Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Mecatrónica

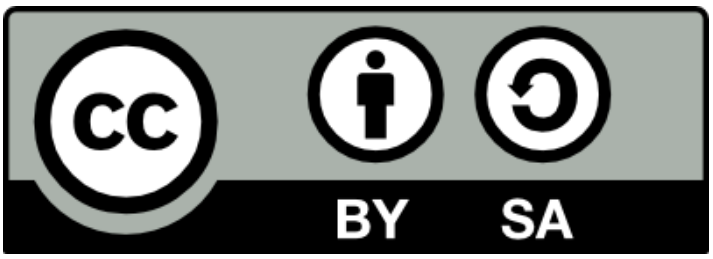
Diseño de un sistema de control de vehículos autónomos mediante aprendizaje por refuerzo con garantías de seguridad

Informe de Proyecto de Graduación para optar por el título de

Ingeniero/a en Mecatrónica con el grado académico de Licenciatura

Dryan Alonso López Fernández

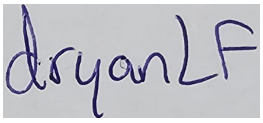
Cartago, 5 de agosto de 2025



Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos 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 graduación realizado y por el contenido del correspondiente informe final.

Dryan Alonso López Fernández



Cartago, 5 de agosto de 2025

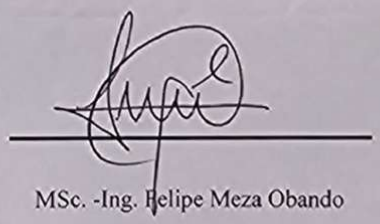
Céd: 6-0471-0186

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: Dryan López Fernández.

Proyecto: Desarrollo de un sistema de control de vehículos autónomos mediante aprendizaje por refuerzo con garantías de seguridad.



MSc. -Ing. Felipe Meza Obando

Asesor

Lugar y fecha de la presentación: Cartago, 5 de agosto de 2025.

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: Dryan López Fernández.

Proyecto: Desarrollo de un sistema de control de vehículos autónomos mediante aprendizaje por refuerzo con garantías de seguridad.

Miembros del jurado evaluador

Dra. -Ing. Gabriela Ortiz León	MSc. Marta Eugenia Vílchez Monge
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.

Lugar y fecha de la presentación: Cartago, 5 de agosto de 2025.

Resumen

Los métodos de control convencionales presentan limitaciones para responder de forma eficaz en entornos cambiantes e inciertos, lo que impulsa el desarrollo de enfoques más seguros y adaptativos. No obstante, validar estas soluciones directamente en escenarios reales implica altos costos y largos tiempos de validación, por lo que se deben explorar métodos alternativos.

El presente proyecto consiste en el desarrollo de un sistema de control para vehículos autónomos mediante aprendizaje por refuerzo profundo. Se implementó un modelo dinámico vehicular validado en la plataforma Webots, que permitió entrenar un agente capaz de mantenerse dentro del carril, evitar colisiones y regular su velocidad.

Posteriormente, se desarrolló un sistema de percepción basado en visión por computadora, encargado de detectar carriles y estimar variables como la desviación lateral, ángulo de conducción, la curvatura de la vía, así como la distancia y velocidad a otros vehículos, las cuales fueron validadas estadísticamente. El sistema completo fue evaluado en un entorno tridimensional simulado, mostrando un buen desempeño general durante su validación, con una desviación máxima de 0.73 m y un error máximo del ángulo de conducción de 0.28 rad.

Palabras clave: vehículos autónomos, modelo dinámico, aprendizaje por refuerzo, control seguro, visión por computadora, simulación 3D.

Abstract

Conventional control methods show limitations in effectively responding to dynamic and uncertain environments, which drives the development of safer and more adaptive approaches. However, validating these solutions directly in real-world scenarios involves high costs and long validation times, making it necessary to explore alternative methods.

This project focuses on the development of a control system for autonomous vehicles using deep reinforcement learning. A dynamic vehicle model was implemented and validated in the Webots platform, enabling the training of an agent capable of staying within its lane, avoiding collisions, and regulating speed.

Subsequently, a perception system based on computer vision was developed to detect lanes and estimate variables such as lateral deviation, heading angle, road curvature, and the distance and speed of surrounding vehicles. These measurements were statistically validated. The complete system was evaluated in a simulated three-dimensional environment, showing good overall performance during validation, with a maximum lateral deviation of 0.73 m and a maximum heading angle error of 0.28 rad.

Keywords: autonomous vehicles, dynamic model, reinforcement learning, safe control, computer vision, 3D simulation.

A mis padres, quienes siempre me han apoyado en todos los aspectos de mi desarrollo profesional y personal. Sin ellos, mucho no hubiera sido posible.

Agradecimientos

Agradezco a Ale, Lulu e Isaac, por brindarme tan valiosa compañía a lo largo de los años. A Emanuel, David, Mafe y Glenn por todas las quedadas y trasnochadas que le daban sentido a la vida como estudiante, así como a Kenneth, Yohell y Galeano, por todos los días de vóleibol que pasamos juntos.

Agradecimiento especial al sistema público de enseñanza de Costa Rica y al Sistema Nacional de los Colegios Científicos de Costa Rica, por las oportunidades que me brindaron como estudiante.

Y mi mayor agradecimiento hacia mis padres, quienes me han brindado apoyo en todo momento y cultivaron en mí los valores que han hecho posible la culminación de este período de mi vida.

Dryan Alonso López Fernández

Cartago, 11 de agosto de 2025

Índice general

Índice de figuras

Índice de tablas

Lista de símbolos y abreviaciones

Abreviaciones

AC	Actor-Critic
ADAS	Advanced Driver Assistance Systems
ASPP	Atrous Spatial Pyramid Pooling
AV	Autonomous Vehicle
BEV	Bird's Eye View
BLVS	Bilateral Line Voting Scheme
CARLA	Car Learning to Act
CLAHE	Contrast Limited Adaptive Histogram Equalization
CNN	Convolutional Neural Network
CS2AC	Centro de Investigación en Supervisión, Seguridad y Control Automático
DBN	Deep Belief Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DCGNN	Dynamic Continuous Graph Neural Network
DDPG	Deep Deterministic Policy Gradient
DETR	DEtection TRansformer
DL	Deep Learning
DPM	Deformable Part Model
DQN	Deep Q Network
DRL	Deep Reinforcement Learning
DT	Decision Tree
FCOS	Fully Convolutional One-Stage Object Detection
FOV	Field of View
GPR	Gaussian Process Regression
HG	Hypothesis Generation
HOG	Histogram of Oriented Gradients
HV	Hypothesis Verification
IA	Inteligencia Artificial
KNN	K-Nearest Neighbors
LaneNet	Lane Estimation Network
LBP	Local Binary Patterns
LDNet	Lane Detection Network
LIDAR	Light Detection and Ranging
LSD	Line Segment Detector

MA-MFFC	Multi-Attention Multi-Fusion Feature Clustering
MDV	Modelo Dinámico Vehicular
ML	Machine Learning
ModEL	Modular End-to-End Learning
MRF	Markov Random Field
PCA	Principal Component Analysis
PDV	Point-based Dynamic Voxelization
PIXOR	Pixel-wise Oriented Region
PPO	Proximal Policy Optimization
PV-RCNN	Point-Voxel Region-based Convolutional Neural Network
RANSAC	Random Sample Consensus
RCNN	Region-based Convolutional Neural Network
RCS	Radar Cross Section
Ri-Fusion	Range-Image Fusion
ROI	Region of Interest
RS-Lane	ResNeSt-based Lane Detection
RT-DETR	Real-Time Detection Transformer
SAC	Soft Actor-Critic
SAD	Self-Attention Distillation
SCNN	Spatial Convolutional Neural Network
SD	Sistema Desarrollado
SEB	Semantic Enhancement Block
SIFT	Scale-Invariant Feature Transform
SNR	Signal-to-Noise Ratio
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
TORCS	The Open Racing Car Simulator
TTC	Time To Collision
UPC	Universidad Politécnica de Cataluña
VGG	Visual Geometry Group
YOLO	You Only Look Once
YOLOv4	You Only Look Once version 4
YOLOv5	You Only Look Once version 5
YOLOv7	You Only Look Once version 7
YOLOX	You Only Look Once – Extended version

Notación general

$[\dot{V}]$	Aceleración resultante del vehículo.
β	Peso del término de entropía en la función de pérdida.
$\beta(\delta)$	Ángulo de deslizamiento del centro de gravedad.
δ	Control del ángulo.
$\hat{D}$	Valor de la función de ventaja.
dr	Distancia relativa hasta el vehículo más cercano.
e_1	Desviación lateral del vehículo.
e_2	Error en el ángulo de guiñada.

e_3	Error entre la velocidad longitudinal y la de referencia.
E	Error máximo aceptable para la medición.
e_n	Medición del error n-ésimo.
ϵ	Parámetro de recorte.
S	Vector de estado.
F_F	Fuerza longitudinal en la llanta delantera.
F_R	Fuerza longitudinal en la llanta trasera.
γ	Factor de descuento en aprendizaje por refuerzo.
$\hat{G}$	Valor de retorno.
$\mathcal{H}$	Pérdida por entropía.
K	Número de épocas por iteración de entrenamiento.
$\mathcal{L}^{-1}$	Transformada inversa de Laplace.
l_f	Distancia del centro de gravedad al eje de las llantas frontales.
l_r	Distancia del centro de gravedad al eje de las llantas traseras.
m	Masa del vehículo.
M	Total de mini-batches.
N	Número de muestras necesario para la comprobación de la confianza.
ϕ	Parámetros del crítico V.
$y_p(x)$	Polinomio de grado 2
ψ	Ángulo del vehículo con respecto al eje X global.
$\dot{\psi}$	Velocidad angular de guiñada del vehículo.
R	Valor de la recompensa cada iteración.
ρ	Curvatura de la autopista.
s	Variable en el dominio de Laplace.
σ	Desviación estándar de la medición.
T	Horizonte de experiencia en pasos de simulación.
τ_1	Constante de aceleración.
τ_2	Constante de frenado.
τ	Control de aceleración.
θ	Parámetros del actor π.
TTC	Tiempo para colisión (Time To Collision).
V	Velocidad resultante del vehículo.
vr	Velocidad relativa respecto al vehículo más cercano.
w	Peso asociado a la entropía.
w_n	Peso asociado al error n-ésimo.
$\dot{X}$	Velocidad longitudinal del vehículo.
xr	Posición longitudinal relativa al vehículo.
$\dot{Y}$	Velocidad lateral del vehículo.
yr	Posición lateral relativa al vehículo.

Capítulo 1

Introducción

1.1 Entorno del proyecto

En este proyecto, desarrollado en el Centro de Investigación en Supervisión, Seguridad y Control Automático (CS2AC) de la Universidad Politécnica de Cataluña (UPC), se emplea aprendizaje por refuerzo profundo para controlar un vehículo autónomo en entornos dinámicos.

1.2 Defínción del problema

1.2.1 Generalidades

Los sistemas de conducción tradicionales enfrentan importantes desafíos al operar en entornos dinámicos, caracterizados por cambios repentinos en el tráfico, condiciones climáticas adversas y comportamientos impredecibles de otros usuarios de la vía. Ante esta complejidad, surge la necesidad, por parte del centro de investigación CS2AC-UPC, de explorar estrategias más flexibles y adaptativas que permitan a los vehículos tomar decisiones robustas en tiempo real.

No obstante, la validación de los sistemas de conducción autónoma mediante métodos convencionales resulta difícil, debido al tiempo y recursos que implica su entrenamiento y verificación en escenarios reales.

1.2.2 Justificación

El desarrollo de sistemas de control más adaptativos para vehículos autónomos es crucial no solo a nivel técnico, sino también social y económico. Estudios muestran que hasta un 94% de los accidentes de tránsito se deben a errores humanos [1], por lo que la implementación de vehículos autónomos podría reducir drásticamente la siniestralidad en carretera.

Esta reducción tendría un impacto positivo en la seguridad ciudadana y también en el ámbito económico. En algunos países, los accidentes representan pérdidas de cientos de miles de millones de dólares anuales, incluyendo costos médicos, legales y materiales [2]. Esto evidencia la urgencia de implementar soluciones de control más eficaces para estos sistemas.

Además, resulta crucial validar nuevas estrategias de control en entornos de simulación antes de su implementación en vehículos reales, ya que conlleva un gran desafío estadístico la validación de dichos sistemas mediante métodos convencionales [3]. La simulación permite evaluar la seguridad, eficiencia y robustez del sistema sin los riesgos asociados a pruebas en carretera, reduciendo considerablemente los costos y tiempos de desarrollo. Herramientas como el simulador CARLA, entre otros ambientes de simulación, han demostrado ser efectivas para entrenar y evaluar controladores de conducción autónoma en escenarios complejos, permitiendo optimizar políticas de control y detectar fallos potenciales en condiciones controladas antes de proceder a su aplicación práctica [4].

1.2.3 Diagrama de Ishikawa

En la figura 1.1 se muestra el diagrama de Ishikawa relacionado a las limitaciones para la implementación de sistemas de conducción autónoma.

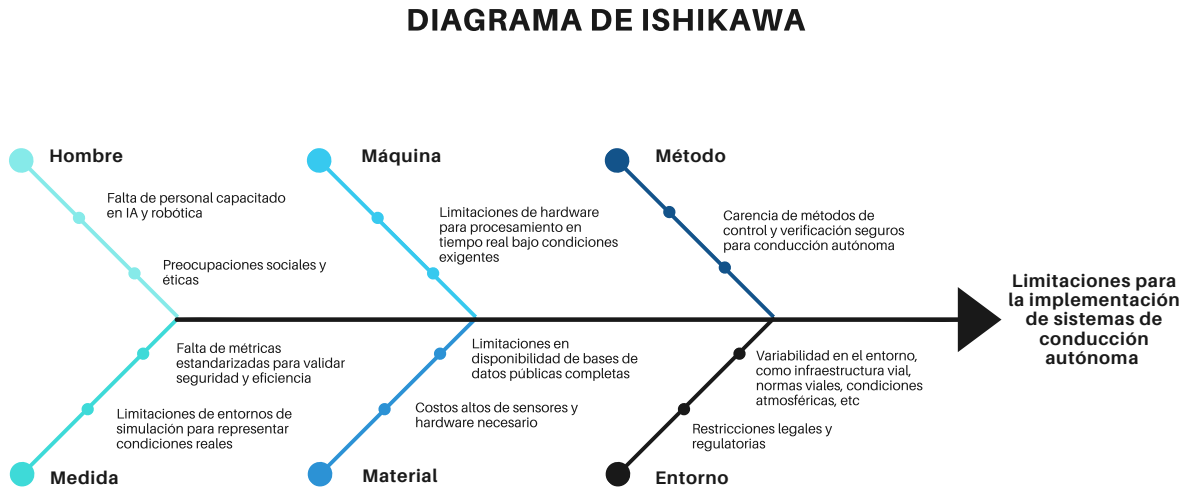


Figura 1.1: Diagrama de Ishikawa para el problema. Fuente: elaboración propia.

1.2.4 Síntesis del problema

Los sistemas de conducción tradicionales no logran adaptarse adecuadamente a entornos dinámicos, lo que motiva la búsqueda de estrategias de control más flexibles y seguras; sin embargo, validar estas soluciones en escenarios reales resulta costoso y complejo.

1.2.5 Alcance de la solución

El proyecto propone el desarrollo de un sistema de control para vehículos autónomos mediante técnicas de aprendizaje por refuerzo profundo, orientado a operar en entornos simulados. La solución contempla la construcción de un modelo vehicular, el diseño de un entorno tridimensional de prueba y la incorporación de restricciones de seguridad que permitan una toma de decisiones segura por parte del agente. Asimismo, se considera la implementación de un sistema de percepción que proporcione al controlador información relevante sobre el entorno. El alcance se limita a una validación en simulación, sin incluir pruebas en plataformas físicas ni escenarios con condiciones externas variables.

1.3 Esbozo de solución

La solución del proyecto sigue el flujo expuesto en el diagrama 1.2, como se puede apreciar se cuenta con un escenario 3D donde se modela con precisión los elementos físicos como vehículos, terreno y carreteras y los sensores del vehículo. A partir del escenario se implementa un sistema de percepción que interpreta la información proveniente de la simulación de los sensores y la transforma en entradas relevantes para el agente. Con esta información el agente decide la acción a ejecutar en función de dichas entradas. Finalmente, es necesario simular el comportamiento dinámico del vehículo, con el fin de transformar la acción tomada por el agente en un movimiento que represente de forma más realista la respuesta de un automóvil.

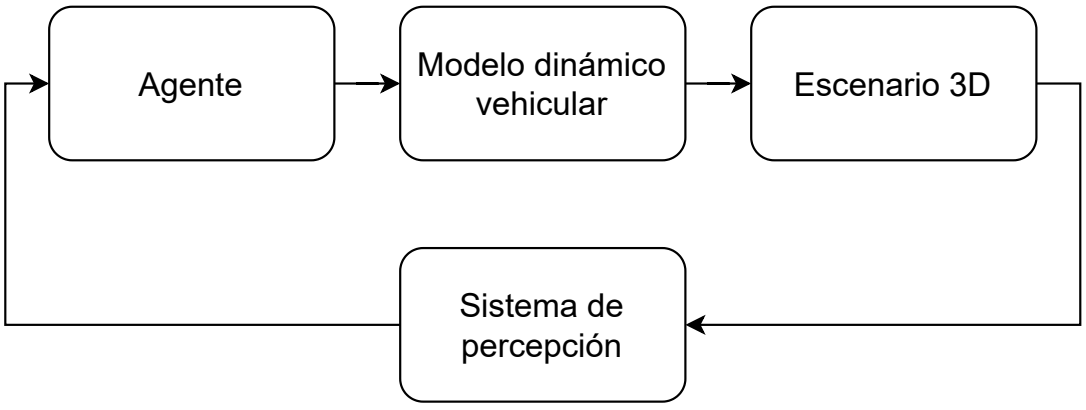


Figura 1.2: Diagrama de la solución implementada. Fuente: elaboración propia.

1.4 Objetivos y estructura del documento

1.4.1 Objetivo general

Desarrollar un controlador basado en aprendizaje por refuerzo que incorpore garantías de seguridad para optimizar la capacidad de los vehículos autónomos para tomar decisiones seguras y eficientes en entornos de conducción dinámicos.

1.4.2 Objetivos específicos

1. Realizar un estudio del estado del arte sobre los principales métodos de control en vehículos autónomos que incluya las principales causas de fallos.
2. Implementar un modelo de movimiento vehicular con el cual simular la medición de la posición, orientación y velocidad del vehículo según las acciones tomadas por el controlador.
3. Programar la etapa inicial del controlador de aprendizaje por refuerzo, con el fin de verificar la implementación de las garantías de seguridad solicitadas por el centro de investigación.
4. Desarrollar un sistema de percepción, basado en visión por computadora y/o fusión sensorial para obtener las mediciones necesarias para el funcionamiento del controlador.
5. Evaluar la capacidad del sistema de control, como un todo, para implementar las garantías de seguridad requeridas durante la conducción, mediante pruebas en un ambiente de simulación 3D.

El documento se estructura de la siguiente manera: el capítulo 2 presenta una investigación del estado del arte en sistemas de percepción y métodos de control autónomo basados en aprendizaje por refuerzo, así como de los fundamentos teóricos relacionados con el aprendizaje por refuerzo, la percepción vehicular y el modelo dinámico vehicular. En el capítulo 3, se describe la metodología adoptada para diseñar e implementar el sistema propuesto. La solución implementada se presenta en el capítulo 4. El capítulo 5 expone los resultados obtenidos en las simulaciones, mientras que el capítulo 6 resume las conclusiones más relevantes y sugiere posibles líneas de trabajo futuro.

Capítulo 2

Marco Teórico

2.1 Metodología utilizada para la investigación

Para el diseño de este sistema se sigue la metodología planteada por Ulrich y Eppinger [55]. La metodología constituye un elemento clave en la ejecución de todo proyecto, ya que ofrece una estructura organizada para enfrentar los retos y asegurar que las acciones estén alineadas con los objetivos definidos. Esta metodología se divide en las etapas mostradas en la figura 2.1. En esta investigación se comprende hasta la prueba de conceptos de producto.

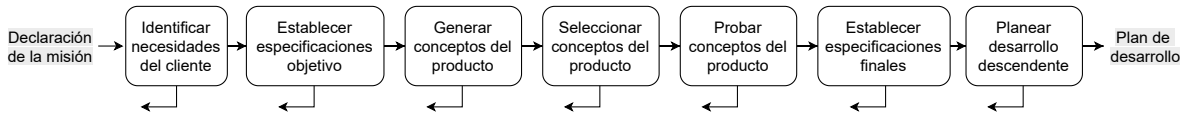


Figura 2.1: Etapas de la metodología de Ulrich Eppinger. Fuente: elaboración propia a partir de [55].

2.1.1 Identificar necesidades del cliente

En esta etapa se recopila información sobre las expectativas, deseos y requisitos del cliente o usuario final. Se utilizan entrevistas y observaciones para entender profundamente las necesidades del cliente. Estas necesidades se organizan y se priorizan, formando una base clara para el desarrollo del producto [55].

2.1.2 Establecer especificaciones objetivo

Basándose en las necesidades identificadas, se establecen especificaciones cuantitativas y cualitativas que el producto debe cumplir. Estas especificaciones son concretas y medibles, y definen los límites de desempeño, funcionalidad y calidad del producto. Las especificaciones sirven como criterios para evaluar el éxito del diseño en las etapas posteriores [55].

2.1.3 Generar conceptos de producto

Esta fase implica el desarrollo de múltiples ideas o conceptos que podrían satisfacer las especificaciones del producto. Se exploran soluciones a través de brainstorming y otros métodos de generación de ideas, y se formulan conceptos iniciales de diseño que aborden las necesidades y especificaciones [55].

2.1.4 Seleccionar concepto(s) de producto

Los conceptos generados se evalúan y comparan en función de su viabilidad técnica y ajuste con las especificaciones. Mediante herramientas como matrices de decisión, se selecciona el o los conceptos de producto que mejor cumplen con los objetivos establecidos y que muestran el mayor potencial de éxito [55].

2.1.5 Probar concepto(s) de producto

En esta etapa, los conceptos seleccionados se someten a pruebas iniciales para verificar su funcionamiento y desempeño. Las simulaciones ayudan a identificar posibles problemas o áreas de mejora antes de pasar a la fase de diseño detallado [55]. Las pruebas se muestran en el capítulo 5.

2.2 Estudio del estado del arte en metodologías relacionadas a la conducción autónoma

En esta sección se presentan los enfoques más relevantes para la detección de carriles en entornos urbanos, la detección de vehículos y los principales algoritmos de aprendizaje por refuerzo profundo. Partes esenciales para el desarrollo de un controlador basado en aprendizaje por refuerzo profundo. Para esto, se consideran principalmente artículos de revisión publicados en revistas indexadas, particularmente desde 2018, con el fin de sintetizar enfoques consolidados y tendencias emergentes.

Un componente esencial de la conducción autónoma corresponde a los sistemas avanzados de asistencia de conducción (ADAS, por sus siglas en ingles). Según [5], un modelo ADAS puede estar compuesto por: control de crucero adaptativo, frenado automático / maniobra evasiva, sistema de mantenimiento de carril, asistencia de punto ciego o sistema de advertencia de salida de carril y detección de carriles. Ya que el principal indicador en un carretera de la posición de auto en el carril son las líneas del carril, la detección de estas líneas cobra importante relevancia para los sistemas de conducción.

2.2.1 Métodos para la identificación del carril de conducción

Métodos tradicionales basados en modelado geométrico

Los métodos tradicionales de detección de carriles siguen una secuencia estructurada compuesta por cuatro etapas principales: preprocesamiento de la imagen, extracción de características, ajuste del modelo y seguimiento de líneas. Este enfoque ha sido ampliamente utilizado debido a su simplicidad y bajo requerimiento computacional en comparación con métodos más recientes [6].

El preprocesamiento de imagen busca eliminar ruido y realzar las características relevantes mediante técnicas como la transformación de perspectiva, umbralización, conversión a escala de grises y filtrado. Una estrategia común es la delimitación de una región de interés (ROI), usualmente en la parte inferior de la imagen, lo que permite reducir información redundante y enfocar el procesamiento en la zona donde normalmente se encuentran las marcas viales. Sin embargo, este enfoque puede verse afectado por sombras o la presencia de otros vehículos [7]. En la etapa de extracción de características se emplean propiedades visuales como color, bordes, geometría, y patrones locales, mediante técnicas como detección de bordes, operadores morfológicos, búsqueda de vecindades, clustering y ventanas deslizantes [6], [5].

Una vez extraídas las características, se ajustan modelos matemáticos a las posibles líneas de carril. Entre las técnicas utilizadas destacan el Line Segment Detector (LSD), ajustes polinomiales (lineales, cuadráticos, parabólicos o hiperbólicos), B-splines, y algoritmos como RANSAC, BLVS, CLAHE, o métodos de optimización como búsqueda por colonia de hormigas o Harmony Search. También se han aplicado métodos como la transformada de Hough, análisis de histograma, fusión de modelos y regresión lineal para mejorar la robustez del ajuste [5].

Para mantener la detección entre fotogramas, se utilizan técnicas de seguimiento como el filtro de Kalman, la categorización de carriles y el uso de modelos parabólicos. Estas herramientas permiten compensar fluctuaciones de iluminación u oclusiones parciales, ofreciendo una detección más estable en secuencias de video [7], [8].

Detección basada en inteligencia artificial

La inteligencia artificial (IA) hace referencia a la capacidad de los sistemas computacionales para imitar procesos cognitivos humanos, tales como el razonamiento, la percepción o la toma de decisiones. Entre sus aplicaciones destacan los sistemas expertos, el procesamiento de lenguaje natural, el reconocimiento de voz y la visión artificial [5].

En el contexto de la detección de carriles, los sistemas basados en IA emplean grandes volúmenes de datos etiquetados para aprender correlaciones y patrones relevantes, que luego son utilizados para realizar predicciones. Las dos ramas principales de esta aproximación son el aprendizaje automático (Machine Learning) y el aprendizaje profundo (Deep Learning). Ambas han sido aplicadas al problema de identificación de carriles, con un desempeño notablemente superior al de los métodos tradicionales, especialmente en entornos complejos o no

estructurados. Sin embargo, el aprendizaje profundo ha adquirido mayor relevancia debido a su capacidad para realizar tareas de clasificación o segmentación directamente a partir de imágenes, con un rendimiento sobresaliente [5].

Recientemente, se han incorporado mecanismos de atención en arquitecturas de aprendizaje profundo para mejorar la detección de carriles bajo condiciones complejas. El modelo Lane-DeepLab, desarrollado por [9], introduce dos innovaciones: un módulo de atención incorporado en la estructura ASPP del encoder y una unidad SEB que fusiona información semántica de distintos niveles para obtener características más representativas. De manera similar, [7] propuso LDNet, una arquitectura encoder-decoder diseñada para sensores de visión dinámica, que integra un bloque ASPP y un decodificador guiado por atención. Esta configuración mejora la localización de características relevantes y elimina la necesidad de postprocesamiento, incluso en imágenes de resolución completa. Por otro lado, RS-Lane, presentado por [6], está basado en LaneNet y emplea ResNeSt como backbone, utilizando Split-Attention para mejorar la representación de carriles delgados y con anotaciones dispersas. Además, implementa Self-Attention Distillation (SAD), permitiendo que las capas inferiores aprendan de las superiores sin añadir complejidad computacional durante la inferencia. Estas estrategias han demostrado mejorar la robustez del sistema frente a condiciones adversas como iluminación variable, clima complejo o múltiples carriles.

2.2.2 Métodos para la detección de vehículos

Además de la detección de líneas, la detección de vehículos es de gran importancia para garantizar la seguridad del control autónomo.

Detección de vehículos mediante cámaras y métodos tradicionales

Los métodos tradicionales para la detección de vehículos se apoyan en características visuales inherentes a su apariencia. Estas técnicas suelen estructurarse en dos etapas principales: generación de hipótesis (HG) y verificación de hipótesis (HV). En la primera, el sistema identifica regiones de interés (ROI) en la imagen que probablemente contengan vehículos, mientras que en la segunda se verifica si dichas regiones efectivamente contienen un objetivo válido. Este enfoque escalonado permite reducir la carga computacional al evitar el análisis exhaustivo de toda la imagen [10].

Durante la etapa de generación de hipótesis, se pueden explotar diversas características visuales de los vehículos:

- **Color:** La distribución de color en los vehículos tiende a ser continua y concentrada, lo cual permite segmentarlos del fondo mediante distintos canales de color y umbrales apropiados. Sin embargo, esta técnica es sensible a variaciones de iluminación y reflejos especulares [11].
- **Simetría:** Muchos vehículos presentan una estructura simétrica, especialmente en

su parte trasera. Este rasgo puede aprovecharse para identificar regiones altamente simétricas dentro de las ROI, optimizando los cuadros delimitadores (*bounding boxes*) y confirmando la presencia de vehículos en la fase de verificación. No obstante, su implementación puede elevar los costos computacionales [12].

- **Bordes:** Elementos como siluetas, parachoques, ventanas traseras y placas muestran texturas lineales fuertes tanto en dirección vertical como horizontal. La extracción de estos bordes característicos permite delimitar con mayor precisión la posición del vehículo, aunque pueden confundirse con líneas del entorno, generando falsos positivos [13].

- **Textura:** Las superficies viales tienden a presentar texturas uniformes, mientras que la superficie de los vehículos muestra mayor variabilidad. Esta diferencia puede ser aprovechada para detectar automóviles, aunque la precisión del método basado únicamente en textura suele ser limitada [14].

- **Sombras:** En condiciones de alta iluminación, los vehículos proyectan sombras claras y estables, las cuales presentan valores de gris considerablemente menores al resto de la vía. Estas sombras pueden segmentarse y utilizarse como indicadores de la presencia de un vehículo. Sin embargo, este enfoque presenta una aplicabilidad restringida a ciertas condiciones ambientales [15].

- **Luces traseras:** Otra característica utilizada en condiciones nocturnas es la detección de luces traseras, las cuales presentan un color característico (generalmente rojo) que puede ser extraído mediante técnicas de procesamiento de imagen [16].

Estos métodos tradicionales, aunque menos exigentes computacionalmente, presentan limitaciones ante escenarios complejos, como variaciones climáticas, entornos poco estructurados o interferencias visuales en el fondo de la imagen.

Detección de vehículos mediante cámaras y aprendizaje automático

Con el avance de la computación, el aprendizaje automático (Machine Learning, ML) se ha consolidado como una herramienta fundamental para la detección de vehículos. Esta técnica, que emula el proceso de aprendizaje humano mediante el análisis de datos, transforma las imágenes de entrada utilizando descriptores manuales para luego aplicar modelos de clasificación entrenados sobre espacios de baja dimensionalidad. El proceso general consta de dos etapas principales: primero, se realiza una extracción de la región de interés (ROI) mediante técnicas tradicionales; segundo, se extraen características locales que son utilizadas por el clasificador para diferenciar entre vehículos y no vehículos [10].

Entre los descriptores más utilizados destaca el histograma de gradientes orientados (HOG), originalmente exitoso en la detección de peatones, y posteriormente adaptado a la detección de vehículos mediante variantes como el HOG piramidal, el HOG simétrico o su integración con modelos deformables (DPM). También se han empleado vectores Haar-like, SURF, PCA,

SIFT, LBP y filtros de Gabor, cada uno con distintas fortalezas en cuanto a robustez frente a cambios de pose, textura o iluminación. En cuanto a los clasificadores, los más representativos incluyen Support Vector Machines (SVM), AdaBoost, Naive Bayes, K-nearest neighbors (KNN) y árboles de decisión (DT), siendo SVM y AdaBoost los más comunes debido a su precisión y buena capacidad de generalización. Estudios recientes han demostrado que la combinación de múltiples descriptores y clasificadores (estrategia conocida como ensemble learning) puede mejorar la precisión global del sistema [10].

Detección de vehículos mediante cámaras y aprendizaje profundo

Los métodos de detección de vehículos basados en aprendizaje profundo surgieron como alternativa a los enfoques de aprendizaje automático tradicional, los cuales dependen de descriptores manuales y clasificadores diseñados a mano. Con el auge de las redes neuronales convolucionales (CNN), ha sido posible aprender representaciones jerárquicas directamente a partir de los datos de imagen, mejorando significativamente el rendimiento en tareas como la clasificación, segmentación y detección de objetos [10]. En este contexto, la detección de vehículos se aborda principalmente desde tres paradigmas: modelos basados en detección de objetos, en segmentación, y modelos de extremo a extremo.

Entre los enfoques basados en detección con anclas (anchor-based), destacan Faster R-CNN con VGG-16 como red base, Mask R-CNN y Cascade R-CNN, con desempeños progresivamente mejores en tareas de detección. Por otro lado, los modelos de una sola etapa como YOLOv4, YOLOv5 y el más reciente YOLOv7 han mostrado mejoras sustanciales tanto en precisión como en eficiencia respecto a las arquitecturas anteriores. En cuanto a los métodos sin anclas (anchor-free), modelos como CenterNet, FCOS y YOLOX han demostrado una alta capacidad para localizar objetos de forma precisa sin requerir regiones propuestas. Finalmente, los modelos de detección directa de extremo a extremo como DETR, Deformable DETR y RT-DETR representan una nueva generación de arquitecturas que integran mecanismos de atención, logrando resultados superiores sin necesidad de etapas intermedias de propuesta o refinamiento [10].

Detección de vehículos mediante radar de ondas milimétricas

El radar de ondas milimétricas funciona mediante la emisión y recepción de señales en frecuencias del espectro milimétrico, utilizando principios como el tiempo de vuelo y el efecto Doppler para estimar parámetros del entorno. Estos incluyen la distancia, velocidad, dirección, tamaño y trayectoria de los objetos. En comparación con sensores ópticos como cámaras o LiDAR, los radares de ondas milimétricas presentan una mayor robustez ante condiciones climáticas adversas, como lluvia, niebla o nieve, lo cual los convierte en una herramienta confiable para la percepción en vehículos autónomos [10]. Según el tipo de la salida de la señal, los métodos se clasifican en a nivel de blanco (target-level) o a nivel de imagen (image-level).

Radar a nivel de blanco

Según [10], el radar a nivel de blanco está orientado al blanco detectado y puede transformar directamente las señales de eco recibidas en información sobre el objetivo, como la distancia, velocidad y ángulo de los vehículos detectados. Los resultados de detección del radar pueden clasificarse en tres categorías: blancos en movimiento, blancos estacionarios y blancos falsos. Los blancos falsos son inválidos debido a interferencias o ruido de fondo. El radar, por sí mismo, no tiene la capacidad de discriminar entre los tipos de blancos detectados. Por tanto, es necesario eliminar en la medida de lo posible la interferencia de los blancos estacionarios y falsos en la detección de vehículos.

Según investigaciones, las señales de blancos falsos solo persisten durante un corto período tras ser detectadas, y pueden eliminarse mediante el filtro de Kalman [17], el enfoque de hipótesis múltiples [18] y el método adaptativo iterativo [19]. Además, algunos autores han observado que la sección transversal radar (RCS) y la relación señal-ruido (SNR) de los vehículos estacionarios son mucho menores que las de los vehículos en movimiento [20]. La RCS se refiere al grado en que un objeto puede ser detectado por radar, mientras que la SNR es la relación entre la potencia de la señal deseada y la del ruido. Según las características del estado de movimiento, se puede establecer un umbral específico para la RCS y la SNR, lo cual permite separar los vehículos en movimiento de los objetos estáticos. El reconocimiento y la clasificación pueden lograrse mediante clasificadores basados en aprendizaje automático, como la máquina de vectores de soporte (SVM) y la red de creencias profundas (DBN) [10].

Radar a nivel de imagen

El radar a nivel de imagen ha emergido como una solución clave en percepción para vehículos autónomos, al integrar capacidades de detección de movimiento con generación de imágenes de alta resolución a partir de señales radar. A diferencia del radar a nivel de blanco, este enfoque permite representar información espacial detallada, útil incluso en condiciones adversas de visibilidad. Entre los principales formatos generados se encuentran los mapas de proyección, los mapas rango–Doppler–azimut, las nubes de puntos y los mapas SAR, cada uno con diferentes niveles de complejidad y precisión espacial [10].

Detección de vehículos mediante LiDAR.

LiDAR (Light Detection and Ranging) es una tecnología óptica que estima la distancia midiendo el tiempo transcurrido entre un pulso láser emitido y su reflexión. A partir de este principio, proporciona información geométrica precisa del entorno, como tamaño y coordenadas 3D de los objetos. El resultado es una nube de puntos tridimensional que representa con alta fidelidad el entorno vial. En comparación con cámaras y radares de onda milimétrica, LiDAR ofrece mayor precisión espacial y es insensible a cambios en la iluminación, lo que lo hace especialmente útil para la detección de vehículos en condiciones complejas [21].

Los métodos de detección basados en LiDAR pueden dividirse en dos grandes categorías: tradicionales y basados en aprendizaje profundo.

Métodos tradicionales

Los enfoques tradicionales se fundamentan en ingeniería de características y procesamiento

geométrico de datos. El flujo típico para detectar vehículos en una nube de puntos de un solo cuadro incluye preprocesamiento, reducción de datos, segmentación del suelo, y agrupamiento para la extracción de características. Debido a la dispersión inherente de los datos LiDAR, es común convertir la nube 3D a representaciones 2D o 2.5D utilizando métodos como imágenes de rango [22], mapas de ocupación [23] o técnicas basadas en grafos [24].

Para reducir el ruido, se eliminan puntos no relevantes, como aquellos que corresponden al pavimento, usando umbrales de características específicas [25],[26]. Sin embargo, estos métodos tienen dificultades en escenarios irregulares, como baches o pendientes. Para afrontarlo, se han propuesto algoritmos de ajuste como campos aleatorios de Markov (MRF) [27], RANSAC [28] y regresión por procesos gaussianos (GPR) [29], que segmentan el suelo en planos suaves. La agrupación posterior se realiza con algoritmos como DBSCAN o K-means, y los clústeres resultantes son clasificados mediante técnicas de aprendizaje automático para identificar vehículos.

Métodos basados en aprendizaje profundo

Con el avance de la visión por computador, los métodos basados en redes neuronales han demostrado ventajas claras en precisión y velocidad. A diferencia del enfoque tradicional, estos algoritmos aprenden representaciones jerárquicas directamente de los datos mediante arquitecturas end-to-end. Se clasifican en tres tipos según su entrada: point-based, que operan directamente sobre las coordenadas 3D; projection-based, que transforman los datos a imágenes; y voxel-based, que discretizan el espacio en volúmenes cúbicos [10].

Métodos basados en segmentación de nubes de puntos

En escenas viales, la segmentación semántica asigna una categoría a cada punto, como vehículo, peatón o vegetación. Los métodos tradicionales para segmentar nubes de puntos incluyen crecimiento de regiones [30], clustering [31] y ajuste de modelos [32], pero dependen de una costosa ingeniería de características y son sensibles a umbrales mal definidos. En cambio, las técnicas modernas de segmentación basadas en aprendizaje profundo permiten segmentaciones más precisas y eficientes. Al igual que en detección, se agrupan en point-based, projection-based y voxel-based, y constituyen hoy el enfoque más robusto para la detección de vehículos mediante LiDAR [10].

Entre los métodos point-based, destacan PointRCNN, Part-A2, y DCGNN, con AP superiores al 85% incluso en escenarios difíciles. En la categoría projection-based, PIXOR y Ri-Fusion muestran el mejor desempeño, con Ri-Fusion alcanzando 85.62% en condiciones fáciles y manteniendo solidez en las más complejas. Finalmente, en los métodos voxel-based, PV-RCNN, PDV y MA-MFFC sobresalen, superando el 90% de precisión en entornos fáciles y manteniendo métricas altas en escenarios moderados y difíciles, lo que los posiciona como referentes actuales en detección vehicular basada en LiDAR [10].

Para finalizar la sección de algoritmos de percepción, también es importante destacar la detección de vehículos y líneas mediante el uso de visión multisensor, en la tabla 2.1, obtenida de [10], se puede apreciar los diferentes pesos para criterios de detección según el tipo de

sensor utilizado. Ya que cada sensor tiene sus fuertes y sus bajos, al implementar una fusión sensorial es posible mejorar la calidad de las mediciones.

Tabla 2.1: Análisis comparativo de distintos sensores. Los valores del "1" al "5" indican el nivel desde extremadamente bajo hasta extremadamente alto. Fuente: obtenido de [10].

Criterio	**Cámara**	**Radar**	**LiDAR**
Representación de siluetas	5	2	3
Percepción del color	5	1	1
Medición de velocidad	2	5	2
Resolución angular	5	3	4
Resolución de alcance	2	4	5
Detección de objetos	5	2	4
Clasificación de objetos	5	1	3
Campo de visión	3	4	4
Adaptabilidad a condiciones climáticas complejas	2	5	2
Tamaño del sensor	2	2	4
Costo	3	1	5

2.2.3 Conducción autónoma basada aprendizaje por refuerzo profundo

Para efectos de conducción autónoma, destacan 5 metodologías utilizadas en ambientes de simulación: red Q profunda (DQN, deep Q network), actor-crítico (AC, actor-critic), gradiente de política determinista profundo (DDPG, deep deterministic policy gradient), optimización proximal de políticas (PPO, Proximal Policy Optimization) y actor-crítico suave (SAC, Soft Actor-Critic).

Para la resolución de desafíos relacionados con el seguimiento de trayectoria y el mantenimiento de carril, investigaciones recientes han logrado avances importantes utilizando diversas plataformas de simulación (como CARLA, Paramics y TORCS). Estos estudios aplican algoritmos de aprendizaje por refuerzo como DQN, DDPG, Actor-Critic y PPO, demostrando no solo la eficiencia y adaptabilidad de estas técnicas para controlar sistemas dinámicos complejos, sino también aportando nuevas soluciones para mejorar el rendimiento y la seguridad de los vehículos autónomos [4].

Carreteras con alta curvatura o situaciones de evasión de obstáculos, las variaciones en la trayectoria suponen grandes retos para mantener la precisión en el seguimiento. La propuesta de [33] mejora el algoritmo DDPG al integrar una red dual de críticos y un mecanismo de repetición de experiencias priorizadas, lo que incrementa su estabilidad y precisión en caminos con curvas cerradas y condiciones complejas. Al simular entornos con escenarios difíciles, se logra prevenir fallos en el seguimiento y se alcanza una alta exactitud. Esta versión mejorada de DDPG no solo optimiza el seguimiento de carriles, sino que también mejora la capacidad del modelo para adaptarse a distintas condiciones de conducción.

En [34] se desarrolló un sistema práctico basado en un modelo actor-critic, enfocado en el seguimiento de trayectoria bajo restricciones de seguridad. Este sistema se implementó como controlador lateral en vehículos reales. Un aspecto clave de su enfoque fue el uso de un adaptador ligero para conectar el entorno simulado con el mundo real. Validado en hardware de simulación de conducción, el modelo demostró efectividad en condiciones reales, facilitando así la transición de la simulación a aplicaciones prácticas.

Por otro lado, el estudio de [35] demuestra que a medida que aumenta la proporción de vehículos autónomos (AV) en el tráfico, se mejora la eficiencia en la movilidad, el consumo energético y la política de entrenamiento de controladores. El tráfico se vuelve más fluido y con menos congestión, siendo el escenario de autonomía total el que ofrece mejores resultados: se incrementa la velocidad promedio y se reduce el tiempo de retraso en 1.38 y 2.55 veces, respectivamente, en comparación con el caso de solo conductores humanos. Utilizando un enfoque basado en aprendizaje profundo por refuerzo (Deep RL) con el algoritmo PPO y un conjunto optimizado de hiperparámetros, el experimento con vehículos autónomos líderes superó tanto al tráfico completamente humano como al liderado por humanos. Este marco de control mostró ser eficaz en intersecciones sin semáforos, y se propone como solución para mitigar las ondas de parada y arranque.

En [35] se explora cómo aplicar aprendizaje profundo por refuerzo (DRL) en conducción autónoma. Se confirma que los autoencoders variacionales aceleran el aprendizaje y que distintas representaciones del estado afectan el desempeño del agente. Se desarrolló un sistema DRL funcional en CARLA, usando imágenes segmentadas, puntos de referencia y el algoritmo PPO para aprender planificación y control. El trabajo demuestra que los métodos basados en políticas son más adecuados para problemas con espacios continuos, como la conducción autónoma.

El artículo en [36] presenta ModEL, un nuevo marco modular de extremo a extremo basado en aprendizaje por refuerzo para conducción autónoma. Al combinar una arquitectura modular con el algoritmo SAC (Soft Actor-Critic), se mejora el rendimiento de la percepción, se refuerza la capacidad de planificación del agente y se reduce la brecha entre simulación y mundo real. El sistema fue validado en un vehículo real y entrenado en menos de un día gracias a un entorno distribuido. Se demostró su capacidad de generalizar ante escenarios visuales y dinámicos difíciles, superando a métodos tradicionales de aprendizaje por imitación y enfoques end-to-end.

2.2.4 Causas de fallos en sistemas de conducción autónoma

Los fallos en sistemas de conducción autónoma en los últimos años han sido provocados tanto por limitaciones técnicas como por errores humanos. A continuación, se detallan las principales causas con ejemplos documentados.

Fallos técnicos

Los errores técnicos en vehículos autónomos incluyen problemas de percepción, predicción, planificación y ejecución. Entre los más destacados se encuentran:

- **Errores de percepción:** Los sistemas han fallado al clasificar correctamente objetos, como confundir bicicletas con otros vehículos o no detectar peatones, especialmente en condiciones de baja visibilidad o iluminación [37, 38].
- **Frenado fantasma (*phantom braking*):** Se han registrado eventos donde vehículos autónomos frenan bruscamente ante estímulos inexistentes, como sombras o señales mal interpretadas [39].
- **Fallas de planificación y control:** Algunos sistemas han mostrado comportamientos inseguros al realizar giros o al cambiar de carril, provocando accidentes menores o la necesidad de intervención humana [40, 41].
- **Problemas de software y calibración:** La falta de pruebas exhaustivas ha ocasionado fallos de software críticos. Por ejemplo, un robotaxi de Cruise arrastró a una peatona por más de 6 metros tras un fallo de detección y lógica de evasión [38].
- **Ciberseguridad:** Simulaciones muestran que un malware puede provocar frenados peligrosos en más del 50% de las veces, revelando vulnerabilidades importantes en los sistemas de control [42].
- **Limitaciones sensoriales:** Las cámaras y sensores tienen dificultades para operar en condiciones climáticas adversas, de noche o cuando los objetos no están bien contrastados respecto al fondo [37].

Errores humanos

A pesar de los avances en autonomía, la intervención humana sigue siendo necesaria, especialmente en niveles 2 y 3 de automatización. Las causas humanas más comunes incluyen:

- **Distracción del conductor:** Un caso documentado ocurrió cuando un conductor utilizaba su celular mientras el sistema Autopilot de Tesla no detectó un camión detenido, lo que resultó en una colisión fatal [43]. En otra ocasión, la conductora de seguridad de un vehículo Uber en Phoenix, estaba viendo un programa de televisión y no reaccionó a tiempo al peatón cruzando [37].
- **Confianza excesiva:** Muchos conductores asumen que el vehículo es completamente autónomo y no supervisan adecuadamente su comportamiento [44, 45].
- **Interacciones imprevistas con usuarios viales:** Las IA aún fallan al interpretar comportamientos no estructurados de peatones o ciclistas. Un ejemplo reciente es el choque entre un robotaxi Zoox y una bicicleta en San Francisco durante una maniobra de giro [46].

Estos incidentes demuestran que la seguridad en la conducción autónoma requiere tanto de algoritmos más robustos como de una gestión adecuada de la interacción humano-máquina.

2.3 Identificación de carril mediante métodos tradicionales en MATLAB

Para la detección de carriles se emplea principalmente cámaras con las cuales es posible cubrir todos los ángulos alrededor del vehículo.

2.3.1 Características de las cámaras

La resolución y el campo de visión corresponden a los parámetros de mayor importancia para la detección de carriles mediante cámaras.

Resolución: Nivel de detalle que puede verse en la imagen. Expresado en pixeles [47].

Campo de visión (FOV, Field Of View): Tamaño del área el cual puede ser visto a través de la imagen. Expresado mediante el ángulo de apertura de la imagen [47].

2.3.2 Métodos para el procesamiento de imágenes

Vista de pájaro: Uno de los métodos utilizados para detectar los trazos de las líneas de carril, así como sus dimensiones, es la transformación a vista de pájaro (BEV, Bird's Eye View) de las imágenes obtenidas por las cámaras. Mediante la vista de pájaro se obtiene una vista aérea de la posición del automóvil en los carriles de la carretera. De ser aplicada correctamente, es una forma muy precisa de medir la distancia hasta un punto en terrenos planos. Para realizar una transformación es necesario saber el campo de visión, la resolución y la posición de la cámara con respecto al suelo. Por otro lado, también es importante especificar las dimensiones en pixeles y metros de la imagen de salida [48]. Para obtenerlo se utilizan las funciones `birdsEyeView` y `transformImage`.

Filtrado: Para llevar a cabo el filtrado de una imagen puede utilizarse correlación o convolución, para esto se aplica un filtro o máscara sobre los píxeles de una imagen, donde el color del píxel central varía según el filtro utilizado y los píxeles cercanos. La función `imfilter` ayuda con esta función.

Suavizado Gaussiano: El suavizado mediante un filtro gaussiano bidimensional permite reducir el ruido de alta frecuencia preservando los bordes significativos. La función `imgaussfilt` aplica dicho filtro a la imagen de entrada y utiliza como parámetro la desviación estándar σ, la cual controla la intensidad del desenfoque. Este preprocesamiento ayuda a mejorar la detección de bordes en etapas posteriores.

Elemento estructurante: La función `strel` crea un objeto que define la forma y tamaño del vecindario que será utilizado en operaciones morfológicas como dilatación o erosión. Al

emplear una estructura lineal, se le puede dar dirección al efecto de la operación en una orientación específica (por ejemplo, horizontal para carriles).

Dilatación morfológica: La función `imdilate` permite realizar operaciones morfológicas que expanden las regiones blancas de una imagen binaria. Utiliza un elemento estructurante definido mediante `strel`, el cual puede tener diferentes formas, como líneas o discos. En este caso, se usa para unir bordes fragmentados o paralelos que forman parte de una misma estructura.

2.3.3 Métodos de visión por computadora

Detección de bordes: La función `edge` permite detectar los contornos de una imagen en escala de grises o binaria. Uno de los métodos más usados es el algoritmo de Canny, que combina suavizado, cálculo del gradiente y umbrales adaptativos para obtener bordes definidos y conectados. Esta operación produce una imagen lógica donde los píxeles blancos representan los bordes detectados.

Eliminación de componentes pequeños: Con la función `bwareaopen` es posible eliminar regiones conectadas con área menor a un umbral determinado, lo que resulta útil para descartar ruido o detecciones irrelevantes posteriores a la segmentación.

Componentes conectados: La función `bwconncomp` analiza una imagen binaria y agrupa los píxeles blancos que están conectados entre sí, devolviendo una estructura con información de conectividad, tamaño de imagen, y lista de índices para cada componente.

2.3.4 Representación de carriles

Como se mencionó en [5], los carriles pueden ser representados mediante ajustes polinómicos, para esto se aplica un ajuste parabólico a los puntos identificados mediante la función `polyfit`.

Para calcular la curvatura de un carril a partir de un polinomio de grado 2 $y_p(x)$ es posible utilizar la formula 2.1, evaluada en el punto de interés en la curva.

$$\rho = \frac{\ddot{y}_p(x)}{(1 + \dot{y}_p(x)^2)^{3/2}} \tag{2.1}$$

2.4 Detección de vehículos mediante métodos tradicionales en MATLAB

2.4.1 Características de los sensores LiDAR

Los sensores LiDAR se utilizan ampliamente en vehículos autónomos por su capacidad de generar representaciones tridimensionales precisas del entorno. Entre sus parámetros más relevantes destacan el rango de detección, el campo de visión y la precisión en las mediciones.

Rango de detección: Distancia máxima a la que el sensor puede detectar objetos con fiabilidad. Está influenciado por factores como la potencia del láser, la reflectividad de los objetos y las condiciones ambientales [49].

Campo de visión (FOV, Field Of View): Área angular que puede ser captada por el sensor. Puede alcanzar hasta 360° en el plano horizontal para sensores rotatorios, y entre 20° y 40° en el plano vertical dependiendo del número de canales del dispositivo [49].

Precisión y exactitud: La exactitud (accuracy) representa qué tan cerca está una medición del valor real, mientras que la precisión (precision) se refiere a la consistencia entre múltiples mediciones. Ambos aspectos son críticos para reconstruir con fiabilidad el entorno [49].

2.4.2 Métodos para el procesamiento de nubes de puntos

Para la detección de vehículos se utiliza en su mayoría un enfoque basado en ingeniería de características y procesamiento geométrico de datos, el cual aprovecha directamente la estructura espacial de la nube de puntos [5]. Estos métodos tradicionales no requieren entrenamiento previo y se fundamentan en operaciones como el filtrado por región de interés (ROI), la segmentación por altura, la agrupación por proximidad espacial y el análisis de dimensiones físicas del objeto. En el sistema implementado, este enfoque se refleja en el preprocesamiento de la nube mediante filtros espaciales y de validez, reducción por rejilla de vóxeles, y la agrupación de puntos usando el algoritmo DBSCAN para identificar objetos individuales. Cada clúster resultante se evalúa por su altura y posición relativa, descartando aquellos que no cumplen con criterios geométricos compatibles con vehículos.

Agrupamiento por densidad: El método *DBSCAN* (Density-Based Spatial Clustering of Applications with Noise) permite agrupar puntos en función de su densidad espacial, identificando regiones densamente pobladas separadas por áreas de baja densidad. Este algoritmo clasifica los puntos en tres categorías: puntos núcleo, puntos frontera y ruido, dependiendo del número de vecinos dentro de un radio especificado. En MATLAB, la función `dbscan` implementa este algoritmo y requiere definir el radio de vecindad y el número mínimo de puntos que constituyen una región densa. Los puntos que no pertenecen a ningún grupo se etiquetan como ruido [48].

2.4.3 Tiempo para colisión

El tiempo hasta la colisión se refiere al intervalo restante antes de que dos vehículos impacten entre sí, considerando sus velocidades actuales [50]. El cálculo de esta métrica es importante ya que permite reducir el número de observaciones necesarias, pasando de registrar posición y velocidad de todos los vehículos a reportar únicamente los parámetros críticos del objeto más relevante (ya que aquel con el TTC más bajo representa el peligro inminente), sin pérdida de información esencial para el control del sistema. Para este escenario se utiliza la aproximación descrita en 2.2.

$$TTC = \frac{d_r}{v_r} \tag{2.2}$$

Donde d_r y v_r son la distancia y velocidad relativa hasta el vehículo.

2.5 Filtro de Hampel

El filtro de Hampel es una técnica robusta para la detección y eliminación de valores atípicos (outliers) en señales temporales o series de datos [51]. A diferencia de métodos como el promedio móvil, que pueden verse fuertemente afectados por valores extremos, el filtro de Hampel utiliza estadística robusta para preservar la estructura real de la señal mientras atenúa las anomalías.

El funcionamiento se basa en analizar una ventana deslizante de tamaño fijo alrededor de cada punto de la señal. Si el valor central de la ventana se aleja más de un umbral respecto a la mediana local, se considera un atípico y se reemplaza por la mediana. Este método es especialmente útil cuando se trabaja con señales ruidosas. Un ejemplo de su implementación se muestra en la figura 2.2.

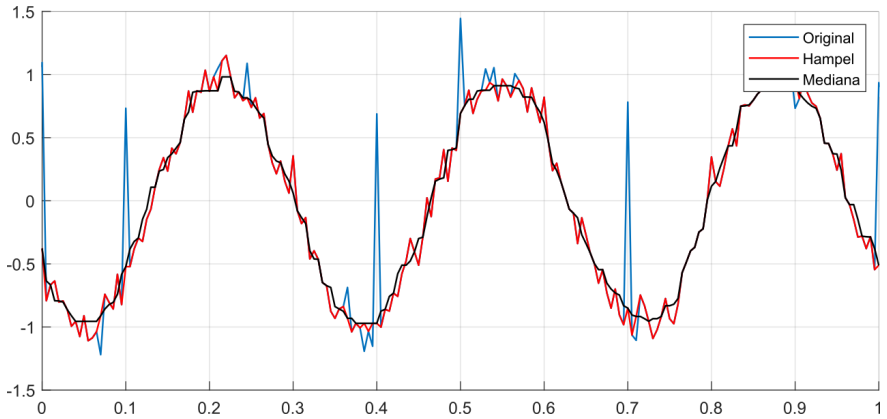


Figura 2.2: Ejemplo del funcionamiento de un filtrado por Hampel. Fuente: obtenido de [51].

2.6 Modelo dinámico vehicular

Para simular el comportamiento de un vehículo, se utiliza un modelo dinámico basado en el modelo cinemático de la bicicleta. Este modelo tiene como entradas la aceleración y el ángulo del volante de conducción. Como se expone en [52], al asumir que solo la rueda de adelante se puede maniobrar, se obtienen las escuaciones (2.3), (2.4), (2.6), (2.7).

$$\dot{X} = V\cos\left(\psi + \beta(\delta)\right) \tag{2.3}$$

$$\dot{Y} = V\sin\left(\psi + \beta(\delta)\right) \tag{2.4}$$

$$\dot{V} = \tau \tag{2.5}$$

$$\delta = \delta \tag{2.6}$$

$$\dot{\psi} = \frac{V}{l_r}\sin\left(\beta(\delta)\right) \tag{2.7}$$

Donde τ es el control de aceleración, $\beta(\delta)$ (2.8) es el ángulo de deslizamiento del centro de gravedad según el control del ángulo δ:

$$\beta(\delta) = \arctan\left(\tan(\delta)\cdot\frac{l_r}{l_f + l_r}\right), \tag{2.8}$$

y l_f y l_r la distancia del centro de gravedad al eje de las llantas. Las variables se pueden apreciar en el diagrama 2.3.

2.6.1 Modelo de bicicleta con entrada de fuerza en MATLAB

El modelo seleccionado en MATLAB tiene como entradas las fuerzas en las llantas en vez de la aceleración del vehículo, en vez de la aceleración, por lo que es necesario plantear una conversión entre la aceleración del vehículo y la fuerza que experimentan las llantas. Para esto, se plantea un tren de potencia y sistema de frenado trivial, según lo planteado en [53]. Las ecuaciones que describen el sistema se muestran en 2.9 y 2.10.

$$FxF = FxR = \frac{m\tau}{2}\mathcal{L}^{-1}\left(\frac{1}{\tau_1 * s + 1}\right) \tag{2.9}$$

$$FxF = FxR = \frac{m\tau}{2}\mathcal{L}^{-1}\left(\frac{tanh(0.8V)}{\tau_2 * s + 1}\right) \tag{2.10}$$

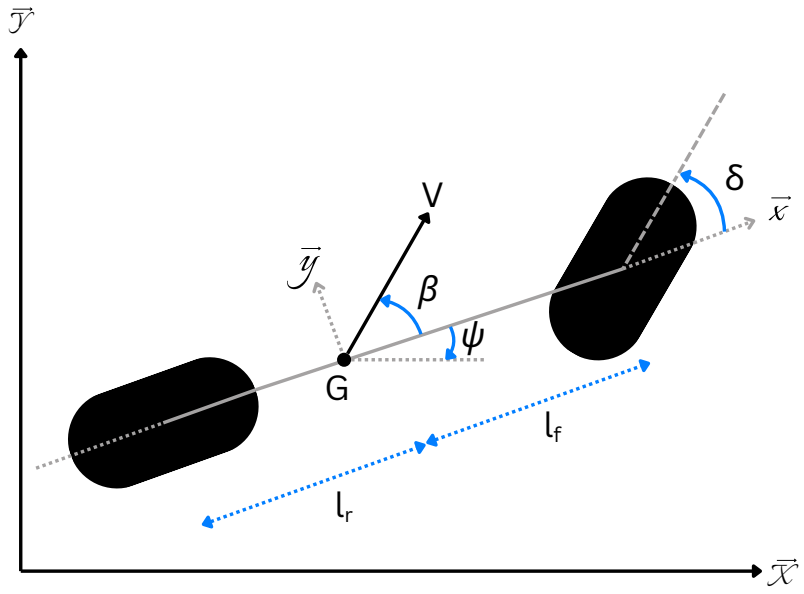


Figura 2.3: Diagrama del modelo cinemático de bicicleta. Fuente: elaboración propia basado en [52].

Donde m es la masa del vehículo, τ el control de aceleración del agente, s la variable en el dominio de Laplace, y τ_1 y τ_2 las constantes de aceleración y frenado.

2.6.2 Validación mediante Webots

Para validar el comportamiento del modelo dinámico del vehículo, se utilizó el simulador Webots, el cual proporciona un motor físico integrado que permite simular con precisión la interacción de cuerpos rígidos, fuerzas y fricción. A través del nodo Physics, Webots permite ajustar propiedades físicas como la masa, el centro de masa, la fricción, la inercia o la restitución, lo que permite aproximar de forma realista el comportamiento dinámico de un vehículo real en distintos escenarios [54].

Dado que no se cuenta con un vehículo físico para realizar pruebas reales, el uso de Webots representa una alternativa eficaz y segura. Al emular las condiciones físicas del entorno, es posible observar la respuesta del modelo ante acciones de control y verificar si el comportamiento simulado corresponde con las expectativas del modelo dinámico implementado en MATLAB/Simulink.

2.7 Optimización Proximal de Políticas (PPO) para conducción autónoma

2.7.1 Formulación matemática

Para llevar a cabo la conducción autónoma se selecciona el algoritmo PPO, el cual es un método de aprendizaje por refuerzo de gradiente de política y en línea (on-policy) para entornos con espacios de acción discretos o continuos. El actor guarda la experiencia (S, A, R, S'), luego estima directamente una política estocástica $\pi(A|S;\theta)$ y emplea un crítico de función de valor $V(S;\phi)$ para evaluar dicha política [48].

El agente busca minimizar la función de pérdida del crítico (2.11) y actor (2.12).

$$L_{\text{critic}}(\boldsymbol{\phi}) = \frac{1}{2M}\sum_{i=1}^{M}\left(G - V(S;\boldsymbol{\phi})\right)^2 \tag{2.11}$$

$$L_{\text{actor}}(\theta) = \frac{1}{M}\sum_{i=1}^{M}\left(-\min\left(r(\theta)\cdot\hat{D},\ c(\theta)\cdot\hat{D}\right) - w\mathcal{H}(\theta,S)\right) \tag{2.12}$$

donde el cociente de probabilidad $r(\theta)$ (2.13) y el factor de recorte $c(\theta)$ (2.14) se define como:

$$r(\theta) = \frac{\pi(A|S;\theta)}{\pi(A|S;\theta_{\text{old}})} \tag{2.13}$$

$$c(\theta) = \max\left(\min\left(r(\theta), 1+\epsilon\right), 1-\epsilon\right) \tag{2.14}$$

con los siguientes elementos:

- $\hat{D}, \hat{G}$: valor de la función de ventaja y el valor de retorno, respectivamente, basados en la recompensa obtenida R y el factor de descuento γ, especificado mediante la opción `DiscountFactor`.
- $\pi(A|S;\theta)$: probabilidad de tomar la acción A en el estado S con la política actual.
- $\pi(A|S;\theta_{\text{old}})$: probabilidad de la política anterior.
- ϵ: parámetro de recorte (clip), especificado mediante la opción `ClipFactor`.
- $\mathcal{H}(\theta, S)$: pérdida por entropía en el estado S.
- w: peso asociado a la entropía, especificado mediante la opción `EntropyLossWeight`.
- θ: parámetros del actor π.
- ϕ: parámetros del crítico V.
- M: Total de mini-batches.

Estado

El estado se representa mediante el vector descrito en (4.3).

$$S = [e_1, e_2, e_3, \rho, d_r, v_r], \tag{2.15}$$

donde e_1 es la desviación lateral respecto al centro del carril, e_2 es el error en el ángulo de orientación (guiñada) medido respecto a la dirección del carril, e_3 es el error en la velocidad longitudinal, ρ es la curvatura del carril, d_r y v_r la distancia y velocidad relativa, respectivamente, del vehículo más cercano a impacto.

Acción de control

Las acciones del agente están dadas por el vector descrito en (2.16).

$$A = [\delta, \tau], \tag{2.16}$$

donde δ representa el ángulo de giro y τ la aceleración.

Función de recompensa

El aprendizaje se puede guiar por una recompensa inmediata como en (2.17):

$$R = -w_1 e_1 - w_2 e_2 - ... - w_n e_n, \tag{2.17}$$

donde w_n denota el peso para cada error e_n. Cabe destacar que este es el ejemplo de una forma para la ecuación de recompensa. En el apartado de solución se entra en detalle en la ecuación utilizada específicamente para el proyecto.

Capítulo 3

Metodología de trabajo

En este capítulo se describe el enfoque metodológico utilizado para llevar a cabo el desarrollo del sistema de control de vehículos autónomos.

3.1 Identificar necesidades del cliente

3.1.1 Proceso de entrevista y necesidades interpretadas

Para obtener las necesidades se llevó a cabo varias reuniones en el periodo inicial del proyecto con el profesor del centro de investigación, en las cuales se aprovechó para determinar las necesidades para el proyecto mostradas en el cuadro 3.1.

Tabla 3.1: Necesidades del sistema desarrollado(SD).

Número	Necesidades		Imp.
1	El SD...	toma en cuenta garantías de seguridad	5
2		lee correctamente las variables del entorno	5
3		incorpora paradigmas de aprendizaje por refuerzo para tomar las decisiones de conducción	5
4		cuenta con un ambiente de simulación	5
5		modela correctamente la dinámica del vehículo	5

3.1.2 Análisis de la etapa

Se realizó una revisión exhaustiva de las necesidades identificadas para garantizar que todas las expectativas del cliente estuvieran cubiertas. Este enfoque permitió sentar bases sólidas para el desarrollo y asegurar la alineación con los objetivos del cliente.

3.2 Establecer especificaciones objetivo

3.2.1 Definir métricas y valores objetivo

Es importante definir métricas para garantizar que el sistema desarrollado se encuentra dentro de lo esperado por el cliente. Estas métricas están acompañadas de valores marginales e ideales, los cuales aseguran que el funcionamiento del sistema esté dentro de un rango aceptable. Las métricas y los valores objetivo se muestran en el cuadro 3.2.

Tabla 3.2: Métricas de evaluación para las necesidades del sistema.

# Nec.	# Mét.	Métrica	Unidades	Marginal	Ideal
1	1	Tasa de incidentes	i/MKm		<1.3
	2	Error absoluto en trayectoria	m	<0.75	
2	3	Error absoluto de la medición lateral	m	<0.2 (95%)	
	4	Error absoluto de la medición de guiñada	°	<0.51 (95%)	
	5	Error medio absoluto de traslación	m	<0.8 (95%)	
	6	Error medio absoluto de velocidad	m/s	<0.8 (95%)	
3	7	Incorpora paradigmas de aprendizajes por refuerzo	Binaria	Sí	
4	8	Variedad de escenarios de simulación	Lista (1)	1 a 2	1 a 5
5	9	Error absoluto de la velocidad longitudinal	m/s	<0.8	
	10	Error absoluto de la aceleración lateral	m/s^2	<0.8	

Lista (1):

1. Seguimiento de trayectoria
2. Incorporación de obstáculos
3. Límites de velocidad
4. Semáforos
5. Cambios de carril

Las métricas de evaluación del sistema fueron definidas con base en literatura científica relevante y entrevistas con el profesor del centro de investigación, y se organizaron según las necesidades funcionales identificadas.

Para la necesidad 1, se establecieron dos métricas. La primera es la tasa de incidentes, basada en el análisis estadístico de desempeño de vehículos autónomos en condiciones reales. En [56] se reporta que los sistemas actuales como los de Waymo pueden alcanzar tasas de incidentes menores a 1.3 por millón de kilómetros recorridos . En este proyecto, al

tratarse de una prueba de concepto, solo se considera el valor ideal como referencia. En caso de no alcanzarlo, se consultará con el profesor del centro de investigación para verificar si los resultados pueden considerarse satisfactorios. La segunda métrica, error absoluto en trayectoria, se basa en los criterios de precisión establecidos para evaluar el seguimiento de rutas por parte de vehículos automatizados [57].

La necesidad 2 considera la precisión de percepción y localización del sistema, agrupando cuatro métricas. El error absoluto de la medición lateral (métrica 3) y error absoluto de guiñada (métrica 4) también se fundamentan en la evaluación de precisión de localización en sistemas de conducción autónoma [58]. Las métricas 5 y 6, error medio absoluto de traslación y velocidad, se basan en los requisitos mínimos establecidos para localización confiable de vehículos autónomos, como lo indica [59].

Para la necesidad 3, se establece una única métrica binaria que evalúa si el sistema implementa paradigmas de aprendizaje por refuerzo (métrica 7). Este criterio fue definido a través de entrevistas con el profesor del centro de investigación, quien considera indispensable que el sistema incluya capacidades de toma de decisiones aprendidas mediante interacción con el entorno.

En la necesidad 4, la métrica 8 corresponde a la variedad de escenarios de simulación. Esta se evalúa en función de cuántos elementos de la Lista (1) se incluyen durante la validación. La selección de los escenarios mínimos requeridos queda a criterio del profesor del centro de investigación, quien indica cuántos de los cinco aspectos deben estar presentes para considerar que la simulación es suficiente.

Finalmente, para la necesidad 5, las métricas 9 y 10 cuantifican la precisión del modelo dinámico implementado. Se evalúa el error absoluto en la velocidad longitudinal y en la aceleración lateral, tomando como referencia la evaluación de modelos dinámicos analíticos utilizados para planificación de maniobras seguras, como se describe en el trabajo [60].

3.2.2 Matriz de correlación entre métricas y necesidades

Para representar visualmente la relación entre las métricas y las necesidades, se elaboró la matriz de necesidades, la cual se puede apreciar en el cuadro 3.3.

Tabla 3.3: Matriz necesidades del sistema.

Necesidad/Métrica	1	2	3	4	5	6	7	8	9	10
1	*	*								
2			*	*	*	*				
3							*			
4								*		
5									*	*

3.2.3 Análisis de la etapa

Las necesidades del cliente se tradujeron en especificaciones claras y medibles, como el error máximo en trayectoria y la tasa de incidentes. Antes de pasar a la siguiente etapa, se revisaron estas especificaciones para confirmar que fueran alcanzables y compatibles con las capacidades técnicas, asegurando así que el diseño cumpliría con los estándares de calidad esperados.

3.3 Generar conceptos de producto

3.3.1 Descomposición funcional del problema

Se cuenta con el sistema inicial mostrado en la figura 3.1.

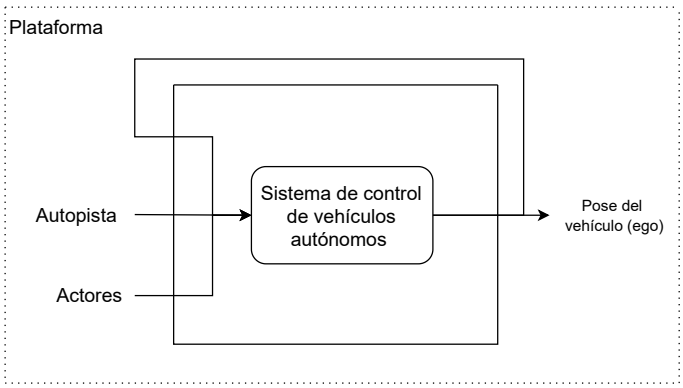


Figura 3.1: Diagrama del sistema inicial. Fuente: elaboración propia.

El cuál se puede descomponer en el diagrama de alto nivel mostrado en la figura 3.2.

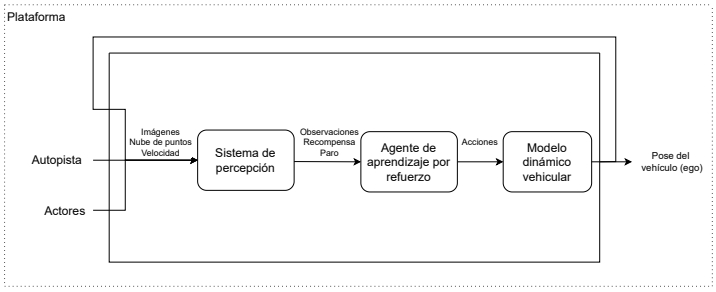


Figura 3.2: Diagrama de alto nivel del sistema. Fuente: elaboración propia.

Para analizar detalladamente el funcionamiento del sistema, se llevó a cabo una descomposición en sub-problemas, los cuales son más fáciles de resolver individualmente. Esta descomposición se muestra en la figura 3.3.

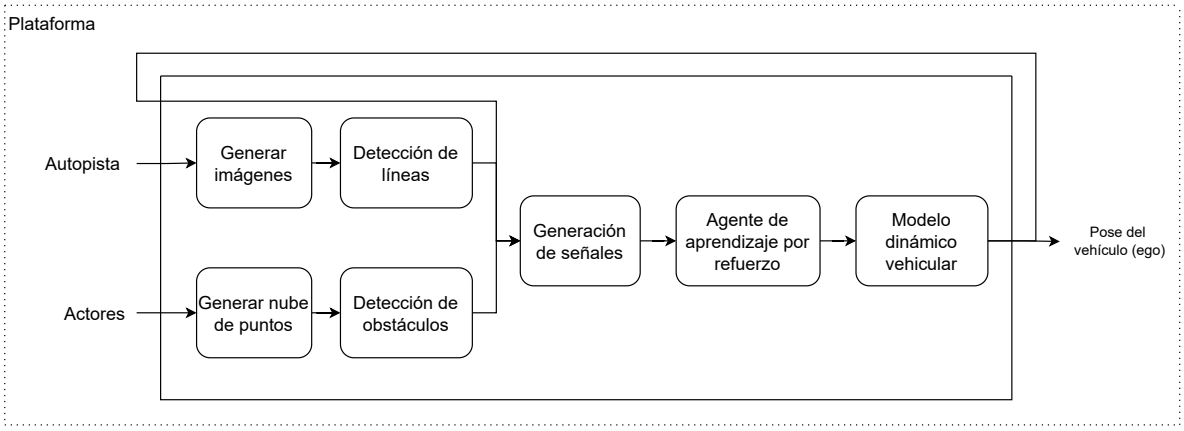


Figura 3.3: Descomposición funcional del sistema. Fuente: elaboración propia.

A continuación se explica brevemente cada sub-problema:
Plataforma: El entorno o lenguaje de programación que se usará para la simulación del sistema.

Generar imágenes: Dispositivos mediante los cuales se puede tomar fotos de la autopista.

Generar nube de puntos: Dispositivos que sirven para generar nubes de puntos con información sobre obstáculos.

Detección de líneas: Algoritmos para la generación de información correspondiente al carril de conducción.

Detección de obstáculos: Algoritmos para la detección de la distancia, velocidad, etc. de obstáculos en la autopista.

Generación de señales: Corresponde a la generación de las observaciones, la recompensa y la señal de paro. Corresponde a operaciones matemáticas sencillas.

Agente de aprendizaje por refuerzo: Metodología de aprendizaje por refuerzo profundo utilizada para tomar las acciones de conducción.

Modelo dinámico vehicular Modelo para la simulación del comportamiento de un vehículo para un comportamiento realista.

3.3.2 Proceso de búsqueda interna

Las siguientes opciones para los sub-problemas se determinaron mediante una lluvia de ideas.

- **Plataforma**

  1. Python.
  2. CARLA.
  3. MATLAB.

- **Generar imágenes**
    1. Cámara.
    2. Cámara infrarroja.
- **Generar nube de puntos**
    1. LiDAR.
- **Modelo dinámico vehicular**
    1. Modelo de bicicleta.

Es importante mencionar que los métodos utilizados en los procesos de filtrado y selección posteriores dependen en gran medida de la plataforma, por lo que esta se selecciona desde el inicio del proyecto según entrevistas con el profesor del centro de investigación. Se seleccionó MATLAB al ser considerada la forma más rápida de implementar todos los subsistemas que conforman el proyecto en un solo sistema, además de tenerse buen conocimiento de la plataforma.

3.3.3 Proceso de búsqueda externa

Por otro lado, también se llevó a cabo una búsqueda externa en aquellos sub-problemas en los cuales no se obtuvieron opciones luego la lluvia de ideas.

- **Generar nube de puntos**:
    1. Radar.
- **Detección de líneas**:
    1. Métodos tradicionales [5].
    2. LaneNet [61].
    3. Red Neuronal Convolucional Espacial [62].
- **Detección de obstáculos**:
    1. Métodos tradicionales [10].
    2. Voxel RCNN [63].
- **Agente de aprendizaje por refuerzo [4]**:
    1. PPO.
    2. SAC.
    3. DDPG.
- **Modelo dinámico vehicular**:
    1. Modelo dual [64].

3.3.4 Construcción de conceptos

En el cuadro 3.4 se muestra todas las posibles opciones para cada concepto. En este caso se elaboró un filtrado preliminar de las opciones que consideramos que no cumplían de buena forma los requisitos del proyecto.

Tabla 3.4: Opciones para cada sub-problema de la descomposición.

Generar imágenes	Generar nube de puntos	Detección de líneas
Cámara	LiDAR	Tradicional
Cámara infrarroja	RADAR	LaneNet
		SCNN

Detección de obstáculos	Agente	Modelo dinámico vehicular
Tradicional	DDPG	Modelo bicicleta
VoxelR RCNN	SAC	Modelo dual
	PPO	

Posteriormente se eligió 4 combinaciones (conceptos) para llevarlas al proceso de filtrado, estas se pueden apreciar en el cuadro 3.5.

Tabla 3.5: Combinaciones utilizadas para el filtrado de conceptos.

Combinación	Generar imágenes	Generar nube de puntos	Detección de líneas
A	Cámara	LiDAR	Tradicional
B	Cámara	LiDAR	Tradicional
C	Cámara	LiDAR	LaneNet
D	Cámara	LiDAR	LaneNet

Combinación	Detección de obstáculos	Agente	MDV
A	Tradicional	PPO	Modelo bicicleta
B	Tradicional	SAC	Modelo bicicleta
C	RCNN	PPO	Modelo bicicleta
D	RCNN	SAC	Modelo bicicleta

3.3.5 Análisis de la etapa

Se generaron varias ideas para cumplir con las especificaciones. Durante esta fase, se identificaron las opciones que mejor abordaban las necesidades clave del cliente, y se descartaron aquellas menos prácticas. Esto permitió un enfoque claro para la etapa de selección.

3.4 Seleccionar concepto(s) de producto

3.4.1 Filtrado de conceptos

Para determinar cuáles conceptos pasar a la selección es importante primero hacer un filtrado, tanto el filtrado como la selección deben estar planteadas con base en las necesidades del sistema. El proceso de filtrado se muestra en el cuadro 3.6. El filtrado y selección se hace con base en los siguientes criterios:

- **Tiempo de implementación:** Se refiere al tiempo estimado necesario para desarrollar e integrar la solución dentro del marco temporal del proyecto. Conceptos con tiempos excesivos pueden descartarse si ponen en riesgo la entrega.

- **Seguridad de operación:** Evalúa el nivel de riesgo que representa cada concepto para el vehículo y su entorno. Se priorizan aquellos enfoques que garanticen un funcionamiento seguro ante fallos o condiciones inciertas.

- **Requerimientos computacionales:** Considera la cantidad de recursos de cómputo necesarios (memoria, procesamiento, tiempo de ejecución). Los conceptos que exijan hardware especializado o cómputo intensivo pueden ser descartados si no son viables para su implementación en el entorno de simulación o en hardware embebido.

- **Precisión de control:** Mide la capacidad del sistema para seguir trayectorias deseadas y mantener el vehículo en una zona segura. Conceptos con mayor capacidad de respuesta y menor error en la ejecución del control son preferidos.

- **Precisión de medición:** Hace referencia a la exactitud con la que el sistema es capaz de interpretar su entorno (posición, velocidad, carriles, obstáculos). Conceptos que dependen de mediciones ruidosas o poco confiables pueden descartarse.

Tabla 3.6: Proceso de filtrado de conceptos.

Filtrado	**Conceptos**			
Criterio de selección	**A**	**B**	**C**	**D**
Tiempo de implementación	1	1	0	-1
Seguridad de operación	-1	-1	0	0
Requerimientos computacionales	1	1	0	0
Precisión de control	0	0	0	0
Precisión de medición	-1	-1	0	0
Total	0	0	0	-1
Lugar	1	1	1	2
¿Continuar?	Sí	Sí	Sí	No

3.4.2 Selección del candidato a prototipo

Una vez realizado el filtrado, los 3 mejores candidatos se llevan al proceso de selección. En este caso es importante agregarle un peso a cada criterio de selección, por lo que se calcula un peso basado en lo que mencionó el cliente durante las entrevistas. El proceso de selección se muestra en el cuadro 3.7.

Tabla 3.7: Evaluación ponderada de conceptos.

Evaluación / **Criterio de selección**	**Peso**	**Conceptos**					
		Calif A	**Pond A**	**Calif B**	**Pond B**	**Calif C**	**Pond C**
Tiempo de implementación	8	5	40	4	32	3	24
Seguridad de operación	10	5	50	5	50	5	50
Requerimientos computacionales	8	5	40	5	40	3	24
Precisión de control	10	5	50	5	50	5	50
Precisión de medición	10	4	40	4	40	5	50
Total			220		212		198
Lugar		Primero		Segundo		Tercero	
¿Continua en proceso?		Desarrollar		No		No	

Por lo que se tiene como concepto seleccionado la combinación A. Este concepto consta de los componentes del cuadro 3.8.

Tabla 3.8: Concepto ganador.

Combinación	**Generar imágenes**	**Generar nube de puntos**	**Detección de líneas**
A	Cámara	LiDAR	Tradicional

Combinación	**Detección de obstáculos**	**Agente**	**MDV**
A	Tradicional	PPO	Modelo bicicleta

3.4.3 Análisis de la etapa

Se evaluaron todos los conceptos generados para determinar cuál ofrecía el mejor balance entre sus características. Este proceso incluyó el uso de herramientas de comparación, como una matriz de decisión, que permitió evaluar cada alternativa en función de los criterios clave del proyecto. Tras la revisión, se seleccionó el concepto que mejor cumplía con las especificaciones objetivo, asegurando una alta probabilidad de éxito en las pruebas y fases posteriores. Algunos conceptos que estaban basados fuertemente en inteligencia artificial para las mediciones se descartan al requerir mucho tiempo de implementación. Aunque según lo investigado en el estado del arte estos métodos brindarían los mejores resultados, el tiempo prolongado de entrenamiento e implementación no los hace viables, al ser más importante la demostración del funcionamiento del agente de aprendizaje por refuerzo.

Capítulo 4

Diseño del sistema de control de vehículos autónomos mediante aprendizaje por refuerzo con garantías de seguridad

Como se ha mencionado anteriormente, primero se comienza con la implementación del modelo dinámico vehicular, luego se procede con el diseño del agente PPO y del sistema de percepción encargado de la detección de líneas y obstáculos. Finalmente se implementa el agente, el sistema de percepción, el modelo dinámico vehicular y el escenario 3D, en conjunto, con el fin de validar posteriormente el comportamiento del agente en condiciones más realistas.

4.1 Implementación del modelo dinámico vehicular

4.1.1 Implementación en MATLAB

Ya que las acciones del agente corresponden a la aceleración del auto y el ángulo de conducción, se debe transformar la aceleración en las fuerzas en cada llanta, para esto se implementa un tren de potencia y sistema de frenado trivial. Para el tren de potencia y el sistema de freno trivial se debe definir la masa y las constantes de aceleración y frenado, según lo planteado en [53]. La masa m se define en 1575, mientras que las constantes de aceleración τ_1 y frenado τ_2 en 0.5 y 0.07, respectivamente. La fuerza en cada llanta, según las ecuaciones 2.9 y 2.10, estaría dada según la ecuaciones 4.1 (aceleración) y 4.2 (frenado).

$$FxF = FxR = 787.5 \cdot a \cdot \mathcal{L}^{-1}(\frac{1}{0.5 \cdot s + 1}) \tag{4.1}$$

$$FxF = FxR = 787.5 \cdot a \cdot \mathcal{L}^{-1}(\frac{tanh(0.8 \cdot \dot{X})}{0.07 \cdot s + 1}) \tag{4.2}$$

Una vez se ingresa la fuerza en cada llanta y el ángulo de conducción al modelo dinámico en MATLAB, es posible obtener la velocidad longitudinal del automóvil, a partir del cual se calcula el error en la velocidad longitudinal (e_3) como la diferencia entre la velocidad de referencia y la velocidad longitudinal leída, para dar más realismo a las lecturas se les agrega ruido para simular el comportamiento de un sensor real. Además de esto, se registran los datos de la aceleración lateral para verificar el funcionamiento correcto del automóvil al momento de moverse lateralmente.

En las figuras A.1,A.2 se especifican en su totalidad los parámetros utilizados.

4.2 Diseño del agente de aprendizaje por refuerzo

4.2.1 Observaciones

Basado en [65], [66], [35], y como se mencionó anteriormente, el espacio de observaciones se define con el vector en 4.3.

$$S = [e_1, e_2, e_3, \rho, d_r, v_r], \tag{4.3}$$

La descripción de las variables se especifica a continuación:

Desviación lateral: la desviación lateral e_1 se define como la distancia más pequeña desde el centro del vehículo (punto A) hasta el centro del carril (punto B). Según [58], su valor máximo debe ser de 0.75 m durante la conducción. Para las observaciones es ideal que el valor se encuentre entre -1 y 1, si bien el valor ya se encuentra en este rango, igualmente se normaliza al dividir las lecturas entre 0.75 m, como en 4.4.

$$e_{1(Obs)} = \frac{e_1}{0.75m} \tag{4.4}$$

Error en el ángulo de guiñada: el error en el ángulo de guiñada e_2 se define como la diferencia entre el ángulo descrito por la tangente al centro del carril en el punto B y el eje longitudinal del automóvil. Se permite un error máximo de 20° o 0.35 rad, un error más alto que este y el auto invadirá otros carriles. Por lo tanto, se normaliza la lectura del ángulo según la ecuación 4.5.

$$e_{2(Obs)} = \frac{e_2}{0.35rad} \tag{4.5}$$

Error de velocidad: el error en la velocidad e_3 es simplemente la diferencia entre la velocidad longitudinal medida y la velocidad de referencia. Se define la velocidad mínima

como 0 m/s (no se consideran velocidades longitudinales negativas para el proyecto), la máxima como 15 m/s y la velocidad objetivo como 5 m/s. Con esto es posible definir los errores máximos y mínimos según las ecuaciones 4.6 y 4.7.

$$e_{3max1} = abs(5 - 0) = 5m/s \tag{4.6}$$

$$e_{3max2} = abs(5 - 15) = 10m/s \tag{4.7}$$

Con los valores máximos definidos, se puede normalizar el error de la velocidad según las ecuación en 4.8.

$$e_{3(Obs)} = \begin{cases} \frac{e3}{5m/s}, e3 < 0 \\ \frac{e3}{10m/s}, e3 \geq 0 \end{cases} \tag{4.8}$$

Curvatura de la autopista: la medida de la curvatura ρ también se agrega como una observación para que el agente tenga una referencia para la curvatura de la autopista. Ya que se encuentra en el rango de -1 y 1, no se normaliza.

Distancia y velocidad relativa: con d_r y v_r se brinda la información más importante del vehículo con el tiempo para colisión mas bajo. El rango de medición máximo para la distancia relativa se limita a 30 m, por lo que para normalizar la medición se divide d_r según la ecuación 4.9.

Con esto es posible normalizar la distancia relativa hasta el vehículo según la ecuación 4.9.

$$d_{r(Obs)} = \frac{d_r}{30m} \tag{4.9}$$

Para normalizar la lectura de la velocidad relativa del vehículo se utiliza la velocidad máxima definida anteriormente de 15 m/s, según la ecuación 4.10.

$$v_{r(Obs)} = \frac{v_r}{15m/s} \tag{4.10}$$

4.2.2 Acciones

Como se mencionó en la sección 2, el espacio de acciones está formado por el ángulo de conducción y la aceleración del automóvil. Estas se definen en un rango de -1 rad a 1 rad para el ángulo de conducción y -1 a 1 m/s^2 para la aceleración. Debido a que la aceleración es muy baja, se aplica una ganancia estática de 3 a la salida del control.

4.2.3 Recompensa

Para diseñar la ecuación de recompensa durante el entrenamiento se tomó como base el paper [36]. Las ecuaciones de recompensa buscan premiar errores bajos en los errores medidos, se muestran en las ecuaciones 4.11, 4.12 y 4.13.

$$R_1 = clip(1 - abs(e_{1(Obs)}), 0, 1) \tag{4.11}$$

$$R_2 = clip(1 - abs(e_{2(Obs)}), 0, 1) \tag{4.12}$$

$$R_3 = clip(1 - abs(e_{3(Obs)}), 0, 1) \tag{4.13}$$

Se implementa una compuerta "AND" con las 3 recompensas calculadas para que el agente aprenda a mejorar en todos los aspectos.

$$R_{AND} = R_1 \cdot R_2 \cdot R_3 \tag{4.14}$$

Los comportamientos que se castigan son aquellos que atenten contra la seguridad de la conducción, en este caso se castiga con 20 invadir otro carril y con 30 el impacto con otro auto u obstáculo. Con esto, se define finalmente la recompensa según la ecuación 4.15.

$$R = R_1 \cdot R_2 \cdot R_3 - 20(inv.) - 30(imp.) \tag{4.15}$$

4.2.4 Señales de paro

La simulación se detiene en el momento que se detecte un impacto, una invasión o un estancamiento en la conducción. Se define como impacto cuando la distancia relativa hasta otro vehículo d_r sea inferior 4 m, la invasión se da cuando la desviación lateral e_1 sea superior a 0.75 m, o cuando el error en el ángulo e_2 sea mayor a 0.35 rad. Para detectar estancamiento en la simulación se compara la espacio de observaciones pasado con el actual, cuando se repita el mismo espacio por 10 veces consecutivas, se levanta la señal.

4.2.5 Hiperparámetros de entrenamiento

En el cuadro 4.1 se muestra los hiperparámetros utilizados para el entrenamiento del agente. Estos se obtuvieron con base en las investigaciones de [65], [66] y según pruebas realizadas durante el proyecto.

Hiperparámetro	Valor
Factor de descuento γ	0.99
Factor de recorte ϵ	0.2
Tasa de aprendizaje del actor	3e-4
Tasa de aprendizaje del crítico	3e-4
Umbral de gradiente del actor	1
Umbral de gradiente del crítico	1
Peso de pérdida de entropía β	0.01
Horizonte de experiencia T	2048
Tamaño del mini-lote M	64
Número de épocas K	7
Máx. mini-lotes por época	100
Frecuencia de aprendizaje	2048
Unidades ocultas	512

Tabla 4.1: Hiperparámetros del agente PPO configurado en MATLAB.

4.3 Diseño del escenario de simulación

4.3.1 Escenario para la validación inicial del agente

Autopista

Para probar inicialmente el comportamiento correcto del agente se diseño una autopista de 4 carriles básica en forma circular, según se aprecia en la figura 4.1. Para esto se utilizó el app de MATLAB "Driving Scenario Desginer".

Actores

Además del automóvil controlado por aprendizaje por refuerzo ("ego"), se añaden 2 automóviles más en el mismo carril en que se encuentra el ego, uno al frente y otro detrás, a una determinada velocidad de conducción en la misma dirección que el ego.

Sensores

Para la comprobación inicial se utilizaron dos cámaras al frente y detrás del auto, la detección de líneas y obstáculos se hace mediante las cámaras. Para la detección del carril y la detección de obstáculos o vehículos se utilizan las mediciones brindadas por el Driving Scenario Designer. Más específicamente, las observaciones e_1, e_2, ρ, d_r y v_r fueron obtenidas mediante el Driving Scenario Designer, el cual brinda lecturas con muy poco error basadas en el escenario diseñado.

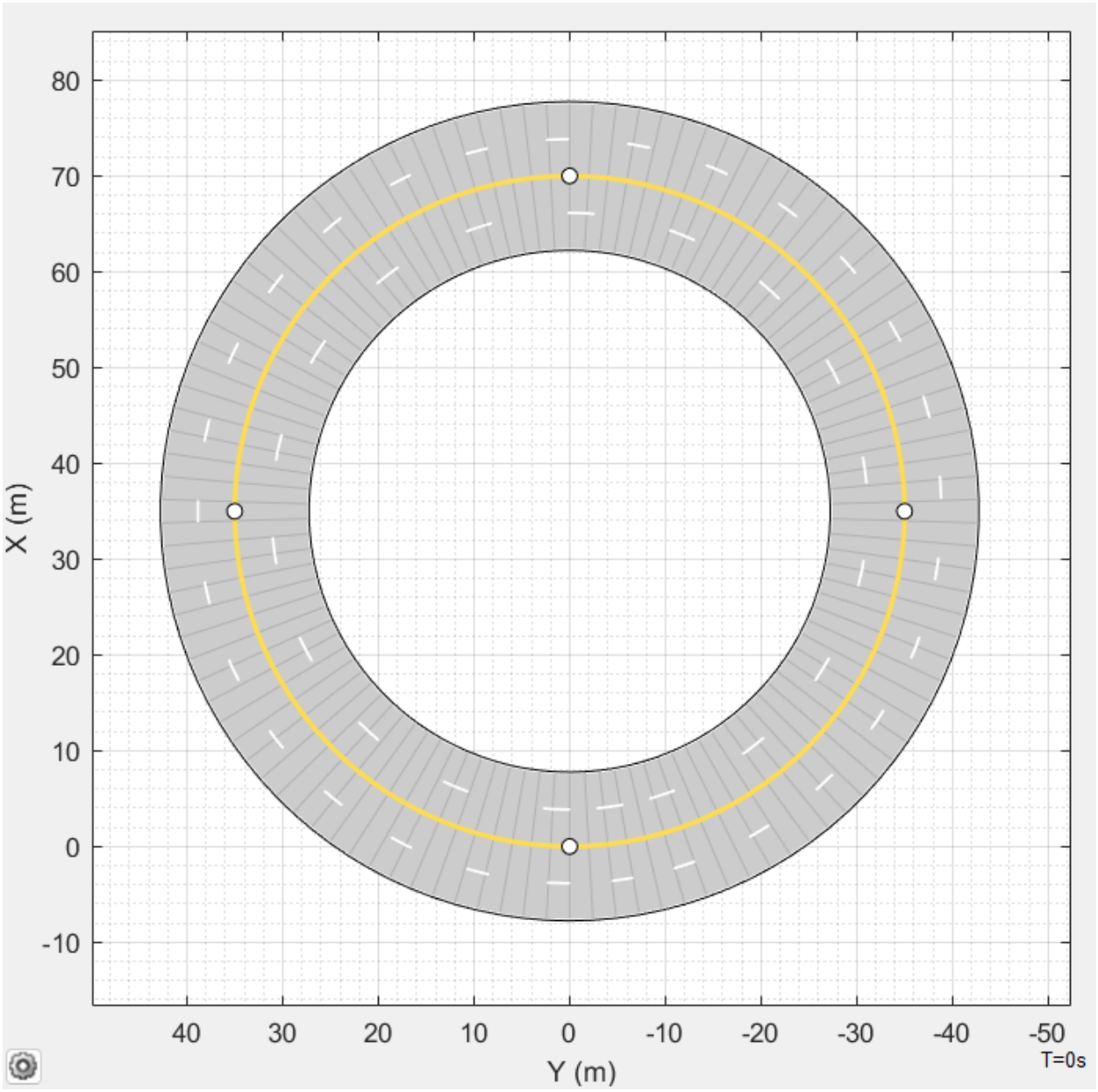


Figura 4.1: Autopista seleccionada para la validación inicial del agente. Fuente: elaboración propia.

4.3.2 Escenario para la validación final del agente

Autopista

Para validar el comportamiento final del agente se seleccionó la escena Carretera Curva (Curved Road), diseñada por MATLAB usando Unreal Engine, la cual corresponde a una autopista de 4 carriles con curvatura cambiante, esta se puede ver en la figura 4.2.

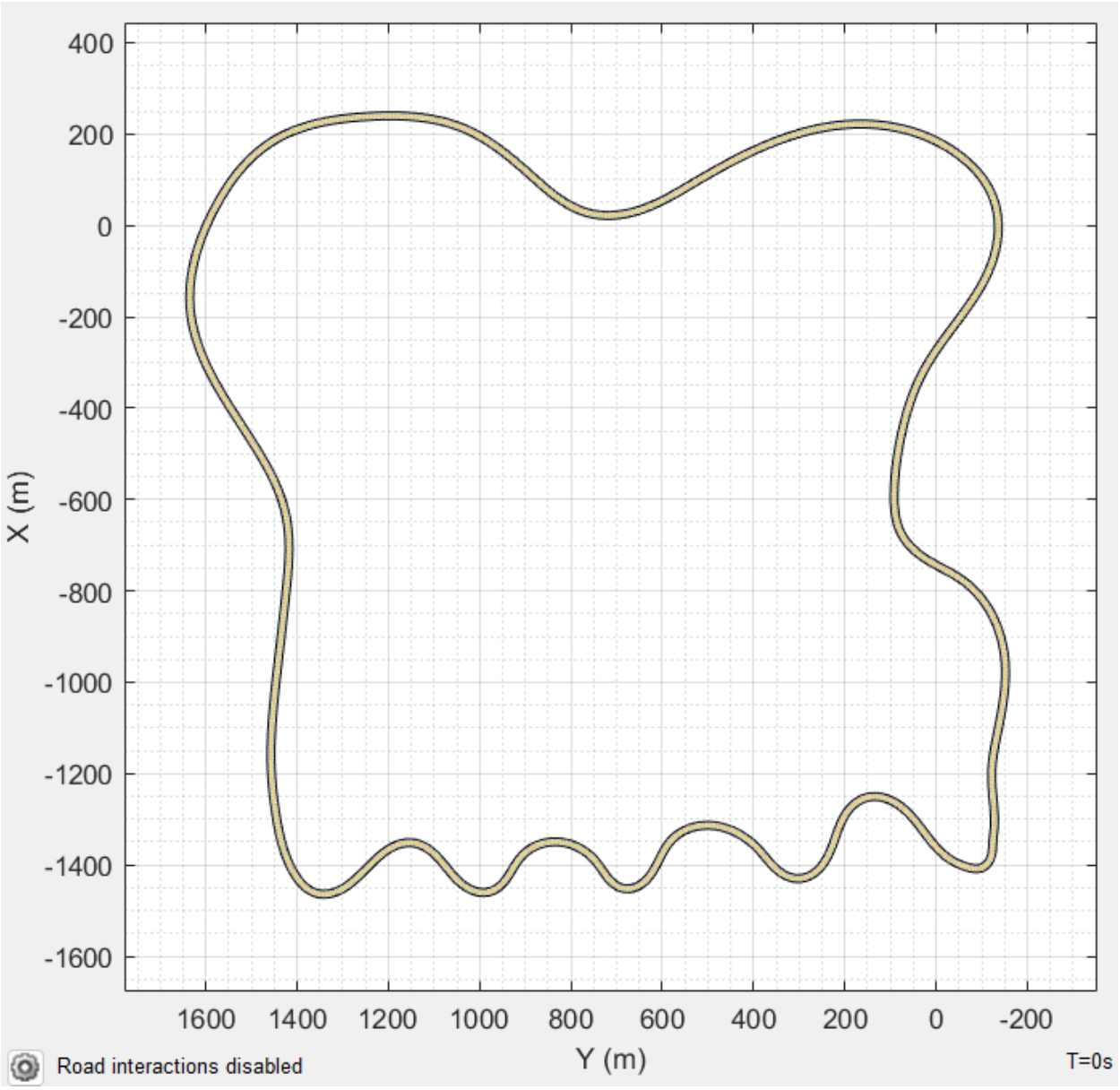


Figura 4.2: Autopista seleccionada para la validación final del agente. Fuente: obtenido de [48].

Actores

En este escenario también se añaden 2 automóviles en el mismo carril en que se encuentra el ego a una determinada velocidad de conducción en la misma dirección.

Sensores

Para la validación final mediante Unreal Engine, se selecciona 4 cámaras para cubrir los 360° alrededor del ego. Estas cámaras tienen una resolución de 1080x1920 pixeles y un campo de visión de 74°. Además de las cámaras, se incluye un LiDAR, sus características más importantes se incluyen en la tabla 4.2. Para esta validación se utiliza el sistema de percepción diseñado en las siguientes secciones.

Parámetro	Valor
Rango de detección (m)	45
Resolución de rango (m)	0.002
Campo de visión vertical (°)	40
Resolución vertical (°)	1.25
Campo de visión horizontal (°)	360
Resolución horizontal (°)	0.16

Tabla 4.2: Parámetros del sensor LiDAR utilizado en la simulación.

4.4 Diseño del sistema de percepción

4.4.1 Detección de carril

En la figura 4.3 se especifica el flujo de procesamiento para la detección de carril bajo métodos tradicionales.

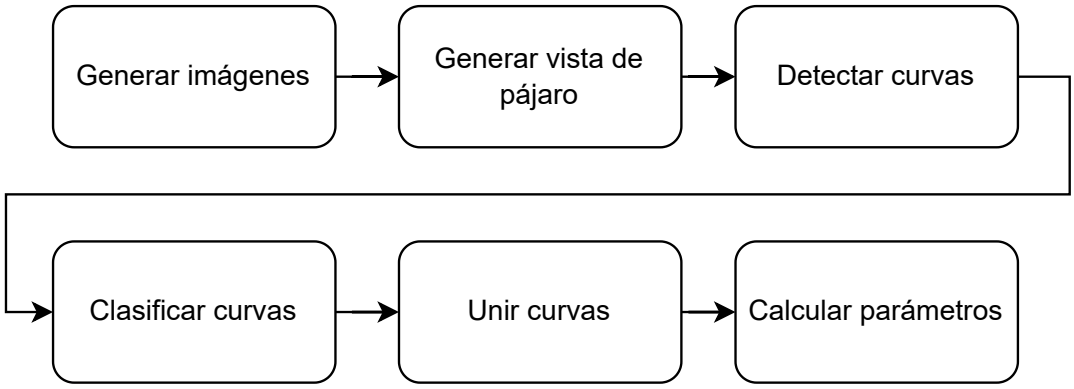


Figura 4.3: Flujo de procesamiento para la detección del carril. Fuente: elaboración propia.

Generar imágenes

Para esto se utilizan los sensores mencionados en la sección del diseño del escenario de simulación. Corresponde a 4 imágenes de 1080x1920 px.

Generar vista de pájaro

Se le aplica las funciones `birdsEyeView` y `transformImage` a cada una de las imágenes, para luego crear un montaje de las 4 imágenes en una sola. Con esto se logra captar la mayor cantidad de información de la autopista.

Detectar curvas

Para detectar las curvas primero la vista de pájaro se convierte a escala de grises, luego se aplica una máscara con `imfilter` que filtra aquellos pixeles en los que hay una transición lateral de colores, posteriormente se suaviza la imagen con `imgaussfilt` y se detecta los bordes haciendo uso de `edge`. De los bordes detectados se elimina los bordes cortos mediante `bwareaopen`, los bordes restantes se dilatan con `imdilate` y se unen con `bwconncomp` aquellos bordes que se intersecan entre sí mismos. A cada borde detectado se le calcula un ajuste polinómico de grado 2, con lo que se obtiene la curva.

Clasificar curvas

Una vez detectadas las curvas es posible clasificarlas según su tipo (individual, doble), continuidad (continua o discontinua) y color (blanca o amarilla).

Unir curvas

Aquellas curvas que se encuentran a cierto radio de cercanía con otras curvas y son de las mismas características (tipo, continuidad y color), se unen en una sola curva. Esto le da mayor estabilidad y precisión a las lecturas del carril.

Calcular parámetros

A partir de las dos curvas más cercanas al carril se genera una curva resultante de ambas, a la cual se le aplica un ajuste polinómico de grado 2. Esta curva resultante representa el centro de la trayectoria del carril. A partir de la curva resultante se calcula la distancia más cercana hasta el centro de la imagen (centro del automóvil), esta distancia será la desviación lateral (e_1); para calcular el error en el ángulo de conducción se calcula la tangente al punto de la curva más cercano al centro de la imagen, el ángulo entre la tangente y el eje vertical de la imagen será el error en el ángulo de conducción (e_2). La curvatura ρ se obtiene aplicando la ecuación 2.1 sobre la curva resultante del ajuste polinómico. Es importante mencionar que en cada medición se aplica un filtrado por Hampel en la salida.

Los errores e_1, e_2, e_3 se ilustran en la figura 4.4.

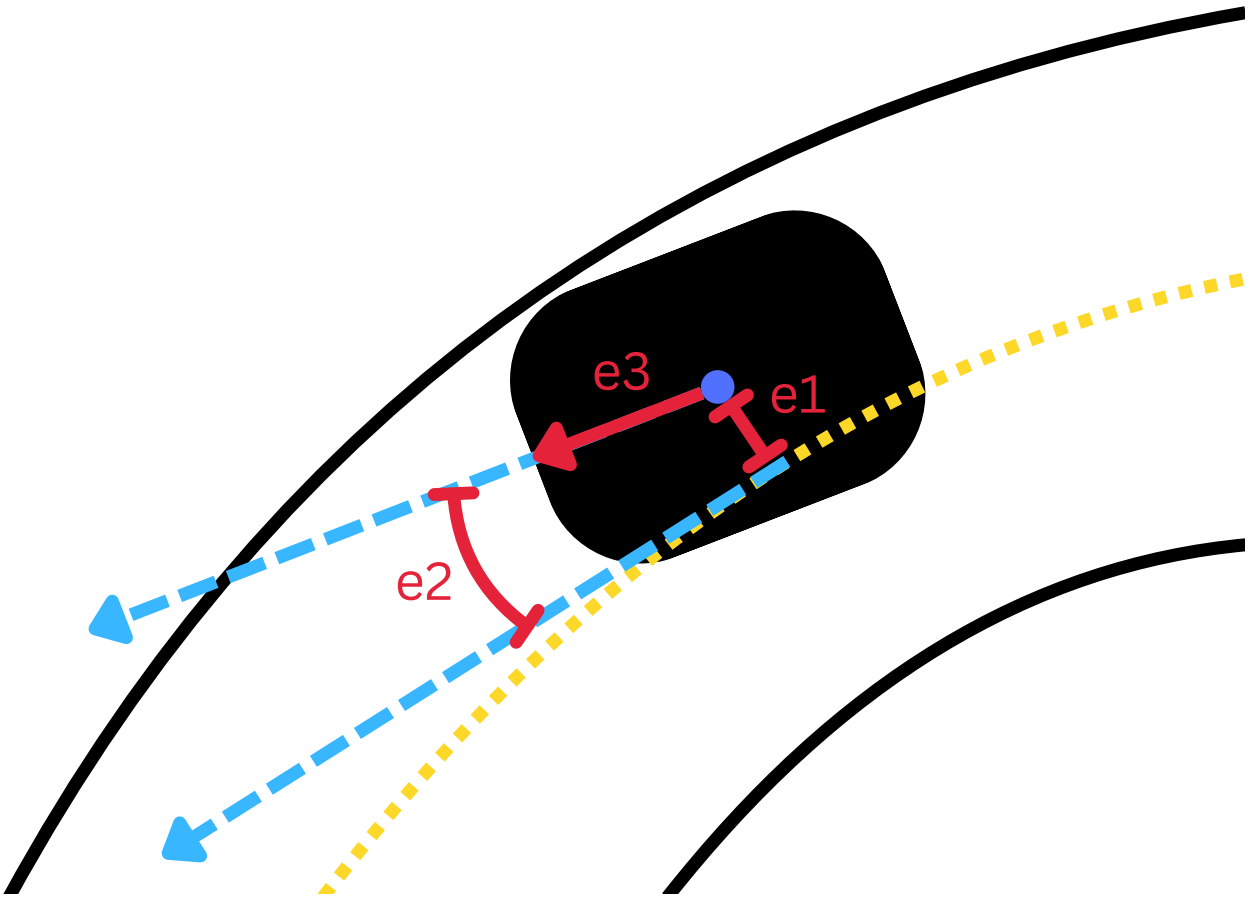


Figura 4.4: Los errores representados gráficamente en un carril. Fuente: elaboración propia.

4.4.2 Detección de vehículos

En la figura 4.5 se especifica el flujo de procesamiento para la detección de vehículos bajo métodos tradicionales.

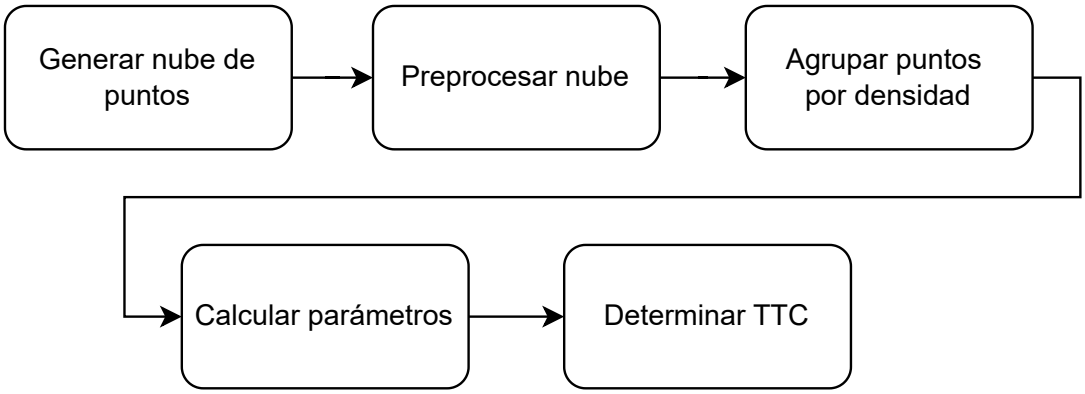


Figura 4.5: Flujo de procesamiento para la detección de vehículos. Fuente: elaboración propia.

Generar nube de puntos

El sistema de percepción obtiene la nube de puntos mediante un sensor LiDAR montado sobre el vehículo. Los 72000 puntos tridimensionales obtenidos luego de cada medición se organizan en un arreglo `[72000×3]` para su procesamiento posterior.

Preprocesar nube

La nube generada se somete a un proceso de limpieza y reducción. Se eliminan los puntos no finitos o con coordenadas nulas, se aplica un recorte espacial a una región de interés (ROI) definida en coordenadas del vehículo, y se excluyen los puntos ubicados dentro del área del propio automóvil (zona cerca del ego). Además, se aplica un filtro de reducción basado en vóxeles (Voxel Grid Filter), que agrupa puntos cercanos en celdas cúbicas y calcula el centroide de cada una. Esto permite reducir significativamente la cantidad de puntos manteniendo la estructura espacial esencial, lo que facilita el procesamiento en tiempo real.

Agrupar puntos por densidad

La nube resultante se segmenta mediante el algoritmo `dbscan`, que permite identificar agrupaciones de puntos (clusters) sin necesidad de especificar su número. Cada agrupación representa un posible objeto. Se calculan las dimensiones de cada grupo y se filtran aquellos cuya altura no esté dentro del rango característico de vehículos, así como los que se encuentran demasiado cerca del sensor. Para los grupos válidos, se identifica el punto más cercano al vehículo dentro de cada uno como su posición representativa.

Calcular parámetros

A partir de las posiciones (x_r, y_r) de los objetos detectados, se calcula la distancia relativa d_r de cada uno al vehículo mediante la norma euclidiana. Luego, utilizando su posición en el instante actual y en el instante anterior (considerando el tiempo de muestreo), se estima la velocidad relativa. Esta aproximación diferencia la posición entre cuadros consecutivos para cada objeto.

Determinar TTC

Finalmente, se calcula el tiempo hasta colisión (TTC) para todos los objetos detectados. Se selecciona el vehículo con menor TTC, y sus valores de distancia d_r y velocidad relativa v_r se toma como salidas del sistema. Es importante mencionar que en cada medición se aplica un filtrado por Hampel en la salida.

Capítulo 5

Resultados y Análisis

Para la comprobación del sistema de conducción autónoma se llevó a cabo pruebas en el modelo dinámico vehicular, el agente de aprendizaje por refuerzo y el sistema de percepción. Primero se valida el modelo dinámico vehicular, luego se entrena una versión inicial del agente de aprendizaje por refuerzo en un entorno de simulación con lecturas de percepción teóricas sobre las cuales es fácil verificar el funcionamiento correcto del agente, la validación inicial de dicho agente es la fase ideal para probar modelos de espacios de observación, ecuaciones de recompensa, entre otros aspectos del diseño de agente de aprendizaje por refuerzo. Más adelante se valida el sistema de percepción, el cual se encarga de transformar las lecturas de los sensores del automóvil del entorno 3D en las observaciones del agente. Finalmente se entrena el agente de aprendizaje por refuerzo en dicho entorno 3D y se realizan pruebas de validación para comprobar su funcionamiento.

5.1 Especificaciones de hardware y software

La validación del sistema se llevó a cabo utilizando un computador portátil con las características y los programas detallados a continuación. El equipo fue adquirido previamente por el estudiante.

1. Sistema operativo: Windows 11 Pro.

2. Versión de MATLAB: R2024b.

3. CPU: Mobil AMD Ryzen™ 7 7735HS (8 nucleos /16 hilos, 16MB L3 caché, hasta 4.7 GHz max boost).

4. GPU: NVIDIA® GeForce RTX™ 4050 Laptop GPU, 2370MHz* at 140W (2320MHz Boost Clock+50MHz OC, 115W+25W Dynamic Boost), GDDR6 de 6 GB.

5. Memoria: 16GB DDR5-4800 SO-DIMM x 2.

5.2 Validación del modelo dinámico vehicular

Con el fin de verificar el funcionamiento adecuado del modelo dinámico vehicular es importante compararlo ante un modelo más realista. Para esto se lleva a cabo una serie de experimentos mediante la plataforma Webots, la cual integra un motor físico que representa de mejor forma el movimiento de un automóvil.

5.2.1 Validación mediante Webots

Para la validación se comparó la velocidad longitudinal y aceleración lateral del modelo implementado en MATLAB contra un automóvil simulado mediante Webots, como se puede ver en la figura 5.1. Este escenario corresponde al modelo de ejemplo "City" (Ciudad) brindado por la plataforma.

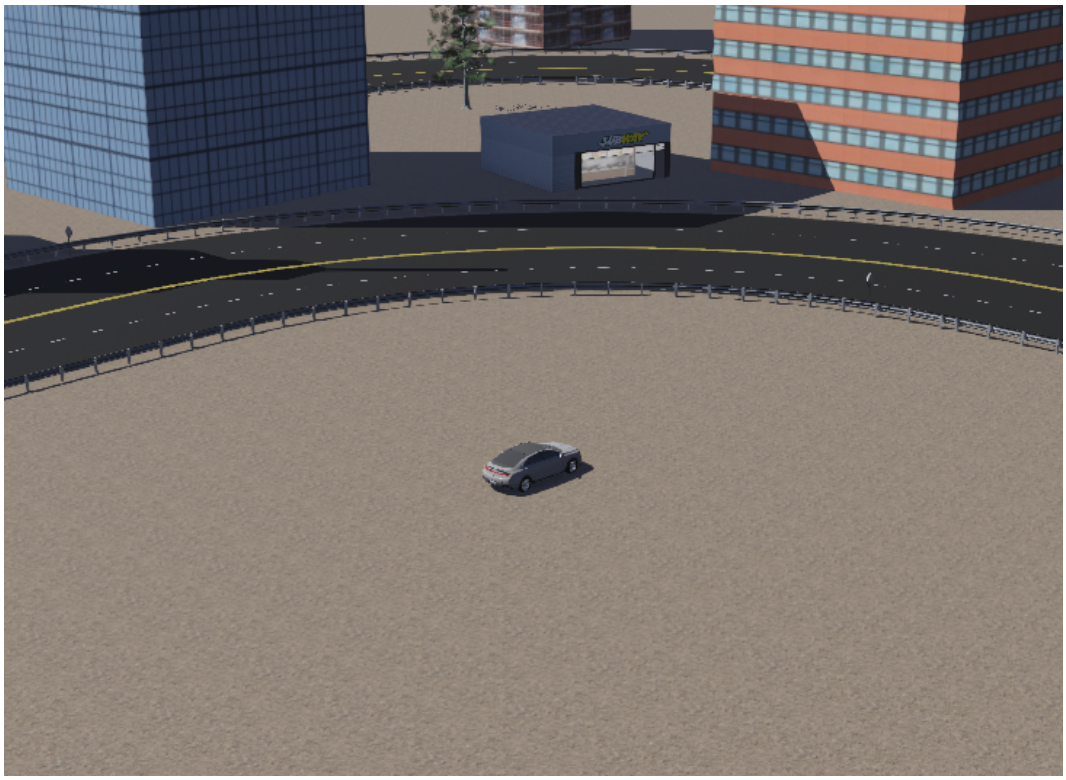


Figura 5.1: Escenario para las pruebas del modelo dinámico vehicular. Fuente: elaboración propia a partir de la plataforma Webots.

La prueba consiste en 3 ángulos de conducción para 5 velocidades longitudinales (2 en el caso del ángulo más cerrado). Para cada velocidad longitudinal y ángulo de conducción se calcula la aceleración lateral, los resultados se muestran en el cuadro 5.1.

Tabla 5.1: Comparación de velocidad longitudinal y aceleración lateral entre Webots y MATLAB.

Ángulo de conducción [rad]	**Webots**		**MATLAB**	
	Velocidad longitudinal [m/s]	Aceleración lateral [m/s^2]	Velocidad longitudinal [m/s]	Aceleración lateral [m/s^2]
0.05	1	0.02	0.99	0,02
	2	0.07	1.99	0,07
	3	0.15	2.99	0,15
	4	0.27	3.99	0,27
	5	0.42	4.98	0,40
0.2	1	0.03	0.99	0,03
	2	0.14	1.99	0,14
	3	0.31	2.97	0,31
	4	0.54	3.96	0,53
	5	0.84	4.93	0,79
0.5	1	0.18	0.89	0,15
	2	0.74	1.98	0,75

El error absoluto para cada medición de velocidad y ángulo de conducción se muestra en el cuadro 5.2.

Tabla 5.2: Errores absolutos en velocidad longitudinal y aceleración lateral.

Ángulo de conducción [rad]	**Velocidad longitudinal [m/s]**	**Error absoluto de velocidad long. [m/s]**	**Error absoluto de aceleración lat. [m/s^2]**
0.05	1	0.01	0.00
	2	0.01	0.00
	3	0.01	0.00
	4	0.01	0.00
	5	0.02	0.02
0.2	1	0.01	0.00
	2	0.01	0.00
	3	0.03	0.00
	4	0.04	0.01
	5	0.07	0.05
0.5	1	0.11	0.03
	2	0.02	-0.01

Según el cuadro 5.3 se obtiene un error absoluto máximo para la velocidad longitudinal y aceleración lateral de 0.03 m/s y 0,01 m/s^2, respectivamente. Estos valores son menores que los requeridos de 0.8 m/s y 0.8 m/s^2, con lo que se cumple con las métricas #9 y #10 y se da como validado el modelo dinámico vehicular.

Tabla 5.3: Resumen de errores en velocidad longitudinal y aceleración lateral.

	Velocidad longitudinal [m/s]	**Aceleración lateral [m/s^2]**
Error absoluto promedio	0.03	0.01
Error absoluto máximo	0.11	0.05

5.2.2 Pruebas de aceleración y frenado

También se hace una prueba de aceleración para verificar el funcionamiento adecuado del modelo. Ya que todo automóvil tienen un perfil de aceleración distinto, se brinda una verificación con el valor teórico de la aceleración. Las pruebas consisten en realizar la lectura de la aceleración longitudinal para un valor teórico de -1, -0.5, 0.5 y 1 m/s^2. Los resultados se pueden ver en el gráfico 5.2. Como se puede apreciar, para los valores de aceleración positiva, se tarda aproximadamente 3 segundos en llegar al valor de aceleración final, por otro lado, para el frenado (aceleración negativa) se observa como alcanza el valor máximo y desciende hasta que el auto llega al estado de reposo, en un período de aproximadamente 7 segundos.

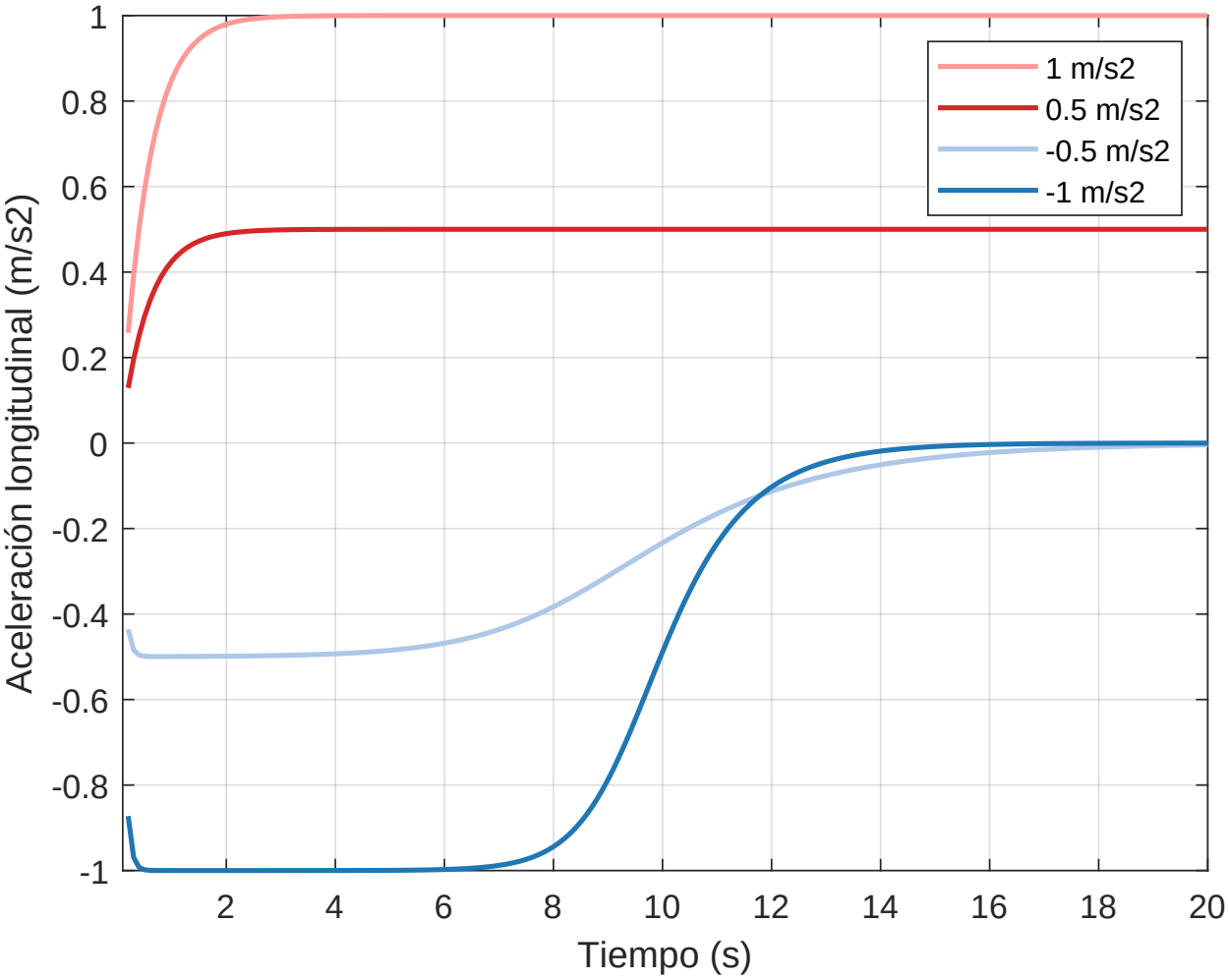


Figura 5.2: Resultados de las pruebas de aceleración. Fuente: elaboración propia.

Una vez validado el modelo dinámico se puede proceder con la validación del agente.

5.3 Validación inicial del agente de aprendizaje por refuerzo

El escenario utilizado para la validación inicial se muestra en la figura 5.3. Las posiciones iniciales del automóvil se muestran en azul con la flecha indicando la dirección. La posición inicial del auto se varía de forma aleatoria en un rango de 0.1 metros, mientras que el ángulo de guiñada inicial se varía en un rango de 0.025 radianes, igualmente para la validación final del agente. Esto se hace con el fin de fomentar la generalización del aprendizaje del agente durante el entrenamiento.

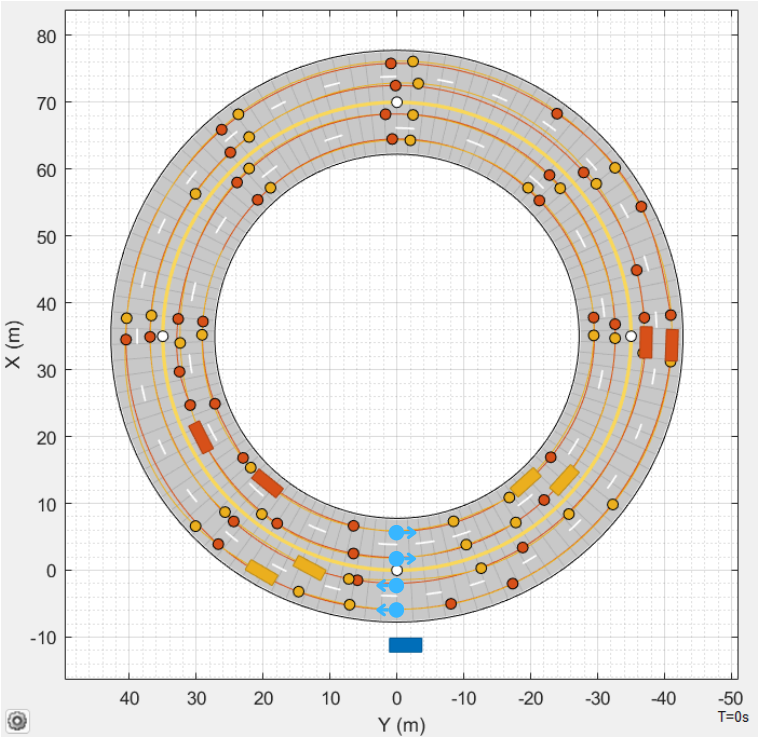


Figura 5.3: Escenario de validación inicial. Fuente: elaboración propia.

Se ejecuta el entrenamiento y en la iteración 768 se logra apreciar una buena recompensa, la cual además se estabiliza alrededor de un valor de 90, como se muestra en la figura 5.4. Debido a esto, se revisa mediante simulaciones el comportamiento del agente, en donde se determina que el agente mantiene una velocidad adecuada, conduce sin salirse del carril y tampoco impacta a otros vehículos durante las simulaciones realizadas, por lo que se procede a realizar la validación.

En la figura 5.5 se puede apreciar la trayectoria tomada por el agente durante las simulaciones. La validación previa se realiza visualmente, la validación del controlador con resultados numéricos se muestra en la validación final.

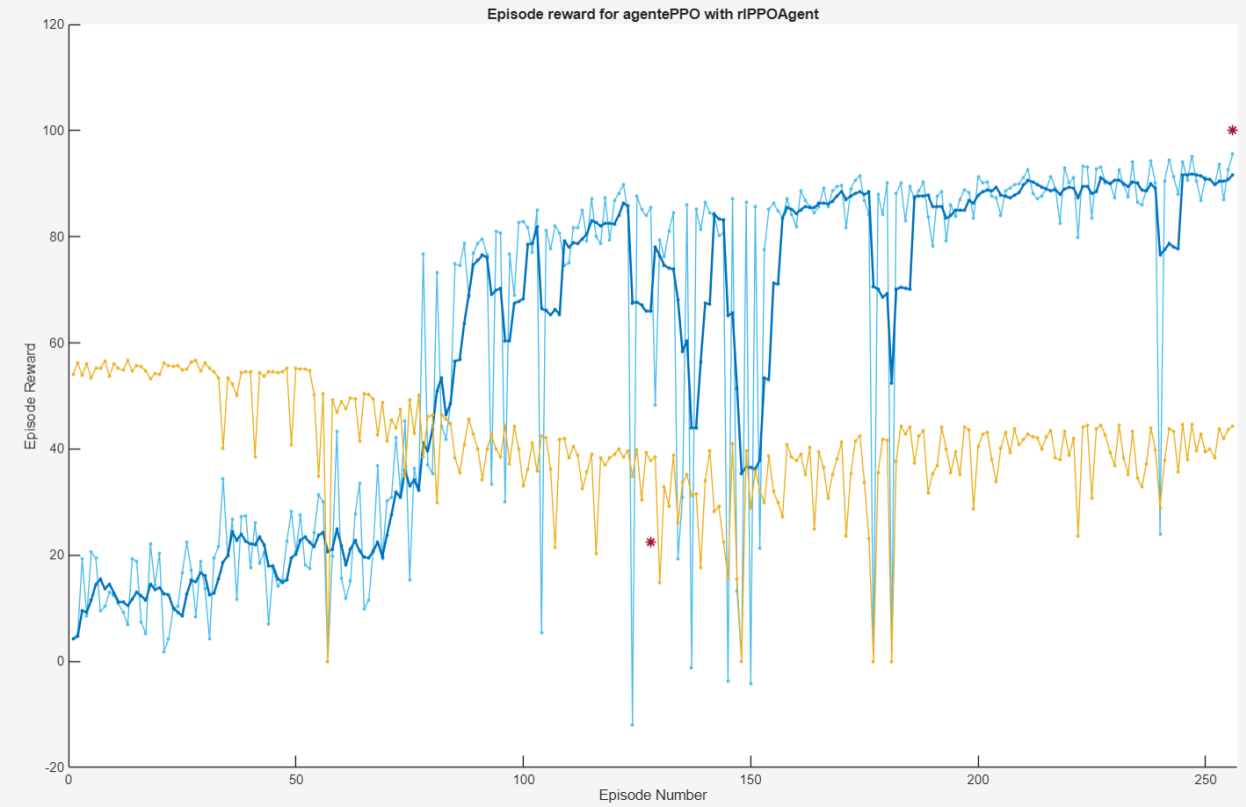


Figura 5.4: Recompensa en la sección final de 256 episodios de entrenamiento durante la fase de validación inicial. La línea azul corresponde a la recompensa por episodio, la celeste a la recompensa promedio. Fuente: elaboración propia a partir de la plataforma de entrenamiento de MATLAB.

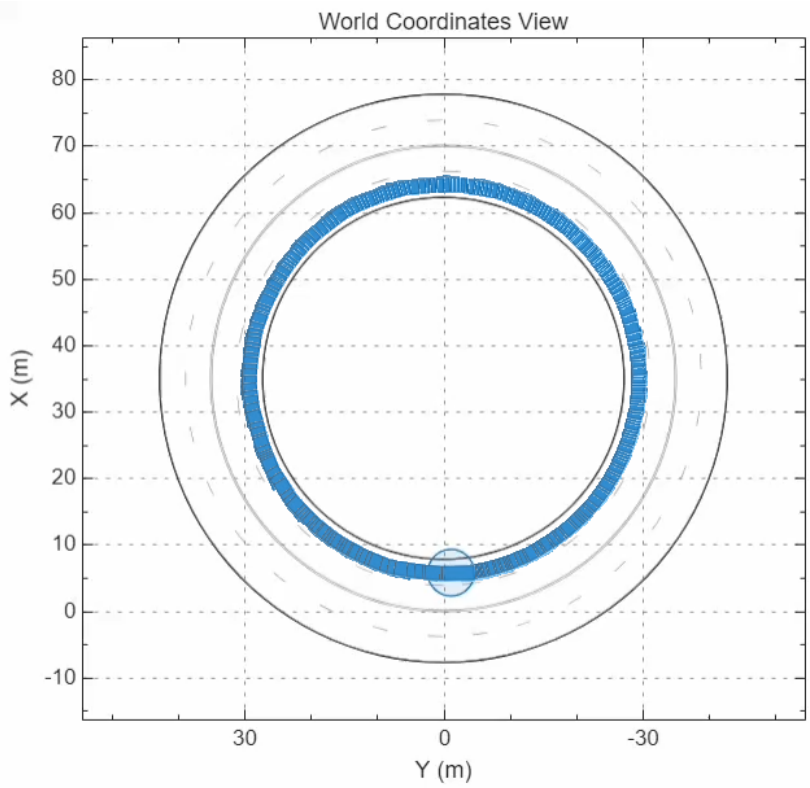


Figura 5.5: Trayectoria tomada por el agente durante la validación inicial. Fuente: elaboración propia.

5.4 Validación del sistema de percepción

Una vez validado el comportamiento del agente se procede con el diseño del sistema de percepción, dicho sistema sustituirá las observaciones generadas por el Driving Scenario Designer y tiene el fin de representar mediciones más realistas para entrenar el agente de aprendizaje por refuerzo.

5.4.1 Detección de carriles

Para la detección de carriles se sigue el flujo especificado en el diagrama 4.3. En la figura 5.6 se muestra un caso para el procesamiento de las imágenes a partir de las cámaras.

En la imagen 1 se muestra las 4 fotos tomadas por las cámaras, a partir de estas se genera la vista de pájaro de la imagen 2, con la vista de pájaro es posible identificar las curvas que se ven de verde en la imagen 3, en la imagen 4 se ven las curvas clasificadas según su continuidad (continua, discontinua o desconocida) y su color (amarilla, blanca o desconocida), para darle más estabilidad a las lecturas se hace un unión de las líneas discontinuas (como la curva #3) y de aquellas líneas que hayan sido fracturadas (como la curva #4). Finalmente se genera la curva resultante (línea del centro de carril color café) como se muestra en la imagen 6 con la cual se puede obtener la desviación lateral, el error en el ángulo de conducción y la curvatura de la autopista.

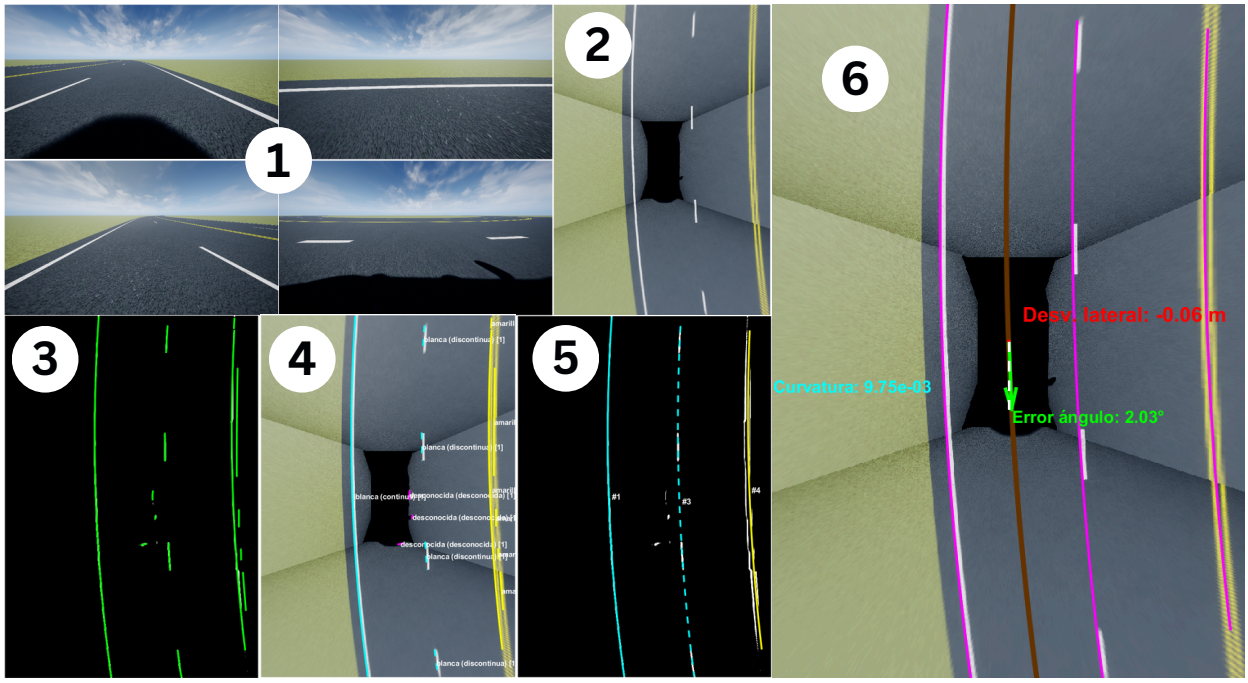


Figura 5.6: Flujo de procesamiento de las imágenes. Fuente: elaboración propia.

Para la validación de la detección de carriles se llevan a cabo dos pruebas: en autopista recta y en autopista curva.

Pruebas en autopista recta

En una autopista recta es fácil definir la desviación lateral para realizar pruebas: se elige el centro del carril y a partir de ahí se define la desviación lateral, en este caso se eligió valores desde -0.75m hasta 0.75m cada 0.25m. Los resultados se muestra en el cuadro 5.4.

Tabla 5.4: Errores absolutos de la desviación lateral.

Desviación lateral [m]	-0.75	-0.5	-0.25	0	0.25	0.5	0.75
Error absoluto máximo [m]	0.03	0.03	0.03	0.02	0.02	0.05	0.02
Error absoluto promedio [m]	0.03	0.03	0.03	0.00	0.00	0.01	0.01

Para revisar si la el tamaño de muestra fue suficiente se utiliza la estimación basada en el valor de z para una confianza del 95% con una distribución normal, el cual tiene un valor aproximado de 1.96 según la tabla disponible en [67], la desviación estándar y el error absoluto máximo como se indica en la ecuación 5.1.

$$N = (\frac{1.96 \cdot \sigma}{E})^2 \tag{5.1}$$

Donde N es el tamaño de muestra necesario, σ es la desviación estándar y E el error absoluto máximo.

Según los resultados mostrados en el Cuadro 5.5, se obtuvo una confianza del 100% para un error absoluto inferior a 0.2 m. De acuerdo con la Ecuación 5.1, una desviación estándar de 0.01 m implica que el tamaño de muestra requerido es de al menos una medición. Esta bajo número de muestras es consecuencia directa de la baja variabilidad observada. Debido a que la cantidad de muestras es inferior a la usada (427 muestras) y el porcentaje de confianza es el adecuado, se considera satisfecha la métrica #3 en estas pruebas.

Tabla 5.5: Estadísticas de la desviación lateral.

Desviación estándar [m]	0.01
Tamaño de muestra	427
Porcentaje de confianza [%]	100

De igual forma, se definen 7 errores en el ángulo de conducción desde 15° hasta -15° cada 5°. Los resultados se muestran en el cuadro 5.6.

Tabla 5.6: Errores absolutos en grados del error de ángulo de conducción.

Error en el ángulo de conducción [°]	15	10	5	0	-5	-10	-15
Error absoluto máximo [°]	0.16	0.44	0.62	0.18	0.16	0.22	5.70
Error absoluto promedio [°]	0.12	0.11	0.12	0.15	0.11	0.10	0.14

Según los resultados mostrados en el Cuadro 5.7, se obtuvo una confianza del 99% para un error absoluto inferior a 0.51°. De acuerdo con la Ecuación 5.1, una desviación estándar de 0.45° determina un tamaño de muestra mínimo de cuatro mediciones. Este número reducido de muestras se debe a la variabilidad moderada observada. Dado que la cantidad de muestras (427) utilizadas supera este mínimo y el nivel de confianza es adecuado, se considera satisfecha la métrica #4 en estas pruebas.

Tabla 5.7: Estadísticas del error de ángulo de conducción.

Desviación estándar [°]	0.45
Tamaño de muestra	427
Porcentaje de confianza [%]	99

Pruebas en carretera curva

En una autopista curva es más difícil definir una desviación lateral constante, por lo que se utiliza los sensores de la aplicación "Driving Scenario Designer" para realizar mediciones teóricas de los carriles. Se diseña una trayectoria (figura 5.7) y se registran los datos durante 40 s, los resultados se muestran en el cuadro 5.8.

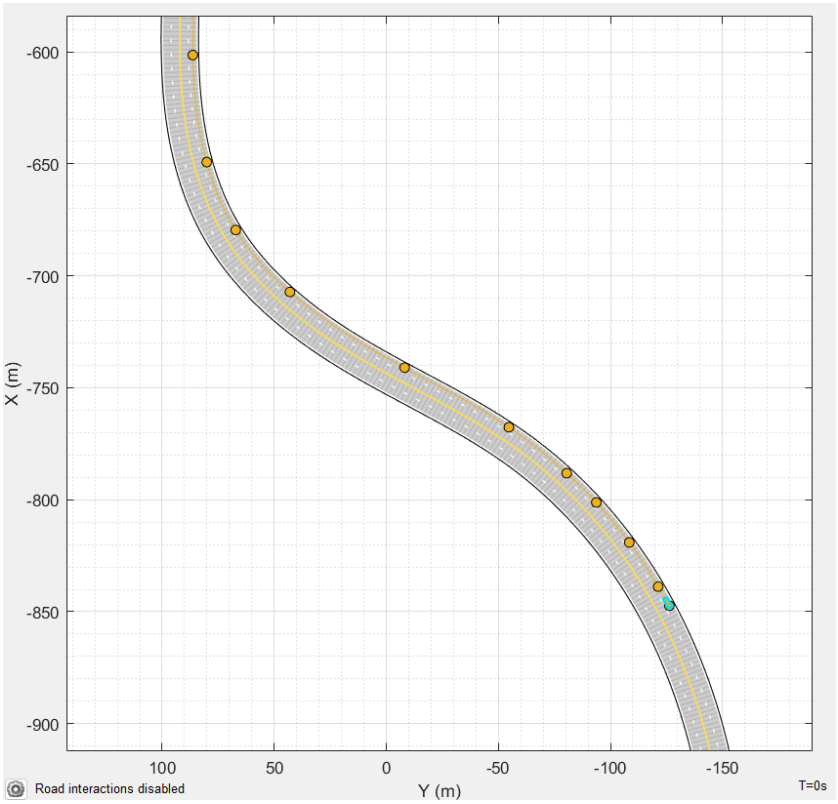


Figura 5.7: Trayectoria del ego para la validación de la detección de carril. Fuente: elaboración propia.

Tabla 5.8: Resumen de errores y estadísticas para las variables de localización.

	Desviación lateral [m]	**Error ángulo [°]**	**Rho [1/m]**
Error absoluto máximo	0.09	0.55	0.00139
Error absoluto promedio	0.03	0.33	0.00056
Desviación estándar	0.02	0.17	0.00028
Tamaño de muestra	444	444	444
Porcentaje de confianza [%]	100	95	98

Según el Cuadro 5.8, se obtuvo una confianza del 100% en la medición de la desviación lateral, y del 95% en el error del ángulo de conducción, con desviaciones estándar de 0.02 m y 0.17°, respectivamente. Conforme a la ecuación 5.1, estas bajas desviaciones implican un tamaño de muestra requerido de al menos una medición en ambos casos, el cual es menor que las 444 muestras tomadas. Debido a esto, y a la confianza igual o superior al 95% obtenida, se considera satisfechas finalmente las métricas #3 y #4. Por otro lado, se logró una confianza del 98% para un error absoluto menor a 0.0012 m^{-1} en la medición de la curvatura.

5.4.2 Detección de vehículos

Para la detección de vehículos se sigue el flujo especificado en el diagrama 4.5. En la figura 5.8 se muestra un caso para el procesamiento de la nube de puntos a partir del LiDAR.

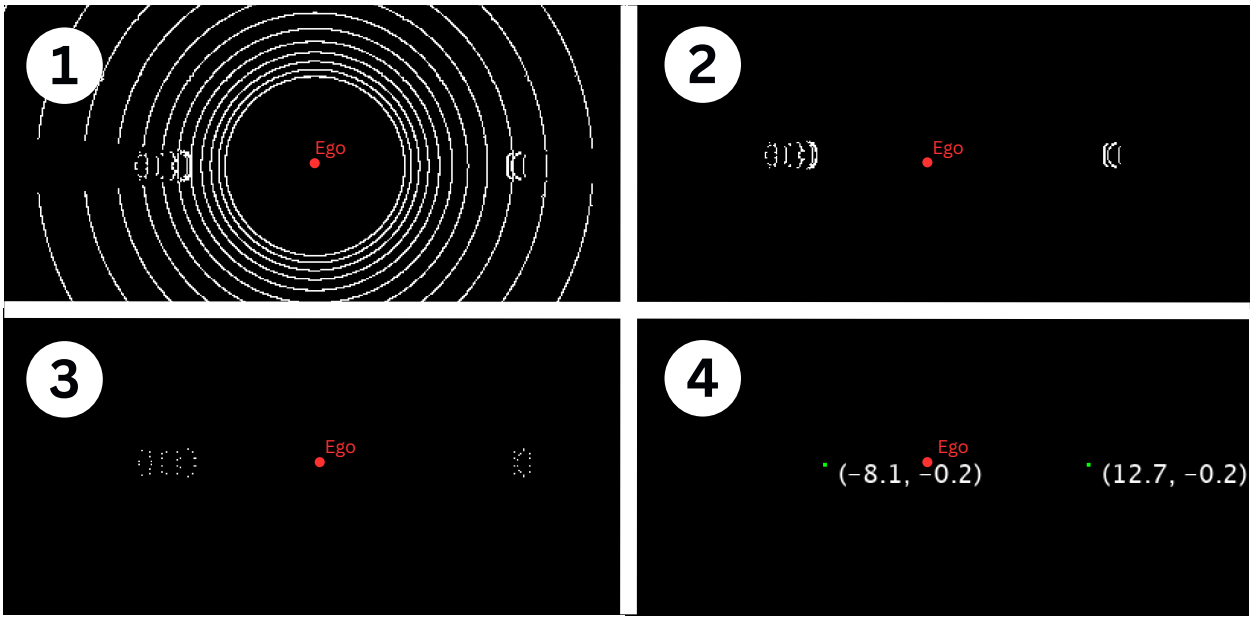


Figura 5.8: Flujo de procesamiento de la nube de puntos. Generar nube de puntos (1), preprocesar nube de puntos (2 y 3), agrupar puntos por densidad y calcular parámetros (4). Fuente: elaboración propia.

En la imagen 1 se puede apreciar el efecto de los anillos de retorno del LiDAR en las mediciones, estos efectos y otros no deseados se eliminar visiblemente en la imagen 2, luego se aplica la reducción por vóxeles y se obtiene la imagen 3 con una cantidad apreciable de puntos reducidos, finalmente en la imagen 4 se aplica el agrupamiento de los puntos por densidad y se calcula la distancia hasta el parachoques de cada vehículo (puntos verdes), a partir de la cual también se calcula la velocidad relativa de cada uno.

Medición de la distancia relativa

Se sitúan 2 automóviles a 5, 10, 15 y 30 metros longitudinal y lateralmente del ego y se mide la distancia relativa hasta el parachoques de cada auto, como se muestra en las figuras 5.9 y 5.10. Los resultados se muestran en el cuadro 5.9.

Tabla 5.9: Errores absolutos de la distancia relativa. X+, X- son las mediciones longitudinales positivas y negativas, Y+ y Y- para las laterales.

Distancia relativa [m]	**5 m**				**10 m**		**15 m**		**30 m**	
	X+	X-	Y+	Y-	Y+	Y-	X+	X-	X+	X-
Error absoluto máximo [m]	0.36	0.66	0.09	0.03	0.12	0.12	0.13	0.16	0.10	0.24
Error absoluto promedio [m]	0.36	0.66	0.09	0.03	0.12	0.12	0.13	0.16	0.10	0.24

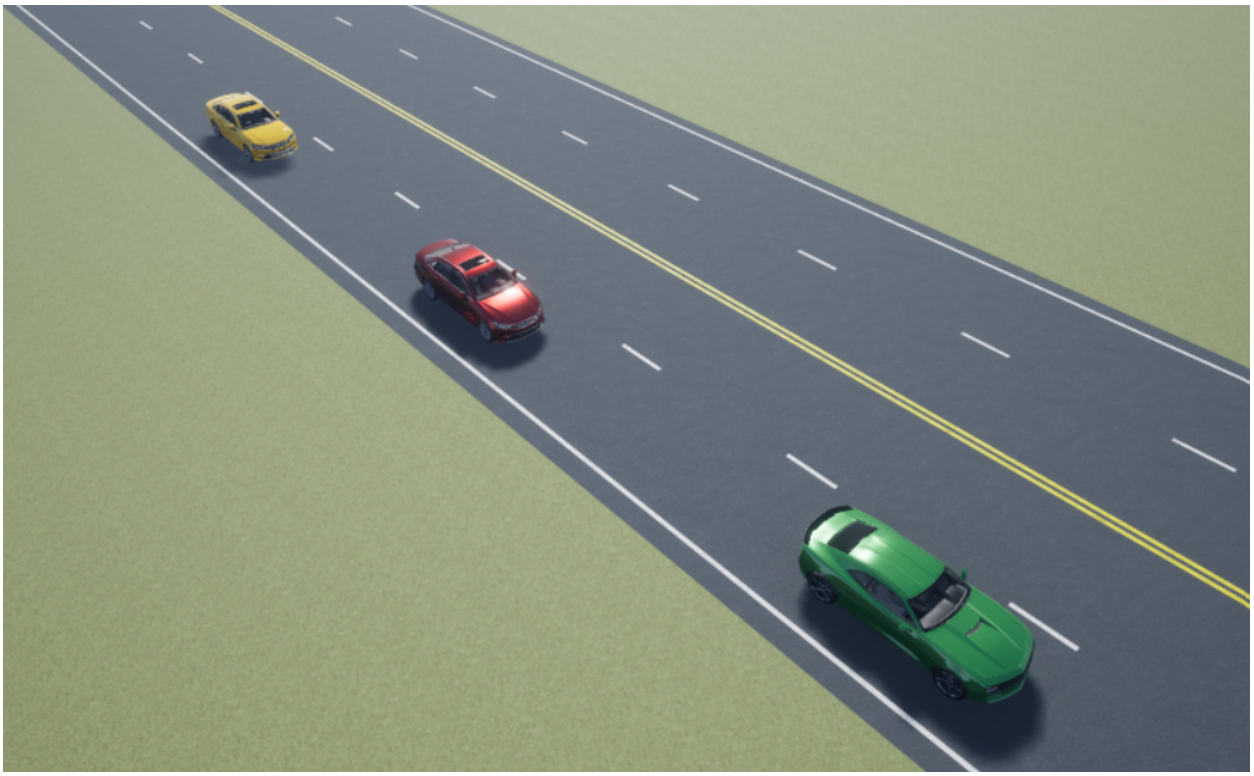


Figura 5.9: Escenario para la validación de la medición longitudinal de distancia relativa (ego vehículo del medio). Fuente: elaboración propia

Figura 5.10: Escenario para la validación de la medición lateral de distancia relativa. Fuente: elaboración propia.

Según el cuadro 5.10 se obtiene un porcentaje de confianza del 100% para un error absoluto promedio menor a 0.8 m y una desviación estándar de 0.18 m. Según la ecuación 5.1, se obtiene un tamaño de muestra necesario de al menos 1 medición. Ya que el porcentaje de confianza es superior al 95% y las mediciones fueron superiores a las necesarias, se considera satisfecha la métrica #5 en estas pruebas.

Tabla 5.10: Estadísticas de la posición relativa.

Desviación estándar [m]	0.18
Tamaño de muestra	740
Porcentaje de confianza [%]	100

Medición de la velocidad relativa

Para medir la velocidad relativa se le imparte una velocidad desde 2.5 hasta 15 m/s cada 2.5 m/s a dos vehículos (por el frente y por detrás), como se muestra en la figura 5.11. Los resultados se muestran en el cuadro 5.11.

Figura 5.11: Escenario para la validación de la medición de velocidad relativa. Fuente: elaboración propia

Tabla 5.11: Errores absolutos de la velocidad relativa.

Velocidad relativa [m/s]	2.5	5	7.5	10	12.5	15
Error absoluto máximo [m/s]	0.26	0.94	1.31	1.28	1.15	0.99
Error absoluto promedio [m/s]	0.06	0.20	0.24	0.23	0.22	0.24

Según el cuadro 5.12 se obtiene una confianza del 96% para un error absoluto promedio

menor a 0.8 m/s y una desviación estándar de 0.24 m. Según la ecuación 5.1, se obtiene un tamaño de muestra necesario de al menos 1 medición. Ya que el porcentaje de confianza es superior al 95% y el tamaño de muestra fue suficiente, se considera satisfecha la métrica #6 en estas pruebas.

Tabla 5.12: Estadísticas de la velocidad relativa.

Desviación estándar [m/s]	0.24
Tamaño de muestra	240
Porcentaje de confianza [%]	96

5.4.3 Pruebas del cálculo del tiempo para colisión (TTO)

El fin de calcular el TTO para cada automóvil es saber con cuál se va a impactar más pronto y determinar dicho automóvil como el "objeto de importancia" para las observaciones de distancia y velocidad relativa. El objetivo de esto es reducir el espacio de observaciones, lo cual llevará a tiempos de entrenamientos más bajos, pero tampoco sacrificar el funcionamiento adecuado del agente.

La primera prueba (figura 5.12) consiste en el ego y el automóvil de atrás con la misma velocidad y el automóvil del frente se encuentra quieto, lo que resulta en un impacto frontal. Como se ve en el gráfico 5.13, la distancia medida es positiva (lo que refleja la posición del automóvil al frente) y va decreciendo en magnitud hasta que se da el impacto. También se aprecia que el valor de la velocidad relativa es de 5 m/s, lo que corresponde a la velocidad relativa del automóvil del frente.

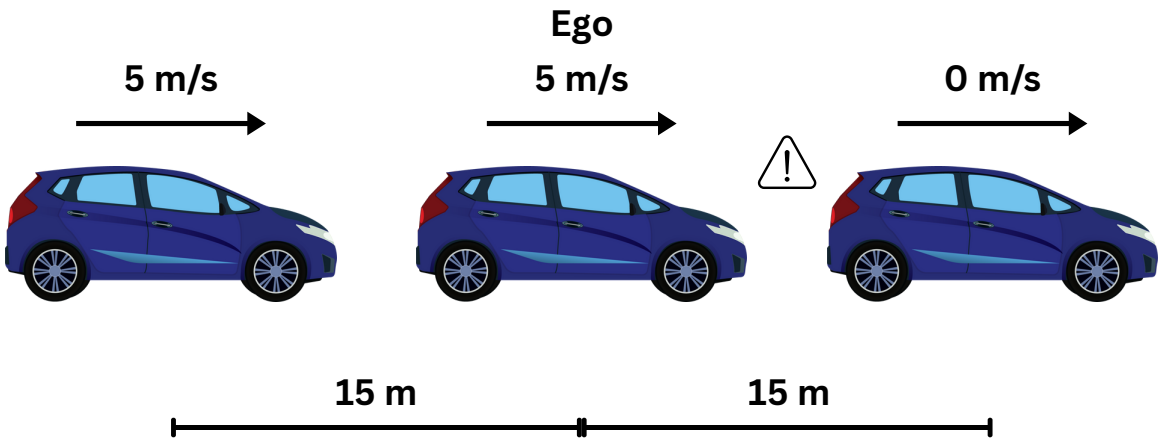


Figura 5.12: Prueba de impacto frontal. Fuente: elaboración propia.

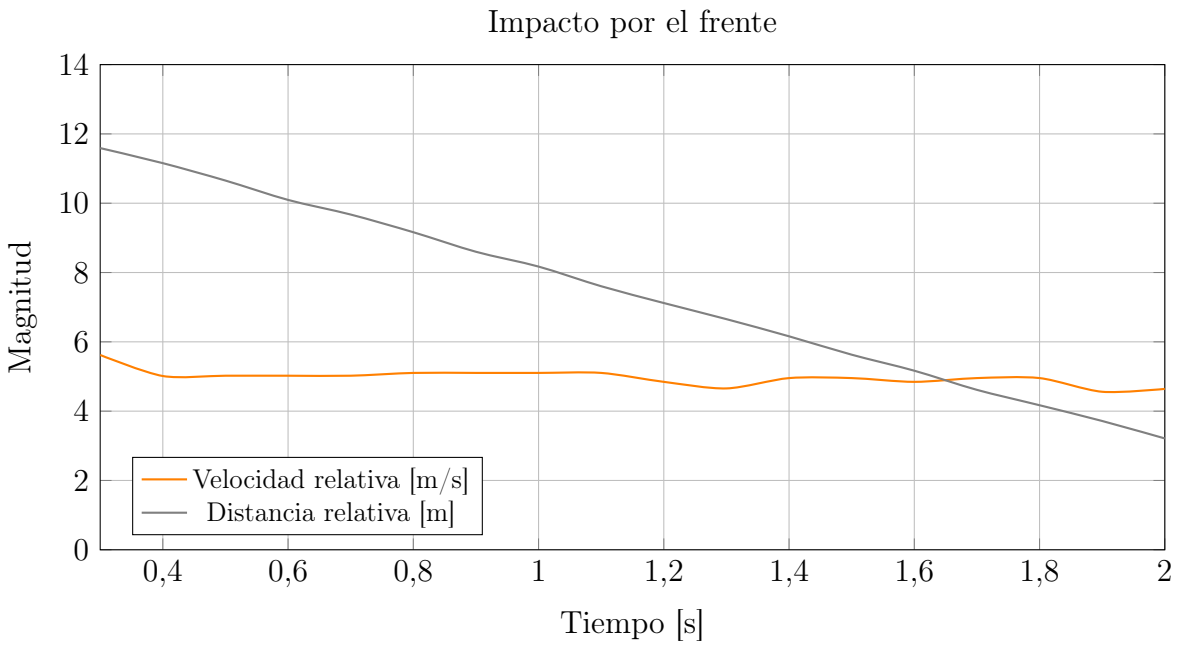


Figura 5.13: Evolución de velocidad y distancia relativa ante impacto frontal. Fuente: elaboración propia.

La siguiente prueba (figura 5.14) consiste en el ego y el automóvil del frente con la misma velocidad y el automóvil del atrás tiene el doble de velocidad, lo que resulta en un impacto trasero. Como se ve en el gráfico 5.15, la distancia medida es negativa (lo que refleja la posición del automóvil atrás) y va decreciendo en magnitud hasta que se da el impacto. También se aprecia que el valor de la velocidad relativa es de 5 m/s, lo que corresponde a la velocidad relativa del automóvil de atrás.

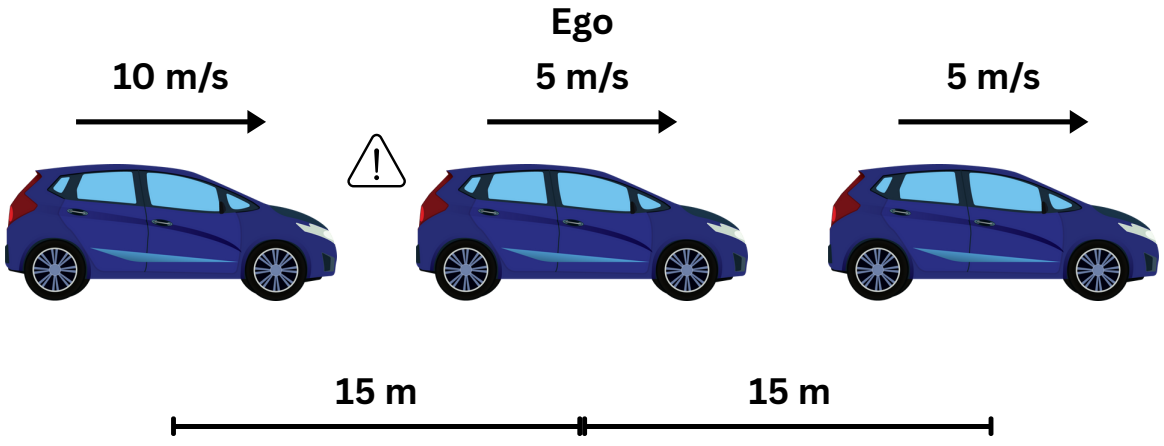


Figura 5.14: Prueba de impacto por detrás. Fuente: elaboración propia.

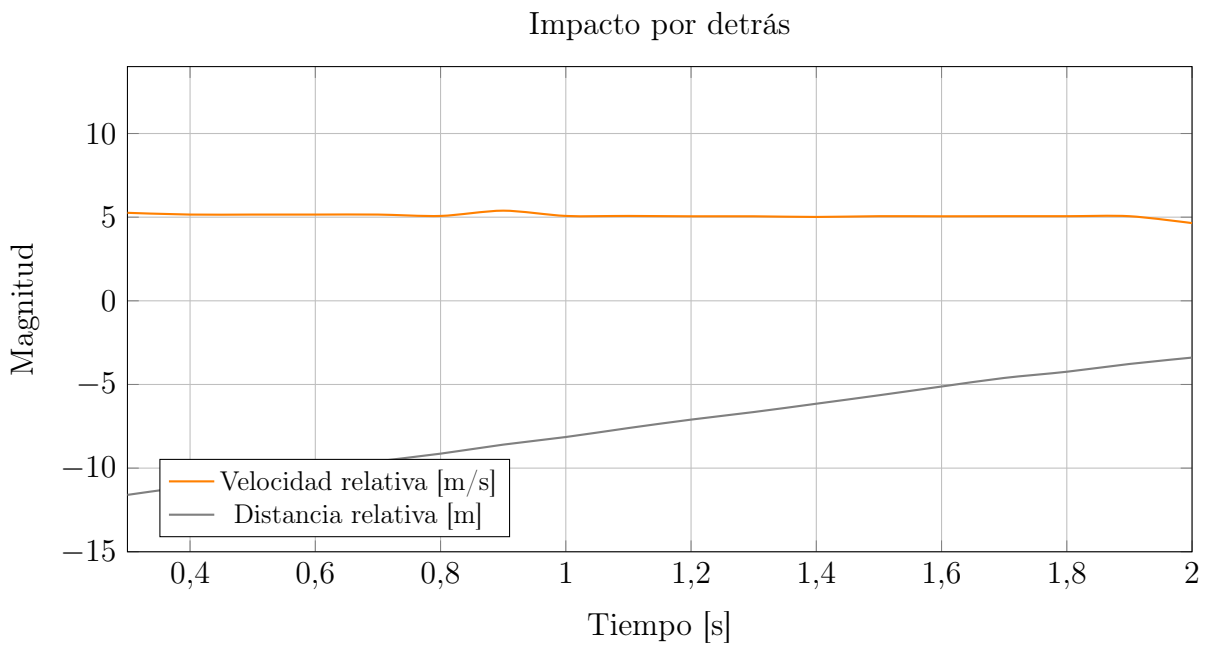


Figura 5.15: Evolución de velocidad y distancia relativa ante impacto trasero. Fuente: elaboración propia

La última prueba prueba (figura 5.16) consiste en el ego siendo impactado por dos automóviles que tienen la misma velocidad relativa a la misma distancia relativa por delante y por detrás, lo que resulta en el mismo TTO, debido a esto, las mediciones fluctúan entre el auto frontal y el trasero (distancia positiva y negativa) hasta que se da el choque (ver figura 5.17). También se aprecia que el valor de la velocidad relativa es de 2.5 m/s, lo que corresponde a la velocidad relativa del ego con respecto a los otros automóviles.

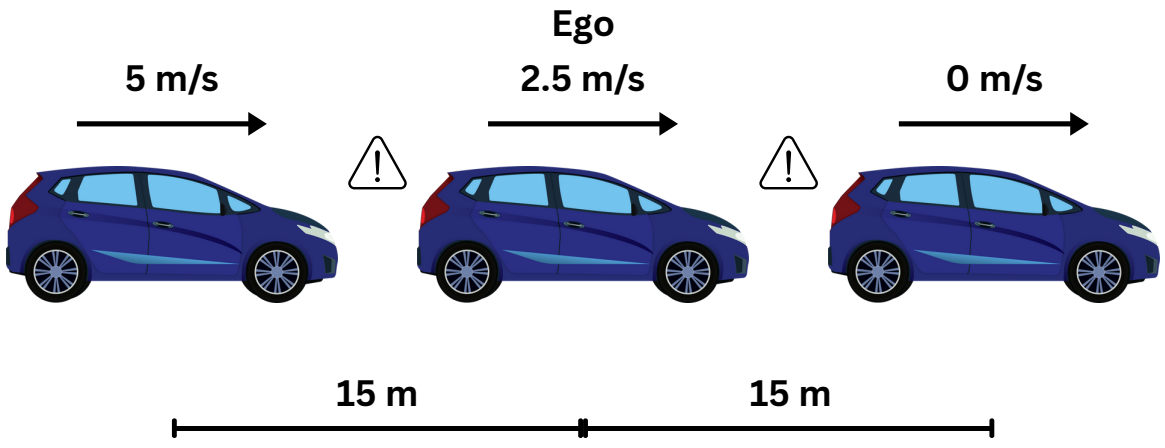


Figura 5.16: Prueba de impacto por ambos lados. Fuente: elaboración propia.

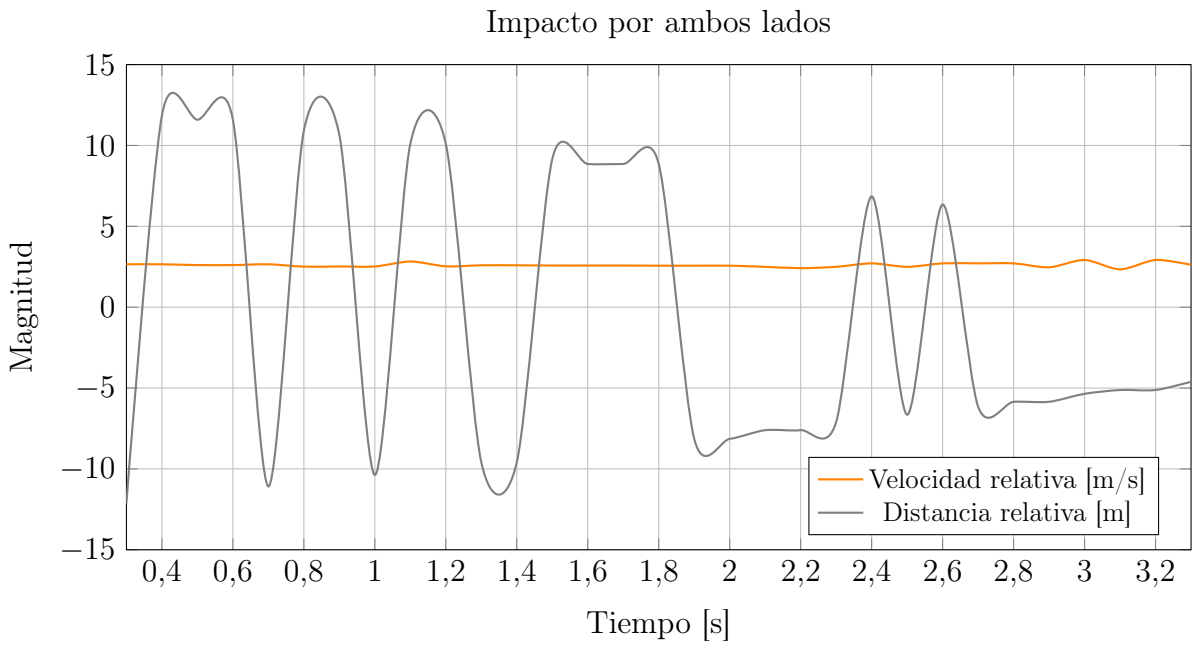


Figura 5.17: Evolución de velocidad y distancia relativa ante impacto por ambos lados. Fuente: elaboración propia.

5.5 Validación final del agente de aprendizaje por refuerzo

En la validación final se implementa el escenario 3D mostrado en la figura 5.18. Consta de una sección de la autopista Carretera Curva, atrás y al frente de la trayectoria del agente hay 2 vehículos a 2 m/s, una velocidad distinta a la objetivo de 5 m/s. La simulación 3D se puede apreciar en la figura 5.19

Luego de 1810 iteraciones de entrenamiento durante la fase final se obtiene una buena recompensa durante el entrenamiento, por lo que se decide probar el funcionamiento del agente en esta iteración. En los cuadros 5.20, 5.21, 5.22 se muestra los resultados del error absoluto para la desviación lateral, error en el ángulo de conducción y error de velocidad, respectivamente.

Para la desviación lateral el error máximo fue de 0.26 m, mientras que se mantuvo un error promedio de 0.02 m.

Para el ángulo de conducción el error máximo fue de 0.19°, mientras que se mantuvo un error promedio de 0.02°.

Para la velocidad objetivo el error promedio fue de 2.96 m/s. Un error cercano a 0 m/s conllevaría chocar otro auto al llevar el ego una velocidad superior que el resto de vehículos en su carril, por lo que el agente prefiere mantener errores de velocidad elevados antes que impactar con otros vehículos.

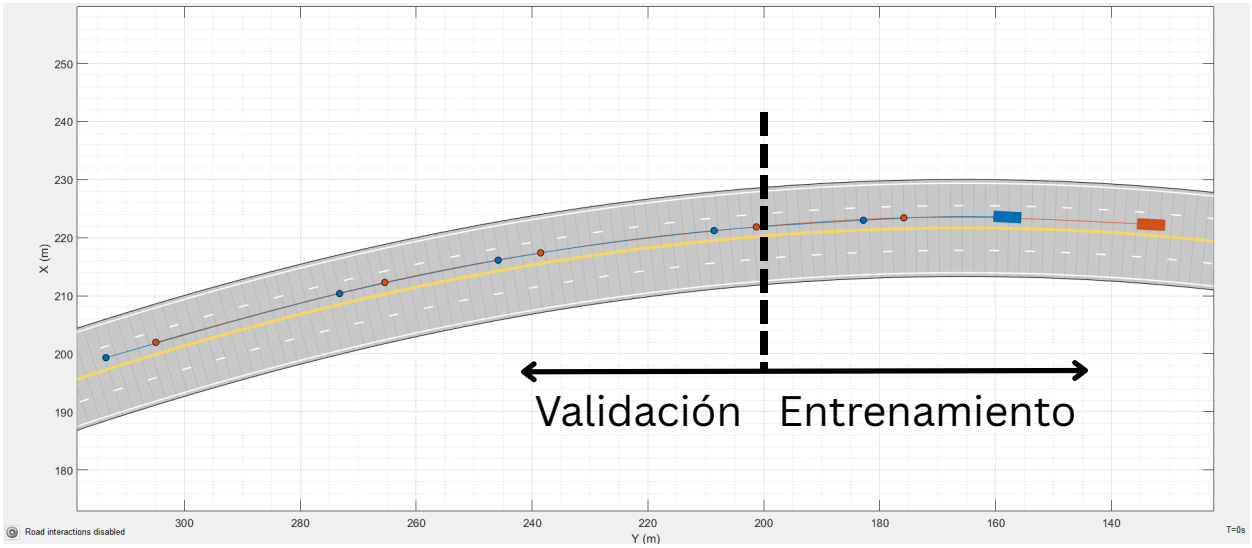


Figura 5.18: Escenario utilizado para la validación final. Vehículos en rojo y azul, el ego se sitúa en medio de ambos. Fuente: elaboración propia.

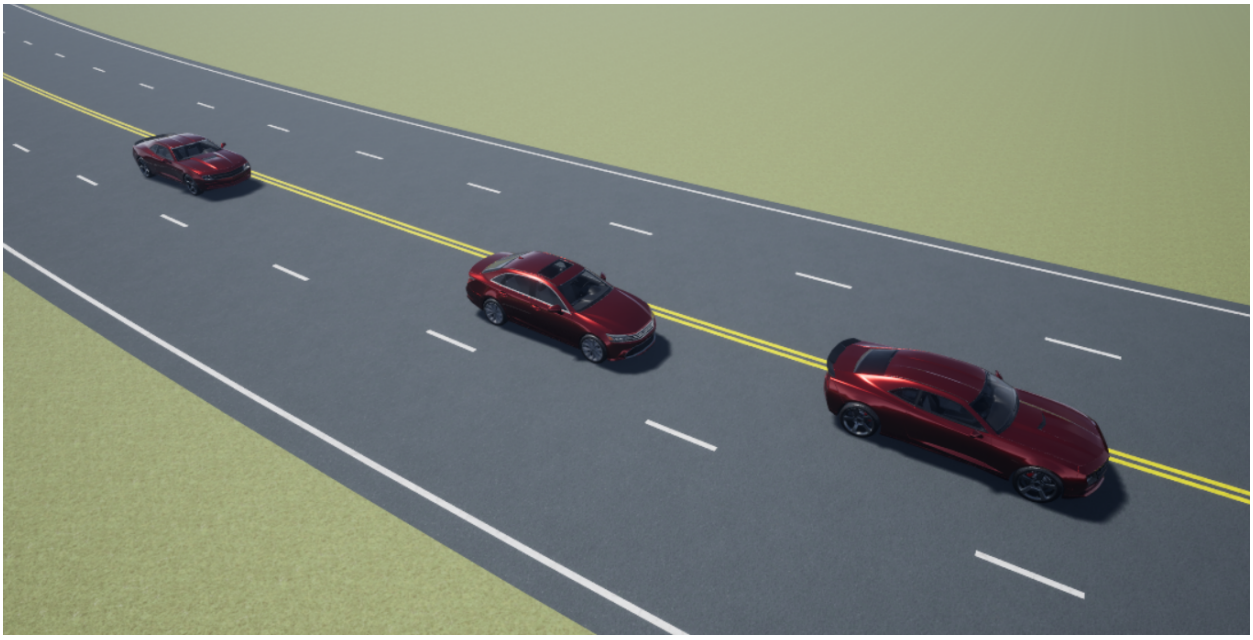


Figura 5.19: Escenario 3D utilizado para la validación final. El automóvil ego se sitúa en medio de los automóviles. Fuente: elaboración propia.

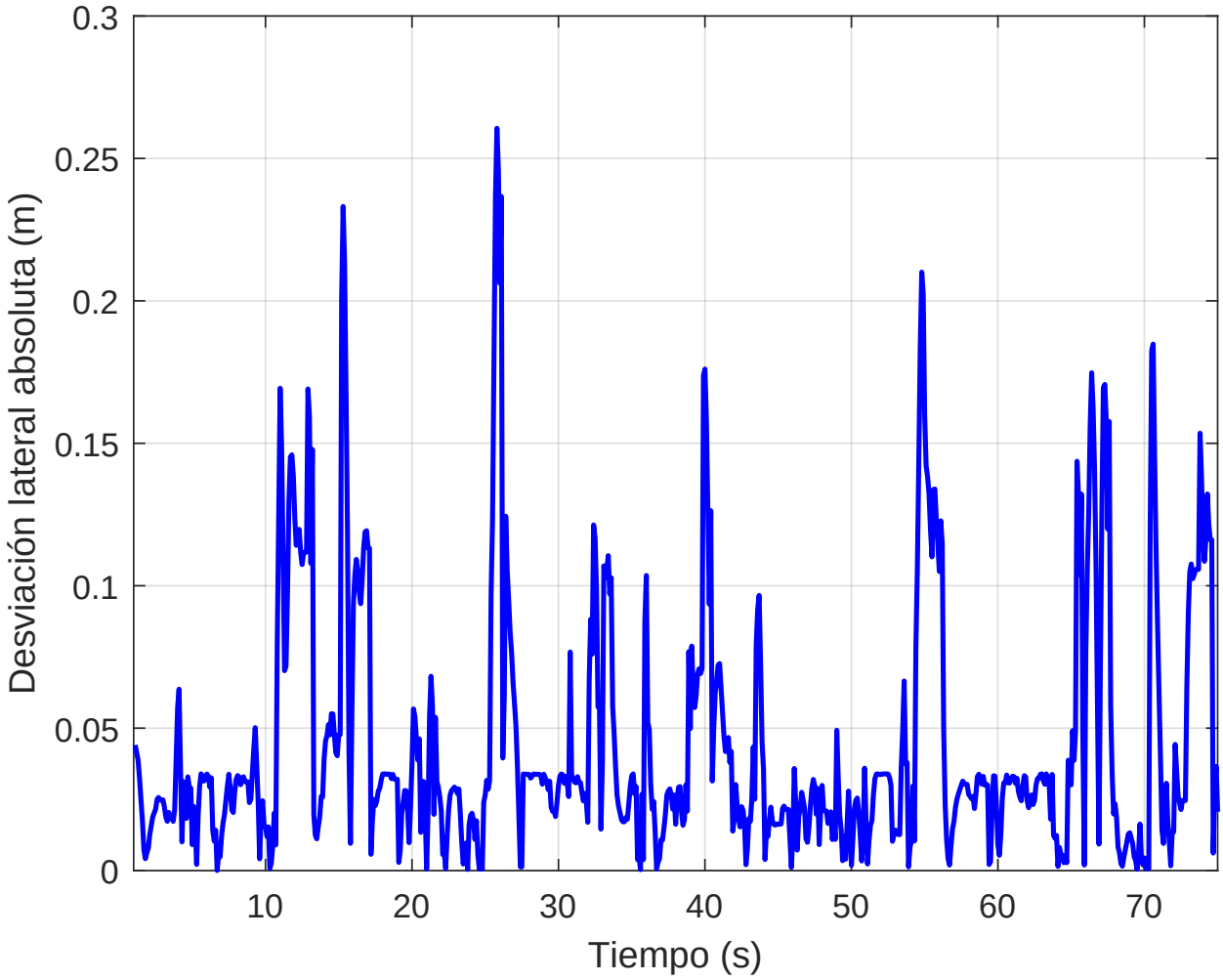


Figura 5.20: Error absoluto en la desviación lateral. Fuente: elaboración propia.

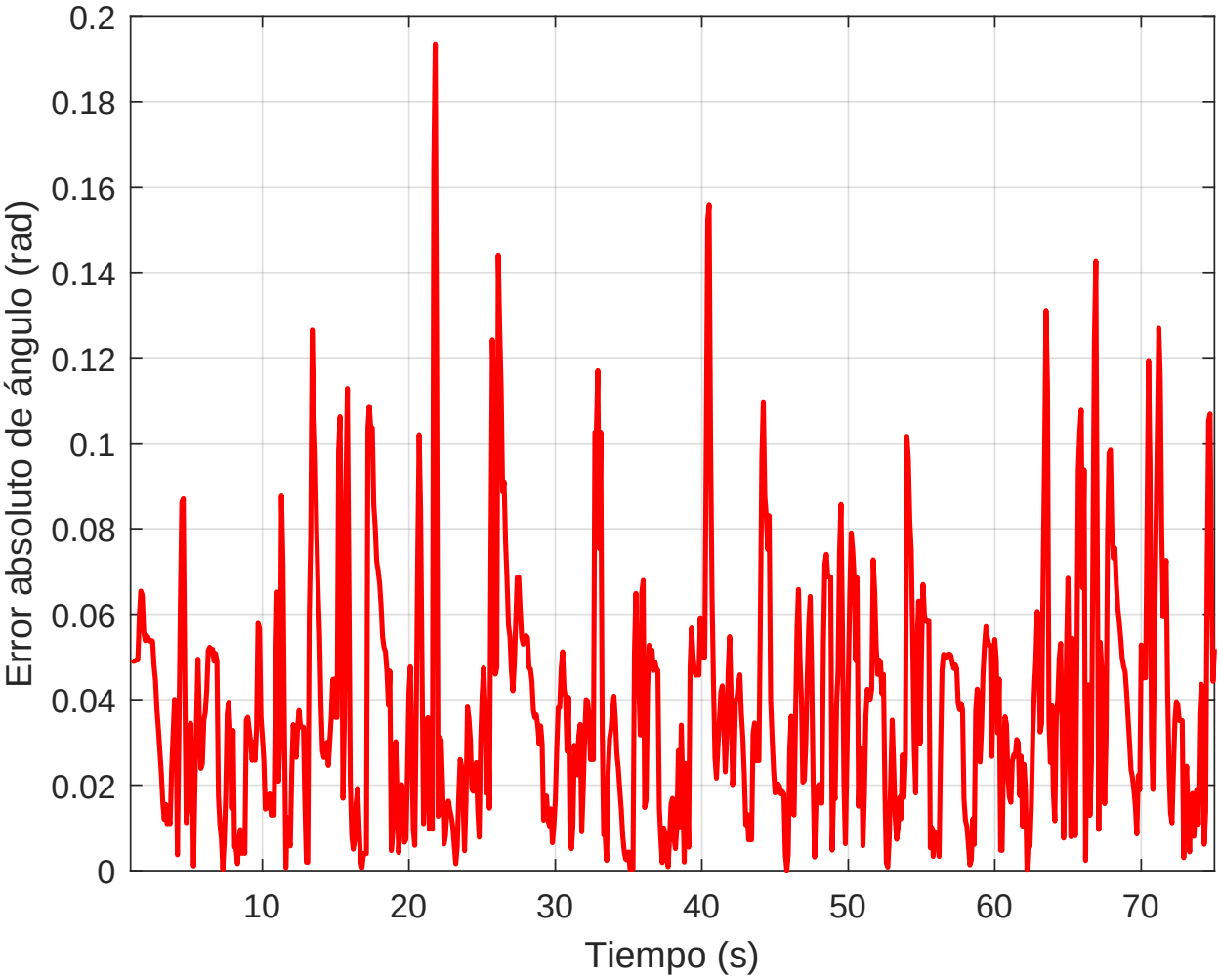


Figura 5.21: Error absoluto en el ángulo de conducción. Fuente: elaboración propia.

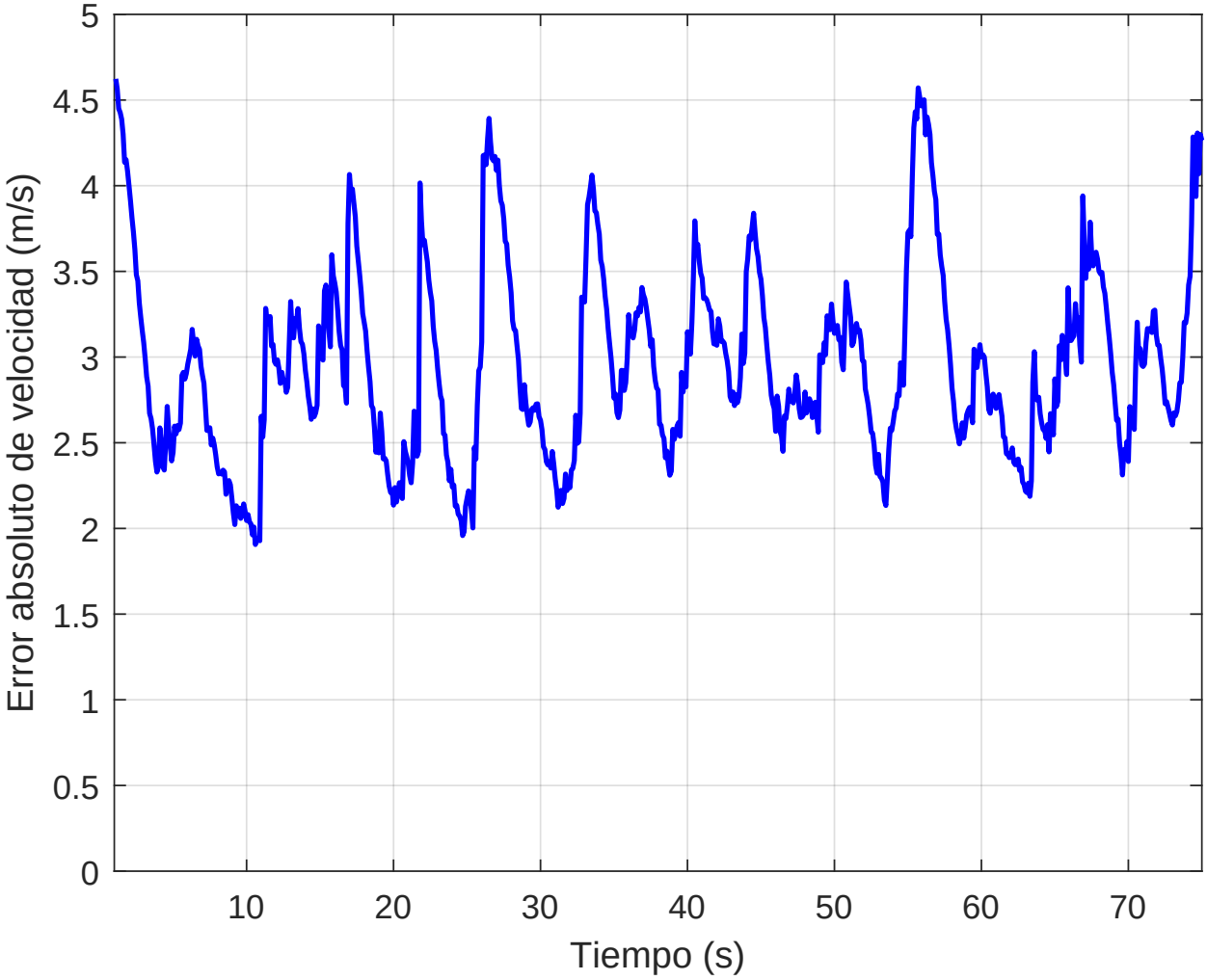


Figura 5.22: Error absoluto en la velocidad objetivo. Fuente: elaboración propia.

Se continuó con 24 mediciones más para validar más profundamente el comportamiento del agente, los errores para la desviación lateral y el error en el ángulo de conducción se muestran en el cuadro 5.13.

Tabla 5.13: Errores máximos y promedios de las pruebas 1 a 25.

Prueba	Desv. máx. [m]	Desv. prom. [m]	Áng. máx. [rad]	Áng. prom. [rad]
1	0.26	0.04	0.19	0.04
2	0.22	0.05	0.19	0.04
3	0.29	0.05	0.18	0.04
4	0.31	0.05	0.27	0.04
5	0.23	0.03	0.17	0.03
6	0.27	0.05	0.19	0.04
7	0.32	0.05	0.16	0.04
8	0.31	0.05	0.15	0.04
9	0.73	0.05	0.21	0.03
10	0.29	0.04	0.22	0.04
11	0.27	0.04	0.18	0.04
12	0.36	0.05	0.15	0.03
13	0.40	0.07	0.28	0.04
14	0.24	0.04	0.17	0.03
15	0.22	0.04	0.16	0.04
16	0.22	0.05	0.17	0.03
17	0.24	0.05	0.26	0.04
18	0.29	0.04	0.17	0.04
19	0.33	0.05	0.21	0.04
20	0.20	0.04	0.20	0.04
21	0.30	0.05	0.17	0.04
22	0.35	0.06	0.20	0.04
23	0.35	0.06	0.20	0.04
24	0.25	0.04	0.17	0.04
25	0.43	0.04	0.25	0.04

Como se puede apreciar se obtuvo buenos resultados de seguimiento para la trayectoria definida durante las pruebas, la desviación tuvo un valor máximo registrado de 0.73 m, al ser menor que 0.75 m, se cumple con la métrica #2. Por otro lado, el error máximo en el ángulo de conducción fue de 0.28 rad.

A partir de estas mediciones se calcula la cantidad de incidentes por millón de kilómetros (métrica #1). Ya que no se obtuvo ninguna invasión o choque durante las simulaciones, se obtiene un resultado de 0 incidentes por millón de kilómetros, por lo que se cumple con la métrica #1, la cual especifica una tasa de incidentes menor a 1.3 incidentes por millón de kilómetros. Con la clara salvedad de que se requiere más tiempo y kilómetros de simulación en escenarios más diversos para una validación más adecuada de esta métrica.

5.6 Análisis económico

En esta sección se presenta el análisis de los recursos invertidos en el desarrollo del presente proyecto. Dado que se trata de un proyecto de investigación sin fines comerciales, el análisis no se enfoca en el retorno económico, sino en el desglose de la inversión necesaria para su ejecución.

Inicialmente, se detallan los recursos materiales, herramientas y servicios utilizados durante el desarrollo del sistema de control basado en aprendizaje por refuerzo con garantías de seguridad. Estos se presentan en la Tabla 5.14. La computadora personal ya había sido comprada con anterioridad, así como la licencia de MATLAB por parte del ITCR, aún así, para el costo de la licencia se utiliza como referencia la licencia académica de MATLAB, la cual tiene un costo aproximado de 138 mil colones [68]. La remuneración se calcula como la mitad del salario mínimo (al ser de medio tiempo) para un ingeniero en España, país donde se realizó el proyecto, el cual es de 704479,61 colones (1200 euros) [69], para un período de 4 meses. Es importante destacar que este proyecto no es remunerado, por lo que realmente no representó un gasto para el CS2AC-UPC.

Tabla 5.14: Inversión estimada en el desarrollo del proyecto

Descripción	**Subtotal (¢)**	**Subtotal (€)**	**Observación**
Computadora personal	675 000	1 147,50	Ya disponible
Licencia MATLAB	138 722,73	235,83	Proporcionada por la institución
Remuneración	1 408 959,22	2 395,23	Cálculo proporcional a 50 % del salario de referencia
Total	**2 222 681,95**	**3 778,56**	

Este proyecto tiene una naturaleza de investigación con fines científicos, por lo tanto, en este momento no se cuenta con los insumos para realizar un retorno económico directo. Por otro lado, el proyecto produce una serie de retornos valiosos que fortalecen la investigación. Entre los principales recursos generados se destacan:

- **Controlador como base para futuras investigaciones:** Se desarrolló un sistema de control basado en aprendizaje por refuerzo que incorpora criterios de seguridad, representando un aporte relevante al estudio de la conducción autónoma en entornos dinámicos. El proyecto es pionero en el centro de investigación y refuerza la línea de investigación del grupo CS2AC-UPC en control inteligente con garantías de seguridad, y servirá como base para futuras publicaciones científicas en el centro de investigación.

- **Plataforma para la validación de sistemas de conducción autónoma:** Se diseñó e implementó un entorno de simulación en MATLAB con múltiples escenarios que permiten probar y validar distintas configuraciones de control. Este entorno es re-

utilizable para futuras investigaciones, reduciendo tiempos de desarrollo y facilitando la evaluación comparativa de estrategias.

- **Documentación técnica exhaustiva:** Se generó un conjunto completo de documentos técnicos que incluye el diseño del entorno de simulación, el código de los sistemas más importantes, las configuraciones de entrenamiento y los análisis de desempeño. Los criterios que se utilizaron para la validación del sistema sirven como base para la elaboración de investigaciones posteriores en sistemas de conducción autónomos.

Como se observa, a pesar de que la inversión para el desarrollo de este proyecto es considerable, cercana a 2.2 millones de colones, realmente los gastos en inversión para el proyecto para el centro de investigación fueron muy bajos. Por otro lado, aunque no se puede calcular el retorno a la inversión de los recursos generados de manera monetaria, los resultados obtenidos aportan valor significativo al CS2AC y a las líneas de investigación que promueve.

Capítulo 6

Conclusiones y recomendaciones

El desarrollo del presente proyecto permitió comprobar la viabilidad de aplicar técnicas de aprendizaje por refuerzo profundo (DRL) para el control seguro de vehículos autónomos en entornos simulados tridimensionales. Se logró integrar desde etapas tempranas del diseño restricciones de seguridad solicitadas, las cuales fueron implementadas y validadas tanto de forma visual como mediante métricas cuantitativas.

En primer lugar, la implementación del modelo vehicular permitió simular con precisión los desplazamientos del agente, registrando errores mínimos en velocidad y aceleración durante su validación mediante la plataforma Webots, lo que confirma la validez del modelo dinámico vehicular para simular el comportamiento vehicular de forma adecuada para el entrenamiento del agente.

Durante el entrenamiento del controlador, el agente aprendió a conducir manteniéndose dentro del carril, evitar colisiones y controlar su velocidad adecuadamente. Debido a los resultados observados, en donde se aprecia un buen funcionamiento en la toma de decisiones del agente, se decide proseguir con el desarrollo del sistema de percepción.

El sistema de percepción, basado en visión por computadora, fue capaz de detectar carriles y estimar propiedades como la curvatura, el ángulo de conducción y la desviación lateral del vehículo, estas mediciones fueron validados con un análisis estadístico para comprobar su confiabilidad. Este sistema fue esencial para alimentar al controlador con información confiable del entorno simulado.

No se registraron incidentes en las 25 pruebas realizadas durante la validación final del agente de aprendizaje por refuerzo. La desviación lateral del vehículo se mantuvo dentro de los límites aceptados, así como la cantidad de incidentes por kilómetro, lo que confirma que el enfoque propuesto es funcional como prueba de concepto.

En conjunto, el proyecto cumplió con los objetivos planteados y demuestra que el uso de aprendizaje por refuerzo profundo con restricciones de seguridad explícitas es una estrategia viable para el control autónomo de vehículos. Esto sienta una base sólida para futuras investigaciones que busquen extender la robustez del sistema a entornos más complejos o incluso su eventual validación en hardware físico.

Finalmente, a modo de recomendación para futuras investigaciones, debe destacarse que la implementación completa del entorno tridimensional de simulación, junto con el desarrollo de un sistema de percepción basado en visión por computadora y el entrenamiento de agentes inteligentes con restricciones de seguridad, representa una carga computacional y metodológica considerable. El algoritmo de detección de carril es notablemente lento de ejecutar en tiempo real, esto debido a ser basado en métodos de visión por computadora tradicionales, los cuales son inferiores en tiempo de procesamiento en comparación a métodos basados en inteligencia artificial. Esto limita la cantidad de escenarios y repeticiones que pueden demostrarse dentro del periodo de tiempo asignado para el proyecto y bajo la infraestructura disponible, por lo que es un importante área de mejora. Por otro lado, es importante considerar más escenarios que incluyan peatones, ciclistas, señales de tránsito, entre muchos otros elementos, que mejoren la capacidad de generalización del agente a la hora de conducción.

Bibliografía

[1] U.S. Department of Transportation, National Highway Traffic Safety Administration, Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey, DOT HS 812 506, Washington, DC, USA, Mar. 2018. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812506 [Accessed: Mar. 13, 2025].

[2] National Highway Traffic Safety Administration (NHTSA), The Economic and Societal Impact of Motor Vehicle Crashes, 2019 (Revised), Report No. DOT HS 813 403, Feb. 2022. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813403

[3] H. Suk, Y. Lee, T. Kim, and S. Kim, "Addressing uncertainty challenges for autonomous driving in real-world environments," Advances in Computers, vol. 123, pp. 317–361, 2023. doi: 10.1016/bs.adcom.2023.06.004

[4] Y. Chen, C. Ji, Y. Cai, T. Yan, y B. Su, "Deep Reinforcement Learning in Autonomous Car Path Planning and Control: A Survey," arXiv preprint arXiv:2404.00340, 2024. [En línea]. Disponible en: https://arxiv.org/abs/2404.00340

[5] N. J. Zakaria, M. I. Shapiai, R. A. Ghani, M. N. M. Yassin, M. Z. Ibrahim and N. Wahid, "Lane Detection in Autonomous Vehicles: A Systematic Review," in IEEE Access, vol. 11, pp. 3729-3765, 2023, doi: 10.1109/ACCESS.2023.3234442.

[6] R. Zhang, Y. Wu, W. Gou, and J. Chen, "RS-Lane: A robust lane detection method based on ResNeSt and self-attention distillation for challenging traffic situations," *Journal of Advanced Transportation*, vol. 2021, pp. 1–12, Aug. 2021.

[7] F. Munir, S. Azam, M. Jeon, B.-G. Lee, and W. Pedrycz, "LDNet: End-to-end lane marking detection approach using a dynamic vision sensor," *arXiv preprint arXiv:2009.08020*, Sep. 2020.

[8] S. Ghanem, P. Kanungo, G. Panda, S. C. Satapathy, and R. Sharma, "Lane detection under artificial colored light in tunnels and on highways: An IoT-based framework for smart city infrastructure," *Complex & Intelligent Systems*, pp. 1–12, May 2021, doi: 10.1007/s40747-021-00381-2.

[9] J. Li, F. Jiang, J. Yang, B. Kong, M. Gogate, K. Dashtipour, and A. Hussain, "Lane-DeepLab: Lane semantic segmentation in automatic driving scenarios for high-definition maps," *Neurocomputing*, vol. 465, pp. 15–25, Nov. 2021, doi: 10.1016/j.neucom.2021.08.105.

[10] L. Liang, H. Ma, L. Zhao, X. Xie, C. Hua, M. Zhang, and Y. Zhang, "Vehicle detection algorithms for autonomous driving: A review," *Sensors*, vol. 24, no. 10, p. 3088, 2024, doi: 10.3390/s24103088.

[11] A. Russell and J. J. Zou, "Vehicle detection based on color analysis," in *Proc. 2012 Int. Symp. on Communications and Information Technologies (ISCIT)*, Gold Coast, Australia, Oct. 2–5, 2012, pp. 620–625.

[12] S. S. Teoh and T. Bräunl, "Symmetry-based monocular vehicle detection system," *Machine Vision and Applications*, vol. 23, pp. 831–842, 2012, doi: 10.1007/s00138-011-0355-7.

[13] W.-K. Tsai, S.-L. Wu, L.-J. Lin, T.-M. Chen, and M.-H. Li, "Edge-based forward vehicle detection method for complex scenes," in *Proc. 2014 IEEE Int. Conf. on Consumer Electronics–Taiwan (ICCE-TW)*, Taipei, Taiwan, May 26–28, 2014, pp. 173–174.

[14] C. Creusot and A. Munawar, "Real-time small obstacle detection on highways using compressive RBM road reconstruction," in *Proc. 2015 IEEE Intelligent Vehicles Symposium (IV)*, Seoul, Republic of Korea, Jun. 28–Jul. 1, 2015, pp. 162–167.

[15] X. Chen, H. Chen, and H. Xu, "Vehicle detection based on multifeature extraction and recognition adopting RBF neural network on ADAS system," *Complexity*, vol. 2020, Article ID 8842297, 2020, doi: 10.1155/2020/8842297.

[16] N. Kosaka and G. Ohashi, "Vision-based nighttime vehicle detection using CenSurE and SVM," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2599–2608, Oct. 2015, doi: 10.1109/TITS.2015.2413971.

[17] G. Welch and G. Bishop, *An Introduction to the Kalman Filter*, Univ. of North Carolina, Chapel Hill, NC, USA, 1995.

[18] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, Santiago, Chile, Dec. 7–13, 2015, pp. 4696–4704.

[19] W. Roberts, P. Stoica, J. Li, T. Yardibi, and F. A. Sadjadi, "Iterative adaptive approaches to MIMO radar imaging," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, pp. 5–20, 2010, doi: 10.1109/JSTSP.2009.2038964.

[20] S. Pang, Y. Zeng, Q. Yang, B. Deng, H. Wang, and Y. Qin, "Improvement in SNR by adaptive range gates for RCS measurements in the THz region," *Electronics*, vol. 8, no. 7, p. 805, 2019, doi: 10.3390/electronics8070805.

[21] J. Karangwa, J. Liu, and Z. Zeng, "Vehicle detection for autonomous driving: A review of algorithms and datasets," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, pp. 11568–11594, 2023, doi: 10.1109/TITS.2023.3292278.

[22] M. Wen, S. Cho, J. Chae, Y. Sung, and K. Cho, "Range image-based density-based spatial clustering of application with noise clustering method of three-dimensional point clouds," *International Journal of Advanced Robotic Systems*, vol. 15, pp. 1735–1754, 2018, doi: 10.1177/1729881418762302.

[23] S.-M. Lee, J. J. Im, B.-H. Lee, A. Leonessa, and A. Kurdila, "A real-time grid map generation and object classification for ground-based 3D LIDAR data using image analysis techniques," in *Proc. 2010 IEEE Int. Conf. on Image Processing (ICIP)*, Hong Kong, China, Sep. 26–29, 2010, pp. 2253–2256.

[24] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3D LIDAR point clouds," in *Proc. 2011 IEEE Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, May 9–13, 2011, pp. 2798–2805.

[25] C. Reymann and S. Lacroix, "Improving LiDAR point cloud classification using intensities and multiple echoes," in *Proc. 2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, Sep. 28–Oct. 2, 2015, pp. 5122–5128.

[26] I. Bogoslavskyi and C. Stachniss, "Efficient online segmentation for sparse 3D laser scans," *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 85, pp. 41–52, 2017, doi: 10.1007/s41064-016-0003-y.

[27] J. Byun, K.-I. Na, B.-S. Seo, and M. Roh, "Drivable road detection with 3D point clouds based on the MRF for intelligent vehicle," in *Field and Service Robotics: Results of the 9th International Conference*, vol. 105, Springer, Berlin/Heidelberg, Germany, 2015, pp. 49–60.

[28] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, 1981, doi: 10.1145/358669.358692.

[29] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes, "3D LIDAR-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes," *Robotics and Autonomous Systems*, vol. 83, pp. 299–311, 2016, doi: 10.1016/j.robot.2016.06.007.

[30] X. Wang, L. Zou, X. Shen, Y. Ren, and Y. Qin, "A region-growing approach for automatic outcrop fracture extraction from a three-dimensional point cloud," *Computers & Geosciences*, vol. 99, pp. 100–106, 2017, doi: 10.1016/j.cageo.2016.11.002.

[31] S. Sun, C. Li, P. W. Chee, A. H. Paterson, Y. Jiang, R. Xu, J. S. Robertson, J. Adhikari, and T. Shehzad, "Three-dimensional photogrammetric mapping of cotton bolls in situ

based on point cloud segmentation and clustering," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 160, pp. 195–207, 2020, doi: 10.1016/j.isprsjprs.2019.12.011.

[32] B. Zhao, X. Hua, K. Yu, W. Xuan, X. Chen, and W. Tao, "Indoor point cloud segmentation using iterative Gaussian mapping and improved model fitting," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, pp. 7890–7907, 2020, doi: 10.1109/TGRS.2020.2984943.

[33] R. He, H. Lv, S. Zhang, D. Zhang, y H. Zhang, "Lane following method based on improved DDPG algorithm," Sensors, vol. 21, no. 14, 2021.

[34] C. Wang et al., "A safe reinforcement learning based trajectory tracker framework," IEEE Transactions on Intelligent Transportation Systems, vol. 24, no. 6, pp. 5765–5780, 2023.

[35] Q. T. D. Tran y S.-H. Bae, "Proximal Policy Optimization Through a Deep Reinforcement Learning Framework for Multiple Autonomous Vehicles at a Non-Signalized Intersection," Applied Sciences, vol. 10, no. 16, Art. no. 5722, 2020. [En línea]. Disponible en: https://www.mdpi.com/2076-3417/10/16/5722.

[36] G. Wang, H. Niu, D. Zhu, J. Hu, X. Zhan, y G. Zhou, "ModEL: A Modularized End-to-end Reinforcement Learning Framework for Autonomous Driving," arXiv preprint arXiv:2110.11573, 2022. [En línea]. Disponible en: https://arxiv.org/abs/2110.11573.

[37] D. West, "The evolving safety and policy challenges of self-driving cars," Brookings, 2024. [Online]. Available: https://www.brookings.edu/articles/the-evolving-safety-and-policy-challenges-of-self-driving-cars

[38] M. Davies, "Cruise admits fault in crash where robotaxi dragged woman," Wired, 2024. [Online]. Available: https://www.wired.com/story/gm-cruise-robotaxi-strategy-admitting-software-fault-in-gruesome-crash

[39] H. C. Joshi and S. Kumar, "Artificial Intelligence Failures in Autonomous Vehicles: Causes, Implications, and Prevention" in Computer, vol. 57, no. 11, pp. 18-30, Nov. 2024, doi: 10.1109/MC.2024.3449435.

[40] P. Koopman, "Lessons from the Cruise Robotaxi Pedestrian Dragging Mishap," arXiv preprint, arXiv:2406.05281, 2024. [Online]. Available: https://arxiv.org/abs/2406.05281

[41] Press Publications, "Examining the safety of self-driving technology," 2025. [Online]. Available: https://presspublications.com/premium/stacker/stories/examining-the-safety-of-self-driving-technology,58535

[42] S. Jha, S. Cui, S. S. Banerjee, T. Tsai, Z. Kalbarczyk, and R. K. Iyer, "ML-driven malware that targets AV safety," *arXiv preprint arXiv:2004.13004*, 2020. [Online]. Available: https://arxiv.org/abs/2004.13004

[43] T. Lerman, "Tesla has avoided trials with money. This case made it to court," The Washington Post, Jul. 2025. [Online]. Available: https://www.washingtonpost.com/technology/2025/07/18/tesla-autopilot-fatal-crash-wrongful-death

[44] Johnston Law Firm, "Human error in self-driving car accidents," 2023. [Online]. Available: https://johnston-lawfirm.com/human-error-in-self-driving-car-accidents

[45] Q. Zhang, C. D. Wallbridge, D. M. Jones, and P. L. Morgan, "Public perception of autonomous vehicle capability determines judgment of blame and trust in road traffic accidents," *Transportation Research Part A: Policy and Practice*, vol. 179, p. 103887, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965856423003075

[46] R. Prieve, "A Zoox and an e-bike collided in S.F.," SF Chronicle, Jul. 2025. [Online]. Available: https://www.sfchronicle.com/sf/article/zoox-ebike-crash-robotaxi-safety-20313079.php

[47] Scopio Labs, "Trade-off between field of view and resolution," *Scopio Labs*, Accessed: Jul. 6, 2025. [Online]. Available: https://scopiolabs.com/full-field-imaging/trade-off-between-field-of-view-and-resolution/

[48] The MathWorks, Inc., "MATLAB and Simulink Release R2024b," Natick, Massachusetts, United States, 2024.

[49] Blickfeld GmbH, "Understanding LiDAR specifications," Blickfeld Blog, Apr. 27, 2021. [Online]. Available: https://www.blickfeld.com/blog/understanding-lidar-specifications/ [Accessed: Jul. 13, 2025].

[50] Criticality Metrics Documentation, "Time To Collision (TTC)," *Read the Docs*, 2025. [En línea]. Disponible en: https://criticality-metrics.readthedocs.io/en/latest/time-scale/TTC.html. [Accedido: 9-jul-2025].

[51] G. Vilches, "Cómo eliminar atípicos en señales: filtro de Hampel," *LinkedIn*, 2023. [En línea]. Disponible en: https://www.linkedin.com/pulse/c%C3%B3mo-eliminar-at%C3%ADpicos-en-se%C3%B1ales-filtro-de-hampel-gast%C3%B3n-vilches/

[52] P. Polack, F. Altché, B. Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Redondo Beach, CA, USA, Jun. 2017, pp. 812–818, doi: 10.1109/IVS.2017.7995816.

[53] MathWorks, "Highway Lane Following," MATLAB & Simulink Documentation, 2024. [En línea]. Disponible en: https://la.mathworks.com/help/driving/ug/highway-lane-following.html [Accessed: Jul. 13, 2025].

[54] Cyberbotics, "Webots Physics Reference." [En línea]. Disponible en: https://cyberbotics.com/doc/reference/physics. [Accedido: 17-jul-2025].

[55] K. Ulrich and S. Eppinger, Product design and developement, 6th ed. New York: McGrawHill Education, 2016

[56] K. D. Kusano, J. M. Scanlon, Y. H. Chen, T. L. McMurry, R. Chen, T. Gode, and T. Victor, "Comparison of Waymo rider-only crash data to human benchmarks at 7.1 million miles," *Traffic Injury Prevention*, vol. 25, no. sup1, pp. S66–S77, 2024, doi: 10.1080/15389588.2024.2380786.

[57] K. Rehrl and S. Gröchenig, "Evaluating Localization Accuracy of Automated Driving Systems," *Sensors*, vol. 21, no. 17, art. 5855, Aug. 2021, doi: 10.3390/s21175855.

[58] T. G. R. Reid, M. A. Wills, J. N. Gross, and B. M. Sadler, "Localization Requirements for Autonomous Vehicles," *SAE Int. J. Connected and Automated Vehicles*, vol. 2, no. 3, Sep. 2019, doi: 10.4271/12-02-03-0012.

[59] Z. Lai, C. Liu, S. Sheng, and Z. Zhang, "KAN-RCBEVDepth: A multi-modal fusion algorithm in object detection for autonomous driving," *arXiv preprint* arXiv:2408.02088, 2024. [Online]. Available: https://arxiv.org/abs/2408.02088

[60] J. R. Ziehn, M. Ruf, M. Roschani, and J. Beyerer, "Accuracy Evaluation of a Lightweight Analytic Vehicle Dynamics Model for Maneuver Planning," in *Proc. 2020 5th Int. Conf. on Robotics and Automation Engineering (ICRAE)*, pp. 197–205, Nov. 2020, doi: 10.1109/ICRAE50850.2020.9310898.

[61] MathWorks, "Lane Detection Optimized with GPU Coder," MathWorks, 2024. [Online]. Available: https://es.mathworks.com/help/gpucoder/ug/lane-detection-optimized-with-gpu-coder.html

[62] MathWorks Deep Learning Toolbox Team, "Pretrained Spatial CNN for Lane Detection," GitHub, 2021. [Online]. Available: https://github.com/matlab-deep-learning/pretrained-spatial-CNN

[63] MathWorks, "voxelRCNNObjectDetector," MathWorks, 2024. [Online]. Available: https://la.mathworks.com/help/lidar/ref/voxelrcnnobjectdetector.html

[64] MathWorks, "vehicleBody3DOF," MathWorks, 2024. [Online]. Available: https://es.mathworks.com/help/vdynblks/ref/vehiclebody3dof.html

[65] M. I. Sabbir, "Comparative analysis of double deep Q-network (Double DQN) and Proximal Policy Optimization (PPO) for lane-keeping in autonomous driving," *International Journal of Science and Research Archive*, vol. 13, no. 2, pp. 1207–1222, 2024, doi: 10.30574/ijsra.2024.13.2.2237.

[66] A. I. Razak, "Implementing a Deep Reinforcement Learning model in ..." BSc thesis, supervised by D.-C. Kim, Budapest University of Technology and Economics, Dec. 2022. [Online]. Available: https://faculty.utrgv.edu/dongchul.kim/6379/t14.pdf [Accessed: Jul. 13, 2025].

[67] W. Mendenhall, R. J. Beaver y B. M. Beaver, *Introduction to Probability and Statistics*, 15ª ed. Boston, MA: Cengage Learning, 2019.

[68] MathWorks, "Pricing and Licensing – MATLAB for Education," 2025. [En línea]. Disponible en: https://la.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=edu. [Consulta: julio 2025].

[69] SIPEM - Sociedad de Ingenieros Profesionales de España, "¿Cuánto cobra un ingeniero recién graduado en España?", 2023. [En línea]. Disponible en: https://sipem.upct.es/noticia/cuanto-cobra-un-ingeniero-recien-graduado-en-espana. [Consulta: julio 2025].

Apéndice A

Modelo dinámico vehicular

A.1 Parámetros del modelo

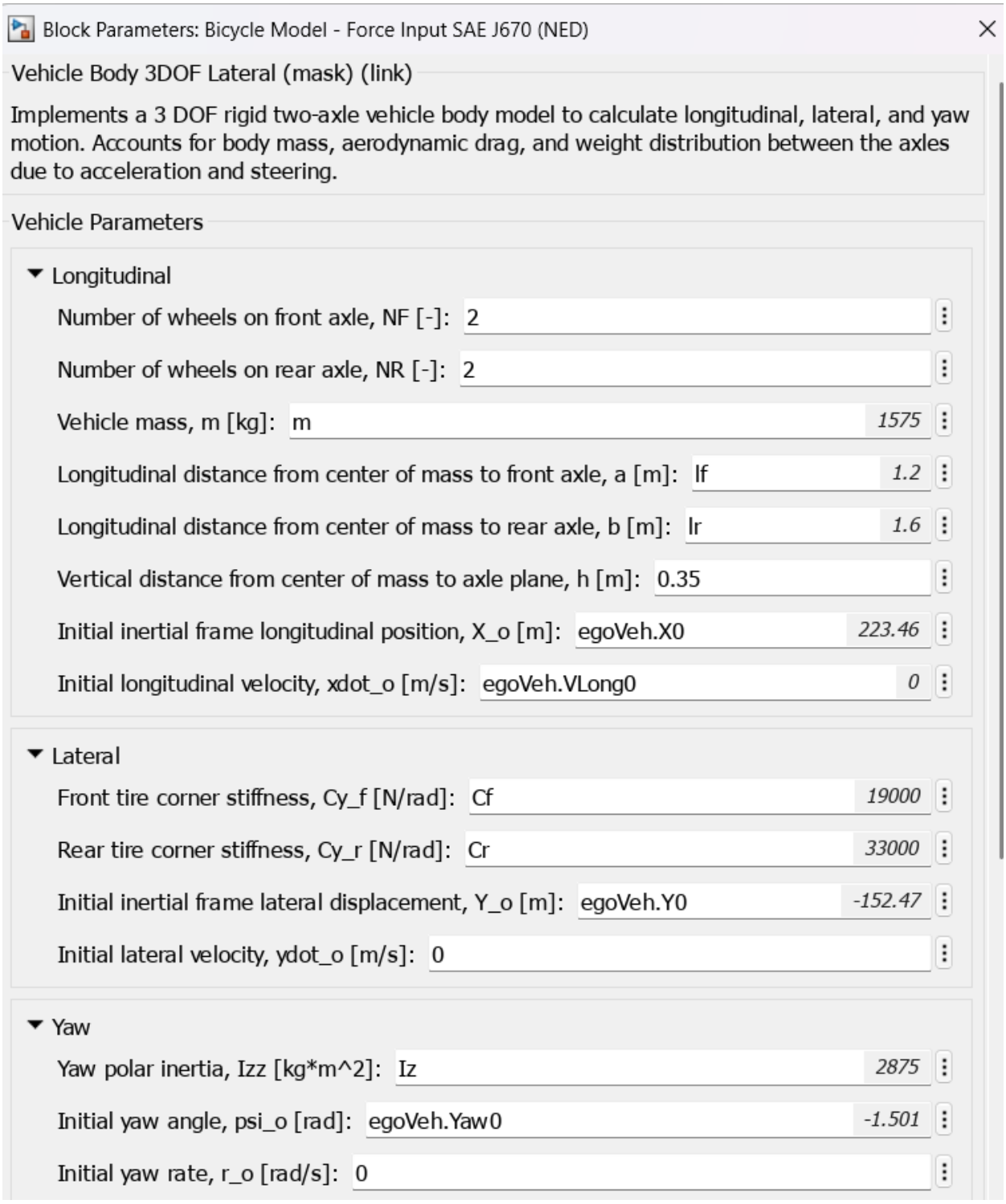


Figura A.1: Parte 1 de los parámetros del modelo dinámico vehicular.

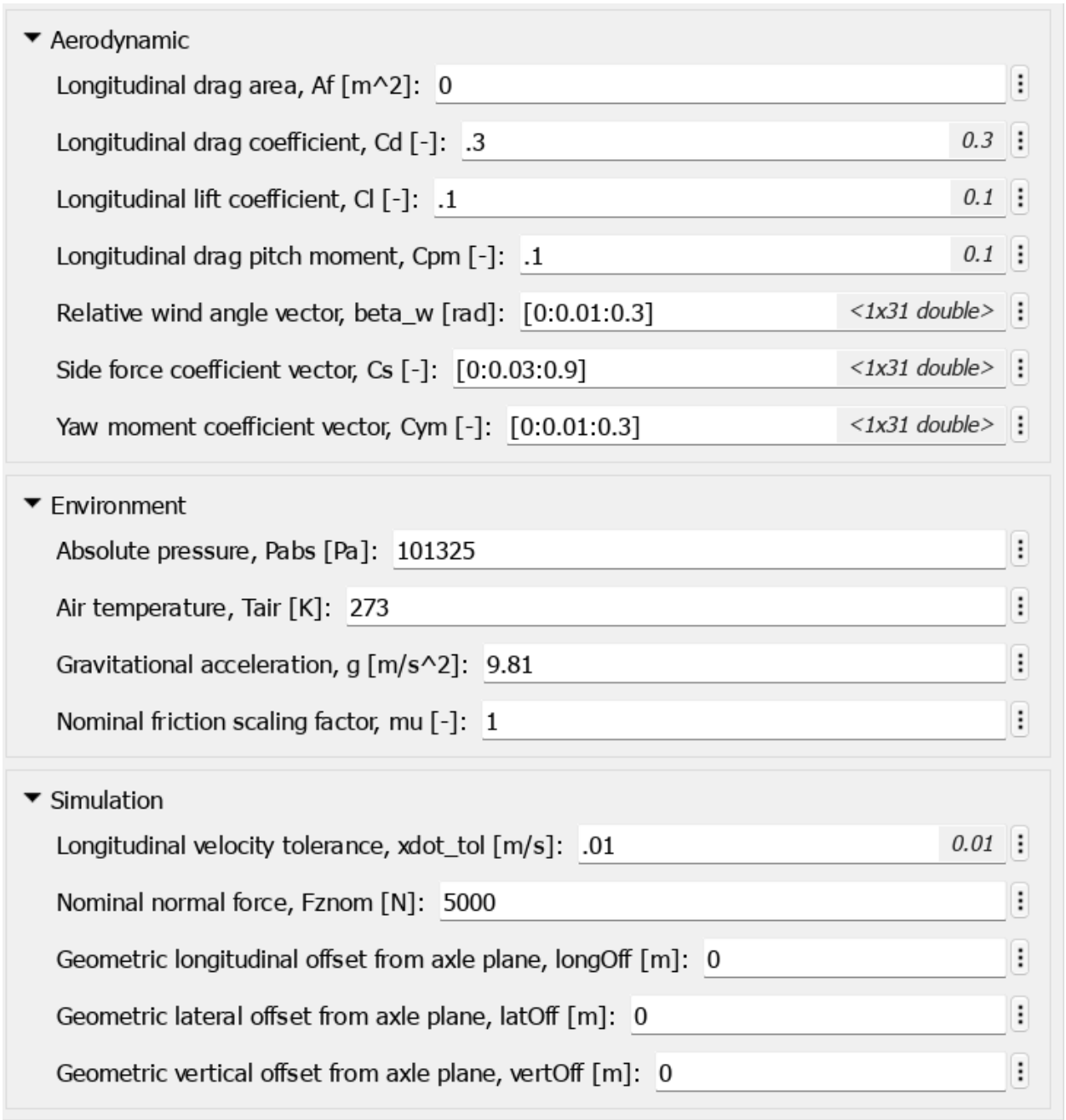


Figura A.2: Parte 2 de los parámetros del modelo dinámico vehicular.

Apéndice B

Sistemas de MATLAB/Simulink

B.1 Validación del modelo dinámico vehicular

B.1.1 Código para la inicialización de variables en MATLAB

```
%% Par metros de entrenamiento
T = 20;                 % Duraci n total del episodio en segundos
Ts = 0.1;               % Tiempo de muestreo (intervalo entre pasos de
    simulaci n)
egoActorID = double(1); % ID del veh culo ego (convertido a tipo
   double)

%% Posici n de inicio

egoVeh.X0 = 223.4;  % Posici n inicial el X (m)
egoVeh.Y0 = -152.471;         % Posici n inicial en Y (m)
egoVeh.Yaw0 = -87 * pi / 180; % Orientaci n inicial en radianes (-87
    grados)

egoVeh.VX0 = 0;         % Velocidad inicial en X (m/s)
egoVeh.VY0 = 0;         % Velocidad inicial en Y (m/s)
egoVeh.VLong0 = 0;      % Velocidad longitudinal inicial (m/s)

%% Par metros del modelo din mico
egoVeh.Length = 4.7;                          % Longitud total del
   veh culo (m)
egoVeh.RearOverhang = 1;                      % Voladizo trasero (m)
egoVeh.CGToFrontAxle = egoVeh.Length/2 - 1;  % Distancia desde el
   centro de masa al eje delantero (m)
egoVeh.CGToRearAxle  = egoVeh.Length/2 - 1;  % Distancia desde el
   centro de masa al eje trasero (m)

```

```
m = 1575;      % Masa total del veh culo (kg)
Iz = 2875;     % Momento de inercia alrededor del eje de gui ada (
    m Ns )
lf = 1.2;      % Distancia desde el centro de masa hasta el eje
    delantero (m)
lr = 1.6;      % Distancia desde el centro de masa hasta el eje
    trasero (m)
Cf = 19000;    % Rigidez lateral de los neum ticos delanteros (N/rad)
Cr = 33000;    % Rigidez lateral de los neum ticos traseros (N/rad)

%% Cargar bus
load('busPose.mat')    % Cargar estructura de datos para el veh culo
    (bus)

%% Algunas constantes

% Constantes de aceleraci n y frenado
tau = 0.5;
tau2 = 0.07;

% L mites de acci n para el agente (aceleraci n lateral y
    longitudinal)
u_min = -1;     % L mite inferior para acci n de direcci n (giros)
u_max = 1;      % L mite superior para acci n de direcci n
ax_min = -1;    % L mite inferior para aceleraci n longitudinal
ax_max = 1;     % L mite superior para aceleraci n longitudinal
```

Listing B.1: helperVDM.m

B.1.2 Diagramas de Simulink

B.1.3 Códigos de Simulink

```
function R = calcularRadio(x, y)
%#codegen
% Radio de la circunferencia circunscrita a partir de un flujo de
    posiciones.
% El bloque recuerda las dos mediciones previas (1 s y 2 s antes).
% Devuelve NaN hasta que hay tres puntos distintos y no colineales.

% Variables de estado internas
persistent xs ys k          % buffers circulares y contador
if isempty(xs)
    xs = zeros(1,3);        % preasignaci n para c digo generado
    ys = zeros(1,3);
```

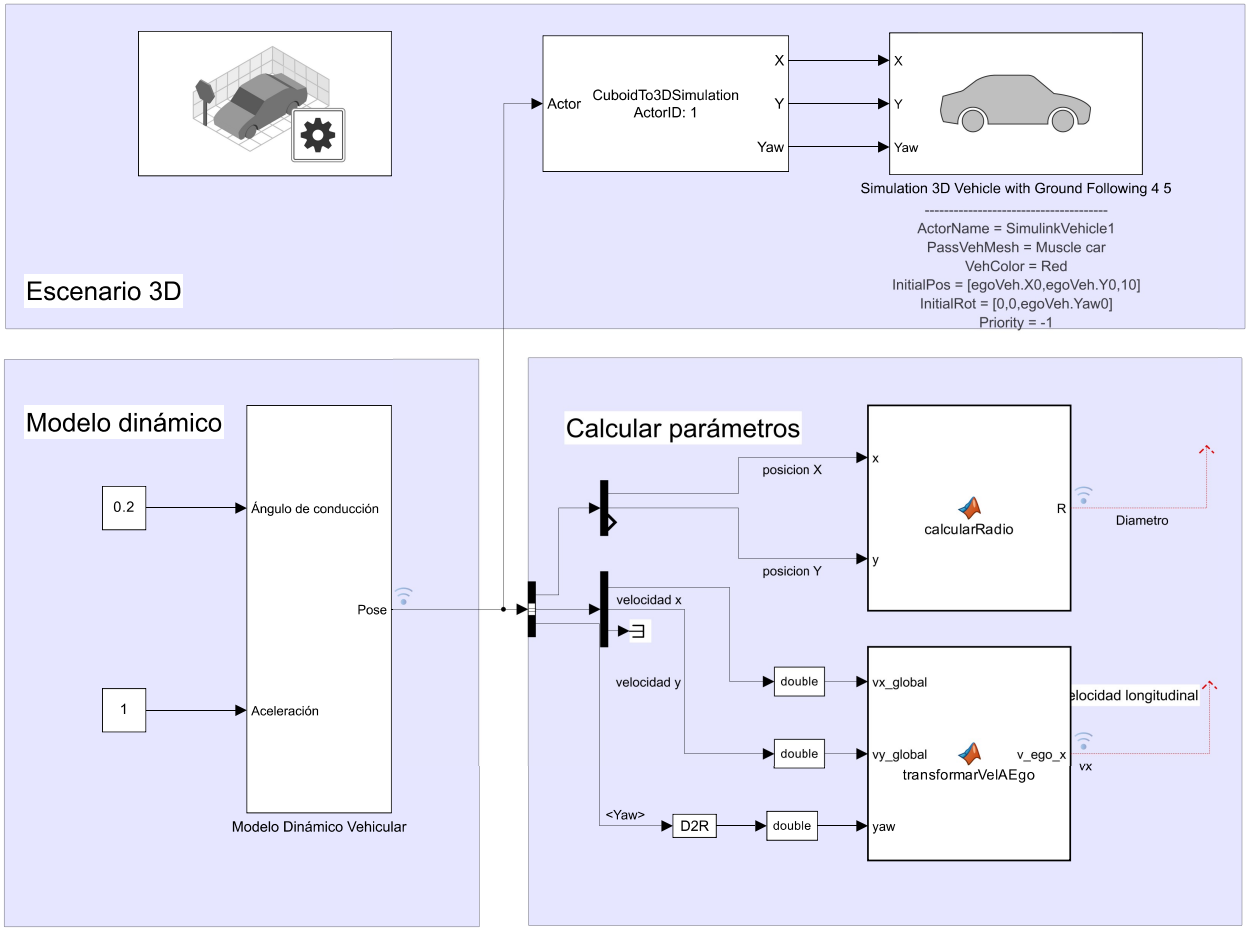


Figura B.1: Diagrama para las pruebas del modelo dinámico vehicular.

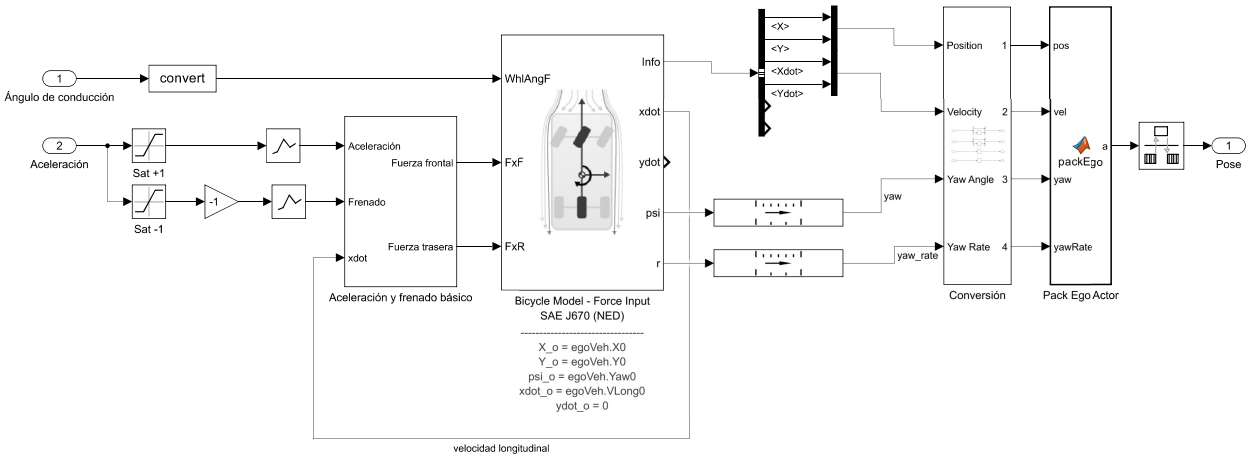


Figura B.2: Diagrama del modelo dinámico vehicular implementado en el proyecto.

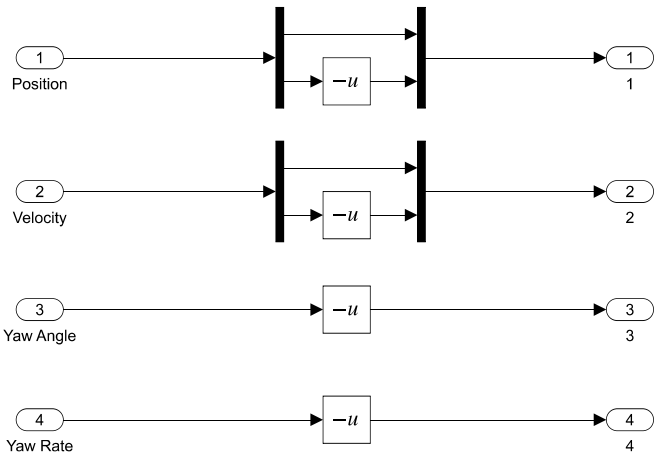


Figura B.3: Diagrama de la conversión de signos.

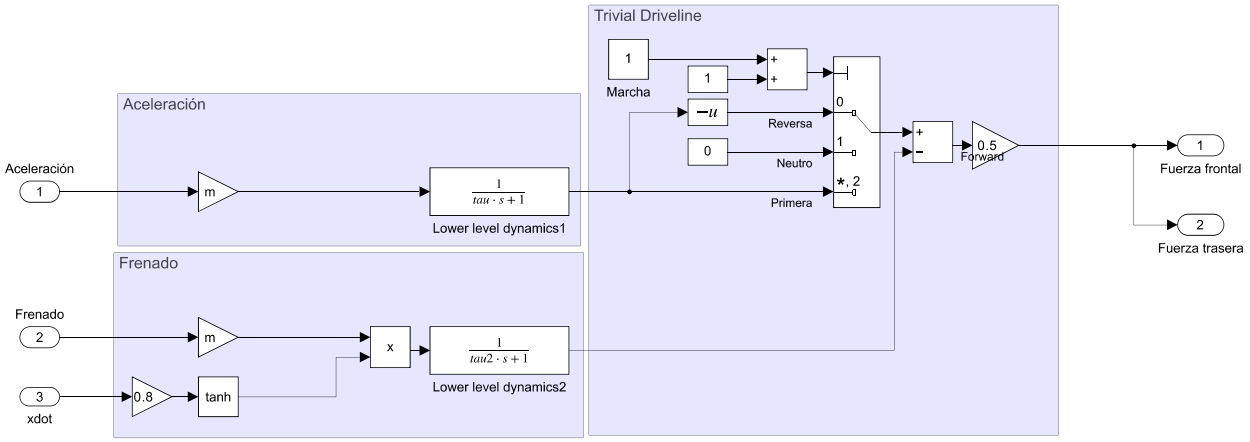


Figura B.4: Diagrama del acelerado y frenado.

```
    k  = uint32(0);
end

% Actualizar  ndice  circular y almacenar nueva muestra
k  = k + 1;
idx = mod(double(k-1),3) + 1;    % 1-2-3-1-2-3...
xs(idx) = x;
ys(idx) = y;

% A n no hay suficientes datos
if k < 3
    R = NaN;
    return
end

% Recuperar las tres posiciones m s recientes en orden temporal
id = mod(double(k-1):-1:double(k-3),3) + 1;    %  ltimo , pen ltimo ,
    antepen ltimo
x1 = xs(id(3));  y1 = ys(id(3));
x2 = xs(id(2));  y2 = ys(id(2));
x3 = xs(id(1));  y3 = ys(id(1));

% Longitudes de los lados
a = hypot(x2 - x3, y2 - y3);
b = hypot(x1 - x3, y1 - y3);
c = hypot(x1 - x2, y1 - y2);

% Semiper metro y  rea  (Her n)
s = 0.5 * (a + b + c);
area2 = s * (s - a) * (s - b) * (s - c);
if area2 <= 0           % tri ngulo degenerado
    R = NaN;
    return
end
A = sqrt(area2);

% Radio circunscrito
R = 2*a * b * c / (4 * A);
end
```

Listing B.2: calcularRadio.m

```
function v_ego_x  = transformarVelAEgo(vx_global, vy_global, yaw)
% Transforma velocidades desde el marco global al marco del ego

% Matriz de rotaci n basada en yaw

```

```
v_ego_x =  sin(yaw) * vy_global + cos(yaw) * vx_global;
end
```

Listing B.3: transformarVelAEgo.m

```
function a = packEgo(pos,vel,yaw,yawRate, egoActorID)
% Pack ego information into a single ego actor bus
%
% Important note:
% Output is a bus of type BusVehiclePose. This bus must match the bus
% elements expected by Scenario Reader. You can view the
% bus association using 'Edit Data' in the menu above.

a = struct(...
    'ActorID', egoActorID,...
    'Position', [pos(1) pos(2) 0], ...
    'Velocity', [vel(1) vel(2) 0], ...
    'Roll', 0, ...
    'Pitch',0, ...
    'Yaw', yaw, ...
    'AngularVelocity',[0 0 yawRate]);
```

Listing B.4: packEgo.m

B.2 Validación del sistema de percepción: detección de carril

B.2.1 Código para la inicialización de variables en MATLAB

```

%% Algunas constantes

% Constantes de aceleraci n y frenado
tau = 0.5;
tau2 = 0.07;

% L mites de acci n para el agente (aceleraci n lateral y
   longitudinal)
u_min = -1;     % L mite inferior para acci n de direcci n (giros)
u_max = 1;      % L mite superior para acci n de direcci n
ax_min = -1;    % L mite inferior para aceleraci n longitudinal
ax_max = 1;     % L mite superior para aceleraci n longitudinal

T = 20;
Ts = 0.1;
```

```
16
17 egoActorID=double(1);
18
19 %% Posici n de inicio
20
21 egoVeh.X0 = 0;  % Posici n inicial el X (m)
22 egoVeh.Y0 = 0;        % Posici n inicial en Y (m)
23 egoVeh.Yaw0 = 0; % Orientaci n inicial en radianes (-87 grados)
24
25 egoVeh.VX0 = 0;         % Velocidad inicial en X (m/s)
26 egoVeh.VY0 = 0;         % Velocidad inicial en Y (m/s)
27 egoVeh.VLong0 = 0;      % Velocidad longitudinal inicial (m/s)
28
29 %% Configuraci n de las c maras
30
31 % Par metros del sensor de c mara
32 NumColumns=1920;
33 NumRows=1080;
34 FOV=74;                          % Campo de visi n
35 pitchCam=30;                     % Inclinaci n hacia el suelo (grados)
36
37
38
39 % Configuraci n com n a todas las c maras
40 camera.NumColumns = NumColumns;
41 camera.NumRows = NumRows;
42 camera.FieldOfView = [FOV,FOV];
43 camera.ImageSize = [camera.NumRows, camera.NumColumns];
44 camera.PrincipalPoint = [camera.NumColumns,camera.NumRows]/2;
45 camera.FocalLength = [camera.NumColumns / (2*tand(FOV)), camera.
      NumColumns / (2*tand(FOV))];
46
47 camera2 = camera;
48 camera3 = camera;
49 camera4 = camera;
50
51 camera.Position = [2.5, 0, 0.7];              % Frontal
52 camera2.Position = [-2.5, 0, 0.894];          % Trasera
53 camera3.Position = [0, 1.88/2, 0.7];          % Lateral derecha
54 camera4.Position = [0, -1.88/2, 0.7];         % Lateral izquierda
55
56 camera.Rotation = [0, pitchCam, 0];
57 camera2.Rotation = [0, pitchCam, 180];
58 camera3.Rotation = [0, pitchCam, 90];
59 camera4.Rotation = [0, pitchCam, 270];
60
```

```
%% Cargar buses
load('busPose.mat')
load('busLane.mat')
load('busVision.mat')
```

Listing B.5: helperCV.m

B.2.2 Funciones para la detección de carril

```
function [deviacionLateral,errorAngulo,curvatura]= procesarFrame(
    imgEntrada1, imgEntrada2,imgEntrada3,imgEntrada4,sensor1,sensor2,
    sensor3,sensor4)

    % Obtener vista de p jaro desde las c maras frontal y trasera
    vistaPajaro1 = obtenerVistaPajaroFrenteAtras(imgEntrada1,sensor1)
        ;
    vistaPajaro2 = obtenerVistaPajaroFrenteAtras(imgEntrada2,sensor2)
        ;

    % Obtener vista de p jaro desde las c maras laterales
    vistaPajaro3 = obtenerVistaPajaroLados(imgEntrada3,sensor3);
    vistaPajaro4 = obtenerVistaPajaroLados(imgEntrada4,sensor4);

    densidadPixeles = 50; % Resoluci n espacial en pixeles por metro
        (escala)

    % Unir vistas frontal y trasera
    [vistaPajaroA,centroX,centroY] = unirImagenes(vistaPajaro2,
        vistaPajaro1);

    % Unir vistas laterales
    [vistaPajaroB] = unirImagenes2(vistaPajaro4,vistaPajaro3);

    % Unir todas las vistas en una imagen final
    vistaPajaroFinal = unirImageneS3(vistaPajaroA,vistaPajaroB);

    % Preprocesar imagen final para binarizar (resaltar carriles)
    imgBinaria = preprocesarImagen(vistaPajaroFinal);

    % Detectar l neas de carril en la imagen binaria
    lineasDetectadas = detectarLineas(imgBinaria,vistaPajaroFinal);
        % o m todo con Hough

    % Etiquetar l neas detectadas seg n su continuidad y tipo
    lineasEtiquetadas = etiquetadoLineas(lineasDetectadas,
        densidadPixeles);
```

```
30
31      % Unir segmentos de l neas similares en carriles completos
32      lineasUnidas = unirSegmentos(lineasEtiquetadas);
33
34      % Obtener la l nea central del carril en base a la posici n del
            veh culo
35      centroCarril = unirCarrilesCentrales(lineasUnidas, centroX,
            centroY);
36
37      % Calcular desviaci n lateral,  ngulo  de error y curvatura
            respecto al centro del carril
38      [deviacionLateral, errorAngulo, curvatura] = distanciaDesdePunto(
            centroCarril, centroX, centroY, densidadPixeles);
39
40
41  end
```

Listing B.6: procesarFrame.m

```
1
2   function vistaPajaro = obtenerVistaPajaroFrenteAtras(imgEntrada,
        sensor)
3       %#codegen  %       Necesario para Simulink Code Generation
4
5       vistaSalida = [0 7.5 -7 7]; %  rea  en metros [frente atr s
            izquierda derecha] respecto a la c mara
6
7       densidadPixeles = 50; % N mero de pixeles por metro (escala de
            salida)
8
9       tamanoImagenSalida = [7.5 * densidadPixeles, 14 * densidadPixeles
            ];
10      % Tama o de la imagen de salida en pixeles: alto    ancho
11
12      bev = birdsEyeView(sensor, vistaSalida, tamanoImagenSalida);
13      % Crear objeto de vista de p jaro con los par metros dados
14
15      vistaPajaro = transformImage(bev, imgEntrada);
16      % Transformar la imagen de entrada a vista de p jaro
17
18  end
```

Listing B.7: obtenerVistaPajaroFrenteAtras.m

```
1   function vistaPajaro = obtenerVistaPajaroLados(imgEntrada,sensor)
2       %#codegen  %       Necesario para Simulink Code Generation
3
```

```
    vistaSalida = [0 8.06 -5 5]; %  rea  de transformaci n en metros
       : [frente atr s izquierda derecha]

    tamanoImagenSalida = [403, 500];
    % Tama o de la imagen de salida en pixeles: alto    ancho

    bev = birdsEyeView(sensor, vistaSalida, tamanoImagenSalida);
    % Crear objeto de vista de p jaro con la configuraci n indicada

    vistaPajaro = transformImage(bev, imgEntrada);
    % Aplicar la transformaci n de perspectiva a la imagen original

end
```

Listing B.8: obtenerVistaPajaroLados.m

```
function [imgSalida, centroX, centroY] = unirImagenes(imgA, imgB) %#
   codegen
    % Par metros fijos
    alto_total = 1000;
    % ancho_total = 1500;}
    ancho_total = 700;

    canales = 3;
    h_sep = 250;  % Separaci n vertical en p xeles
    % alto_img = 1500;
    alto_img2 = 375;
    alto_img1 = 375;
    % Inicializar salida
    imgSalida = zeros(alto_total, ancho_total, canales, 'uint8');

    % Rotar imgB
    imgB_rot = flip(flip(imgB,1),2);

    % Posiciones de pegado
    yA_ini = 1;
    yA_fin = yA_ini + alto_img1 - 1;

    yB_ini = yA_fin + h_sep + 1;
    yB_fin = yB_ini + alto_img2 - 1;

    % Centro visible (la parte  til )
    y_imgSalida_ini = round((yA_fin + yB_ini)/2 - alto_total/2);
    y_imgSalida_fin = y_imgSalida_ini + alto_total - 1;

    % Pegado dentro de imagen grande
    big_canvas = zeros(yB_fin, ancho_total, canales, 'uint8');
```

```
31
32      if yA_fin <= size(big_canvas,1)
33          big_canvas(yA_ini:yA_fin, :, :) = imgA;
34      end
35
36      if yB_ini <= size(big_canvas,1)
37          big_canvas(yB_ini:yB_fin, :, :) = imgB_rot;
38      end
39
40      % Recorte central
41      if y_imgSalida_fin <= size(big_canvas,1)
42          imgSalida = big_canvas(y_imgSalida_ini:y_imgSalida_fin, :, :)
                ;
43      else
44          imgSalida = big_canvas(end - alto_total + 1:end, :, :); %
                backup por seguridad
45      end
46
47      % Coordenadas del centro
48      centroX = int32(ancho_total / 2);
49      centroY = int32(alto_total / 2);
50  end
```

Listing B.9: unirImagenes.m

```
1   function imgSalida = unirImagenes2(imgA, imgB) %#codegen
2       %203,297
3       % Par metros fijos
4       alto_total = 1000;
5       % ancho_total = 1500;}
6       ancho_total = 700;
7
8       canales = 3;
9       v_sep = 94;
10      % alto_img = 1500;
11      ancho_img = 403;
12      % Inicializar salida
13      imgSalida = zeros(alto_total, ancho_total, canales, 'uint8');
14
15      % Rotar imgB
16      imgA_rot = rot90(imgA, 1);
17      imgB_rot = rot90(imgB,3);
18
19      imgA_rot = pad_to_height(imgA_rot,alto_total);
20      imgB_rot = pad_to_height(imgB_rot,alto_total);
21
22      % Posiciones de pegado
```

```
23      xA_ini = 1;
24      xA_fin = xA_ini + ancho_img - 1;
25
26      xB_ini = xA_fin + v_sep + 1;
27      xB_fin = xB_ini + ancho_img - 1;
28
29      % Centro visible (la parte  til )
30      x_imgSalida_ini = round((xA_fin + xB_ini)/2 - ancho_total/2);
31      x_imgSalida_fin = x_imgSalida_ini + ancho_total - 1;
32
33      % Pegado dentro de imagen grande
34      big_canvas = zeros(alto_total, xB_fin, canales, 'uint8');
35
36
37      if xA_fin <= size(big_canvas,2)
38          big_canvas(:,xA_ini:xA_fin,  :) = imgA_rot;
39      end
40
41      if xB_ini <= size(big_canvas,2)
42          big_canvas(:,xB_ini:xB_fin,  :) = imgB_rot;
43      end
44
45      % Recorte central
46      if x_imgSalida_fin <= size(big_canvas,2)
47          imgSalida = big_canvas(:, x_imgSalida_ini:x_imgSalida_fin, :)
                ;
48      else
49          imgSalida = big_canvas(:,end - ancho_total + 1:end, :); %
                backup por seguridad
50      end
51
52  end
53
54
55  function img_padded = pad_to_height(img, alto_objetivo)
56      %el padding se usa para ajustar una imagen a una altura
            espec fica
57
58      [alto_original, ancho, canales] = size(img); % Obtener
            dimensiones originales de la imagen
59      padding = alto_objetivo - alto_original;      % Calcular cu ntas
            filas faltan para llegar al alto deseado
60
61      if padding <= 0
62          img_padded = img(1:alto_objetivo,:,:);    % Si la imagen es
                m s alta, se recorta
```

```
63          return;
64      end
65
66      % Padding arriba y abajo (centrado)
67      pad_top = floor(padding / 2);                  % Mitad del padding
            hacia arriba
68      pad_bottom = padding - pad_top;                % Resto del padding
            hacia abajo
69
70      img_padded = padarray(img, [pad_top, 0], 0, 'pre');        %
            Agregar ceros arriba
71      img_padded = padarray(img_padded, [pad_bottom, 0], 0, 'post'); %
            Agregar ceros abajo
72  end
```

Listing B.10: unirImagenes2.m

```
1   function out = unirImagenes3(vistaPajaroA, vistaPajaroB)
2   %#codegen
3       % Asegurarse de que ambas im genes tengan el mismo tama o
4       assert(all(size(vistaPajaroA) == size(vistaPajaroB)), 'Las
            im genes deben tener el mismo tama o');
5
6       [h, w, ~] = size(vistaPajaroA);  % alto y ancho
7
8       % Crear rejilla de coordenadas
9       [xGrid, yGrid] = meshgrid(1:w, 1:h);
10
11      % Definir rectas
12      y_sup =-0.330144*xGrid+722.33014;
13      y_inf =  0.320574*xGrid+282.67943;
14
15      y_sup2=0.328571*xGrid+492;
16      y_inf2=-0.428571*xGrid+560;
17      % Regiones izquierda y derecha
18      mascara_izq = (xGrid <= 350) & (yGrid < y_sup) & (yGrid > y_inf);
19      mascara_der = (xGrid >= 350) & (yGrid < y_sup2) & (yGrid > y_inf2
            );
20
21      % M scara final combinada
22      mascara = mascara_izq | mascara_der;
23      mascaraRGB = repmat(mascara, [1, 1, 3]);  % expandir para RGB
24
25      % Inicializar salida
26      out = vistaPajaroA;
27
28      % Copiar pixeles de vistaPajaroB solo donde la m scara lo
```

```
        permite
    out(mascaraRGB) = vistaPajaroB(mascaraRGB);
end
```

Listing B.11: unirImagenes3.m

```
function imgBinaria = preprocesarImagen(vistaPajaro)
    %#codegen
    imagenGris = rgb2gray(vistaPajaro); % Convertir a escala de
        grises

    kernel = [-1 -1 -1 -1 0 1 1 1;
              -1 -1 -1 -1 0 1 1 1;
              -1 -1 -1 -1 0 1 1 1;
              -1 -1 -1 -1 0 1 1 1;
              -1 -1 -1 -1 0 1 1 1;
              -1 -1 -1 -1 0 1 1 1;
              -1 -1 -1 -1 0 1 1 1;
              -1 -1 -1 -1 0 1 1 1]; % Kernel para detecci n
                 direccional

    imgFiltrada = imfilter(imagenGris, kernel, 'replicate'); %
        Aplicar el filtro

    if size(imgFiltrada, 3) == 3
        imgFiltrada = rgb2gray(imgFiltrada);
    end
    imgBinaria = imgFiltrada;

    [h, w] = size(imgBinaria);
    mascara = zeros(h, w); % Inicializar m scara negra (en pixeles)

    % Pol gono ajustado para tapar la silueta completa del veh culo
        (parch n negro)

    % Pol gono central
    poligonoCentro = [...
        282, 372;
        355, 338;
        433, 370;
        407, 495;
        420, 632;
        350, 621;
        280, 629;
        292, 489
    ];

```

```
38      % Convertir a formato insertShape (X1,Y1,X2,Y2,...)
39      coords = reshape(poligonoCentro.', 1, []);
40
41      % Dibujar el parche sobre la imagen (en blanco)
42      mascara = insertShape(mascara, 'FilledPolygon', coords, 'Color',
            [255 255 255], 'Opacity', 1);
43
44      % Convertir a imagen binaria l gica
45      mascara = im2gray(mascara);
46      mascara = imbinarize(mascara);
47
48      % Aplicar la m scara: elimina los pixeles del veh culo que
            interfieren con la detecci n
49      imgBinaria(mascara) = 0;
50
51  end
```

Listing B.12: preprocesarImagen.m

```
1   function lineas = detectarLineas(imgBinaria, imagenColor)
2   %#codegen
3
4       % === PREPROCESAMIENTO DE LA IMAGEN BINARIA ===
5
6       suavizado = imgaussfilt(double(imgBinaria), 2);
7       % Aplicar un filtro gaussiano con sigma=2 para suavizar la imagen
            y reducir ruido
8
9       bordes = edge(suavizado, 'Canny', 0.1);
10      % Detectar bordes con el detector de Canny usando un umbral bajo
            (0.1) para mayor sensibilidad
11
12      bordes = bwareaopen(bordes, 20);
13      % Eliminar objetos peque os aislados que tengan menos de 20
            p xeles conectados
14
15      bordes = imdilate(bordes, strel('line', 5, 0));
16      % Expandir horizontalmente los bordes detectados usando una
            l nea estructurante de 5 p xeles ( ngulo  0  )
17      % Esto ayuda a conectar bordes que est n alineados pero
            separados por peque os huecos
18
19      cc = bwconncomp(bordes);
20      % Identificar componentes conectados: grupos de p xeles blancos
            conectados entre s  (posibles l neas)
21
22      stats = regionprops(cc, 'PixelList');
```

```
23      % Para cada componente conectado, obtener la lista de coordenadas
            de p xeles que lo conforman
24
25
26      % === CONFIGURACI N PARA SALIDA DE L NEAS ===
27
28      tamano = 100;           % Tama o m nimo de grupo de p xeles
            para ser considerado l nea
29      maxlineas = 100;        % M ximo n mero de l neas a detectar
30
31      estructuraVacia = struct( ...
32          'point1', [0 0], ...
33          'point2', [0 0], ...
34          'theta', 0, ...
35          'rho', 0, ...
36          'x_fit', zeros(1,100), ...
37          'y_fit', zeros(1,100), ...
38          'color', "", ...
39          'tipo', int32(0), ...
40          'continuidad', "desconocida" ...
41      ); % Estructura base vac a para cada l nea detectada
42
43      lineas = repmat(estructuraVacia, maxlineas, 1);
44      % Prealocar arreglo de l neas con tama o fijo para
            compatibilidad con codegen
45
46      indiceLinea = 1; % Contador de l neas v lidas
47
48      % === PROCESAMIENTO DE CADA COMPONENTE CONECTADO ===
49
50      for k = 1:min(length(stats), maxlineas)
51          pixels = stats(k).PixelList; % Lista de p xeles de la
                regi n k
52
53          if size(pixels,1) > tamano % Solo se consideran regiones
                suficientemente grandes
54              dy = double(max(pixels(:,2))) - double(min(pixels(:,2)));
55              dx = double(max(pixels(:,1))) - double(min(pixels(:,1)));
56
57              % Elegir el eje de mayor variaci n (X o Y) para ajustar
                    la curva
58              if dx < dy
59                  [~, idx] = sort(pixels(:,2));
60                  y = pixels(idx,2);
61                  x = pixels(idx,1);
62                  p = polyfit(y, x, 2); % Ajustar curva x = f(y)
```

```
63                  y_fit = linspace(min(y), max(y), 100);
64                  x_fit = polyval(p, y_fit);
65              else
66                  [~, idx] = sort(pixels(:,1));
67                  x = pixels(idx,1);
68                  y = pixels(idx,2);
69                  p = polyfit(x, y, 2); % Ajustar curva y = f(x)
70                  x_fit = linspace(min(x), max(x), 100);
71                  y_fit = polyval(p, x_fit);
72              end
73
74              % === ESTIMACI N DE COLOR ===
75
76              lin_idx = sub2ind(size(imgBinaria), pixels(:,2), pixels
                    (:,1)); %  ndices  lineales
77              R = imagenColor(:,:,1); G = imagenColor(:,:,2); B =
                    imagenColor(:,:,3);
78              prom = [mean(R(lin_idx)), mean(G(lin_idx)), mean(B(
                    lin_idx))]; % Promedio RGB del trazo
79
80              % Clasificaci n de color: heur sticas simples
81              if prom(1) > 120 && prom(2) > 120 && prom(3) < 160 && ...
82                      prom(1) - prom(3) > 20 && prom(2) - prom(3) > 20
83                  color = "amarilla";
84              elseif max([abs(prom(1) - prom(2)), abs(prom(2) - prom(3)
                    ), abs(prom(1) - prom(3))]) < 30 && mean(prom) > 130
85                  color = "blanca";
86              else
87                  disp(prom(1))
88                  disp(prom(2))
89                  disp(prom(3))
90                  color = "desconocida";
91              end
92
93              % === C LCULOS GEOM TRICOS ===
94
95              x1 = round(x_fit(1)); y1 = round(y_fit(1));
96              x2 = round(x_fit(end)); y2 = round(y_fit(end));
97
98              theta = atand((y2 - y1) / (x2 - x1 + eps)); %  ngulo  de
                    inclinaci n de la l nea
99              rho = x1 * cosd(theta) + y1 * sind(theta); % Distancia
                    desde el origen
100
101             % Asignar valores a la estructura de l nea
102             lineas(indiceLinea).point1 = [x1, y1];
```

```
103             lineas(indiceLinea).point2 = [x2, y2];
104             lineas(indiceLinea).theta = theta;
105             lineas(indiceLinea).rho = rho;
106             lineas(indiceLinea).x_fit = x_fit;
107             lineas(indiceLinea).y_fit = y_fit;
108             lineas(indiceLinea).color = color;
109             lineas(indiceLinea).tipo = int32(0);  % Inicializaci n
                    sin clasificaci n
110
111             indiceLinea = indiceLinea + 1;
112         end
113     end
114
115     % Truncar la salida al n mero real de l neas detectadas
116     lineas = lineas(1:indiceLinea-1);
117 end
```

Listing B.13: detectarLineas.m

```
1  function lineas = etiquetadoLineas(lineas, densidadPixeles)
2  %#codegen
3  % Clasifica continuidad y tipo b sico sin generar campos nuevos ni
       acceder antes de inicializar
4
5      umbral_largo_px = round(1.8 * densidadPixeles);  % 1.8 metros
           convertidos a p xeles
6      umbral_corto_px = round(1.3 * densidadPixeles);  % 1.3 metros
           convertidos a p xeles
7
8      for k = 1:length(lineas)
9          % Leer solo los datos estrictamente necesarios
10         x_fit = lineas(k).x_fit;
11         y_fit = lineas(k).y_fit;
12
13         % Inicializar etiquetas
14         tipo = int32(0);                % Tipo gen rico sin
               clasificar
15         continuidad = "desconocida";    % Estado por defecto si no se
               puede determinar
16
17         % Evitar errores por curvas vac as
18         if isempty(x_fit) || isempty(y_fit)
19             lineas(k).tipo = tipo;
20             lineas(k).continuidad = continuidad;
21             continue; % Saltar a la siguiente l nea
22         end
23
```

```
        % Calcular largo vertical en p xeles de la curva
        largo_px = abs(max(y_fit) - min(y_fit));

        % Clasificaci n por continuidad seg n longitud:
        if largo_px <= umbral_corto_px
            continuidad = "desconocida";  % Muy corta para decidir
        elseif largo_px > umbral_largo_px
            continuidad = "continua";     % Suficientemente larga
        else
            continuidad = "discontinua";  % Intermedia: probablemente
                l nea discontinua
        end

        tipo = int32(1); % Marcar como l nea v lida (gen rica)

        % Asignar etiquetas al struct
        lineas(k).tipo = tipo;
        lineas(k).continuidad = continuidad;
    end
end
```

Listing B.14: etiquetadoLineas.m

```
function lineasSalida = unirSegmentos(linesIn)
%#codegen

    R_doble = 50;   % Distancia m xima en X para considerar que una
        l nea es doble
    R_extX = 150;   % Rango en X para extender l neas similares
    R_extY = 600;   % Rango en Y para extender l neas similares
    maxLineas = 100; % N mero m ximo de l neas de salida

    usados = false(1, length(linesIn)); % Marcador para saber qu
        l neas ya se procesaron

    lineasSalida = repmat(struct( ...
        'x_fit', zeros(1,100), ...
        'y_fit', zeros(1,100), ...
        'tipo', int32(0), ...
        'color', "", ...
        'continuidad', "desconocida"), maxLineas, 1); % Prealocar
            salida

    outIdx = 1; %  ndice  para la l nea de salida actual

    % Recorrer todas las l neas de entrada
    for i = 1:length(linesIn)
```

```
22          if usados(i), continue; end % Saltar si ya fue usada
23
24          tipo = 1;                    % Tipo 1 por defecto (simple)
25          linesLeft = true;            % Control para seguir buscando
               extensiones
26          grupo = i;                   % Inicializar grupo con la l nea
               actual
27          usados(i) = true;            % Marcar como usada
28          base = linesIn(i);           % L nea base actual para
               comparaci n
29
30          while linesLeft
31              x_base = base.x_fit;
32              y_base = base.y_fit;
33              extension = 0;           % Indicador de si se extendi  el
                   grupo
34
35              % === PRIMER BUCLE: Unir l neas dobles ===
36              for j = i+1:length(linesIn)
37                  if usados(j), continue; end
38
39                  candidato = linesIn(j);
40                  % Solo unir si tienen mismo color y continuidad
41                  if candidato.color ~= base.color || candidato.
                       continuidad ~= base.continuidad
42                      continue;
43                  end
44
45                  % Comparar extremos
46                  pi1 = [x_base(1), y_base(1)];
47                  pi2 = [x_base(end), y_base(end)];
48                  pj1 = [candidato.x_fit(1), candidato.y_fit(1)];
49                  pj2 = [candidato.x_fit(end), candidato.y_fit(end)];
50
51                  distX = min([ ...
52                      abs(pi1(1) - pj1(1)), abs(pi1(1) - pj2(1)), ...
53                      abs(pi2(1) - pj1(1)), abs(pi2(1) - pj2(1))]);
54
55                  if distX < R_doble
56                      usados(j) = true;
57                      base.color = candidato.color; % Actualizar base
                           con color consistente
58                      grupo(end+1) = j;              % Agregar al grupo
59                  end
60              end
61
```

```
62              % === SEGUNDO BUCLE: Extender l neas similares en
                    posici n y direcci n ===
63              for j = i+1:length(linesIn)
64                  if usados(j), continue; end
65
66                  candidato = linesIn(j);
67                  if candidato.color ~= base.color || candidato.
                        continuidad ~= base.continuidad
68                      continue;
69                  end
70
71                  % Comparar extremos
72                  pi1 = [x_base(1), y_base(1)];
73                  pi2 = [x_base(end), y_base(end)];
74                  pj1 = [candidato.x_fit(1), candidato.y_fit(1)];
75                  pj2 = [candidato.x_fit(end), candidato.y_fit(end)];
76                  yCentro = linesIn(j).y_fit(floor(end/2));
77
78                  distX = min([ ...
79                      abs(pi1(1) - pj1(1)), abs(pi1(1) - pj2(1)), ...
80                      abs(pi2(1) - pj1(1)), abs(pi2(1) - pj2(1))]);
81
82                  distY = min([ ...
83                      abs(pi1(2) - pj1(2)), abs(pi1(2) - pj2(2)), ...
84                      abs(pi2(2) - pj1(2)), abs(pi2(2) - pj2(2))]);
85
86                  % Si est n cerca, extender
87                  if distX < R_extX && distY < R_extY
88                      base = linesIn(j);      % Nueva base
89                      usados(j) = true;
90                      grupo(end+1) = j;
91                      extension = 1;
92                  end
93
94                  % Detectar l neas dobles solapadas en Y
95                  yMin = min(linesIn(i).y_fit);
96                  yMax = max(linesIn(i).y_fit);
97                  if yCentro >= yMin && yCentro <= yMax && distX <
                        R_doble
98                      tipo = 2; % Marcar como l nea doble
99                  end
100             end
101
102             if ~extension
103                 linesLeft = false; % Salir si no se encontr  m s
                        para extender
```

```
            end
        end

        % === Calcular el rango Y com n para ajustar ===
        n = length(grupo);
        y_min_list = zeros(n,1);
        y_max_list = zeros(n,1);

        for k = 1:n
            idx = grupo(k);
            y_vals = linesIn(idx).y_fit;
            y_min_list(k) = min(y_vals);
            y_max_list(k) = max(y_vals);
        end

        % Eliminar duplicados y ordenar
        y_min_unicos = unique(y_min_list);
        y_max_unicos = unique(y_max_list);

        y_min_ordenado = sort(y_min_unicos, 'ascend');
        y_max_ordenado = sort(y_max_unicos, 'descend');

        if length(y_min_ordenado) >= 2 && tipo == 2
            y_min = y_min_ordenado(2);
        else
            y_min = y_min_ordenado(1);
        end
        if length(y_max_ordenado) >= 2 && tipo == 2
            y_max = y_max_ordenado(2);
        else
            y_max = y_max_ordenado(1);
        end

        % === Fusionar puntos de todo el grupo ===
        total = 0;
        x_all = zeros(1, 100 * length(grupo));
        y_all = zeros(1, 100 * length(grupo));

        for k = 1:length(grupo)
            idx = grupo(k);
            n = length(linesIn(idx).x_fit);
            x_all(total+1:total+n) = linesIn(idx).x_fit;
            y_all(total+1:total+n) = linesIn(idx).y_fit;
            total = total + n;
        end

```

```
150         % Filtrar por el rango Y com n
151         mask = (y_all >= y_min) & (y_all <= y_max);
152         x_all = x_all(mask);
153         y_all = y_all(mask);
154 
155         % Ajustar polinomio cuadr tico a los puntos fusionados
156         [y_all, idxSort] = sort(y_all);
157         x_all = x_all(idxSort);
158 
159         p = polyfit(y_all, x_all, 2);
160         y_fit = linspace(y_min, y_max, 100);
161         x_fit = polyval(p, y_fit);
162 
163         % Guardar resultado en la salida
164         lineasSalida(outIdx).x_fit = x_fit;
165         lineasSalida(outIdx).y_fit = y_fit;
166         lineasSalida(outIdx).tipo = int32(tipo);
167         lineasSalida(outIdx).color = base.color;
168         lineasSalida(outIdx).continuidad = base.continuidad;
169 
170         outIdx = outIdx + 1;
171     end
172 
173     % Recortar la salida al n mero real de l neas generadas
174     lineasSalida = lineasSalida(1:outIdx-1);
175 end
176 
177 function out = fixedLengthString10(str)
178 % Devuelve un char(1,10) con espacios rellenados
179     out = char(32 * ones(1,10)); % Espacios
180     str = char(str);
181     n = min(length(str), 10);
182     out(1:n) = str(1:n);
183 end
184 
185 function out = fixedLengthString15(str)
186 % Devuelve un char(1,15) con espacios rellenados
187     out = char(32 * ones(1,15)); % Espacios
188     str = char(str);
189     n = min(length(str), 15);
190     out(1:n) = str(1:n);
191 end
```

Listing B.15: unirSegmentos.m

```
1 function curvaCentral = unirCarrilesCentrales(curvas, A, B)
2 %#codegen
```

```
3
4       minDistIzq = inf;    % M nima distancia a una curva a la
            izquierda del punto A
5       minDistDer = inf;    % M nima distancia a una curva a la derecha
            del punto A
6       idxIzq = -1;         %  ndice  de la curva izquierda m s cercana
7       idxDer = -1;         %  ndice  de la curva derecha m s cercana
8
9       A = double(A);       % Coordenada X del centro del veh culo (por
            ejemplo)
10      B = double(B);       % Coordenada Y del centro del veh culo
11
12      % Buscar la curva m s cercana a la izquierda y a la derecha del
            punto (A,B)
13      for k = 1:length(curvas)
14          x = curvas(k).x_fit;
15          y = curvas(k).y_fit;
16
17          idx_izq = x < A;    % Puntos a la izquierda del veh culo
18          idx_der = x > A;    % Puntos a la derecha del veh culo
19
20          if any(idx_izq)
21              dist = sqrt((x(idx_izq) - A).^2 + (y(idx_izq) - B).^2);
22              [dmin, ~] = min(dist);
23              if dmin < minDistIzq
24                  minDistIzq = dmin;
25                  idxIzq = k;
26              end
27          end
28
29          if any(idx_der)
30              dist = sqrt((x(idx_der) - A).^2 + (y(idx_der) - B).^2);
31              [dmin, ~] = min(dist);
32              if dmin < minDistDer
33                  minDistDer = dmin;
34                  idxDer = k;
35              end
36          end
37      end
38
39      % Estructura vac a por defecto
40      curvaCentral = struct( ...
41          'x_fit', zeros(1,100), ...
42          'y_fit', zeros(1,100));
43
44      if idxIzq == -1 || idxDer == -1
```

```
45          return;  % No se encontr  una curva a cada lado del
                veh culo
46      end
47
48      % Extraer curvas seleccionadas
49      tempIzq = curvas(idxIzq);
50      tempDer = curvas(idxDer);
51
52      x_izq = tempIzq.x_fit;
53      y_izq = tempIzq.y_fit;
54      x_der = tempDer.x_fit;
55      y_der = tempDer.y_fit;
56
57      % Determinar el rango Y en com n entre ambas curvas
58      y_min = max(min(y_izq), min(y_der));
59      y_max = min(max(y_izq), max(y_der));
60      y_fit = linspace(y_min, y_max, 100);
61
62      % Interpolar las curvas izquierda y derecha al mismo vector Y
63      x_izq_interp = interp1(y_izq, x_izq, y_fit, 'linear', 'extrap');
64      x_der_interp = interp1(y_der, x_der, y_fit, 'linear', 'extrap');
65
66      % Promediar para obtener la curva central
67      x_central = (x_izq_interp + x_der_interp) / 2;
68
69      curvaCentral.x_fit = x_central;
70      curvaCentral.y_fit = y_fit;
71
72      % Ajustar polinomio cuadr tico a la curva central calculada
73      p = polyfit(y_fit, x_central, 2);
74
75      % Extender la curva central para que cubra toda la imagen
            vertical
76      y_extendido = linspace(0, 1200, 100);  % Asume imagen de altura
            1200 px
77      x_extendido = polyval(p, y_extendido);
78
79      curvaCentral.x_fit = x_extendido;
80      curvaCentral.y_fit = y_extendido;
81  end
```

Listing B.16: unirCarrilesCentrales.m

```
1  function [distancia, angulo, curvatura] = distanciaDesdePunto(curva,
       A, B, densidad)
2  %#codegen
3  % Calcula la distancia m nima desde el punto (A,B) a una curva
```

```
% También estima la curvatura (1/m) y el ángulo (rad) con respecto al eje vertical

    x = curva.x_fit;
    y = curva.y_fit;
    A = double(A);
    B = double(B);

    % Distancia mínima (en metros)
    d = sqrt((x - A).^2 + (y - B).^2);
    distancia = -min(d) / densidad;

    % Porción central ±100 px verticalmente
    margen = 100;
    idx = y >= (B - margen) & y <= (B + margen);

    if nnz(idx) < 5
        angulo = 0;
        curvatura = 0;
        return;
    end

    % Recorte de los puntos
    x_recorte = x(idx);
    y_recorte = y(idx);

    % Convertir a metros para el ajuste
    x_m = x / densidad;
    y_m = y / densidad;

    % Ajuste en metros
    p = polyfit(y_m, x_m, 2);  % x = a*y^2 + b*y + c
    a = p(1);
    b = p(2);

    % Evaluar en el centro (B convertido a metros)
    y0 = B / densidad;

    dx1 = 2 * a * y0 + b;         % primera derivada
    dx2 = 2 * a;                  % segunda derivada

    % Curvatura para x = f(y)         = f'' / (1 + f'^2)^(3/2)
    curvatura = dx2 / (1 + dx1^2)^(3/2);  % en 1/m


    %  ángulo  con respecto al eje vertical
```

```
49      angulo = atan(dx1);   % en radianes
50
51  end
```

Listing B.17: distanciaDesdePunto.m

B.2.3 Diagramas de Simulink

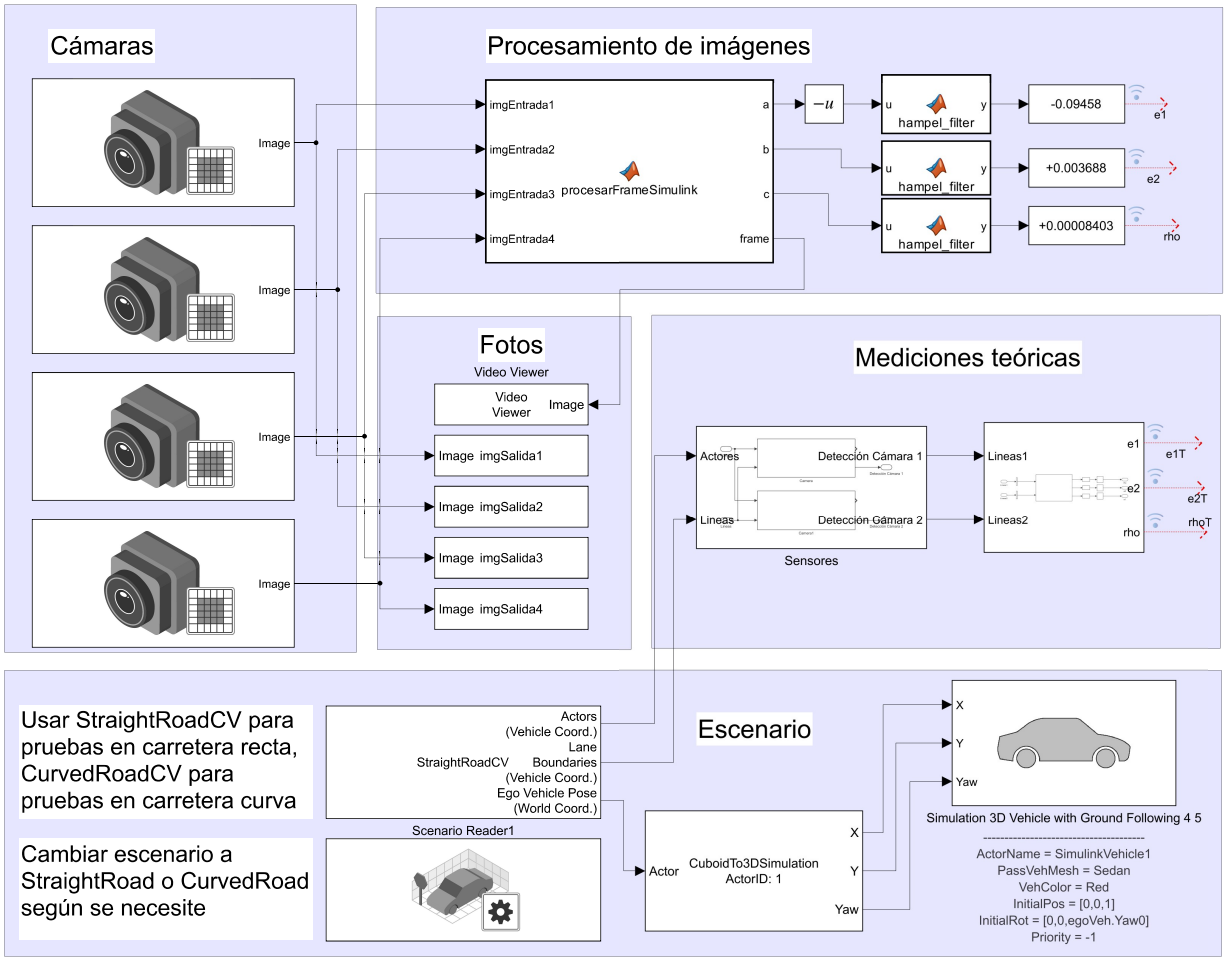


Figura B.5: Diagrama para las pruebas de la detección de carril.

B.2.4 Códigos de Simulink

```
1  function [a,b,c,frame] = procesarFrameSimulink(imgEntrada1,
       imgEntrada2,imgEntrada3, imgEntrada4)
2  %#codegen
3
4      % Asegurarse de que las im genes tienen la dimensi n esperada
5      imgEntrada1 = imgEntrada1(:, :, :, end);
6      imgEntrada2 = imgEntrada2(:, :, :, end);
```

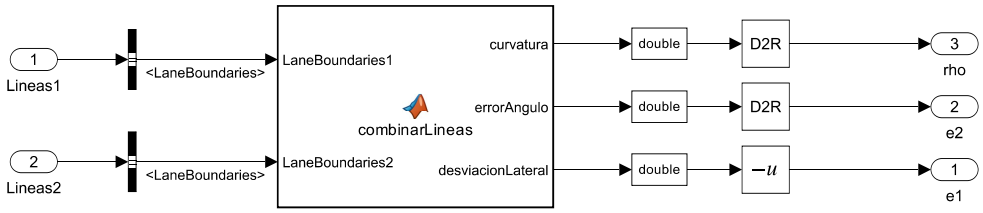


Figura B.6: Diagrama para combinar las lecturas de carril teóricas.

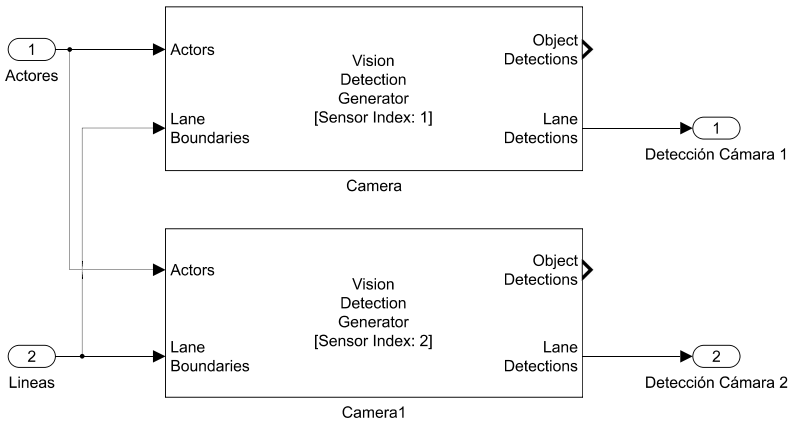


Figura B.7: Diagrama para el sistema de detección teórico.

```
    imgEntrada3 = imgEntrada3(:, :, :, end);
    imgEntrada4 = imgEntrada4(:, :, :, end);
    % Declarar sensores persistentes
    persistent sensor1 sensor2 sensor3 sensor4

    if isempty(sensor1)
        % Par metros fijos
        sz = [1080, 1920];  % Alto x ancho
        fov = 74;
        f = [sz(2)/(2*tand(fov)), sz(2)/(2*tand(fov))];
        pp = [sz(2)/2, sz(1)/2];

        intrinsics = cameraIntrinsics(f, pp, sz);

        sensor1 = monoCamera(intrinsics, 0.7, 'Pitch', 30);
        sensor2 = monoCamera(intrinsics, 0.894, 'Pitch', 30);
        sensor3 = monoCamera(intrinsics, 0.7, 'Pitch', 30);
        sensor4 = monoCamera(intrinsics, 0.7, 'Pitch', 30);
    end

    % Ejecutar el procesamiento
    [a,b,c,frame] = procesarFrame2(imgEntrada1, imgEntrada2,
       imgEntrada3, imgEntrada4, sensor1, sensor2,sensor3,sensor4);
end
```

Listing B.18: procesarFrameSimulink.m

```
function [curvatura, errorAngulo, desviacionLateral] = combinarLineas
   (LaneBoundaries1, LaneBoundaries2)
% Ajusta curvatura, heading y error lateral combinando m ltiples
   l neas fuertes
% con correcci n seg n posici n relativa del carril

LaneWidth = 3.6;

% Valores por defecto
curvatura = 0;
errorAngulo = 0;
desviacionLateral = 0;

% Unir listas
if isempty(LaneBoundaries1)
    allBoundaries = LaneBoundaries2(:);
elseif isempty(LaneBoundaries2)
    allBoundaries = LaneBoundaries1(:);
else
    allBoundaries = [LaneBoundaries1(:); LaneBoundaries2(:)];
```

```
19 end
20
21 if isempty(allBoundaries)
22     return;
23 end
24
25 % Inicializar acumuladores
26 sumCurv = 0;
27 sumAngulo = 0;
28 sumDesvLat = 0;
29 count = 0;
30
31 % Recorrer todos los boundaries
32 for i = 1:numel(allBoundaries)
33     B = allBoundaries(i);
34
35     if B.Strength >= 0.05
36         % Corregir lateral offset con respecto al centro del carril
37         if B.LateralOffset >= 0
38             lateralError = B.LateralOffset - LaneWidth / 2; % l nea
                   a la izquierda
39         else
40             lateralError = B.LateralOffset + LaneWidth / 2; % l nea
                   a la derecha
41         end
42
43         % Acumular
44         sumCurv = sumCurv + B.Curvature;
45         sumAngulo = sumAngulo + B.HeadingAngle;
46         sumDesvLat = sumDesvLat + lateralError;
47         count = count + 1;
48     end
49 end
50
51 % Calcular promedios si hubo al menos una l nea fuerte
52 if count > 0
53     curvatura = sumCurv / count;
54     errorAngulo = sumAngulo / count;
55     desviacionLateral = sumDesvLat / count;
56 end
57
58 end
```

Listing B.19: combinarLineas.m

B.3 Validación del sistema de percepción: detección de vehículos

B.3.1 Código para la inicialización de variables en MATLAB

```
%% Algunas constantes

% Constantes de aceleraci n y frenado
tau = 0.5;
tau2 = 0.07;

% L mites de acci n para el agente (aceleraci n lateral y
   longitudinal)
u_min = -1;     % L mite inferior para acci n de direcci n (giros)
u_max = 1;      % L mite superior para acci n de direcci n
ax_min = -1;    % L mite inferior para aceleraci n longitudinal
ax_max = 1;     % L mite superior para aceleraci n longitudinal

T = 20;
Ts = 0.1;

egoActorID=double(1);

%% Posici n de inicio

egoVeh.X0 = 0;  % Posici n inicial el X (m)
egoVeh.Y0 = 0;         % Posici n inicial en Y (m)
egoVeh.Yaw0 = 0; % Orientaci n inicial en radianes (-87 grados)

egoVeh.VX0 = 0;         % Velocidad inicial en X (m/s)
egoVeh.VY0 = 0;         % Velocidad inicial en Y (m/s)
egoVeh.VLong0 = 0;      % Velocidad longitudinal inicial (m/s)

%% Cargar bus
load("Sistema de percepci n\AlgoritmoRADAR\busPose.mat")
```

Listing B.20: helperLidar.m

B.3.2 Diagramas de Simulink

B.3.3 Códigos de Simulink

```
function [centrosXY,frame1,frame2,frame3,frame4]  =
   procesarLidarSimulink(rawLidarFrame)
```

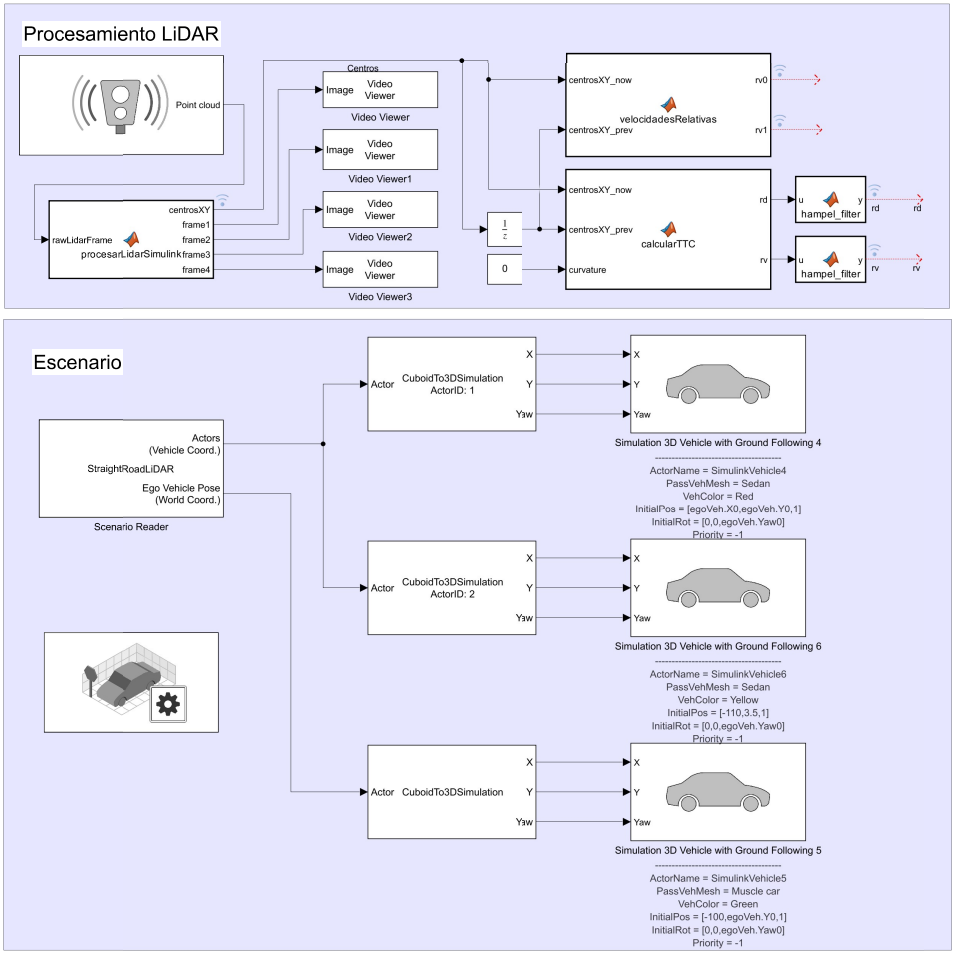


Figura B.8: Diagrama para las pruebas de la detección de vehículos.

```
2
3       %% Parte gr fica
4       canvasSize = 600;  % Tama o del lienzo en p xeles (600x600)
5       escala = 10;       % Escala: 10 p xeles por metro
6       centroCanvas = round(canvasSize / 2);  % Centro del lienzo
7
8       % Inicializar los 4 frames para visualizaci n de distintas
           etapas
9       frame1 = uint8(zeros(canvasSize, canvasSize, 3));
10      frame2 = uint8(zeros(canvasSize, canvasSize, 3));
11      frame3 = uint8(zeros(canvasSize, canvasSize, 3));
12      frame4 = uint8(zeros(canvasSize, canvasSize, 3));
13
14      % rawLidarFrame: tama o [32x2250x3]     convertir a [72000x3]
15      xyzAll = reshape(rawLidarFrame, [], 3);
16
17      xyzOut = zeros(72000, 3);  % Prealocaci n para eficiencia
18
19      % === Filtrar puntos v lidos (ni NaN ni todos ceros) ===
20      valid = all(isfinite(xyzAll),2) & any(xyzAll ~= 0, 2);
21      xyzAll = xyzAll(valid,:);
22
23      % === Pintar todos los puntos v lidos en frame1 ===
24      for k = 1:size(xyzAll,1)
25          x_px = round(centroCanvas + xyzAll(k,1) * escala);
26          y_px = round(centroCanvas - xyzAll(k,2) * escala);
27          if x_px > 0 && x_px <= canvasSize && y_px > 0 && y_px <=
               canvasSize
28              frame1(y_px, x_px, :) = 255; % blanco
29          end
30      end
31
32      % Reconfirmar puntos v lidos manualmente (redundante con filtro
           anterior)
33      count = 0;
34      for i = 1:size(xyzAll,1)
35          pt = xyzAll(i,:);
36          if all(isfinite(pt)) && any(pt ~= 0)
37              count = count + 1;
38              xyzOut(count,:) = pt;
39          end
40      end
41
42      % Rellenar con NaN el resto del arreglo
43      for i = count+1:size(xyzOut,1)
44          xyzOut(i,:) = NaN;
```

```
45      end
46
47      xyz = xyzOut;
48
49      maxObjs = 2;                        % M ximo n mero de objetos a
            detectar
50      roi = [-45 45 -10 10 -1.8 1.2];  % Regi n de inter s (Xmin Xmax
            Ymin Ymax Zmin Zmax)
51
52      centrosXY = NaN(maxObjs, 2);      % Inicializar salida
53
54      % Recortar puntos fuera del ROI y del ego-car
55      xyz = cropPointCloudByROI(xyz, roi);
56
57      % === Pintar los puntos dentro del ROI en frame2 ===
58      for k = 1:size(xyz,1)
59          x_px = round(centroCanvas + xyz(k,1) * escala);
60          y_px = round(centroCanvas - xyz(k,2) * escala);
61          if x_px > 0 && x_px <= canvasSize && y_px > 0 && y_px <=
                canvasSize
62              frame2(y_px, x_px, :) = 255;
63          end
64      end
65
66      % === Aplicar reducci n por v xeles ===
67      voxelSize = 0.2;
68      maxVoxels = 500;
69      xyz = voxelGridFilter(xyz, voxelSize, maxVoxels);
70
71      % === Pintar resultado de voxelizaci n en frame3 ===
72      for k = 1:size(xyz,1)
73          x_px = round(centroCanvas + xyz(k,1) * escala);
74          y_px = round(centroCanvas - xyz(k,2) * escala);
75          if x_px > 0 && x_px <= canvasSize && y_px > 0 && y_px <=
                canvasSize
76              frame3(y_px, x_px, :) = 255;
77          end
78      end
79
80      % === Agrupar con DBSCAN ===
81      epsilon = 1;
82      minPts = 5;
83      labels = dbscan(xyz, epsilon, minPts);
84      uniqueLabels = unique(labels);
85      uniqueLabels(uniqueLabels == -1) = []; % Eliminar ruido
86
```

```
87      % Inicializar variables de acumulaci n
88      todosXY = NaN(maxObjs, 2);
89      distancias = NaN(maxObjs, 1);
90      count = 1;
91
92      for i = 1:numel(uniqueLabels)
93          idx = labels == uniqueLabels(i);
94          cluster = xyz(idx, :);
95
96          minXYZ = min(cluster, [], 1);
97          maxXYZ = max(cluster, [], 1);
98          dims = maxXYZ - minXYZ;
99          height = dims(3);  % altura del objeto
100
101         centro = (minXYZ + maxXYZ) / 2;
102         objXY = centro(1:2);
103         distancia = norm(objXY);
104
105         if height < 0.4 || height > 4.0 || distancia < 0.5
106             continue;  % descartar objetos fuera del rango esperado
107         end
108
109         % Obtener punto m s cercano al origen
110         clusterXY = cluster(:,1:2);
111         dists = vecnorm(clusterXY, 2, 2);
112         [~, idxMin] = min(dists);
113         objXY = clusterXY(idxMin, :);
114
115         % Guardar posici n del objeto
116         todosXY(count,:) = objXY;
117         distancias(count) = norm(objXY);
118
119         % Dibujar objeto detectado en frame4
120         x_px = round(centroCanvas + objXY(1) * escala);
121         y_px = round(centroCanvas - objXY(2) * escala);
122         for dx = -1:1
123             for dy = -1:1
124                 xi = x_px + dx;
125                 yi = y_px + dy;
126                 if xi > 0 && xi <= canvasSize && yi > 0 && yi <=
                        canvasSize
127                     frame4(yi, xi, :) = [0, 255, 0]; % verde
128                 end
129             end
130         end
131
```

```
132         % A adir coordenadas como texto sobre la imagen
133         texto = sprintf('(%.1f, %.1f)', objXY(1), objXY(2));
134         frame4 = insertText(frame4, [x_px+5, y_px-10], texto, ...
135                             'FontSize', 20, 'BoxColor', [0 0 0], '
                                TextColor',[255 255 255]);
136
137         count = count + 1;
138         if count > maxObjs
139             break;
140         end
141     end
142
143     % === Ordenar por distancia al origen ===
144     if count > 1
145         [~, orden] = sort(distancias(1:count-1));
146         centrosXY(1:length(orden), :) = todosXY(orden(1:min(maxObjs,
              length(orden))), :);
147     end
148
149     % === Dibujar el veh culo ego en el centro ===
150     for dx = -1:1
151          for dy = -1:1
152              xi = centroCanvas + dx;
153              yi = centroCanvas + dy;
154              if xi > 0 && xi <= canvasSize && yi > 0 && yi <=
                     canvasSize
155                  frame4(yi, xi, 1) = 255; % rojo (R canal)
156              end
157          end
158     end
159 end
160
161 function croppedXYZ = cropPointCloudByROI(xyzPoints, pointCloudRange)
162 %#codegen
163 % Recorta puntos dentro del ROI 3D especificado
164
165     % Zona rectangular central que representa el ego-car (a excluir)
166     xInnerMin = -2.5; xInnerMax = 2.5;
167     yInnerMin = -0.9; yInnerMax = 0.9;
168
169     % Eliminar puntos dentro del ego-car
170     inEgoBox = xyzPoints(:,1) >= xInnerMin & xyzPoints(:,1) <=
            xInnerMax & ...
171                xyzPoints(:,2) >= yInnerMin & xyzPoints(:,2) <=
                      yInnerMax;
172
```

```
    xyzPoints = xyzPoints(~inEgoBox,:);  % filtrar fuera del ego-car

    % Extraer l mites del ROI
    xMin = pointCloudRange(1); xMax = pointCloudRange(2);
    yMin = pointCloudRange(3); yMax = pointCloudRange(4);
    zMin = pointCloudRange(5); zMax = pointCloudRange(6);

    % Filtrar puntos fuera del rango 3D
    mask = xyzPoints(:,1) >= xMin & xyzPoints(:,1) <= xMax & ...
           xyzPoints(:,2) >= yMin & xyzPoints(:,2) <= yMax & ...
           xyzPoints(:,3) >= zMin & xyzPoints(:,3) <= zMax;

    croppedXYZ = xyzPoints(mask, :);  % salida filtrada
end
function xyz_ds = voxelGridFilter(xyz, voxelSize, maxOut)
%#codegen
% Reduce la nube xyz con v xeles  c bicos de lado voxelSize

    if isempty(xyz)
        xyz_ds = NaN(maxOut, 3);
        return;
    end

    % Convertir puntos a  ndices  de v xeles
    voxelIdx = floor(xyz / voxelSize);

    % Inicializar estructuras para hash y acumulaci n
    xyz_ds = NaN(maxOut, 3);
    hashMap = zeros(maxOut, 3);
    hashUsed = false(maxOut,1);
    accum = zeros(maxOut, 3);
    counts = zeros(maxOut,1);

    % Funci n hash personalizada para 3D
    hashFn = @(i,j,k) mod(i*73856093 + j*19349663 + k*83492791,
       maxOut) + 1;

    for n = 1:size(xyz,1)
        idx = voxelIdx(n,:);
        h = hashFn(idx(1), idx(2), idx(3)); % calcular hash

        % Colisi n     linear probing
        while hashUsed(h) && any(hashMap(h,:) ~= idx)
            h = mod(h, maxOut) + 1;
        end

```

```
218         if ~hashUsed(h)
219             hashUsed(h) = true;
220             hashMap(h,:) = idx;
221         end
222 
223         accum(h,:) = accum(h,:) + xyz(n,:);
224         counts(h) = counts(h) + 1;
225     end
226 
227     % Calcular centroides por v xel
228     outIdx = 1;
229     for h = 1:maxOut
230         if hashUsed(h)
231             xyz_ds(outIdx,:) = accum(h,:) / counts(h);
232             outIdx = outIdx + 1;
233             if outIdx > maxOut
234                 break;
235             end
236         end
237     end
238 
239     % Rellenar con NaN si hay espacio sobrante
240     for i = outIdx:maxOut
241         xyz_ds(i,:) = NaN;
242     end
243 end
```

Listing B.21: procesarLidarSimulink.m

```
1  function [rd, rv] = calcularTTC(centrosXY_now, centrosXY_prev,
       curvature)
2  % Busca el veh culo  l der con menor TTC usando solo posici n XY y
       velocidad estimada
3  
4      rangoMaxCam = 30;    % Valor m ximo de distancia detectable
5      Ts = 0.1;            % Tiempo entre muestras (delta t)
6      laneWidth = 4;       % Ancho de carril en metros
7      halfLaneWidth = laneWidth / 2;
8  
9      rd = rangoMaxCam;    % Distancia inicial (por defecto, m xima)
10     rv = 0;              % Velocidad relativa inicial
11     minTTC = inf;        % Valor inicial del menor Time To Collision (
           TTC)
12 
13     % Iterar sobre todos los veh culos detectados
14     for i = 1:size(centrosXY_now,1)
15         posNow = centrosXY_now(i,:);    % Posici n actual [x, y]
```

```
16          posPrev = centrosXY_prev(i,:); % Posici n previa [x, y]
17
18          if any(isnan(posNow)) || any(isnan(posPrev))
19              continue; % Omitir si hay datos faltantes
20          end
21
22          rx = posNow(1);  % Posici n X relativa actual
23          ry = posNow(2);  % Posici n Y relativa actual
24          vx = (posNow(1) - posPrev(1)) / Ts;  % Velocidad X estimada
25          vy = (posNow(2) - posPrev(2)) / Ts;  % Velocidad Y estimada
26
27          % Evaluar si el objeto est  dentro de los carriles izquierdo
                o derecho,
28          % utilizando una par bola centrada con curvatura
29          yLeft  = polyval([curvature/2, 0,  halfLaneWidth], rx);
30          yRight = polyval([curvature/2, 0, -halfLaneWidth], rx);
31
32          inLeftBand  = (ry < yLeft  && ry > yLeft  - halfLaneWidth);
33          inRightBand = (ry > yRight && ry < yRight + halfLaneWidth);
34
35          if ~(inLeftBand || inRightBand)
36              continue;  % No est  en el carril actual     descartar
37          end
38
39          dist = sqrt(rx^2 + ry^2);         % Distancia euclidiana al
                objeto
40          vrel = sqrt(vx^2 + vy^2);         % Magnitud de la velocidad
                relativa
41
42          % Considerar solo si se aproxima (direcci n opuesta) y
                velocidad no nula
43          if sign(vx) ~= sign(rx) && vrel > 0
44              ttc = dist / vrel;            % Tiempo estimado hasta
                    colisi n
45              if ttc < minTTC
46                  minTTC = ttc;
47                  if rx < 0
48                      rd = -dist;           % Distancia con signo
                            negativo si est  detr s
49                  else
50                      rd = dist;
51                  end
52                  rv = vrel;                % Guardar velocidad relativa
                        m nima
53              end
54          end
```

```
55
56          if rv > 15
57              rv = 15;                        % Limitar la velocidad a 15
                    m/s
58          end
59      end
60  end
```

Listing B.22: calcularTTC.m

```
1  function [rv0, rv1] = velocidadesRelativas(centrosXY_now,
       centrosXY_prev)
2  % Busca el veh culo l der con menor TTC usando solo posici n XY y
       velocidad estimada
3  Ts=0.1;
4  rv1 = 0;
5  rv0 = 0;
6
7  posNow = centrosXY_now(1,:);
8  posPrev = centrosXY_prev(1,:);
9
10
11 vx = (posNow(1) - posPrev(1)) / Ts;
12 vy = (posNow(2) - posPrev(2)) / Ts;
13
14 rv0 = sqrt(vx^2 + vy^2);
15
16 posNow = centrosXY_now(2,:);
17 posPrev = centrosXY_prev(2,:);
18
19
20
21 vx = (posNow(1) - posPrev(1)) / Ts;
22 vy = (posNow(2) - posPrev(2)) / Ts;
23
24 rv1 = sqrt(vx^2 + vy^2);
```

Listing B.23: velocidadesRelativas.m

B.4 Validación del agente: fase previa

B.4.1 Código para la inicialización de variables en MATLAB

```
1  %% Par metros de entrenamiento
2  usarParalelo=true;          % Activar entrenamiento en paralelo
```

```
T = 60;                          % Duraci n del episodio (segundos)
Ts = 0.1;                        % Tiempo de muestreo (segundos)
egoActorID=double(1);            % ID del veh culo ego
doTraining=3;                    % Modo: 1=nuevo, 2=reentrenar, 3=cargar
    modelo

%% Posici n de inicio
Start.P1.X0=5.775;               % Posici n inicial X
Start.P1.Y0= 1.305;              % Posici n inicial Y
Start.P1.Yaw0=pi/2;              % Orientaci n inicial (rad)

s=rand;                          % Ruido aleatorio auxiliar
r1=-0.1+0.2*rand;                % Ruido en X
r2=-0.025+0.05*rand;             % Ruido en orientaci n

egoVeh.X0  = Start.P1.X0 - r1;   % Posici n X con ruido
egoVeh.Y0  = Start.P1.Y0;        % Posici n Y sin ruido
egoVeh.Yaw0 = Start.P1.Yaw0 - r2; % Orientaci n con ruido
egoVeh.VX0 =  0;                 % Velocidad en X (m/s)
egoVeh.VY0 = 0;                  % Velocidad en Y (m/s)
egoVeh.VLong0 = 0;               % Velocidad longitudinal (m/s)

%% Par metros del modelo din mico
egoVeh.Length = 4.7;                          % Largo del veh culo
egoVeh.RearOverhang = 1;                      % Voladizo trasero
egoVeh.CGToFrontAxle = egoVeh.Length/2 - 1;   % CG a eje delantero
egoVeh.CGToRearAxle  = egoVeh.Length/2 - 1;   % CG a eje trasero

m = 1575;      % Masa total del veh culo (kg)
Iz = 2875;     % Inercia de gui ada (mNs )
lf = 1.2;      % Distancia CG a eje delantero (m)
lr = 1.6;      % Distancia CG a eje trasero (m)
Cf = 19000;    % Rigidez de las llantas delanteras (N/rad)
Cr = 33000;    % Rigidez de las llantas traseras (N/rad)


%% Configuraci n de entorno
executionEnvironment = "auto";     % Simulaci n: auto = GPU o CPU
    seg n disponibilidad
assignin('base', 'executionEnvironment', executionEnvironment);

%% Algunas constantes
rangoXCam=30;                      % Rango de detecci n en X (m)
rangoYCam=6;                       % Rango de detecci n en Y (m)
rangoMaxCam=sqrt(rangoYCam^2+rangoXCam^2);
distanciaImpacto=7;                % Distancia de colisi n (m)
```

```
47 vTarget = 5;                          % Velocidad objetivo (m/s)
48
49 % Normalizaci n de observaciones
50 e1Max=0.75;
51 e2Max=20*pi/180;
52 vMax=15;
53 vMin=0;
54 e3Max1=abs(vTarget-vMin);
55 e3Max2=abs(vTarget-vMax);
56
57 % Constantes de aceleraci n y frenado
58 tau=0.5;
59 tau2=0.07;
60
61 % L mites de acci n para el agente
62 u_min = -1;
63 u_max = 1;
64 ax_min = -1;
65 ax_max = 1;
66
67 %% Configuraci n agente RL
68
69 pasosSimu = 1024;                % N mero de episodios
70 maxepisodes = pasosSimu;
71 maxsteps = T/Ts;
72
73 % Configuraci n del entrenamiento
74
75 trainingOpts = rlTrainingOptions(...
76     MaxEpisodes = maxepisodes, ...
77     MaxStepsPerEpisode = maxsteps, ...
78     Verbose = false, ...
79     Plots = "training-progress", ...
80     SaveAgentCriteria="EvaluationStatistic", ...
81     SaveAgentValue=550);
82
83 trainingOpts.UseParallel = usarParalelo;
84 trainingOpts.ParallelizationOptions.Mode = "sync";
85
86 % Evaluaci n
87 evl = rlEvaluator( ...
88     NumEpisodes=10, ...
89     EvaluationFrequency=256, ...
90     RandomSeeds=101:110);
91
92 % Configuraci n del entorno Simulink
```

```
93 taskNumber = 1;
94 env.ResetFcn = @(in)localResetFcn(in, taskNumber);
95
96 mdl = "preagentePPO";
97 agentblk = mdl + "/Agente PPO previo/RL Agent";
98
99 % Especificaci n del espacio de observaci n
100 obsInfo = rlNumericSpec([6 1], ...
101     LowerLimit=-1*ones(6,1), ...
102     UpperLimit=1*ones(6,1));
103 obsInfo.Name = "observations";
104
105 % Especificaci n del espacio de acciones
106 actInfo = rlNumericSpec([2 1], ...
107     LowerLimit=[u_min; ax_min], ...
108     UpperLimit=[u_max; ax_max]);
109 actInfo.Name = "action";
110
111 env = rlSimulinkEnv(mdl, agentblk, obsInfo, actInfo);
112 env.ResetFcn = @(in)localResetFcn(in);
113
114 %% Entrenamiento del agente
115 if doTraining==1
116     rng(0);
117
118     initOptions = rlAgentInitializationOptions("NumHiddenUnit", 512);
119     agentPPO = rlPPOAgent(obsInfo, actInfo, initOptions);
120
121     % Configuraci n de opciones del agente
122
123     agentPPO.AgentOptions.SampleTime = Ts;
124     agentPPO.AgentOptions.DiscountFactor = 0.99;
125     agentPPO.AgentOptions.EntropyLossWeight = 0.01;
126     agentPPO.AgentOptions.ExperienceHorizon = 2048;
127     agentPPO.AgentOptions.MiniBatchSize = 64;
128     agentPPO.AgentOptions.NumEpoch = 7;
129     agentPPO.AgentOptions.MaxMiniBatchPerEpoch = 100;
130     agentPPO.AgentOptions.LearningFrequency = 2048;
131     agentPPO.AgentOptions.ClipFactor = 0.2;
132
133     % Configuraci n de los optimizadores
134
135     agentPPO.AgentOptions.ActorOptimizerOptions.LearnRate = 3e-4;
136     agentPPO.AgentOptions.ActorOptimizerOptions.GradientThreshold =
        1;
137     agentPPO.AgentOptions.CriticOptimizerOptions.LearnRate = 3e-4;
```

```
138     agentPPO.AgentOptions.CriticOptimizerOptions.GradientThreshold =
            1;
139
140     trainingStats = train(agentPPO, env, trainingOpts, Evaluator=evl)
            ;
141     save(fullfile(sprintf("Agentes Guardados\\Entrenamiento previo\\
            agent_Iter%d.mat", pasosSimu)), 'agentPPO');
142
143 elseif doTraining==2
144     inicio=512;
145     load(fullfile(sprintf("Agentes Guardados\\Entrenamiento previo\\
            agent_Iter%d.mat", inicio)));
146     trainingStats = train(agentPPO, env, trainingOpts, Evaluator=evl)
            ;
147     save(fullfile(sprintf("Agentes Guardados\\Entrenamiento previo\\
            agent_Iter%d.mat", inicio + pasosSimu)), 'agentPPO');
148
149 elseif doTraining==3
150     load("Agentes Guardados\\Entrenamiento previo\\agent_Iter768.mat
            ");
151 end
152
153 %% Cargar buses
154
155 load('Agente\\Entrenamiento previo\\busVision.mat');
156 load('Agente\\Entrenamiento previo\\busPose.mat');
157 load('Agente\\Entrenamiento previo\\busLane.mat');
```

Listing B.24: prehelperPPO.m

B.4.2 Código para la re-inicialización de variables en MATLAB

```
1  function in = localResetFcn(in)
2
3          Start.P1.X0=5.775;
4          Start.P1.Y0= 1.305;
5          Start.P1.Yaw0=pi/2;
6
7          Start.P2.X0=1.925;
8          Start.P2.Y0= -1.305;
9          Start.P2.Yaw0=pi/2;
10
11         Start.P3.X0=-1.925;
12         Start.P3.Y0= -1.305;
13         Start.P3.Yaw0=-pi/2;
14
```

```
15          Start.P4.X0=-5.775;
16          Start.P4.Y0=1.305;
17          Start.P4.Yaw0=-pi/2;
18
19          s=rand;
20          r1=-0.1+0.2*rand;
21          s=rand;
22          r2=-0.025+0.05*rand;
23          if s<0.25
24              egoVeh.X0  = Start.P1. X0-r1; % (m) =x-511.33/100
25              egoVeh.Y0  = Start.P1.Y0; % (m) =-y+40612/100
26              egoVeh.Yaw0 = Start.P1.Yaw0-r2; % (rad) =180-angulo
27          elseif s<0.5
28              egoVeh.X0  = Start.P2. X0-r1; %
29              egoVeh.Y0  = Start.P2.Y0; %
30              egoVeh.Yaw0 = Start.P2.Yaw0-r2; %
31          elseif s<0.75
32              egoVeh.X0  = Start.P3. X0-r1; %
33              egoVeh.Y0  = Start.P3.Y0; %
34              egoVeh.Yaw0 = Start.P3.Yaw0-r2; %
35          elseif s<=1
36              egoVeh.X0  = Start.P4. X0-r1; % (m)
37              egoVeh.Y0  = Start.P4.Y0; % (m)
38              egoVeh.Yaw0 = Start.P4.Yaw0-r2; % (rad)
39          end
40          egoVeh.VX0 =  0; % (m)
41          egoVeh.VY0 = 0; % (m)
42          % Longitudinal velocity
43          egoVeh.VLong0 =0; % (m/sec)
44
45          % Set random value for lateral deviation.
46          in = setVariable(in,'egoVeh.X0', egoVeh.X0);
47          in = setVariable(in,'egoVeh.Y0', egoVeh.Y0);
48          in = setVariable(in,'egoVeh.Yaw0', egoVeh.Yaw0);
49          in = setVariable(in,'egoVeh.VX0', egoVeh.VX0);
50          in = setVariable(in,'egoVeh.VY0', egoVeh.VY0);
51          in = setVariable(in,'egoVeh.VLong0', egoVeh.VLong0);
52  end
```

Listing B.25: localResetFcn.m

B.4.3 Diagramas de Simulink

B.4.4 Códigos de Simulink

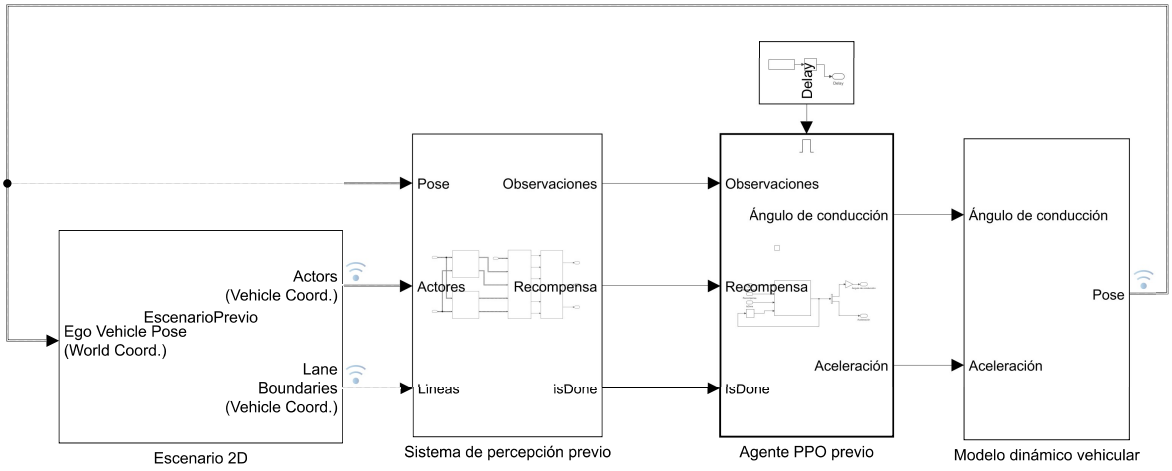


Figura B.9: Diagrama del entorno de simulación para el entrenamiento y validación del agente previo.

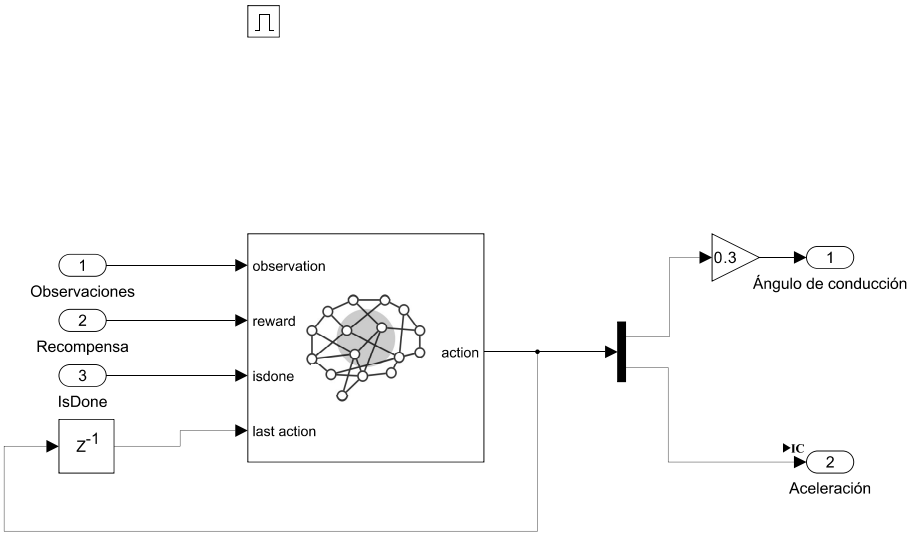


Figura B.10: Diagrama de conexiones del agente PPO previo.

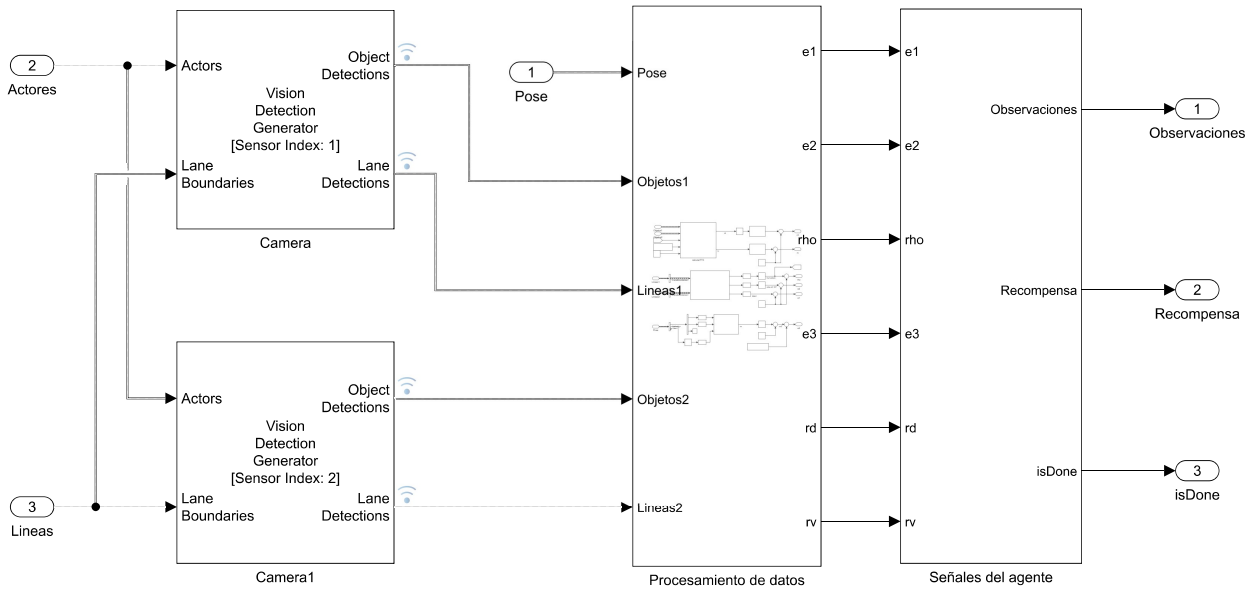


Figura B.11: Diagrama de conexiones del sistema de percepción previo.

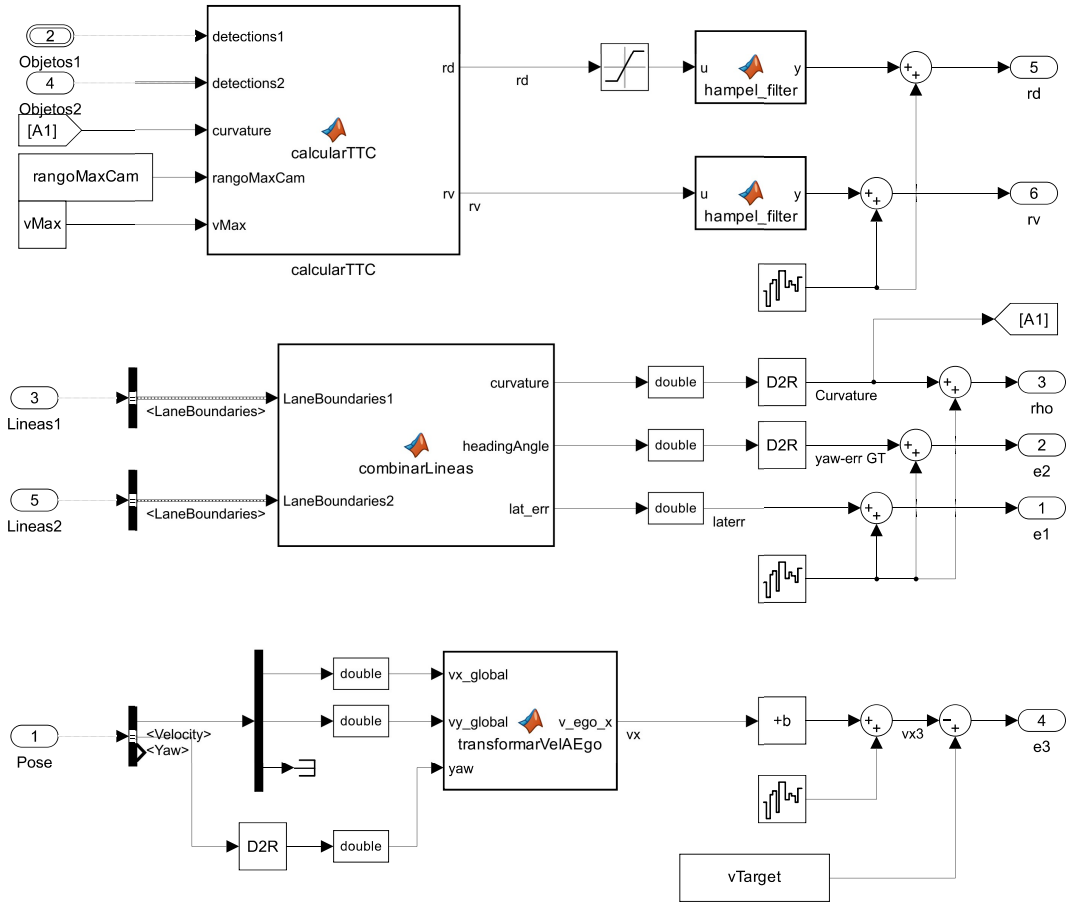


Figura B.12: Diagrama de conexiones del procesamiento de datos.

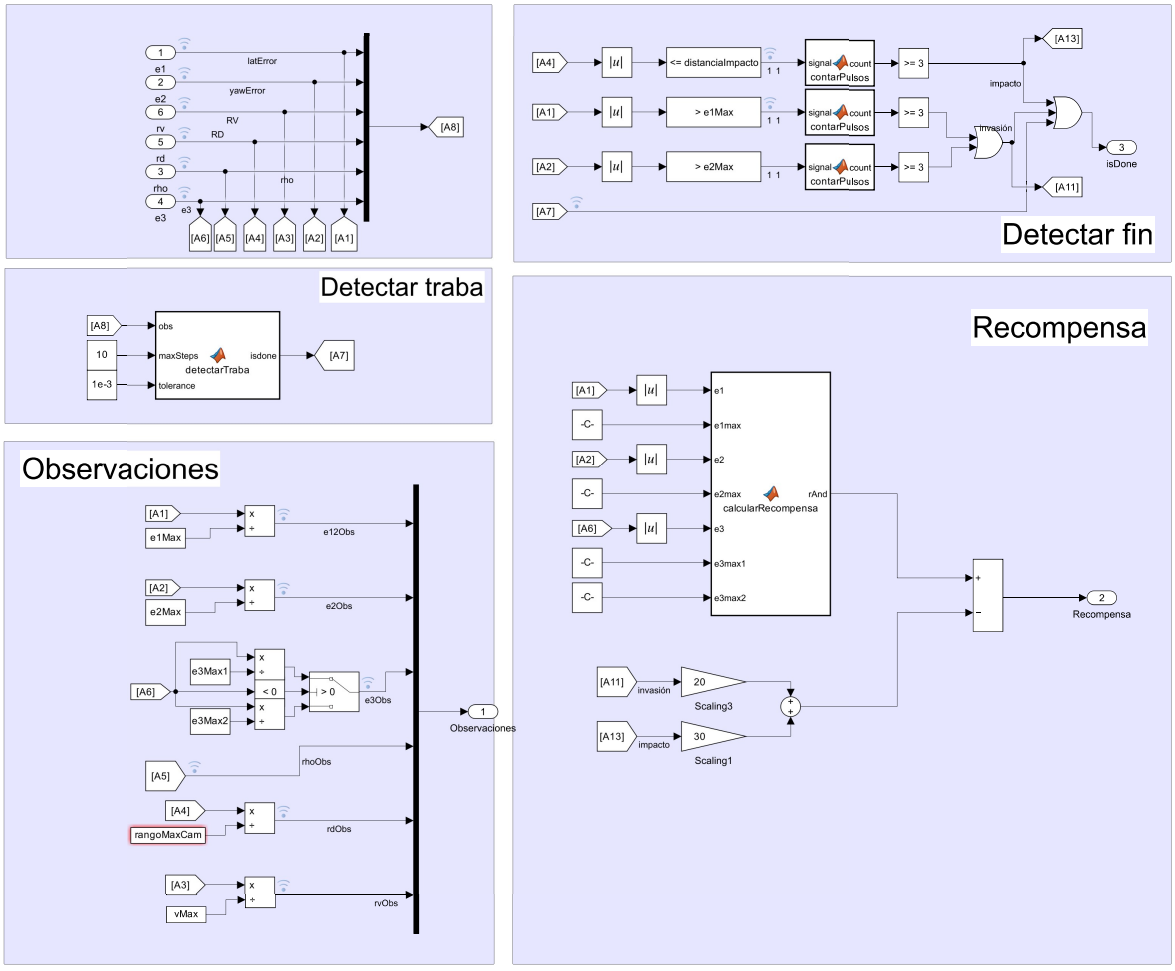


Figura B.13: Diagrama de conexiones de las señales del agente PPO previo.

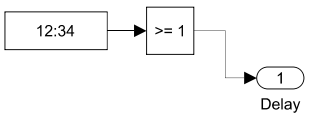


Figura B.14: Diagrama de conexiones del delay para el agente PPO previo.

```
function rAnd = calcularRecompensa(e1, e1max, e2, e2max, e3, e3max1,
    e3max2)
%#codegen

a=1;
b=1;
c=1;

% recompensa asociada a e1
a=1-e1/e1max;
rCenter=clip(a,0,1);

% recompensa asociada a e2
b=1-e2/e2max;
rHeading=clip(b,0,1);

% recompensa asociada a e3
if e3<0
    c=1-abs(e3)/e3max1;
end
if e3==0
    c=1;
end
if e3>0
    c=1-abs(e3)/e3max2;
end

rSpeed=clip(c,0,1);

% recompensa and
rAnd = rCenter*rHeading*rSpeed;

end


function y = clip(x, minVal, maxVal)
    y = min(max(x, minVal), maxVal);
end
```

Listing B.26: calcularRecompensa.m

```
function count = contarPulsos(signal)
% Cuenta cu ntos pasos consecutivos la se al ha estado en 1

persistent stepCount

if isempty(stepCount)
```

```
    stepCount = 0;
end

if signal == 1
    stepCount = stepCount + 1;
else
    stepCount = 0;
end

count = stepCount;
end
```

Listing B.27: contarPulsos.m

```
function isdone  = detectarTraba(obs, maxSteps, tolerance)
%codegen

% obs: vector [6x1]
% maxSteps: pasos m ximos permitidos sin cambios significativos
% tolerance: tolerancia para considerar que hubo un cambio

persistent lastObs stuckCount

if isempty(lastObs)
    lastObs = obs;
    stuckCount = 0;
end

% Comparar el vector completo
if norm(obs - lastObs) < tolerance
    stuckCount = stuckCount + 1;
else
    stuckCount = 0;
end

lastObs = obs; % actualizar para siguiente paso

if stuckCount >= maxSteps
    isdone = true;
else
    isdone = false;
end

end
```

Listing B.28: detectarTraba.m

B.5 Validación del agente: fase final

B.5.1 Código para la inicialización de variables en MATLAB

```
%% Par metros de entrenamiento
usarParalelo=true;          % Indica si se usar  entrenamiento en
    paralelo
T = 30;                     % Tiempo total del episodio (segundos)
Ts = 0.1;                   % Tiempo de muestreo (segundos)
egoActorID=double(1);       % ID del veh culo ego

doTraining=3;               % Modo de entrenamiento: 1=nuevo, 2=
    continuar, 3=cargar

%% Posici n de inicio
Start.P1.X0=223.4;           % Posici n inicial X del escenario
Start.P1.Y0= -152.471;       % Posici n inicial Y del escenario
Start.P1.Yaw0=-87*pi/180;    % Orientaci n inicial (rad)

r1=-0.1+0.2*rand;            % Ruido aleatorio en X
r2=-0.025+0.05*rand;         % Ruido aleatorio en yaw

egoVeh.X0  = Start.P1.X0-r1;
egoVeh.Y0  = Start.P1.Y0;
egoVeh.Yaw0 = Start.P1.Yaw0-r2;
egoVeh.VX0 =  0;             % Velocidad en X (m/s)
egoVeh.VY0 = 0;              % Velocidad en Y (m/s)
egoVeh.VLong0 = 0;           % Velocidad longitudinal (m/s)

%% Posici n de inicio de actores
egoVeh.X1  = 0;
egoVeh.Y1  = 0;
egoVeh.Yaw1 = 0;

egoVeh.X2  = 0;
egoVeh.Y2  = 0;
egoVeh.Yaw2 = 0;

%% Par metros del modelo din mico
egoVeh.Length=4.7;                  % Largo total del veh culo
egoVeh.RearOverhang=1;              % Voladizo trasero
egoVeh.CGToFrontAxle = egoVeh.Length/2 - 1; % CG a eje delantero
egoVeh.CGToRearAxle  = egoVeh.Length/2 - 1; % CG a eje trasero

m = 1575;                           % Masa total del veh culo (kg)
```

```
Iz = 2875;                          % Momento de inercia de gui ada (
    mNs^2)
lf = 1.2;                           % Distancia CG a llantas delanteras
     (m)
lr = 1.6;                           % Distancia CG a llantas traseras (
    m)
Cf = 19000;                         % Rigidez lateral llantas
    delanteras (N/rad)
Cr = 33000;                         % Rigidez lateral llantas traseras
    (N/rad)

%% Configuraci n de las c maras

% Par metros del sensor de c mara
NumColumns=1920;
NumRows=1080;
FOV=74;                         % Campo de visi n
pitchCam=30;                    % Inclinaci n hacia el suelo (grados)



% Configuraci n com n a todas las c maras
camera.NumColumns = NumColumns;
camera.NumRows = NumRows;
camera.FieldOfView = [FOV,FOV];
camera.ImageSize = [camera.NumRows, camera.NumColumns];
camera.PrincipalPoint = [camera.NumColumns,camera.NumRows]/2;
camera.FocalLength = [camera.NumColumns / (2*tand(FOV)), camera.
    NumColumns / (2*tand(FOV))];

camera2 = camera;
camera3 = camera;
camera4 = camera;

camera.Position = [2.5, 0, 0.7];              % Frontal
camera2.Position = [-2.5, 0, 0.894];          % Trasera
camera3.Position = [0, 1.88/2, 0.7];          % Lateral derecha
camera4.Position = [0, -1.88/2, 0.7];         % Lateral izquierda

camera.Rotation = [0, pitchCam, 0];
camera2.Rotation = [0, pitchCam, 180];
camera3.Rotation = [0, pitchCam, 90];
camera4.Rotation = [0, pitchCam, 270];

%% Configuraci n de entorno
executionEnvironment = "auto";     % Entorno de ejecuci n autom tico
```

```
assignin('base', 'executionEnvironment', executionEnvironment);


%% Algunas constantes
rangoXMax=30;                % Rango en X de la c mara (m)
rangoYMax=6;                 % Rango en Y de la c mara (m)
rangoMax=sqrt(rangoYMax^2+rangoXMax^2);
distanciaImpacto=4;            % Distancia m nima para considerar
    colisi n (m)
vTarget = 5;                 % Velocidad objetivo (m/s)

% L mites de observaci n normalizados
e1Max=0.75;
e2Max=20*pi/180;
vMax=15;
vMin=0;
e3Max1=abs(vTarget-vMin);
e3Max2=abs(vTarget-vMax);

% Constantes de aceleraci n y frenado
tau=0.5;
tau2=0.07;

% L mites de acci n
u_min = -1;
u_max = 1;
ax_min = -1;
ax_max = 1;

%% Configuraci n agente RL

pasosSimu=256;                 % N mero de episodios

maxepisodes = pasosSimu;
maxsteps = T/Ts;

% Configuraci n del entrenamiento

trainingOpts = rlTrainingOptions(...
    MaxEpisodes = maxepisodes, ...
    MaxStepsPerEpisode = maxsteps, ...
    Verbose = false, ...
    Plots = "training-progress", ...
    StopTrainingCriteria='None', ...
    SaveAgentCriteria="EvaluationStatistic", ...
    SaveAgentValue=550);
```

```
125
126 trainingOpts.UseParallel=usarParalelo;
127 trainingOpts.ParallelizationOptions.Mode = "sync";
128
129 % Evaluaci n
130
131 evl = rlEvaluator( ...
132     NumEpisodes=10, ...
133     EvaluationFrequency=128, ...
134     RandomSeeds=101:110);
135
136 % Configuraci n del entorno Simulink
137
138 mdl = "agentePPO";
139 agentblk = mdl + "/Agente PPO/RL Agent";
140
141
142 % Especificaci n del espacio de observaci n
143
144 obsInfo = rlNumericSpec([6 1],...
145     LowerLimit=-1*ones(6,1), ...
146     UpperLimit= 1*ones(6,1));
147 obsInfo.Name = "observations";
148
149 % Especificaci n del espacio de acciones
150
151 actInfo = rlNumericSpec([2 1], ...
152     LowerLimit=[u_min; ax_min], ...
153     UpperLimit=[u_max; ax_max]);
154 actInfo.Name = "action";
155
156 env = rlSimulinkEnv(mdl,agentblk,obsInfo,actInfo);
157 env.ResetFcn = @(in)localResetFcn(in);
158
159 % === Modo de entrenamiento ===
160 if doTraining==1
161     rng(0);
162
163     % Configuraci n de opciones del agente
164
165     initOptions = rlAgentInitializationOptions("NumHiddenUnit",512);
166     agentPPO = rlPPOAgent(obsInfo, actInfo, initOptions);
167     agentPPO.AgentOptions.SampleTime = Ts;
168     agentPPO.AgentOptions.DiscountFactor=0.99;
169     agentPPO.AgentOptions.EntropyLossWeight=0.01;
170     agentPPO.AgentOptions.ExperienceHorizon=2048;
```

```
171     agentPPO.AgentOptions.MiniBatchSize = 64;
172     agentPPO.AgentOptions.NumEpoch=7;
173     agentPPO.AgentOptions.MaxMiniBatchPerEpoch=100;
174     agentPPO.AgentOptions.LearningFrequency = 2048;
175     agentPPO.AgentOptions.ClipFactor=0.2;
176
177     % Configuraci n de los optimizadores
178     agentPPO.AgentOptions.ActorOptimizerOptions.LearnRate  = 3e-4;
179     agentPPO.AgentOptions.ActorOptimizerOptions.GradientThreshold =
            1;
180     agentPPO.AgentOptions.CriticOptimizerOptions.LearnRate = 3e-4;
181     agentPPO.AgentOptions.CriticOptimizerOptions.GradientThreshold =
            1;
182
183     trainingStats = train(agentPPO, env, trainingOpts, Evaluator=evl)
            ;
184     save(fullfile(sprintf("Agentes Guardados\\Entrenamiento final\\
            agentUE_Iter%d.mat", pasosSimu)), 'agentPPO');
185
186 elseif doTraining==2
187     inicio=1864;
188     load(fullfile(sprintf("Agentes Guardados\\Entrenamiento final\\
            agentUE_Iter%d.mat", (inicio))));
189     trainingStats = train(agentPPO,env,trainingOpts,Evaluator=evl);
190     save(fullfile(sprintf("Agentes Guardados\\Entrenamiento final\\
            agentUE_Iter%d.mat", inicio+pasosSimu)), 'agentPPO');
191
192 elseif doTraining==3
193     load("Agentes Guardados\\Entrenamiento final\\agentUE_Iter1810.
            mat");  % Cargar modelo entrenado
194 end
195
196 %% Cargar bus
197 load('Agente\Entrenamiento final\busPose.mat');
```

Listing B.29: helperPPO.m

B.5.2 Código para la re-inicialización de variables en MATLAB

```
1 function in = localResetFcn(in)
2
3
4         Start.P1.X0=223.4;
5         Start.P1.Y0= -152.471;
6         Start.P1.Yaw0=-87*pi/180;
7
```

```
8
9           egoVeh.X0  = Start.P1. X0-r1; % (m) =x-511.33/100
10          egoVeh.Y0  = Start.P1.Y0; % (m) =-y+40612/100
11          egoVeh.Yaw0 = Start.P1.Yaw0-r2; % (rad) =180-angulo
12
13
14          egoVeh.VX0 =  0; % (m)
15          egoVeh.VY0 = 0; % (m)
16          % Longitudinal velocity
17          egoVeh.VLong0 =0; % (m/sec)
18
19          % Set random value for lateral deviation.
20          in = setVariable(in,'egoVeh.X0', egoVeh.X0);
21          in = setVariable(in,'egoVeh.Y0', egoVeh.Y0);
22          in = setVariable(in,'egoVeh.Yaw0', egoVeh.Yaw0);
23          in = setVariable(in,'egoVeh.VX0', egoVeh.VX0);
24          in = setVariable(in,'egoVeh.VY0', egoVeh.VY0);
25          in = setVariable(in,'egoVeh.VLong0', egoVeh.VLong0);
26  end
```

Listing B.30: localResetFcn.m

B.5.3 Diagramas de Simulink

B.5.4 Códigos de Simulink

```
1  function y = hampel_filter(u)
2  % Hampel Filter para eliminar picos en se ales
3  % Basado en la mediana y desviaci n robusta
4
5  % Par metros configurables
6  N = 5;            % Tama o de la ventana (debe ser impar)
7  nSigma = 2.5;       % Multiplicador de umbral para detecci n de
       outliers
8
9  % Variables persistentes
10 persistent window
11
12 if isempty(window)
13     window = u*ones(N,1);  % Inicializa con primer valor
14 end
15
16 % Actualizar ventana
17 window = [window(2:end); u];
18
```

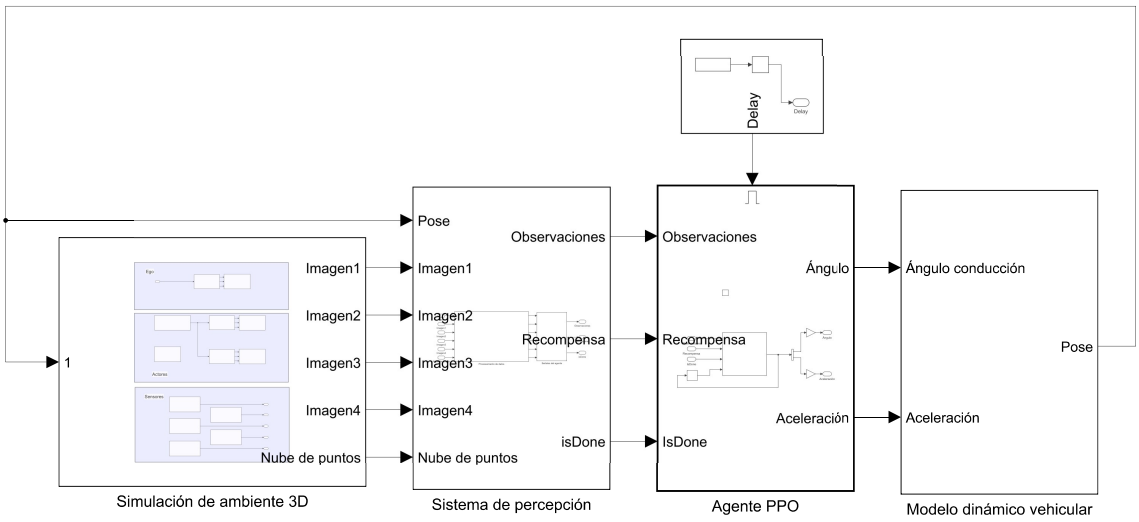


Figura B.15: Diagrama del entorno de simulación para el entrenamiento y validación del agente final.

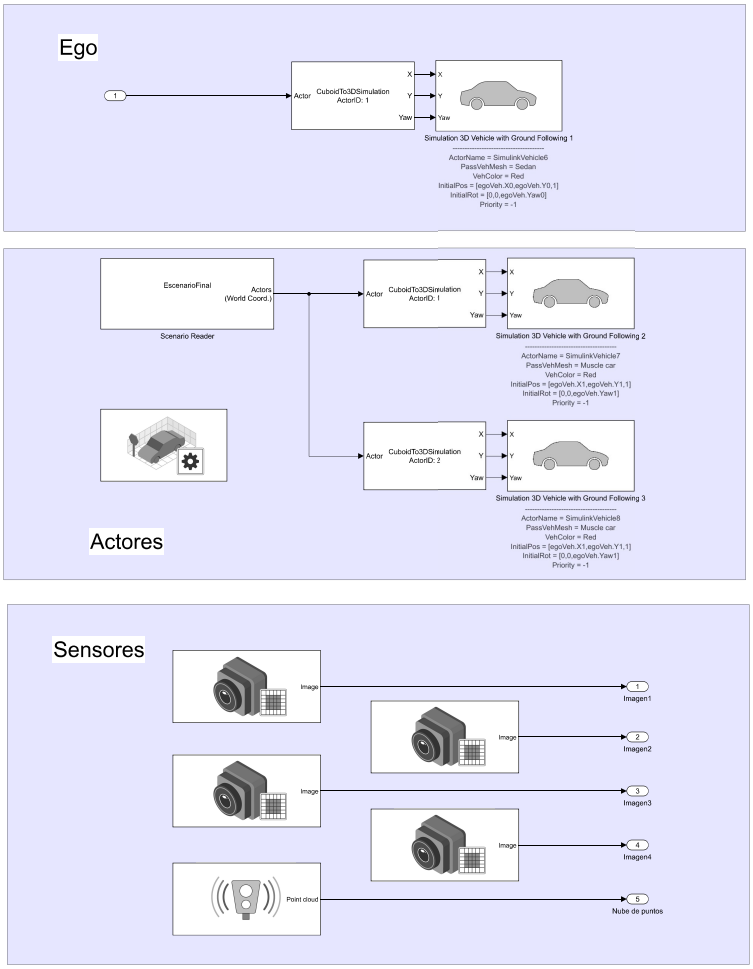


Figura B.16: Diagrama de conexiones del escenario final.

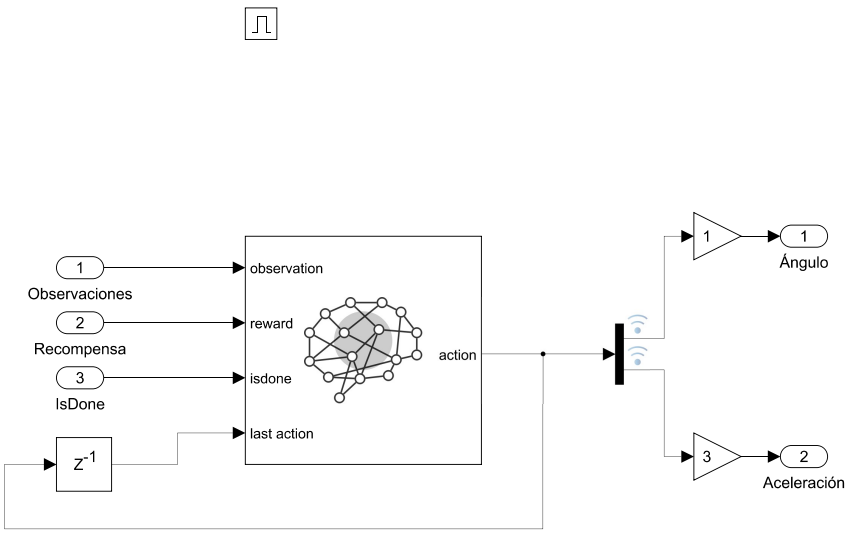


Figura B.17: Diagrama de conexiones del agente PPO previo.

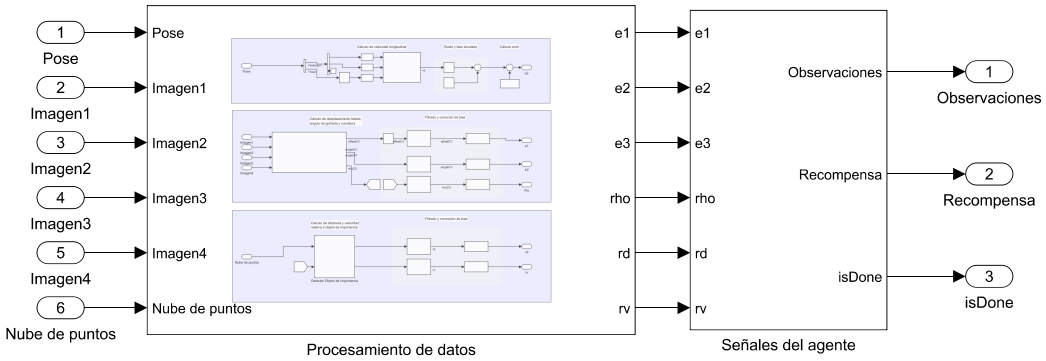


Figura B.18: Diagrama de conexiones del sistema de percepción previo.

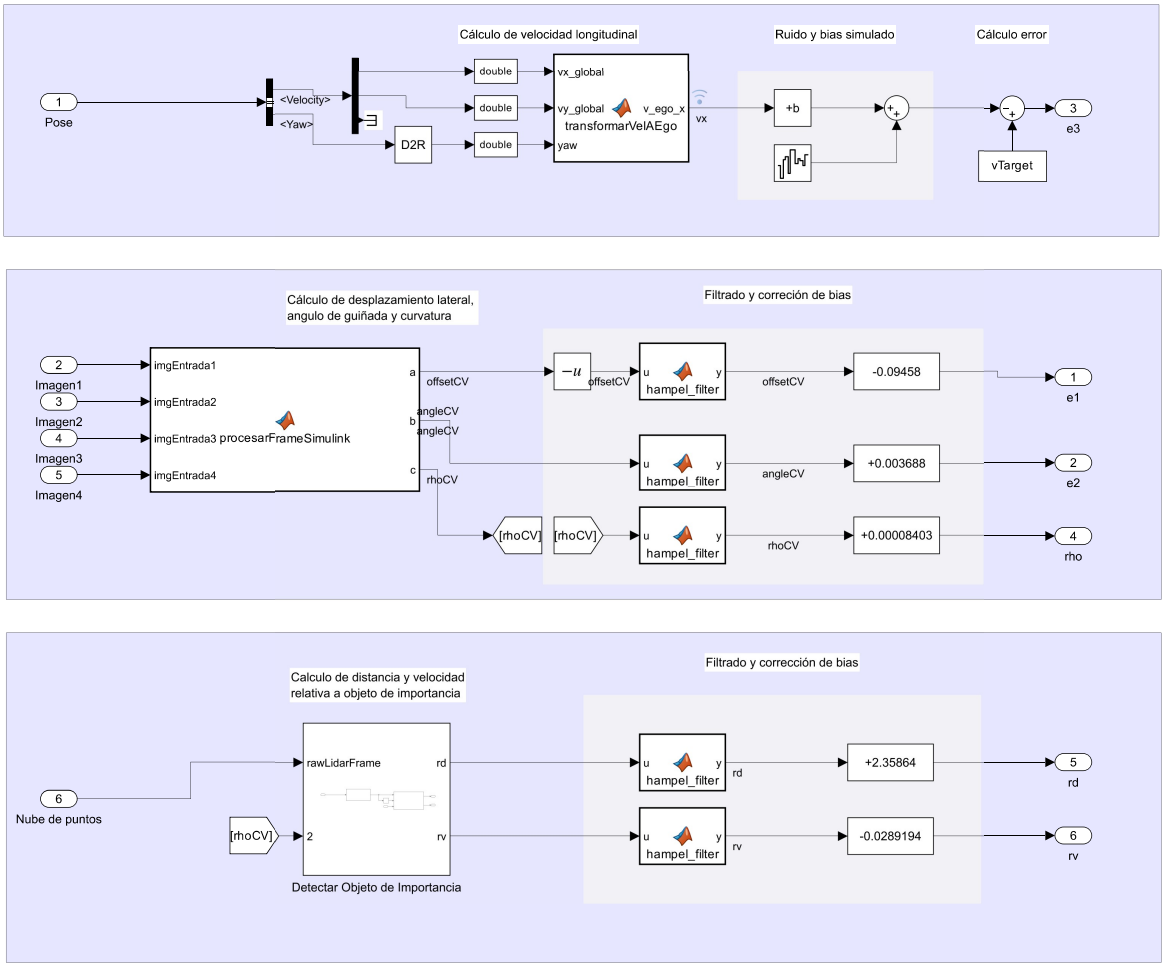


Figura B.19: Diagrama de conexiones del procesamiento de datos.

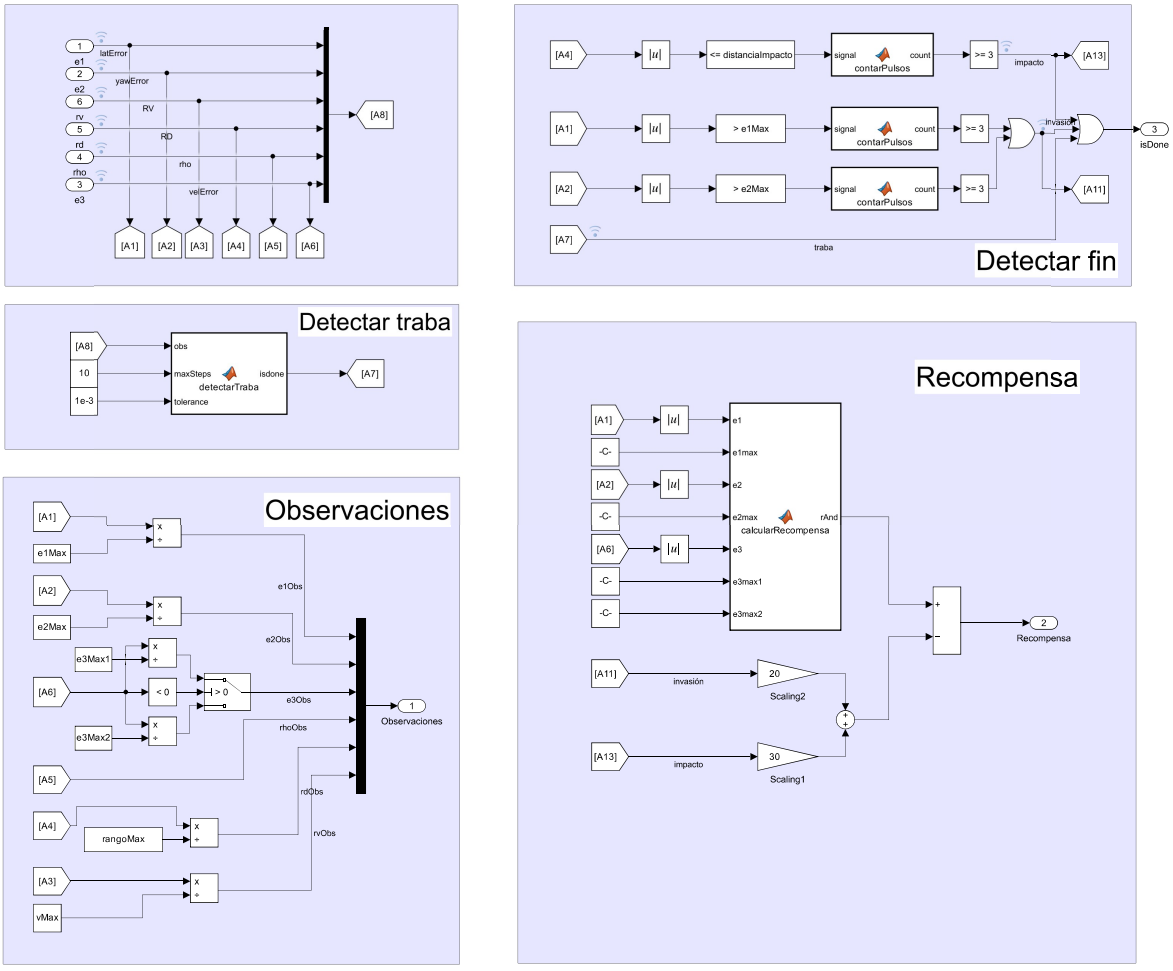


Figura B.20: Diagrama de conexiones de las señales del agente PPO previo.

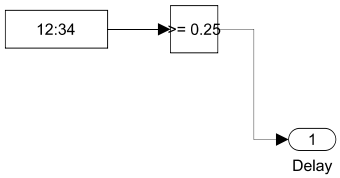


Figura B.21: Diagrama de conexiones del delay para el agente PPO previo.

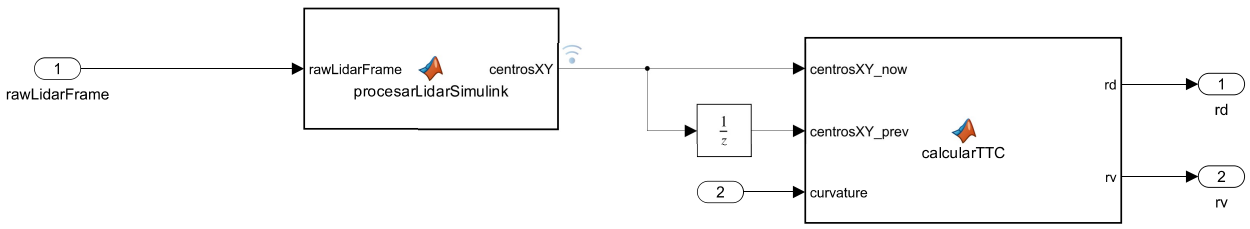


Figura B.22: Diagrama de conexiones para detectar el objeto de importancia.

```
% Calcular estad sticos robustos
mediana_local = median(window);
MAD = median(abs(window - mediana_local)) + eps; % Mediana de
    diferencias absolutas (no 0)

% Definir umbral adaptativo
threshold = nSigma * 1.4826 * MAD;  % 1.4826 ajusta MAD a std normal

% Detecci n de outlier
if abs(u - mediana_local) > threshold
    y = mediana_local;  % Corrige usando la mediana
else
    y = u;              % Deja pasar la se al sin cambios
end
end
```

Listing B.31: hampel_filter.m