Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería Mecatrónica

Programa de Licenciatura en Ingeniería Mecatrónica

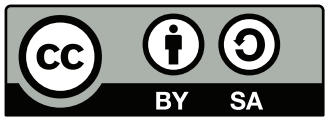
Detección y seguimiento de objetos mediante aprendizaje profundo para conducción autónoma

Informe de Trabajo Final de Graduación para optar por el título de

Ingeniero en Mecatrónica con el grado académico de Licenciatura

Fernando Flores Gómez

Cartago, 12 de Marzo, 2024



Declaro que el presente documento de tesis ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos y resultados experimentales propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de tesis realizado y por el contenido del presente documento.

Fernando Flores Gómez

Cartago, 19 de marzo de 2024

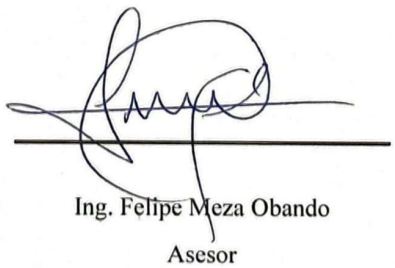
Céd: 3-0495-0294

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

El profesor asesor del presente trabajo final de graduación, indica que el documento presentado por el estudiante cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica del Instituto Tecnológico de Costa Rica para ser defendido ante el jurado evaluador, como requisito final para aprobar el curso Proyecto Final de Graduación y optar así por el título de Ingeniero(a) en Mecatrónica, con el grado académico de Licenciatura.

Estudiante: Fernando Flores Gómez

Proyecto: Detección y seguimiento de objetos mediante aprendizaje profundo para conducción autónoma



Ing. Felipe Meza Obando
Asesor

Cartago, 12 de Marzo, 2024

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

Proyecto final de graduación defendido ante el presente jurado evaluador como requisito para optar por el título de Ingeniero(a) en Mecatrónica con el grado académico de Licenciatura, según lo establecido por el programa de Licenciatura en Ingeniería Mecatrónica, del Instituto Tecnológico de Costa Rica.

Estudiante: Fernando Flores Gómez

Proyecto: Detección y seguimiento de objetos mediante aprendizaje profundo para conducción autónoma

Miembros del jurado evaluador

Ing. Ana María Murillo Morgan	Msc. Ing. Juan Carlos Brenes Torres
Jurado	Jurado

Los miembros de este jurado dan fe de que el presente proyecto final de graduación ha sido aprobado y cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica.

Cartago, 12 de Marzo, 2024

Resumen

El laboratorio del ICAI de la Universidad de Málaga, se dedica a realizar soluciones con implementación de Inteligencia Artificial, para distintos sectores.

El siguiente documento se detalla la metodología de diseño, aplicando métodos ingenieriles, el cual permita crear un sistema de frenado asistido, donde su accionamiento dependerá de la detección y seguimiento de objetos por medio de imágenes procesadas por medio de un modelo de aprendizaje profundo.

Este modelo generara una salida la cual permitirá que un sistema de control analice los datos y a partir de estos actúa acorde a la situación.

Además, se realizan pruebas de funcionamiento en distintos ambientes, los cuales permitan verificar el funcionamiento del sistema en ambientes no controlados. Validando la implementación del sistema a un posible mecanismo de asistencia de frenado en un automóvil autónomo.

Palabras clave: Aprendizaje profundo, Detección de objetos, LiDAR, Conducción autónoma, Inteligencia Artificial

Abstract

The ICAI laboratory at the University of Malaga is dedicated to developing solutions with the implementation of Artificial Intelligence for various sectors.

The following document details the design methodology, applying engineering methods, to create an assisted braking system where activation depends on the detection and tracking of objects through images processed by a deep learning model.

This model will generate an output that allows a control system to analyze the data and act accordingly to the situation.

In addition, operational tests are conducted in different environments to verify the system's performance in uncontrolled settings, validating the implementation of the system as a potential braking assistance mechanism in an autonomous vehicle.

Keywords: Deep Learning, Object detection, LiDAR, Autonomous Drive, Artificial Intelligence

a mi familia y amigos

Agradecimientos

El resultado de este trabajo no hubiese sido posible sin el apoyo de mi familia, amigos y profesores. Agradezco a mi madre por todo el apoyo incondicional y cariño que me ha brindado en mis años de vida.

Además quiero agradecer a compañeros de carrera, sin los cuales no hubiese podido disfrutar mis estudios universitarios, Jasson, Carlos y Randall, gracias a ellos logre hacer proyectos muy chivas y de calidad. Me apoyaron tanto en los buenos como malos momentos.

También quiero agradecer al Laboratorio ICAI y el profesor Ezequiel López quienes me acogieron para la realización de este proyecto, además de guiarme y ayudarme cuando lo necesitara.

A mi universidad y profesor asesor Felipe por la atención, apoyo y guía durante la realización de este largo proceso llamado proyecto de graduación.

Por ultimo, quiero agradecer a mis amigos en Málaga, sin ellos no hubiese sido posible adaptarme a otro país ni disfrutar tanto de este, gracias José Tomas, Marta, Alberto, Andres y Alex por su constante apoyo.

Fernando Flores Gómez

Cartago, 12 de Marzo, 2024

Índice general

Índice de figuras IV

Índice de tablas VII

1. Introducción 1
- 1.1. Entorno del proyecto . . . 1
- 1.2. Definición del problema . . . 1
- 1.3. Justificación . . . 2
- 1.4. Síntesis del problema . . . 2
- 1.5. Objetivos . . . 3
  - 1.5.1. Objetivo General . . . 3
  - 1.5.2. Objetivos Específicos . . . 3
- 1.6. Estructura del documento . . . 3
- 1.7. Aporte de ingeniería . . . 4

2. Marco teórico 5
- 2.1. Conducción autónoma . . . 5
- 2.2. Sistemas de visión artificial . . . 6
- 2.3. Inteligencia artificial . . . 7
  - 2.3.1. Redes neuronales . . . 8
  - 2.3.2. Redes neuronales convolucionales . . . 12
  - 2.3.3. Aprendizaje Profundo . . . 13
- 2.4. Segmentación de Imágenes . . . 14
- 2.5. Sistemas sensoriales en vehículos . . . 14
  - 2.5.1. Sistemas LiDAR . . . 14

3. Metodología de diseño 15
- 3.1. Metodología Ulrich-Eppinger . . . 15
- 3.2. Identificación del problema . . . 16
- 3.3. Identificación de necesidades . . . 18
- 3.4. Establecimiento de especificaciones objetivo . . . 19
- 3.5. Generación de conceptos . . . 20
  - 3.5.1. Descomposición funcional del problema . . . 20
  - 3.5.2. Búsqueda interna y externa de conceptos . . . 23

3.6. Selección de conceptos . . . 28
3.6.1. Filtrado . . . 29
3.6.2. Evaluación . . . 30
3.7. Concepto escogido . . . 31

4. Propuesta de diseño **32**
4.1. Segmentación y detección de objetos con Complex YOLOv4 . . . 32
4.1.1. Obtención de los datos . . . 33
4.1.2. Conversión de datos LiDAR en imágenes . . . 33
4.1.3. Arquitectura . . . 34
4.2. Calculo de distancia . . . 36
4.3. Control . . . 37
4.3.1. Normativas y legislación Española en conducción autónoma . . . 37
4.3.2. Calculo de velocidad a partir de la distancia al objeto. . . . 39
4.3.3. Control de velocidad actual del vehículo . . . 40

5. Resultados y análisis **41**
5.1. Pruebas de validación y resultados . . . 44
5.1.1. Prueba 1: Capacidad de detectar objetos vehiculares . . . 45
5.1.2. Prueba 2: Capacidad de segmentar . . . 46
5.1.3. Prueba 3: Tiempo de inferencia del modelo . . . 46
5.1.4. Prueba 4: Capacidad de medir la distancia de un objeto detectado . 47
5.1.5. Prueba 5: Integración con sistemas LiDAR . . . 48
5.1.6. Prueba 6: Capacidad de actuar al detectar un objeto . . . 49
5.1.7. Prueba 7: Tiempo de acción de parar, mantener o disminuir velocidad 51
5.1.8. Prueba 8: Tiempo total del sistema . . . 53
5.2. Análisis de resultados . . . 54
5.2.1. Prueba 1 . . . 54
5.2.2. Prueba 2 . . . 54
5.2.3. Prueba 3 . . . 55
5.2.4. Prueba 4 . . . 55
5.2.5. Prueba 5 . . . 56
5.2.6. Prueba 6 . . . 56
5.2.7. Prueba 7 . . . 56
5.2.8. Prueba 8 . . . 57
5.2.9. Análisis de casos específicos . . . 57
5.3. Análisis de económico y social . . . 63
5.3.1. Análisis económico . . . 63
5.3.2. Beneficios Académicos . . . 64
5.3.3. Beneficios económicos . . . 64
5.3.4. Beneficios sociales . . . 65

6. Conclusiones **67**

7. Recomendaciones **69**

Bibliografía **70**

A. Métricas de resultados **73**

B. Resultados de casos **79**

C. Configuración ambiente de trabajo **87**

D. Configuración Complex-YOLOv4 **89**

E. Código Complex-YOLOv4 **92**

F. Código Calculo de Distancia **103**

G. Código Control Lineal **105**

H. Código final del sistema. **106**

Índice de figuras

2.1. Niveles de autonomía en coches [1]. 6
2.2. Neurona y Perceptrón [5] 8
2.3. Algunas funciones de activación [6]. 8
2.4. Red Neuronal [6] 9
2.5. Propagación hacia adelante [8]. 10
2.6. Back Propagation [9]. 12

3.1. Fases metodología Ulrich-Eppinger [15]. 15
3.2. Descomposición del problema. 20
3.3. Fase de segmentación. 20
3.4. Segmentación de una imagen [17]. 21
3.5. Fase de detección. 21
3.6. Detección de un objeto. 21
3.7. Fase de cálculo de distancia. 22
3.8. Fase control. 22
3.9. Arquitectura de U-Net [19]. 23
3.10. Modelo R-CNN [20]. 24
3.11. Modelo YOLO [21]. 24
3.12. Modelo ResNet [22]. 24
3.13. Detección con máscaras [23]. 25
3.14. Calculo de distancia a partir de píxeles en una imagen [24]. 25
3.15. Método de triangulación [25]. 26
3.16. Control PID [26]. 26
3.17. Interpolación Lineal [27]. 27
3.18. Comparación en modelos de segmentación [28]. 28
3.19. Comparación en modelos de clasificación [29]. 28

4.1. Pipeline modelo Complex YOLOv4 [30]. 33
4.2. Características de nube de puntos para sistema LiDAR [30]. 34
4.3. Arquitectura de modelo Complex YOLOv4 [30]. 34
4.4. Dimensiones de cuadro delimitador [30]. 35
4.5. Proyección de cuadros delimitador de imagen 2D a 3D. 36
4.6. Calculo de distancia entre dos puntos [31]. 36
4.7. Tabla de valores de velocidad y distancia según legislación Española [32]. . 37

4.8. Gráfica con valores de distancia mínima de seguridad y velocidad actual del vehículo. . . . 38
4.9. Árbol de toma de decisiones para el control de velocidad de vehículo. . . . 39

5.1. Relación entre numero de objetos detectado y el tiempo de procesamiento. 57
5.2. Resultados imagen DS11. . . . 58
5.3. Casos de disminución de velocidad. . . . 59
5.4. Resultados imagen DS53. . . . 60
5.5. Resultados imagen DS42. . . . 61
5.6. Resultados imagen DS61. . . . 61
5.7. Resultados imagen DS63. . . . 62
5.8. Tabla de presupuesto del proyecto. . . . 63

A.1. Matriz de confusión conjuntos de datos 1. . . . 73
A.2. Métricas conjuntos de datos 1. . . . 73
A.3. Sensibilidad y especificidad conjuntos de datos 1. . . . 74
A.4. Matriz de confusión conjuntos de datos 2. . . . 74
A.5. Métricas conjuntos de datos 2. . . . 74
A.6. Sensibilidad y especificidad conjuntos de datos 2. . . . 74
A.7. Matriz de confusión conjuntos de datos 3. . . . 75
A.8. Métricas conjuntos de datos 3. . . . 75
A.9. Sensibilidad y especificidad conjuntos de datos 3. . . . 75
A.10. Matriz de confusión conjuntos de datos 4. . . . 75
A.11. Métricas conjuntos de datos 4. . . . 76
A.12. Sensibilidad y especificidad conjuntos de datos 4. . . . 76
A.13. Matriz de confusión conjuntos de datos 5. . . . 76
A.14. Métricas conjuntos de datos 5. . . . 76
A.15. Sensibilidad y especificidad conjuntos de datos 5. . . . 77
A.16. Matriz de confusión conjuntos de datos 6 77
A.17. Métricas conjuntos de datos 6. . . . 77
A.18. Sensibilidad y especificidad conjuntos de datos 6. . . . 77
A.19. Matriz de confusión conjuntos de datos 7. . . . 78
A.20. Métricas conjuntos de datos 7. . . . 78
A.21. Sensibilidad y especificidad conjuntos de datos 7. . . . 78

B.1. Resultados imagen DS11. . . . 79
B.2. Resultados imagen DS12. . . . 79
B.3. Resultados imagen DS13. . . . 80
B.4. Resultados imagen DS21. . . . 80
B.5. Resultados imagen DS22. . . . 80
B.6. Resultados imagen DS23. . . . 81
B.7. Resultados imagen DS41. . . . 81
B.8. Resultados imagen DS42. . . . 82
B.9. Resultados imagen DS43. . . . 82

B.10.Resultados imagen DS51. 83
B.11.Resultados imagen DS52. 83
B.12.Resultados imagen DS53. 84
B.13.Resultados imagen DS61. 84
B.14.Resultados imagen DS62. 84
B.15.Resultados imagen DS63. 85
B.16.Resultados imagen DS71. 85
B.17.Resultados imagen DS72. 85
B.18.Resultados imagen DS73. 86

C.1. Bibliotecas utilizadas lista 1. 87
C.2. Bibliotecas utilizadas lista 2. 88

D.1. Configuración paso 1. 89
D.2. Configuración paso 2. 89
D.3. Configuración paso 3. 90
D.4. Configuración paso 4. 91

E.1. Complex-YOLOv4 código 1. 92
E.2. Complex-YOLOv4 código 2. 93
E.3. Complex-YOLOv4 código 3. 94
E.4. Complex-YOLOv4 código 4. 95
E.5. Complex-YOLOv4 código 5. 96
E.6. Complex-YOLOv4 código 6. 97
E.7. Complex-YOLOv4 código 7. 97
E.8. Complex-YOLOv4 código 8. 98
E.9. Complex-YOLOv4 código 9. 99
E.10.Complex-YOLOv4 código 10. 100
E.11.Complex-YOLOv4 código 11. 101
E.12.Complex-YOLOv4 reultados de sensibilidad y especificidad. 102

F.1. Código de calculo de distancia. 103
F.2. Resultado de valores de calculo de distancia para cada objeto. 104
F.3. Resultado final de calculo de distancia. 104

G.1. Código árbol de decisiones control lineal. 105

H.1. Código final parte 1. 106
H.2. Código final parte 2. 107
H.3. Código final resultado. 107

Índice de tablas

3.1. Identificación del problema raíz 16
3.2. Necesidades del problema 18
3.3. Especificaciones del problema 19
3.4. Identificación del problema raíz 23
3.5. Conceptos generados 27
3.6. Filtrado de conceptos combinados 29
3.7. Evaluación de conceptos 30
3.8. Conceptos generados 31

5.1. Características de conjunto de datos 41
5.2. Imágenes seleccionadas para resultados finales 42
5.3. Factores de prueba del sistema objetivo 1. 44
5.4. Sensibilidad y Especificidad. 45
5.5. Métrica IoU de segmentación 46
5.6. Tiempos de inferencia en diferentes rutas 46
5.7. Error medio en diferentes rutas 47
5.8. Factores de prueba del sistema objetivo 2. 48
5.9. Tiempos de inferencia en diferentes rutas 48
5.10. Factores de prueba del sistema objetivo 3. 49
5.11. Tiempos total del sistema. 50
5.12. Tiempos de acción del control. 51
5.13. Factores de prueba del sistema objetivo 4. 52
5.14. Tiempos total del sistema. 53

Lista de abreviaciones

Abreviaciones

CNN	Convolutional Neural Network
DL	Deep Learning
E-RPN	Euler-Region-Proposal
ICAI	Laboratorio de Inteligencia Computacional y Análisis de imágenes
IoU	Intersection over Union
ITCR	Instituto Tecnológico de Costa Rica
LiDAR	Light Detection and Ranging
ML	Machine Learning
ms	milisegundos
PID	Proporcional, Integral y Derivativo
R-CNN	Region-based Convolutional Neural Network
ResNet	Residual Network
U-Net	U-shape Network
UMA	Universidad de Málaga
YOLO	You Only Look Once

Capítulo 1

Introducción

1.1. Entorno del proyecto

El grupo de investigación de Inteligencia Computacional y Análisis de Imágenes (ICAI) de la Universidad de Málaga, España. El cual fue fundado desde hace 30 años por el catedrático José Muñoz, un pionero español en el estudio de redes neuronales, tiene como propósito el desarrollo e investigación de la inteligencia artificial en distintas áreas. Como aprendizaje computacional y visión por computadora, bussines intelligence, sistemas inteligentes para la toma de decisiones, modelado biológico, vida y creatividad artificiales.

Actualmente liderado por el profesor Ezequiel López, es hoy uno de los referentes en neuro computación, aprendizaje profundo e inteligencia artificial en España. Además de la investigación y desarrollo académico también funge como un grupo atento a nuevas demandas sociales, en una cultura cada vez más tecnológica, aplicando los nuevos conocimientos desarrollando herramientas útiles para las personas.

1.2. Definición del problema

La motivación para desarrollar este proyecto nace del avance exponencial en el mundo tecnológico, cada vez más dependiente de los datos digitales y sistemas de inteligencia artificial. Dentro de estos sectores tecnológicos se encuentra la industria automovilística la cual durante años ha ido incorporando nuevas tecnologías a los autos para crear una conducción más segura y eficiente.

Gracias a estos avances los automóviles de las últimas décadas incorporan cada vez más sistemas tecnológicos como sensores, cámaras y computadoras. Debido a esto han surgido dos grandes áreas en la industria de autos, las cuales son asistencia inteligente en la conducción y sistemas de conducción automática.

Uno de los temas de investigación por parte de grupo ICAI es la visión por computadora y

detección de objetos por medio de aprendizaje profundo, el cual puede ser implementado en distintos sistemas, como lo puede ser una la conducción automática.

1.3. Justificación

En un mundo donde prevalece el desarrollo tecnológico y este avanza acelerando cada vez más, es imperativo para el sector académico e industrial el mantener un el ritmo por medio de la investigación y desarrollo de nuevas tecnologías para estas nuevas demandas. Una de las que más ha avanzado en este sentido es la digitalización y procesamiento de datos digitales, dentro de este ámbito se encuentran las imágenes y visión por computadora.

Debido al desarrollo de mejores cámaras y algoritmos se ha podido avanzar a pasos agigantados los algoritmos de aprendizaje profundo y redes convolucionales, permitiendo herramientas tales como detección y seguimiento de objetos, segmentación de imágenes, clasificación de imágenes y estimación de poses.

Esto combinado a otras tecnologías como la electrónica, robótica, biotecnología, por nombrar algunas, ha llevado a la humanidad a una nueva era, la cual cada vez los sistemas son más inteligentes y pueden realizar más funciones, de manera más eficiente, permitiendo crear nuevas áreas de investigación y desarrollo, además de incorporarse cada vez más al diario vivir.

Gracias a estas tecnologías, la humanidad está realizando un esfuerza cada vez más grande en generar el alimento de esta nueva tecnología, la cual es información digital, esto al recolectar cada vez más información con sensores electrónicos, generando más y mejores conjuntos de datos, digitalizar la información ya existente, entre muchos otros.

Debido a este es posible el desarrollar sistemas inteligentes de conducción que cuentan con cámaras y sensores acoplados a los autos. Uno de los pioneros en llevar la conducción automática al mercado fue Tesla Motors, una empresa de automóviles eléctricos fundada por Martin Eberhard y Marc Tarpenning, posteriormente liderada por el cofundador Elon Musk.

A raíz de lo anteriormente mencionado, como parte de los proyectos de investigación y desarrollo el grupo ICAI busca desarrollar un sistema de visión el cual sea capaz de detectar y seguir objetos en tiempo real por medio de herramientas de aprendizaje profundo, el cual presente dos factores claves, una buena precisión y una alta velocidad.

1.4. Síntesis del problema

Necesidad de construir un modelo de aprendizaje profundo para la conducción autónoma que detecte y siga objetos en tiempo real.

1.5. Objetivos

1.5.1. Objetivo General

Desarrollar un sistema de detección y segmentación de objetos por medio de métodos de aprendizaje profundo para controlar un sistema de asistencia de frenado en un vehículo autónomo.

1.5.2. Objetivos Específicos

1. **Desarrollo de Modelos de Detección y Segmentación:** Crear y entrenar modelos de detección de objetos y segmentación de objetos que sean precisos y eficientes en la identificación y clasificación de obstáculos en el entorno del vehículo.
2. **Integración de Datos Imágenes y LiDAR (Light Detection And Ranging):** Integrar de imágenes y datos LiDAR, en un sistema de fusión sensorial que proporcione una visión completa de la escena circundante, lo que permitirá una detección más robusta.
3. **Desarrollo de Algoritmos de Toma de Decisiones:** Diseñar algoritmos de toma de decisiones que utilicen la información de detección y segmentación para determinar cuándo y cómo aplicar la asistencia de frenado de manera segura y eficiente.
4. **Validación y Pruebas en Condiciones Simuladas:** Realizar pruebas exhaustivas en condiciones simuladas para evaluar el rendimiento del sistema.

1.6. Estructura del documento

Se muestra a continuación los capítulos que conforman el documento, además de una breve descripción de sus contenidos.

- Capítulo 1 - Introducción: Esta sección brinda al lector el contexto sobre la problemática y los objetivos del presente documento.
- Capítulo 2 - Marco Teórico: En este apartado se da al lector el conocimiento técnico necesario para un entendimiento posterior de conceptos, formulas y técnicas, posteriormente utilizadas en la solución.
- Capítulo 3 - Metodología: En este capitulo se explica paso a paso la metodología implementada para logra diseñar la mejor solución posible.
- Capítulo 4 - Propuesta de Diseño: En este siguiente sección se explica y detalla los aspectos técnicos necesarios para la solución propuesta.

- Capítulo 5 - Resultados y Análisis: Para esta sección se muestran los resultados obtenidos, además se analizan estos para evaluar las métricas y el logro de los objetivos propuestos.
- Capítulo 6 - Conclusiones: Resume todos los puntos importantes del documento y si se lograron los objetivos planteados inicialmente.
- Capítulo 7 - Recomendaciones: Indica recomendaciones posteriores a la realización del proyecto los cuales permita mejoras en una posterior aplicación del proyecto.

1.7. Aporte de ingeniería

El aporte de ingeniería que abarca proyecto consiste en el desarrollo de una optimización de una solución a un problema propuesto. La solución propuesta contempla las siguientes áreas de la ingeniería Mecatrónica, siendo estas la Computación, Control Automático e Inteligencia Artificial.

Capítulo 2

Marco teórico

En este capítulo se explora los fundamentos teóricos del proyecto los cuales permitan entender los aspectos técnicos de la implementación de un sistema de detección y seguimiento de objetos mediante aprendizaje profundo para conducción autónoma, definiendo los conceptos básicos de conducción autónoma, sistemas de visión artificial, inteligencia artificial, métodos de detección y segmentación de objetos, sistemas sensoriales en vehículos y por último el estado del arte en este campo. A continuación, se procede a definir los aspectos teóricos.

2.1. Conducción autónoma

Tal como comenta [1], con el desarrollo tecnológico en los últimos años, cada vez se integran más sistemas inteligentes a la cotidianidad humana, entre esto se encuentran los automóviles. Esto a su vez se ha vuelto un tema de interés global y en datos de la organización mundial de la salud (OMS), cerca de 1,3 millones de personas mueren en accidentes de tráfico anualmente, esto sin contar lesiones o discapacitados, de esto la importancia que ha tomado la conducción segura.

Se define a la conducción autónoma como la capacidad de un sistema inteligente en imitar capacidades humanas de manejo y control, en este se definen 6 niveles de conducción autónoma, tal como se muestra en la Figura 2.1, según su nivel de autonomía.

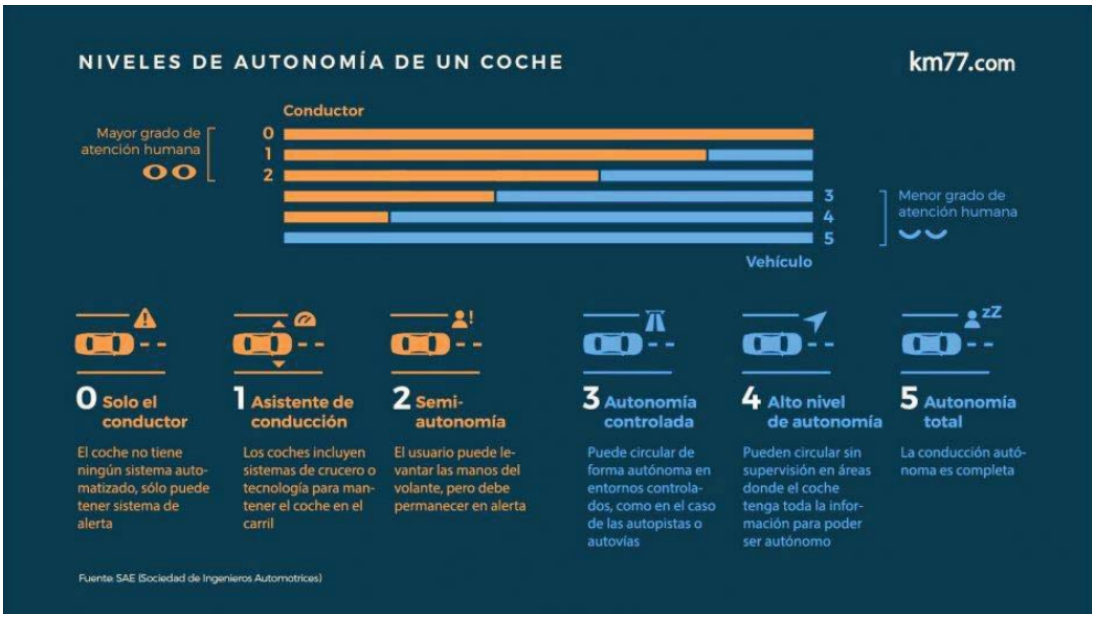


Figura 2.1: Niveles de autonomía en coches [1].

- Nivel 0: No existe autonomía, solamente el conductor controla el vehículo.
- Nivel 1: Existen sistemas de control no inteligente como control crucero.
- Nivel 2: Sistema semi-autónomo con ayuda del conductor, el conductor debe esta atento en todo momento y el sistema debe desactivarse cuando el conductor tome el control.
- Nivel 3: El vehículo puede circular de forma autónoma en entornos controlados y con supervisión, un ejemplo son los sistemas de Autopilot.
- Nivel 4: El vehículo puede circular sin supervisión en áreas acotadas.
- Nivel 5: Autonomía completa del vehículo, este puede circular en cualquier área permitida legalmente y gracias a su tecnología el vehículo puede reaccionar a cualquier eventualidad imprevista.

2.2. Sistemas de visión artificial

El concepto de visión artificial hace referencia a la interpretación del mundo físico por medio de cámaras en datos digitales los cuales pueden ser entendidos por sistemas de computación. Tal como indica [2], los sistemas de visión cumplen una serie de etapas, tales como:

1. Sensorial: la cual consiste en la captura de imágenes.

2. Tratamiento o procesamiento de la imagen: esto puede ser cambios de tamaño, filtros y formatos.

3. Segmentación: consisten en la separación de la imagen en elementos que permitan comprender la información.

4. Reconocimiento: en este caso es la distinción de información de interés del resto de la imagen, por ejemplo, si se desea encontrar los contornos en la imagen y se ignora el resto.

Con el desarrollo actual en la computación, ha cambiado enormemente las etapas 3 y 4, dentro de estos desarrollos se tienen las Redes Neuronales Convolucionales (CNN, por sus siglas en ingles), tal como comenta [3], esto ha llevado a un cambio de paradigma en la visión artificial y sus alcances, dando como resultado dos grandes vertientes, la visión computacional tradicional y el aprendizaje profundo, se ahondará más respecto a este ultima en la siguiente sección.

La gran popularidad de los algoritmos de aprendizaje profundo, se ha debido a un problema fundamental encontrado en los sistemas tradicionales de visión computacional, ya que algunos algoritmos para realizar la etapa 3 y 4 pueden llegar a ser muy complicados, indica [3], si se realiza una segmentación de las imágenes por medio de métodos tradicionales, estos necesitan distintos tipos de algoritmos para reconocer los objetos y pueden variar de gran forma los resultados dependiendo del ambiente en el cual ha sido tomada la imagen.

Otro de los inconvenientes se muestra en la etapa 4, si se quiere identificar o clasificar un objeto, es necesario definir cuantas características del objeto es necesario discriminar para poder identificar el objeto y cuanto dependen estas de la calidad de la imagen.

Gracias a estas limitaciones el aprendizaje profundo ha logrado sobresalir en estas áreas, donde a pesar de ser un proceso más pesado, tal y como demuestra [3], suele ser más robusto y confiables que las técnicas tradicionales de visión computacional. Es importante aclarar, esto no significa que los métodos tradicionales hayan perdido utilidad, para tareas simples y de procesamiento liviano estas suelen ser muy utilizadas, también se han combinado al aprendizaje profundo, para solucionar distintos tipos de problemas.

2.3. Inteligencia artificial

Desde la antigüedad el ser humano a intentado emular las capacidades físicas que poseen los seres vivos en objetos inanimado, ejemplo de esto son los mecanismos, los cuales realizan tareas repetitivas por medio de “músculos” mecánicos, como comenta [4], este mismo principio ha sido aplicado a la computación, “¿Es posible construir maquinas inteligentes? ¿Es el cerebro una maquina?”, a partir de estas preguntas se a llego a basar el estudio de la inteligencia artificial (IA). Siguiendo esta línea de pensamiento se procede a definir el mecanismo más fundamental del cerebro humano, la neurona.

2.3.1. Redes neuronales

Una de las bases fundamentales en el cerebro humano es la neurona, la cual es la neurona, la cual es una "célula nerviosa eléctricamente excitable e interconectadas dentro del cerebro que procesan y transmiten información a través de señales eléctricas y químicas" [5]. Basado en la biología se tiene su semejante a nivel computacional llamado perceptrón, tal y como se muestra en 2.2, al igual que su equivalente la función esencial de este es recibir una estimulación, procesar y dar un resultado.

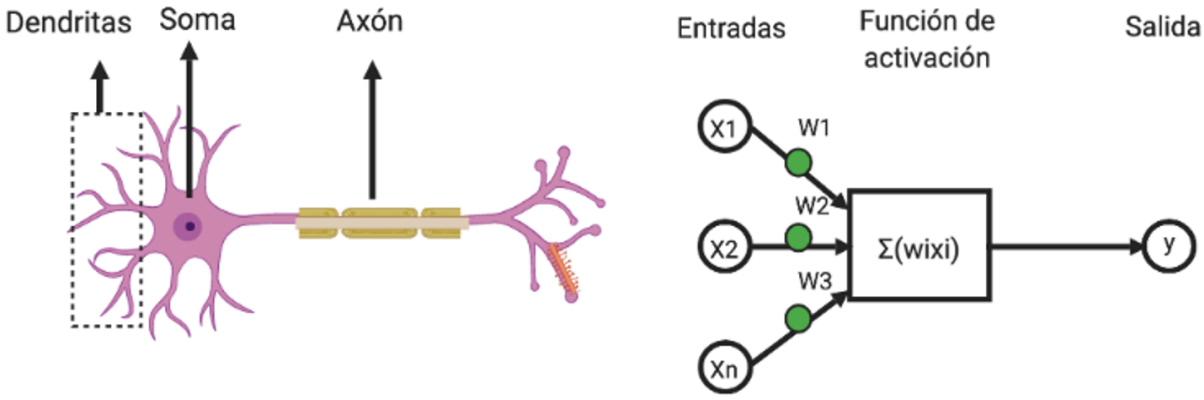


Figura 2.2: Neurona y Perceptrón [5]

Siguiendo el ejemplo biológico se desarrolla el siguiente proceso, señales a las cuales se le asigna un peso, como se muestra en la 2.2, estas se suman junto a un sesgo y dan como resultado un valor al cual se le aplica una función, llamada función de activación obteniendo el valor "y", escrito matemáticamente quedaría como la 2.1.

$$y = b + \sum_{i=1}^{n} W_i \cdot X_i \tag{2.1}$$

A partir de este ponderado se pueden aplicar distintos tipos de funciones de activación, a continuación, se muestran algunas en la siguiente 2.3:

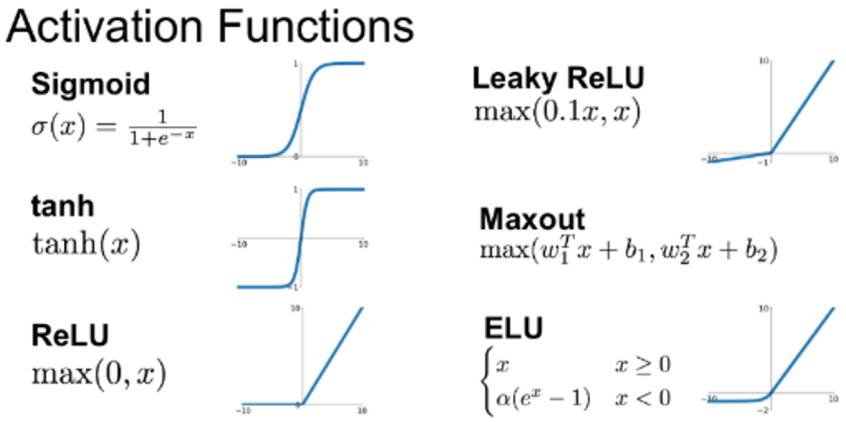


Figura 2.3: Algunas funciones de activación [6].

Siguiendo el sistema biológico, la neurona es la pieza fundamental para generar un proceso de pensamiento y toma de decisiones, esto lo realiza por medio de una interconexión entre neuronas, para el caso de la computación sería una red de perceptrones interconectados, como se muestra en 2.4.

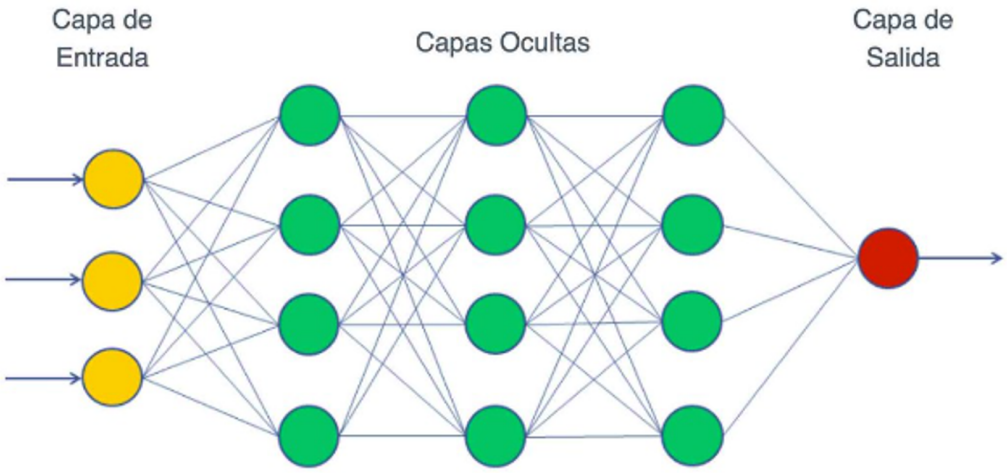


Figura 2.4: Red Neuronal [6]

Se muestra la topología o arquitectura de una red neuronal, donde se pueden destacar 3 grandes secciones, la capa de entrada, la cual conecta con las entradas o estimulaciones del sistema, posteriormente se tiene las capas ocultas las cuales son una red interconectada de perceptrones y por último se tiene la capa de salida, en esta estará el resultado del sistema.

Función de coste

Como todo sistema, es posible optimizar, para esto es necesario algún tipo de evaluación la cual permita conocer que tan lejos o cerca se encuentra el sistema del valor ideal. Tal como indica [7], las funciones de coste más comunes son:

- Raíz cuadrada media: $RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$
- Error absoluto medio: $MAE = \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{n}$
- Error absoluto medio escalado: $MASE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{\frac{n}{n-1} \sum_{i=2}^{n} |\hat{y}_n - y_{n-1}|}$
- Entropía cruzada categórica: $\mathcal{L}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} y_{ij} log(p_{ij})$
- Entropía cruzada binaria: $\mathcal{L}(\theta) = -\frac{1}{n} \sum_{i=1}^{n} y_i log(p_i + (i - y) log(1 - p_i))$

Su uso dependerá del problema y como este se desee abordar, entendiendo el problema apropiadamente se escogerá una función de coste adecuada a cada uno de los problemas.

Optimización

Habiendo definido la función de coste o perdida, se procede a escoger alguna de las funciones de optimización, esta función se aplicará de forma iterativa buscando el valor óptimo el cual disminuya el valor de la función de coste, como comenta [7] entre estos optimizadores se cuentan:

- Descenso estocástico del gradiente: Esta es una aproximación estocástica del gradiente descendiente, el cual busca minimizar funciones objetivas, por medio de iteraciones se optimiza la función diferencial hasta encontrar sus máximos o mínimos.
- Adam: Busca fijar el ratio de aprendizaje de forma adaptativa aumentando el valor de ratio si sus parámetros se encuentran muy dispersos y disminuyéndolo cuando estos se acercan.
- Adagrad: Este algoritmo se basa en la gradiente, aplicando grandes actualizaciones al ratio cuando los parámetros son poco frecuentes y pequeñas actualizaciones si son muy frecuentes.
- Adadelta: Algoritmo extensión de Adagrad.

Propagado hacia adelante (Forward Propagation)

Según [8], este método es la suma ponderada de cada uno de los perceptrones de cada una de sus capas, a continuación, se presenta un ejemplo a partir de la Figura 2.5.

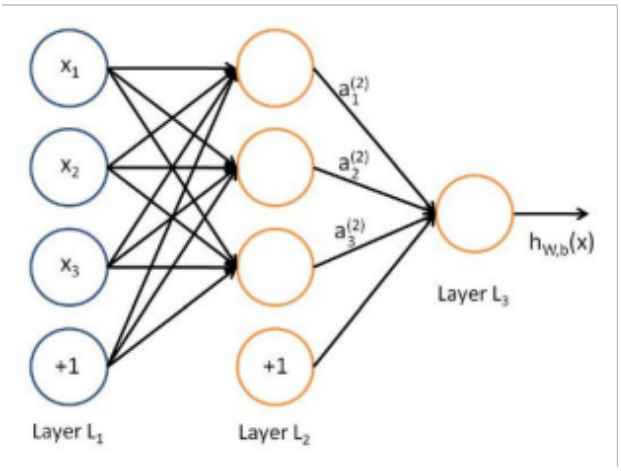


Figura 2.5: Propagación hacia adelante [8].

Donde se denotara al numero de capas como $n_l = 3$ y la capa L_l, donde $l = 1,..,n_l$. La red cuenta con los siguientes parámetros:

- W_{ij}^{l} es el peso asociado a la conexion entre el perceptron j de la capa l y el perceptron j de la capa siguiente $l+1$.
- $W^{(l)}$ representa la matriz de pesos entre los perceptrones de la capa l y $l+1$.
- a_i^l es la salida o activacion del perceptron en la capa l.
- b_i^l corresponde al sesgo asociado al perceptron i en la capa $l+1$
- s_l es el numero de perceptrones en la capa l.

Las siguientes ecuaciones 2.2 ejemplifican la red mostrada en la Figura 2.5.

$$\begin{aligned} a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\ a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\ a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\ h_{W,b}(x) = &a_1^{(3)} = f(W_{11}^{(2)}a_1^{(2)} + W_{22}^{(2)}a_2^{(2)} + W_{33}^{(2)}a_3^{(2)} + b_1^{(1)}) \end{aligned} \tag{2.2}$$

En resumen, se puede simplificar la ecuación 2.2 como la suma ponderada de las entradas y sesgo para i, $h_{W,b}(x) = f(W^T\overrightarrow{x}) = f(\sum_{i=1}^{3} W_i x_i + b)$.

Propagado hacia atrás (Backward Propagation)

Tal como describe [9], esta técnica permite cambiar los pesos de las conexiones, esto lo logra por medio de minimizar el error en la función de coste C mostrada en 2.3 y coste total 2.4, a partir de esta función de coste se podrá minimizar su error por medio del algoritmo de descenso del gradiente, es importante aclarar que como su nombre indica la propagación hacia atrás inicia desde la última capa y propaga sus errores hacia atrás, como se observa en la Figura 2.6.

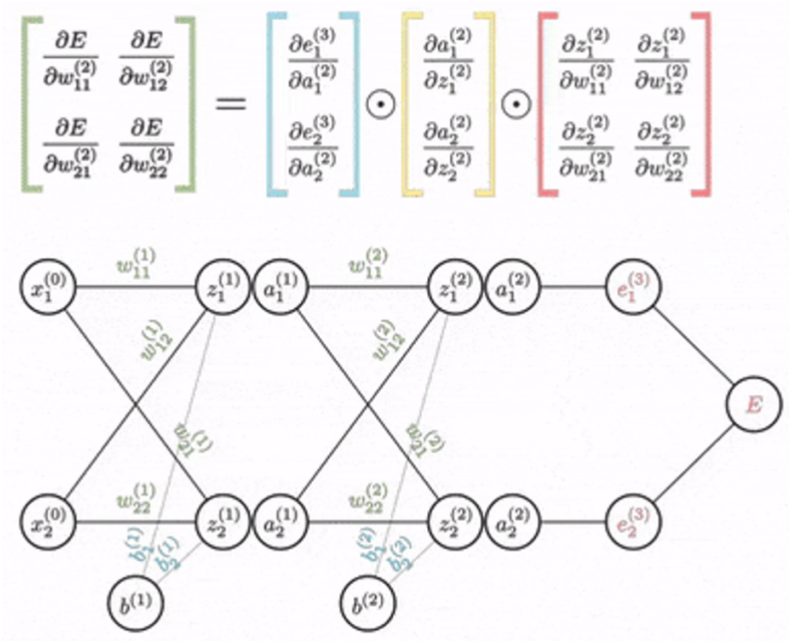


Figura 2.6: Back Propagation [9].

$$C(W, b; x^{(i),y^{(i)}}) = \frac{1}{2}||h_{W,b}(x^{(i)}) - y^{(i)}||^2 \tag{2.3}$$

$$C(W, b) = \left[\frac{1}{m}\sum_{i=1}^{m}(\frac{1}{2}||h_{W,b}(x^{(i)}) - y^{(i)}||^2)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(W_{ij}^{(l)})^2 \tag{2.4}$$

Actualización de pesos

Como indica [8], a partir de los errores calculados se puede calcular las derivadas parciales, o en otras palabras las gradientes de W y b, con respecto a la función de coste C como se muestra en 2.5, de esta forma se pueden actualizar el valor de los pesos W y sesgos b, al ser un proceso computacional se puede aplicar iterativamente hasta minimizar el error y de esta forma la red neuronal podrá adaptarse al problema que busca resolver.

$$\begin{aligned} W_{ij}^{l} &= W_{ij}^{l} - \alpha\frac{\partial}{\partial W_{ij}^{l}}C(W, b) \\ b_i^{(i)} &= b_i^{(i)} - \alpha\frac{\partial}{\partial b_i^{(i)}}C(W, b) \end{aligned} \tag{2.5}$$

2.3.2. Redes neuronales convolucionales

Las CNN (Convolutional Neural Network) consisten en un tipo específico de arquitectura de redes neuronales definidas para trabajar y analizar datos de tipo matriz. Ejemplos de

datos tipo matriz consisten en imágenes, audio, texto y tabulaciones.

Estas redes se valen de capas de convolución para la detección de patrones locales y características comunes. Una aplicación de este tipo de redes son los sistemas de visión, donde son empleadas para reconocimiento de imágenes, clasificación y segmentación.

Una red convolucional está compuesto por tres tipos de capas: convolucional, agrupación y totalmente conectada [10]. A continuación, se detalla la función de cada capa por separado.

- Capa convolucional: corresponde al bloque principal donde se realizan la mayoría de los cálculos. Esta capa posee tres componentes: un filtro, un mapa de características y datos de entrada. El filtro se aplica a la imagen y este recorre la totalidad de la matriz convolucionando la imagen con una matriz de pesos de la imagen. Esto permite obtener un mapa de características de la imagen.
- Capa de Agrupación: la función de esta capa consiste en reducir el tamaño de los parámetros de entrada de la imagen. Para ello, aplica un filtro, que al igual que en la capa convolucional consiste en una matriz, pero en este caso sin pesos. Esta matriz calcula el valor promedio de un rango operacional de la imagen y con esta información llena una matriz de salida, reduciendo el tamaño inicial de parámetros.
- Capa totalmente conectada: esta capa se encarga de realizar tareas de clasificación de los datos obtenidos en las capas anteriores. Esta capa utiliza funciones de activación para poder clasificar entradas y generar entradas con valores entre 0 y 1.

2.3.3. Aprendizaje Profundo

Corresponde a la traducción de Deep Learning. Consiste en un área del Machine Learning donde se enfoca en la utilización de redes neuronales artificiales para aprender y clasificar datos de manera jerárquica.

Este modelo de aprendizaje tiene la capacidad de reconocer patrones de imágenes, textos y sonidos con el fin de generar predicciones precisas [11].

Entre las aplicaciones del aprendizaje profundo resaltan los vehículos autónomos, sistemas de defensa, detección de patrones en imágenes médicas entre otros.

Esta área de conocimiento se enfoca en optimizar el proceso de aprendizaje de las máquinas [12]. Generalmente se entregan datos sin procesar a un algoritmo y el sistema analiza los datos sin tomar en cuenta normas o características específicas. El sistema de aprendizaje profundo realiza predicciones con un grupo selecto de datos y verifica el nivel de precisión de las predicciones iterativamente.

2.4. Segmentación de Imágenes

Corresponde a la técnica que divide una imagen digital en regiones con características visuales comunes. Estas características pueden ser similitud de bordes, texturas, colores entre otras.

La segmentación se puede hacer en función del contenido de cada píxel (segmentación semántica), o según áreas locales de la imagen (contornos) [13].

2.5. Sistemas sensoriales en vehículos

2.5.1. Sistemas LiDAR

De las siglas Light Detection and Ranging, es un componente electrónico que emplea pulsos de láser para medir distancias y crear a su vez, mapas tridimensionales del ambiente al que se exponga.

Esta tecnología utiliza los mismos principios de un radar y un sonar. Esta analogía se sustenta que al igual que las mencionadas, el LiDAR emite energía de forma de luz reflejada en objetos y con esto puede calcular distancias y tamaños de objetos [14].

Los componentes principales de un sistema LiDAR son:

- Escáner láser: se encarga de la emisión de pulsos láser a alta frecuencia.
- Sensor LiDAR: sensor que detecta y recoge los pulsos de luz reflejados.
- Procesador: se encarga de realizar los cálculos matemáticos de tiempo y distancia. También construye el conjunto de datos resultante del sistema.

Capítulo 3

Metodología de diseño

3.1. Metodología Ulrich-Eppinger

El siguiente capitulo, describe las fases y procedimientos realizados para desarrollar una solución al problema, en el ámbito ingenieril existen distintas formas de enfrentar el diseño de una solución, en este proyecto se utilizará la metodología Ulrich-Eppinger, la cual se basa en desarrollar las soluciones por fases iterativas, con el propósito de lograr el mejor diseño posible. Debido a ser un proceso iterativo dividido en fases muestra una gran versatilidad y flexibilidad, las cuales encajan con el problema de este proyecto.

La metodología define las siguientes fases de desarrollo, mostradas en 3.1.

En los siguientes apartados se profundiza en el proceso de cada fase, desde la identificación del problema hasta la prueba de concepto. Es importante recalcar que esta es una metodología iterativa en cada fase y solo se muestra el resultado final de cada fase, además la metodología variara en relación con el problema y necesidades encontradas, por lo cual no siempre se aplicaran todas las fases que plantea la metodología.

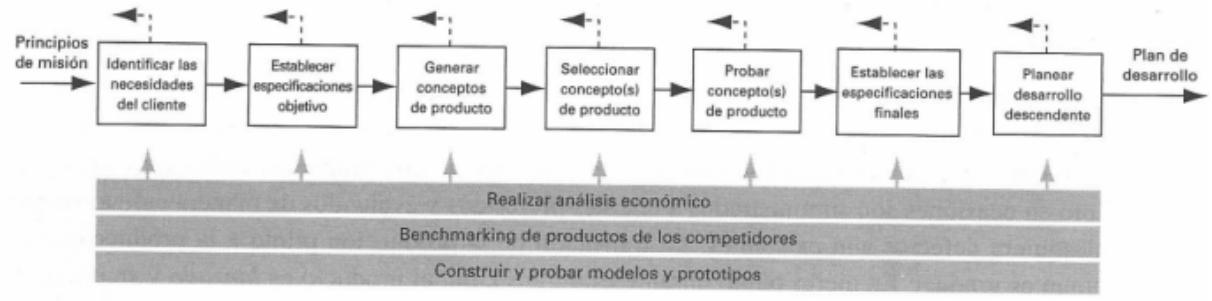


Figura 3.1: Fases metodología Ulrich-Eppinger [15].

3.2. Identificación del problema

En esta sección se busca encontrar el problema raíz y no solamente el "problema percibido", tal y como comenta [16], en ciertos casos existen dificultades al identificar un problema, ya que este se suele confundir con los síntomas y no lo que los genera. En este caso entra la experiencia y pericia del ingeniero en distinguir entre estos dos, el problema real y los problemas percibidos.

Existen distintos métodos para abordar la identificación de problemas, como grupos de enfoque, observaciones del producto en funcionamiento y entrevistas, en este caso es importante notar que dependiendo del problema se deberán hacer uno o varios de estos métodos a fin de encontrar el problema raíz. Por la naturaleza del problema ha solucionar se utilizará la metodología de la entrevista, la cual se adecua mejor a problemas que buscan crear soluciones desde cero.

En este caso se realizó una entrevista al profesor encargado del laboratorio ICAI el Dr. Ezequiel López. Se realizaron múltiples entrevistas las cuales permitan definir necesidades, aclarar dudas y características deseadas en el funcionamiento del sistema diseñado.

Tabla 3.1: Identificación del problema raíz

N°	Pregunta	Respuesta
1	¿Podrías explicarnos en detalle cuál es tu visión para este sistema de visión artificial y qué objetivos específicos tienes en mente?	Se busca desarrollar un sistema de visión artificial que pueda detectar y segmentar objetos en tiempo real para asistir en la conducción autónoma. Se quiere que sea capaz de identificar vehículos, peatones y bicicletas, señales de tráfico, semáforos y otros objetos relevantes en carretera.
2	¿Cuál diría que es el fin de este proyecto?	El objetivo principal es mejorar la seguridad vial y permitir una conducción más eficiente y autónoma.
3	¿Tiene alguna preferencia en cuanto a las tecnologías que te gustaría que se utilicen para desarrollar este sistema?	Como parte de los alineamientos del laboratorio se debe utilizar sistemas de inteligencia artificial, como técnicas de aprendizaje profundo y redes neuronales convolucionales, ya que estas tecnologías han demostrado un rendimiento excepcional en tareas similares.

4	¿Tienes algún requisito específico en cuanto al hardware o la plataforma en la que se debe ejecutar este sistema de visión artificial?	Gracias al avance tecnológica, existen sistemas embebidos los cuales son relativamente potentes, por lo cual la única limitante es que sea capaz de correr en una GPU de portátil.
5	Además de la detección y segmentación de objetos, ¿hay otras características o funcionalidades que te gustaría que se incluyan en el sistema?	Al ser un sistema de asistencia de frenado a la conducción autónoma, uno de los puntos más importantes es que el sistema diseñado sea capaz de medir la distancia de los objetos alrededor del vehículo, de esta forma poder actuar acorde a la situación.
6	¿Tienes algún plazo en mente para la entrega de este sistema?	Al ser un laboratorio de investigación no se tiene un plazo definido para el desarrollo del proyecto, pero al ser un proyecto de graduación este deberá ser terminado en un plazo de 16 semanas, lo que equivale a un semestre.
7	¿Se tiene algún presupuesto estimado para la realización de este proyecto?	No realmente, lo que puede brindar el laboratorio es un espacio de investigación y acceso a los recursos que cuenta el laboratorio ICAI, tal como bibliografía, servidores y asesoramiento.

3.3. Identificación de necesidades

Como una de las partes fundamentales en la metodología de diseño de Ulrich-Eppinger, es necesario generar necesidades específicas a partir de la entrevista anterior, mostrada en la Tabla 3.1. La Tabla 3.2 que se muestra a continuación se creó conversando con el cliente en diversas ocasiones y revisadas por el criterio experto del mismo cliente. También las necesidades fueron calificadas en orden de importancia con el siguiente criterio:

1. Función no deseada en el sistema.
2. Función no importante.
3. Función buena de tener pero no indispensable.
4. Función altamente deseable.
5. Función totalmente indispensable.

Tabla 3.2: Necesidades del problema

N°	Necesidad	Jerarquia
	Sistema de detección y segmentación de objetos	
1	El SD debe contar con un algoritmo de detección de objetos, por medio de aprendizaje profundo.	5
2	El SD debe contar con un algoritmo de segmentación de los objetos detectados.	4
3	El SD debe detector objetos en la carretera, tal como vehículos, peatones, bicicletas, señales de tráfico, semáforos y obstáculos.	5
4	La detección del SD debe ser precisa para asegurar una toma de decisiones segura.	4
5	El SD deberá etiquetar adecuadamente cada objeto de interés.	4
6	El SD deberá funcionar en una gran variedad de condiciones.	3
7	El tiempo de procesamiento del SD debe ser veloz para reaccionar a tiempo al frenado.	3
	Sistema de medición de distancia	
8	El SD posee una integración sensorial por medio datos LiDAR.	5
9	El SD deberá medir la distancia a la cual se encuentran los objetos.	4
	Sistema de toma de decisiones	
10	El SD cuenta con un algoritmo de decisiones el cual permita realizar un frenado de emergencia o desaceleración.	5
11	El SD desacelerara automáticamente al detectar un objeto y dependiendo de su velocidad, según la siguiente tabla.	4
12	El tiempo de reacción del SD deberá ser veloz para actuar y frenar lo más estable posible.	3

3.4. Establecimiento de especificaciones objetivo

En esta sección se identificarán las especificaciones objetivo. Como indica [15] el objetivo de esta tabla es que cada necesidad sea evaluada por al menos una especificación. La idea de esta tabla es cuantificar de alguna manera las necesidades las cuales reflejen el cumplimiento de cada una por medio de valores marginales e ideales. Se muestran las especificaciones en la siguiente Tabla 3.3.

Tabla 3.3: Especificaciones del problema

N°	Necesidades	Especificación	Unidad	Valor Marginal	Valor Ideal
1	1, 3, 5 y 6	Capacidad de detectar objetos vehiculares	Sensibilidad	97 %	>97 %
			Especificidad	85 %	>85 %
2	2, 3 y 4	Capacidad de segmentar	IoU	60 %	>60 %
3	7	Tiempo de inferencia del modelo	ms	1000ms	<1000ms
4	9	Capacidad de medir la distancia de un objeto detectado	Porcentual	5 %	<5 %
5	8	Integración con sistemas LiDAR	Binario	Si	Si
6	10 y 11	Capacidad de actuar al detectar un objeto	Binario	Si	Si
7	12	Tiempo de frenado y/o desaceleración	ms	1000ms	<1000ms
8	7 y 12	Tiempo total del sistema	ms	2000ms	<2000ms

3.5. Generación de conceptos

3.5.1. Descomposición funcional del problema

A fin de reducir la complejidad del problema, se subdivide el problema principal en segmentos funcionales los cuales podrán generar soluciones, las cuales posteriormente se acoplen para llegar a un resultado final.

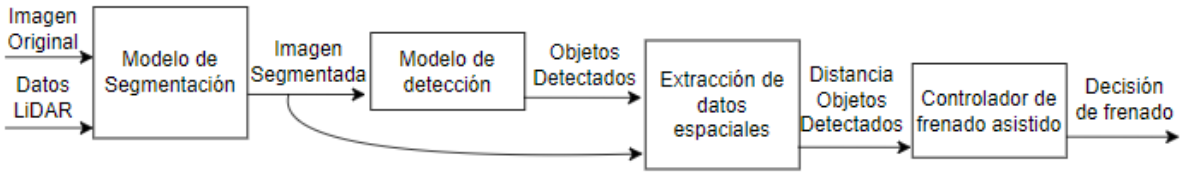


Figura 3.2: Descomposición del problema.

Como se muestra en la Figura 3.2, se descompone el problema en 4 grandes bloques, los cuales ayudaran a subdividir el problema y generar múltiples conceptos los cuales podrán dar solución al problema general. Además, de poder evaluar, esto para llegar a la mejor solución posible. A continuación, se procederá a explicar bloque a bloque:

Como primer punto es necesario segmentar los objetos deseados, separando los objetos de interés del resto información en las imágenes, para ese caso se muestra en la siguiente Figura 3.3 el bloque funcional que genera esta función, este recibe como entrada las imágenes y la información espacial, generando como salida una imagen segmentada.

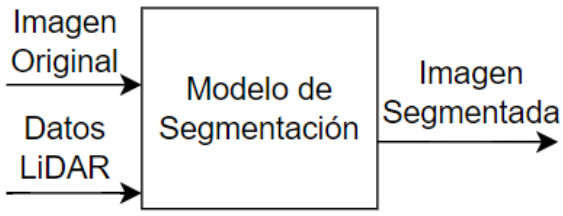


Figura 3.3: Fase de segmentación.

Tal como indica [17], la segmentación subdivide una imagen en objetos, separando los objetos de interés de otros objetos y su fondo, esta segmentación dependerá del problema a resolver. Un ejemplo de segmentación se muestra en la Figura 3.4.

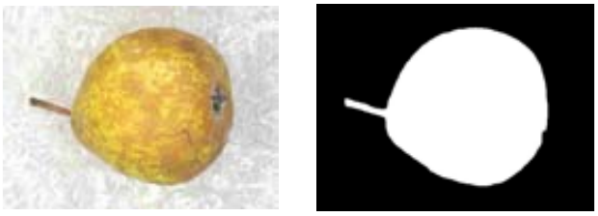


Figura 3.4: Segmentación de una imagen [17].

Posteriormente se procederá a dar un sentido semántico, el cual podrá clasificar cada objeto en la imagen segmentada, tal como se muestra en la Figura 3.5.

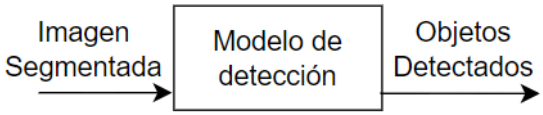


Figura 3.5: Fase de detección.

Según explica [18], la detección de objetos hace referencia a la clasificación de un objeto por medio de la extracción de características, las cuales permiten clasificar el objeto, en la siguiente Figura 3.6 se muestra cómo se podría clasificar una imagen como persona a partir de características propias de la imagen.

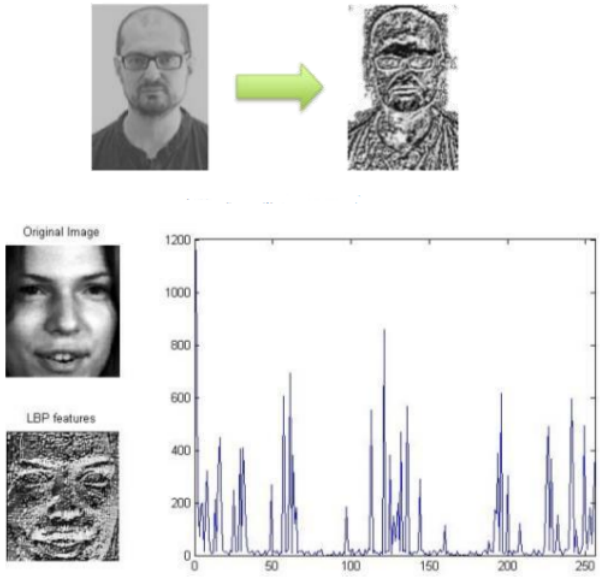


Figura 3.6: Detección de un objeto.

Una vez reunida la información necesaria se podrá calcular la distancia de cada objeto detectado, con esto se lograra tener valores que puedan ser utilizados en el bloque de control, los cuales se ejemplifican en la Figura 3.7.

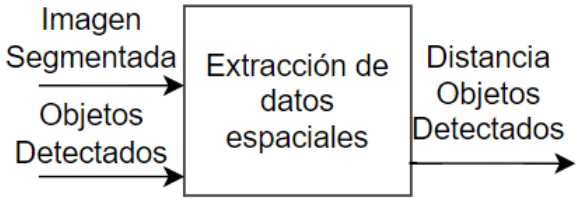


Figura 3.7: Fase de cálculo de distancia.

Como última etapa se procederá a controlar el sistema de frenado asistido, mostrado en la figura 3.8.

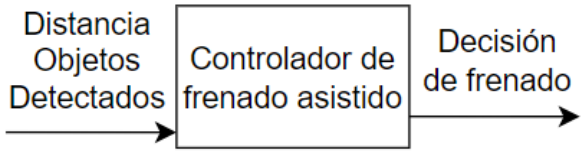


Figura 3.8: Fase control.

3.5.2. Búsqueda interna y externa de conceptos

Árboles de conceptos

Tabla 3.4: Identificación del problema raíz

N°	Bloque	Posibles soluciones
1	Segmentación	U-Net Faster R-CNN Complex YOLOv4
2	Detección	Resnet50 Complex YOLOv4 Técnica de Mascara
3	Calculo de distancia	Geometría Método Triangular
4	Control de frenado asistido	Control PID Modelo de aprendizaje automático Función linear

Segmentación

- U-Net: Como muestra [19], este modelo consta de dos partes, llamadas codificación y decodificación, como se muestra en la Figura 3.9, la parte izquierda comprime la información de la entrada, codificando la imagen en menos valores y la parte derecha decodifica esta información generando, en casos de segmentación las máscaras de los objetos.

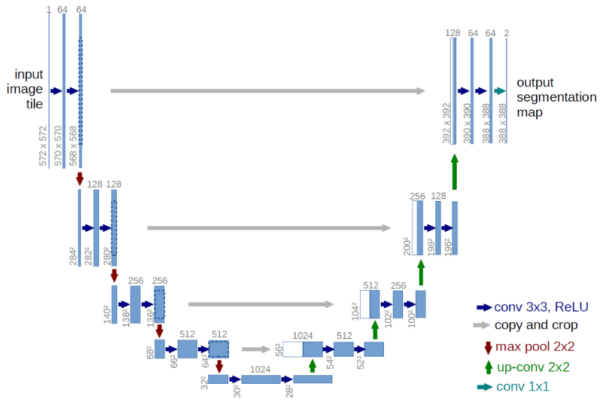


Figura 3.9: Arquitectura de U-Net [19].

- Faster R-CNN: Tal como indica [20], este modelo busca dividir la imagen regiones, llamadas regiones propuestas.º regiones candidatas", tal como se muestran en la Figura 3.10

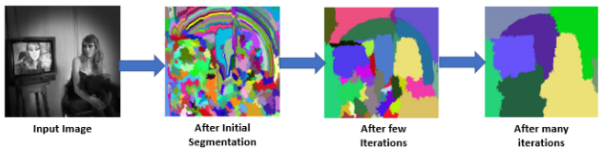


Figura 3.10: Modelo R-CNN [20].

- Complex YOLOv4: Indica [21] que el modelo utilizar las características de la imagen para predecir el recuadro y lo realiza para todas las clases simultáneamente. Esto lo logra dividiendo la imagen en celdas y prediciendo cada recuadro en cada celda, como se muestra en la Figura 3.11.

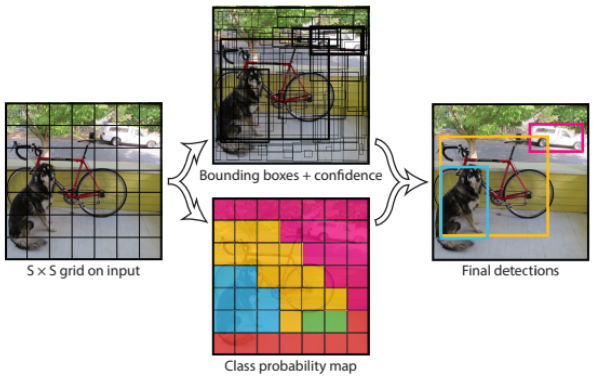


Figura 3.11: Modelo YOLO [21].

Detección

- ResNet50: Explica [22] esta red se basa en el aprendizaje residual profundo, esto se logra al apilar capas que se ajusten a un mapeo residual, lo cual es mas fácil que dejar que se ajusten directamente al mapeo subyacente deseado.

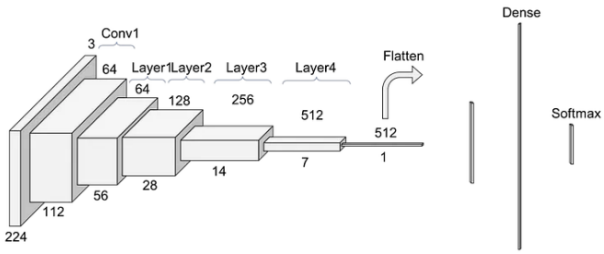


Figura 3.12: Modelo ResNet [22].

- Técnica de Mascara: Comenta [23] una de las formas de aislar zonas concretas y extraer información de estas partes en una imagen. En particular son muy útiles en la detección de objetos y ubicar objetos específicos en imágenes.

Figura 3.13: Detección con máscaras [23].

Calculo de distancia

- Geometría de imagen: Este método consiste en identificar las características de la imagen a nivel de píxel, tal como ejemplifica [24], se identifica el cuadro delimitador del objeto y su centro a esto se realiza una cuenta en píxeles de la distancia entre dos centros de cuadros delimitadores y por medio de una regla de tres se llevan a valores físicos, tal como se muestra en la Figura 3.14.

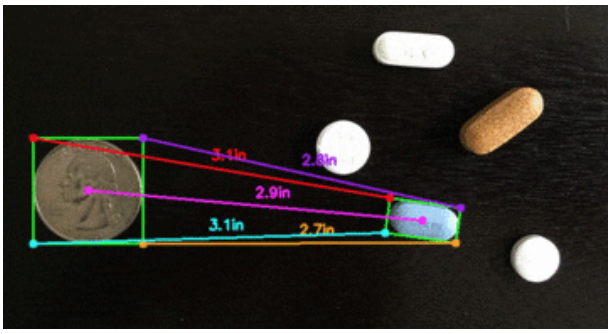


Figura 3.14: Calculo de distancia a partir de píxeles en una imagen [24].

- Método Triangular: Indica [25], esta técnica utiliza las relaciones entre triángulos tomando en cuenta características conocidas de la cámara como longitud focal y tamaño del sensor, como se muestra en la Figura 3.15 se puede disponer de al menos dos cámaras las cuales permita triangular la posición de puntos desde dos perspectivas, a partir de esto se puede generar una ecuación con la cual se calcule la distancia de un punto especifico.

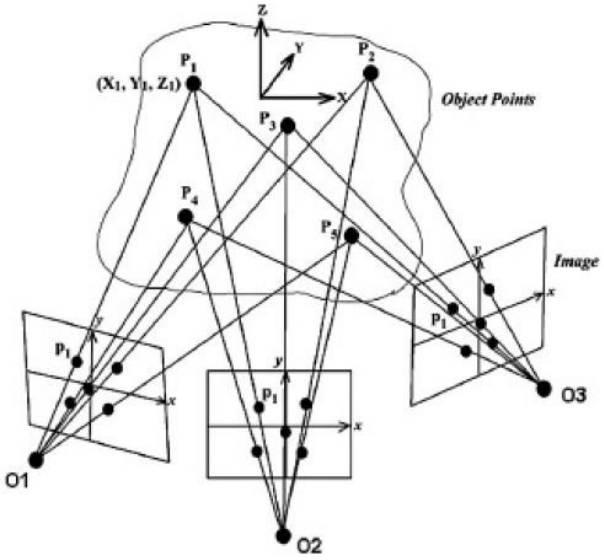


Figura 3.15: Método de triangulación [25].

Control de frenado asistido

- Control Proporcional-Integral-Derivativo (PID): Comenta [26], este control consta de tres elementos de accion proporcional, integral y derivativo, tal como se observa en la Figura 3.16, este control es de lazo cerrado, lo cual significa que posee retroalimentación donde la señal de referencia $r(t)$ y la salida $y(t)$ se relacionan por medio de el sensor $h(t)$, con este es posible calcular un error y disminuirlo por medio de los valores PID.

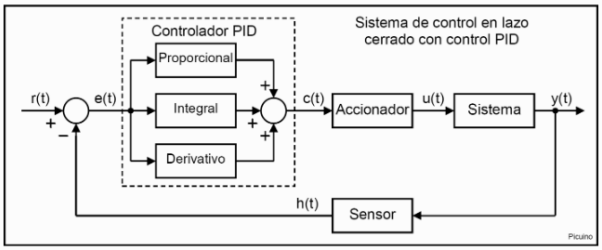


Figura 3.16: Control PID [26].

- Modelo de aprendizaje automático: Tal como indica [4], el aprendizaje automático es una rama de la inteligencia artificial que se centra en el desarrollo de sistemas capaces de aprender y mejorar automáticamente a partir de la experiencia, entre estas tenemos las siguientes:
  - Modelos de Regresión
  - Redes Neuronales
  - Modelos de Aprendizaje Profundo

- Función linear: Como propone [27], la función linear se puede calcular a partir de dos puntos, tal como se muestra en la Figura 3.17, se puede calcula el valor de y a partir de dos puntos, como $y = m(x - x_0) + b$, donde los términos m y b se calculan como 3.1.

$$m = \frac{y_1 - y_0}{x_1 - x_0} b = y_0 - mx_0 \tag{3.1}$$

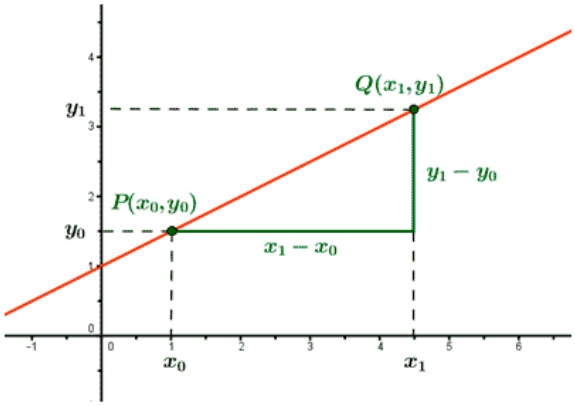


Figura 3.17: Interpolación Lineal [27].

Combinación de conceptos

Como siguiente paso se procede a combinar los conceptos para generar soluciones posibles, tomando en cuenta que estas combinaciones sean posibles. Se muestra en la Tabla 3.5, 5 soluciones generadas.

Tabla 3.5: Conceptos generados

C.	Segmentación	Detección	Calculo de distancia	Control
A	U-Net	ResNet50	Geometría	Modelo de aprendizaje automático
B	Faster R-CNN	Técnica de Mascara	Método Triangular	Función Lineal
C	Complex Yolov4	Complex Yolov4	Geometría	Función Lineal
D	Faster R-CNN	ResNet50	Método Triangular	Función Lineal
E	Complex Yolov4	Complex Yolov4	Método Triangular	PID

3.6. Selección de conceptos

En la siguiente Figura 3.18 se puede observar la comparativa de los modelos U-Net, Fast R-CNN y YOLO para la segmentación, los cuales muestran métricas de la precisión y IoU promedios, de aquí se puede observar que en promedio la U-Net es la que genera mejores resultados, seguidos por YOLO y finalmente el Fast R-CNN.

Table 3: The performance of the vanilla U-Net, Faster R-CNN, Yolo, and the proposed RU-Net model.

Model	Class A		Class B		Class C		Class D	
	Mean acc.	Mean IOU	Mean acc.	Mean IOU	Mean acc.	Mean IOU	Mean acc.	Mean IOU
-Proposed RU-Net	0.997	0.981	0.929	0.937	0.995	0.930	0.940	0.928
-Vanilla U-Net	0.921	0.873	0.931	0.912	0.912	0.911	0.939	0.912
-Faster R-CNN	0.763	0.717	0.799	0.732	0.733	0.689	0.881	0.797
-Yolo	0.886	0.814	0.878	0.791	0.911	0.778	0.910	0.828

Figura 3.18: Comparación en modelos de segmentación [28].

La siguiente Figura 3.19, muestra una comparativa entre los modelos YOLO y Resnet para problemas de clasificación de objetos, además de métricas de rendimiento. Como se indica, a pesar de que la ResNet50 posee una mayor precisión, este es un modelo que toma un mayor tiempo de inferencia, entre 6 a 9 veces mayor que el YOLO, lo cual para sistemas en tiempo, el sistema no actuaria en el tiempo preciso.

Table 3. Comparison of YOLOv5 and Faster R-CNN performance.

Metrics	YOLOv5			Faster R-CNN [43]				
	Y_l	Y_m	Y_S	ResNet50 (FPN)	VGG16	MVGG16	Mobile-Net V2	Inception V3
Precision (P)	86.43%	86.96%	76.73%	91.9%	69.8%	81.4%	63.1%	72.3%
Training Loss	0.015	0.017	0.020	0.065	0.226	0.136	0.209	0.194
Mean Average Precision (mAP@0.5–0.95)	63.43%	61.54%	58.9%	64.12%	35.3%	45.4%	30.5%	32.3%
Inference speed: Image resolution (1774 × 2365)	0.014 s	0.012 s	0.009 s	0.098 s	0.114 s	0.047 s	0.036 s	0.052 s
Inference speed: Image resolution (204 × 170)	0.018 s	0.013 s	0.009 s	0.065 s	0.119 s	0.052 s	0.032 s	0.056 s
Training time/epoch	26 s	16 s	12 s	124 s	173 s	105 s	80 s	95 s
Total training time	31,200 s	19,200 s	14,400 s	12,400 s	17,300 s	10,500 s	8000 s	9500 s
Model Size (MB)	95.3	43.3	14.8	165.7	175.5	134.5	329.8	417.2

Figura 3.19: Comparación en modelos de clasificación [29].

3.6.1. Filtrado

Como primer paso se deben filtrar los conceptos, de esta forma se logra solamente evaluar las mejores propuestas de solución.

Tabla 3.6: Filtrado de conceptos combinados

Filtrado	**Conceptos**				
Criterios	**Solución** A	**Solución** B	**Solución** C	**Solución** D (base)	**Solución** E
Capacidad de detectar objetos vehiculares	0	0	0	0	0
Capacidad de segmentar	+	0	+	0	+
Tiempo de inferencia del modelo	-	0	+	0	+
Integración con sistemas LiDAR	0	0	+	0	+
Capacidad de medir la distancia de un objeto detectado	+	0	+	0	0
Tiempo de frenado y/o desaceleración	-	0	0	0	-
Suma +	2	0	4	0	3
Suma 0	2	6	2	6	2
Suma −	2	0	0	0	1
Evaluación	0	0	4	0	2
Posición	3	3	1	3	2
¿Pasa?	No	No	Si	No	Si

3.6.2. Evaluación

Tabla 3.7: Evaluación de conceptos

Selección de Conceptos		Conceptos			
Criterio	**Peso**	**Solución C (base)**		**Solución E**	
		Calificación	**Evaluación**	**Calificación**	**Evaluación**
Capacidad de detectar objetos vehiculares	10	7	0.7	9	0.9
Capacidad de segmentar	15	10	0.67	13	0.87
Tiempo de inferencia del modelo	15	15	1.0	7	0.47
Integración con sistemas LiDAR	20	20	1.0	5	0.33
Capacidad de medir la distancia de un objeto detectado	20	18	0.9	15	0.75
Tiempo de frenado y/o desaceleración	20	19	0.95	16	0.8
Total de puntos		89	5.22	65	4.12
Posición		1^{er}		2^{do}	

3.7. Concepto escogido

El concepto escogido es la solución C la cual corresponde a la combinación mostrada en la Tabla 3.8.

Tabla 3.8: Conceptos generados

Segmentación	Detección	Calculo de distancia	Control
Complex Yolov4	Complex Yolov4	Geometría	Función Lineal

Uno de sus mayores factores que hacen destacar esta solución es su integración con sistemas LiDAR, lo cual evita un procesamiento extra en comparación a las opciones A, B y D. Esto facilita el calculo de distancia, ya que los datos LiDAR son datos espaciales y esto disminuye el error que posiblemente tenga la medición.

En el caso del cálculo de distancia, comparando el método geométrico al método triangular, este afectaría en el sentido de que es necesario otra cámara más para realizar el cálculo o dos fotogramas seguido, lo cual aumentaría el tiempo y complejidad de la solución.

Por parte del control, en comparación a la D, para el tipo de problema que se tiene de calcular la velocidad, la opción D su implementación es más complicada, siendo en este caso un enfoque mejor la facilidad de implementación para un cálculo más veloz, ya que se quiere que el sistema actué de la forma más rápida posible enfrente de situaciones de un acercamiento peligroso a objetos.

Capítulo 4

Propuesta de diseño

4.1. Segmentación y detección de objetos con Complex YOLOv4

Tal como comenta [30], en los últimos años se ha estado desarrollando cada vez más las tecnologías en conducción autónoma, dentro de estas tecnologías se encuentran los sistemas LiDAR los cuales tienen la capacidad de generar mapas espaciales por medio de sensores, generando nubes de puntos, debido a esto se han desarrollado sistemas de detección de objetos 3D basados en esta tecnología.

La siguiente proposición se desarrolló basándose en la famosa arquitectura de YOLOv4, utilizando estos mapas de nubes de puntos se logra realizar la detección de objetos alrededor de vehículos con sistemas LiDAR

Tal como se muestra en la siguiente Figura 4.1, el modelo propuesto consta de tres partes, conversión de los mapas de nube de puntos a imágenes, detección de objetos en imágenes y utilización de E-RPN para proyectar los recuadros predichos en imágenes de vista de frontal.

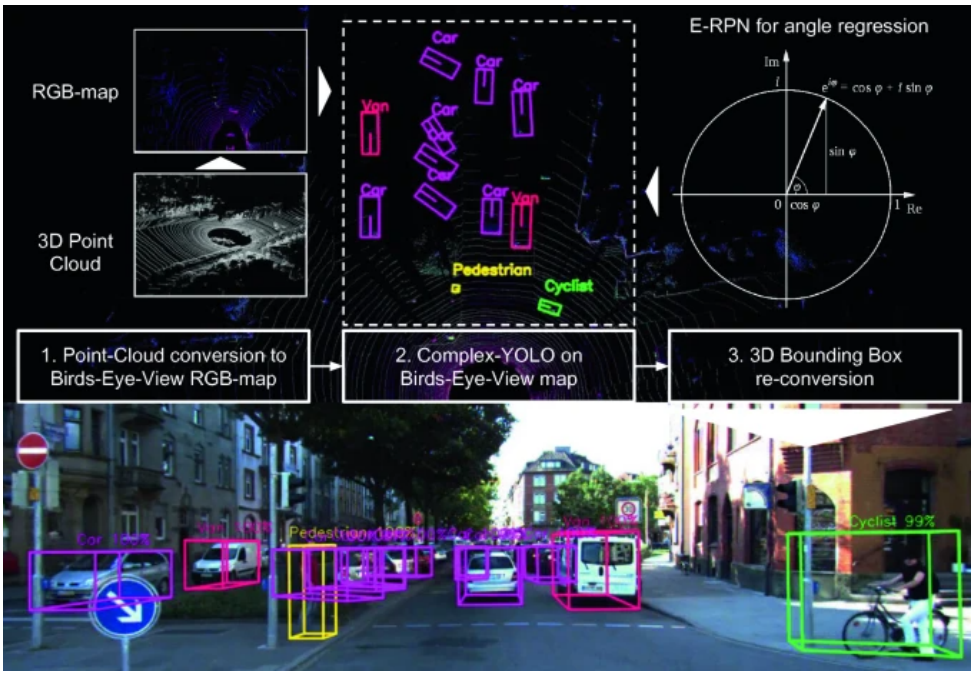


Figura 4.1: Pipeline modelo Complex YOLOv4 [30].

4.1.1. Obtención de los datos

Como primer punto se debe obtener datos LiDAR los cuales puedan ser utilizados en el sistema propuesto, estos se obtienen de una base de datos libre para estos tipos de modelos, tal como KITTI Database, como indica [30] tiene datos LiDAR tomados a partir de un láser escáner Velodyne HDL64, los cuales cubren un espacio de $80mX40m$ como se muestra en la Figura 4.2.

4.1.2. Conversión de datos LiDAR en imágenes

Los datos de nube de puntos son convertidos en imágenes RGB-map, estas imágenes son desde una perspectiva superior, además, codifican la información de altura, intensidad y densidad de puntos en una sola imagen, haciendo posible identificar objetos y sus formas generales, esta información es posteriormente segmentada generando datos que pueden ser utilizados por modelos de detección de objetos tal como YOLOv4 y entrenados para inferir la posición y dimensión de objetos a partir de sistemas LiDAR. En la siguiente Figura 4.2 se muestra a la izquierda la detección de objetos, en este caso automóviles y a la derecha la imagen se muestra las características de los datos tomados por el sistema LiDAR.

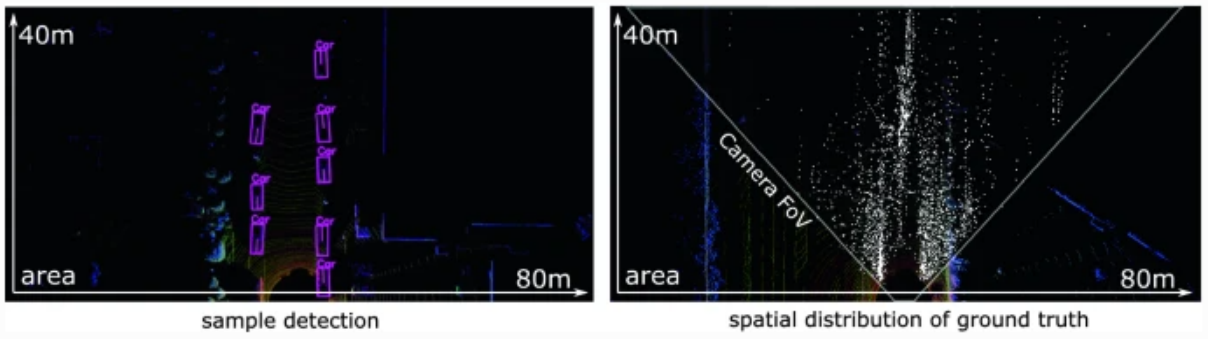


Figura 4.2: Características de nube de puntos para sistema LiDAR [30].

4.1.3. Arquitectura

Tal como menciona [30], el modelo Complex YOLOv4 toma como entrada las imágenes RGB-map, las cuales pasan por una arquitectura YOLO simplificada, agregando un sistema de regresión de ángulo complejo y E-RPN para detectar objetos 3D multi-clase, tal como se muestra en la Figura 4.3.

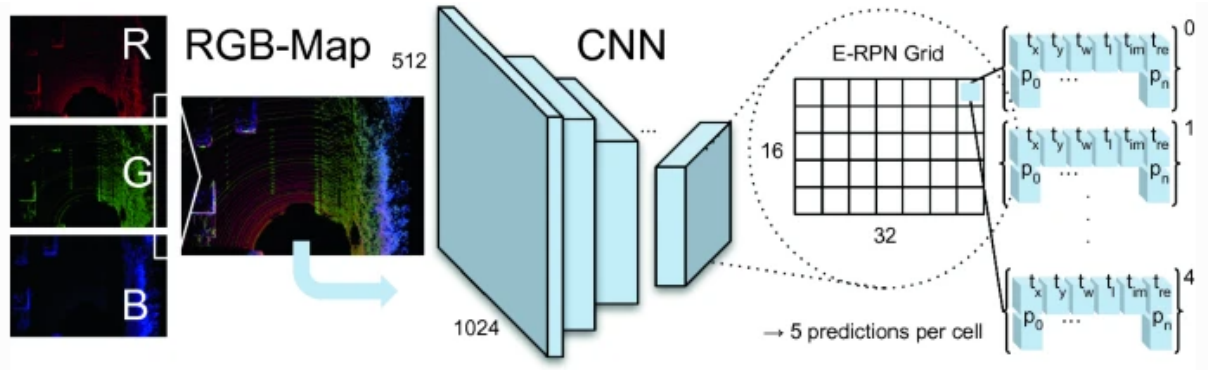


Figura 4.3: Arquitectura de modelo Complex YOLOv4 [30].

Aplicación de algoritmo de Propuesta de región de Euler (E-RPN)

A partir de la información de posicionamiento 3D $b_{x,y}$, dimensiones del objeto $b_{w,l}$, la probabilidad de la clase p_0 y finalmente el ángulo complejo de orientación b_ϕ, mostrado en la Figura 4.4, estos son calculados de la siguiente forma:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_l &= p_l e^{t_l} \\ b_\phi &= arctan(t_{Im}, t_{Re}) \end{aligned} \tag{4.1}$$

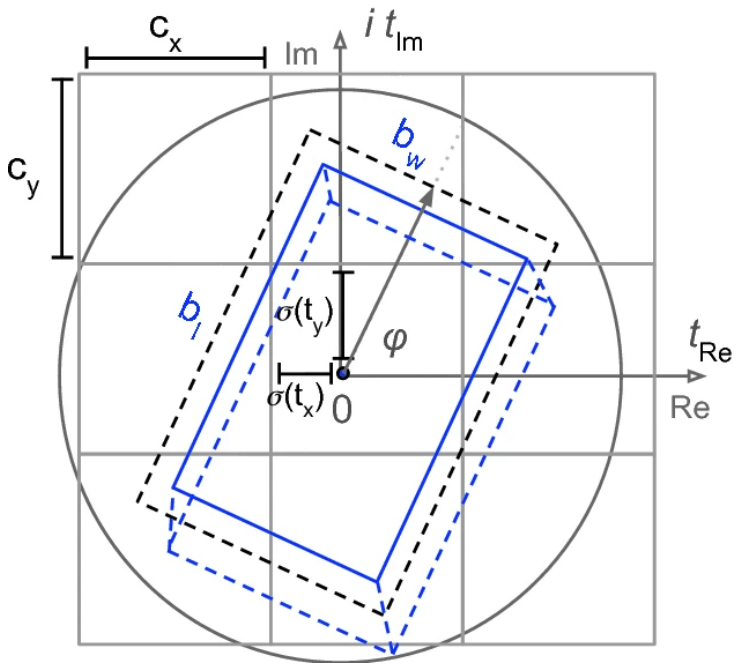


Figura 4.4: Dimensiones de cuadro delimitador [30].

Diseño de cuadro delimitador

Debido a las características de la base de datos KITTI, se realiza la clasificación de objetos determinando tres posibles casos:

- Vehículo o Carro
- Ciclista
- Peatón

Gracias a la naturaleza de los objetos los cuales se pueden diferenciar por medio del cuadro delimitador que tiene cada uno, ya que se pueden diferenciar fácilmente a partir del área de cada uno, es posible la utilización de un modelo de detección de objetos entrenado con datos propios, como lo es YOLOv4.

Regresión de ángulo complejo

El ángulo del cuadro delimitador tiene como objetivo dar la orientación y área del objeto, lo más cercana a la realidad del objeto, también al aplicar E-RPN permite encerrar en un paralelepípedo rectangular el objeto de esto es proyectar el cuadro delimitador de un espacio 2D a un espacio 3D. Esto se ejemplifica en la siguiente Figura 4.5.

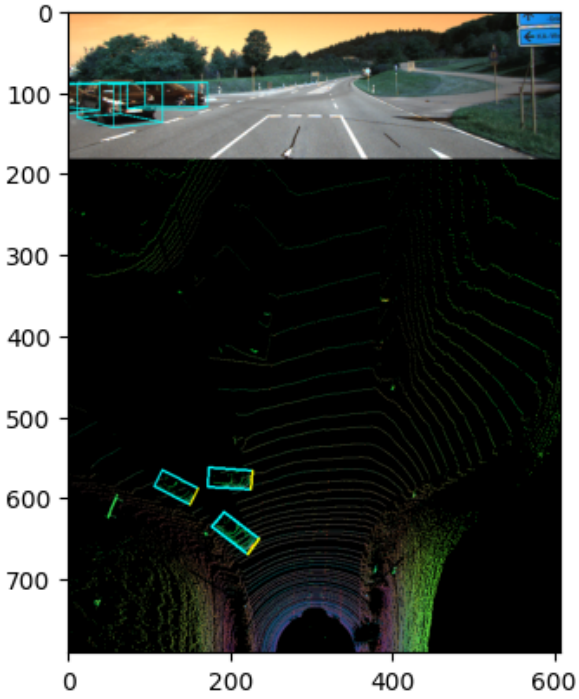


Figura 4.5: Proyección de cuadros delimitador de imagen 2D a 3D.

4.2. Calculo de distancia

Para calcular la distancia de los objetos, se propone como segundo punto calcular el centro del cuadro delimitador y como primer punto la ubicación del sistema LiDAR, tal como se muestra en 4.2, este se encuentra en la parte inferior y centrada de la imagen de RGB-map. Se muestra en la Figura 4.6 la representación gráfica del cálculo de distancia entre dos puntos.

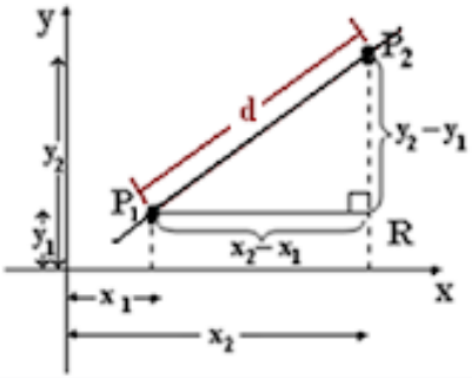


Figura 4.6: Calculo de distancia entre dos puntos [31].

$$d_{P_1,P_2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4.2}$$

4.3. Control

4.3.1. Normativas y legislación Española en conducción autónoma

Se indica en la Agencia Estatal Boletín Oficial del Estado de España, según la normativa DOUE-L-2021-80297, en el artículo 5.2.3.3, el cual se muestra en la Figura 4.7 la distancia reglamentaria mínima que debe tener un vehículo de otro objeto es la siguiente, dependiendo de la velocidad del vehículo.

La distancia mínima de seguridad se calculará por medio de la siguiente fórmula:
dmin= vALKS* tfront
Donde:

dmin	=	distancia mínima de seguridad
vALKS	=	velocidad actual del vehículo equipado con sistema automático de mantenimiento del carril en m/s
tfront	=	intervalo mínimo de seguridad en segundos entre el vehículo equipado con sistema automático de mantenimiento del carril y un vehículo precedente conforme al cuadro siguiente:

Velocidad actual del vehículo equipado con sistema automático de mantenimiento del carril		Intervalo mínimo de seguridad	Distancia mínima de seguridad
(km/h)		(m/s)	(s)	(m)
7,2	2,0	1,0	2,0
10	2,78	1,1	3,1
20	5,56	1,2	6,7
30	8,33	1,3	10,8
40	11,11	1,4	15,6
50	13,89	1,5	20,8
60	16,67	1,6	26,7

Con valores de velocidad no mencionados en el cuadro, se aplicará una interpolación lineal.
Sin perjuicio del resultado de la fórmula anterior para velocidades actuales inferiores a 2 m/s, la distancia mínima de seguridad nunca será inferior a 2 m.

Figura 4.7: Tabla de valores de velocidad y distancia según legislación Española [32].

Tal como se muestra en la Figura 4.8, la relación entre la distancia mínima de seguridad y la velocidad actual del vehículo es lineal, esto permite representar estos puntos como una ecuación lineal.

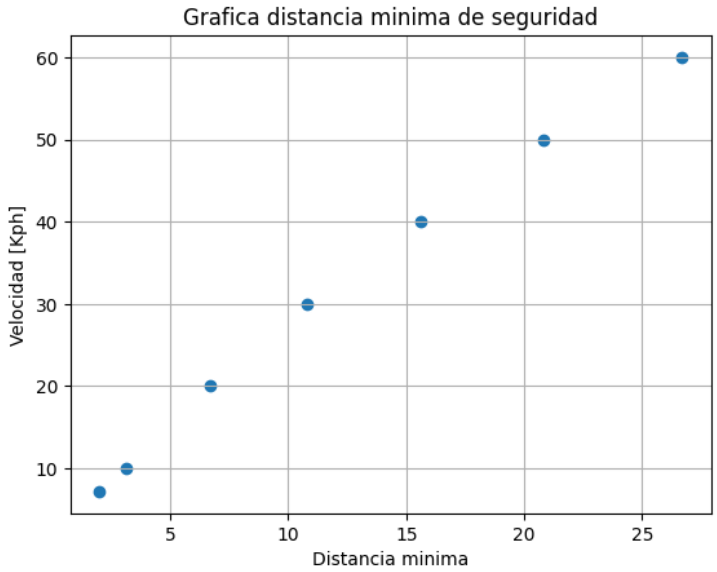


Figura 4.8: Gráfica con valores de distancia mínima de seguridad y velocidad actual del vehículo.

4.3.2. Calculo de velocidad a partir de la distancia al objeto.

A partir de la tabla en la Figura 4.7, se genera el siguiente árbol de decisiones el cual actúa como el controlador del sistema, permitiendo actuar al sistema en respuesta a distintas situaciones en la carretera, esto se ejemplifica en la siguiente Figura 4.9, donde dependiendo del rango de distancia en la que este el vehículo se encuentra, se calcula la velocidad que debería tener en el momento.

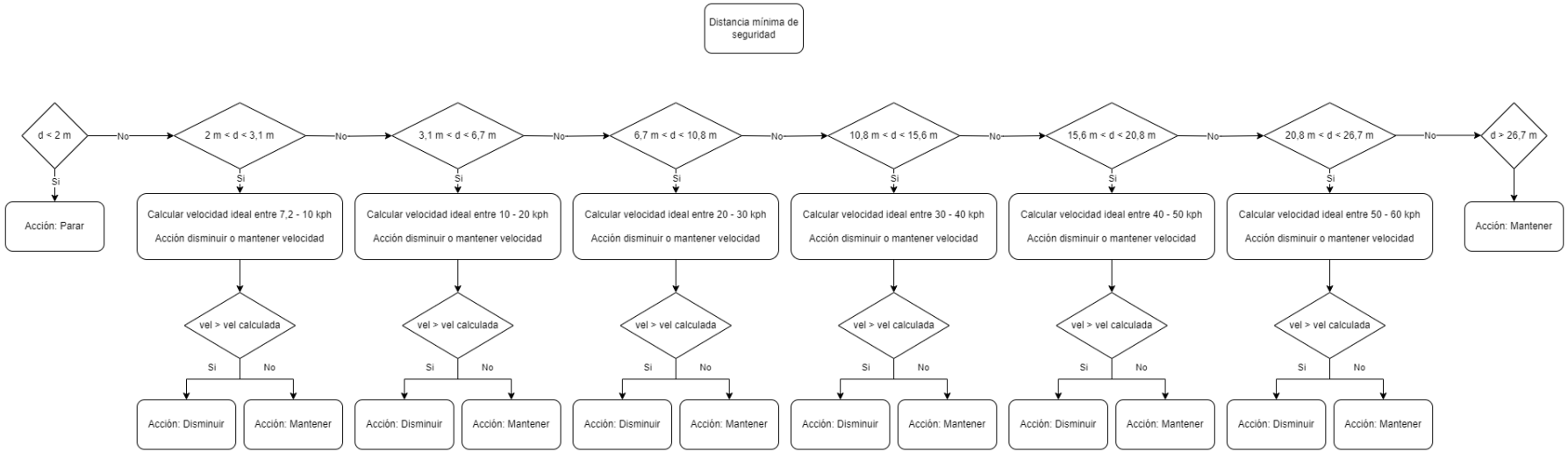


Figura 4.9: Árbol de toma de decisiones para el control de velocidad de vehículo.

Tomando como ejemplo un vehículo que tiene una velocidad de 12 kph y se encuentra a una distancia de 2,7 m, según se muestra en la Figura 4.9 la velocidad que debería tener el vehículo esta entre 10 a 7,2, por lo cual según el árbol de decisión el vehículo deberá disminuir su velocidad a la calculada en ese rango.

4.3.3. Control de velocidad actual del vehículo

Esta propuesta de solución es enteramente teórica, debido a que el Laboratorio ICAI no cuenta con los recursos para realizar una prueba de campo práctica. Por lo tanto, la validación de funcionamiento se realiza mediante un algoritmo computacional.

Así mismo el algoritmo de toma de acciones se basa en el árbol de decisiones mostrado en la Figura 4.9, el sistema podrá calcular la velocidad que debería tener el vehículo a partir de la distancia medida por el sistema, en base a esto podrá tomar tres decisiones, tal como se muestra a continuación.

- Parar o frenar vehículo: Este caso el sistema de frenado estará a una distancia igual o menor a 2 metros y sin importar la velocidad su accionar es frenar completamente.
- Disminuir velocidad: En esta situación el sistema detecta que la velocidad a la cual debería ir el vehículo es mayor a la velocidad calculada en el árbol de decisiones, con esta diferencia se deberá disminuir la velocidad de forma gradual hasta estar en la velocidad definida por la normativa.
- Mantener velocidad: Para este caso el sistema detecta que la velocidad se encuentra dentro de los rangos de seguridad y por lo tanto solo mantendrá la velocidad actual del vehículo.

Cabe resaltar que en este proyecto solamente se identifica si el accionar que toma el sistema es correcto al reconocer los tres casos definidos anteriormente, ya que este no sera acoplado a un sistema móvil o de simulación.

El algoritmo planteado se muestra en el Apéndice G.

Capítulo 5

Resultados y análisis

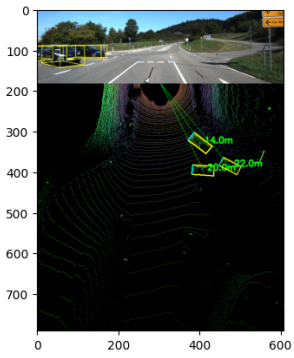
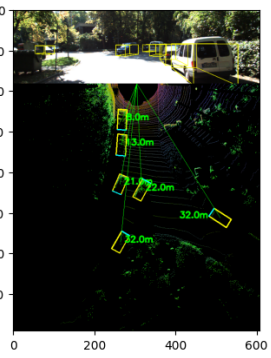
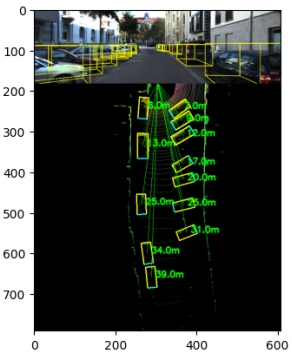
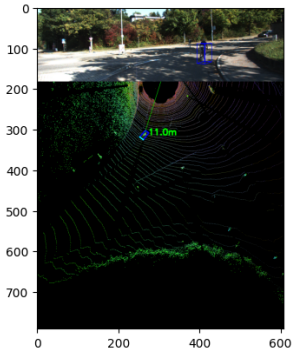
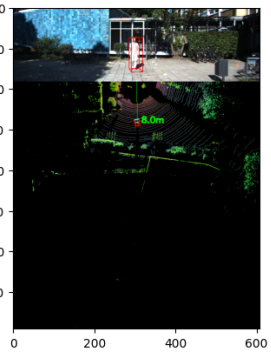
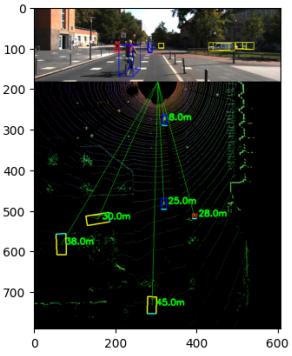
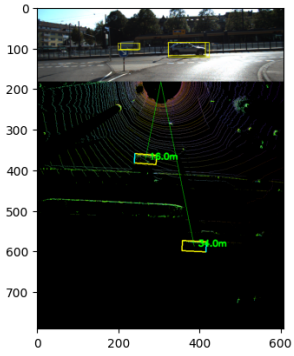
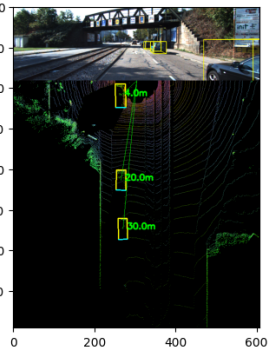
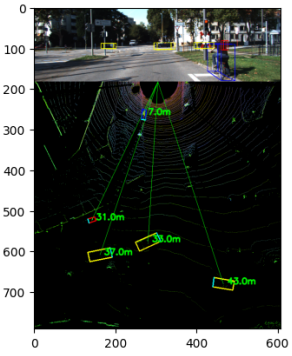
A continuación, se muestra en la Tabla 5.1 los diferentes conjuntos de datos utilizados en la validación del sistema diseñado, estas son diferentes rutas de ciudades, en las cuales varían, objetos, cantidad de datos y ambientes. Estas diferencias sirven para evaluar diferentes cantidades de peatones, carros y ciclistas, además de contar con diferentes ambientes, desde ciudad, suburbios, periferias y climas, estos también contando con distintas horas, lo cual varia la exposición al sol.

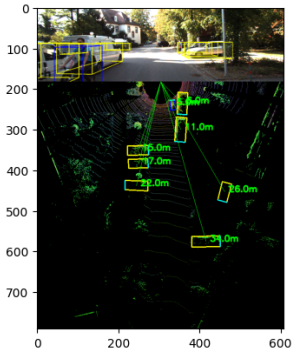
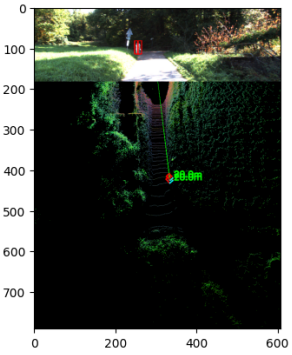
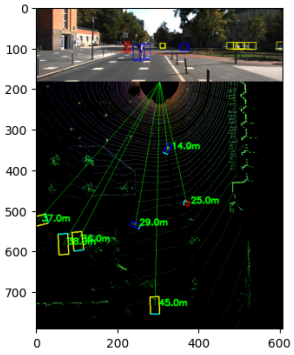
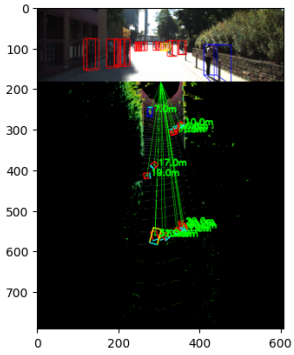
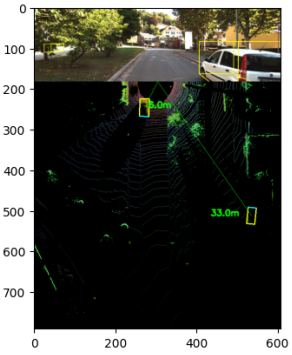
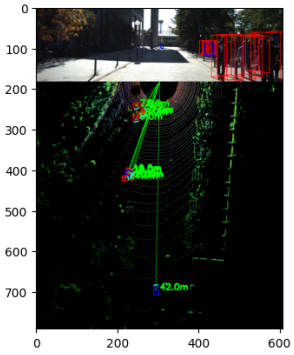
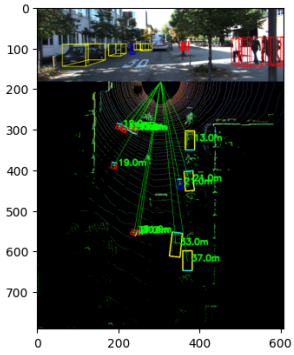
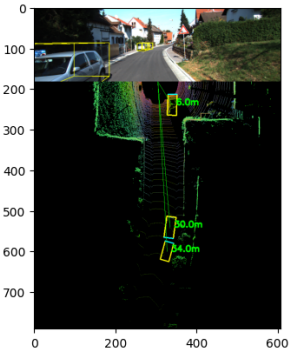
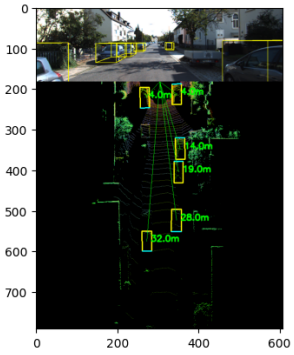
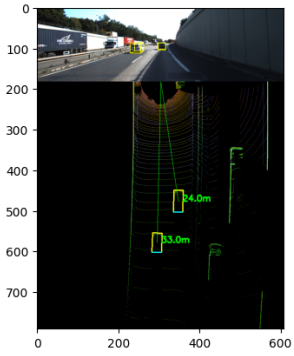
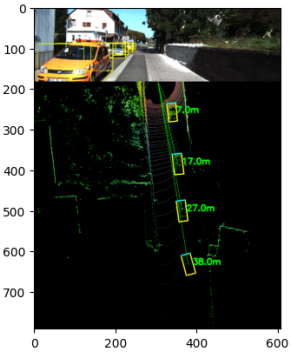
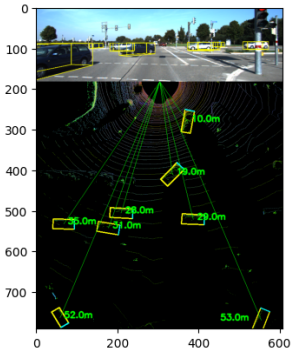
Tabla 5.1: Características de conjunto de datos

Ruta	Características
Conjunto de datos 1	 2011_09_26_drive_0011 (0.9 GB) **Length:** 238 frames (00:23 minutes) **Image resolution:** 1392 x 512 pixels **Labels:** 15 Cars, 1 Vans, 1 Trucks, 1 Pedestrians, 0 Sitters, 1 Cyclists, 0 Trams, 1 Misc **Downloads:** [unsynced+unrectified data] [synced+rectified data] [calibration] [tracklets]
Conjunto de datos 2	 2011_09_26_drive_0001 (0.4 GB) **Length:** 114 frames (00:11 minutes) **Image resolution:** 1392 x 512 pixels **Labels:** 12 Cars, 0 Vans, 0 Trucks, 0 Pedestrians, 0 Sitters, 2 Cyclists, 1 Trams, 0 Misc **Downloads:** [unsynced+unrectified data] [synced+rectified data] [calibration] [tracklets]
Conjunto de datos 3	 2011_09_26_drive_0117 (2.6 GB) **Length:** 666 frames (01:06 minutes) **Image resolution:** 1392 x 512 pixels **Labels:** 0 Cars, 0 Vans, 0 Trucks, 0 Pedestrians, 0 Sitters, 0 Cyclists, 0 Trams, 0 Misc **Downloads:** [unsynced+unrectified data] [synced+rectified data] [calibration]
Conjunto de datos 4	 2011_09_26_drive_0091 (1.3 GB) **Length:** 346 frames (00:34 minutes) **Image resolution:** 1392 x 512 pixels **Labels:** 2 Cars, 1 Vans, 0 Trucks, 42 Pedestrians, 14 Sitters, 8 Cyclists, 0 Trams, 1 Misc **Downloads:** [unsynced+unrectified data] [synced+rectified data] [calibration] [tracklets]
Conjunto de datos 5	 2011_09_28_drive_0002 (1.5 GB) **Length:** 382 frames (00:38 minutes) **Image resolution:** 1392 x 512 pixels **Labels:** 0 Cars, 0 Vans, 0 Trucks, 0 Pedestrians, 0 Sitters, 0 Cyclists, 0 Trams, 0 Misc **Downloads:** [unsynced+unrectified data] [synced+rectified data] [calibration]
Conjunto de datos 6	 2011_09_29_drive_0071 (4.1 GB) **Length:** 1065 frames (01:46 minutes) **Image resolution:** 1392 x 512 pixels **Labels:** 0 Cars, 0 Vans, 0 Trucks, 0 Pedestrians, 0 Sitters, 0 Cyclists, 0 Trams, 0 Misc **Downloads:** [unsynced+unrectified data] [synced+rectified data] [calibration]
Conjunto de datos 7	 2011_09_26_drive_0096 (1.9 GB) **Length:** 481 frames (00:48 minutes) **Image resolution:** 1392 x 512 pixels **Labels:** 0 Cars, 0 Vans, 0 Trucks, 0 Pedestrians, 0 Sitters, 0 Cyclists, 0 Trams, 0 Misc **Downloads:** [unsynced+unrectified data] [synced+rectified data] [calibration]

Además de estos de conjuntos de datos, se seleccionaron tres imágenes de cada uno para probar diferentes situaciones en las cuales se pueda evaluar el control del sistema diseñado como se muestra en la Tabla 5.2, tal como distintos objetos detectados a distintas distancias, en diferentes ambientes.

Tabla 5.2: Imágenes seleccionadas para resultados finales

Ruta	Image DS#1	Image DS#2	Image DS#3
Conjunto de datos 1			
Conjunto de datos 2			
Conjunto de datos 3			

Ruta	Image DS#1	Image DS#2	Image DS#3
Conjunto de datos 4			
Conjunto de datos 5			
Conjunto de datos 6			
Conjunto de datos 7			

5.1. Pruebas de validación y resultados

En este apartado se realizan pruebas para evaluar el primer objetivo, tal como se muestra en la Tabla 5.3, en este se evaluaron las métricas relacionadas a la detección y segmentación de los objetos vehiculares.

Tabla 5.3: Factores de prueba del sistema objetivo 1.

Objetivo	Necesidad	Especificación	Variable de muestra	Factor de influencia
1. Desarrollo de modelos de detección y segmentación	1. El SD debe contar con un algoritmo de detección de objetos, por medio de aprendizaje profundo.	1. Capacidad de detectar objetos vehiculares.	Sensibilidad & Especificidad	Ambiente, Condiciones de Luz, Variedad de Objetos
	3. El SD debe detector objetos en la carretera, tal como vehículos, peatones, bicicletas, señales de tráfico, semáforos y obstáculos.			
	5. El SD deberá etiquetar adecuadamente cada objeto de interés.			
	6. El SD deberá funcionar en una gran variedad de condiciones.			
	2. El SD debe contar con un algoritmo de segmentación de los objetos detectados.	2. Capacidad de segmentar.	IoU	Cantidad de Objetos, Cantidad de Datos
	4. La detección del SD debe ser precisa para asegurar una toma de decisiones segura.			
	7. El tiempo de procesamiento del SD debe ser veloz para reaccionar a tiempo al frenado.	3. Tiempo de inferencia del modelo	Tiempo	Cantidad de Objetos, Cantidad de Datos
	9. El SD deberá medir la distancia a la cual se encuentran los objetos.	4. Capacidad de medir la distancia de un objeto detectado correctamente.	Error	Cantidad de Objetos, Distancia de Objetos

5.1.1. Prueba 1: Capacidad de detectar objetos vehiculares

En esta prueba se evaluó la habilidad del sistema en identificar objetos, como se muestra en la siguiente Tabla 5.4 se logra tener una sensibilidad mayores al 0,97 y especificidad mayores al 0,85. En el Apéndice A se muestran los resultados de los cuales se basan la tabla de sensibilidad y especificidad, en este se puede resaltar que la mayoría de los objetos son carros en comparación a los peatones y ciclistas, esto debido a que la mayoría son rutas vehiculares.

Tabla 5.4: Sensibilidad y Especificidad.

Ruta	Objeto	Sensibilidad	Especificidad
Conjunto de datos 1	Carro	1,0000	1,0000
	Peatón	1,0000	0,9841
	Ciclista	0,9967	1,0000
Conjunto de datos 2	Carro	1,0000	1,0000
	Peatón	1,0000	0,9778
	Ciclista	0,9960	1,0000
Conjunto de datos 3	Carro	0,9912	1,0000
	Peatón	1,0000	0,9770
	Ciclista	0,9987	1,0000
Conjunto de datos 4	Carro	0,9714	0,9965
	Peatón	0,9933	0,8500
	Ciclista	0,9901	0,8997
Conjunto de datos 5	Carro	0,9861	0,9953
	Peatón	1,0000	0,9778
	Ciclista	0,9958	1,0000
Conjunto de datos 6	Carro	0,9941	1,0000
	Peatón	0,9966	0,9646
	Ciclista	0,9968	0,9474
Conjunto de datos 7	Carro	0,9935	1,0000
	Peatón	0,9939	0,9840
	Ciclista	0,9987	0,8621

5.1.2. Prueba 2: Capacidad de segmentar

Para esta prueba se evaluó, la intersección sobre la unión (IoU) entre el área de la segmentación original y la segmentación predicha. En este se logró un valor mayor de IoU a 0,60 con un promedio por encima del 0,85, lo cual indica que la segmentación en la mayoría de los casos se logró predecir efectivamente la posición de los objetos. Es importante recalcar la importancia de esta métrica al calcular la distancia de los objetos y sus posiciones alrededor del vehículo.

Tabla 5.5: Métrica IoU de segmentación

Ruta	IoU Promedio	IoU Mínimo	IoU Máximo
Conjunto de datos 1	0,8591	0,6061	0,9888
Conjunto de datos 2	0,8595	0,6354	0,9888
Conjunto de datos 3	0,8690	0,6026	1,0000
Conjunto de datos 4	0,8717	0,6523	0,9691
Conjunto de datos 5	0,8697	0,6436	0,9888
Conjunto de datos 6	0,8678	0,6036	1,0000
Conjunto de datos 7	0,8690	0,6026	1,0000

5.1.3. Prueba 3: Tiempo de inferencia del modelo

La siguiente prueba evaluó la métrica de tiempo en la inferencia, el cual logra ser en todos los casos menor a 1 segundo, tal como se muestra en la Tabla 5.6. La importancia de esta métrica radica en el tiempo de detección y cálculo de distancia, ya que permite al sistema de frenado del vehículo actuar con tiempo en situaciones de riesgo o fuera de los rangos estipulados por normativa, tal como se habló en secciones anteriores.

Tabla 5.6: Tiempos de inferencia en diferentes rutas

Ruta	Tiempo Promedio (ms)	Inferencia más larga (ms)
Conjunto de datos 1	225,83	965,12
Conjunto de datos 2	205,47	755,30
Conjunto de datos 3	255,77	943,76
Conjunto de datos 4	269,26	643,76
Conjunto de datos 5	244,06	843,76
Conjunto de datos 6	275,78	733,73
Conjunto de datos 7	255,77	943,66

5.1.4. Prueba 4: Capacidad de medir la distancia de un objeto detectado

En esta etapa se evaluó el error en la distancia calculada de la distancia real, esta se calcula como se muestra en la ecuación 5.1. Este error se logró mantener por debajo del 5 % en todos los casos, como se puede observar en la Tabla 5.7, esta métrica permite observar cuanta diferencia existen entre la distancia real y la distancia calculada, a partir del modelo de detección y segmentación de objetos.

$$error = \left| \frac{dist_{real} - dist_{calculada}}{dist_{real}} \right| \cdot 100 \tag{5.1}$$

Tabla 5.7: Error medio en diferentes rutas

Ruta	Error en porcentaje
Conjunto de datos 1	0,8324
Conjunto de datos 2	0,9203
Conjunto de datos 3	1,0234
Conjunto de datos 4	3,5125
Conjunto de datos 5	3,0216
Conjunto de datos 6	2,0312
Conjunto de datos 7	3,1234

El siguiente apartado, evaluó la integración de los datos LiDAR de cada uno de los conjuntos, lo cual independientemente del sistema LiDAR utilizado este puede ser utilizado por el sistema diseñado, permitiendo una integración con una gran variedad de sistemas LiDAR diferentes.

Tabla 5.8: Factores de prueba del sistema objetivo 2.

Objetivo	Necesidad	Especificación	Variable de muestra	Factor de influencia
2. Integración de datos de imágenes y LiDAR	8. El SD posee una integración sensorial por medio datos LiDAR.	5. Integración con sistemas LiDAR	Binario	Diferentes Datos LiDAR, Cantidad de Datos, Ambientes

5.1.5. Prueba 5: Integración con sistemas LiDAR

Se evaluó a continuación la integración de los sistemas LiDAR con cada uno de los conjuntos, tal como se muestra en la siguiente Tabla 5.9 se tiene una integración con los sistemas LiDAR del 100 %.

Tabla 5.9: Tiempos de inferencia en diferentes rutas

Ruta	¿Posee datos LiDAR? (binario)
Conjunto de datos 1	Si
Conjunto de datos 2	Si
Conjunto de datos 3	Si
Conjunto de datos 4	Si
Conjunto de datos 5	Si
Conjunto de datos 6	Si
Conjunto de datos 7	Si

En este apartado se evaluó el algoritmo de control por medio de un árbol de toma de decisiones, este permite apreciar el accionar del sistema a distintas situaciones típicas de la circulación vial de un vehículo.

Tabla 5.10: Factores de prueba del sistema objetivo 3.

Objetivo	Necesidad	Especificación	Variable de muestra	Factor de influencia
3. Desarrollo de algoritmos de toma de decisiones	10. El SD cuenta con un algoritmo de decisiones el cual permita realizar un frenado de emergencia o desaceleración.	6. Capacidad de actuar al detectar un objeto	Binario	Ambientes, Velocidades, Distancia de Objetos
	11. El SD desacelerara automáticamente al detectar un objeto y dependiendo de su velocidad, según la siguiente tabla.			
	12. El tiempo de reacción del SD deberá ser veloz para actuar y frenar lo más estable posible.	7. Tiempo de frenado y/o desaceleración	Tiempo	Distancias de Objetos

5.1.6. Prueba 6: Capacidad de actuar al detectar un objeto

A continuación se muestra la Tabla 5.11, donde se puede apreciar diferentes situaciones simuladas, en donde se obtiene como salida del sistema el accionar y se evaluó si este accionar fue correcto en cada situación según la normativa, como se aprecia el sistema actúa adecuadamente en el 100 % de los casos. Los resultados de estos casos se muestran en el Apéndice B. Además debido a la imposibilidad de encontrar casos en los cuales el vehículo este a menos de 2 metros de un objeto, ya que estaría fuera de los estándares legales o a punto de colisión, se optó por simular el caso al indicar la distancia de 2 metros con distintas velocidades.

Tabla 5.11: Tiempos total del sistema.

Ruta	Imagen	Distancia (m)	Velocidad (kph)	Vel. Calculada (kph)	Acción	**¿La acción es correcta?**
Cojunto de datos 1	DS11	14,0	25,0	36,67	Mantener	Si
	DS12	6,0	25,0	18,06	Disminuir	Si
	DS13	8,0	25,0	23,17	Disminuir	Si
Cojunto de datos 2	DS21	11,0	32,0	30,42	Disminuir	Si
	DS22	8,0	24,0	23,17	Disminuir	Si
	DS23	8,0	21,0	23,17	Mantener	Si
Cojunto de datos 3	DS31	16,0	23,0	40,76	Mantener	Si
	DS32	4,0	17,0	12,50	Disminuir	Si
	DS33	7,0	21,5	20,73	Disminuir	Si
Cojunto de datos 4	DS41	5,0	16,7	15,28	Disminuir	Si
	DS42	20,0	20,0	48,46	Mantener	Si
	DS43	14,0	20,0	36,67	Mantener	Si
Cojunto de datos 5	DS51	7,0	20,8	20,73	Disminuir	Si
	DS52	6,0	17,0	18,06	Mantener	Si
	DS53	6,0	17,0	18,06	Mantener	Si
Cojunto de datos 6	DS61	11,0	30,4	30,42	Disminuir	Si
	DS62	6,0	18,0	18,06	Mantener	Si
	DS63	4,0	15,0	12,50	Disminuir	Si
Cojunto de datos 7	DS71	24,0	60,0	55,42	Disminuir	Si
	DS72	7,0	22,0	20,73	Disminuir	Si
	DS73	10,0	20,0	28,05	Mantener	Si
Datos prueba frenado	DS01	2,0	5,0	0,00	Parar	Si
	DS02	2,0	24,0	0,00	Parar	Si
	DS03	2,0	55,0	0,00	Parar	Si

5.1.7. Prueba 7: Tiempo de acción de parar, mantener o disminuir velocidad

En esta prueba se evaluó el tiempo en determinar la acción que debe realizar el sistema, logrando estar por debajo de 1 segundo en todos los casos propuestos.

Tabla 5.12: Tiempos de acción del control.

Ruta	Imagen	Tiempo (ms)
Cojunto de datos 1	DS11	221,4
	DS12	550,7
	DS13	604,8
Cojunto de datos 2	DS21	116,9
	DS22	207,4
	DS23	767,6
Cojunto de datos 3	DS31	125,5
	DS32	273,1
	DS33	580,2
Cojunto de datos 4	DS41	745,6
	DS42	162,4
	DS43	561,0
Cojunto de datos 5	DS51	745,6
	DS52	186,5
	DS53	743,2
Cojunto de datos 6	DS61	831,0
	DS62	189,2
	DS63	739,0
Cojunto de datos 7	DS71	231,9
	DS72	501,9
	DS73	703,4

Para este apartado se evaluó el tiempo de procesamiento total del sistema diseñado, esto relacionado al objetivo 4, tal como se muestra en 5.13.

Tabla 5.13: Factores de prueba del sistema objetivo 4.

Objetivo	Necesidad	Especificación	Variable de muestra	Factor de influencia
4. Validación y pruebas en condiciones simuladas	7. El tiempo de procesamiento del SD debe ser veloz para reaccionar a tiempo al frenado.	8. Tiempo total del sistema	Tiempo	Ambiente, Cantidad de Objetos
	12. El tiempo de reacción del SD deberá ser veloz para actuar y frenar lo más estable posible.			

5.1.8. Prueba 8: Tiempo total del sistema

La siguiente prueba se muestra la evaluación del tiempo que toma todo el sistema en procesar la información y dar una acción como resultado, como se aprecia en la Tabla 5.14, todos los tiempos se encontraron por debajo de 2 segundos.

Tabla 5.14: Tiempos total del sistema.

Ruta	Imagen	Cantidad de Objetos	Tiempo (ms)
Cojunto de datos 1	DS11	3	620,3
	DS12	12	1650,7
	DS13	6	1454,8
Cojunto de datos 2	DS21	1	417,3
	DS22	1	504,9
	DS23	6	1467,6
Cojunto de datos 3	DS31	2	554,8
	DS32	3	774,6
	DS33	5	1423,4
Cojunto de datos 4	DS41	7	1467,4
	DS42	2	461,4
	DS43	7	1541,9
Cojunto de datos 5	DS51	13	1702,6
	DS52	2	404,8
	DS53	9	1528,5
Cojunto de datos 6	DS61	12	1640,4
	DS62	3	475,8
	DS63	6	1425,5
Cojunto de datos 7	DS71	2	540,3
	DS72	4	970,7
	DS73	8	1404,4

5.2. Análisis de resultados

A partir de las tablas de resultados, se procedió a realizar el análisis de cada prueba y como estos afectan el sistema diseñado. Las pruebas fueron diseñadas para responder a cada especificación, que a su vez fue definida para calificar cada necesidad y su nivel de cumplimiento.

5.2.1. Prueba 1

Esta prueba permite conocer que tan bueno o no es un modelo para detectar los objetos y clasificarlos de forma correcta, altos valores de sensibilidad indican que el modelo logro detectar la mayoría de objetos de interés y clasificarlos de manera correcta, debido a esto es importante en la conducción automática valores muy altos de sensibilidad, caso contrario se puede permitir un menor valor en la especificidad, ya que esta es la medida que indica cuantos de los objetos se identificaron como falsos positivos, en caso de una emergencia es más importante detectar realmente lo que si está al frente y en caso de un falso positivo el vehículo, no frena si no que solo reduce la velocidad permitiendo una situación más segura.

Es apreciable en la Tabla 5.4 como los valores que más bajan en sensibilidad y especificidad, suelen ser para el Peatón y Ciclista, esto se debe en parte a que un ciclista es una persona montada en un bicicleta y también que a comparación de un Carro, las dimensiones pueden ser más parecidas.

Es importante evidenciar que estas métricas fueron tomadas de diferentes conjuntos de datos, lo cual permite una mayor variedad de casos como diferentes exposiciones de luz, distintos tipos de caminos, carreteras y pasos peatonales, etc. Esto a fin de generar un modelo robusto que funcione en la mayoría de las situaciones.

5.2.2. Prueba 2

En este caso la prueba busca relacionar las distancias de los objetos, ya que, si el objeto y la inferencia se encuentran superpuestas, esto indica que la ubicación del objeto está bien calculada. Como se observa en la Tabla 5.5, en promedio en los distintos conjuntos de datos la intersección sobre la unión es mayor a un 85 %, esto permite confiar en que el objeto detectado por el modelo está en una ubicación muy cercana al objeto real y a partir de esto se disminuye la probabilidad de error en la distancia.

Además, es apreciable, que ninguno de los conjuntos obtuvo una métrica menor al 60 %, el cual es tomado por parte del Laboratorio ICAI como un buen valor de umbral para modelos semejantes y el promedio está por encima de este umbral lo cual indica que el modelo está detectando de forma muy precisa la ubicación de los objetos.

Estos valores también se deben al tipo de sistema que se utilizar, el LiDAR debido a que

este es un sistema de alta precisión, mapeando correctamente el alrededor del vehículo, para que posteriormente el modelo pueda detectar y segmentar los objetos.

5.2.3. Prueba 3

Para esta prueba, se toma evaluó una de las métricas más importante en un sistema en tiempo real, ya que uno de los grandes problemas en la conducción autónoma es el tiempo que le toma a un modelo inferir, para posteriormente actuar. Tal como se muestra en la Tabla 5.6, se puede observar los tiempos promedios y máximos. Este último es de suma importancia, debido a que se quiere que sin importar la situación el modelo detecte en menos de un segundo el objeto, para poder actuar adecuadamente.

También se logró evidenciar la correcta selección del modelo de detección y segmentación, ya que otros modelos que se estudiaron, aun teniendo mayor precisión, esto duraban alrededor de 6 veces realizando la inferencia.

5.2.4. Prueba 4

Esta prueba de una de mayor distancia, ya que el accionar del sistema dependerá directamente de la ubicación de los objetos alrededor del vehículo. Como se muestra en la Tabla 5.7, los errores se mantienen en un rango de 0,83 % y 3,13 %. Este error permite al sistema tener un rango para accionar, ya que errores mayores podrían detectar que un objeto que está a 1 metro de distancia, sea detectado a 2 metros, en cuyo caso el accionar del sistema no sería correcto.

Es importante aclarar que este valor es proporcional a la distancia del objeto, siendo de 5 metros para un objeto a 100 metros y 5 centímetros para una distancia de 1 metro. La magnitud de este valor puede ser la diferencia entre un atropello o accidente a un accionamiento de seguridad.

5.2.5. Prueba 5

En esta prueba se buscó cuantificar la integración del modelo actual a distintos sistemas LiDAR con distintas cámaras y configuraciones. Esto sin la necesidad de entrenar el modelo para cada sistema LiDAR diferente.

La importancia de este radica, en demostrar la viabilidad de aplicar este modelo a distintos sistemas LiDAR sin la preocupación de cómo se comportara en cada sistema nuevo.

5.2.6. Prueba 6

Otra prueba de suma importancia es la comprobación de que el sistema actúa como debería a distintas situaciones típicas y límites del tránsito vehicular. En esta se determina el correcto actuar en cada situación, esto validando si al comparar la velocidad calculada por la normativa y la velocidad actual, el sistema toma la acción apropiada.

En este caso la velocidad calculada dependerá totalmente de la distancia del objeto más cercano, a partir de esto el sistema detectará la acción correspondiente a cada caso.

Como se muestra en la Tabla 5.11, se debió generar datos que no existen en los conjuntos de datos, ya que estos casos serían muy cercanos a coalición y estos no se encuentran en ninguno de los conjuntos de datos encontrado.

5.2.7. Prueba 7

Para esta siguiente prueba se evaluó el tiempo del accionar del sistema, este se encuentra bastante por debajo del límite propuesto de 1 segundo. En este caso al ser un árbol de decisiones este suele ser bastante rápido al solamente evaluar en que condición esta y cuál es su velocidad calculada para comparar con la velocidad actual.

Las grandes diferencias de tiempos que se pueden encontrar en la Tabla 5.12, se deben a que el sistema debe predecir la distancia del objeto más cercano y este dependerá de la cantidad de objetos que estén presentes en la imagen o que sean detectados por el sistema.

5.2.8. Prueba 8

Como última prueba se evaluó el tiempo total que le toma a todo el sistema realizar la detección, segmentación y control. Al mostrar los datos tabulados en la Tabla 5.14, no se logra apreciar, pero al graficar estos como en la Figura 5.1 en esta se puede observar una relación directa entre la cantidad de objetos y el tiempo de procesamiento del sistema diseñado.

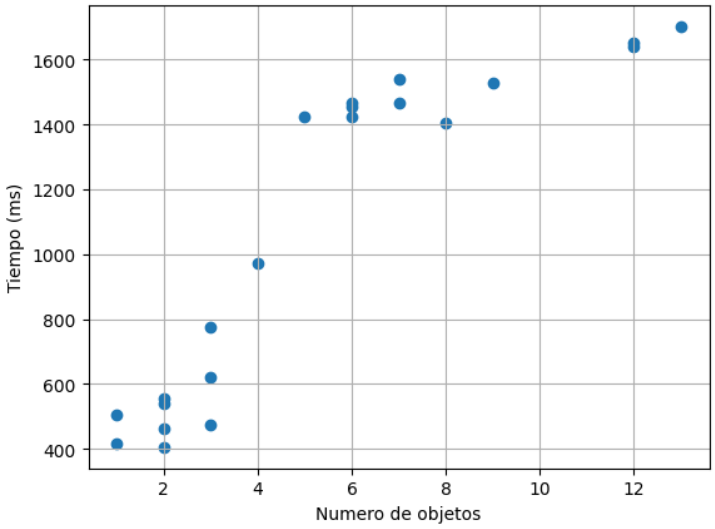


Figura 5.1: Relación entre numero de objetos detectado y el tiempo de procesamiento.

5.2.9. Análisis de casos específicos

En la siguiente Figura 5.2 se observa el accionar del sistema al calcular la distancia del objeto más cercano y medir la velocidad del vehículo, donde el vehículo mantiene la velocidad debido a que la velocidad se encuentra por debajo de la reglamentaria, el cual como se muestra en la Figura 4.9, según el árbol de decisiones es la acción correcta.

También se observa que el modelo logra identificar los 3 vehículos presentes en la carretera correctamente y se delimita el volumen de los vehículos correctamente, con lo cual se muestra que también el correcto funcionamiento de la predicción del cuadro delimitador de los datos LiDAR y la proyección a la imagen frontal de la cámara.

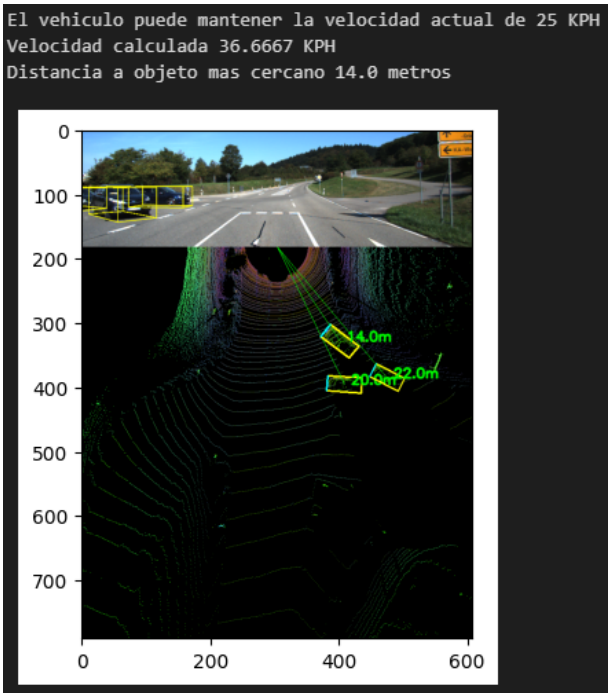


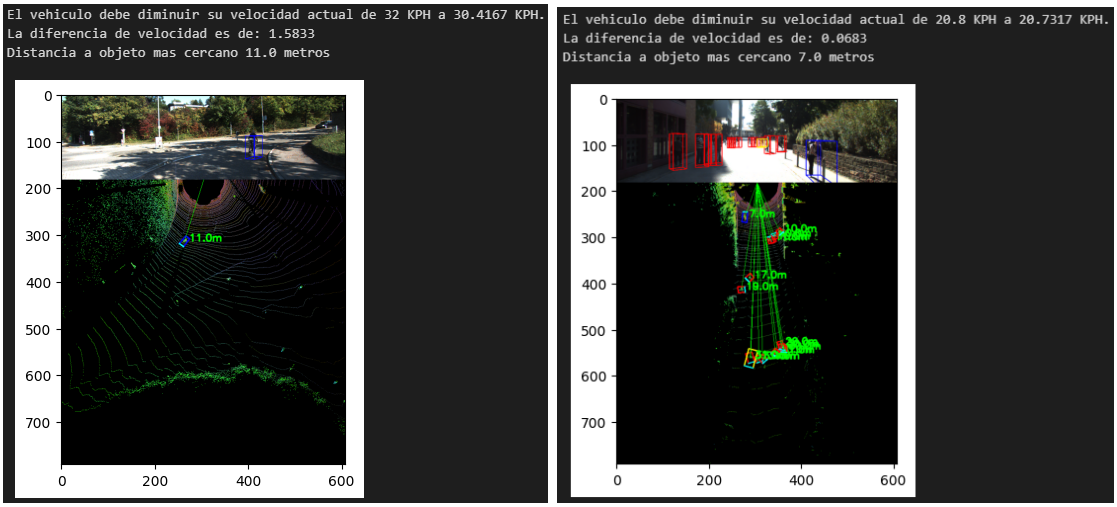
Figura 5.2: Resultados imagen DS11.

Para este caso se muestra en las Figuras 5.3a y 5.3b, el caso de disminución de velocidad, como se observa en ambos casos el sistema detecta que la velocidad debe ser ajustada a la velocidad determinada por la normativa, además el sistema muestra en la información superior de cada imagen la velocidad actual, la velocidad por normativa, la diferencia entre las velocidades y la distancia del objeto más cercano.

Un caso limite es cuando la velocidad actual es muy parecida a la velocidad por normativa, se muestra en la Figura 5.3b que sin importar como, a pesar de que la diferencia de velocidad sea pequeña esta debe ser ajustada para mantener el vehículo dentro de las normativas.

Este es un punto importante, ya que permite ejemplificar el por qué muchos sistemas se desean automatizar, esto debido a que las personas ante esta diferencia pueden reaccionar de distintas formas, un caso sería que disminuya la velocidad y siga la normativa, por otro lado, esta personar podría decidir que al ser una diferencia tan pequeña no es necesario disminuir la velocidad, rompiendo la reglamentación establecida y por ultimo si la persona no conoce la normativa, no podrá actuar acorde al ignorar la situación.

En este punto es donde la automatización se distingue, ya que este actuara siempre de la misma manera bajo las misma situaciones, sin importar que la diferencia de velocidades sea mínima.



(a) Resultados imagen DS21. **(b)** Resultados imagen DS51.

Figura 5.3: Casos de disminución de velocidad.

En la próxima Figura 5.4 se muestra como el sistema logra detectar objetos a una distancia considerables, tal como se aprecia a continuación se identifica un ciclista a 42 metros, el cual costaría aun para un humado reconocer tal objeto a esa distancia, como se muestra a la derecha se debió acercar la imagen para poder reconocer al ciclista.

Este es un gran indicativo de la confiabilidad que se le puede tener al sistema diseñando, ya que gracias a su sistema LiDAR, permite identificar objetos a largas distancia y clasificarlo, esto puede ser de gran interés para zonas especiales como hospitales o escuelas, donde es importante detectar a las personas a cualquier distancia.

Otra comparación que se puede realizar con el humano, es que la detección humana depende del foco de atención, en cambio en un sistema LiDAR puede ser de 360 grados, permitiendo actuar a eventos en los cuales el conductor no este poniendo atención a su alrededores.

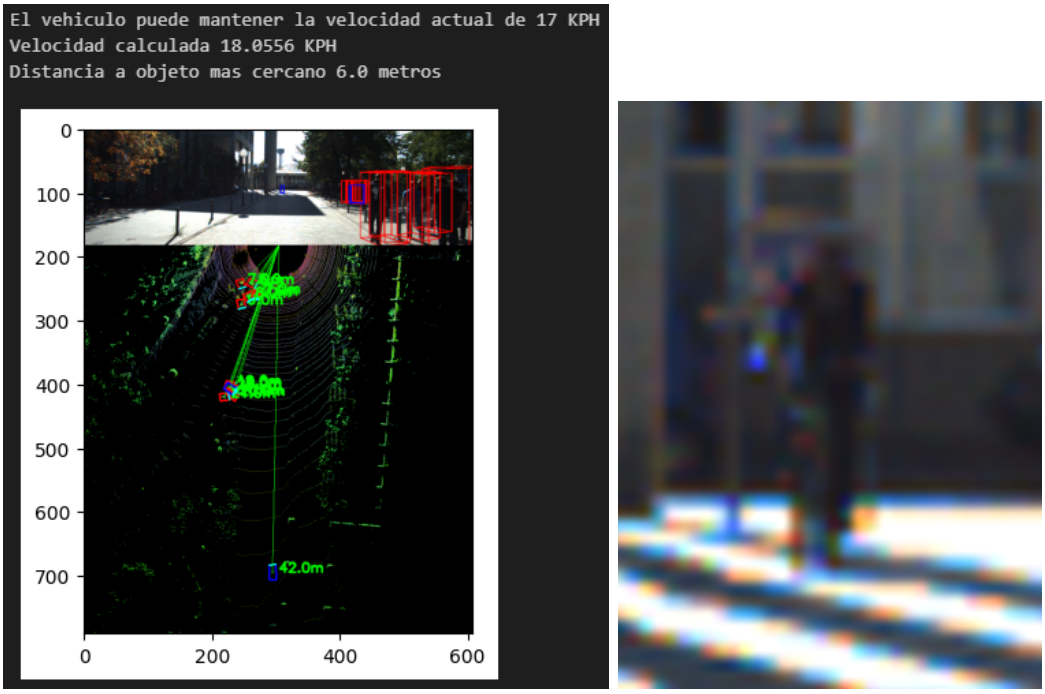


Figura 5.4: Resultados imagen DS53.

En esta Figura 5.5 se observa como el sistema logra identificar dos objetos, a pesar de que uno de los objetos o persona interfiere con el otro, lo cual permite tener una gran confiabilidad en el sistema, de que este realmente logra identificar los objetos en situaciones límites.

Como se muestra la imagen con mayor acercamiento, solamente al acercar y analizar la imagen se pueden observar dos personas, una caminando detrás de otra, lo cual muestra que el sistema es robusto a esta situación al no confundir los datos con una persona grande o confundir con un ciclista. Se observa en este ejemplo que aún un humano puede no reconocer a las dos personas, debido a su lejanía o posición, las cuales si son detectados por el sistema diseñado.

Figura 5.5: Resultados imagen DS42.

Para el siguiente caso mostrado en la Figura 5.6 se evalúa el caso donde la velocidad calculada y la velocidad del vehículo es la misma, en este caso el sistema detecta que se debe disminuir, con la salvedad de que esta tiene una diferencia de 0,0 kph, lo cual implica que no hay cambios en la velocidad, a pesar de que se haya determinado la acción de disminuir.

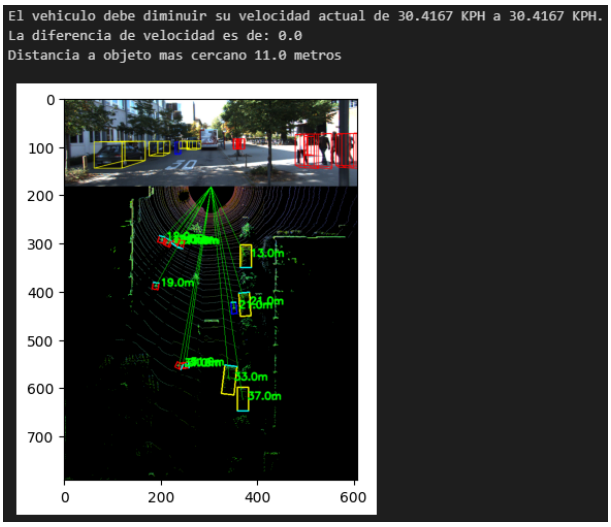


Figura 5.6: Resultados imagen DS61.

Otro ejemplo de la funcionalidad del sistema se encuentra en la siguiente Figura 5.7 donde se observa objetos a los lados del vehículo, este es importante recalcar que estos objetos son detectados, también, por lo cual el sistema logra identificar objetos aun a los lados

del vehículo.

A pesar de no ser parte de los objetivos del sistema, ya que el modelo se enfoca en los objetos del foco frontal del vehículo, en cuyo caso estos datos a los laterales del vehículo se podrían utilizar para otras funcionalidades, haciendo el sistema cada vez más versátil y flexible, para funcionalidades fuera de este proyecto, un ejemplo puede ser el estacionado automático, o que no permita abrir la puesta si se encuentra un obstáculo a la par.

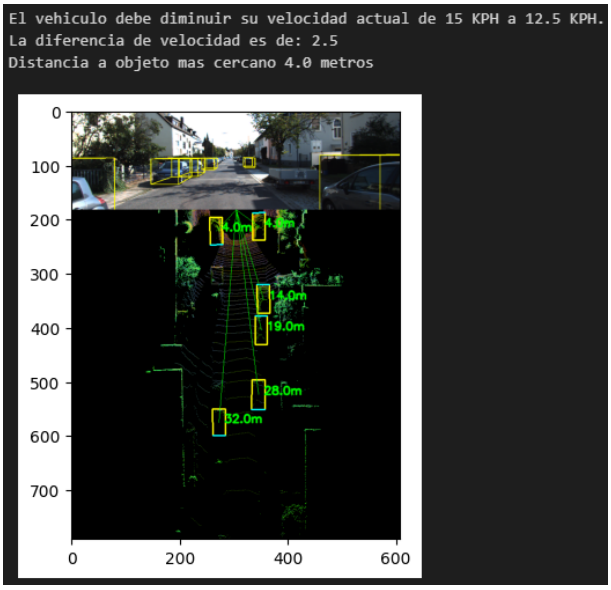


Figura 5.7: Resultados imagen DS63.

5.3. Análisis de económico y social

5.3.1. Análisis económico

Como se comentó anteriormente, este proyecto se realizó en conjunto al Laboratorio ICAI de la Universidad de Málaga y al ser un grupo de investigación en algunas ocasiones es difícil medir monetariamente el impacto que tienen los proyecto sobre el Laboratorio.

Se muestra en la siguiente Figura 5.8 el presupuesto utilizado en la realización de este proyecto, en el cual se muestran los gastos por conceptos de materiales, herramientas y servicios generales para un desarrollo de 16 semanas de proyecto.

Descripción	Cantidad	Valor estimado por unidad (colones)	Subtotal (colones)	Disponible en la empresa actualmente
Materiales y Herramientas				
Portátil para desarrollar el proyecto	1	675.000	675.000	NO
Laboratorio	-	-	-	SI
Conjunto de datos	-	-	-	SI
Servicios Generales				
Remuneración económica para proyecto de graduación.	-	-	-	---
Vivienda	4	150.000	600.000	
Alimentación	80	5.500	440.000	---
Transporte	160	1500	240.000	---
Imprevistos	1	120.000	120.000	---
Total			2.075.000	

Figura 5.8: Tabla de presupuesto del proyecto.

5.3.2. Beneficios Académicos

En primer lugar, estos proyectos suelen tener objetivos que van más allá de los beneficios económicos directos, como la generación de conocimiento, el desarrollo de habilidades y la creación de redes de colaboración. Estos aspectos intangibles pueden ser difíciles de cuantificar en términos monetarios.

Además, el impacto de dichos proyectos puede manifestarse de diversas maneras, como la mejora de la reputación y la visibilidad del laboratorio en la comunidad académica y científica, la atracción de talento humano y la obtención de financiamiento para futuras investigaciones. Estos efectos indirectos también son importantes, pero pueden resultar difíciles de traducir en cifras monetarias concretas.

En este caso se analizó el impacto que tiene la realización de este proyecto en tres aspectos:

- Primero: Se podrá publicar una investigación del proyecto realizado, permitiendo tener mayor visibilidad en esta área y además la posibilidad de expandir la investigación de conceptos o ideas que se hablaron pero no se pusieron en práctica. Esto permitirá al menos dos publicaciones más para el año 2024.
- Segundo: En este caso la realización de este proyecto permitirá abrir un nuevo campo de investigación, ya que antes de su desarrollo el Laboratorio no había realizado ningún trabajo en este campo. En este caso se podrá expandir este sistema a mayores funcionalidades por parte de otros investigadores o estudiantes los cuales quieran trabajar con datos LiDAR.
- Tercero: Al ser un sistema basado en datos LiDAR, se puede implementar a cualquier sistema de sensores LiDAR, un ejemplo de esto sería en sistemas de localización para distintos dispositivos, tales como drones, vehículos no tripulados, entre otros.

5.3.3. Beneficios económicos

Como indica [33], "las colisiones debidas al tránsito cuestan a la mayoría de los países el 3 % de su PIB", el cual es posible reducir significativamente implementado sistemas de conducción autónoma. Esta reducción en los costos asociados con las colisiones de tránsito no solo libera recursos financieros que pueden ser asignados a otros sectores prioritarios, sino que también contribuye al crecimiento económico sostenible al mitigar las pérdidas relacionadas con la interrupción del comercio, la productividad laboral reducida y los costos de atención médica y rehabilitación.

1. Reducción de costos por accidentes: Al disminuir la cantidad de accidentes de tráfico, se reducen los gastos asociados con reparaciones mecánicas de vehículos, gastos médicos y de rehabilitación para los involucrados, así como también costos legales y de seguros.

2. Ahorros en infraestructura: Menos accidentes significan menos necesidad de reparaciones y mantenimiento en la infraestructura vial, como calles, señalización y semáforos, lo que ahorra dinero tanto a nivel municipal como estatal.

3. Optimización de recursos de emergencia: La reducción de accidentes implica una menor movilización de patrullas policiales, ambulancias y bomberos, lo que se traduce en ahorros en combustible, mantenimiento de vehículos y salarios del personal de emergencia.

4. Incremento en la eficiencia del transporte: Con menos congestionamientos causados por accidentes, se mejora la fluidez del tráfico, lo que resulta en ahorros de combustible y tiempo para los conductores, así como una reducción en los costos logísticos para empresas de transporte.

5.3.4. Beneficios sociales

Como comenta [33], "las colisiones causadas por el tránsito se cobran la vida de aproximadamente 1,19 millones de personas". En el 94 % de casos son debido al error humano por distintos factores, tales como:

- Velocidad: "Existe una relación directa entre el aumento de la velocidad media y la probabilidad de que ocurra una colisión, así como con la gravedad de sus consecuencias. Por ejemplo, por cada aumento del 1 % en la velocidad media, el riesgo de que se produzca una colisión mortal se incrementa en un 4 %, y el de colisión grave, en un 3 %."
- Conducción en estados anormales como alcohol: El alcohol aumenta exponencialmente el riesgo de accidentes ya aun con bajos índices de alcohol el aumento de riesgo es considerable.
- Distracciones: En estos tiempos es muy común las distracciones mientras se conduce, tal como la utilización de celular u otros dispositivos.
- Incumplimiento de normas de tránsito: Este aspecto, engloba todas las anteriores además de agregar más como no respetar señales de tránsito, no seguir la demarcación vial correspondiente o conducción temeraria, por comentar algunas.

De forma generalizada los beneficios son:

1. Seguridad mejorada: La principal ventaja social de la conducción autónoma es la mejora de la seguridad vial. Al reducir la incidencia de errores humanos, como la distracción, la fatiga o la imprudencia, se pueden prevenir lesiones y fatalidades en accidentes de tráfico. Esto lleva a una reducción del sufrimiento humano, así como a una disminución de los costos asociados con la atención médica y la rehabilitación de víctimas de accidentes.

2. Cumplimiento normativo: Los vehículos autónomos están programados para cumplir estrictamente las normativas de tráfico establecidas por la ley. Esto significa que respetarán límites de velocidad, señales de tráfico y otras regulaciones de manera consistente. Como resultado, se reducirán las infracciones de tráfico y, por lo tanto, las multas y sanciones asociadas. Además, al seguir las normativas, se promueve un comportamiento vial más seguro y ordenado en general.

3. Accesibilidad mejorada: La conducción autónoma tiene el potencial de hacer que el transporte sea más accesible para personas con discapacidades físicas, visuales o cognitivas, así como para personas mayores que pueden tener dificultades para conducir. Al proporcionar una forma segura y confiable de transporte, se promueve la inclusión social y se brinda a más personas la oportunidad de participar activamente en la sociedad.

Capítulo 6

Conclusiones

En conclusión, el modelo de detección y segmentación desarrollado exhibe un desempeño notable, con valores de sensibilidad y especificidad superiores al 97 % y al 85 % respectivamente, junto con un índice de intersección sobre la unión (IoU) superior al 60 %. Además, su tiempo de procesamiento, inferior a 1 segundo, garantiza una respuesta rápida y eficiente. Estas características hacen que el sistema sea altamente confiable y adecuado para su implementación en sistemas de conducción autónoma, proporcionando una base sólida para la seguridad y la eficacia en la toma de decisiones en tiempo real.

Otra de las conclusiones es que el sistema diseñado se ha desarrollado con una compatibilidad del 100 % con datos LiDAR, lo que lo hace versátil y adaptable a una amplia gama de sistemas LiDAR de diferentes marcas y dispositivos. Esta característica permite una implementación fluida y eficiente en diversos contextos, brindando a los usuarios la capacidad de aprovechar este modelo para una variedad de tareas y aplicaciones en entornos LiDAR. La interoperabilidad del sistema no solo aumenta su utilidad, sino que también amplía su potencial de aplicación en diversos sectores y escenarios, mejorando así la accesibilidad y la eficacia de la tecnología LiDAR en general.

Además, el sistema de control desarrollado ha demostrado una capacidad excepcional al identificar la distancia de los objetos y tomar decisiones basadas en la velocidad del vehículo y su proximidad al objeto más cercano en el 100 % de los casos, todo ello en un tiempo inferior a 1 segundo. Este logro subraya la efectividad y la fiabilidad del sistema en la medición precisa de la distancia de los objetos y sus características físicas relevantes para la seguridad y la operatividad del vehículo. La capacidad de respuesta rápida del sistema garantiza una gestión eficiente y oportuna de las situaciones potencialmente peligrosas en la carretera, lo que contribuye significativamente a la seguridad y la confianza en los sistemas de control vehicular.

También, el sistema desarrollado ha alcanzado un nivel de confiabilidad superior al 90 % y un tiempo de procesamiento inferior a 2 segundos, lo que representa un hito significativo en su eficacia operativa. La rapidez de respuesta del sistema es un aspecto crucial, ya que asegura que, además de ser preciso, pueda reaccionar a tiempo frente a las situaciones

cambiantes del entorno. La combinación de precisión y eficacia resalta la robustez del sistema, asegurando su utilidad y seguridad en diversas aplicaciones. Este resultado valida la viabilidad y la relevancia del sistema como una herramienta confiable y eficiente en entornos dinámicos donde la toma de decisiones precisa y rápida es fundamental.

En resumen, las pruebas y resultados destacan el potencial transformador de la integración de la inteligencia artificial en la conducción autónoma. No solo se vislumbra una mejora significativa en la seguridad vial y la reducción de accidentes, sino que también se proyecta una optimización del flujo de tráfico y una mejora en la accesibilidad del transporte. Estos resultados subrayan el papel crucial que la IA puede desempeñar en la evolución de la movilidad, no solo como una tecnología disruptiva, sino como un catalizador para un sistema de transporte más seguro, eficiente y accesible para todos.

Capítulo 7

Recomendaciones

Se sugiere diseñar el sistema de control de tal manera que las acciones se basen únicamente en la información proveniente del foco frontal del vehículo. Esto se debe a la existencia de situaciones donde pueden aparecer objetos cercanos al lateral del vehículo que no representan un riesgo inminente. Al limitar las decisiones únicamente al área frontal, se puede reducir la probabilidad de falsos positivos y garantizar una respuesta más precisa y adecuada a las condiciones de riesgo reales en la carretera.

También se recomienda implementar un algoritmo de cálculo de la diferencia de velocidad entre el vehículo y los objetos circundantes utilizando secuencias de fotogramas. Esto permitirá una evaluación dinámica y precisa de la velocidad relativa entre el vehículo y los objetos en su entorno. Al incorporar esta medida en el sistema de control, se mejorará la capacidad para anticipar y responder eficazmente a situaciones de riesgo, lo que contribuirá a la seguridad y la eficiencia en la conducción.

Además, se aconseja aumentar la capacidad del modelo para reconocer una mayor cantidad y variedad de objetos en su entorno. Esto se traducirá en un sistema más versátil y adaptable, capaz de satisfacer una amplia variedad de propósitos y escenarios. Al incrementar la diversidad de objetos reconocidos, el modelo será más flexible y podrá ofrecer una mayor cobertura en términos de detección y segmentación, lo que mejorará su utilidad y eficacia en diferentes contextos de aplicación.

Por último se plantea explorar una variedad de modelos de segmentación y detección de objetos como parte del proceso de mejora continua. Al evaluar diferentes enfoques y arquitecturas de modelos, es posible identificar soluciones que mejoren aspectos clave del modelo actual, como la capacidad para reconocer una mayor variedad de objetos. Esto podría implicar la adopción de modelos más avanzados, el ajuste de parámetros o la combinación de múltiples modelos para obtener un sistema más robusto y versátil. Dicha exploración facilitará la evolución del sistema hacia un estado óptimo, adaptado a las necesidades específicas y los desafíos cambiantes del entorno de conducción autónoma.

Bibliografía

[1] J. C. Valdivieso Infante, «ANÁLISIS DE LA CONDUCCIÓN AUTÓNOMA,» Universidad Internacional del Ecuador, inf. téc., 2017.

[2] J. F. Vélez Serrano, A. B. Moreno Díaz, Á. Sánchez Calle y J. L. E. Sánchez Marín, *Visión por Computadora*. Delta Publicaciones, 2012.

[3] N. O'Mahony y S. Campbell, «Deep Learning vs. Traditional Computer Vision,» Institute of Technology Tralee, Tralee, Ireland, inf. téc.

[4] P. Meseguer González y R. López de Mántaras Badia, *Inteligencia artificial*. Editorial CSIC Consejo Superior de Investigaciones Científicas, 2017.

[5] U. García, «FutureLab,» inf. téc., 2019, [En línea]. dirección: https://futurelab.mx/redes%20neuronales/inteligencia%20artificial/2019/06/25/intro-a-redes-neuronales-pt-1/.

[6] «KeepCoding,» inf. téc., [En línea]. dirección: https://keepcoding.io/blog/funcion-de-activacion-en-deep-learning/.

[7] D. Calvo, «Función de coste – Redes neuronales,» inf. téc., 2018, [En línea]. dirección: https://www.diegocalvo.es/funcion-de-coste-redes-neuronales.

[8] C. Bonilla Carrión, *Redes Convolucionales*. Sevilla, España: Universidad de Sevilla, 2020, [En línea]. dirección: https://idus.us.es/bitstream/handle/11441/115221/TFG%20DGMyE%20Bonilla%20Carri%c3%b3n%2c%20Carmelo.pdf?sequence=1&isAllowed=y.

[9] B. Scarff, «Understanding Backpropagation,» inf. téc., 2021, [En línea]. dirección: https://towardsdatascience.com/understanding-backpropagation-abcc509ca9d0.

[10] IBM. «¿Qué son las redes neuronales convolucionales?» [En linea]. (2023), dirección: https://www.ibm.com/es-es/topics/convolutional-neural-networks.

[11] AWS. «¿Qué es el aprendizaje profundo?» [En linea]. (2023), dirección: https://aws.amazon.com/es/what-is/deep-learning/.

[12] HPE. «Aprendizaje Profundo.» [En linea]. (2023), dirección: https://www.hpe.com/mx/es/what-is/deep-learning.html.

[13] MathWorks. «Segmentación de Imágenes.» [En linea]. (2023), dirección: https://es.mathworks.com/help/images/image-segmentation.html.

[14] IBM. «¿Qué es LiDAR?» [En linea]. (2023), dirección: https://www.ibm.com/mx-es/topics/lidarl.

[15] K. Ulrich y S. Eppinger, *Diseño y desarrollo de productos*, 3rd. McGraw Hill Education, 2005.

[16] J. L. Crespo Mariño, *Módulo 1. Sesión 2: "Fase 0" (o de problema percibido) y determinación de necesidades*, 2020.

[17] N. La Serna Palomino y U. Román Concha, «Técnicas de Segmentación en Procesamiento Digital de Imágenes,» Universidad Nacional Mayor de San Marcos, inf. téc., 2007.

[18] R. Martín Villanueva, «DETECCIÓN DE OBJETOS POR COMPUTADOR,» Instituto Universitario Aeronautico, inf. téc.

[19] «U-NET : todo lo que tienes que saber sobre la red neuronal de Computer Vision,» DataScientest, inf. téc., 2022, [En linea]. dirección: https://datascientest.com/es/u-net-lo-que-tienes-que-saber.

[20] P. Potrimba, «What is R-CNN? How Does R-CNN Work?» RoboFlow, inf. téc., 2023, [En linea]. dirección: https://blog.roboflow.com/what-is-r-cnn/.

[21] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection,» University of Washington, inf. téc., 2016.

[22] P. Ruiz, «Understanding and visualizing ResNets,» Towards Data Science, inf. téc., 2018.

[23] M. Martín Romero, «Reconocimiento de objetos mediante técnicas de visión por computador y aprendizaje automático,» Escuela Técnica Superior de Ingeniería Universidad de Sevilla, inf. téc., 2022.

[24] A. Rosebrock, «Measuring distance between objects in an image with OpenCV,» Educacion calculo matematico, inf. téc., 2016, [En linea]. dirección: https://pyimagesearch.com/2016/04/04/measuring-distance-between-objects-in-an-image-with-opencv/.

[25] E. Cabrera Revuelta, «Optimización en el Posicionamiento para la realización de un Levantamiento Arquitectónico,» Universidad de Cádiz, inf. téc., 2017.

[26] K. Ogata, «Ingeniería de Control Moderna,» inf. téc., 2017.

[27] C. CC. «Ecuación de la recta punto - pendiente.» [En linea]. (2012), dirección: https://calculo.cc/temas/temas_bachillerato/primero_ciencias_sociales/funciones_elementales/teoria/interpolacion_lineal.html.

[28] D. Abdelhafiz, S. Nabavi, R. Ammar, C. Yang y J. Bi, «Residual Deep Learning System for Mass Segmentation and Classification in Mammography,» University of Connecticut, inf. téc., 2019.

[29] K. R. Ahmed, «Smart Pothole Detection Using Deep Learning Based on Dilated Convolution,» Southern Illinois University, inf. téc., 2021.

[30] M. Simon, S. Milz, K. Amende y H. M. Gross, «Complex-YOLO: An Euler-Region-Proposal for Real-Time 3D Object Detection on Point Clouds,» Ilmenau University of Technology, Ilmenau, Germany, inf. téc., 2019.

[31] R. M. Fernández, «Distancia entre Dos Puntos y Punto Medio,» Educacion calculo matematico, inf. téc., 2010, [En linea]. dirección: https://educacioncalculomatematico.blogspot.com/2010/08/distancia-entre-dos-puntos-y-punto.html.

[32] Agencia Estatal Boletín Oficial del Estado, «Reglamento nº 157 de las Naciones Unidas - Disposiciones uniformes relativas a la homologación de los vehículos de motor por lo que respecta al sistema automático de mantenimiento del carril [2021/389],» 75 a 137, 2021, [En linea]. dirección: https://www.boe.es/buscar/doc.php?id=DOUE-L-2021-80297#:~:text=El%20presente%20Reglamento%20es%20el%20primer%20paso%20normativo,con%20la%20evaluaci%C3%B3n%20de%20la%20seguridad%20del%20sistema..

[33] Organizacion Mundial de la Salud. «Traumatismos causados por el tránsito.» [En linea]. (2024), dirección: https://www.who.int/es/news-room/fact-sheets/detail/road-traffic-injuries.

Apéndice A

Métricas de resultados

Figura A.1: Matriz de confusión conjuntos de datos 1.

Reporte (Presicion - Recall - F1)

	Carro	Peaton	Ciclista	accuracy	macro avg	weighted avg
precision	1.0	1.000000	0.923077	0.996815	0.974359	0.997060
recall	1.0	0.984127	1.000000	0.996815	0.994709	0.996815
f1-score	1.0	0.992000	0.960000	0.996815	0.984000	0.996866
support	239.0	63.000000	12.000000	0.996815	314.000000	314.000000

Figura A.2: Métricas conjuntos de datos 1.

Metricas de Sensibilidad-Especificidad

	Objeto	Especificidad	Sensibilidad
0	Carro	1.000000	1.000000
1	Peaton	1.000000	0.984127
2	Ciclista	0.996689	1.000000

Figura A.3: Sensibilidad y especificidad conjuntos de datos 1.

Figura A.4: Matriz de confusión conjuntos de datos 2.

Reporte (Presicion - Recall - F1)

	Carro	Peaton	Ciclista	accuracy	macro avg	weighted avg
precision	1.0	1.000000	0.916667	0.996169	0.972222	0.996488
recall	1.0	0.977778	1.000000	0.996169	0.992593	0.996169
f1-score	1.0	0.988764	0.956522	0.996169	0.981762	0.996230
support	205.0	45.000000	11.000000	0.996169	261.000000	261.000000

Figura A.5: Métricas conjuntos de datos 2.

Metricas de Sensibilidad-Especificidad

	Objeto	Especificidad	Sensibilidad
0	Carro	1.000	1.000000
1	Peaton	1.000	0.977778
2	Ciclista	0.996	1.000000

Figura A.6: Sensibilidad y especificidad conjuntos de datos 2.

Figura A.7: Matriz de confusión conjuntos de datos 3.

Reporte (Presicion - Recall - F1)

	Carro	Peaton	Ciclista	accuracy	macro avg	weighted avg
precision	0.998501	1.000000	0.964286	0.997436	0.987595	0.997484
recall	1.000000	0.977011	1.000000	0.997436	0.992337	0.997436
f1-score	0.999250	0.988372	0.981818	0.997436	0.989813	0.997433
support	666.000000	87.000000	27.000000	0.997436	780.000000	780.000000

Figura A.8: Métricas conjuntos de datos 3.

Metricas de Sensibilidad-Especificidad

	Objeto	Especificidad	Sensibilidad
0	Carro	0.991228	1.000000
1	Peaton	1.000000	0.977011
2	Ciclista	0.998672	1.000000

Figura A.9: Sensibilidad y especificidad conjuntos de datos 3.

Figura A.10: Matriz de confusión conjuntos de datos 4.

Reporte (Presicion - Recall - F1)

	Carro	Peaton	Ciclista	accuracy	macro avg	weighted avg
precision	0.996491	0.894737	0.812500	0.98125	0.901243	0.981507
recall	0.996491	0.850000	0.866667	0.98125	0.904386	0.981250
f1-score	0.996491	0.871795	0.838710	0.98125	0.902332	0.981302
support	285.000000	20.000000	15.000000	0.98125	320.000000	320.000000

Figura A.11: Métricas conjuntos de datos 4.

Metricas de Sensibilidad-Especificidad

	Objeto	Especificidad	Sensibilidad
0	Carro	0.971429	0.996491
1	Peaton	0.993333	0.850000
2	Ciclista	0.990164	0.866667

Figura A.12: Sensibilidad y especificidad conjuntos de datos 4.

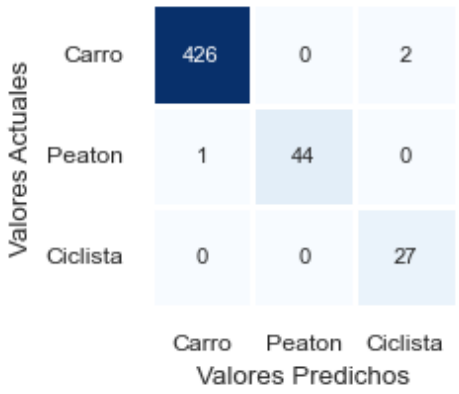


Figura A.13: Matriz de confusión conjuntos de datos 5.

Reporte (Presicion - Recall - F1)

	Carro	Peaton	Ciclista	accuracy	macro avg	weighted avg
precision	0.997658	1.000000	0.931034	0.994	0.976231	0.994271
recall	0.995327	0.977778	1.000000	0.994	0.991035	0.994000
f1-score	0.996491	0.988764	0.964286	0.994	0.983180	0.994057
support	428.000000	45.000000	27.000000	0.994	500.000000	500.000000

Figura A.14: Métricas conjuntos de datos 5.

Metricas de Sensibilidad-Especificidad

	Objeto	Especificidad	Sensibilidad
0	Carro	0.986111	0.995327
1	Peaton	1.000000	0.977778
2	Ciclista	0.995772	1.000000

Figura A.15: Sensibilidad y especificidad conjuntos de datos 5.

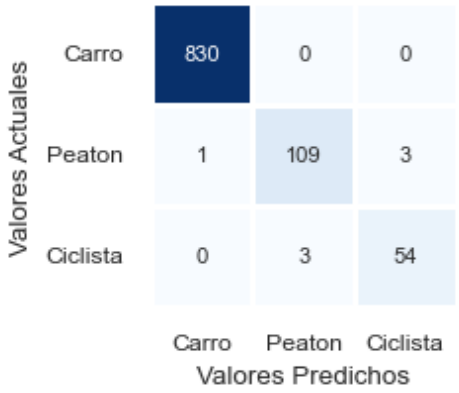


Figura A.16: Matriz de confusión conjuntos de datos 6 .

Reporte (Presicion - Recall - F1)

	Carro	Peaton	Ciclista	accuracy	macro avg	weighted avg
precision	0.998797	0.973214	0.947368	0.993	0.973126	0.992974
recall	1.000000	0.964602	0.947368	0.993	0.970657	0.993000
f1-score	0.999398	0.968889	0.947368	0.993	0.971885	0.992985
support	830.000000	113.000000	57.000000	0.993	1000.000000	1000.000000

Figura A.17: Métricas conjuntos de datos 6.

Metricas de Sensibilidad-Especificidad

	Objeto	Especificidad	Sensibilidad
0	Carro	0.994118	1.000000
1	Peaton	0.996618	0.964602
2	Ciclista	0.996819	0.947368

Figura A.18: Sensibilidad y especificidad conjuntos de datos 6.

Figura A.19: Matriz de confusión conjuntos de datos 7.

Reporte (Presicion - Recall - F1)

	Carro	Peaton	Ciclista	accuracy	macro avg	weighted avg
precision	0.998405	0.968504	0.961538	0.992308	0.976149	0.992243
recall	1.000000	0.984000	0.862069	0.992308	0.948690	0.992308
f1-score	0.999202	0.976190	0.909091	0.992308	0.961494	0.992164
support	626.000000	125.000000	29.000000	0.992308	780.000000	780.000000

Figura A.20: Métricas conjuntos de datos 7.

Metricas de Sensibilidad-Especificidad

	Objeto	Especificidad	Sensibilidad
0	Carro	0.993506	1.000000
1	Peaton	0.993893	0.984000
2	Ciclista	0.998668	0.862069

Figura A.21: Sensibilidad y especificidad conjuntos de datos 7.

Apéndice B

Resultados de casos

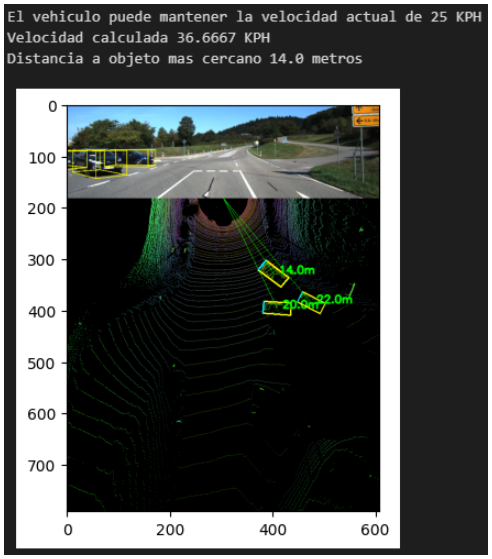


Figura B.1: Resultados imagen DS11.

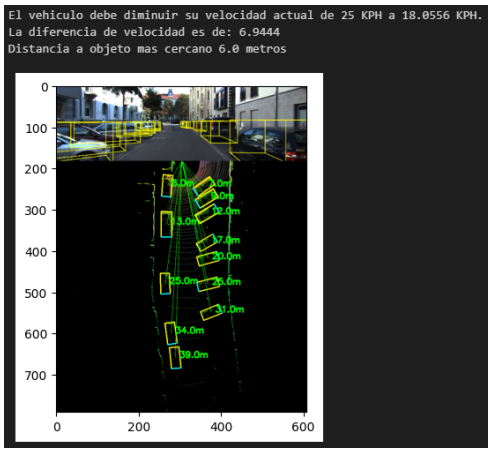


Figura B.2: Resultados imagen DS12.

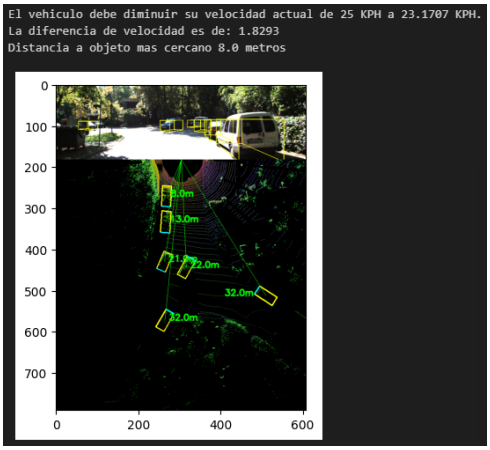


Figura B.3: Resultados imagen DS13.

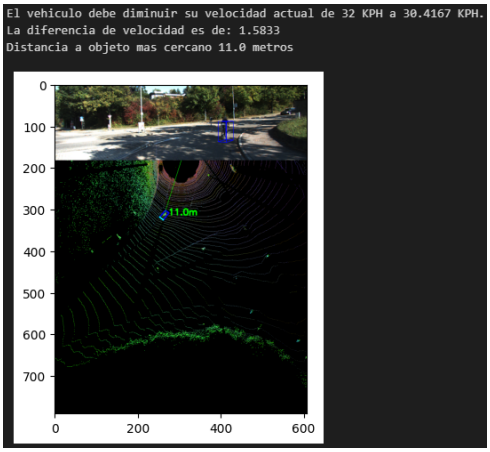


Figura B.4: Resultados imagen DS21.

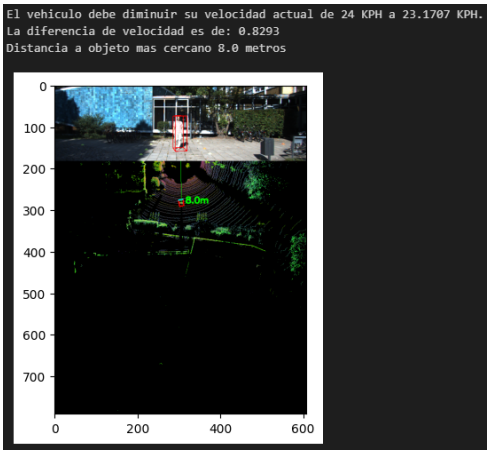


Figura B.5: Resultados imagen DS22.

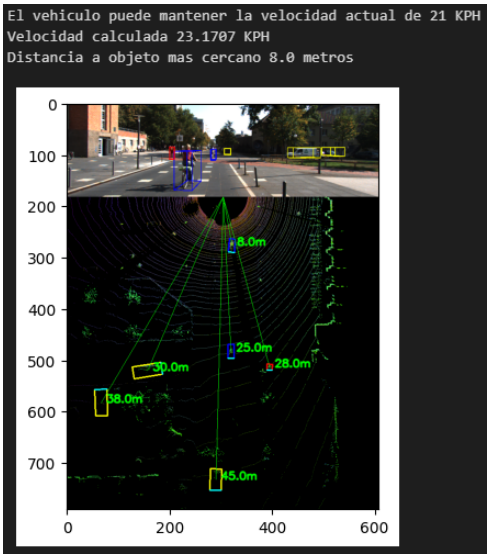


Figura B.6: Resultados imagen DS23.

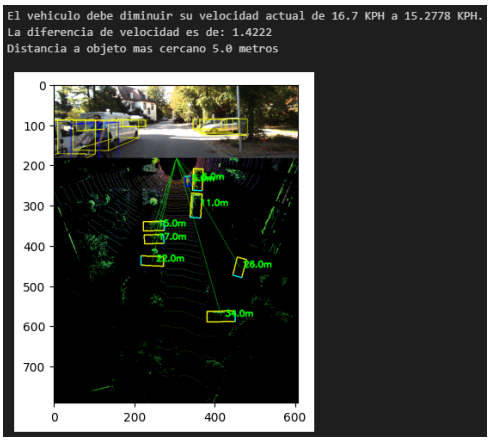


Figura B.7: Resultados imagen DS41.

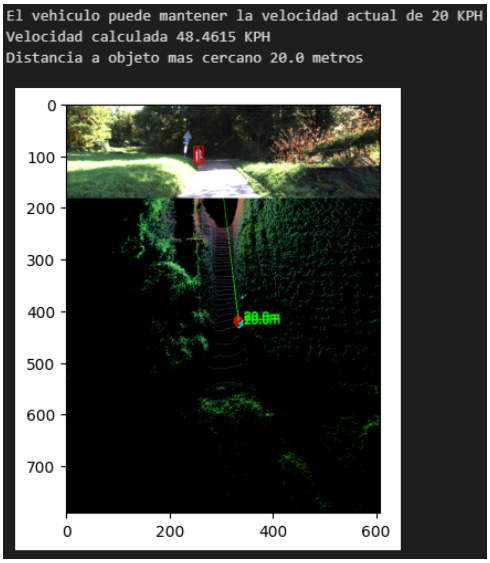


Figura B.8: Resultados imagen DS42.

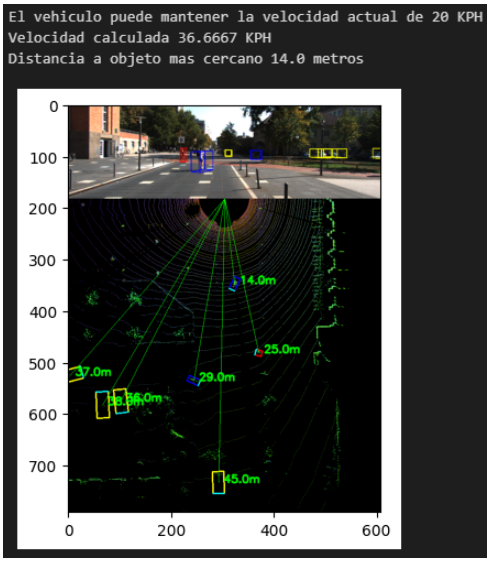


Figura B.9: Resultados imagen DS43.

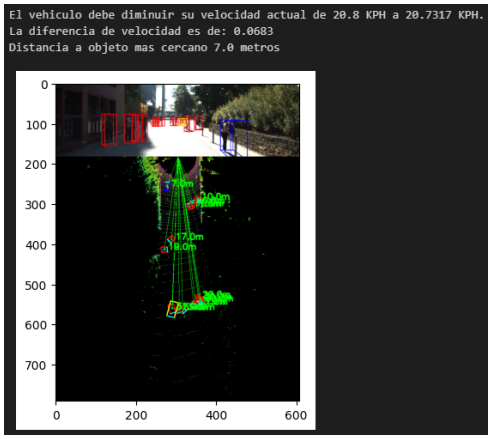


Figura B.10: Resultados imagen DS51.

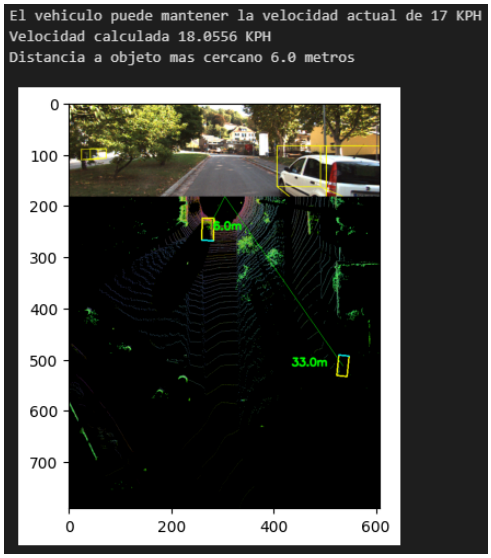


Figura B.11: Resultados imagen DS52.

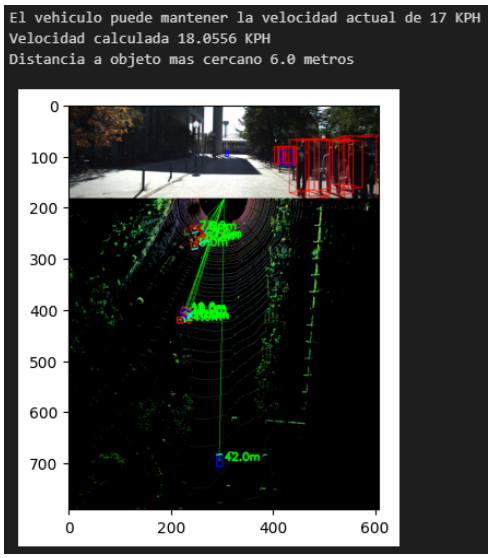


Figura B.12: Resultados imagen DS53.

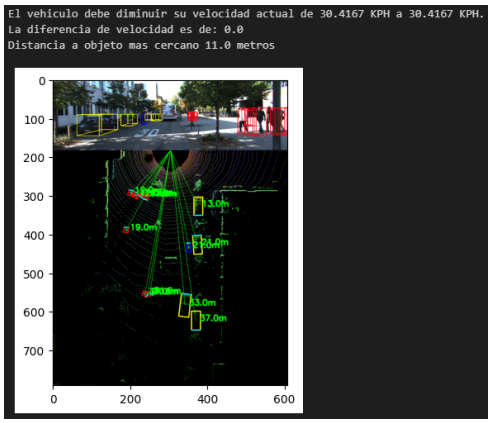


Figura B.13: Resultados imagen DS61.

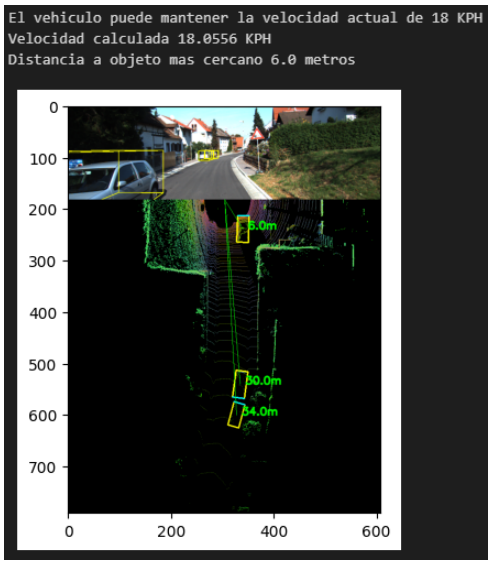


Figura B.14: Resultados imagen DS62.

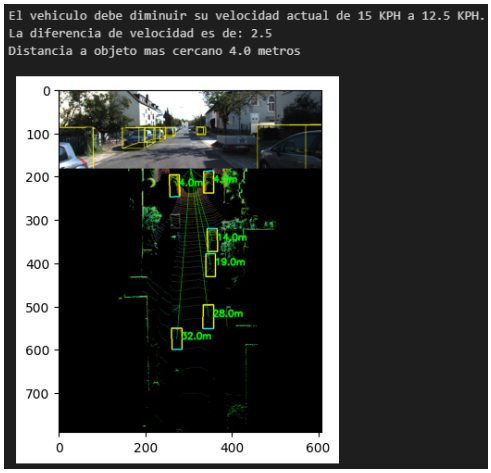


Figura B.15: Resultados imagen DS63.

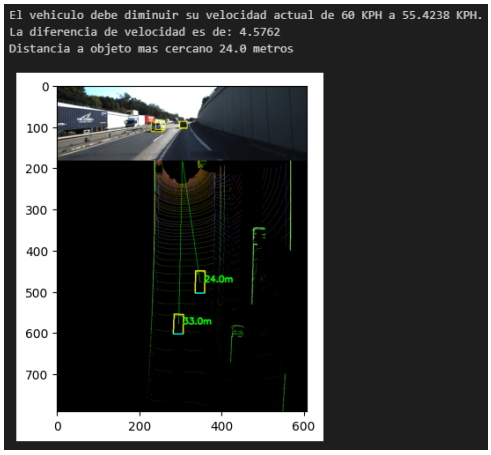


Figura B.16: Resultados imagen DS71.

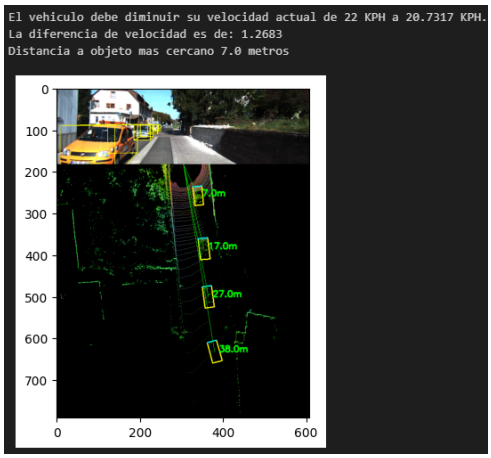


Figura B.17: Resultados imagen DS72.

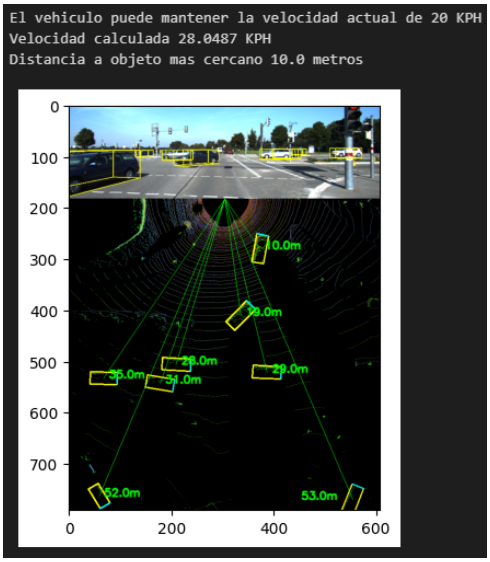


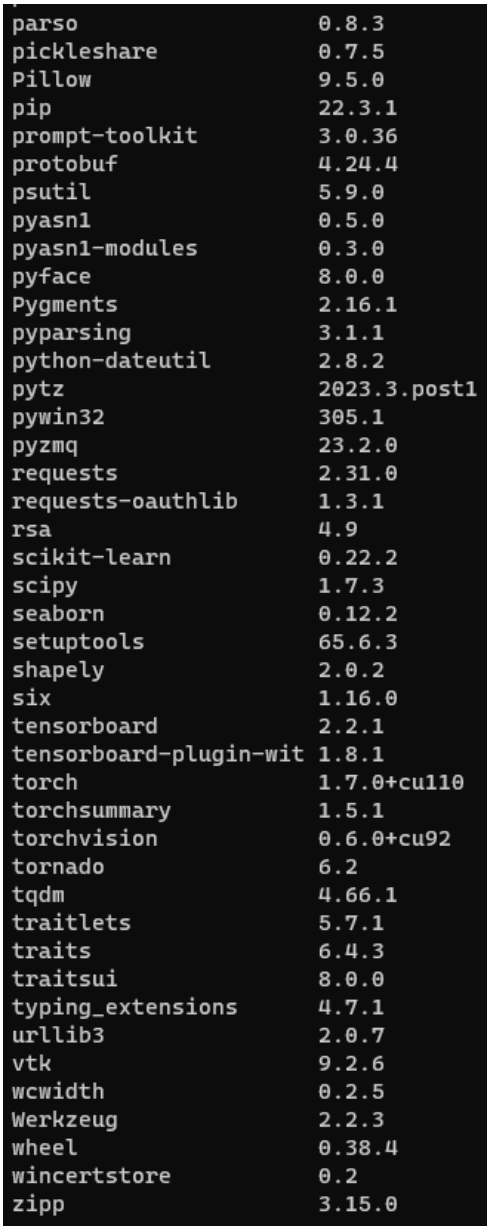
Figura B.18: Resultados imagen DS73.

Apéndice C

Configuración ambiente de trabajo

```
(complex) C:\Users\ferna>pip list
Package                Version
---------------------- ------------
absl-py                2.0.0
apptools               5.2.1
backcall               0.2.0
cachetools             4.2.4
certifi                2022.12.7
charset-normalizer     3.3.2
colorama               0.4.6
configobj              5.0.8
cycler                 0.11.0
dataclasses            0.6
debugpy                1.5.1
decorator              5.1.1
easydict               1.9
entrypoints            0.4
envisage               7.0.3
fonttools              4.38.0
future                 0.18.3
google-auth            1.35.0
google-auth-oauthlib   0.4.6
grpcio                 1.59.2
idna                   3.4
importlib-metadata     6.7.0
importlib-resources    5.12.0
ipykernel              6.15.2
ipython                7.31.1
jedi                   0.18.1
joblib                 1.3.2
jupyter_client         7.4.9
jupyter_core           4.11.2
kiwisolver             1.4.5
Markdown               3.4.4
MarkupSafe             2.1.3
matplotlib             3.5.3
matplotlib-inline      0.1.6
mayavi                 4.8.1
nest-asyncio           1.5.6
numpy                  1.21.6
oauthlib               3.2.2
opencv-python          4.2.0.34
packaging              23.2
pandas                 1.3.5
```

Figura C.1: Bibliotecas utilizadas lista 1.



parso	0.8.3
pickleshare	0.7.5
Pillow	9.5.0
pip	22.3.1
prompt-toolkit	3.0.36
protobuf	4.24.4
psutil	5.9.0
pyasn1	0.5.0
pyasn1-modules	0.3.0
pyface	8.0.0
Pygments	2.16.1
pyparsing	3.1.1
python-dateutil	2.8.2
pytz	2023.3.post1
pywin32	305.1
pyzmq	23.2.0
requests	2.31.0
requests-oauthlib	1.3.1
rsa	4.9
scikit-learn	0.22.2
scipy	1.7.3
seaborn	0.12.2
setuptools	65.6.3
shapely	2.0.2
six	1.16.0
tensorboard	2.2.1
tensorboard-plugin-wit	1.8.1
torch	1.7.0+cu110
torchsummary	1.5.1
torchvision	0.6.0+cu92
tornado	6.2
tqdm	4.66.1
traitlets	5.7.1
traits	6.4.3
traitsui	8.0.0
typing_extensions	4.7.1
urllib3	2.0.7
vtk	9.2.6
wcwidth	0.2.5
Werkzeug	2.2.3
wheel	0.38.4
wincertstore	0.2
zipp	3.15.0

Figura C.2: Bibliotecas utilizadas lista 2.

Apéndice D

Configuración Complex-YOLOv4

Información encontrada en: **https://github.com/maudzung/Complex-YOLOv4-Pytorch**

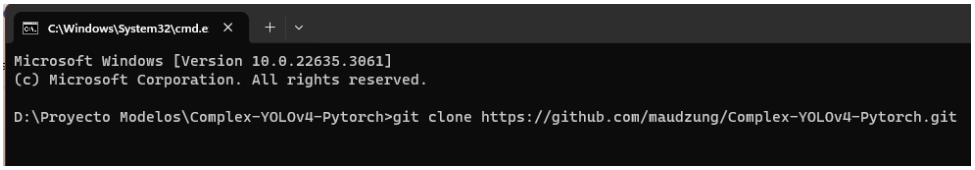


Figura D.1: Configuración paso 1.

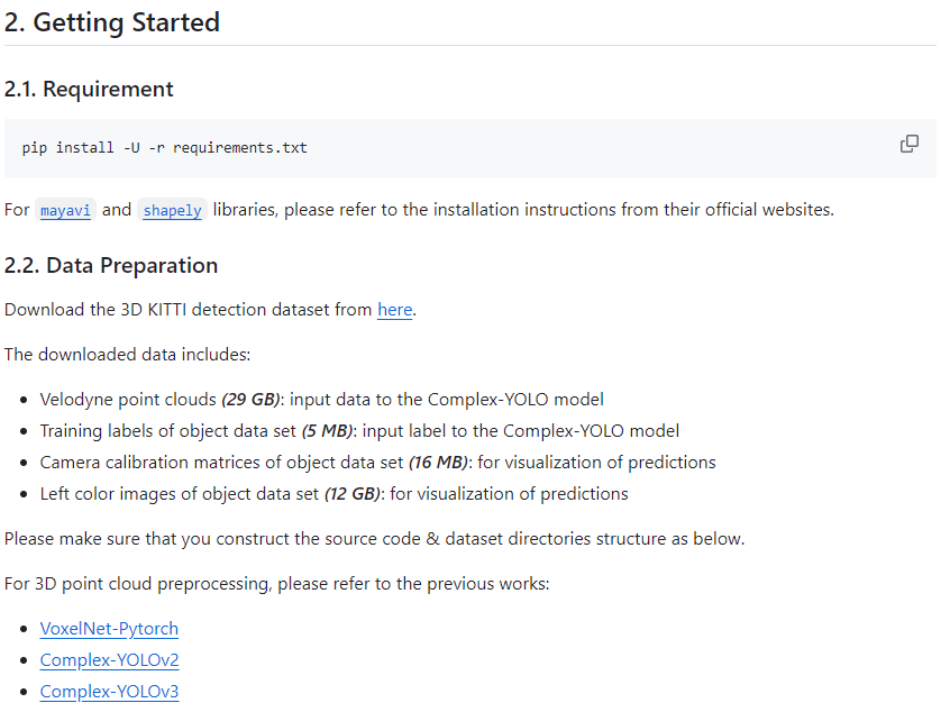


Figura D.2: Configuración paso 2.

2.4. How to run

2.4.1. Visualize the dataset (both BEV images from LiDAR and camera images)

```
cd src/data_process
```

- To visualize BEV maps and camera images (with 3D boxes), let's execute ***(the `output-width` param can be changed to show the images in a bigger/smaller window)***:

```
python kitti_dataloader.py --output-width 608
```

- To visualize mosaics that are composed from 4 BEV maps (Using during training only), let's execute:

```
python kitti_dataloader.py --show-train-data --mosaic --output-width 608
```

By default, there is ***no padding*** for the output mosaics, the feature could be activated by executing:

```
python kitti_dataloader.py --show-train-data --mosaic --random-padding --output-width 608
```

- To visualize cutout augmentation, let's execute:

```
python kitti_dataloader.py --show-train-data --cutout_prob 1. --cutout_nholes 1 --cutout_fill_value 1.
```

2.4.2. Inference

Download the trained model from ***here***, then put it to `${ROOT}/checkpoints/` and execute:

```
python test.py --gpu_idx 0 --pretrained_path ../checkpoints/complex_yolov4/complex_yolov4_mse_loss.pth
```

Figura D.3: Configuración paso 3.

Folder structure

```
${ROOT}
└── checkpoints/
    ├── complex_yolov3/
    └── complex_yolov4/
└── dataset/
    └── kitti/
        ├──ImageSets/
        │   ├── train.txt
        │   └── val.txt
        ├── training/
        │   ├── image_2/ <-- for visualization
        │   ├── calib/
        │   ├── label_2/
        │   └── velodyne/
        └── testing/
        │   ├── image_2/ <-- for visualization
        │   ├── calib/
        │   └── velodyne/
        └── classes_names.txt
└── src/
    ├── config/
    ├── cfg/
        │   ├── complex_yolov3.cfg
        │   ├── complex_yolov3_tiny.cfg
        │   ├── complex_yolov4.cfg
        │   ├── complex_yolov4_tiny.cfg
    │   ├── train_config.py
    │   └── kitti_config.py
    ├── data_process/
    │   ├── kitti_bev_utils.py
    │   ├── kitti_dataloader.py
    │   ├── kitti_dataset.py
    │   ├── kitti_data_utils.py
    │   ├── train_val_split.py
    │   └── transformation.py
    ├── models/
    │   ├── darknet2pytorch.py
    │   ├── darknet_utils.py
    │   ├── model_utils.py
    │   ├── yolo_layer.py
    └── utils/
    │   ├── evaluation_utils.py
    │   ├── iou_utils.py
    │   ├── logger.py
    │   ├── misc.py
    │   ├── torch_utils.py
    │   ├── train_utils.py
    │   └── visualization_utils.py
    ├── evaluate.py
    ├── test.py
    ├── test.sh
    ├── train.py
    └── train.sh
├── README.md
└── requirements.txt
```

Figura D.4: Configuración paso 4.

Apéndice E

Código Complex-YOLOv4

```
import argparse
import os
import time
import numpy as np
import sys
import warnings

warnings.filterwarnings("ignore", category=UserWarning)

import torch
import torch.utils.data.distributed
from tqdm import tqdm
from easydict import EasyDict as edict

sys.path.append('./')

from data_process.kitti_dataloader import create_val_dataloader
from models.model_utils import create_model
from utils.misc import AverageMeter, ProgressMeter
from utils.evaluation_utils import load_classes, post_processing_v2

working_dir = '../'

config = {
    "gpu_idx":0,
    "num_samples":20,
    "no_cuda":False,
    # Default values
    "classnames_infor_path":'../dataset/kitti/classes_names.txt',
    "batch_size":1,
    "num_workers":1,
    "arch":"darknet",
    "cfgfile":"./config/cfg/complex_yolov4.cfg",
    "use_giou_loss":True,
    "pretrained_path":"../checkpoints/complex_yolov4/complex_yolov4_mse_loss.pth",
    "saved_fn":"complexer_yolov4",
    "output_video_fn":'out_complexer_yolov4',
    "img_size":608,
    "conf_thresh":0.5,
    "nms_thresh":0.5,
    "iou_thresh":0.6,
    }

configs = edict(config)

configs.dataset_dir = os.path.join(working_dir, 'dataset', 'kitti')
configs.pin_memory = True
configs.distributed = False  # For evaluation
```

```
✓ 4.7s
:\Users\ferna\anaconda3\envs\complex\lib\site-packages\tqdm\auto.py:21: TqdmWarning: I
from .autonotebook import tqdm as notebook_tqdm
```

Figura E.1: Complex-YOLOv4 código 1.

```python
from shapely.geometry import Polygon
import data_process.kitti_bev_utils as bev_utils

def cvt_box_2_polygon(box):
    """
    :param box: an array of shape [4, 2]
    :return: a shapely.geometry.Polygon object
    """
    return Polygon([(box[i, 0], box[i, 1]) for i in range(len(box))]).buffer(0)

def get_corners_vectorize(x, y, w, l, yaw):
    """bev image coordinates format - vectorization

    :param x, y, w, l, yaw: [num_boxes,]
    :return: num_boxes x (x,y) of 4 conners
    """
    bbox2 = np.zeros((x.shape[0], 4, 2), dtype=np.float32)
    cos_yaw = np.cos(yaw)
    sin_yaw = np.sin(yaw)

    # front left
    bbox2[:, 0, 0] = x - w / 2 * cos_yaw - l / 2 * sin_yaw
    bbox2[:, 0, 1] = y - w / 2 * sin_yaw + l / 2 * cos_yaw

    # rear left
    bbox2[:, 1, 0] = x - w / 2 * cos_yaw + l / 2 * sin_yaw
    bbox2[:, 1, 1] = y - w / 2 * sin_yaw - l / 2 * cos_yaw

    # rear right
    bbox2[:, 2, 0] = x + w / 2 * cos_yaw + l / 2 * sin_yaw
    bbox2[:, 2, 1] = y + w / 2 * sin_yaw - l / 2 * cos_yaw

    # front right
    bbox2[:, 3, 0] = x + w / 2 * cos_yaw - l / 2 * sin_yaw
    bbox2[:, 3, 1] = y + w / 2 * sin_yaw + l / 2 * cos_yaw

    return bbox2
```

Figura E.2: Complex-YOLOv4 código 2.

```python
def iou_rotated_single_vs_multi_boxes_cpu(single_box, multi_boxes):
    """
    :param pred_box: Numpy array
    :param target_boxes: Numpy array
    :return:
    """

    s_x, s_y, s_w, s_l, s_im, s_re = single_box
    s_area = s_w * s_l
    s_yaw = np.arctan2(s_im, s_re)
    s_conners = bev_utils.get_corners(s_x, s_y, s_w, s_l, s_yaw)
    s_polygon = cvt_box_2_polygon(s_conners)

    m_x, m_y, m_w, m_l, m_im, m_re = multi_boxes.transpose(1, 0)
    targets_areas = m_w * m_l
    m_yaw = np.arctan2(m_im, m_re)
    m_boxes_conners = get_corners_vectorize(m_x, m_y, m_w, m_l, m_yaw)
    m_boxes_polygons = [cvt_box_2_polygon(box_) for box_ in m_boxes_conners]

    ious = []
    for m_idx in range(multi_boxes.shape[0]):
        intersection = s_polygon.intersection(m_boxes_polygons[m_idx]).area
        iou_ = intersection / (s_area + targets_areas[m_idx] - intersection + 1e-16)
        ious.append(iou_)

    return torch.tensor(ious, dtype=torch.float)
```

Figura E.3: Complex-YOLOv4 código 3.

```python
def get_batch_statistics_rotated_bbox(outputs, targets, iou_threshold):
    # print(outputs, targets, iou_threshold)
    """ Compute true positives, predicted scores and predicted labels per sample """
    batch_metrics = []
    for sample_i in range(len(outputs)):

        if outputs[sample_i] is None:
            continue

        output = outputs[sample_i]
        pred_boxes = output[:, :6]
        pred_scores = output[:, 6]
        pred_labels = output[:, -1]

        true_positives = np.zeros(pred_boxes.shape[0])

        annotations = targets[targets[:, 0] == sample_i][:, 1:]
        if len(annotations) > 0:
            target_labels = annotations[:, 0]
            detected_boxes = []
            target_boxes = annotations[:, 1:]

            for pred_i, (pred_box, pred_label) in enumerate(zip(pred_boxes, pred_labels)):

                # If targets are found break
                if len(detected_boxes) == len(annotations):
                    break

                # Ignore if label is not one of the target labels
                if pred_label not in target_labels:
                    continue

                iou, box_index = iou_rotated_single_vs_multi_boxes_cpu(pred_box, target_boxes).max(dim=0)

                if iou >= iou_threshold and box_index not in detected_boxes:
                    true_positives[pred_i] = 1
                    detected_boxes += [box_index]
        batch_metrics.append([true_positives, pred_scores, pred_labels])

    return batch_metrics
```

Figura E.4: Complex-YOLOv4 código 4.

```
def get_corners(x, y, w, l, yaw):
    bev_corners = np.zeros((4, 2), dtype=np.float32)
    cos_yaw = np.cos(yaw)
    sin_yaw = np.sin(yaw)
    # front left
    bev_corners[0, 0] = x - w / 2 * cos_yaw - l / 2 * sin_yaw
    bev_corners[0, 1] = y - w / 2 * sin_yaw + l / 2 * cos_yaw

    # rear left
    bev_corners[1, 0] = x - w / 2 * cos_yaw + l / 2 * sin_yaw
    bev_corners[1, 1] = y - w / 2 * sin_yaw - l / 2 * cos_yaw

    # rear right
    bev_corners[2, 0] = x + w / 2 * cos_yaw + l / 2 * sin_yaw
    bev_corners[2, 1] = y + w / 2 * sin_yaw - l / 2 * cos_yaw

    # front right
    bev_corners[3, 0] = x + w / 2 * cos_yaw - l / 2 * sin_yaw
    bev_corners[3, 1] = y + w / 2 * sin_yaw + l / 2 * cos_yaw

    return bev_corners
```

Figura E.5: Complex-YOLOv4 código 5.

```
import matplotlib.pyplot as plt
import cv2
import os
import time

img_dir = 'D:/Proyecto Modelos/Complex-YOLOv4-Pytorch/dataset/kitti/training/image_2'

class_names = load_classes(configs.classnames_infor_path)
device_string = 'cpu' if configs.no_cuda else 'cuda:{}'.format(configs.gpu_idx)

model = create_model(configs)
print('\n\n' + '-*=' * 30 + '\n\n')
assert os.path.isfile(configs.pretrained_path), "No file at {}".format(configs.pretrained_path)
model.load_state_dict(torch.load(configs.pretrained_path, map_location=device_string))

configs.device = torch.device('cpu' if configs.no_cuda else 'cuda:{}'.format(configs.gpu_idx))
model = model.to(device=configs.device)
```

7.7s

```
ing darknet
NFO] No error, the convolution haven't activate linear
NFO] No error, the convolution haven't activate linear
NFO] No error, the convolution haven't activate linear


=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=-*=
```

Figura E.6: Complex-YOLOv4 código 6.

```
configs.num_samples = 7000
print('Create the validation dataloader')
val_dataloader = create_val_dataloader(configs)
val_dataloader
```

✓ 31.2s

```
Create the validation dataloader

<torch.utils.data.dataloader.DataLoader at 0x273b1b5fec8>
```

Figura E.7: Complex-YOLOv4 código 7.

```
iou_mean = []
label_list = []
pred_time = []

coord_real = []
coord_pred = []

with torch.no_grad():
    for batch_idx, batch_data in enumerate(val_dataloader):
        print(batch_idx)
        start = time.time()
        img_name, imgs, targets = batch_data
        img_name = img_name[0].split('\\')[-1]
        img_path = f'{img_dir}/{img_name}'
        img = cv2.imread(img_path)
        targets[:, 2:6] *= configs.img_size
        imgs = imgs.to(configs.device, non_blocking=True)
        outputs = model(imgs)
        outputs = post_processing_v2(outputs, conf_thresh=configs.conf_thresh, nms_thresh=configs.nms_thresh)
        output = outputs[0]

        # plt.imshow(img)
        # plt.show()

        for target in targets:
            x, y, w, l, im, re = target[2:]
            yaw = np.arctan2(im, re)

            coord_real.append([x, y, w, l, im, re, yaw])

            img_target = np.zeros([608, 608],dtype=np.uint8)
            bev_corners = get_corners(x, y, w, l, yaw)
            corners_int = bev_corners.reshape(-1, 1, 2).astype(int)
            cv2.fillPoly(img_target, [corners_int], 1)
            label = int(target[:2][1])



            try:

                for pred in outputs[0]:
                    x, y, w, l, im, re = pred[:6]
                    yaw = np.arctan2(im, re)
                    img_pred = np.zeros([608, 608],dtype=np.uint8)
                    bev_corners = get_corners(x, y, w, l, yaw)
                    corners_int = bev_corners.reshape(-1, 1, 2).astype(int)
                    cv2.fillPoly(img_pred, [corners_int], 1)
                    overlap = img_target * img_pred  # Logical AND
                    union = (img_target + img_pred) > 0  # Logical OR
                    iou = overlap.sum() / float(union.sum())

                    if iou > 0.6:
                        label_pred = int(pred[-1])
                        end = time.time()
```

Figura E.8: Complex-YOLOv4 código 8.

```
                    # print(f"Real: {label}, Predict: {label_pred}")

                    # print(target)
                    # print(label_pred)

                    # print(label)
                    # plt.imshow(img_target, 'gray')
                    # plt.show()
                    # print(label_pred)
                    # plt.imshow(img_pred, 'gray')
                    # plt.show()

                    # plt.imshow(overlap, 'gray')
                    # plt.show()

                    # plt.imshow(union, 'gray')
                    # plt.show()

                    iou_mean.append(iou)
                    label_list.append([label, label_pred])
                    pred_time.append(end-start)

                    coord_pred.append([x, y, w, l, im, re, yaw])

                    # print(iou)

                    break
                except: pass

            #     break

            # break
✓ 5m 12.8s
```

Figura E.9: Complex-YOLOv4 código 9.

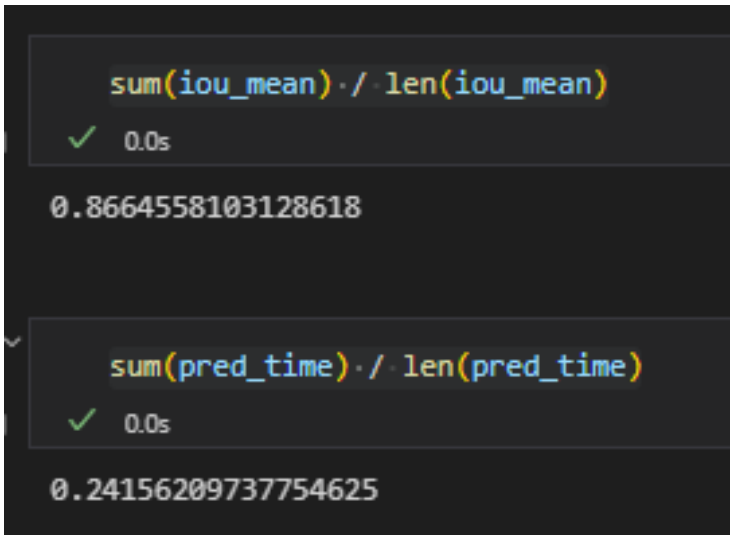


Figura E.10: Complex-YOLOv4 código 10.

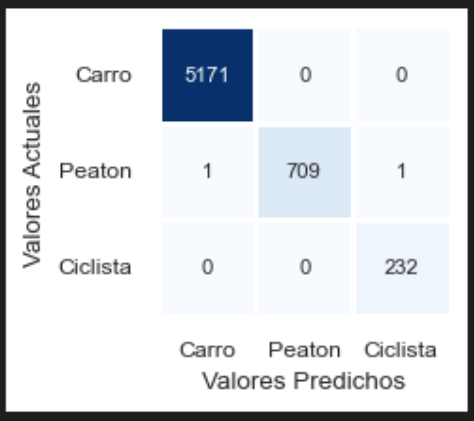
```
import seaborn as sb
import pandas as pd

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import precision_recall_fscore_support

def plot_scores(data):
    df = pd.DataFrame(  data,
                        columns = ['label','predict'])
    y_true = df['label'].values
    y_pred = df['predict'].values
    cm = confusion_matrix(y_true, y_pred)
    cm_df = pd.DataFrame(cm,
                         index = ['Carro','Peaton','Ciclista'],
                         columns = ['Carro','Peaton','Ciclista'])
    plt.figure(figsize=(2,2))
    sb.set(font_scale=0.8)
    sb.heatmap(cm_df,annot_kws={"size": 8},
            annot=True, linewidths=1,
            cbar=False, cmap="Blues",
            fmt='g')
    plt.ylabel('Valores Actuales')
    plt.yticks(rotation=0)
    plt.xlabel('Valores Predichos')
    print('Matriz de confusion')
    plt.show()
    print('Reporte (Presicion - Recall - F1)')
    target_names = ['Carro','Peaton','Ciclista']
    report = classification_report(y_true, y_pred, target_names=target_names, output_dict=True)
    display(pd.DataFrame.from_dict(report))
    print('Metricas de Sensibilidad-Especificidad')
    res = []
    objets = ["Carro", "Peaton", "Ciclista"]
    for l in [0,1,2]:
        prec,recall,_,_ = precision_recall_fscore_support(np.array(y_true)==l,
                                                          np.array(y_pred)==l,
                                                          pos_label=True,average=None)
        res.append([objets[l],recall[0],recall[1]])

    sensitivity = pd.DataFrame(res,columns = ['Objeto','Especificidad','Sensibilidad'])

    display(sensitivity)
```

Figura E.11: Complex-YOLOv4 código 11.

```
plot_scores(label_list)
```

✓ 0.4s

Matriz de confusion



Reporte (Presicion - Recall - F1)

	Carro	Peaton	Ciclista	accuracy	macro avg	weighted avg
precision	0.999807	1.000000	0.995708	0.999673	0.998505	0.999674
recall	1.000000	0.997187	1.000000	0.999673	0.999062	0.999673
f1-score	0.999903	0.998592	0.997849	0.999673	0.998781	0.999673
support	5171.000000	711.000000	232.000000	0.999673	6114.000000	6114.000000

Metricas de Sensibilidad-Especificidad

	Objeto	Especificidad	Sensibilidad
0	Carro	0.99894	1.000000
1	Peaton	1.00000	0.997187
2	Ciclista	0.99983	1.000000

Figura E.12: Complex-YOLOv4 reultados de sensibilidad y especificidad.

Apéndice F

Código Calculo de Distancia

```python
index = '0000000000'
img_rgb, img_lidar = load_img_only(index)
rgbArray = np.zeros((608,608,3), 'uint8')
rgbArray[..., 0] = img_lidar[0]*256
rgbArray[..., 1] = img_lidar[1]*256
rgbArray[..., 2] = img_lidar[2]*256
rgb_pil = Image.fromarray(rgbArray)
rgb_tensor = convert_tensor(rgb_pil).to(device=configs.device)
rgb_tensor = torch.unsqueeze(rgb_tensor, dim=0)
outputs = model(rgb_tensor)
detections = post_processing_v2(outputs, conf_thresh=configs.conf_thresh, nms_thresh=configs.nms_thresh)

img_bev = rgb_tensor.squeeze() * 255
img_bev = img_bev.cpu().permute(1, 2, 0).numpy().astype(np.uint8)
img_bev = cv2.resize(img_bev, (configs.img_size, configs.img_size))

img_detections = []  # Stores detections for each image index
img_detections.extend(detections)

for detections in img_detections:
    if detections is None:
        continue
    # Rescale boxes to original image
    detections = rescale_boxes(detections, configs.img_size, img_bev.shape[:2])
    for x, y, w, l, im, re, *_, cls_pred in detections:
        # print(f'{detections} \n')
        yaw = np.arctan2(im, re)
        x_real = (x-304)* 25/304
        y_real = y*50/608
        dist = np.round(np.sqrt(np.dot(x_real, x_real)+np.dot(y_real, y_real)), 0)
        print(f"Distance: {dist}")
        print(f"X: {x}, Y:{y}, W:{w}, L:{l}, IM:{im}, RE:{re}, YAW:{yaw}")
        # Draw rotated box
        kitti_bev_utils.drawRotatedBox(img_bev, x, y, w, l, yaw, cnf.colors[int(cls_pred)])
        cv2.line(img_bev, (x, y), (304, 0), (0, 255, 0), thickness=1)
        if x > 500:
            cv2.putText(img_bev, f'{dist}m', (x-100, y), cv2.FONT_HERSHEY_SIMPLEX , 0.7, (0, 255, 0), 2, cv2.LINE_AA)
        else:
            cv2.putText(img_bev, f'{dist}m', (x+10, y), cv2.FONT_HERSHEY_SIMPLEX , 0.7, (0, 255, 0), 2, cv2.LINE_AA)

objects_pred = predictions_to_kitti_format(img_detections, calib, img_rgb.shape, configs.img_size)
img_rgb = show_image_with_boxes(img_rgb, objects_pred, calib, False)

out_img = merge_rgb_to_bev(img_rgb, img_bev, output_width=608)
plt.imshow(out_img)
```

Figura F.1: Código de calculo de distancia.

```
Distance: 6.0
X: 283.6962585449219, Y:73.96898651123047, W:10.429625511169434, L:25.352535247802734, IM:-0.22937127947807312, RE:0.9613816738128662, YAW:-0.2342066466808319
Distance: 11.0
X: 227.70643615722656, Y:106.02310943603516, W:12.604942321777344, L:15.535676956176758, IM:0.25609543919563293, RE:0.7091596126556396, YAW:0.3465513586997986
Distance: 15.0
X: 359.396240234375, Y:168.1300811767578, W:25.179655075073242, L:59.902793884277344, IM:-0.4624607563018799, RE:0.8652252554893494, YAW:-0.49086326360702515
Distance: 40.0
X: 603.9463500976562, Y:380.8551940917969, W:22.820390701293945, L:51.50769805908203, IM:0.3450571894645691, RE:0.5498044490814209, YAW:0.5604667067527771
```

Figura F.2: Resultado de valores de calculo de distancia para cada objeto.

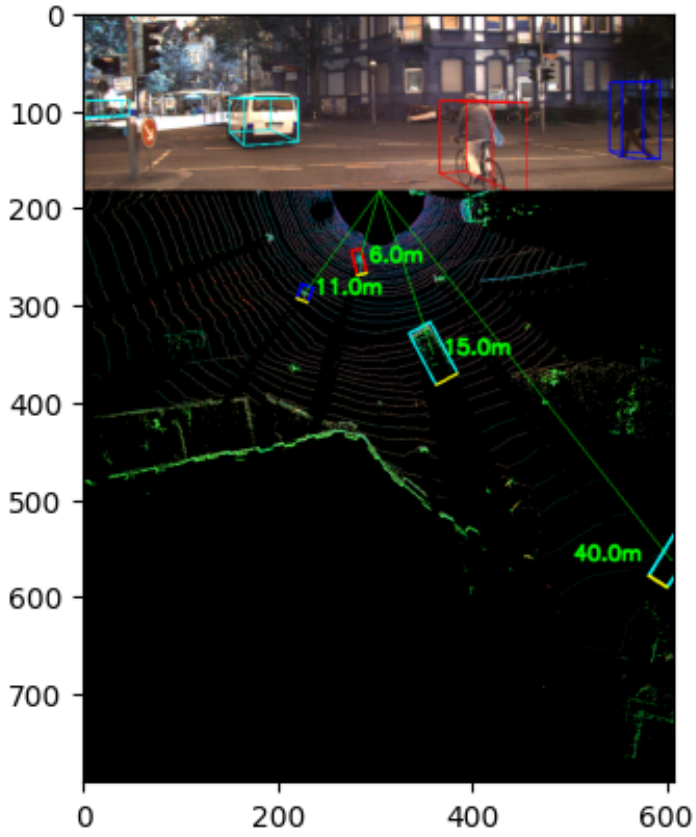


Figura F.3: Resultado final de calculo de distancia.

Apéndice G

Código Control Lineal

```python
def control(current_vel, dist):
    if dist <= 2:
        law_vel = 0
        action = "Parar"
    elif dist < 3.1:
        law_vel = (((10-7.2)*dist)/(3.1-2))+2.1091
        if current_vel > law_vel:
            action = "Disminuir"
        else:
            action = "Mantener"
    elif dist < 6.7:
        law_vel = (((20-10)*dist)/(6.7-3.1))+1.3889
        if current_vel > law_vel:
            action = "Disminuir"
        else:
            action = "Mantener"
    elif dist < 10.8:
        law_vel = (((30-20)*dist)/(10.8-6.7))+3.6585
        if current_vel > law_vel:
            action = "Disminuir"
        else:
            action = "Mantener"
    elif dist < 15.6:
        law_vel = (((40-30)*dist)/(15.6-10.8))+7.5
        if current_vel > law_vel:
            action = "Disminuir"
        else:
            action = "Mantener"
    elif dist < 20.8:
        law_vel = (((50-40)*dist)/(20.8-15.6))+10
        if current_vel > law_vel:
            action = "Disminuir"
        else:
            action = "Mantener"
    elif dist < 26.7:
        law_vel = (((60-50)*dist)/(26.7-20.8))+14.7458
        if current_vel > law_vel:
            action = "Disminuir"
        else:
            action = "Mantener"
    else:
        law_vel = current_vel
        action = "Mantener"
    return action, np.round(law_vel, 4)
```

Figura G.1: Código árbol de decisiones control lineal.

Apéndice H

Código final del sistema.

```
start = time.time()

index = '000439'
vel_vehiculo = 36

img_rgb, img_lidar = load_img_only(index)
rgbArray = np.zeros((608,608,3), 'uint8')
rgbArray[..., 0] = img_lidar[0]*256
rgbArray[..., 1] = img_lidar[1]*256
rgbArray[..., 2] = img_lidar[2]*256
rgb_pil = Image.fromarray(rgbArray)
rgb_tensor = convert_tensor(rgb_pil).to(device=configs.device)
rgb_tensor = torch.unsqueeze(rgb_tensor, dim=0)
outputs = model(rgb_tensor)
detections = post_processing_v2(outputs, conf_thresh=configs.conf_thresh, nms_thresh=configs.nms_thresh)

img_bev = rgb_tensor.squeeze() * 255
img_bev = img_bev.cpu().permute(1, 2, 0).numpy().astype(np.uint8)
img_bev = cv2.resize(img_bev, (configs.img_size, configs.img_size))

img_detections = []  # Stores detections for each image index
img_detections.extend(detections)

obj_detections = []

for detections in img_detections:
    if detections is None:
        continue
    detections = rescale_boxes(detections, configs.img_size, img_bev.shape[:2])
    for x, y, w, l, im, re, *_, cls_pred in detections:
        yaw = np.arctan2(im, re)
        x_real = (x-304) * 25/304
        y_real = y * 50/608
        dist = np.round(np.sqrt(np.dot(x_real, x_real)+np.dot(y_real, y_real)), 0)
        obj_detections.append(dist)
        kitti_bev_utils.drawRotatedBox(img_bev, x, y, w, l, yaw, cnf.colors[int(cls_pred)])
        cv2.line(img_bev, (x, y), (304, 0), (0, 255, 0), thickness=1)
        if x > 500:
            cv2.putText(img_bev, f'{dist}m', (x-100, y), cv2.FONT_HERSHEY_SIMPLEX , 0.7, (0, 255, 0), 2, cv2.LINE_AA)
        else:
            cv2.putText(img_bev, f'{dist}m', (x+10, y), cv2.FONT_HERSHEY_SIMPLEX , 0.7, (0, 255, 0), 2, cv2.LINE_AA)

start2 = time.time()

objects_pred = predictions_to_kitti_format(img_detections, calib, img_rgb.shape, configs.img_size)
img_rgb = show_image_with_boxes(img_rgb, objects_pred, calib, False)

out_img = merge_rgb_to_bev(img_rgb, img_bev, output_width=608)
out_img = cv2.cvtColor(out_img, cv2.COLOR_BGR2RGB)
plt.imshow(out_img)


obj_min = np.array(obj_detections).min()
```

Figura H.1: Código final parte 1.

```python
accion, velocidad = control(vel_vehiculo, obj_min)


if accion == "Disminuir":
    vel_dism = vel_vehiculo - velocidad
    print(f"El vehiculo debe diminuir su velocidad actual de {vel_vehiculo} KPH a {velocidad} KPH.")
    print(f"La diferencia de velocidad es de: {np.round(vel_dism, 4)}")
    print(f"Distancia a objeto mas cercano {obj_min} metros")
elif accion == "Parar":
    print("El vehiculo se debe detener inmediatamente.")
    print(f"Distancia a objeto mas cercano {obj_min} metros")
else:
    print(f"El vehiculo puede mantener la velocidad actual de {vel_vehiculo} KPH")
    print(f"Distancia a objeto mas cercano {obj_min} metros")

end2 = time.time()

print(f"Tiempo de procesamiento {np.round(start2-start, 4)}")

print(f"Tiempo de accion {np.round(end2-start2, 4)}")

print(f"Tiempo de procesaminto {np.round(time.time()-start, 4)}")
```

Figura H.2: Código final parte 2.

```
El vehiculo debe diminuir su velocidad actual de 36 KPH a 28.0487 KPH.
La diferencia de velocidad es de: 7.9513
Distancia a objeto mas cercano 10.0 metros
Tiempo de procesamiento 0.5105
Tiempo de accion 0.0402
Tiempo de procesaminto 0.5507
```

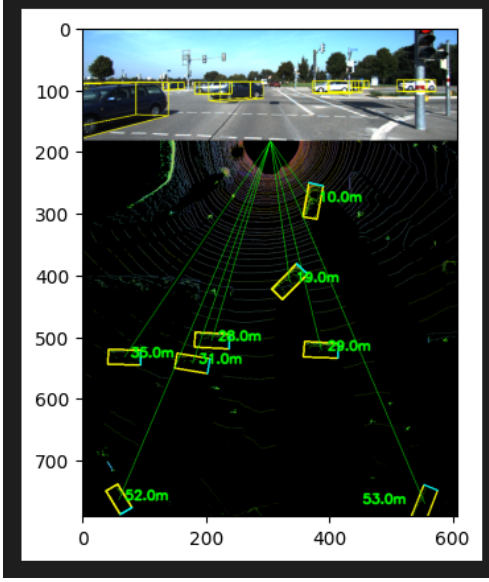


Figura H.3: Código final resultado.