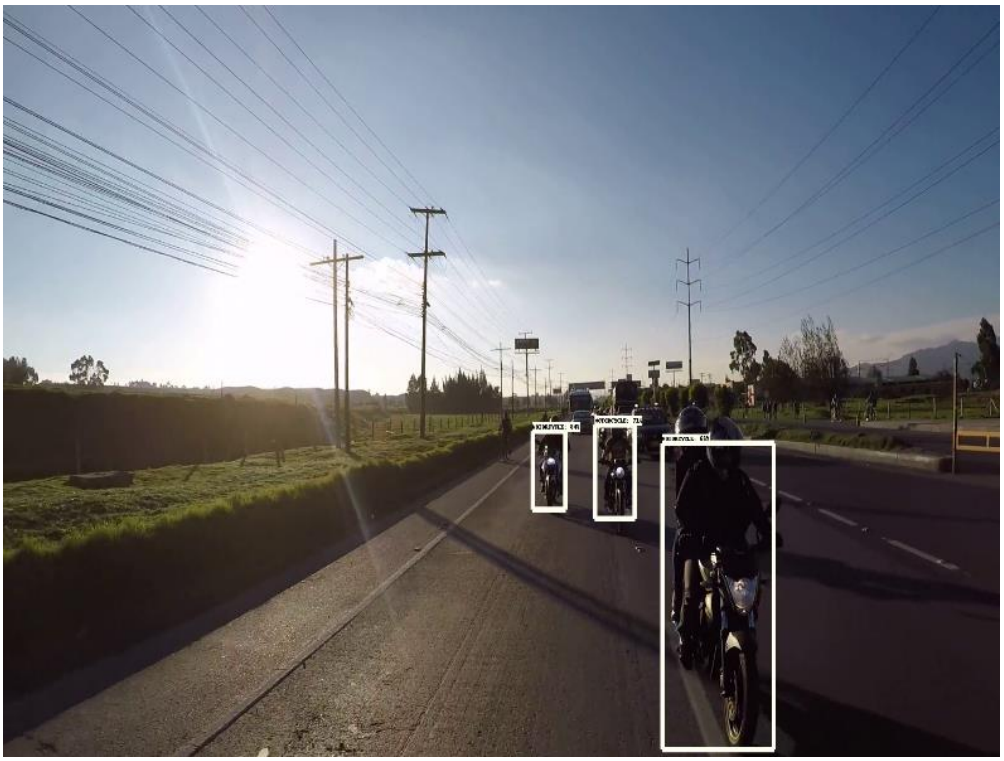

Figura 59. Resultados de la detección de vehículos, prueba tres, fuente autor.

Por último, se presenta la figura 60 donde se evidencia el reconocimiento tanto de carros como motocicletas.

Figura 60. Resultados de la detección de vehículos, prueba cuatro, fuente autor.

En la figura 61 se presentará el flujo de trabajo que se llevó a cabo para entrenar los modelos con los que se realiza la detección de objetos y profundidad.

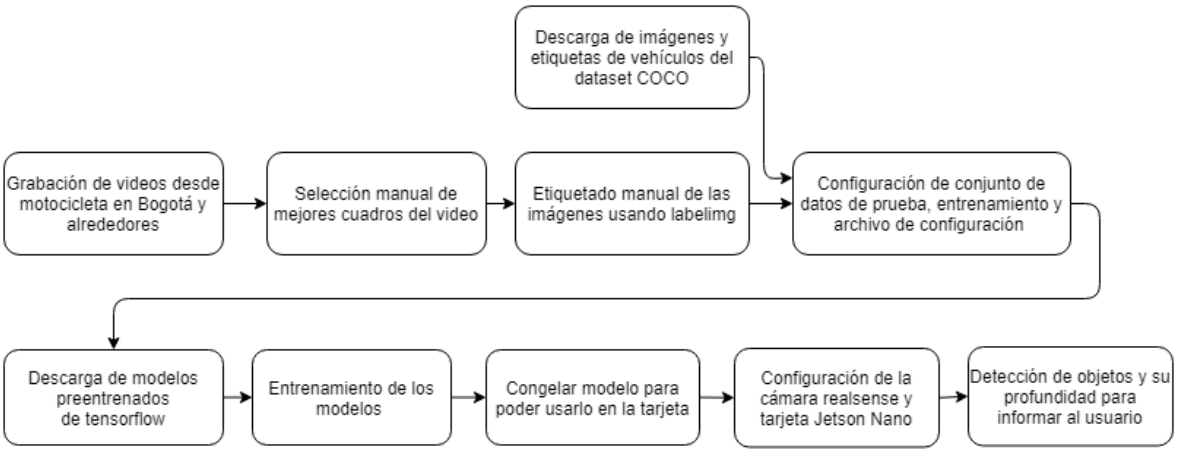

Figura 61. Flujo de trabajo para el entrenamiento de los modelos, fuente autor.

En la figura 62 se presenta el flujo general de la propuesta final con la que se consigue generar resultados aceptables.

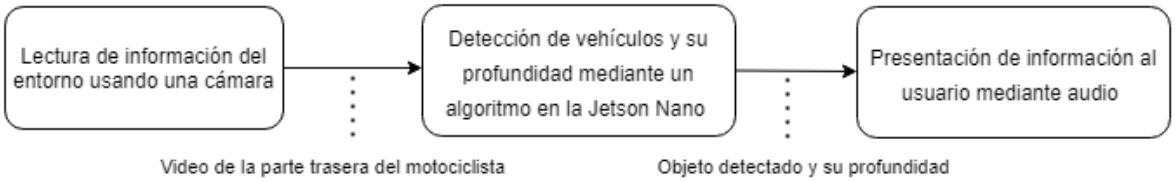

Figura 62. Flujo general de la propuesta final, fuente autor

A continuación, se presenta la tabla 3 donde se muestran las velocidades de inferencia y precisiones de los modelos utilizados para el desarrollo del proyecto. Teniendo en cuenta que la presión mide el porcentaje en que las predicciones son correctas, el mAP corresponde a la precisión media. Estos parámetros se obtienen a partir de la documentación oficial de tensorflow y en los entrenamientos la precisión tiende a converger en el mismo valor como se muestra en la figura 63 el cual es un entrenamiento hecho sobre la red ssd_mobilenet_v2_coco con el conjunto de datos creado manualmente.

Nombre del modelo	Velocidad (ms)	COCO mAP[^1]
ssd_mobilenet_v1_coco	30	21
ssd_mobilenet_v2_coco	31	22

Tabla 3. Velocidades y presiones de los modelos entrenados.

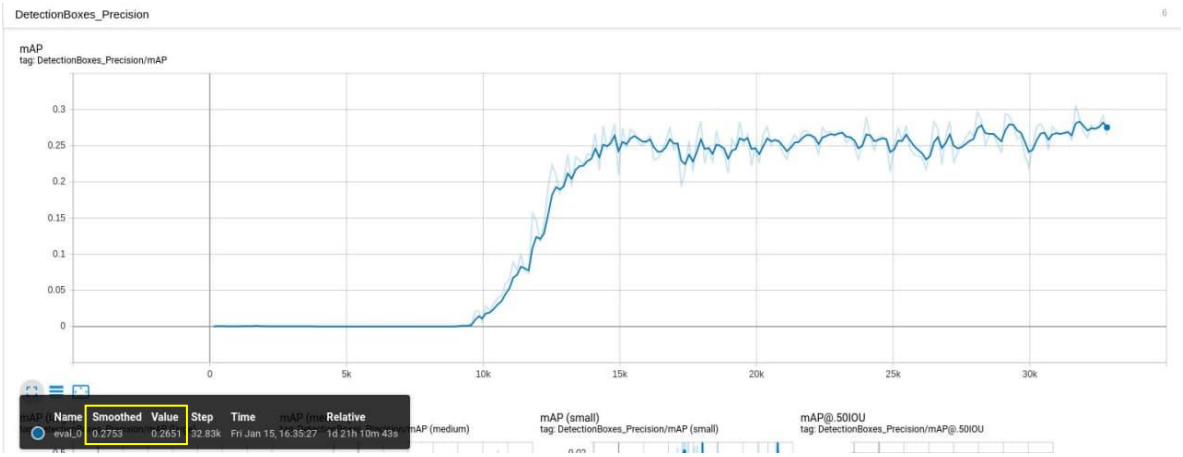

Figura 63. Resultados del mAP del entrenamiento del modelo ssd_mobilenet_v2_coco, fuente autor

A continuación, se presentarán las matrices de confusión de los modelos de aprendizaje automático entrenados.

Las matrices de confusión son una forma de visualizar el rendimiento de un modelo denotando el número de predicciones realizadas correctas o incorrectas. A partir de la creación de la matriz se pueden inferir varios parámetros, para ello es necesario entender las siguientes palabras clave:

- ✓ **Verdadero positivo (TP):** Se refiere al número de predicciones donde el clasificador predice correctamente la clase positiva como positiva.
- ✓ **Verdadero negativo (TN):** Se refiere al número de predicciones donde el clasificador predice correctamente la clase negativa como negativa.
- ✓ **Falso positivo (FP):** Se refiere al número de predicciones donde el clasificador predice incorrectamente la clase negativa como positiva.
- ✓ **Falso negativo (FN):** se refiere al número de predicciones donde el clasificador predice incorrectamente la clase positiva como negativa.

Teniendo en cuenta lo anterior se pueden calcular varios parámetros que determinan la efectividad de las predicciones de un modelo.

Precisión: Brinda la precisión general del modelo, es decir, la fracción del total de muestras que el clasificador clasificó correctamente.

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

Tasa de clasificación errónea: Indica qué fracción de las predicciones fueron incorrectas. También se conoce como error de clasificación**.**

$$\frac{FP + FN}{TP + TN + FP + FN} \tag{145}$$

***Recall*:** Dice qué fracción de todas las muestras positivas fue predicha correctamente como positiva por el clasificador. También se conoce como Tasa de Positivos Verdaderos (TPR), Sensibilidad, Probabilidad de Detección.

$$\frac{TP}{TP + FN} \tag{156}$$

Especificidad: Indica qué fracción de todas las muestras negativas predice correctamente el clasificador como negativas. También se conoce como Tasa Negativa Verdadera (TNR).

$$\frac{TN}{TN + FP} \tag{167}$$

Puntuación F1: combina precisión y recuperación en una sola medida. Matemáticamente es la media armónica de precisión y recall.

$$\frac{2 * TP}{2 * TP + FP + FN} \tag{178}$$

Ahora, en el estado ideal, se quiere un modelo que tenga una precisión de 1 y un *recall* de 1. Eso significa una puntuación de F1 de 1, es decir, una precisión del 100%, que a menudo no es el caso del aprendizaje automático. Entonces, lo que deberíamos intentar es obtener una mayor precisión con un valor de recuperación alto.

Matriz de confusión para ssd_mobilenet_v1_coco:

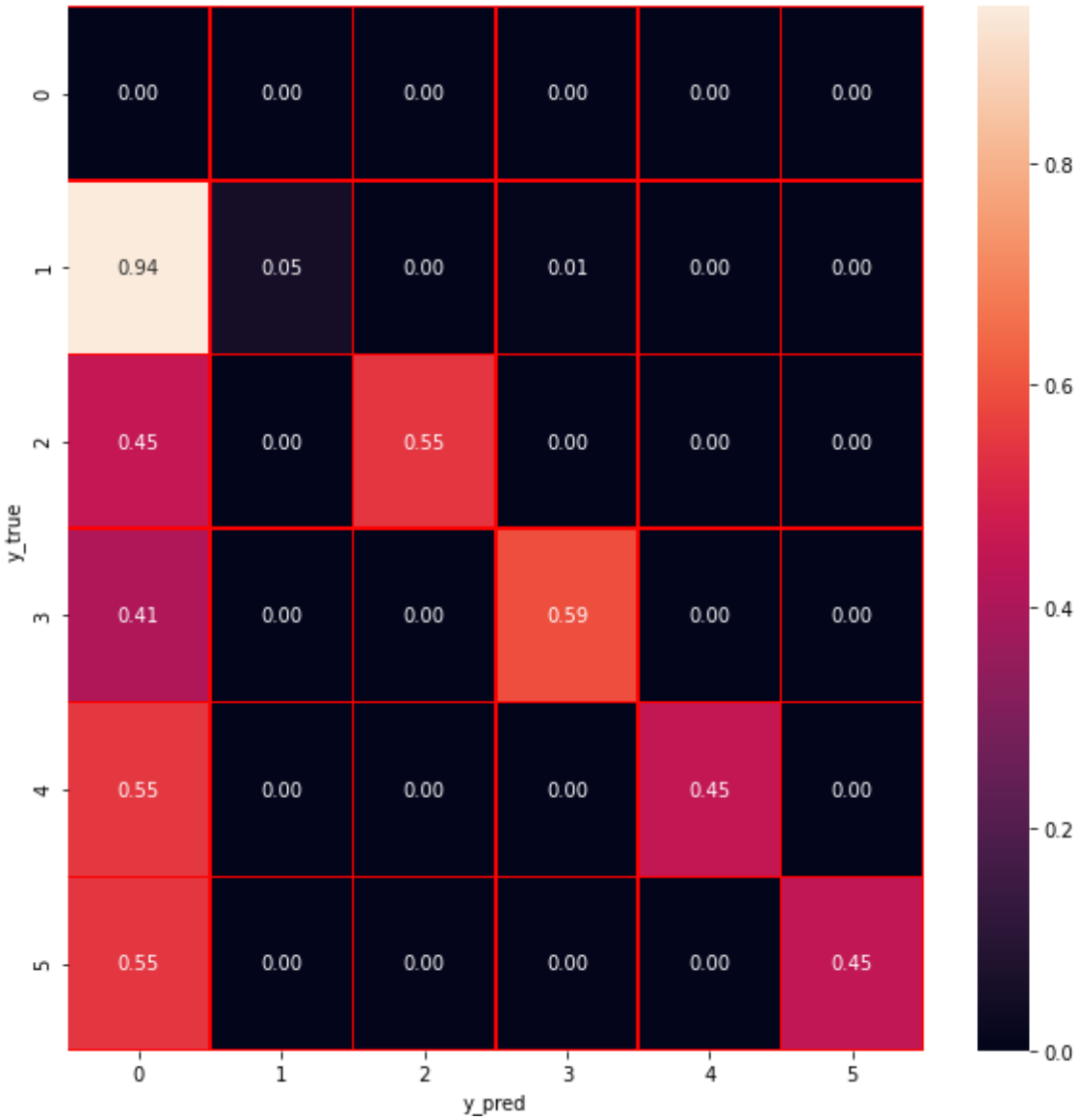

Figura 64. Matriz de confusión para ssd_mobilenet_v1_coco, fuente autor

```
Accuracy: 0.50

Micro Precision: 0.50
Micro Recall: 0.50
Micro F1-score: 0.50

Macro Precision: 0.83
Macro Recall: 0.35
Macro F1-score: 0.46

Weighted Precision: 1.00
Weighted Recall: 0.50
Weighted F1-score: 0.65

Classification Report

              precision    recall  f1-score   support

     Class 1       0.00      0.00      0.00         0
     Class 2       1.00      0.05      0.10       135
     Class 3       0.99      0.55      0.71       276
     Class 4       1.00      0.59      0.74       787
     Class 5       1.00      0.45      0.62       465
     Class 6       1.00      0.45      0.62       242

    accuracy                           0.50      1905
   macro avg       0.83      0.35      0.46      1905
weighted avg       1.00      0.50      0.65      1905
```

Figura 65. Métricas de desempeño del modelo ssd_mobilenet_v1_coco, fuente autor

Matriz de confusión para ssd_mobilenet_v2_coco:

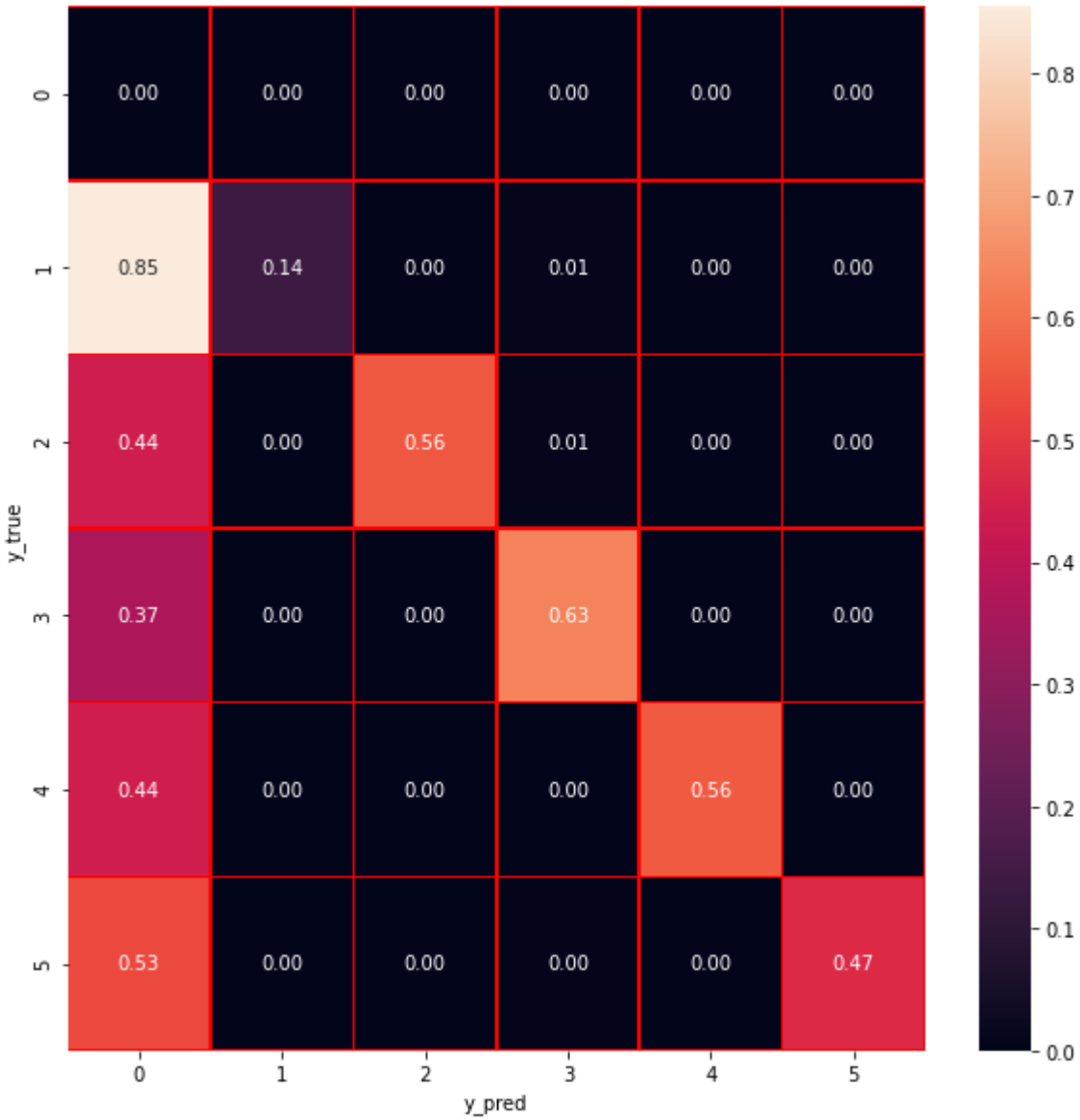

Figura 66. Matriz de confusión para ssd_mobilenet_v2_coco, fuente autor

```
Accuracy: 0.55

Micro Precision: 0.55
Micro Recall: 0.55
Micro F1-score: 0.55

Macro Precision: 0.83
Macro Recall: 0.39
Macro F1-score: 0.51

Weighted Precision: 1.00
Weighted Recall: 0.55
Weighted F1-score: 0.70

Classification Report

              precision    recall  f1-score   support

     Class 1       0.00      0.00      0.00         0
     Class 2       1.00      0.14      0.24       131
     Class 3       0.99      0.56      0.71       261
     Class 4       0.99      0.63      0.77       809
     Class 5       1.00      0.56      0.72       428
     Class 6       1.00      0.47      0.64       252

    accuracy                           0.55      1881
   macro avg       0.83      0.39      0.51      1881
weighted avg       1.00      0.55      0.70      1881
```

Figura 67. Métricas de desempeño del modelo ssd_mobilenet_v2_coco, fuente autor

Matriz de confusión para ssd_mobilenet_v2_coco entrenamiento con conjunto de datos hibrido:

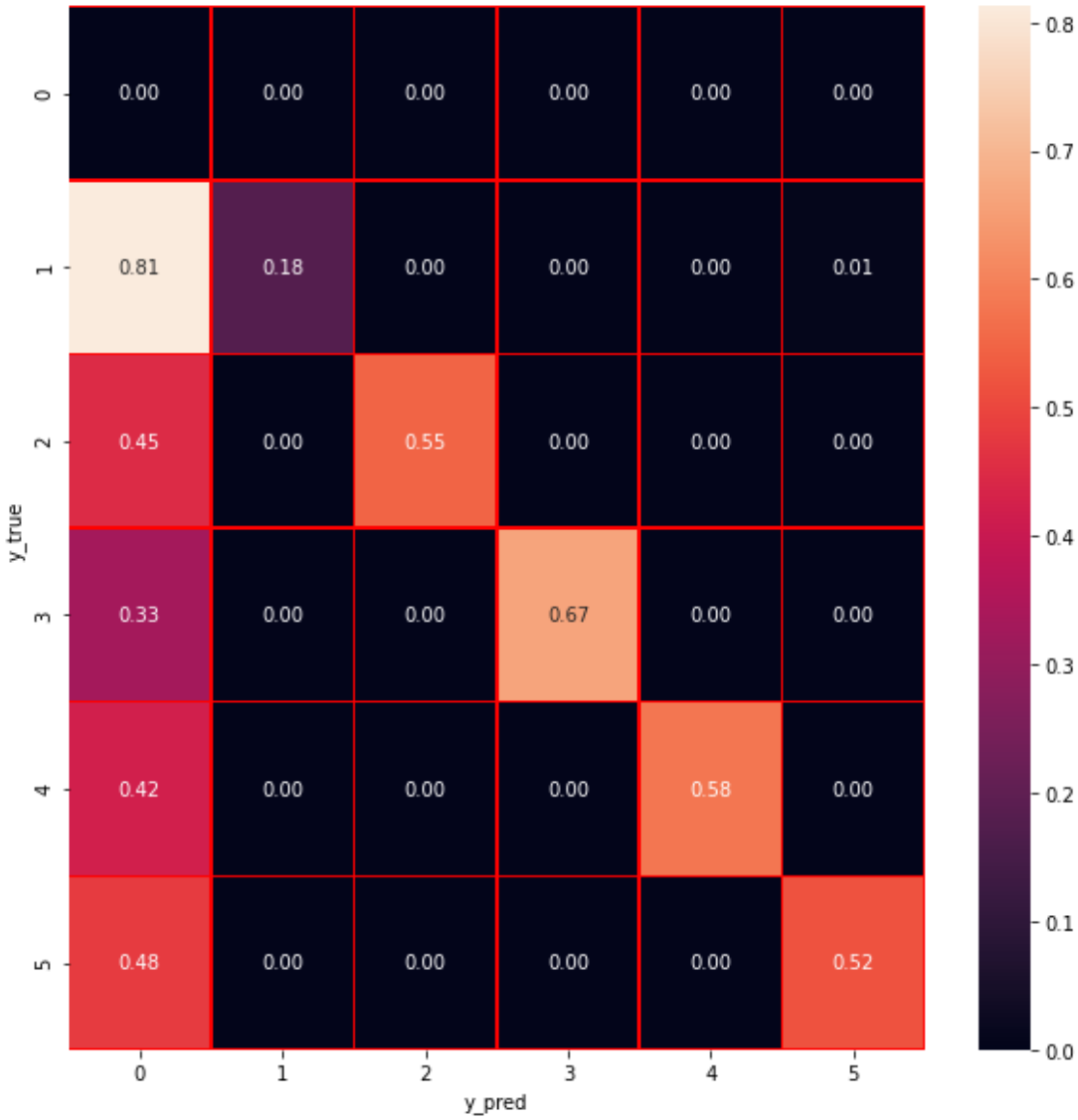

Figura 68. Matriz de confusión para ssd_mobilenet_v2_coco entrenamiento con conjunto de datos hibrido, fuente autor

```
Accuracy: 0.58

Micro Precision: 0.58
Micro Recall: 0.58
Micro F1-score: 0.58

Macro Precision: 0.83
Macro Recall: 0.42
Macro F1-score: 0.54

Weighted Precision: 1.00
Weighted Recall: 0.58
Weighted F1-score: 0.73

Classification Report

              precision    recall  f1-score   support

     Class 1       0.00      0.00      0.00         0
     Class 2       1.00      0.18      0.31       144
     Class 3       1.00      0.55      0.71       318
     Class 4       1.00      0.67      0.80      1049
     Class 5       1.00      0.58      0.73       495
     Class 6       0.99      0.52      0.68       305

    accuracy                           0.58      2311
   macro avg       0.83      0.42      0.54      2311
weighted avg       1.00      0.58      0.73      2311
```

Figura 69. Métricas de desempeño del modelo ssd_mobilenet_v2_coco entrenamiento con conjunto de datos hibrido, fuente autor

Al analizar los resultados de las matrices de confusión se puede inferir que las presiones de los modelos varían en el rango del 50 al 60 porciento siendo las métricas de la clase bicicleta (id=1, figura 28) representadas en las gráficas de las matrices de confusión las que presentan menor desempeño esto se puede deber a que las imágenes de bicicletas en los cuadros de los videos son pequeñas lo cual dificulta su detección.
La clase 0 que se presenta en las matrices hace referencia a la no detección de un objeto, también se puede observar que el modelo que presenta mejor resultado es el mobilenet v2 entrenado con el conjunto de datos hibrido.

En la tabla 4 se encontrará un comparativo de los dos sensores utilizados en el desarrollo de este proyecto, los cuales tienen características similares.

Sensor	Distancia teórica (m)	Distancia practica (m)	Velocidad de respuesta (s)
Radar HW-MS03	10	4	2.1
Radar de microondas HB100	20	6.5	2.5

Tabla 4. Comparación de sensores utilizados.

La principal limitación de los sensores validados es el hecho de que solo puedan detectar un objeto a la vez además que con la configuración de los mismos se lograron calcular distancias limitadas es por ello que opta por utilizar el ultimo método presentado en el desarrollo de este proyecto el cual es el uso de una cámara.

Posteriormente cuando el sistema estaba funcionando en interiores con el algoritmo de detección de objetos se crea un pequeño prototipo el cual se ve representado en la figura 70, se utiliza un sistema de ventilación para refrigerar el disipador de la Jetson y se adquiere un módulo wifi AC600 tp-link UBS con el fin de que el sistema pueda hacer las peticiones a internet necesarias para el asistente.

Figura 70. Prototipo final, fuente autor

Luego se realizan pruebas funcionales en un entorno con poca congestión vehicular como lo es un conjunto vehicular, en la figura 71 se puede ver como quedo acoplada la cámara al casco de un motociclista.

Figura 71. Cámara RealSense en el casco del motociclista, fuente autor

En la figura 72 se puede observar el prototipo final usado por un motociclista y en la figura 73 un ejemplo de las detecciones realizadas.

Figura 72. Motociclista usando el prototipo final, fuente autor

Figura 73. Detecciones realizadas con el sistema en funcionamiento, fuente autor

En la tabla 5 se comparan los resultados de los dos métodos de detección de profundidad utilizados en el desarrollo de este proyecto.

Prueba	Distancia real (cm)	Distancia visión estéreo (cm)	Distancia Realsense (cm)
1	85	74,5	84,8
2	121	129,4	121
3	274	259,2	273,9
4	362	349,9	361,7

Tabla 5. Resultados pruebas de medición distancias con los métodos utilizados.

En la tabla 6 se presenta una comparación de los costos de los métodos y sensores utilizados en el desarrollo de este proyecto.

Sensor	Resultado de funcionalidad	Costo
Radar HW-MS03	En las pruebas de laboratorio se presenta obtienen los resultados presentados en la tabla 2, no obstante, uno de los principales problemas es el hecho de que solo puede	20.000 COP

	detectar un objeto.	
Radar de microondas HB100	En las pruebas de laboratorio se presenta obtienen los resultados presentados en la tabla 2; también presenta el mismo problema del anterior sensor, pero presenta un mejor desempeño.	75.000 COP
Cámaras estéreo	En este caso al usar dos cámaras Logitech se incrementa el costo del desarrollo final, pero se logra solucionar el problema de detección de un solo objeto, no obstante, se presentan algunas dificultades al momento de implementar este método debido a que se debe tener en cuenta los ángulos de inclinación de las cámaras y las dos cámaras a utilizar deberían ser iguales.	660.000 COP
Cámara RealSense	En el caso de la cámara Realsense se logra un resultado favorable logrando detectar múltiples objetos y quitando los problemas del método de la visión estéreo, además el valor final comparado con el anterior sistema es muy semejante lo que lleva a inclinarse a usar la RealSense.	585.000 COP

Tabla 6. Comparación de costos de los sensores y métodos utilizados.

9. CONCLUSIONES Y TRABAJO FUTURO

- Cuando se va a realizar un entrenamiento de una red previamente capacitada para la detección de objetos no es necesario un conjunto de datos muy grande con unas pocas imágenes (alrededor de 1000) bien etiquetadas se logran obtener excelentes resultados.

- A la hora de crear un sistema que requiera obtener la profundidad de un objeto con métodos de procesamiento de imágenes o circuitos integrados se recomienda usar dispositivos como la cámara RealSense (ver tabla 6) debido a su facilidad de configuración y la amplia comunidad de desarrolladores con la que cuenta.

- El uso de tarjetas diseñadas para el trabajo de proyectos de inteligencia artificial como las que se mencionan en este proyecto proporciona gran versatilidad y facilidad a la hora de ejecutar los códigos desarrollados no obstante se deben tener en cuenta las especificaciones de cada diseñador en este caso concreto la tarjeta Dev Board presento algunos inconvenientes al momento de buscar documentación asociada a la misma y problemas de compatibilidad del sistema operativo.

- Los radares económicos no son recomendables para la detección de varios objetos al mismo tiempo debido a que por su arquitectura solo detectan el objeto más cercano a ellos sin importar la posición del objeto en el cuadrante.

- Las métricas de la clase bicicleta son las que presentan menor desempeño (ver figuras 64, 66 y 68), al realizar el análisis de las imágenes que el modelo toma como entrada se puede evidenciar que se debe en gran medida al tamaño que presentan las bicicletas lo que hace difícil la detección.

Para trabajos futuros:

- Implementar el sistema presentado en este documento usando dispositivos de fácil acceso para el público en general como lo son los celulares inteligentes.

- El aplicativo descrito en el desarrollo de este proyecto también puede usarse en otras áreas como la medicina para ayudar a una persona con ceguera total o parcial a desplazarse en su entorno.

- Implementar un sistema para conductores de motocicleta que además de advertirle de posibles vehículos acercándose le ayude a navegar a través de una ciudad tan congestiona como lo es Bogotá, Colombia.

10. BIBLIOGRAFÍA

[1] Learn on the internet, «Internet Geography,» 2007. [En línea]. Available: http://www.geography.learnontheinternet.co.uk/activities/popn.html. [Último acceso: 12 agosto 2018].

[2] Gobierno Digital Colombia, «Datos Abiertos,» MInisterio de tecnologías de la información y las comunicaciones, 2018. [En línea]. Available: https://www.datos.gov.co/Estad-sticas-Nacionales/Proyecciones-de-poblaci-n-por-Departamento-Colombi/5b7v-4tvp/data. [Último acceso: 13 Agosto 2018].

[3] El Tiempo, «BOGOTÁ: ALTERNATIVAS DE MOVILIDAD URBANAS,» *Moverse dentro de la ciudad para las tareas diarias parece cada vez más complicado, pero opciones son lo que sobran.*, 17 Agosto 2018.

[4] «Los motociclistas son los que más mueren en accidentes viales,» *El TIEMPO*, 24 julio 2017.

[5] «Causas de accidentalidad de motocicletas en Colombia,» *Auto crash*, vol. 41, 2017.

[6] soloautos.mx, «Análisis Especial: El Autopilot de Tesla, cómo funciona, sistemas similares, ventajas, desventajas y precauciones,» 25 abril 2017. [En línea]. Available: https://soloautos.mx/noticias/detalle/an%C3%A1lisis-especial-el-autopilot-de-tesla-c%C3%B3mo-funciona-sistemas-similares-ventajas-desventajas-y-precauciones/ED-LATAM-6701/. [Último acceso: 22 11 2018].

[7] H. J. O. F. y. J. E. Joerg Deigmoeller, «Monocular Rear Approach Indicator for Motorcycles,» Offenbach, Germany, 2014.

[8] IBERDROLA, «¿Somos conscientes de los retos y principales aplicaciones de la Inteligencia Artificial?,» [En línea]. Available: https://www.iberdrola.com/te-interesa/tecnologia/que-es-inteligencia-artificial. [Último acceso: 9 Septiembre 2018].

[9] ERIKIADO, «Inteligencia Artificial 101 - Vision Artificial,» 6 Febrero 2015. [En línea]. Available: https://inteligenciaartificial101.wordpress.com/2015/02/06/vision-artificial/. [Último acceso: 28 Agosto 2018].

[10] «Vision Artificial,» [En línea]. Available: https://sites.google.com/site/sitwebia/vision-artificial. [Último acceso: 3 Septiembre 2018].

[11] ERIKIADO, «Inteligencia Artificial 101 - Procesamiento de Imagenes,» 14 Febrero 2015. [En línea]. Available: https://inteligenciaartificial101.wordpress.com/2015/02/14/procesamiento-de-imagenes/. [Último acceso: 29 Agosto 2018].

[12] Grupo de Investigación EDMANS, TÉCNICAS Y ALGORITMOS BÁSICOS DE VISIÓN ARTIFICIAL, España: Universidad de La Rioja. Servicio de Publicaciones., 2006.

[13] Universidad de los Andes , «HELMY, UN CASCO SALVA VIDAS,» 10 11 2020. [En línea]. Available: https://electricayelectronica.uniandes.edu.co/es/noticia/helmy-un-casco-salva-vidas?s=08. [Último acceso: 20 06 2021].

[14] R. S. S. K. K. Prashant vaya, «Motorcycle Safety Solution using the Internet of,» de *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, GLA University, Mathura, UP, India., 2019.

[15] Department of Electronics and Communication Amity School of Engineering & Technology, «IMPLEMENTATION AND ANALYSIS OF SMART HELMET,» de *International Conference on Signal Processing*, Solan, India, 2017.

[16] Department of ECE Mepco Schlenk Engineering, «Intelligent Transportation System for

Accident Prevention and Detection,» de *International Conference on Intelligent Computing and Control Systems*, Sivakasi, Tamilnadu, 2017.

[17] Department of Electronics and Instrumentation Engineering Sri Ramakrishna Engineering College, «ACCIDENTAL IDENTIFICATION AND NAVIGATION,» Coimbatore, Tamilnadu, India, 2017.

[18] WorldServe Education - K.S. School of Engineering and Management, «A Novel Video Processing Based Smart Helmet for Rear Vehicle Intimation & Collision Avoidance,» de *Conference on Computing and Network Communications*, Bengaluru, India, 2015.

[19] H. R. -. H. ALEJANDRO, «SISTEMA DE VISIÓN R.S. PARA CASCOS,» Mexico.

[20] Fundacion Gonzalo Rodriguez, «Estudio sobre las condiciones del traslado de niños en motocicletas en América Latina,» Fundación Gonzalo Rodríguez - Montevideo, Uruguay (2017), 2017.

[21] «Evidencias para la prevención y control de lesiones en motociclistas,» *Revista Cubana de Salud Pública,* vol. 42, nº 4, 2016.

[22] Virginia Transportation , «News,» 15 Agosto 2005. [En línea]. Available: http://vtrc.virginiadot.org/briefdetails.aspx?id=19. [Último acceso: 16 Agosto 2018].

[23] Mobileye, «DOCPLAYER,» 2015. [En línea]. Available: https://docplayer.es/3161694-Lider-en-sistema-anticolision.html. [Último acceso: 22 agosto 2018].

[24] A. GONZÁLEZ, «CleverData,» 1 Julio 2014. [En línea]. Available: https://cleverdata.io/que-es-machine-learning-big-data/53/. [Último acceso: 03 09 2019].

[25] A. PALMER POL y J. MONTAÑO MORENO, «¿Qué son las redes neuronales artificiales?,» Universidad de las Islas Baleares, Valldemossa.

[26] X. B. Olabe, REDES NEURONALES ARTIFICIALES Y SUS APLICACIONES.

[27] G. J. P. R. ARTEAGA, «APLICACIÓN DEL APRENDIZAJE PROFUNDO ("DEEP LEARNING") AL PROCESAMIENTO DE SEÑALES DIGITALES,» SANTIAGO DE CALI, 2015.

[28] TensorFlow, «TensorFlow Lite guide,» [En línea]. Available: https://www.tensorflow.org/lite/guide. [Último acceso: 14 03 2020].

[29] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, «MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,» de *Computer Vision and Pattern Recognition*, 2017.

[30] Google, «Cloud Speech-to-Text,» [En línea]. Available: https://cloud.google.com/speech-to-text/?utm_source=google&utm_medium=cpc&utm_campaign=na-US-all-en-dr-skws-all-all-trial-b-dr-1009892&utm_content=text-ad-none-any-DEV_c-CRE_491337308301-ADGP_Desk%20%7C%20SKWS%20-%20BMM%20%7C%20Txt%20~%20AI%20%26%20ML%20~. [Último acceso: 29 06 2021].

[31] Google Cloud, «Conceptos básicos de Cloud Text-to-Speech,» [En línea]. Available: https://cloud.google.com/text-to-speech/docs/basics. [Último acceso: 15 Marzo 2021].

[32] Coral beta, «Get started with the Dev Board,» [En línea]. Available: https://coral.withgoogle.com/docs/dev-board/get-started/. [Último acceso: 19 08 2019].

[33] «Central Processing Unit (CPU) vs Graphics Processing Unit (GPU) vs Tensor Processing Unit (TPU),» [En línea]. Available: https://iq.opengenus.org/cpu-vs-gpu-vs-tpu/. [Último acceso: 03 11 2019].

[34] Coral, «TensorFlow models on the Edge TPU,» [En línea]. Available: https://coral.ai/docs/edgetpu/models-intro/#compatibility-overview. [Último acceso: 12 enero 2020].

[35] Nvidia, «Nvidia Developer,» [En línea]. Available: https://developer.nvidia.com/buy-jetson?product=jetson_nano&location=US. [Último acceso: 12 12 2020].

[36] Intel, «Intel RealSense,» [En línea]. Available: https://www.intelrealsense.com/tracking-camera-t265/. [Último acceso: 15 01 2021].

[37] Intel, «Intelrealsense,» [En línea]. Available: https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf. [Último acceso: 16 marzo 2021].

[38] A. Olafenwa, «Image Segmentation With 5 Lines 0f Code,» 17 Mayo 2020. [En línea]. Available: https://towardsdatascience.com/image-segmentation-with-six-lines-0f-code-acb870a462e8. [Último acceso: 15 Marzo 2021].

[39] Google cloud, «Supported voices and languages,» [En línea]. Available: https://cloud.google.com/text-to-speech/docs/voices?hl=en. [Último acceso: 14 02 2021].

[40] RUNT, «Balance del sector tránsito y transporte,» 2018.

[41] Coral, «Dev Board datasheet,» [En línea]. Available: https://coral.withgoogle.com/docs/dev-board/datasheet/. [Último acceso: 03 10 2019].

11. Anexos

Anexo 1:

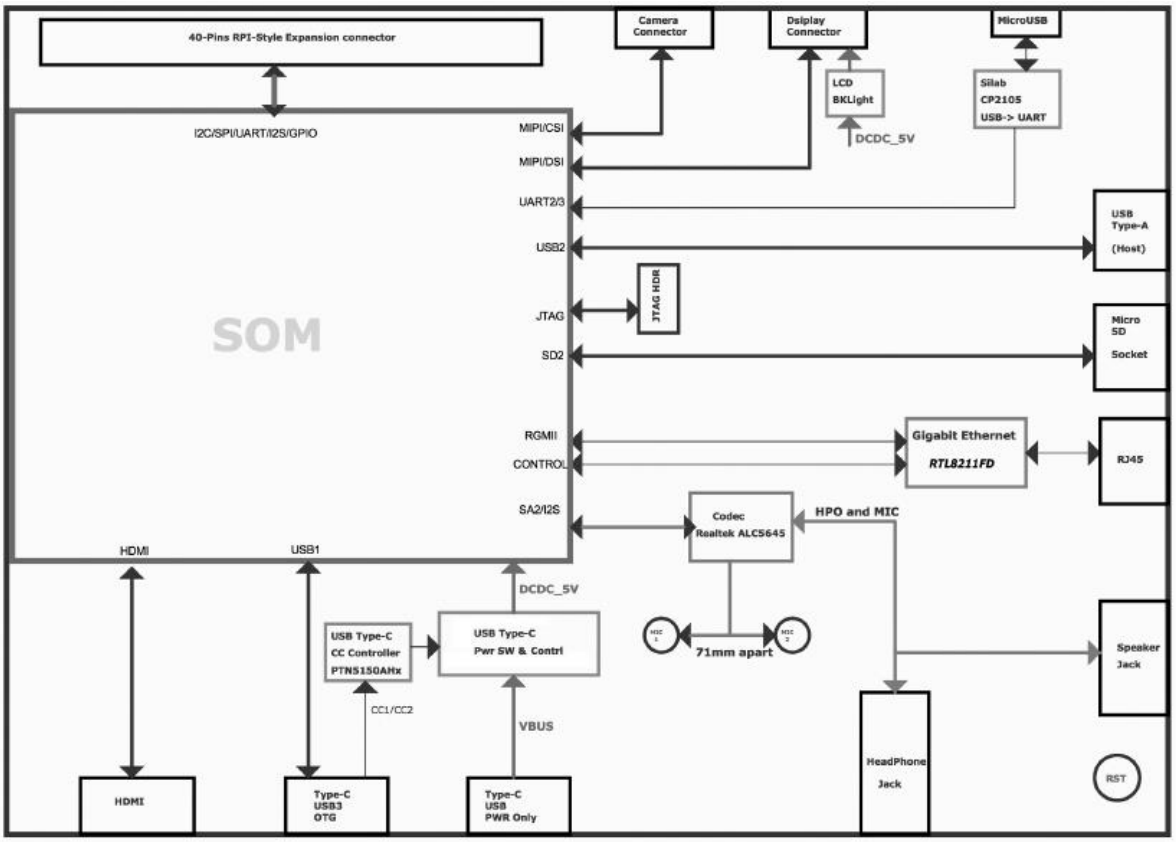

Figura 74. Diagrama de bloques de los componentes de la placa base (Dev Board) [36].

Anexo 2:

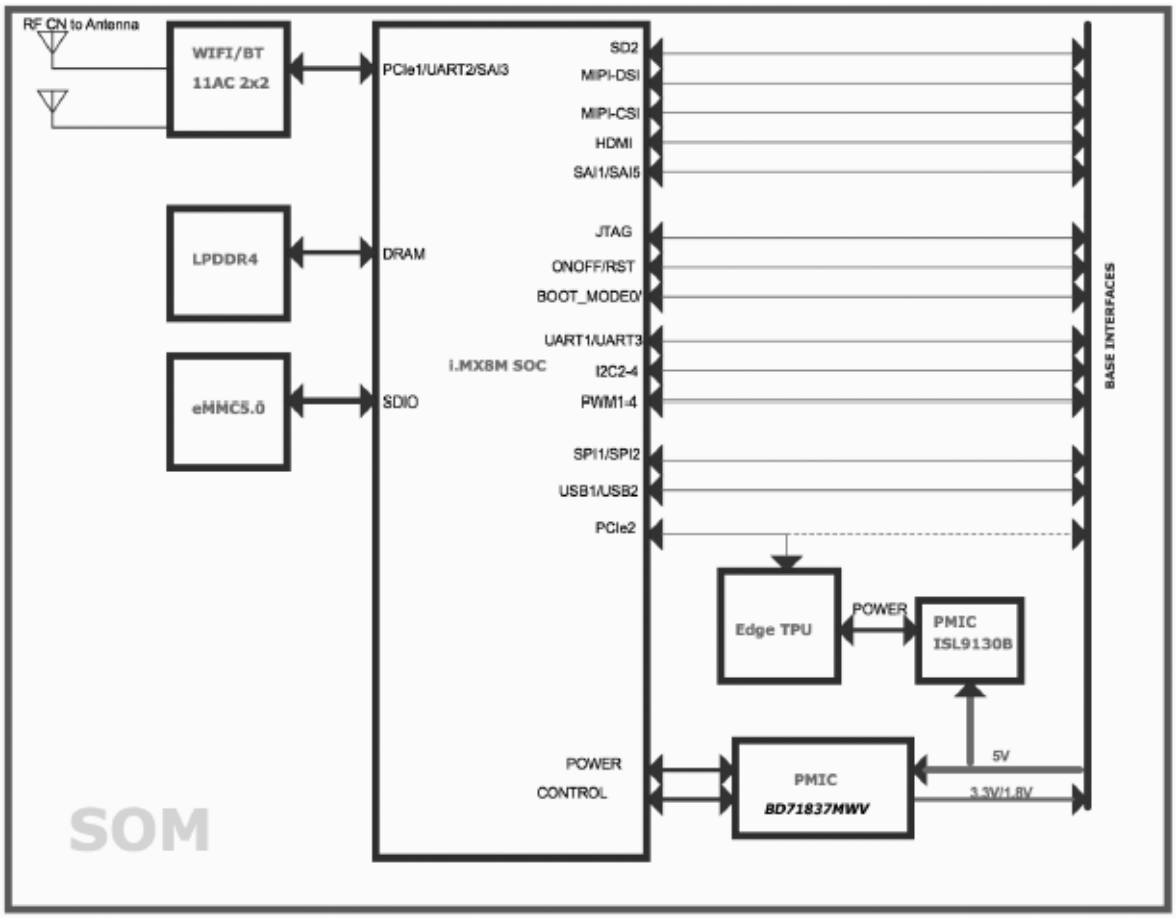

Figura 75. Diagrama de bloque de los componentes de SOM (Dev Board) [36].

Anexo 3:

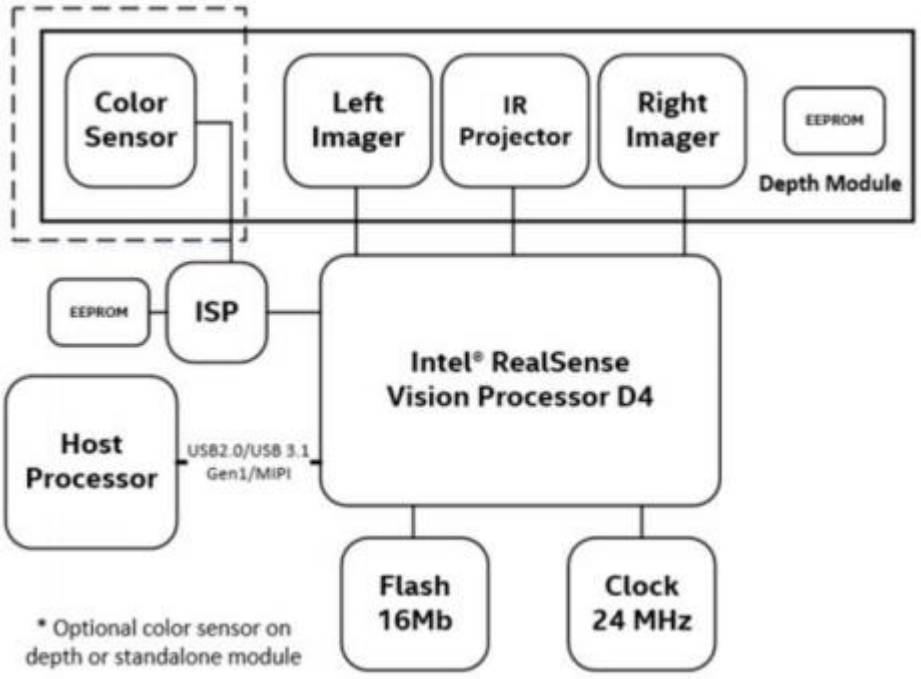

Figura 76. Arquitectura de la cámara Intel RealSense [37].