EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Conducción Autónoma de un Vehículo Simulado mediante un Modelo de Red Neuronal Convolucional Recurrente

presentada por

Ing. Jesús Antonio Luna Álvarez

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dr. Dante Mújica Vargas

Cuernavaca, Morelos, México. Enero de 2020.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Mor., 19/diciembre/2019

OFICIO No. DCC/174/2019
Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

C. DR. GERARDO VICENTE GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del **C. Ing. Jesús Antonio Luna Álvarez**, con número de control M18CE009, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado **"Conducción autónoma de un vehículo simulado mediante un modelo de Red Neuronal Convolucional Recurrente"** y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

Dr. Dante Mújica Vargas
Doctor en Comunicaciones y Electrónica
09131756
Director de tesis

M.C. Gerardo Reyes Salgado
Maestro en Ciencias de la Computación
2493370
Revisor 1

Dr. Manuel Mejía Lavalle
Doctor en Ciencias Computacionales
8342472
Revisor2

C.c.p. Depto. Servicios Escolares.
Expediente / Estudiante
JGGS/lmz

"2019, Año del Caudillo del Sur, Emiliano Zapata"

Cuernavaca, Mor., 08/ENERO/2020
No. de Oficio: SAC/002/2020
Asunto: Autorización de impresión de Tesis

ING. JESÚS ANTONIO LUNA ÁLVAREZ
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Conducción autónoma de un vehículo simulado mediante un modelo de Red Neuronal Convolucional Recurrente"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Conocimiento y tecnología al servicio de México"

SEP TecNM
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

DR. GERARDO VICENTE GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO

C.p. M.E. Guadalupe Garrido Rivera.- Jefa del Departamento de Servicios Escolares.
Expediente

GVGR/ego

Dedicatoria

A mis padres que fueron principal apoyo.
A mi esposa quien compartió dedicación y esfuerzo.
A mi hijo que es mi motivación para seguir.
Sigo agradeciendo con hechos.

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología CONACYT por el apoyo económico brindado durante mis estudios de maestría. Al Tecnológico Nacional de México / Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET por brindar las instalaciones y permitirme realizar los estudios de Maestría en Ciencias.

De forma especial agradezco a el Dr. Dante Mújica Vargas, por dirigir mi investigación y especialmente por el apoyo brindado durante mi formación. De igual manera agradezco a mi comité revisor conformado por el Dr. Manuel Mejía Lavalle y el Dr. Gerardo Reyes Salgado por su disponibilidad en la revisión de mi investigación.

Resumen

En este trabajo se propone un modelo de Red Neuronal Híbrido compuesto por capas de Convolución y Recurrentes para realizar el control de un vehículo simulado en el Sistema Operativo Robótico. El vehículo en cuestión es datado de sensores visuales y espaciales que permiten la percepción del entorno, esta información es procesada por el modelo neuronal para extraer información visual del camino y transformarla en ángulos de giro del volante.

El modelo en cuestión se deriva de un modelo secuencial llamado Chauffeur utilizado para evaluar la correcta predicción de dirección de un vehículo en condiciones climáticas no favorables. El modelo propuesto cambia el bloque convolucional del Chauffeur por un bloque llamado convolución distribuida en el tiempo, que toma como entrada secuencias de imágenes en lugar de una sola. Las secuencias de entrada se conforman de imágenes con canal de profundidad creadas a partir de la fusión de sensores visuales y espaciales. Además, realiza el ajuste de las capas recurrentes a partir de pesos sinápticos autodirigidos y reduce la información para emitir el control del vehículo a partir de una neurona con función de activación Tangente Hiperbólica.

La propuesta fue comparada con su forma base para demostrar la mejoría, desglosándose la evaluación en la tarea de clasificación y autonomía del vehículo utilizando métricas reportadas en la literatura y proponiendo nuevas formas de medición. También se comparó con el modelo neuronal profundo más citado de la literatura, el cual es propuesto por investigadores de la empresa NVIDIA. Se realizaron entrenamientos equitativos usando la base de datos propuesta creada a partir de fusión de sensores, adaptando la estructura tensorial para que cada modelo reciba solamente los datos para los que fueron diseñados. De igual manera, los experimentos se realizaron en la simulación diseñada con cuatro tipo de escenarios distintos. Los resultados muestran que el modelo propuesto otorga al vehículo una mejora media en la autonomía de 11.4% con respecto a los métodos de comparación en escenarios libres, y que el uso del canal de profundidad mejora en 8.9% la autonomía en escenarios con obstáculos.

Abstract

This document proposes a Hybrid Neural Network model composed of Convolution and Recurring layers to control a simulated vehicle in the Robotic Operating System. The vehicle in question is dated to visual and spatial sensors that allow the perception of the environment, this information is processed by the neuronal model to extract visual information from the road and transform it into steering angles.

The model in question is derived from a sequential model called Chauffeur used to evaluate the correct direction prediction of a vehicle in unfavorable weather conditions. The proposed model changes the convolutional block of the Chauffeur to a block called time-distributed convolution, which takes as input sequences of images instead of just one. The input sequences are made up of images with depth channels created from the fusion of visual and spatial sensors. In addition, it adjusts the recurrent layers from self-directed synaptic weights and reduces the information to emit control of the vehicle from a neuron with Hyperbolic Tangent activation function.

The proposal was compared with its base form to demonstrate improvement, disaggregating the evaluation in the task of classification and autonomy of the vehicle using metrics reported in the literature and proposing new forms of measurement. It was also compared with the most cited deep neuronal model in the literature, which is proposed by researchers at NVIDIA. Equitable training was done using the proposed database created from sensor fusion, adapting the tensor structure so that each model receives only the data for which they were designed. Similarly, the experiments were performed in the simulation designed with four different types of scenarios. The results show that the proposed model gives the vehicle an average improvement in the autonomy of 11.4% with respect to the comparison methods in free scenarios, and that the use of the depth channel improves in 8.9% the autonomy in scenarios with obstacles.

Índice General

Resumen I

Nomenclatura VIII

1. Introducción **1**
1.1. Descripción del Problema 2
1.1.1. Delimitación del problema 3
1.1.2. Complejidad del problema 3
1.2. Objetivos 4
1.2.1. Objetivo general 4
1.2.2. Objetivos específicos 4
1.3. Alcances y Limitaciones 4
1.3.1. Alcances 4
1.3.2. Limitaciones 5
1.4. Justificación 5
1.5. Organización de la tesis 6

2. Marco Teórico **7**
2.1. Vehículo Autónomo 7
2.2. Aprendizaje Profundo 9
2.2.1. Redes Neuronales Convolucionales 10
2.2.2. Redes Neuronales Recurrentes 12

3. Estado del Arte **17**
3.1. Antecedentes 17
3.2. Estado del Arte 19

4. Metodología **45**
4.1. Simulación del vehículo autónomo 45
4.1.1. Vehículo y sensores 45
4.1.2. Base de datos propuesta 47
4.2. Modelo neuronal convolucional recurrente RGB-D 49
4.2.1. Modelo Chauffeur 50
4.2.2. Modelo Chauffeur mejorado 54

4.3. Integración del Sistema 57

5. Experimentación y resultados **60**
5.1. Entorno de desarrollo 60
5.2. Datos de entrenamiento y prueba 61
5.3. Métodos de comparación 61
5.4. Experimentos 63
5.5. Métricas 66
5.5.1. Calidad de la fusión 67
5.5.2. Métricas de clasificación 67
5.5.3. Métricas de autonomía 68
5.6. Resultados 70
5.6.1. Calidad de fusión 70
5.6.2. Clasificación 71
5.6.3. Autonomía 74

6. Conclusiones **79**
6.1. Objetivos y alcances logrados 79
6.2. Resultados del trabajo 80
6.2.1. Productos 80
6.2.2. Aportaciones 81
6.2.3. Conclusiones 82
6.2.4. Trabajo futuro 83

A. Documentos correspondientes a participaciones **90**

Índice de Figuras

2.1. Operación de convolución (Goodfellow et al., 2016). 11
2.2. Esquema básico de una RNN (Bianchi et al., 2017). 13
2.3. Célula LSTM (Greff et al., 2017). 16

3.1. Modelo propuesto (Pérez, 2001). 19
3.2. Esquema de entrenamiento (Bojarski et al., 2016). 20
3.3. Modelo PilotNet (Bojarski et al., 2017). 22
3.4. Modelo de red neuronal utilizado (Alexeev et al., 2018). 22
3.5. Modelo CNN DAVE-2 (Bechtel et al., 2017). 25
3.6. Representación de la configuración espacial utilizada en el robot móvil (De Silva et al., 2018). 34
3.7. Descripción del modelo Point Cloud-CNN (Du et al., 2017). 36

4.1. Sensores utilizados en la simulación probados en el escenario propuesto. 46
4.2. Ubicación de los sensores (De Silva et al., 2018). 46
4.3. Sensores utilizados en la simulación. 47
4.4. Imagen de profundidad obtenida al transformar coordenadas. 48
4.5. Modelo Neuronal Chauffeur (Tian et al., 2017). 50
4.6. Modelo de la neurona LSTM (Venna et al., 2018). 52
4.7. Modelo Neuronal Chauffeur distribuido en el tiempo propuesto. . . 55
4.8. Diagrama de clases del sistema propuesto en fase de entrenamiento. 57
4.9. Diagrama de clases del sistema propuesto en fase de pruebas. 58

5.1. Modelo Pilotnet (Bojarski et al., 2016). 62
5.2. Ejemplo de caminos rectos. 64
5.3. Ejemplo de camino con curva. 64
5.4. Caminos con obstrucción parcial. 65
5.5. Escenario con obstrucción total en un camino. 65
5.6. Prueba final en el escenario *VRC Task 1* (Sardinha, 2017). 66
5.7. Distribución del error absoluto medio en la fusión de sensores. . . . 70
5.8. Comparación cualitativa de la imagen a color y el canal de profundidad. 71
5.9. Función de pérdida en entrenamiento y validación. 72
5.10. Métrica *logcosh* en entrenamiento y validación. 72

5.11. Distancia coseno en etapa de entrenamiento y validación. 73
5.12. Resumen de los resultados de autonomía. 78

A.1. Constancia como ponente en la Escuela de Inteligencia Artificial y Robótica 2018, UTEZ. 90
A.2. Constancia como ponente en el 1er coloquio .Artificial Intelligence", Tecnológico de Iguala. 91
A.3. Constancia como concursante en concurso de posters, ICMEAE 2018. 92
A.4. Constancia como coautor de artículo publicado en la Segunda Jornada de Ciencia y Tecnología Aplicada, CENIDET 2018. 93
A.5. Constancia como autor en la Escuela de Inteligencia Artificial y Robótica 2019, UTEZ. 94
A.6. Constancia como autor la Tercera Jornada de Ciencia y Tecnología Aplicada, CENIDET 2019. 95
A.7. Constancia de 3^{er} lugar del artículo presentado en la Tercera Jornada de Ciencia y Tecnología Aplicada, CENIDET 2019. 96
A.8. Constancia como coautor de artículo publicado en el congreso MCPR 2019. 97
A.9. Constancia como autor de artículo publicado en el congreso CORE 2019, CIC-IPN. 97
A.10. Artículo sometido a la revista *Applied Soft Computing, Elsevier*. 98

Índice de Tablas

2.1. Niveles de conducción automatizada (Litman, 2017). 8

3.1. Discusión del Estado del Arte. 37

5.1. Resultados en camino libre. 75
5.2. Resultados en camino con obstrucción parcial. 75
5.3. Resultados en camino con obstrucción total. 75
5.4. Resultados en escenario *VRC Task 1*. 76

6.1. Objetivos y alcances realizados. 79

Nomenclatura

u	Ancho de una imagen transformada a partir de información espacial.
v	Largo de una imagen transformada a partir de información espacial.
f	Distancia focal obtenida de la calibración de la cámara.
X	Tensor de entrada en una red profunda de dimensión $w \times h \times d \times t$.
w	Ancho del tensor X.
h	Largo del tensor X.
d	Profundidad del tensor X (interpretado como canales de color de una imagen).
t	Tiempo abstracto utilizado en las capas recurrentes (*timestep*).
$\mathcal{C}$	Capa convolucional.
$\mathcal{R}$	Capa recurrente LSTM.
$\mathcal{F}$	Capa de reducción de dimensionalidad (*Flatten*).
$\mathcal{M}$	Capa perceptrón multicapa.
h	Capa oculta de una red recurrente.
w	Parámetro entrenable de una neurona (peso o *kernel*).
b	Parámetro de ajuste bias.
Δw	Incremento o decremento de parámetros durante el entrenamiento.
k	Número de filtros en una capa convolucional
$\mathcal{C}^k$	k-ésimo mapa de características obtenido de la convolución.
R	Matriz de recurrencia.
y	Salida esperada de la red.
y^p	Salida obtenida de la red.
α	Tasa de aprendizaje (*learning rate*).
∇f	Gradiente de error obtenido en el entrenamiento.
$f(\cdot)$	Función rectificador lineal.
$\sigma(\cdot)$	Función no lineal sigmoide.

Capítulo 1

En este Capítulo se abordan los aspectos fundamentales de este trabajo de investigación. Por lo cual se aborda en cada Sección los aspectos fundamentales que servirán de soporte, tales como: la descripción, delimitación y complejidad del problema, los objetivos, alcances y limitaciones, así como la justificación.

Introducción

Las actividades comunes que realiza el conductor promedio son la detección y evasión de peatones (Zhang et al., 2018; Li et al., 2019), reconocimiento de señalamientos de transito (de Oliveira et al., 2018; Pei et al., 2018), cambios de carril e intersecciones (Chen et al., 2017a), entre otras. Sin embargo, una de las tareas más importantes es el control de dirección del vehículo en movimiento, ésta involucra la evasión de obstáculos en el camino (Ramos et al., 2017) y control del carril (Maqueda et al., 2018), ya que de este sistema depende la seguridad del vehículo al evitar colisiones y descarrilamientos.

La autonomía de los vehículos puede medirse en 5 niveles, iniciando desde un automóvil totalmente manual hasta uno que pueda conducirse por sí mismo en todos los escenarios posibles (Litman, 2017). En la actualidad un vehículo que cubre los requerimientos de autonomía nivel 3 puede ser considerado un vehículo autónomo. Para que un automóvil común pueda ser un vehículo autónomo se requiere de dos componentes indispensables: un módulo de percepción dado por múltiples sensores que registran las variables físicas del entorno, y un módulo de control que apoyado en el módulo de percepción, toma las decisiones para mover el vehículo.

Enfocado en el control de dirección se han publicado diversas propuestas, sistemas de control basados en lógica difusa (Olson et al., 2017), reforzamiento de aprendizaje (Sallab et al., 2017; Nageshrao et al., 2019), pero las publicaciones que resaltan son aquellas soluciones basadas en Redes Neuronales de Aprendizaje Profundo (Bojarski et al., 2016, 2017; Chen et al., 2017a). Las publicaciones antes citadas tienen en común el uso de cámaras monoculares para obtener imágenes de entra-

da a los bloques convolucionales de las redes presentadas. Aunque se muestran resultados que cubren el Nivel 3 de autonomía en vehículos (Rödel et al., 2014), la autonomía puede ser mejorada con el uso de otros sensores que respalden la parte de visión, específicamente una nube de puntos 3D que de información de distancia a la que se ubican los objetos visibles en las imágenes de entrada (Chen et al., 2017b).

Para mejorar la autonomía se propone un modelo híbrido de Red Neuronal Convolucional Recurrente para imágenes RGB-D obtenidas de la fusión una cámara monocular y un sensor *Light Detection and Ranging* (LiDAR). Aunque existen sistemas que utilizan cámaras estéreo para medir distancias (Salman et al., 2017), el uso de sensores como el LiDAR permite la medición de distancias con mayor exactitud, así como la posibilidad de ser fusionado con imágenes (De Silva et al., 2018; Li, 2015) para crear el canal de profundidad. El modelo diseñado se evaluó en una variedad de entornos simulados aproximados a las características de caminos reales, además se creó un vehículo dotado de sensores en el Sistema Operativo Robótico (ROS).

1.1. Descripción del Problema

En general la autoconducción es referida a mantener al vehículo en la zona adecuada del carril usando como referencias los bordes del camino y las líneas divisorias de cada carril; sin embargo, la autonomía de los vehículos se ha tratado como un problema reactivo, es decir, que actúa al momento utilizando técnicas de control que toman como entrada solamente datos de la escena presente, sin importar los datos en tiempos anteriores, como la solución propuesta por (Bojarski et al., 2016). Sin embargo, en este problema existe una dependencia en el tiempo debido a que la actividad de conducción depende en gran medida de las acciones que realiza el conductor para decidir las acciones posteriores.

El problema incrementa su complejidad cuando el vehículo se desplaza en un entorno con obstáculos, siendo estos objetos en el camino, peatones u otros vehículos. Para prevenir la colisión con alguno de estos objetos es necesario dotar al vehículo de la percepción visual y espacial mediante sensores especializados, permitiendo hacer la detección y evasión autónoma, permitiendo posteriormente retomar el camino en la zona adecuada.

Hipótesis: un modelo de Red Neuronal Híbrida Convolucional y Recurrente permite realizar la autoconducción con mayor precisión en comparación a un enfoque reactivo clásico. El bloque convolucional de la red permite identificar la zona adecuada del carril y al mismo tiempo la detección de objetos con la información visual y espacial, mientras que el bloque recurrente permite aprender de datos secuenciales en instantes anteriores para realizar una mejor predicción a futuro.

1.1.1. Delimitación del problema

Este trabajo está enfocado en el control de dirección de un vehículo simulado utilizando un modelo de Red Neuronal Convolucional Recurrente, el cual permite trabajar con breves secuencias de imágenes obtenidas del camino fusionadas con información espacial de sensores especializados. Utilizando estos datos para entrenamientos del modelo neuronal, en una etapa posterior, este es capaz de conducir de forma adecuada en caminos que no son conocidos. Dicho modelo permite al vehículo realizar las acciones de frenar, acelerar y girar el volante para mantenerse dentro del carril de forma segura.

1.1.2. Complejidad del problema

Las complejidad del problema radica en los siguientes puntos:

- Crear un modelo neuronal híbrido combinando capas de convolución para extracción de información y capas recurrentes que aprendan de datos distribuidos en el tiempo.
- Modificar el bloque convolucional para trabajar con secuencias de imágenes, creando un bloque convolucional distribuido en el tiempo.
- Diseñar e implementar la fusión de información visual y espacial para crear imágenes con canal de profundidad.
- Adaptar el bloque convolucional distribuido en el tiempo para procesar imágenes con canal de profundidad.
- Realizar el correcto entrenamiento del modelo, sintonizando parámetros de entrenamiento y balanceo de información, para operar en caminos que no son conocidos.

- Crear un entorno simulado que se aproxime a las características que tiene un camino real.

1.2. Objetivos

1.2.1. Objetivo general

Dotar de autonomía mediante un modelo de Red Neuronal Convolucional Recurrente de Aprendizaje Profundo a un vehículo simulado en el Sistema Operativo Robótico.

1.2.2. Objetivos específicos

- Desarrollar la simulación del vehículo en ROS y dotarlo de sensores disponibles.
- Estudiar y comprender las Redes Neuronales Convolucionales y sus aplicaciones en detección y clasificación visual.
- Estudiar y comprender las Redes Neuronales Recurrentes basadas en Aprendizaje Profundo, sus variantes y sus aplicaciones en el reconocimiento de objetos y clasificación.
- Utilizar bancos de imágenes y datos de sensores capturados durante la conducción de un automóvil real y/o simulado para el entrenamiento del modelo.
- Integrar el modelo a la simulación del vehículo para permitir que éste realice las acciones control de dirección y velocidad del vehículo.
- Evaluar el desempeño del modelo neuronal diseñado en términos de clasificación y autonomía del vehículo.

1.3. Alcances y Limitaciones

1.3.1. Alcances

- Desarrollar la simulación del vehículo en ROS.
- Definir la arquitectura del modelo.

- Entrenar el modelo para realizar las acciones de viraje y control de velocidad.
- Experimentar con el vehículo en un ambiente simulado estático.
- Evaluar la autonomía del vehículo con base en las métricas definidas en la literatura.

1.3.2. Limitaciones

- El vehículo contará sólo con los sensores simulados que se encuentran disponibles en ROS.
- La programación de los modelos se realizarán utilizando el *framework* Tensorflow y librerías compatibles.
- Las pruebas se harán en un ambiente simulado predefinido.
- Se regulará la velocidad y dirección con base a los obstáculos y formas del camino, no a señalamientos.
- Factores del clima que afecten la claridad de las imágenes y datos capturados, así como los efectos físicos en el automóvil no serán contemplados.
- Los tiempos de la simulación de conducción son variables y dependientes a la distancia aproximada a una calle de 1 km aproximadamente.

1.4. Justificación

El control de un vehículo autoconducido ha sido abordado en la última década como el seguimiento de una línea delimitada por sensores GPS y técnicas de visión para la detección de líneas divisorias de carril. En años recientes la capacidad del *hardware* ha permitido el uso de las Redes de Aprendizaje Profundo para la detección automática del carril mediante información visual, incluso en ausencia de las líneas de carril. La capacidad de generalizar de estas permiten controlar el vehículo en escenarios conocidos y desconocidos, sin embargo este enfoque solo permite realizar control reactivo.

La conducción en caminos con curvas e intersecciones, así como las variaciones visuales entre cada instante de tiempo pueden afectar al módulo de control haciendo que realice cambios de dirección y velocidad muy grandes, afectando la calidad de la

conducción. En este enfoque se propone una solución a este problema empleando un modelo que procesa y memoriza información de instantes anteriores para realizar un control más preciso ante variaciones de los datos.

1.5. Organización de la tesis

La presente tesis se organiza de la siguiente manera: En el Capítulo 2 se describen los conceptos teóricos necesarios para comprender el funcionamiento de las Redes Neuronales de Aprendizaje Profundo en el dominio del control de un vehículo auto conducido.

En el Capítulo 3 se presenta un análisis de los trabajos más recientes relacionados al tema de los vehículos autoconducidos, tecnologías utilizadas, técnicas de obtención de datos y de control, así como un resumen que incluye la utilidad de cada uno para este trabajo de investigación.

En el Capítulo 4 se describe a detalle el método propuesto para controlar el vehículo. Como contribución principal, basándose en un modelo híbrido de Red Convolucional y Recurrente, se propone un modelo distribuido en el tiempo que trabaja con imágenes con canal de profundidad que realiza el control del vehículo.

En el Capítulo 5 se describen los experimentos y condiciones en las que se realizaron. El modelo propuesto es evaluado en las tareas de clasificación y autonomía del vehículo utilizando métricas obtenidas de la literatura y dos métodos de medición propuestos en el dominio de autonomía.

Finalmente, en el Capítulo 6, se exponen las conclusiones obtenidas, recomendaciones para trabajos futuros, así como un análisis de los objetivos completados y de los alcances planteados en esta investigación.

Capítulo 2

Marco Teórico

En la presente Capítulo se presentan los conceptos básicos que apoyan al entendimiento de los temas relacionados, de forma específica y complementaria, con la presente investigación.

2.1. Vehículo Autónomo

Los Vehículos Autónomos son automóviles capaces de detectar su entorno y navegar sin intervención humana evitando obstáculos, al tiempo que entienden con precisión la interpretación semántica compleja de las escenas y las actividades dinámicas del entorno. Hay dos definiciones que describen este tipo de vehículos: autónomo (que es más común) y automatizado (que es más preciso) (Novitskiy and Boyarskaya, 2016).

- *Autónomo*: actúa solo o independientemente.
- *Automatizado*: connota control u operación por una máquina.

En la actualidad, la mayoría de los vehículos a los que se les llama autónomos tienen a una persona al volante, también utilizan conexión con la nube u otros vehículos similares para obtener información de estos, por lo que no son totalmente independientes del usuario.

Niveles de autonomía

En 2014 la Sociedad de Ingenieros de Automóviles (*Society of Automobile Engineers*, SAE) definió cinco niveles de conducción autónoma. Los niveles 1-3 requieren un conductor con licencia pero los niveles 4 y 5 permiten la operación sin conductor, como se puede observar en la Tabla 2.1.

Tabla 2.1: Niveles de conducción automatizada (Litman, 2017).

Nivel	Nombre	Descripción
El conductor interviene en la conducción		
0	Sin automatización en la conducción	Todas las acciones son realizadas en su totalidad por el conductor.
1	Asistencia en la conducción	Este nivel está pensado para que el conductor pueda tener una conducción más cómoda, es una ayuda que mejora la seguridad al volante. Por ejemplo: Mantenimiento del carril, controles de velocidad adaptativos.
2	Automatización parcial	El vehículo cuenta con control de movimiento tanto longitudinal como lateral, aunque no tiene detección y respuesta ante objetos.
3	Automatización condicionada	Tiene sistemas de automatización en lo referente al control de movimiento longitudinal y lateral; detección y respuesta ante objetos. El vehículo puede decidir cuándo cambiar de carril, frenar para evitar colisionar, etc., pero el factor humano seguirá siendo clave.
No se precisa conductor		
4	Automatización elevada	No se precisa de la intervención humana en ningún momento ya que el vehículo es quién controla el tráfico y las condiciones del entorno, definirá la ruta o alternativas y responde ante cualquier situación.
5	Automatización completa	A este nivel la figura del conductor no existe, el pasajero indica el destino y el vehículo se pondrá en marcha. Cuentan con un sistema de automatización que en caso de fallo se respalda con otro sistema, por lo que él mismo solucionará cualquier imprevisto.

Sensores y equipamiento

Para que un vehículo sea capaz de percibir su entorno y entender las condiciones del entorno en el que se encuentra se necesita principalmente una serie de sensores que proveen de información necesaria al sistema para la toma de decisiones. En algunos trabajos realizados en el estado del arte, se evaluó el desempeño de modelos y técnicas de aprendizaje profundo solamente utilizando las imágenes provistas por cámaras, mientras que algunos otros, por el contrario utilizan combinaciones de cámaras, sensores infrarrojos, sensor LIDAR, entre otros.

Para los vehículos equipados con sensores, cada uno de éstos cumple una función en especifico que aporta información al sistema, entre otros se encuentran:

- LIDAR le da al vehículo la capacidad de monitorear 360 grados de su entorno para que pueda detectar objetos al frente, al lado y detrás de sí mismo al

mismo tiempo. El láser también ayuda al vehículo a determinar su ubicación en el entorno real.

- El sensor de posición, ubicado en el cubo de la rueda, detecta las rotaciones hechas por las ruedas del automóvil para ayudar al vehículo a comprender su posición en el entorno.
- El radar detecta vehículos muy adelante y mide su velocidad para que el automóvil pueda reducir la velocidad o acelerar con otros vehículos en la carretera.

La información de los sensores es verificada y procesada por el software para que los diferentes objetos alrededor del vehículo puedan ser detectados y diferenciados con precisión, y luego se pueden tomar decisiones de manejo seguras basadas en toda la información recibida.

2.2. Aprendizaje Profundo

El Aprendizaje Profundo (*Deep Learning*) es un paradigma de Redes Neuronales Artificiales compuesto por una variedad de algoritmos de aprendizaje automático expresados como capas neuronales. El conjunto de capas estructuradas de forma jerárquica es llamado modelo de red neuronal, éste puede componerse de combinaciones de capas especificas que realizan extracción, transformación o memorización de datos, según sea necesario para la aplicación del modelo.

El Aprendizaje Profundo tiene ventajas frente a otros algoritmos de aprendizaje automático, de los cuales destacan la robustez ante variaciones naturales en los datos, la capacidad de generalización que permite al un mismo modelo el ser utilizado para otras aplicaciones o tipos de datos, y la escalabilidad, puesto que el rendimiento puede mejorar con más datos de entrenamiento (Chollet, 2017).

Algunos de las redes neuronales más populares se encuentran (Goodfellow et al., 2016):

- *Redes neuronales profundas de propagación hacia adelante*: también llamadas perceptrón multicapa profundo o capas densas, sigue el mismo objetivo que su predecesor, con la diferencia que este tipo de red neuronal consta de un

número mayor de neuronas y capas ocultas que mejoran la capacidad de clasificación y el procesamiento de gran cantidad de datos.

- *Redes Neuronales Convolucionales*: se utilizan principalmente para el procesamiento en imágenes, en tareas de segmentación, extracción de características, detección y clasificación de objetos.
- *Autoencoders*: Un autoencoder es una red neuronal que está entrenada para intentar copiar su entrada a su salida. Internamente, tiene una capa oculta h que describe el código utilizado para representar la entrada.
- *Redes Neuronales Recurrentes*: derivadas del perceptrón multicapa, las RNN son utilizadas para procesar y aprender datos secuenciales.

2.2.1. Redes Neuronales Convolucionales

Este tipo de Red Neuronal tiene grandes ventajas de automatización en la tarea de extraer información de imágenes. En este paradigma, no se aplica ningún filtro de procesamiento de imagen conocido, pero a través del entrenamiento, aprende los filtros de procesamiento y extracción. En general, la primera capa convolucional aprende a detectar bordes, mientras que la segunda puede aprender a detectar formas más complejas que se pueden formar combinando diferentes bordes, como círculos y rectángulos. La tercera capa y más allá aprende características mucho más complicadas basadas en las características generadas en la capa anterior (Pattanayak, 2016).

Las Redes Neuronales convolucionales (CNN) se basan en la convolución de señales en dos dimensiones y en la detección de características basadas en filtros (llamadas *kernel*) que aprenden a través del entrenamiento. Para extraer la información, el *kernels* realiza la operación de convolución 2D, expresada en la Ecuación 2.2, a la imagen de entrada. Donde N y M son las dimensiones de la imagen, n_1 y n_2 representan los índices de columna y fila del píxel que se procesa, k_1 y k_2 los índices del filtro (LeCun et al., 2015).

$$y(n_1,n_2) = \sum_{k_2=0}^{N-1} \sum_{k_1=0}^{M-1} x(k_1,k_2)h(n_1-k_1,n_2-k_2) \tag{2.1}$$

$$0 \leq n_1 \leq N-1, 0 \leq n_2 \leq M-1. \tag{2.2}$$

Para entrenar las capas convolucionales, se utiliza la técnica de retropropagación del error a través de las capas convolucionales. Esto es muy similar a como se realiza en una red de Perceptrón Multicapa, la diferencia es que las conexiones de peso están dispersas, ya que las diferentes áreas de entrada comparten los mismos pesos para crear un mapa de características de salida, como se muestra en la Figura 2.1. En general, el mapa de características se obtiene a través de la función expresada en (2.3). Donde w representa los pesos del filtro y a un primer mapa de características obtenido. Por otro lado, los pesos del filtro se ajustan calculando el gradiente descendente que se expresa en (2.4). Donde L representa la función de error.

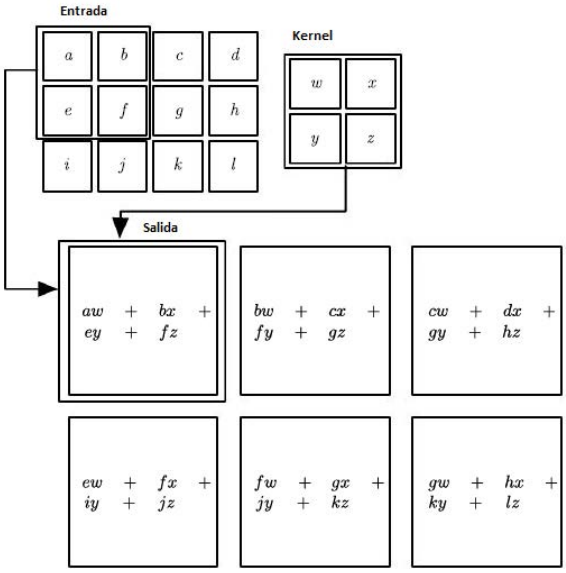


Figura 2.1: Operación de convolución (Goodfellow et al., 2016).

$$S_{ij} = \sum_{n=1}^{2} \sum_{m=1}^{2} w_{(3-m)(3-n)} \cdot a_{(i-1+m)(j-1+m)} \tag{2.3}$$

$$\frac{\partial L}{\partial w_{ij}} = \sum_{j=1}^{2} \sum_{j=1}^{2} \frac{\partial L}{S_{ij}} \frac{\partial S_{ij}}{\partial w_{ij}} \tag{2.4}$$

En la generalidad de las Redes Neuronales Convolucionales, hay capas de reducción de datos. Las capas de Agrupación Máxima (*max-pooling*) están dadas por una función discriminante donde el valor máximo se selecciona en cada región de la imagen moviendo un filtro a través de ella, generando así un nuevo mapa de características de menor tamaño que el original. La función de este filtro se proporciona en (2.5), donde M y N representan el largo y ancho del filtro.

$$y_{i,j} = \text{máx}(\sum_{m=0}^{M}\sum_{n=0}^{N} y_{i+m,j+n}) \tag{2.5}$$

Por otro lado, existen capas de Agrupación Promedio, que sigue el mismo principio que la anterior, con la diferencia de que en esto, el filtro toma el+ promedio de la región de acuerdo con (2.6) (Scherer et al., 2010).

$$y_{i,j} = \frac{\sum_{m=0}^{M}\sum_{n=0}^{N} y_{i+m,j+n}}{M \times N} \tag{2.6}$$

2.2.2. Redes Neuronales Recurrentes

Las redes neuronales recurrentes (*Recurrent Neural Networks*, RNN) están diseñadas para utilizar y aprender de la información secuencial (Bianchi et al., 2017). Estas han sido de gran utilidad en la tarea del procesamiento del lenguaje natural debido a la dependencia secuencial de las palabras en cualquier idioma.

En su forma más general, un RNN puede verse como un gráfico ponderado, dirigido y cíclico que contiene tres tipos diferentes de nodos, que son los nodos de entrada, oculto y de salida (Zhang et al., 2016). Los nodos de entrada no tienen conexiones entrantes, los nodos de salida no tienen conexiones salientes, los nodos ocultos tienen ambos. Existen tres tipos de tareas esenciales que se pueden realizar con este tipo de redes (Bonet Cruz et al., 2007):

1. Reconocimiento de secuencias: Se produce un patrón de salida particular cuando se especifica una secuencia de entrada.

2. Reproducción de secuencias: La red debe ser capaz de generar el resto de Una secuencia cuando ve parte de ella.

3. Asociación temporal: En este caso una secuencia de salida particular se debe producir en respuesta a una secuencia de entrada especifica

Como es observado en la Figura 2.2, un esquema simple de un modelo de RNN consta, al igual que los modelos clásicos, de un capa de entrada, una de salida y n capas ocultas. Lo particular de este tipo de Red Neuronal es el parámetro añadido t el cual representa el tiempo, siendo x_{t-1}, x_t y x_{t+1} entradas en distintos instantes de tiempo.

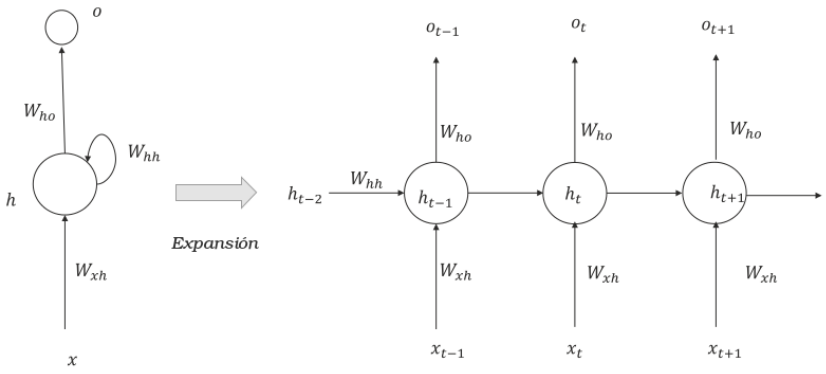


Figura 2.2: Esquema básico de una RNN (Bianchi et al., 2017).

La retropropagación para las redes neuronales recurrentes es la misma que para las redes neuronales de retroalimentación, con la única diferencia de que el gradiente es la suma del gradiente con respecto a la pérdida en cada paso. El procedimiento de cálculo del gradiente consiste en repetir dos pasos básicos hasta alcanzar la convergencia:

1. La función de pérdida L_k se evalúa en el RNN actualizando los pesos W_k, cuando se procesa un conjunto de datos de entrada X_k. (Nota: el índice k identifica sus valores en época k).

2. El gradiente $\frac{\partial L_k}{\partial W_k}$ se propaga a través de la red para actualizar sus parámetros.

La función de pérdida se puede define en la ecuación 2.7.

$$L_k = E(X_k, Y_k^*; W_k) + R_\lambda(W_k) \tag{2.7}$$

Donde E es una función que evalúa el error de predicción de la red cuando se alimenta con entradas en X_k, con respecto a una respuesta deseada Y_k^*. R_λ es una función de regularización que depende de un hiperparámetro λ, que pondera la contribución de la regularización en la pérdida total. Una de las funciones de error para E más mencionada en la literatura es la función del Error Cuadrático Medio (*Mean Square Error*, MSE) 2.8.

$$MSE(\mathcal{Y}_k, \mathcal{Y}_k^*) = \frac{1}{|\mathcal{X}_k|} \sum_{x \in X_k} (y_x - y_x^*)^2 \tag{2.8}$$

Donde $y_x \in \mathcal{Y}_k$ es la salida del RNN (actualizado con los pesos W_k) cuando se procesa la entrada $x \in \mathcal{X}_k$ y $Y_x^* \in \mathcal{Y}_x^*$ es el valor que la red debe aprender a

reproducir. El término de regularización R_λ introduce un sesgo (*bias*) que mejora las capacidades de generalización del modelo RNN, al reducir el sobreajuste en los datos de entrenamiento. Entre los parámetros de regularización mencionados en la literatura se encuentran (Bianchi et al., 2017):

$$L1 : R_\lambda(W_k) = \lambda_1 ||W_k||_1 \tag{2.9}$$

$$L2 : R_\lambda(W_k) = \lambda_2 ||W_k||_2 \tag{2.10}$$

$$R_\lambda(W_k) = \lambda_1 ||W_k||_1 + \lambda_2 ||W_k||_2 \tag{2.11}$$

donde λ_1 y λ_2 son expresadas como variables de regularización iniciadas en 0 y W_k los pesos sinápticos de la neurona.

Elman RNN ó Vanilla RNN es el modelo clásico y más básico de las redes recurrentes. Propuesto por (Elman, 1995) para aplicaciones de procesamiento del lenguaje. Su arquitectura es básicamente la de una RNN común la mostrada en la Figura 2.2 y se desglosa en el algoritmo 2.1.

Algoritmo 2.1: Entrenamiento de neurona RNN.

input: Patrones $X \in \mathbb{R}^n$, Salida esperada $Y \in \mathbb{R}$, tiempos $\tau \in \mathbb{N}$

output: Capa de salida y

$W \leftarrow \text{Random}([n \times \tau])$

while $t < \tau$ **do**

  $h_t \leftarrow \sigma(W_{hx}x + W_{hh}h_{t-1})$

  $y \leftarrow \sigma(W_o h)$

  $L_t \leftarrow E(X_t, Y_t^*; W_t) + R_\lambda(W_t)$

  $W_t \leftarrow W_t \cdot \frac{\partial L_t}{\partial W_{t-1}}$

end

return y

El Algoritmo 2.1 requiere de la entrada de datos X en forma vectorial, así como un escalar que representa la salida esperada (clase) y un número natural τ que es el total de tiempos en que la neurona se autoajusta. La inicialización de los pesos W se realiza aleatoriamente en forma de maya con respecto a cada una de las n entradas en τ tiempos. Para cada tiempo t se realiza el ajuste de la neurona h_t mediante el producto de la suma de los estados anteriores por el estado actual regularizado por una función no lineal $\sigma(\cdot)$, se calcula la salida y mediante la misma función sobre el ultimo estado de la neurona, se calcula el error de predicción L_t y se ajustan los parámetros W_t tomando en cuenta los estados anteriores.

Long Short-Term Memory (LSTM)

La arquitectura de *Long Long-term Memory* (LSTM) fue propuesta originalmente por Hochreiter y Schmidhuber en 1997 (Hochreiter and Schmidhuber, 1997) y es ampliamente utilizada hoy en día debido a su desempeño superior en el modelado preciso de dependencias a corto y largo plazo en los datos.

Mientras que una neurona de la Red Elman (ERNN) implementa una función de no linealidad $f(\cdot)$, una célula LSTM se compone de 5 componentes no lineales diferentes, que interactúan entre sí de una manera particular. El LSTM modifica el estado interno de una celda solo a través de interacciones lineales. Esto permite que la información se reproduzca de manera suave a lo largo del tiempo, con la consiguiente mejora de la capacidad de memoria de la célula. LSTM protege y controla la información en la célula a través de tres puertas, que son implementadas por un sigmoide y una multiplicación puntual. Para controlar el comportamiento de cada puerta, se entrena un conjunto de parámetros con pendiente de gradiente para resolver una tarea objetivo. Las 5 puertas de una célula LSTM se visualizan en la estructura de la Figura 2.3 y se expresan:

$$Forget\ gate : \sigma_f[t] = \sigma(\mathbf{W}_f\mathbf{x}[t] + \mathbf{R}_f y[t-1] + \mathbf{b}_f) \tag{2.12}$$
$$Candidate\ gate : \tilde{\mathbf{h}}_f[t] = g_1(\mathbf{W}_f\mathbf{x}[t] + \mathbf{R}_h y[t-1] + \mathbf{b}_h) \tag{2.13}$$
$$Update\ gate : \sigma_u[t] = \sigma(\mathbf{W}_u\mathbf{x}[t] + \mathbf{R}_u y[t-1] + \mathbf{b}_u) \tag{2.14}$$
$$Cell\ state : \mathbf{h}[t] = \sigma_u \odot \tilde{\mathbf{h}} + \sigma_f[t] \odot \mathbf{h}[t-1] \tag{2.15}$$
$$Output\ gate : \sigma_o[t] = \sigma(\mathbf{W}_o\mathbf{x}[t] + \mathbf{R}_o y[t-1] + \mathbf{b}_o) \tag{2.16}$$
$$Output : y[t] = \sigma_o[t] \odot g2(\mathbf{h}[t]) \tag{2.17}$$

donde $x[t]$ es el vector de entrada en el tiempo t. W_f, W_h, W_u y W_o son matrices de peso rectangulares, que se aplican a la entrada de la celda LSTM. R_f, R_h, R_u yR_o son matrices cuadradas que definen los pesos de las conexiones recurrentes, mientras que b_f, b_h, b_u y b_o son vectores de *bias*. La función $\sigma(\cdot)$ es sigmoidea, mientras que $g1(\cdot)$ y $g2(\cdot)$ son funciones de activación no lineal puntuales implementadas generalmente como tangentes hiperbólicas que reducen los valores en [1, 1]. Finalmente, $\odot$ es la multiplicación de entrada entre dos vectores (producto Hadamard) (Bianchi et al., 2017).

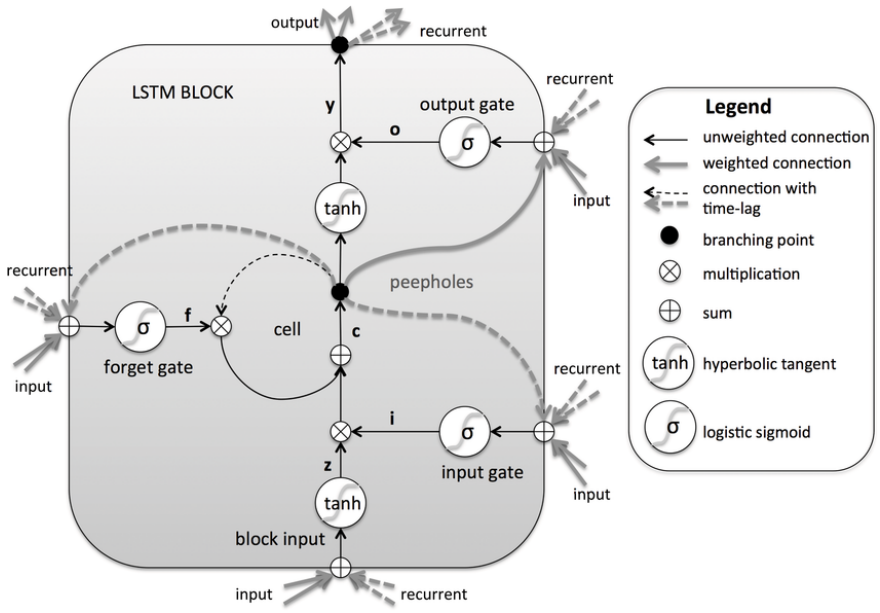


Figura 2.3: Célula LSTM (Greff et al., 2017).

Discusión

Este Capítulo presento un panorama general sobre conceptos que se abordan a lo largo del documento, específicamente de manera introductoria a los Vehículos Autónomos y de manera precisa a las soluciones propuestas en el paradigma del Aprendizaje Profundo. Sin embargo por si mismo un modelo convolucional o recurrente no es capaz de realizar el control del vehículo, por lo que es necesario realizar la combinación de técnicas de procesamiento y fusión de información, así como construir un modelo a la medida de la aplicación.

La construcción de extremo a extremo de un modelo neuronal profundo es una tarea que requiere un amplio análisis del problema a tratar y el tipo de información que se tiene de entrada y lo que se requiere como salida, además de realizar múltiples experimentaciones con equipo de computo especializado para obtener la sintonización de capas y parámetros de industrialización adecuados. Por ello en el siguiente capítulo se presenta el análisis sintetizado de los trabajos más recientes relacionados con aplicaciones de control en vehículos autoconducidos, técnicas de fusión de sensores y modelos propuestos para aplicaciones similares que sirven de base para la construcción del modelo propuesto, así como métricas que permiten evaluar el desempeño de la propuesta en términos de clasificación y autonomia de conducción.

Capítulo 3

Estado del Arte

En este Capítulo se presenta una breve descripción de los trabajos más relevantes relacionados con aplicaciones las Redes Neuronales Convolucionales y Recurrentes, así como de algunas técnicas de control de Vehículos Autónomos y fusión de sensores.

3.1. Antecedentes

Se realizó una revisión de las tesis de investigación realizadas en el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET), específicamente en el área de Aprendizaje Profundo y sus aplicaciones. Como principal uso de las redes profundas se encontró la segmentación de imágenes.

Segmentación no Paramétrica de Tejidos Cerebrales Mediante una Arquitectura Paralela de Redes Neuronales Convolucionales (Morales, 2018)

En este trabajo se desarrolló e implementó un método de segmentación no paramétrica de tejidos cerebrales, basada en una arquitectura paralela de Redes Neuronales Convolucionales (CNN). El modelo U-Net fué utilizado para la adaptación de una arquitectura paralela. La arquitectura implementada se compone de cuatro Redes Neuronales Convolucionales, a las cuales les fue realizado un ajuste paramétrico, para su entrenamiento individual. Cada uno de los modelos que componen a la arquitectura paralela, lograron realizar la segmentación binaria de un tejido cerebral; la segmentación completa de la imagen evaluada se obtuvo mediante la unificación de los tejidos.

Los repositorios de imágenes utilizadas fueron BrainWeb para entrenamiento y BraTS 2017 para validación y pruebas. A los estudios de resonancia magnética utilizados para entrenamiento se les aplicó un pre-procesamiento para la supresión de tejidos no blandos, seguido de una transformación a formato TIF (Tagged Image File Format) para su uso en los framework Keras y TensorFlow; finalmente, se realizó un aumento de información para el entrenamiento de los modelos de CNN utilizados.

Se utilizó una muestra de estudios e imágenes del repositorio BraTS 2017 para pruebas en las cuales se logró la detección de las regiones que componen a la imagen de resonancia magnética cerebral. Ésto sin realizar una inicialización de parámetros o indicación del número de regiones esperadas, aún con la variación de regiones en las imágenes cerebrales. La metodología tiene un buen rendimiento de segmentación con respecto a las métricas: Jaccard, Coeficiente de similaridad Dice (Dice Similarity Coefficient) y Área bajo la curva (AUC). En promedio los resultados obtenidos con el repositorio BrainWeb fueron de 0.9653 y en cuanto al repositorio BraTS 2017 se obtuvo un 0.9692.

Sintonización de una red totalmente contada para segmentación de dos clases de objetos en imágenes (Suárez, 2018)

En este trabajo se realizó la sintonización de una red neuronal convolucional en el entorno de trabajo Caffe, para la segmentación de dos clases (persona y ave), en imágenes del repositorio BSDS500. Se utilizaron los modelos FCN-Alexnet y FCN-8s para segmentar imágenes a color sin ruido, pero con sombras, texturas y colores similares, iluminación no uniforme y oclusión.

Detección de anormalidades en mamografías utilizando la Red Neuronal Convolucional AlexNet (Matuz, 2017)

En este trabajo se presentó una metodología para la detección de anormalidades en mamografías, mediante una Red Neuronal Convolucional. La metodología propuesta fue constituida en dos partes, en la primera se realizó el preprocesamiento de las imágenes y la segunda parte consistió en el diseño y caracterización de la red convolucional, basada en la arquitectura AlexNet para la detección de anormalidades en mamografías. Esta implementación fue probada con la base de datos MIAS (322 mamografías) y una muestra de DDSM con (250 imágenes), fue realizada una clasificación cambiando los parámetros de clasificación al modelo de red neuronal, formando cinco clasificadores de mamografías como sanas o con algún padecimiento. El modelo de clasificación con mejores resultados tenía los siguientes parámetros: imágenes a color con dimensión de píxeles, en formato PNG, con 50 épocas y una taza de aprendizaje de 0.001 la cual obtuvo una exactitud de 92%, sensibilidad del 88% y especificidad del 96%.

Compresión de imágenes usando Redes Neuronales Artificiales Recurrentes (Pérez, 2001)

Este trabajo de tesis del año 2000 utiliza una RNN optimizada por el algoritmo OLL para codificar y decodificar imágenes. Esta RNN se asemeja a una red NARX de aprendizaje por reforzamiento, sin embargo esta solo cuenta de una sola capa oculta como se muestra en la Figura 3.1.

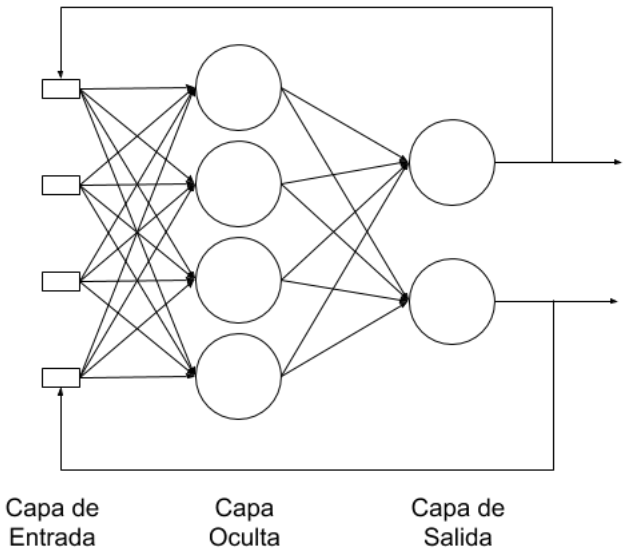


Figura 3.1: Modelo propuesto (Pérez, 2001).

Las ideas básicas del algoritmo OLL son:

- Durante la optimización de los pesos V (salidas) se dejan los pesos W constantes, y viceversa para la optimización de los pesos W. De aquí el nombre de optimización capa por capa.
- La optimización de la capa de salida es un problema que se soluciona por medio de ecuaciones lineales.
- La optimización de los pesos W de la capa oculta se convierte en un problema de solución de ecuaciones lineales, a través de la linealización de las funciones de activación sigmoidales.

3.2. Estado del Arte

En esta sección se describen documentos científicos relacionados al diseño e implementación de vehículos autónomos, Redes Neuronales Convolucionales y

Recurrentes, así como fusión de sensores. Debido a que la autoconducción y el Aprendizaje Profundo es un tema muy abordado en los últimos años, se encontró una gran variedad de artículos publicados por lo que se presentan aquellos que son más relevantes en el estado del arte.

Aprendizaje de extremo a extremo para autos sin conductor (Bojarski et al., 2016)

Este artículo reporta el trabajo realizado en la compañía NVIDIA en un esfuerzo por evaluar un modelo de Red Neuronal Convolucional que procesa imágenes obtenidas de 3 cámaras en la parte frontal de un vehículo, el modelo transforma la información visual en características relevantes del camino como las lineas y bordes del camino, que posteriormente son memorizadas por neuronas Perceptrón Multicapa que realizan el control del vehículo emitiendo comandos de dirección. Esta implementación utiliza la ayuda de GPU's NVIDIA para poder realizar la toma de decisiones en tiempo real, aunado a un disco de estado sólido SSD que almacena los frames de vídeo capturados junto a los comando de dirección obtenidos durante el entrenamiento del modelo.

Para el entrenamiento del modelo CNN se realizaron pruebas de conducción humana, para etiquetar cada frame de vídeo obtenido por las cámaras ubicadas detrás del parabrisas del vehículo se captura el giro del volante ajustado como $\frac{1}{r}$ donde r está dado por radianes en metros. De la misma manera, las cámaras aportan esta información de manera aleatoria y es pasada al modelo CNN, posteriormente se evalúa el error obtenido en cada iteración comparando el comando de dirección obtenido en la CNN con el obtenido del volante para ajustar los pesos de la red, como es mostrado de manera simple en la Figura 3.2.

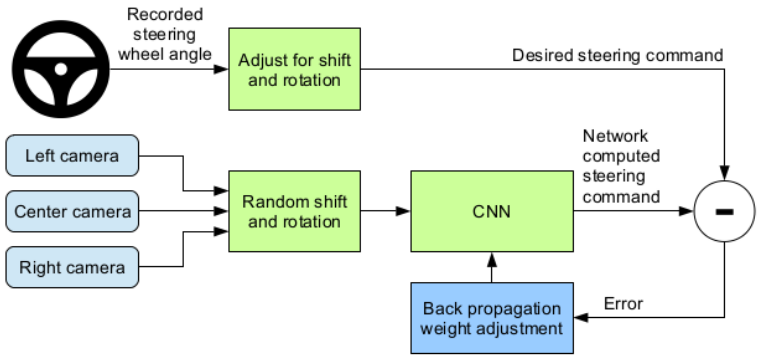


Figura 3.2: Esquema de entrenamiento (Bojarski et al., 2016).

Para la etapa de pruebas, solo es necesaria una cámara (cámara central), las imágenes captadas de esta se procesan en la red y predice un comando de dirección. Para evaluar la autonomía del vehículo se propuso la ecuación 3.1, en la cual se consideran el número de intervenciones humanas para cambiar la dirección del vehículo cuando las decisiones del modelo no son acertados, el tiempo que ha transcurrido desde que se inició la experimentación hasta que se realiza la evaluación.

$$autonomia = (1 - \frac{intervenciones \cdot 6segundos}{tiempo\ transcurrido[segundos]}) \cdot 100 \tag{3.1}$$

Explicando cómo una Red Neuronal de Aprendizaje Profundo es entrenada con aprendizaje de extremo a extremo para dirigir una automóvil (Bojarski et al., 2017)

Este es un trabajo complementario al anterior mencionado. NVIDIA presenta un sistema basado en CNN, conocido como PilotNet. El modelo está entrenado usando imágenes de carreteras junto con los ángulos de dirección generados por un humano que maneja un automóvil mientras se recolectan los datos de entrenamiento (al igual que en el caso anterior). El objetivo de este trabajo es explicar lo que PilotNet aprende y cómo toma sus decisiones. Con este fin se propone un método para determinar qué elementos de la imagen del camino influyen más en la decisiones de PilotNet.

El modelo propuesto consta de 9 capas compuestas por una capa de normalización, 5 capas convolucionales y 3 capas completamente conectadas, mostrado en la Figura 3.3.

- La primera capa de la red realiza la normalización de la imagen, el normalizador no se ajusta en el proceso de aprendizaje.
- Las capas convolucionales se diseñaron para realizar la extracción de características.
- Las capas completamente conectadas están diseñadas para funcionar como un controlador para la dirección.

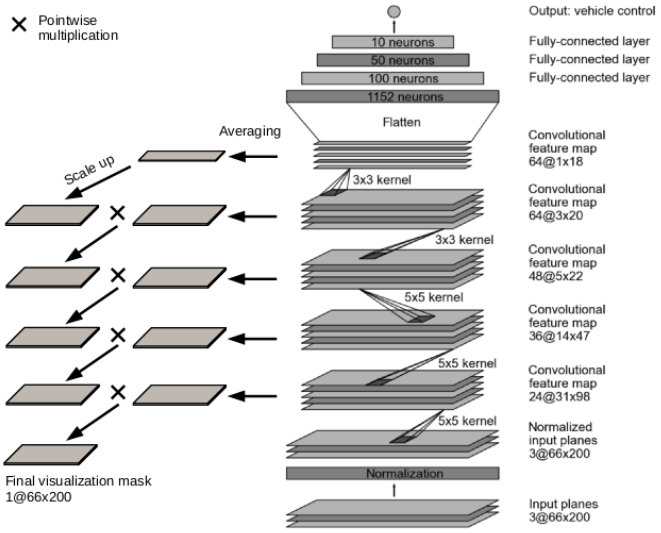


Figura 3.3: Modelo PilotNet (Bojarski et al., 2017).

Aprendizaje de extremo a extremo para un simulador de conducción (Alexeev et al., 2018).

En esta experimentación se realiza el control de un vehículo en el simulador *Udacity Self-Driving Car*. El método de control esta dado por una Red Neuronal Convolucional similar a la propuesta de (Bojarski et al., 2017), conformada por capas convolucionales y Perceptrón. Para los entrenamientos se usan 7,698 instancias. El modelo empleado para la predicción de comandos de dirección es referenciado como una modificación al modelo propuesto por el equipo de investigación de la empresa NVIDIA en (Bojarski et al., 2016), buscando simplificar el procesamiento ya que se experimenta con muchos menos datos. Finalmente el modelo propuesto consta de tres capas convolucionales intercaladas con una capa de *max pooling* para reducción de longitud de los tensores, terminando con tres capas totalmente conectadas de 500, 100 y 20 neuronas. El modelo se puede apreciar de forma más clara en la Figura 3.4.

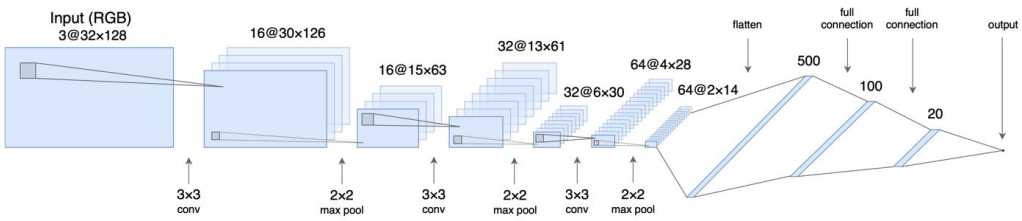


Figura 3.4: Modelo de red neuronal utilizado (Alexeev et al., 2018).

Los resultados que se obtienen al entrenar por 20 épocas con una tasa de aprendizaje (η) de 10^{-4} son mencionados como aceptables, sin embargo no se reportan resultados cuantitativos que respaldan esta opinión.

CARLA: Un simulador libre de conducción urbana (Dosovitskiy et al., 2017).

Car Learning to Act, es un simulador de código abierto para la investigación de conducción autónoma. Es de código abierto y proporciona modelos digitales libres (diseños urbanos, edificios, vehículos) que se crearon para este propósito y se pueden usar libremente. Además la plataforma admite especificaciones flexibles de conjuntos de sensores y condiciones ambientales.

En este trabajo se utiliza este simulador para estudiar el desempeño de tres enfoques para la conducción autónoma:

- Canalización modular, que descompone la tarea de conducción en percepción dada por el modelo *RefineNet* (Lin et al., 2017), un planificador local y un control dado por un controlador PID (Emirler et al., 2014).
- Modelo entrenado a través del aprendizaje por imitación, éste utiliza un conjunto de datos de conducción registrados por conductores humanos en la ciudad de entrenamiento. El conjunto de datos $D = o_i, c_i, a_i$ consta de tuplas, cada una de las cuales contiene una observación o_i, un comando c_i y una acción a_i.
- Modelo entrenado a través del aprendizaje por refuerzo, donde se usó el algoritmo actor crítico de ventaja asíncrona (A3C) (Mnih et al., 2016).

Los enfoques se evaluaron en escenarios controlados de dificultad creciente y su rendimiento se examina mediante las métricas proporcionadas por CARLA, que ilustran la utilidad de la plataforma para la investigación de conducción autónoma.

DeepTest: Prueba automatizada de Vehículos Autónomos basados en Redes Neuronales de Aprendizaje Profundo (Tian et al., 2017)

En este documento se implementa y evalúa DeepTest, una herramienta de prueba sistemática para detectar automáticamente los comportamientos erróneos de los vehículos conducidos por DNN que pueden conducir potencialmente a fallas fatales. Esta herramienta está diseñada para casos de prueba generados

automáticamente aprovechando los cambios del mundo real en condiciones de conducción como lluvia, niebla, condiciones de iluminación, etc. El reporte de este trabajo fue la comparación de 3 modelo distintos:

- *Rambo*. El modelo Rambo consiste en tres CNN cuyas salidas se fusionan utilizando una capa final (Álvarez, 2016). Dos de las CNN están inspiradas en la arquitectura de automóviles autodidacta de NVIDIA, y la tercera CNN se basa en el modelo de dirección de *comma.ai*. A diferencia de otros modelos que toman imágenes individuales como entrada, Rambo toma las diferencias entre tres imágenes consecutivas como entrada.

- *Chauffeur*. El modelo Chauffeur incluye un modelo CNN para extraer características de la imagen y un modelo LSTM para predecir el ángulo de dirección (Higgins, 2016). La entrada del modelo CNN es una imagen, mientras que la entrada del modelo LSTM es la concatenación de 100 características extraídas por el modelo CNN de las 100 imágenes consecutivas anteriores.

- *Epoch*. El modelo Epoch usa una sola CNN. Dado que el modelo pre-entrenado Epoch no está disponible públicamente, se duplicó el modelo utilizando las instrucciones proporcionadas por los autores. Se usó el conjunto de datos CH2_002 del Udacity Self-driving Challenge (Virgo, 2016) para entrenar el modelo.

DeepPicar: un vehículo autónomo de bajo costo basado en una Red Neuronal de Aprendizaje Profundo (Bechtel et al., 2017)

DeepPicar es una implementación del vehículo DAVE-2 de NVIDIA, mencionado en la sección 3.2. Este trabajo replica el modelo de aprendizaje profundo y lo integra en un vehículo a escala de bajo costo, procesada por una Raspberry Pi 3. DeepPicar es entrenado a partir de la conducción manual de la pista de pruebas. El conjunto de imágenes de entrenamiento consta de 30 vueltas a la pista, en total consta de 2556 frames de entrenamiento y 2609 frames de validación. El modelo CNN que comparte con DAVE-2 se muestra en la Figura 3.5. Los recursos libres para usar el modelo se encuentran libres en el repositorio actualizado (Bechtel, 2018).

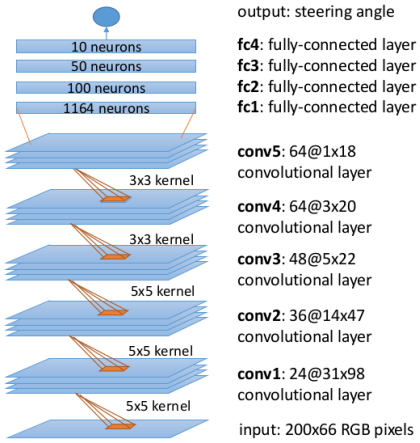


Figura 3.5: Modelo CNN DAVE-2 (Bechtel et al., 2017).

Implementación del algoritmo de detección de vehículos para automóviles autodirigidos en la carretera de peaje Cipularang utilizando el lenguaje Python (Aziz et al., 2017)

Este documento discute los resultados de la implementación del algoritmo de detección de vehículos en carretera como parte del sistema de auto conducción. Para este trabajo se utiliza el algoritmo Gradientes Orientados a Histogramas (*Histogram Oriented Gradients*, HOG). HOG es uno de los algoritmos de extracción de características para fines de detección de objetos.

Modelo cognitivo inspirado en el cerebro con atención para autos de conducción automática (Chen et al., 2017a)

En este artículo se propone un modelo para automóviles autónomos llamado modelo cognitivo inspirado en el cerebro con atención. Este modelo consta de tres partes: una red neuronal convolucional para simular la corteza visual humana, un mapa cognitivo construido para describir las relaciones entre objetos en una escena de tráfico compleja, y una red neuronal recurrente que se combina con el mapa cognitivo actualizado en tiempo real para implementar mecanismo de atención y LSTM. Se pretende que este modelo pueda ser capaz resolver con precisión tres tareas simultáneamente: detección del espacio libre y los límites de los carriles actuales y adyacentes, estimación de la distancia de obstáculos y el vehículo, y aprendizaje del comportamiento de conducción y la toma de decisiones del conductor humano.

Métricas utilizadas Para evaluar el modelo se optó por utilizar las métricas *Precision*, *Recall* y *F-measure*, las cuales son definidas en el sistema de ecuaciones:

$$Precision = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{3.2}$$

$$Recall = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{3.3}$$

$$F1 = \frac{2N_{TP}}{2N_{TP} + N_{FP} + N_{FN}} \tag{3.4}$$

donde N_{TP} es el número de píxeles correctamente etiquetados como terreno libre, N_{FP} es el número de píxeles etiquetados por el modelo pero no es realmente terreno libre y por último N_{FN} que es el número de píxeles en terreno libre real pero no etiquetados por el modelo.

DeepDriving: Affordancia de aprendizaje para la percepción directa en el manejo autónomo (Chen et al., 2015)

Como lo indica este trabajo; hoy en día, existen dos paradigmas principales para los sistemas de conducción autónoma basados en la visión: enfoques de percepción mediados que analiza una escena completa para tomar una decisión de conducción y enfoques de comportamiento reflejo que directamente asigna una imagen de entrada a una acción de conducción por un regresor. En este artículo se propone un tercer paradigma: un enfoque de percepción directa para estimar la posibilidad de conducir. El control de la dirección se calcula utilizando la posición del automóvil, y el objetivo es minimizar la brecha entre la posición actual del automóvil y la línea central del carril. Definiendo *dist_center* como la distancia a la línea central del carril, se tiene la ecuación 3.5.

$$steer = C \cdot (angle - dist\_center/road\_width) \tag{3.5}$$

donde C es un coeficiente que varía bajo diferentes condiciones de manejo, y el ángulo $\in [-\pi, \pi]$. Cuando el automóvil cambia de carril, la línea central cambia del carril actual al carril objetivo.

Detección monocular de objetos tridimensionales para conducción autónoma (Chen et al., 2016)

El objetivo de este trabajo es realizar la detección de objetos tridimensionales a partir de una única imagen monocular en el dominio de la conducción autónoma. El

método propuesto, primero tiene como objetivo generar un conjunto de propuestas de objetos específicos de la clase candidata, que luego se ejecutan a través de una canalización CNN estándar para obtener detecciones de objetos de alta calidad. El objetivo de este documento es la generación de propuestas. En particular, se propone un enfoque de minimización de energía que coloca candidatos a objetos en 3D utilizando el hecho de que los objetos deben estar en el plano de tierra.

Red de detección de objetos 3D de múltiples vistas para conducción autónoma (Chen et al., 2017b)

Este es un trabajo más reciente del mismo autor, el cual tiene como objetivo la detección de objetos 3D de alta precisión en el escenario de conducción autónoma. Se proponen las redes 3D Multi-View (MV3D), un marco de fusión sensorial que toma tanto la nube de puntos LIDAR como las imágenes RGB como entrada y predice cajas de límites 3D orientadas. La red está compuesta por dos subredes: una para la generación de propuestas de objetos 3D y otra para la fusión de funciones de vistas múltiples. La red de propuestas genera cuadros de candidatos 3D de manera eficiente a partir de la representación a vista de pájaro (vistas desde la parte superior, vista aérea) de la nube de puntos 3D. Se diseñó un esquema de fusión profunda para combinar características regionales de múltiples vistas y permitir interacciones entre capas intermedias de diferentes rutas.

Aprendizaje de extremo a extremo de los modelos de conducción con conjuntos de datos de video a gran escala (Xu et al., 2016)

Este trabajo pretende entrenar un modelo genérico de movimiento de vehículos a partir de datos de vídeo a gran escala basados en multitud de fuentes y desarrollar una arquitectura entrenable de extremo a extremo para aprender a predecir una distribución sobre el movimiento futuro de vehículo a partir de observaciones de cámaras monoculares instantáneas y estado previo del vehículo. Este modelo incorpora una arquitectura FCN-LSTM, que puede entrenarse a partir de datos de acción de vehículos de gran escala basados multitud de fuentes a gran escala, y aprovecha las tareas disponibles de segmentación de escenas para mejorar el rendimiento bajo un paradigma de aprendizaje privilegiado. Otro aporte es un nuevo conjunto de datos a gran escala del comportamiento de múltiples conducción, adecuado para el entrenamiento del modelo propuesto. Este dataset es el Berkeley *DeepDrive Video dataset* (BDDV), es un conjunto de datos compuesto por vídeos

reales de conducción y datos de GPS/IMU.

Base de datos. El conjunto de datos *Berkeley DeepDrive Video* (BDDV) es un conjunto de datos compuesto por videos reales de conducción y datos de GPS / IMU. El conjunto de datos de BDDV contiene diversos escenarios de conducción que incluyen ciudades, carreteras, pueblos y áreas rurales en varias ciudades importantes de EE. UU.

Aprendizaje seguro, multiagente y reforzado para conducción autónoma (Shalev-Shwartz et al., 2016)

En este trabajo se aplica el aprendizaje profundo por refuerzo al problema de la formación de estrategias de conducción a largo plazo.

Aquí se hace la observación que hay dos desafíos principales que hacen que la conducción autónoma sea diferente de otras tareas robóticas. Primero, es la necesidad de garantizar la seguridad funcional, algo que el aprendizaje automático tiene dificultades dado que el rendimiento se optimiza en el nivel de una expectativa con muchas instancias. En segundo lugar, el modelo de proceso de decisión de Markov que se usa a menudo en robótica es problemático en este caso debido al comportamiento impredecible de otros agentes en este escenario de múltiples agentes.

Diseño de un sistema autónomo de control de automóviles basado en Lógica Difusa (Widaa and Talha, 2017)

En este trabajo presenta un diseño de un sistema de control de automóvil autónomo utilizando control basado en Lógica Difusa (Fuzzy Logic Control, FLC). El diseño propuesto realiza las tareas de control de manejo como: velocidad, frenos, volante. El sistema de control consta de dos unidades principales para realizar todas las tareas de conducción: la unidad de control basado en lógica difusa y la unidad de visión por computadora.

Unidad de control

- Subsistema de frenado automático:

  Se debe usar un grupo de sensores para detectar y detectar otros vehículos u obstáculos, incluidos RADAR, LIDAR, cámaras de vídeo.

- Subsistema de velocidad automática: Los parámetros se dividen en tres:

  1. Condiciones ambientales: Lluvia: (0 - 200)mm. Velocidad de los vientos: (0 - 150)km/h. Temperatura: $(-30 - 55)^\circ$ C.
  2. Condición del automóvil: Condición de las llantas: (0 - 25)psi. Peso (coche + pasajeros): (1870 - 1500)kg. Calor del motor: $(-25 - 65)^\circ$ C.
  3. Estado de la carretera: A todos estos tres factores se les da una escala de 0 a 10 comenzando desde la condición / estado malo (0) hasta la mejor condición / estado (10)

- Subsistema de dirección automática:

  Para el control del volante se puede determinar dos parámetros:

  1. Cambio angular: (-180 - 180) grado.
  2. Cambio lateral: (-1 - 1) m.

Control difuso de un automóvil autónomo con un teléfono inteligente (Olson et al., 2017)

Este documento presenta resultados preliminares sobre el desarrollo de un controlador basado en visión con un teléfono inteligente en el circuito para vehículos autónomos. Implica el uso de la cámara disponible en un teléfono inteligente para implementar el control de velocidad para una aplicación de seguimiento, en condiciones de tráfico intermitente. El controlador utiliza lógica difusa para determinar los comandos en función de la posición estimada del automóvil y la velocidad con respecto al vehículo objetivo que se está siguiendo. La solución se implementó con base a la integración de tres sistemas:

- Sistema de percepción que estima la distancia al auto utilizando un algoritmo de visión por computadora que se ejecuta en el hardware de los teléfonos inteligentes.
- Controlador que calcula la velocidad requerida del automóvil en función de la distancia al automóvil de enfrente y la velocidad actual. Este sistema debe ser lo suficientemente simple como para ser incluido como un aplicación en el dispositivo móvil.

- Se implementó el sistema de control en un *Ford Escape Hybrid* 2009 actuado por *TORC Robotics 'ByWire XGV* para recibir mensajes que se apegan al protocolo *SAE AE4 Standard* Arquitectura Conjunta para Sistemas No Tripulados, usando generación de código y técnicas de interfaz para traducir mensajes JAUS a ROS (Morley et al., 2013).

Redes Neuronales Recurrentes para la detección de objetos en secuencias de vídeo (Haapala et al., 2017)

Ahora dejando de lado las aplicaciones implícitas para vehículos autónomos, en un enfoque dirigido a Redes Neuronales Recurrentes, se presenta este trabajo de tesis en el cual se evalúan y comparan 5 modelos distintos de DNN en la tarea de detección de objetos en secuencias de vídeo. La comparación se realiza en las arquitecturas CNN, multi-frame CNN, RNN, GRU y una Forecast RNN, de las anteriores solo la multi-frame recibe más de un frame como parámetro por época. Otra aportación interesante son las propuestas de métricas para evaluar el desempeño de estas redes, las métricas constan de:

- **Rendimiento de detección de objetos**.

$$IOU(b_1, b_2) = \frac{Area(Intersection(b_1, b_2))}{Area(Union(b_1, b_2))} \tag{3.6}$$

- **Desviación estándar ponderada de los grupos**.

$$\sigma = \sqrt{\frac{1}{nA} \frac{\Sigma_{n=1}^{n} w_i (b_i - \mu)^2}{\Sigma_{n=1}^{n} w_i}} \tag{3.7}$$

$$\mu = \frac{\Sigma_{n=1}^{n} w_i b_i}{\Sigma_{n=1}^{n} w_i} \tag{3.8}$$

$$A = \frac{\Sigma_{n=1}^{n} w_i Area(b_i)}{\Sigma_{n=1}^{n} w_i} \tag{3.9}$$

donde n es el número de cuadros delimitadores en el grupo, w_i es la cobertura asociada del i-ésimo cuadro delimitador b_i, μ es el recuadro delimitador medio ponderado en el clúster y A es el área del recuadro delimitador ponderado medio.

- **Estimación de la velocidad**.

$$\delta_v = \frac{||v_{predicted} - v_{true}||_2}{||v_{true}||_2} \tag{3.10}$$

- **Tolerancia al ruido**.

  Se agregan diferentes niveles de ruido gaussiano a las imágenes del conjunto de validación. La hipótesis es que las redes recurrentes deberían ser capaces de integrar información temporal de frames previos y producir una precisión promedio más alta aún con ruido que una CNN de línea base.

Sobre la mejora de la Red Neuronal Recurrente para la clasificación de imágenes (Chandra and Sharma, 2017)

Nuevamente un trabajo de comparación, en este se hace la comparación exclusivamente entre variantes de RNN con datasets de imágenes. Las arquitecturas mencionadas en este trabajo son:

- **LSTM**: mencionado en el marco teórico.
- **Identity Initialized RNN**: Identity initialized RNN (IRNN) utiliza la función de activación lineal rectificada para la unidad oculta e inicializa la matriz recurrente oculta como matriz identidad. Dada la secuencia de entrada, la secuencia de salida se obtiene como:

$$h_t = relu(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{3.11}$$
$$y_t = W_{hy} + b_y \tag{3.12}$$

  donde $relu$ denota la función de activación lineal rectificada, W y b corresponden a los pesos y bias.

- **Parallel Channel RNN**: PC-RNN es el método propuesto en este trabajo. Utiliza dos canales paralelos $h^{(1)_t}$ y $h^{(2)_t}$.

  El ajuste de la capa oculta y de salida se calculan de la siguiente manera:

$$h_t = (mod(h^{(1)_t}) + mod(h^{(2)_t}))/2 \tag{3.13}$$
$$y_t = W_{hy} + b_y \tag{3.14}$$

- **Single Channel RNN**: Otra variante del método propuesto se usa en caso de solo un canal. El método se denomina SC-RNN.

Para complementar la comparación resulta interesante ver también el artículo de (Lin et al., 2013), en donde se realizan pruebas similares con métodos distintos.

Red Neuronal Convolucional Recurrente para el reconocimiento de objetos (Liang and Hu, 2015)

Este trabajo integra un modelo de Red Neuronal Convolucional con lo anterior visto sobre Redes Neuronales Recurrentes. Propone el modelo de Red Recurrente Convolucional (*Recurrent Convolutional Neural Network*, RCNN).

Capa Convolucional Recurrente El módulo clave de RCNN es la capa Convolucional recurrente (*Recurrent Convolutional Layer*, RCL). Los estados de las unidades de RCL evolucionan en pasos de tiempo discretos. Para una unidad ubicada en (i,j) en el k-ésimo mapa de características en un RCL, su entrada neta $z_{ijk}(t)$ en el paso de tiempo t está dada por:

$$z_{ijk}(t) = (\mathbf{w}_k^f)^T \mathbf{u}^{(i,j)}(t) + (w_k^r)^T \mathbf{x}^{(i,j)}(t-1) + b_k \tag{3.15}$$

En la ecuación $u_{(i,j)}(t)$ y $x_{(i,j)}(t-1)$ denota la entrada feedforward y recurrente, respectivamente, que son los vectores centrados en (i,j) de los mapas de características en la capa anterior y la actual, w_k^f y w_k^r denotan los pesos vectorizados *feedforward* y los pesos recurrentes, respectivamente, y b_k es el bias. El primer término en (3.15) se usa en CNN estándar y el segundo término es inducido por las conexiones recurrentes.

$$x_{ijk}(t) = g(f(z_{ijk}(t))) \tag{3.16}$$

$$f(z_{ijk}(t)) = max(z_{ijk}(t), 0) \tag{3.17}$$

$$g(f_{ijk}(t)) = \frac{f_{ijk}(t)}{(1 + \frac{\alpha}{N}\Sigma_{k'=max(0,k-N/2)}^{min(K,k+N/2)}(f_{ijk})^2)^\beta} \tag{3.18}$$

La actividad o estado de esta unidad está en función de su entrada, donde f es la función de activación lineal rectificada. g es la función de normalización de respuesta local (LRN), donde K es el número total de mapas de características en la capa actual. Se tiene cuenta que en el denominador ($f_{ijk}(t)$) la suma se ejecuta en N mapas de características en la misma ubicación (i,j) (generalmente $N \leq K$), y α y β son constantes que controlan la amplitud de la normalización. Además $f(z_{ijk}(t))$ se ha abreviado como $f_{ijk}(t)$. LRN imita la inhibición lateral en la corteza, donde diferentes características compiten por grandes respuestas.

Redes Convolucionales Recurrentes de Largo Plazo para Reconocimiento y Descripción Visual (Donahue et al., 2015)

Este trabajo, al igual que el anterior, consta de una combinación de una Red Convolucional y una Recurrente, en este caso LSTM. Propuesto como *Long-term Recurrent Convolutional Network* (LRCN), realiza la clasificación en imágenes, secuencia de imágenes y vídeos. EL modelo LRCN funciona pasando cada entrada visual v_t (una imagen aislada, o un cuadro de un vídeo) a través de una transformación de características $\phi V(v_t)$ parametrizada por V para producir una representación vectorial de longitud fija $\phi_t \in \mathbb{R}^d$.

Fusión de la nube de puntos LiDAR 3D con imagen de cámara digital 2D (Li, 2015).

Esta tesis de maestría presenta la técnica para fusionar una imagen obtenida por una cámara RGB con definición 480×320 píxeles con un sensor LiDAR que da una nube da puntos con una apertura de 180° en sentido horizontal y en vertical. Los datos del sensor LiDAR son visualizados en coordenadas polares. En el documento se detalla la transformación a la escala coordenada, este proceso es necesario para poder empatar la profundidad de cada punto 3D con cada pixel de la imagen de referencia. El objetivo es transformar las coordenadas $(x,y,z) \mapsto (u,v)$, donde (x,y,z) representan las coordenadas del mundo real $x,y \leftarrow [0^\circ,180^\circ]$ y $z \leftarrow [0,300]cm$. Las coordenadas (u,v) son las dimensiones cartesianas equivalentes a la imagen de referencia. Para realizar la transformación se toma en cuenta la distancia focal de la cámara f y se obtiene una equivalencia mediante las Ecuaciones (3.19) y (3.20).

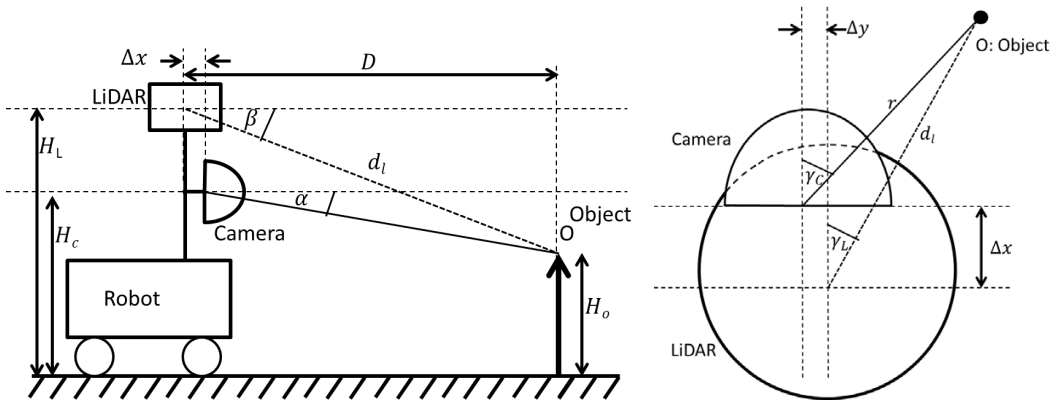
$$u = f\frac{x}{z} \tag{3.19}$$

$$v = f\frac{y}{z} \tag{3.20}$$

Al realizar la transformación la nube de puntos presenta algunas regiones faltantes, para resolver este inconveniente se realiza una interpretación por el método del vecino más cercano, mostrando una imagen de profundidad que es equivalente a la imagen de referencia.

Fusión robusta de LiDAR y datos de cámara de gran angular para robots móviles autónomos (De Silva et al., 2018).

Este artículo muestra un método alterno al anterior, se tiene un sensor LiDAR con apertura horizontal de 180° y una cámara con la misma apertura vertical. El método consta de una alineación geométrica basada en un punto de referencia. Esta alineación depende de una serie de variables espaciales físicas mostradas en la Figura 3.6. Donde H_L, H_c y H_o representa la altura a la que se encuentran el sensor LiDAR, la cámara y el objeto O de interés con respecto al piso en metros, las variables γ_L y γ_c representan el ángulo horizontal al que se encuentra O con respecto a el sensor y la cámara, de igual manera α representa el ángulo vertical de la cámara con respecto a O y β del sensor LiDAR. La variable d_l representa la distancia medida por el sensor LiDAR al objeto O, basándose en estas medidas se calcula la distancia D del robot con respecto al objeto de interés. Δx y Δy representan la diferencia espacial entre ambos sensores.



(a) Vista lateral. (b) Vista superior.

Figura 3.6: Representación de la configuración espacial utilizada en el robot móvil (De Silva et al., 2018).

Para realizar la alineación se considera una serie de ecuaciones para encontrar el valor óptimo de desplazamiento de la nube de puntos. Esta serie de ecuaciones se resumen en las funciones de desplazamiento en latitud (3.21) y longitud (3.22).

$$tan\ \alpha = \frac{((H_c - H_L) + d_l \cdot sin\ \beta) \cdot cos\ \gamma_c}{d_l \cdot cos\ \beta \cdot cos\ \gamma_L - \Delta x} \tag{3.21}$$

$$tan\ \gamma_c = \frac{d_l \cdot cos\ \beta \cdot sin\ \gamma_L + \Delta y}{d_l \cdot cos\ \beta \cdot cos\ \gamma_L - \Delta x} \tag{3.22}$$

Sin embargo este método sigue teniendo el mismo problema que el documento anterior, la diferencia de tamaño entre la nube de puntos LiDAR y la imagen presenta regiones vacías aunque tengan la misma apertura horizontal. Para llenar los espacios faltantes y obtener una imagen RGB-D se utiliza el Proceso de Regresión Gaussiana (3.23) donde $K(\cdot)$ es la función de covarianza entre los datos LiDAR x y la imagen x' y $m(\cdot)$ la matriz de transformación para x.

$$f(x) \sim GP(m(x), K(x, x')) \tag{3.23}$$

Registro de nubes de puntos LiDAR por técnica de detección de imágenes (Han et al., 2012).

Este artículo muestra cómo alinear la nube de puntos LiDAR con una imagen de referencia mediante la identificación de el objeto de interés en ambas señales. En ambos casos se debe realizar la detección de un objeto de interés para poder realizar el mapeo de pixeles. Para la nube de puntos se utiliza una versión modificada del algoritmo SURF para extracción de descriptores 3D, estos sirven para detectar un objeto que coincide con la imagen de referencia.

Una vez identificado el objeto se realiza la alineación de los datos mediante la matriz de traslación (3.24) donde se considera un factor de escala k_i y una matriz de rotación M transpuesta. Las variables X_i, Y_i y Z_i pertenecen a las coordenadas de la imagen así como x_0 y y_0 que indican en centro de la imagen tal como lo indica la Figura **??**. Por otro lado X_L, Y_L y Z_L pertenecen a los datos de la nube de puntos LiDAR.

$$\begin{Bmatrix} X_i \\ Y_i \\ Z_i \end{Bmatrix} = \frac{1}{k_i} M^T \begin{Bmatrix} x_i - x_0 \\ y_i - y_0 \\ -f \end{Bmatrix} + \begin{Bmatrix} X_L \\ Y_L \\ Z_L \end{Bmatrix} \tag{3.24}$$

Detección de automóviles para vehículos autónomos: LIDAR y Fusión Visual. Enfoque a través de Aprendizaje Profundo (Du et al., 2017)

En este documento se propone un sistema de fusión de visión y LIDAR para la detección de automóviles a través de aprendizaje profundo con un modelo llamado PC-CNN. Este modelo es nombrado así debido al término *Point Cloud* o Nube de Puntos.

Este modelo, como se muestra en la Figura 3.7, consiste de tres partes principales:

- La primera parte genera propuestas de semillas para posibles ubicaciones de automóviles en la imagen teniendo en cuenta la nube de puntos LIDAR.
- La segunda parte refina la ubicación de los cuadros de propuesta explorando información de varias capas en la red de propuesta.
- La tercera parte lleva a cabo la tarea de detección final a través de una red de detección que comparte parte de las capas con la red de propuesta.

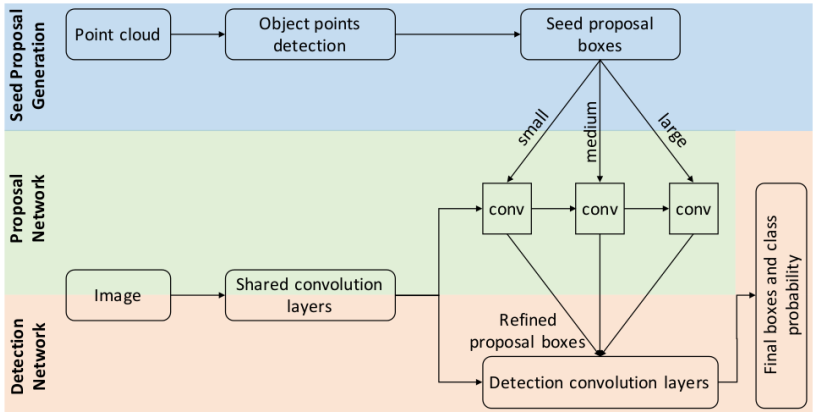


Figura 3.7: Descripción del modelo Point Cloud-CNN (Du et al., 2017).

Discusión del Estado del Arte

Para un mejor orden, la discusión se divide en tres partes; las publicaciones que trabajan directamente en crear un vehículo autónomo independientemente de su técnica, los que trabajan en reconocimiento y clasificación con Redes Neuronales Convolucionales y Recurrentes, y los dedicados a realizar fusión de sensores visuales y espaciales.

Para el estado del arte referente al desarrollo de vehículos autónomos se toma en cuenta como aspectos más importantes el algoritmo o técnica para la toma de decisiones, tipo de entrenamiento, ambiente de pruebas y métricas de evaluación. Y para el estado del arte referente al uso de Redes Neuronales Recurrentes se toma en cuenta arquitectura(s) utilizada(s), tipos de datos, campo de aplicación y resultados obtenidos. Para la parte de fusión de sensores se toma en cuenta la técnica de fusión, los sensores en específico y la utilidad para esta investigación. El resumen de la discusión se muestra en la Tabla 3.1.

Tabla 3.1: Discusión del Estado del Arte.

Vehículos Autónomos					
Publicación	Algoritmo / Técnica	Entrenamiento	Ambiente	Métricas de evaluación	Utilidad
Aprendizaje de extremo a extremo para autos sin conductor.	Red Convolucional.	Conducción manual, vídeo capturado de 3 cámaras y lectura de ángulos de dirección del volante.	Simulación y camino real.	$a = (1 - \frac{intervenciones \cdot 6segundos}{transcurrido[segundos]}) \cdot 100$	**Esquema de entrenamiento y métrica para evaluar la autonomía**.
Explicando cómo una Red Neuronal de Aprendizaje Profundo es entrenada con aprendizaje de extremo a extremo para dirigir una automóvil.	Red Convolucional PilotNet.	Conducción manual, video capturado de 3 cámaras y lectura de ángulos de dirección del volante ($1/r$, radianes).	Camino real.	$a = (1 - \frac{intervenciones \cdot 6segundos}{transcurrido[segundos]}) \cdot 100$	**Modelo utilizado y comandos para el control de dirección**.
Aprendizaje de extremo a extremo para un simulador de conducción	Modelo Neuronal Pilotnet de (Bianchi et al., 2017)	Imágenes de conducción obtenidas de la conducción manual en un simulador	Simulador *Udacity*	Métricas de autonomía del mismo autor de Pilotnet	Uso del simulador *Udacity* para pruebas preliminares
CARLA: Un simulador libre de conducción urbana	Canalización modular, Aprendizaje por refuerzo y Aprendizaje por imitación	Modelo de red neuronal entrenado a partir de la prueba y error acelerados en el simulador	Simulador *Car Learning to Act* CARLA	Evaluación basada en pruebas exitosas sin colisión	Simulador y base teórica para interpretar el funcionamiento de un vehículo autoconducido

Tabla 3.1: Discusión del Estado del Arte (continuación).

Publicación	Algoritmo / Técnica	Entrenamiento	Ambiente	Métricas de evaluación	Utilidad
DeepTest: Prueba automatizada de Vehículos Autónomos basados en Redes Neuronales de Aprendizaje Profundo.	▪ Rambo: 3 CNN. ▪ Epoch: CNN. ▪ Chaffeur: CNN y LSTM.	Simulación: Udacity CH_002 dataset.	Simulación Udacity.	$NeuronCoverage = \frac{\lvert Activated \rvert}{\lvert Total \rvert}$	**Modelos evaluados (CNN y RNN), datos para entrenamiento y el simulador Udacity**.
DeepPicar: un vehículo autónomo de bajo costo basado en una Red Neuronal de Aprendizaje Profundo.	Red Convolucional (mismo modelo mencionado en 3.2), aplicado a vehículo a escala.	Conducción manual a la pista, captura en vídeo de 30 vueltas al circuito.	Pista diseñada para vehículo a escala.	Métricas orientadas al consumo de recursos.	**Modelo utilizado y datos de entrenamiento**.
Implementación del algoritmo de detección de vehículos para automóviles autodirigidos en la carretera de peaje Cipularang utilizando el lenguaje Python.	Algoritmo *Histogram Oriented Gradients* (HOG) detección de objetos.	No necesita entrenamiento, se prueba en carreteras con distintas perspectivas.	Carretera, cámaras estáticas para detectar vehículos.	Matriz de confusión.	**Algoritmo basado en visión implementado para detectar otros vehículos**.
Modelo cognitivo inspirado en el cerebro con atención para autos de conducción automática.	Modelo cognitivo inspirado en el cerebro con atención (CMA).	Road-Vehicule Dataset (RVD): más de 30 mil imágenes capturadas por 3 cámaras	Camino real.	Las métricas utilizadas se basan en el número de píxeles etiquetados como camino libre, bajo el concepto de verdaderos positivos, etc.	**Enfoque de combinación CNN y LSTM, métricas de evaluación y datos de entrenamiento de alta calidad**.

Tabla 3.1: Discusión del Estado del Arte (continuación).

Publicación	Algoritmo / Técnica	Entrenamiento	Ambiente	Métricas de evaluación	Utilidad
DeepDriving: Afordancia de aprendizaje para la percepción directa en el manejo autónomo.	CNN con un enfoque de percepción que busca un camino transitable en donde no existan obstáculos.	se entrenó utilizando la grabación de 12 horas de conducción humana en un videojuego.	Detección en el dataset KITTI.	Falsos positivos, Verdaderos positivos, Falsos negativos y Verdaderos negativos.	**Ecuaciones para producir comandos para viraje y control de velocidad**.
Detección monocular de objetos tridimensionales para conducción autónoma.	Red Neuronal Convolucional para detección de objetos en 3D.	Autos, Peatones y Ciclistas en KITTI dataset, set de entrenamiento.	Detección en el dataset KITTI	Precisión media (AP): 1. $\int_0^1 p(r)dr$ 2. $\Sigma_{k=1}^{N} P(k)\Delta rk$	**Modelo basado en visión para detectar otros vehículos y comparación con otros modelos RNN**.
Red de detección de objetos 3D de múltiples vistas para conducción autónoma.	3D Multi-View (MV3D), fusión entre nube de puntos LIDAR e imágenes RGB. Basado en Red Convolucional.	Imágenes RGB, datos de nube LIDAR tomados de frente y nube LIDAR tomado desde la vista aérea.	KITTI dataset solo para detección 2D.	Precisión media (AP).	**Modelo de alta precisión basado en visión para detectar otros vehículos**.
Aprendizaje de extremo a extremo de los modelos de conducción con conjuntos de datos de vídeo a gran escala.	FCN-LSTM.	Berkeley DeepDrive Video dataset (BDDV), es un conjunto de datos compuesto por vídeos reales de conducción y datos de GPS / IMU.	El dataset BDDV cuenta con un set de prueba con 3,561 vídeos y un set de validación con 1,470. Cada vídeo es de aproximadamente 40 segundos con frames de 640 x 360.	Se utiliza la perplejidad como métrica.	**Modelo de RNN que fusiona datos de sensores e imágenes para la toma de decisiones, datos de entrenamiento y una métrica de evaluación**.

Tabla 3.1: Discusión del Estado del Arte (continuación).

Publicación	Algoritmo / Técnica	Entrenamiento	Ambiente	Métricas de evaluación	Utilidad
Aprendizaje seguro, multiagente y reforzado para conducción autónoma.	LSTM con aprendizaje por refuerzo.	Se ejecuta la simulación y ajusta la red con las acciones realizadas.	Simulación.	No especificada.	Enfoque de aprendizaje por refuerzo integrado a una red LSTM.
Diseño de un sistema autónomo de control de automóviles basado en Lógica Difusa.	Unidad de control basado en Lógica Difusa (FLC).	Funciona creando condiciones *IF-THEN* expresadas como reglas difusas.	Pruebas dentro del sistema, no se creó implementación en vehículos o simulaciones.	Error en relación entre variables, representados gráficamente.	**Parámetros para una unidad de control basado en lógica difusa**.
Control difuso de un automóvil autónomo con un teléfono inteligente.	Se realiza una integración de tres subsistemas: sistema de percepción visual, control basado en lógica difusa y comunicación simulación - vehículo real (ROS - JAUS).	Funciona creando condiciones IF-THEN basado en lógica difusa.	Simulación, vehículo real.	Porcentajes de error medio.	**Introducción a la integración de simulación en ROS con un vehículo real**.

Tabla 3.1: Discusión del Estado del Arte (continuación).

Redes Neuronales Recurrentes					
Publicación	Arquitectura utilizada	Tipos de datos	Campo de Aplicación	Resultados obtenidos	Utilidad
Redes Neuronales Recurrentes para la detección de objetos en secuencias de vídeo.	1. CNN. 2. Multi-frame CNN. 3. RNN. 4. GRU. 5. Forecast CNN.	Secuencias de vídeo.	Detección de objetos en vídeo.	▪ Estimación de la velocidad. ▪ Desviación estándar ponderada de los clusters. ▪ Tolerancia al ruido.	**Métricas de evaluación para modelos para detección de objetos en vídeos**.
Sobre la mejora de la Red Neuronal Recurrente para la clasificación de imágenes.	1. LSTM. 2. Identity Initialized IRNN. 3. PC-RNN1. 4. PC-RNN2. 5. Single Chanel.	Imágenes de los dataset: 1. MNIST 2. CIFAR10 3. CIFAR100 4. NORB 5. SVHN	Clasificación de imágenes.	▪ MNIST: LSTM. ▪ CIFAR10: PC-RNN1. ▪ CIFAR100: PC-RNN2. ▪ NORB: LSTM. ▪ SVHN: PC-RNN2.	**Propuesta de dos modelo RNN y comparación con otros modelos recurrentes**.

Tabla 3.1: Discusión del Estado del Arte (continuación).

Publicación	Arquitectura utilizada	Tipos de datos	Campo de Aplicación	Resultados obtenidos	Utilidad
Red Neuronal Convolucional Recurrente para el reconocimiento de objetos.	Recurrent Convolutional Neural Network (RCNN).	Imágenes de los dataset: 1. MNIST 2. CIFAR10 3. CIFAR100 4. SVHN	Reconocimiento de objetos en imágenes.	El modelo propuesto RCNN resultó tener un índice de error muy bajo para el dataset MNIST con 0.31% en su mejor prueba. En la prueba con el dataset CIFAR-100 resultó con el índice mayor de error con 31.75% en su mejor prueba.	**Un modelo con un enfoque basado en visión combinando CNN y RNN**.
Redes Convolucionales Recurrentes de Largo Plazo para Reconocimiento y Descripción Visual.	Long-term Recurrent Convolutional Network (LRCN).	1. Imagen. 2. Secuencias de imágenes. 3. Vídeo.	1. Imagen: descripción de imágenes. 2. Secuencias de imágenes: reconocimiento de actividades. 3. Video: descripción en secuencias de vídeo.	1. Descripción de imágenes: 65.4% precisión. 2. Reconocimiento de actividades: 82.66%, métrica BLEU. 3. Descripción en secuencias de vídeo: 28.8%, métrica BLEU.	**Un modelo con un enfoque basado en visión combinando CNN y LSTM que puede ser usado para imágenes o vídeos**.

Tabla 3.1: Discusión del Estado del Arte (continuación).

Fusión de sensores					
Publicación	Técnica utilizada	Tipos de sensores	Campo de Aplicación	Resultados obtenidos	Utilidad
Fusión de la nube de puntos LiDAR 3D con imagen de cámara digital 2D	Transformación de coordenadas polares del mundo real a cartesianas del dominio focal de una imagen	Cámara RGB 480 × 320 y sensor LiDAR de 12,000 puntos	Recreación 3D de interiores y exteriores	Obtiene una alineación de mayor calidad, sin embargo es necesario realizar interpolación para llenar espacios vacíos	Configuración de los sensores y transformación de datos
Fusión robusta de LiDAR y datos de cámara de gran angular para robots móviles autónomos	Alineación mediante traslación en longitud y latitud basada en la posición de los sensores y las características de sensado	Robot móvil, escaner LiDAR y cámara monocular de 180°	Robots móviles	Permite la fusión aproximada al tiempo real con la alineación e interpolación basado en un filtro gaussiano	Alineación y posición de los sensores
Registro de nubes de puntos LiDAR por técnica de detección de imágenes	Algoritmo SURF modificado para extracción de descriptores 3D y alineación mediante una matriz de rotación y traslación.	Escaner LiDAR y cámara monocular	Recreación 3D de exteriores	Obtiene la imagen con un canal de profundidad en 3.2 segundos, no apto para tiempo real	Tomar como referencia la matriz de traslación y rotación para la alineación de los sensores
Detección de automóviles para vehículos autónomos: LIDAR y Fusión Visual. Enfoque a través de Aprendizaje Profundo.	Point Cloud, PC-CNN. Fusión de visión y LIDAR.	Dataset creado con datos de nube de puntos LIDAR e imágenes RGB	KITTI object detection dataset	Precisión media (AP).	**Modelo que fusiona los datos del sensor LIDAR con una CNN**.

Como se observó en esta síntesis de documentos científicos recientes, el tema de la autoconducción es un tema actual que se ha trabajado desde inicios de la década y vigentes a la fecha. Particularmente la implementación de nuevos modelos de Redes Neuronales y variantes han sido parte medular del control de los vehículos, así como la fusión de sensores para obtener mayor información del entorno en el que opera.

A partir del anterior análisis se toma como base el modelo Chauffeur evaluado en el artículo de (Tian et al., 2017), el cual es expandido a una variante distribuida en el tiempo similar a la utilizada en los modelos de (Haapala et al., 2017), sin embargo no se limita a las capas recurrentes, sino que se aplica el mismo principio a las capas de convolución permitiendo el procesamiento de secuencias de imágenes, lo cual es el aporte científico principal de esta tesis. Además a estas se les dotó de la capacidad de procesar imágenes con canal de profundidad como en la propuesta de (Chen et al., 2017b), sin embargo el sensor LiDAR es posicionado para capturar el frente del camino en lugar de vista aérea, y mediante una combinación de transformación y alineación de nubes de puntos de los trabajos de (Li, 2015) y (De Silva et al., 2018) se realiza la fusión.

En resumen, el análisis del estado del arte presentado dio cabida a proponer un modelo de control neuronal que ajusta sus parámetros basados en el análisis temporal de los datos, así como usar una combinación de técnicas de fusión cámara-LiDAR utilizando las partes fuertes de cada una para resolver las deficiencias de la otra. La propuesta antes mencionada. así como la simulación desarrollada para las pruebas son descritas detalladamente en el siguiente Capítulo.

Capítulo 4

Metodología

En este Capítulo se aborda el tema principal de esta investigación desglosado en dos módulos constituyentes de un vehículo autoconducido. El primer módulo está orientado a la simulación, como fue diseñado el vehículo y los sensores utilizados para la percepción, así como los ambientes integrados para la evaluación y la fusión de sensores detallando la composición de la base de datos propuesta. El segundo modulo refiere al control realizado por el modelo neuronal, se detalla el funcionamiento del modelo base y la diferencia con el modelo propuesto, así como el procesamiento interno de la información dentro de los modelos profundos.

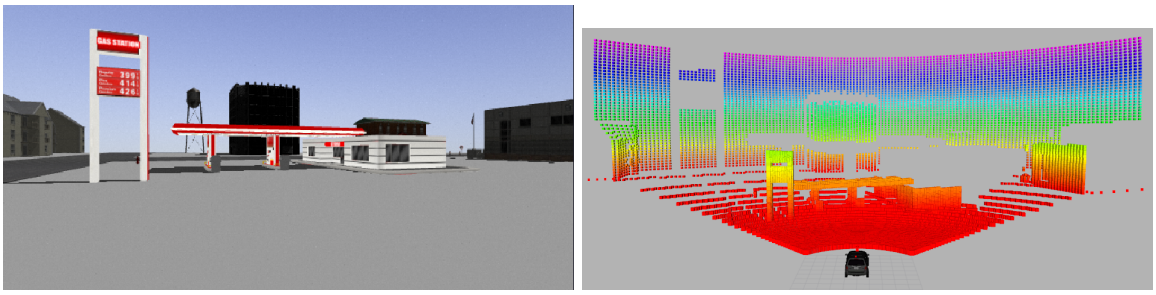
4.1. Simulación del vehículo autónomo

El primero módulo a detallar es el de percepción. Inicialmente se requiere de un vehículo simulado y el entorno en el que se desenvolverá, por ello en esta sección se muestra el diseño del vehículo, los sensores y entornos simulados. También se detalla una base de datos de imágenes con canal de profundidad creada en este simulador y que es utilizada para el entrenamiento del módulo de control.

4.1.1. Vehículo y sensores

Esta tarea se realizó basándose en los diseños de libre acceso de *Cyber-Physical Systems Virtual Organization*. Esta organización proporciona su paquete de ROS llamado *CAT Vehicle Testbed* (CPS-VO, 2012), el cual consta del diseño de un vehículo CUV Ford Escape con las medidas físicas reales a escala del simulador. Además este paquete incluye por defecto algunos sensores incorporados al vehículo, sin embargo fueron modificados a medida de la aplicación aquí mostrada. Originalmente cuenta con un sensor Velodyne LiDAR con una nube de 2,000 puntos, este fue modificado para producir una nube de 12,000 puntos con una apertura horizontal de 90° y vertical de 33.67°. De la misma manera el paquete cuenta con dos cámaras posicionadas sobre el vehículo en ángulos de $-45°$ y $45°$ con origen al frente del vehículo, estas capturan imágenes RGB de tamaño 800×800 píxeles. Para esta

aplicación se requirió de una sola cámara apuntando al origen con tamaño de 640×320 píxeles.



(a) Cámara RGB. (b) Nube de puntos Velodyne LiDAR.

Figura 4.1: Sensores utilizados en la simulación probados en el escenario propuesto.

Para el sensor LiDAR es imprescindible la compatibilidad con GPU ya que en la versión CPU por defecto obtiene muestras a 5 *hz* promedio consumiendo gran parte de los recursos del equipo y ralentizando la simulación. Con la versión GPU se incrementa a una media de 40 *hz* minimizando el efecto en la simulación. Además de los sensores antes mencionados, la simulación incorpora sensores de velocidad y ángulo de giro que son utilizados para el control del vehículo, también incorpora sensores GPS y láser que no fueron considerados. La posición de los sensores en el vehículo puede visualizarse en la Figura 4.2.

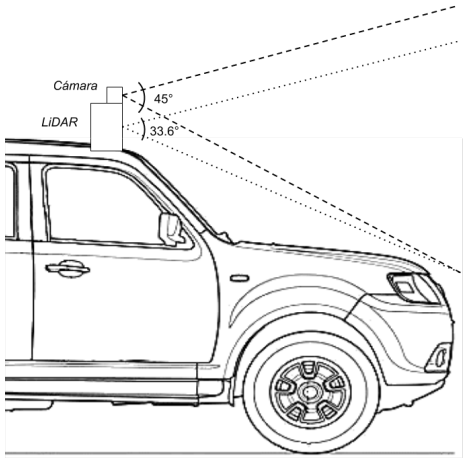
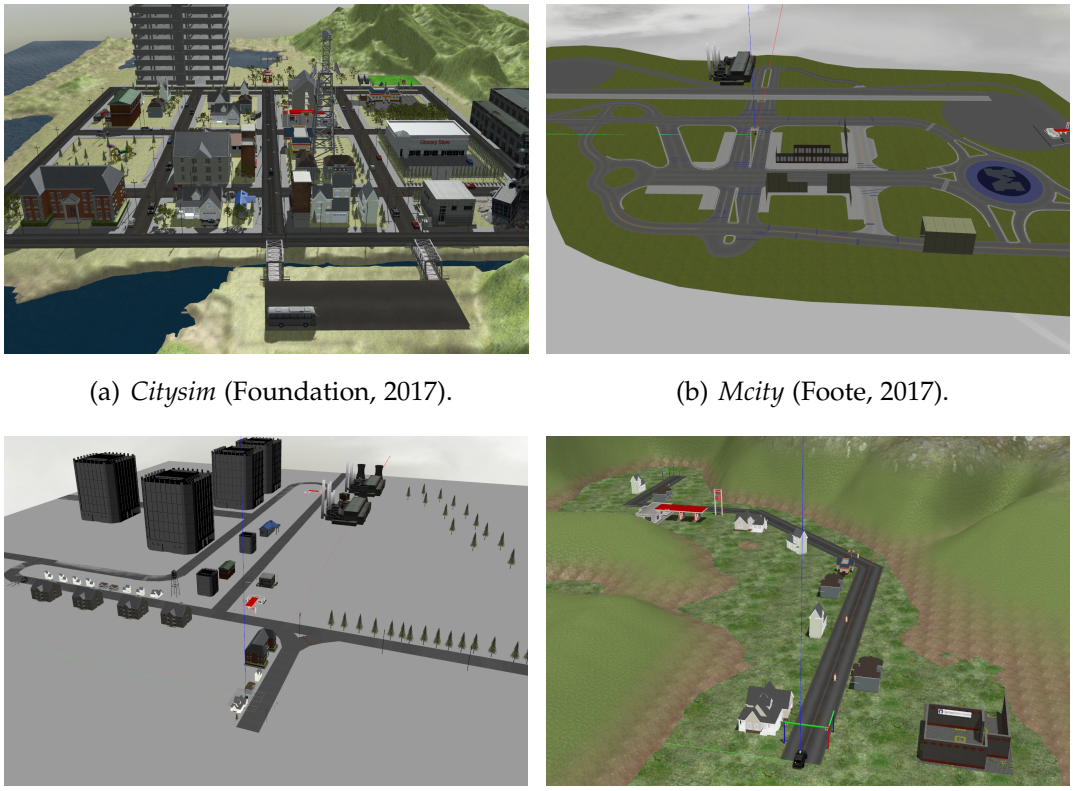


Figura 4.2: Ubicación de los sensores (De Silva et al., 2018).

Para crear la base de datos y evaluar la autonomía del vehículo fue necesario integrar entornos de ciudad con distintas características como intersecciones, casas y edificios, caminos cerrados y obstáculos en el camino. Para tener una variedad

de posibles escenarios se integró el *Vehicle and city simulation* (Foundation, 2017) que representa de forma completa una pequeña urbe, este se muestra en la Figura 4.3(a). También fue integrado el ambiente *Mcity* (Foote, 2017) mostrado en la Figura 4.3(b).



(a) *Citysim* (Foundation, 2017).

(b) *Mcity* (Foote, 2017).

(c) Escenario propuesto.

(d) *VRC Task 1* (Sardinha, 2017).

Figura 4.3: Sensores utilizados en la simulación.

Además, se creó un ambiente que complementa a los anteriores, este ambiente mostrado en la Figura 4.3(c) incluye peatones estáticos en el camino, así como caminos totalmente bloqueados por conos y parcialmente obstruidos por otros objetos. Por ultimo se incluyó el escenario *VRC Task 1* (Sardinha, 2017) (Figura 4.3(d)) solamente para pruebas, ya que es breve y tiene las características de los escenarios antes mencionados.

4.1.2. Base de datos propuesta

Dados el sensor LiDAR y la cámara RGB, se realizó la fusión de estos sensores para obtener una imagen con un canal de profundidad, el cual es de gran ayuda

para la detección y evasión de objetos que se encuentran en el camino del vehículo. Dada a la similaridad de la ubicación de los sensores en el vehículo (Figura 4.2) con el robot mostrado por (De Silva et al., 2018), se realizó la alineación de la nube de puntos con la imagen de referencia mediante el método propuesto en este artículo. Una vez alineada la nube de puntos, es necesario realizar una transformación de coordenadas. Esto se debe a que el sensor LiDAR provee información espacial en coordenadas polares dadas en (x', y', z) donde x' y y' son ángulos horizontales y verticales con respecto al origen y z el radio de estas coordenadas. Es necesario realizar una transformación inicial de coordenadas polares a cartesianas mediante: $x = z \cdot \sin(x')$ y $y = z \cdot \sin(y')$. Posteriormente es necesario realizar la transformación $(x, y, z) \mapsto (u, v)$ de las coordenadas del mundo real a un espacio plano de la imagen. Para realizar la transformación se toma en cuenta la distancia focal f de la cámara a fusionar, posteriormente se transforma mediante (4.1) y (4.2).

$$u = f\frac{x}{z} \tag{4.1}$$

$$v = f\frac{y}{z} \tag{4.2}$$

Al realizar esta transformación se obtiene una imagen de mayor tamaño y con valores faltantes como se muestra en la Figura 4.4.

Figura 4.4: Imagen de profundidad obtenida al transformar coordenadas.

Para llenar los espacios vacíos es necesario realizar una interpolación en la imagen de profundidad. Debido a que el tiempo de respuesta del sistema debe ser mínimo, el proceso de interpolación se debe realizar con métodos simples y de baja complejidad algorítmica.

Para solucionar este problema se requiere una interpolación con un filtro de tamaño 5×5. Se compararon 3 diferentes filtros, el primero es el filtro de media, el cual se obtiene calculando la media del filtro $\bar{x} = \frac{1}{N}\sum_{i=0}^{N} x_i$, sin embargo este método tiene 0 tolerancia a ruido, por lo que los valores faltantes oscurecen la imagen. La segunda prueba fue con el filtro de mediana $\tilde{x} = med[\sum_{i=0}^{N} x(i)]$, este a diferencia del anterior presenta resultados mucho más robustos al definir firmemente los bordes e incrementa la definición de estos, sin ser afectado por los valores faltantes. La

desventaja de este método de interpolación es la complejidad $O(n^2)$ en comparación de la complejidad $O(n)$ del filtro de media, recordando que el tiempo de respuesta es crucial, aunque el filtro de mediana da una mayor calidad de interpolación, no es lo óptimo para esta implementación.

Para balancear la calidad de interpolación y complejidad algorítmica se optó por utilizar el filtro $\alpha - trimmed\ mean$ (Bednar and Watt, 1984). Este método expresado en la Ecuación (4.3) es una combinación de la mediana y la media, particularmente se hace un ordenamiento $[\alpha N]$ donde N es el total de datos en el conjunto y α determina la fracción del conjunto a truncar al inicio y al final de la lista ordenada. Con los objetos que restan al centro de la lista se calcula la media aritmética para así obtener el valor de X_α. La complejidad de este método es de $O(\alpha N)$.

$$X_\alpha = \frac{1}{N - 2 \cdot [\alpha N]} \sum_{i=[\alpha N]+1}^{N-[\alpha N]} x_i \qquad 0 < \alpha < 0.5 \tag{4.3}$$

Esta fusión se realiza de forma paralela con ayuda de la librería PyCUDA que embebe el lenguaje C++ en el lenguaje Python, de esta manera se obtiene una estructura tensorial de dimensión $640 \times 320 \times 4$ equivalentes a el tamaño de la imagen de la cámara del vehículo por los cuatro canales RGB-D, este tensor es almacenado en una estructura de Python llamada *Numpy Array* y guardada en disco con extensión *.npz* en una versión comprimida, la cual es leída con la librería Numpy del lenguaje. La base de datos creada tiene las siguientes características:

- 40,372 instancias en total almacenadas en 28.8 GB comprimidos.
- 20,178 imágenes capturadas en la pista de la Figura 4.3(a), 11,914 de la pista 4.3(b) y 9,280 en la pista de la Figura 4.3(c).
- Cuenta con un archivo en formato *.csv* que contiene las direcciones en disco de cada imagen, el comando de giro en intervalo normalizado $[-1,1]$ y la velocidad del vehículo en el intervalo $[-3,10)$ expresado en m/s.

4.2. Modelo neuronal convolucional recurrente RGB-D

Esta parte del sistema pertenece al módulo de control, el cual está dado por un modelo de Red Neuronal de Aprendizaje Profundo adaptada para trabajar con secuencias de imágenes con canal de profundidad generadas en la simulación

desarrollada. A continuación se detalla el funcionamiento del modelo base y de la versión mejorada.

4.2.1. Modelo Chauffeur

El modelo neuronal base en cuestión refiere a el modelo Chauffeur evaluado por (Tian et al., 2017). El modelo es un híbrido de Red Convolucional para extracción de mapas características en imágenes, capas recurrentes *Long Short-Term Memory* (LSTM) (Hochreiter and Schmidhuber, 1997) que procesan y memorizan los patrones de los datos de entrenamiento, dos capas *dropout* (Srivastava et al., 2014) que previenen el sobreajuste de la red mediante la intervención aleatoria de las conexiones sinápticas de la red con el cuarto tipo de capas, las capas de Perceptrón Multicapa. Originalmente el modelo Chauffeur es habilitado para trabajar con una sola imagen a color y es evaluado en la tarea de clasificación de direcciones en escenarios de conducción en condiciones climáticas no favorables (lluvia, niebla, entre otras).

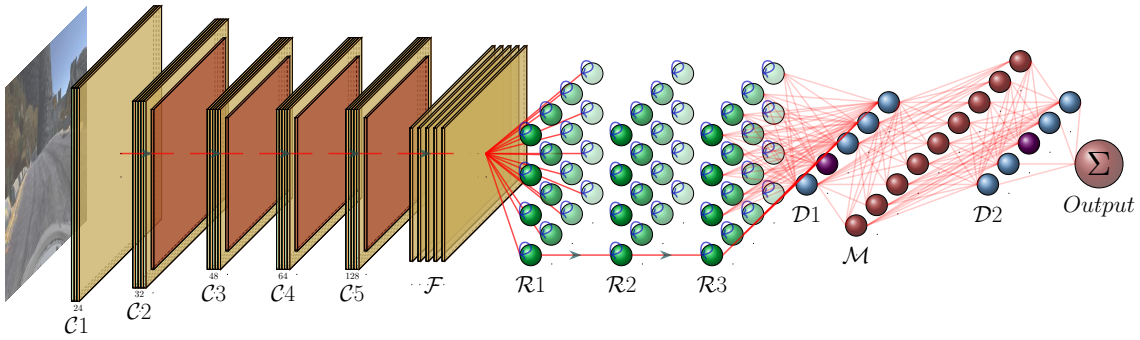


Figura 4.5: Modelo Neuronal Chauffeur (Tian et al., 2017).

Como se muestra en la Figura 4.5, el modelo consta inicialmente de una entrada X con dimensión $w \times h \times 3$ acorde a las dimensiones de las imágenes en la base de datos y 3 canales de color rojo, verde y azul (RGB). La capa $\mathcal{C}1$ se conforma de una capa de convolución; ésta genera k mapas de características obtenidos de la convolución de un filtro w^k de tamaño $m \times n \times d$ por cada valor de píxel y canal de color en X más un bias b^k. Aplicando filtros w de tamaño 5×5 en $\mathcal{C}1$ se generan $k = 24$ mapas de características de tamaño $w - \lfloor 5/2 \rfloor \cdot 2 \times h - \lfloor 5/2 \rfloor \cdot 2$, para cada índice i, j en el mapa de características k:

$$\mathcal{C}1^k_{i,j} = \sum_{m=0}^{4} \sum_{n=0}^{4} \sum_{d=0}^{3} X_{m+i,j+n,d} * w^k_{m,n,d} + b^k \tag{4.4}$$

Estas capas convolucionales utilizan como función de activación la función Unidad Lineal Rectificada (ReLU) $f(x) = \text{máx}(x, 0)$, por lo que la Ecuación (4.4) es reescrita como:

$$\mathcal{C}1_{i,j}^{k} = f\left(\sum_{m=0}^{4}\sum_{n=0}^{4}\sum_{d=0}^{3} X_{m+i,j+n,d} w_{m,n,d}^{k} + b^{k}\right) \tag{4.5}$$

A diferencia de $\mathcal{C}1$, $\mathcal{C}2$ es una capa de convolución con submuestra. Esta técnica reemplaza las capas de *Pooling* para la reducción de valores. Donde además del filtro de convolución $m \times n$ se agregan factores de saltos dW y dH, de esta forma se obtiene un mapa de características reducido cuya dimensión es dada por:

$$w_{\mathcal{C}2} = \frac{w_{\mathcal{C}1} - m}{dW} + 1 \tag{4.6}$$

$$h_{\mathcal{C}2} = \frac{h_{\mathcal{C}1} - n}{dH} + 1 \tag{4.7}$$

donde $w_{\mathcal{C}1}$ y $h_{\mathcal{C}1}$ son la totalidad de los índices i y j de cada mapa de características k obtenidos en la capa $\mathcal{C}1$, $w_{\mathcal{C}2}$ y $h_{\mathcal{C}2}$ son las dimensiones ya reducidas por la capa $\mathcal{C}2$. Aplicando $k = 32$ filtros de convolución w de 5×5 y se submuestreo $dW = 2 \times dH = 2$, la capa $\mathcal{C}2$ es expresada como:

$$\mathcal{C}2_{i,j}^{k} = f\left(\sum_{m=0}^{4}\sum_{n=0}^{4}\sum_{d=0}^{23} \mathcal{C}1_{dW*i+m,dH*j+n,d} * w_{m,n,d}^{k} + b^{k}\right) \tag{4.8}$$

$\mathcal{C}2$ produce 32 mapas de $w_{\mathcal{C}2} = (w_{\mathcal{C}1} - 5)/2 + 1 \times h_{\mathcal{C}2} = (h_{\mathcal{C}1} - 5)/2 + 1$ a partir de los 24 generados de $\mathcal{C}1$. Siguiendo este método $\mathcal{C}3$ extrae $k = 48$ mapas con filtros w de 3×3 y submuestreo de 2×2, $\mathcal{C}4$ $k = 64$ con filtros de 3×3 y submuestreo de 2×2, y $\mathcal{C}5$ $k = 128$ mapas con filtros de 3×3 y submuestreo de 2×1. Para que la información extraída sea memorizada por las capas recurrentes es necesario realizar una reducción de dimensiones. Los mapas de características generados en $\mathcal{C}5$ son compactados en la capa $\mathcal{F}$ de *Flatten*, esta transforma $\mathcal{F} : \mathcal{C}5^{m \times n \times d} \mapsto \mathcal{C}5^{(m \times n \times d)}$ a partir de:

$$\mathcal{F}_i = \sum_{d=0}^{128}\sum_{m=0}^{w_{\mathcal{C}5}}\sum_{n=0}^{h_{\mathcal{C}5}} \mathcal{C}5_{m,n,d} \tag{4.9}$$

de manera que en $\mathcal{F}$ se extrae un vector de $m \times n \times d$ valores. Estos sirven de entrada para $\mathcal{R}1$, la cual consta de 64 neuronas LSTM (Hochreiter and Schmidhuber, 1997). Una neurona comprende de 5 unidades que garantizan la recurrencia y

correcta memorización de los patrones de entrenamiento. Como se muestra en la Figura 4.6, las compuertas son: la compuerta del olvido σ_f, compuerta de actualización σ_u, compuerta de salida σ_o, compuerta de entrada $\tilde{h}$ y el estado de la neurona h:

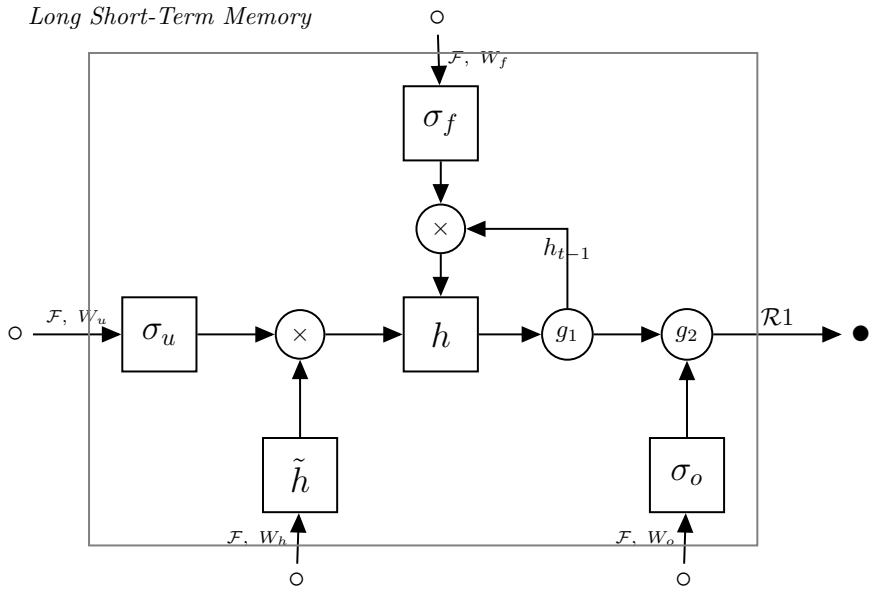


Figura 4.6: Modelo de la neurona LSTM (Venna et al., 2018).

$$\sigma_f^{\mathcal{R}1} = \sigma\left(\mathbf{W}_f^{\mathcal{R}1}\mathcal{F} + \mathbf{R}_f^{\mathcal{R}1}\mathcal{R}1 + \mathbf{b}_f^{\mathcal{R}1}\right) \tag{4.10}$$

$$\tilde{h}^{\mathcal{R}1} = g_1\left(\mathbf{W}_h^{\mathcal{R}1}\mathcal{F} + \mathbf{R}_h^{\mathcal{R}1}\mathcal{R}1 + \mathbf{b}_h^{\mathcal{R}1}\right) \tag{4.11}$$

$$\sigma_u^{\mathcal{R}1} = \sigma\left(\mathbf{W}_u^{\mathcal{R}1}\mathcal{F} + \mathbf{R}_u^{\mathcal{R}1}\mathcal{R}1 + \mathbf{b}_u^{\mathcal{R}1}\right) \tag{4.12}$$

$$\sigma_o^{\mathcal{R}1} = \sigma\left(\mathbf{W}_o^{\mathcal{R}1}\mathcal{F} + \mathbf{R}_o^{\mathcal{R}1}\mathcal{R}1 + \mathbf{b}_o^{\mathcal{R}1}\right) \tag{4.13}$$

$$\mathbf{h}^{\mathcal{R}1} = \sigma_u^{\mathcal{R}1} \odot \tilde{h}^{\mathcal{R}1} + \sigma_f^{\mathcal{R}1} + \mathbf{h}^{\mathcal{R}1} \tag{4.14}$$

donde W son los pesos de cada unidad, b los bias y R son los pesos sinápticos de cada unidad a si misma. $\odot$ es el producto Hadamard entre dos vectores, $\sigma(\cdot)$ representa la función sigmoide $\sigma(x) = \frac{1}{1+e^{-x}}$, $g1(\cdot)$ y $g2(\cdot)$ son funciones de activación no lineales que devuelven valores en el intervalo $[-1, 1]$. La salida de la neurona es obtenida por:

$$\mathcal{R}1 = \sigma_o^{\mathcal{R}1} \odot g_2(\mathbf{h}^{\mathcal{R}1}) \tag{4.15}$$

el mismo método es aplicado para $\mathcal{R}2$ y $\mathcal{R}3$ con 64 neuronas LSTM cada una. La salida de esta capa obtiene un vector de 64 valores. Este se da como entrada a la capa $\mathcal{D}1$ de *dropout*, que bajo un parámetro aleatorio p escala la información obtenida en las capas anteriores teniendo la probabilidad de reducir a 0 el valor de alguna neurona. $\mathcal{D}1$ consta de un total de 256 neuronas:

$$\begin{aligned} \mathcal{D}_i &= \mathcal{R}3 * p_i \\ 0.2 &\leq p_i < 1.0 \end{aligned} \tag{4.16}$$

La capa $\mathcal{M}$ se conforma de 256 neuronas Perceptrón Multicapa con función de activación sigmoidal $\sigma(\cdot)$. A continuación se tiene nuevamente una capa de *dropout* en $\mathcal{D}2$ que pasa la información a la capa de salida. La capa de salida está conformada por una sola neurona Perceptrón Multicapa con función de activación Tangente hiperbólica:

$$tanh(\mathcal{D}2) = \frac{e^{\mathcal{D}2} - e^{-\mathcal{D}2}}{e^{\mathcal{D}2} + e^{-\mathcal{D}2}} \tag{4.17}$$

esta función de activación garantiza el control del vehículo calculando valores en el intervalo $[-1,1]$, que indican el giro del volante normalizado para dirigir el vehículo. A diferencia de otros problemas de clasificación, en esta aplicación se tiene una clase continua, esto quiere decir que no se tienen clases específicas a predecir, sino que son aproximaciones al comando esperado. Este modelo es entrenado con el método *Adaptive Moment Estimation* ADAM (Kingma and Ba, 2014) para la propagación del error. De forma general, el Algoritmo 4.1 describe las operaciones que se realizan para el ajuste de los parámetros. Donde g_i representa el gradiente ∇f calculado a partir de la función de costo y el momentum (4.18), α es la tasa de aprendizaje recomendada con un valor de 0.001, β_1 y β_2 se recomiendan con valores de 0.9 y 0.999 respectivamente y el hiperparámetro ϵ es recomendado en 10^{-8}.

$$g_i = \nabla f(w_{i-1} - \beta_i m_{i-1}) \tag{4.18}$$

Algoritmo 4.1: Algoritmo de propagación del error ADAM.

input: w_i

output: w_i

1 **while** w_i *not converged* **do**

2 $\quad i \leftarrow i + 1$

3 $\quad m_i \leftarrow \beta_1 \cdot m_{i-1} + (1 - \beta_1) \cdot g_i$

4 $\quad v_i \leftarrow \beta_2 \cdot v_{i-1} + (1 - \beta_2) \cdot g_i^2$

5 $\quad \alpha_i \leftarrow \alpha_{i-1} \cdot \frac{\sqrt{1-\beta_2^i}}{1-\beta_1^i}$

6 $\quad w_i \leftarrow w_{i-1} - \frac{\alpha_i \cdot m_i}{\sqrt{v_i}+\epsilon}$

7 **end**

8 **return** w_i

Aquí w_i refiere a todos los parámetros entrenables en todas las capas de la red que se ajustan $w_i \leftarrow w_{i-1} + \Delta w$, donde Δw es el incremento o decremento de los pesos obtenidos en la línea 6 del Algoritmo 4.1. El índice i refiere a las iteraciones de entrenamiento y es indiferente para los tiempos t de las capas recurrentes y convolucionales distribuidas en el tiempo.

4.2.2. Modelo Chauffeur mejorado

Esta versión mejorada recibe como entrada X una sucesión de t imágenes RGB-D ordenadas cronológicamente de la base de datos mostrada en la Sección 4.1.2, por lo que una entrada X es de dimensión $w \times h \times 4 \times t$. El parámetro t refiere a los tiempos a utilizar para las capas recurrentes, interpretado como los cuadros de un breve vídeo. En esta capa de entrada se realizan algunas operaciones de preprocesamiento para generar un incremento de información:

- Inversión aleatoria de la imagen: bajo la probabilidad de 0.5, de forma aleatoria se decide si la imagen de entrada es invertida en el eje Y, así como la etiqueta $y = y * -1$.
- Traslación aleatoria: la imagen se traslada aleatoriamente sobre el eje X en un intervalo $[-0.2, 0.2]$.

Gracias a este incremento de información la base de datos se expande de 40,271 instancias a 80,542 tras la primera parte del preprocesamiento, aplicando la traslación y combinándolo con la inversión se obtiene una tendencia al infinito debido a que se encuentra en un dominio continuo. La ventaja de realizar este

preprocesamiento dentro del modelo neuronal es que se generan los datos durante el entrenamiento y no es necesario almacenarse, por lo que no afecta el peso de la base de datos propuesta.

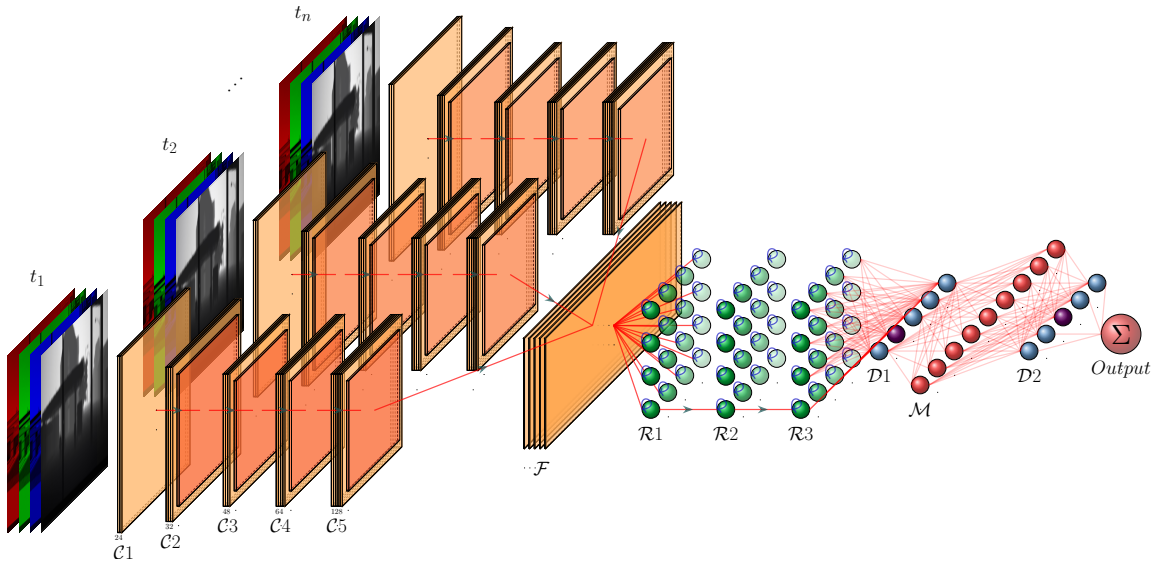


Figura 4.7: Modelo Neuronal Chauffeur distribuido en el tiempo propuesto.

El modelo mejorado que se muestra en la Figura 4.5, consta inicialmente de una entrada X con dimensión $640 \times 320 \times 4 \times t$ de acuerdo al tamaño de las imágenes en la base de datos. Para las capas convolucionales se actualiza la función de activación a la Función Unidades Lineales Exponenciales (ELU) debido a que se ha demostrado que los ELU pueden obtener una mayor precisión de clasificación que los ReLU (Clevert et al., 2015):

$$f(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases} \tag{4.19}$$

En $\mathcal{C}1$ se generan $k = 24$ mapas de características de tamaño 636×316 con filtros 5×5, para cada índice i, j en el mapa de características k en el tiempo t:

$$\mathcal{C}1_{i,j}^{t,k} = f\left(\sum_{m=0}^{4}\sum_{n=0}^{4}\sum_{d=0}^{2} X_{m+i,j+n,d} * w_{m,n,d}^{t,k} + b^{t,k}\right) \tag{4.20}$$

esta operación se realiza para cada t tiempo en la secuencia de cuadros, en esta aplicación se optó por utilizar $t = 3$ por la relación costo-beneficio observada en las experimentaciones. $\mathcal{C}2$ produce 32 mapas de 316×156 a partir de los 24 generados de $\mathcal{C}1$:

$$\mathcal{C}2_{i,j}^{t,k} = f\left(\sum_{d=0}^{24}\sum_{m=0}^{4}\sum_{n=0}^{4}\mathcal{C}1_{d,dW*i+m,d,dH*j+n}^{t}w_{d,m,n}^{t,k} + b^{t,k}\right) \tag{4.21}$$

y siguiendo este método $\mathcal{C}3$ extrae 48 mapas de 157×77, $\mathcal{C}4$ 64 mapas de 78×38 y $\mathcal{C}5$ 128 mapas de 38×36. Los mapas de características generados en $\mathcal{C}5$ son compactados en la capa $\mathcal{F}$ de *Flatten*, esta transforma $\mathcal{F} : \mathcal{C}5^{d\times m\times n\times t} \mapsto \mathcal{C}5^{(d\times m\times n)\times t}$ a partir de:

$$\mathcal{F}_i^t = \sum_{d=0}^{128}\sum_{m=0}^{38}\sum_{n=0}^{36}\mathcal{C}5_{d,m,n}^{t} \tag{4.22}$$

de manera que en $\mathcal{F}$ se extrae un t vectores de 175,104 valores. Estos sirven de entrada para la capa $\mathcal{R}1$ que es ahora distribuida en el tiempo, esto quiere decir que la información en $t-1$ ajusta R y está a la vez autoajusta los pesos sinápticos w en el tiempo t:

$$\sigma_f^{\mathcal{R}1}[t] = \sigma\left(\mathbf{W}_f^{\mathcal{R}1}[t]\mathcal{F}[t] + \mathbf{R}_f^{\mathcal{R}1}\mathcal{R}1[t-1] + \mathbf{b}_f^{\mathcal{R}1}\right) \tag{4.23}$$

$$\tilde{h}^{\mathcal{R}1}[t] = g_1\left(\mathbf{W}_h^{\mathcal{R}1}[t]\mathcal{F}[t] + \mathbf{R}_h^{\mathcal{R}1}\mathcal{R}1[t-1] + \mathbf{b}_h^{\mathcal{R}1}\right) \tag{4.24}$$

$$\sigma_u^{\mathcal{R}1}[t] = \sigma\left(\mathbf{W}_u^{\mathcal{R}1}[t]\mathcal{F}[t] + \mathbf{R}_u^{\mathcal{R}1}\mathcal{R}1[t-1] + \mathbf{b}_u^{\mathcal{R}1}\right) \tag{4.25}$$

$$\sigma_o^{\mathcal{R}1}[t] = \sigma\left(\mathbf{W}_o^{\mathcal{R}1}[t]\mathcal{F}[t] + \mathbf{R}_o^{\mathcal{R}1}\mathcal{R}1[t-1] + \mathbf{b}_o^{\mathcal{R}1}\right) \tag{4.26}$$

$$\mathbf{h}^{\mathcal{R}1}[t] = \sigma_u^{\mathcal{R}1}[t] \odot \tilde{h}^{\mathcal{R}1}[t] + \sigma_f^{\mathcal{R}1}[t] + \mathbf{h}^{\mathcal{R}1}[t-1] \tag{4.27}$$

así la recurrencia es utilizada para ajustar las neuronas con secuencias de imágenes en lugar de una sola imagen y así aportar mayor información a la red. $\mathcal{R}1[t-1]$ refiere a la salida de la neurona en un tiempo $t-1$ obtenido por:

$$\mathcal{R}1[t] = \sigma_o^{\mathcal{R}1}[t] \odot g_2(\mathbf{h}^{\mathcal{R}1}[t]) \tag{4.28}$$

donde se obtiene una de salida de t valores. Para la capa $\mathcal{R}2$ se agrupan las salidas de las 64 en t vectores de forma similar a la salida de $\mathcal{F}$. El mismo método es aplicado para $\mathcal{R}2$ y $\mathcal{R}3$, sin embargo en esta última capa se da como salida solo el último tiempo, que para $t = 3$ se obtiene de salida:

$$\mathcal{R}3[3] = \sigma_o^{\mathcal{R}3}[3] \odot g_2(\mathbf{h}^{\mathcal{R}3}[3]) \tag{4.29}$$

a partir de esta capa se obtiene un solo vector de 64 valores, este se da como entrada a la capa $\mathcal{D}1$ de *dropout* y el resto de la red funciona de la misma forma que el modelo original.

4.3. Integración del Sistema

Como se mencionó anteriormente el sistema comprende de la simulación del vehículo, sensores y entornos, la fusión de sensores y la creación de la base de datos RGBD y el modulo de control dado por el modelo de Red Neuronal Híbrido propuesto. Por separado cada modulo realiza una tarea especifica, sin embargo para concretar la simulación del Vehículo Autónomo es necesario integrar los módulos mediante un programa principal que realice la comunicación y sincronización del flujo de información.

La metodología que sigue el sistema propuesto se divide en dos fases, entrenamiento y pruebas. El motivo de dividir la metodología en dos fases surge a partir de la necesidad de entrenar el modelo previo a las pruebas en el simulador, a la vez que la interacción de los módulos es diferente en cada fase. Para resumir el proceso realizado, en la Figura 4.8 se muestra el diagrama de clases para la fase de entrenamiento.

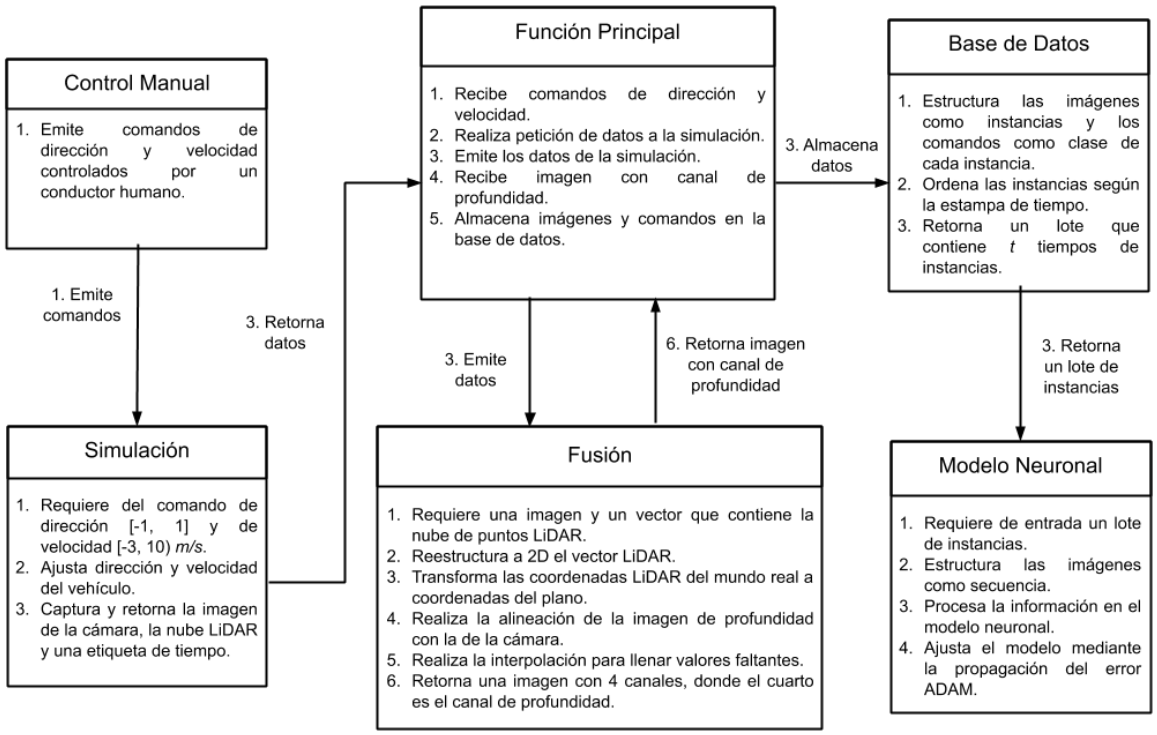


Figura 4.8: Diagrama de clases del sistema propuesto en fase de entrenamiento.

El diagrama de clases en la fase de entrenamiento incluye una función principal que es la encargada de realizar la sincronización de peticiones y retorno de datos. Mostrado como la clase Función Principal en la Figura 4.8, se enumeran los pasos en orden que realiza para interactuar directamente con la Simulación, la Fusión y la Base de Datos. En esta fase existe una clase Control Manual que es la que captura los comandos de velocidad y dirección dados por un conductor humano y sopn emitidos directamente a la simulación, la cual controla el vehículo y captura las imágenes y señales LiDAR. Posteriormente la Función Principal pasa esa información a la clase de Fusión que realiza dicho proceso y retorna la imagen fusionada, que posteriormente se almacena en la base de datos. Como un proceso asíncrono se realiza el entrenamiento del Modelo Neuronal, el cual requiere la lectura de la Base de Datos y se autoajusta como se mencionó en la Sección 4.2.2.

La segunda fase es la de prueba, en la cual se requiere una respuesta en tiempo real para realizar la autoconducción. En la Figura 4.9 se muestra el diagrama de clases correspondiente a esta fase.

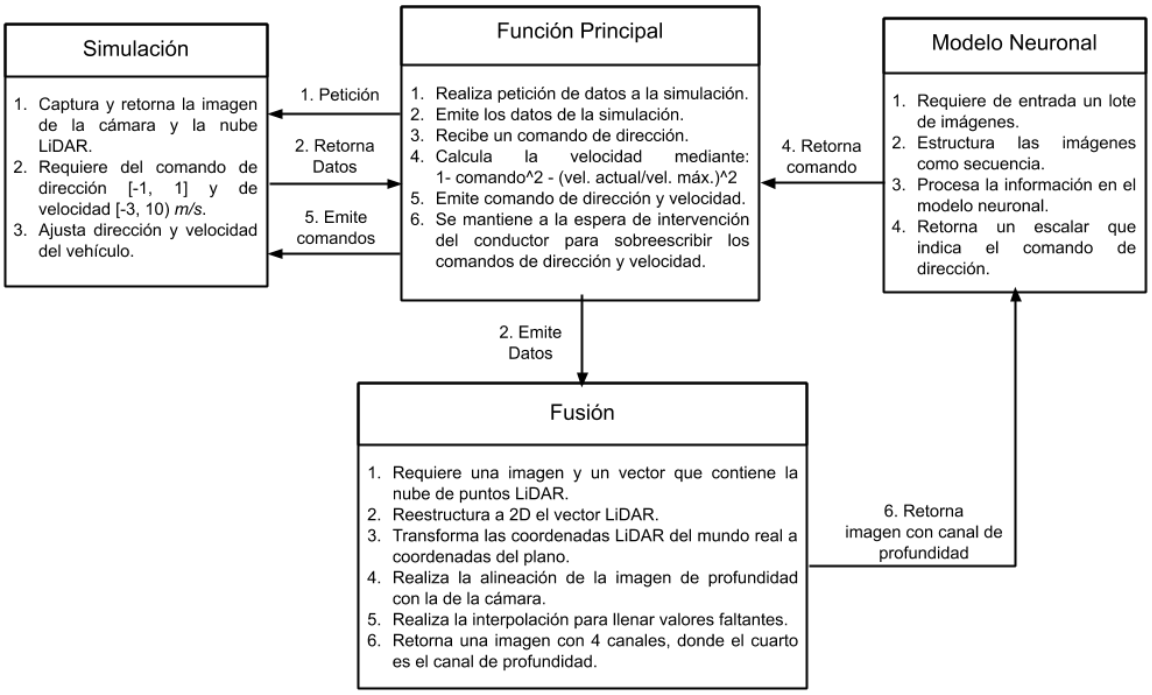


Figura 4.9: Diagrama de clases del sistema propuesto en fase de pruebas.

En esta fase ya no se requiere del control humano, sin embargo es posible que realice ajustes interviniendo directamente en la Función Principal como lo muestra

el paso 6 de esta clase. En esta fase la Función Principal realiza la consulta de datos para emitirla hacia la clase de Fusión, sin embargo ya no es retornada, sino que la imagen fusionada se emite directamente al Modelo Neuronal, que estructura la información en secuencia en orden que se recibe. Una vez procesada la información por el modelo, esta clase emite la predicción hacia la Función Principal, la cual calcula la velocidad del vehículo mediante:

$$vel_t = 1 - \left(y^2 - \left(\frac{vel_{t-1}}{vel_{max}} \right)^2 \right) \tag{4.30}$$

donde y es la predicción del modelo, vel_{t-1} es la velocidad que tiene el vehículo antes de emitir el nuevo comando y vel_{max} la velocidad máxima mencionada en la Sección 4.1.2. Esta información se emite a la simulación en donde se realiza el control del vehículo y se repite el proceso hasta la interrupción del sistema.

Discusión

El modelo propuesto mejora la capacidad del Chauffeur para realizar el control del vehículo al permitir trabajar con secuencias de imágenes, lo que proporciona ventaja al memorizar mayor información del entorno y relacionarla secuencialmente para realizar predicciones en el tiempo presente y futuro, a manera de regresión. Además, el bloque convolucional fue adaptado para procesar imágenes con canal de profundidad obtenidas a partir de la fusión de la cámara con el sensor LiDAR, lo que proporciona información de la proximidad de los objetos en el entorno. La percepción espacial interpretada como información visual aporta la ventaja de detectar y evadir obstáculos encontrados a partir de la extracción visual del bloque convolucional y relacionado al espacio normalizado de distancia en metros con respecto al sensor contenido en el canal de profundidad.

Para demostrar lo antes mencionado se realizaron distintos experimentos en el ambiente simulado de ROS diseñado para esta investigación, se puso realizaron distintos entrenamientos del modelo y se comparó con el modelo base Chauffeur y el modelo Pilotnet de enfoque reactivo que es el más citado en la literatura, utilizando métricas que evalúan la clasificación y la autonomía de la conducción. En el siguiente Capítulo se describen los entornos de pruebas, los diseños de los experimentos, la sintonización de parámetros de entrenamiento de los modelos y las métricas utilizadas y propuestas.

Capítulo 5

Experimentación y resultados

En este Capítulo se describen los requisitos para replicar el sistema propuesto, así como los datos necesarios para realizar los entrenamientos del modelo neuronal y las pruebas en la simulación. También, se detallan los experimentos realizados en el sistema, los métodos con los que se compara y las métricas empleadas para evaluarlo. La evaluación se dividió en dos tareas principales: clasificación, donde se entrenó y evaluó con métricas de error en predicción, y autonomía donde se usa la métrica de (Bojarski et al., 2016) y se proponen dos métodos de medición para autonomía y conducta de conducción. Finalmente se presentan los resultados obtenidos, la discusión y validación de éstos mismos.

5.1. Entorno de desarrollo

Para la implementación del sistema propuesto es necesario el uso de un equipo de cómputo de alto rendimiento, ya que en el mismo equipo se corre la simulación y la red neuronal. Cabe mencionar que es indispensable contar con al menos una unidad de procesamiento gráfico GPU si es que se requiere que el sistema funcione en tiempo real. En cuanto al *software* y librerías utilizadas son estrictas debido a la compatibilidad en conjunto. Debido a que en los experimentos realizados el uso medio de los recursos se mantuvo en 90%, se considera como requisitos mínimos:

Arquitectura de hardware:

- Procesador Intel Core i7-7700 @ 3.60 GHz quad-core.
- 16 GB de memoria RAM.
- Disco de estado sólido de 120 GB para sistema operativo.
- Disco duro de 1 TB para almacenamiento de archivos.
- GPU GTX Titan X con 3072 núcleos CUDA y 12 GB de memoria RAM dedicada.

Arquitectura de software:

- Sistema operativo Ubuntu (o derivados) 14.04 LTS.
- Sistema ROS versión Indigo.
- Simulador Gazebo 7.12 (versión modificada para GPU).
- Lenguaje Python 2.7.
- Framework Tensorflow 1.1.0 (versión GPU).
- Librería de Python Keras 2.2.
- Librería Nvidia librerías CUDA 8.0.
- Librería Nvidia para redes de aprendizaje profundo CuDNN 5.6.

5.2. Datos de entrenamiento y prueba

La base de datos propuesta fue divida en 85% de datos de entrenamiento y 15% de validación, equivalentes a 34,613 y 6,108 instancias respectivamente. Al conjunto de entrenamiento se le aplicó el preprocesamiento mostrado en la Sección 4.2.2 para el incremento de información, esto significa que cada instancia de entrenamiento es única y se incrementa el número a el total de épocas $\times$ iteraciones $\times$ instancias por lote. En cuanto a los parámetros de inicialización del modelo neuronal, se inicializan los filtros w mediante la asignación aleatoria de valores en una distribución normal. De manera resumida, los parámetros de sintonización para el entrenamiento se obtuvieron basándose en las recomendaciones de diversos autores pero sobre todo en la experimentaciones previas. La sintonización utilizada se resume:

- Optimizador ADAM con función de costo Error Cuadrático Medio.
- Tasa de aprendizaje $\alpha = 1.0^{-5}$.
- Probabilidad *dropout* $\mathbb{P} = 0.5$.
- *Timesteps* de recurrencia $t = 3$.
- Lotes de 10 instancias por iteración de entrenamiento.
- 10,000 iteraciones por época.
- 20 épocas de entrenamiento.

5.3. Métodos de comparación

Para la comparación se entrenaron otros dos modelos de red neuronal de aprendizaje profundo. El primer modelo es propuesto por investigadores de la

empresa NVIDIA llamado Pilotnet (Bojarski et al., 2016). Este modelo tiene un enfoque secuencial clásico; en la primera capa se tiene una capa de normalización donde la entrada X es normalizada mediante el método *Batch Normalizing*. Dada una entrada p-dimensional $X = x_1^p, x_2^p, \cdots, x_n^p$ esta transformación implica $BN_{\gamma,\beta}$: $x_i^p \mapsto \gamma \hat{x}_i^p + \beta$ (Ioffe and Szegedy, 2015):

$$BN_{\gamma,\beta}\left(x_i^p\right) \equiv \gamma \hat{x}_i^p + \beta \tag{5.1}$$

donde se conserva la dimensión de X. Las capas convolucionales son operadas con función de activación ReLU y método de reducción de submuestra mostrados en la Sección 4.2.1. Los detalles de los filtros usados y las dimensiones de las capas son detallados en la Figura 5.1 en donde se presenta con las medidas originales con entradas $X^{66\times200}$, sin embargo se adaptó para trabajar con las dimensiones de la base de datos $X^{640\times320}$.

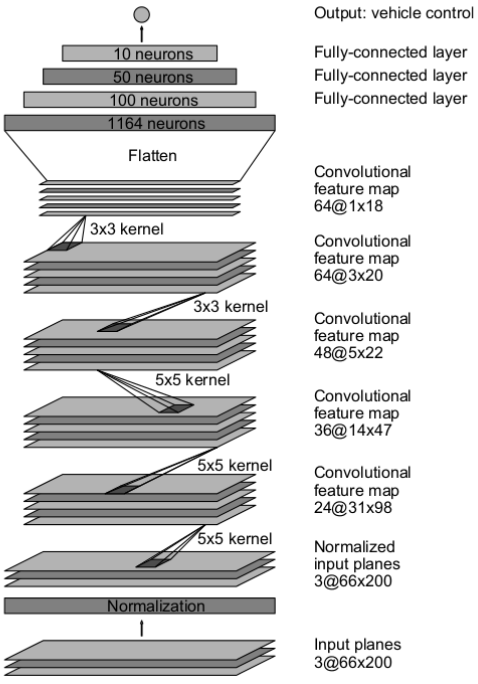


Figura 5.1: Modelo Pilotnet (Bojarski et al., 2016).

Además este modelo tiene la diferencia que la función de activación para las capas de Perceptrón Multicapa es la función ReLU y una Tangente Hiperbólica (4.17) para la salida. Este modelo funciona solamente con imágenes de 3 canales de color, por lo que el canal de profundidad de las imágenes de la base de datos propuesta es truncado. La sintonización de parámetros para este modelo es:

- Optimizador ADAM con función de costo Error Cuadrático Medio.
- Tasa de aprendizaje $\alpha = 1.0^{-5}$.
- Lotes de 40 instancias por iteración de entrenamiento.
- 2,000 iteraciones por época.
- 20 épocas de entrenamiento.

El segundo método de comparación es el mismo modelo Chauffeur base mostrado en la Figura 4.5. De igual manera este modelo solo funciona con 3 canales de color y un solo *timestep* t, es decir una sola imagen. La sintonización de este modelo es la siguiente:

- Optimizador ADAM con función de costo Error Cuadrático Medio.
- Tasa de aprendizaje $\alpha = 1.0^{-4}$.
- Probabilidad *dropout* $\mathbb{P} = 0.5$.
- *Timesteps* de recurrencia $t = 1$.
- Lotes de 40 instancias por iteración de entrenamiento.
- 10,000 iteraciones por época.
- 20 épocas de entrenamiento.

5.4. Experimentos

Los experimentos realizados para evaluar la autonomía se dividieron en tres tareas: conducción en camino libre, camino con obstáculos y el recorrido completo de un camino con las anteriores características.

Camino libre

En este experimento se mide la capacidad de mantener al vehículo en el carril derecho de forma adecuada. Esta tarea se realizó inicialmente en caminos totalmente rectos como los que se muestran en la Figura 5.2, posteriormente en caminos con curvas como el mostrado en la Figura 5.3.

Figura 5.2: Ejemplo de caminos rectos.

La evaluación se realiza de forma asistida, esto quiere decir que si el vehículo toma rumbo a una colisión o al carril contrario el conductor debe realizar el movimiento del volante para evitar la situación de riesgo. En esta prueba se le da mayor importancia a la conducta de conducción, la cual indica si el vehículo realiza los movimientos de forma suavizada como lo haría un conductor promedio, esta medición se detalla en la Subsección 5.5.3.

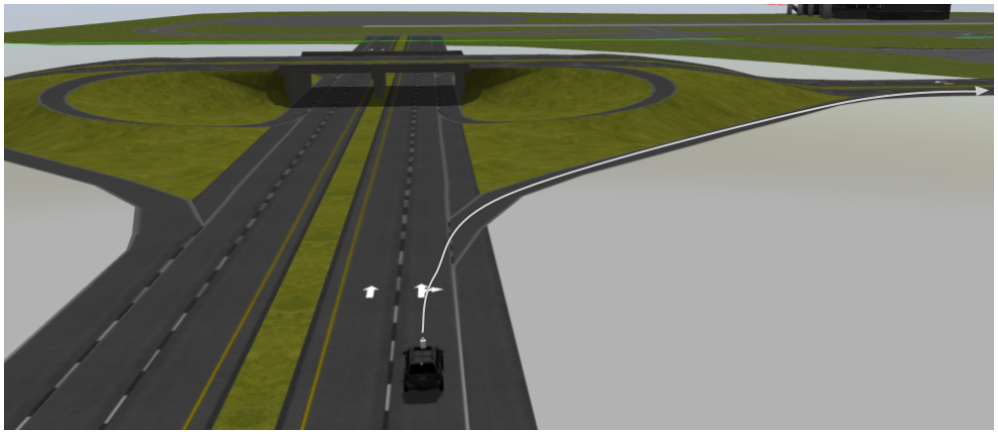


Figura 5.3: Ejemplo de camino con curva.

La experimentación completa se realiza en un total de 12 caminos rectos con un total aproximado de distancia de 15 km y en 8 caminos compuestos por curvas con un total aproximado de 6 km. Adicionalmente se realizó la prueba de cruce en intersecciones sin obstrucción similares a las mostradas en la Figura 5.2. En total fueron 5 intersecciones de aproximadamente 100 metros cada una.

Obstáculos en el camino

La segunda experimentación se realiza en escenarios con obstrucciones en el camino. Este tipo de escenarios son los que ponen a prueba la autonomía y se espera que la información de distancia ayude a evitar las colisiones. Esta experimentación se divide en caminos con obstrucciones parciales como los mostrados en la Figura 5.4, donde un objeto obstruye parte del único camino disponible.

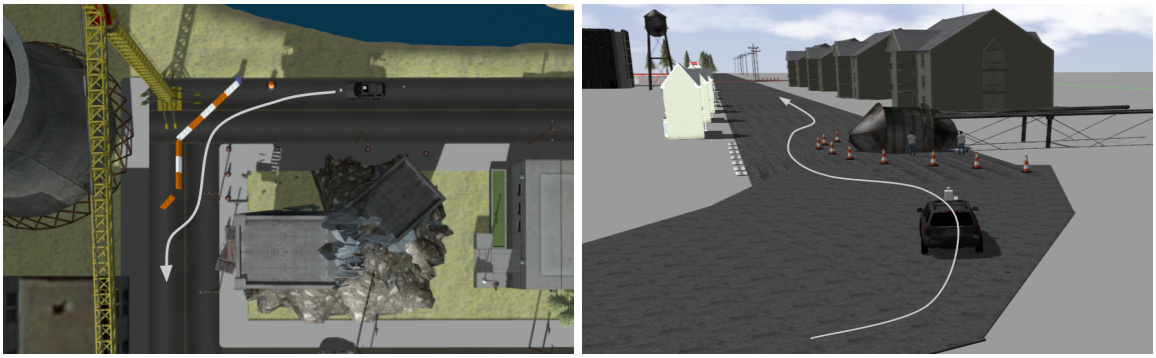


Figura 5.4: Caminos con obstrucción parcial.

El segundo tipo es la obstrucción total de un camino como se muestra en la Figura 5.5. En este existe una intersección donde un camino está totalmente obstruido pero existe otro por el cual transitar. Estos tipos de pruebas son breves ya que solo se espera que realice el control adecuado para evitar el obstáculo, en total son 6 caminos obstruidos parcialmente y 4 totalmente. Los escenarios para las pruebas varían entre 100 y 200 metros cada uno.

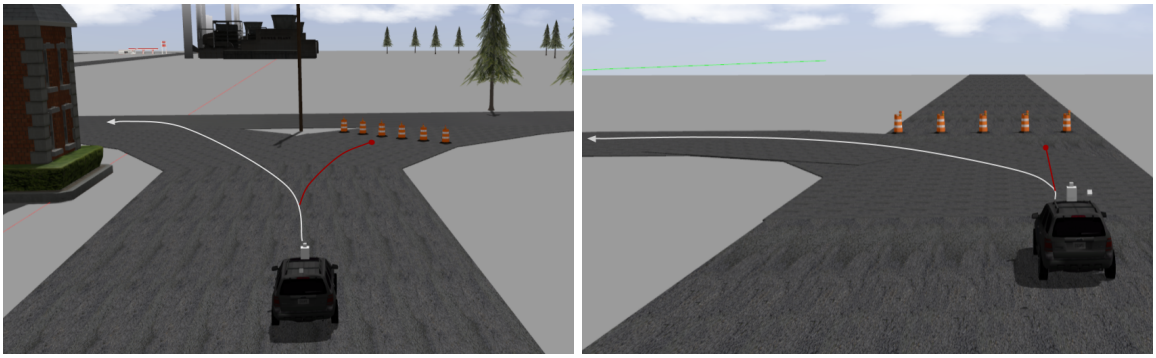


Figura 5.5: Escenario con obstrucción total en un camino.

Autonomía total

En esta última prueba el vehículo es evaluado en un escenario con caminos obstruidos parcialmente y con algunas curvas. En la Figura 5.6 se muestra el escenario *VRC Task 1* (Figura 4.3(d)) del cual no se tiene contemplado en la base de datos de entrenamiento. El camino está marcado con una línea continua blanca, la cual indica el recorrido esperado del vehículo. Este escenario mide 734 metros de longitud y tiene 2 carriles equivalente a 3.5 metros de anchura.

Figura 5.6: Prueba final en el escenario *VRC Task 1* (Sardinha, 2017).

5.5. Métricas

La evaluación del sistema propuesto se divide en tres partes: calidad de la base de datos propuesta, calidad de predicción del modelo y la autonomía de conducción que otorga el control de este. La primera parte refiere a la calidad de la fusión de los sensores para generar las imágenes RGB-D. El segundo tipo de evaluación recae sobre las fases de entrenamiento y validación del modelo, esta evaluación es diferente a las de otros debido a la naturaleza de los datos, los cuales se encuentran en un dominio de clase continua. El tercer tipo de evaluación refiere a que tan seguro es que el modelo conduzca el vehículo.

5.5.1. Calidad de la fusión

En primera instancia es necesario evaluar la calidad de la fusión de los sensores utilizados para generar la base de datos. Para realizar la evaluación se comparó la imagen de profundidad obtenida con el sensor LiDAR con la cámara estereoscópica de ROS ubicada en la posición de la cámara RGB. Para evaluar la calidad de ambas se calculó el Error Absoluto Medio (5.2) donde x_i representa cada píxel de la imagen de profundidad de la cámara estereoscópica y x'_i la imagen obtenida del LiDAR. Esta métrica obtiene valores en el intervalo $[0,1]$, aproximándose a 1 cuanto más se parezcan las imágenes, el caso contrario tiende a 0 cuanto más disimiles sean las imágenes.

$$MAE = \frac{1}{N}\sum_{i=0}^{n}|x_i - x'_i| \tag{5.2}$$

5.5.2. Métricas de clasificación

Como se mencionó anteriormente, no se tienen clases definidas categóricamente sino que en un dominio continuo, por lo que utilizar las métricas derivadas de la matriz de confusión (*accuracy, recall*, etc.) no es la forma adecuada de cuantificar la correcta clasificación del modelo. El uso de la función de pérdida utilizada para el cálculo del gradiente en la propagación del error es un indicador de calidad en el aprendizaje de la red neuronal. El Error Cuadrático Medio (5.3) utilizado como función de pérdida en el método ADAM es utilizado como métrica de evaluación en la predicción ya que es una medida de distancia donde y_i es la clase esperada y y_i^p es la predicción de la red neuronal.

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(y_i - y_i^p)^2 \tag{5.3}$$

En esta métrica un valor de 0 indica que la predicción y el valor esperado son idénticos. También se hace el cálculo de la función Logaritmo del Coseno Hiperbólico (*logcosh*) (5.4) como métrica para el ritmo de aprendizaje de la red neuronal. Al igual que la función MSE, un gradiente que tiende a 0 representa una menor tasa de error. Funciona muy similar el error cuadrático medio, pero no se vé tan fuertemente afectado por la predicción incorrecta con valores en los extremos.

$$logcosh = \sum_{i=0}^{n} log\left(\frac{e^{(y_i^p - y_i)} + e^{-(y_i^p - y_i)}}{2}\right) \tag{5.4}$$

Otra forma interesante de evaluar a la red neuronal con clase continua es mediante la distancia coseno (5.5). A diferencia de otras aplicaciones de aprendizaje profundo en donde se tiene clase binaria o multiclase discreta (identificación de objetos, clasificación de imágenes, predicción de textos, etc.), en la autoconducción se tiene una clase en el dominio continuo ubicado en el intervalo $[-1,1]$. En un caso específico, la clase real $y = -0.01493108$ y la predicción $y^p = -0.01493106$ son diferentes y evaluar la predicción en términos de falsos positivos, etc., presentaría errores en la mayoría de las ocasiones y esto es simplemente innecesario. Sin embargo la distancia coseno obtiene el error en el intervalo $[-1,1]$ donde el valor de $\theta = 0.0$ representa una predicción exacta a la esperada. Además indica mediante el signo de θ hacia donde tiende el error y si los parámetros w deben incrementar o decrementar.

$$cos(\theta) = \frac{\sum_{i=0}^{n} y_i \cdot y_i^p}{\sqrt{\sum_{i=0}^{n}(y_i)^2} * \sqrt{\sum_{i=0}^{n}(y_i^p)^2}} \tag{5.5}$$

5.5.3. Métricas de autonomía

En esta aplicación, las métricas necesarias para evaluar la correcta predicción son aquellas que miden la autonomía obtenida en la conducción del modelo neuronal. La primera métrica de autonomía es la propuesta (Bojarski et al., 2016) mostrada en (5.6). Esta métrica toma en cuenta el número de intervenciones que realiza el conductor para evitar una colisión o descarrilamiento del vehículo simulado, así como el tiempo total obtenido en la prueba expresado en segundos. Los 6 segundos que multiplica al número de intervenciones se obtiene de un estudio estadístico que realizaron los investigadores de Nvidia en donde se demostró que un conductor demora ese tiempo en corregir la dirección del vehículo.

$$Autonomía = \left(1 - \frac{intervenciones \cdot 6\ segundos}{tiempo\ transcurrido}\right) \cdot 100 \tag{5.6}$$

Como segunda métrica de autonomía se propone la métrica Tiempo Absoluto de Autonomía (5.7). Esta métrica se obtiene calculando solamente el tiempo de intervención sobre el tiempo total de la prueba, indistintamente del número de intervenciones. Ya que las intervenciones varían dependiendo de la forma del camino, el tiempo de una intervención se debe calcular de una forma absoluta.

$$T.Autonomía = \left(1 - \frac{tiempo\ de\ intervención}{tiempo\ transcurrido}\right) \cdot 100 \tag{5.7}$$

En anteriores experimentaciones se observó que al comparar los resultados cuantitativos obtenidos con los anteriores métodos de evaluación con los resultados cualitativos se encontró una debilidad en esta forma de evaluación. Aunque las métricas reporten resultados muy buenos, la conducta que toma el vehículo se presenta errática, cambiando de carril y haciendo movimientos en forma de zigzag. En la literatura no se han reportado a la fecha métodos de evaluación para la conducta de conducción, por lo que bajo la recomendación del comité revisor se diseñó una métrica basada en la desviación estándar (5.8) (Spiegel, 1991) de los ángulos de dirección captados del volante x (clase) en los datos de entrenamiento s menos los obtenidos por la conducción de la red neuronal s', esta expresión se presenta la en (5.9).

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})}{N}} \tag{5.8}$$

$$conducta = |s - s'| \tag{5.9}$$

Esta métrica mide la disimilitud de la conducción autónoma con la conducción realizada por un conductor humano. Para funcionar se requiere de los datos de conducción de referencia que comprenden en un intervalo $[-1,1] \in \mathbb{R}$. Para ambos se calcula la desviación estándar, posteriormente se obtiene la diferencia absoluta entre ambos para obtener la disimilitud, donde un valor de 2.0 representa el caso extremo donde los datos son totalmente disímiles.

5.6. Resultados

5.6.1. Calidad de fusión

Se realizó un experimento obteniendo 1000 muestras con la fusión de sensores y con la cámara estereoscópica, ambos sensores se posicionaron para capturar la misma escena y así poder calcular la similitud entre estas. Posteriormente se graficó el histograma de distancias mostrado en la Figura 5.7. La distribución se muestra ligeramente sesgada a la derecha y presenta una desviación estándar de $s = 0.0173$ con media en $\tilde{x} = 0.0815$.

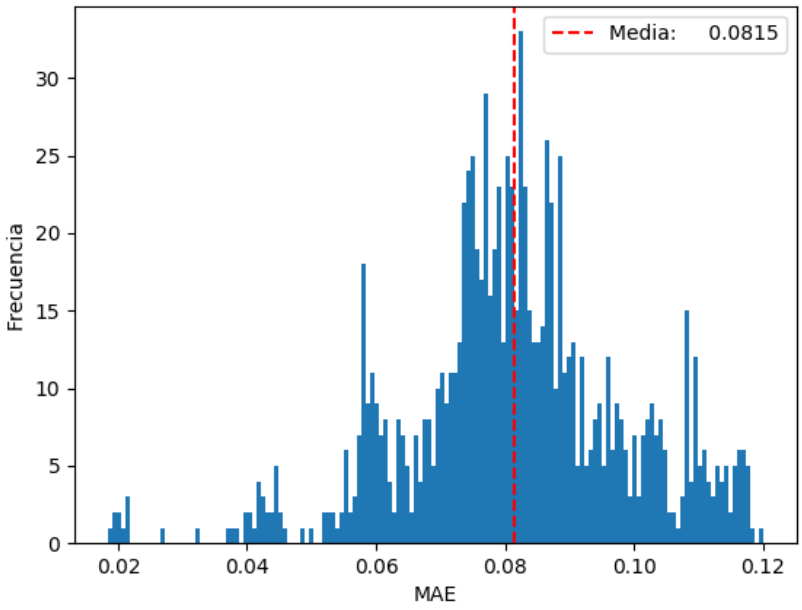
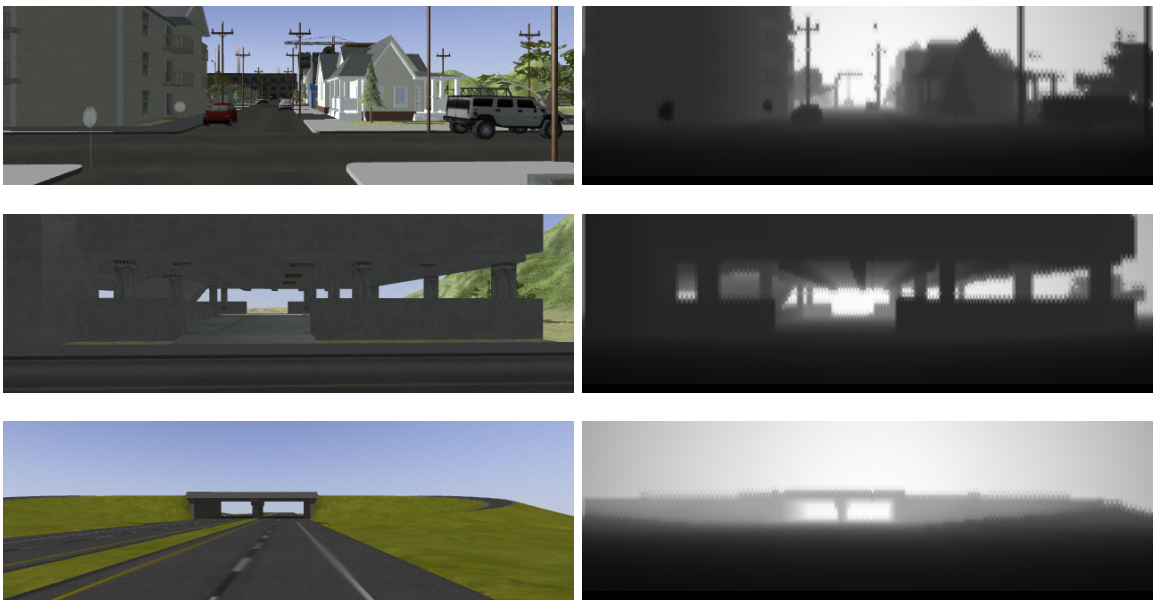


Figura 5.7: Distribución del error absoluto medio en la fusión de sensores.

Esta distribución de errores indica que la fusión de cámara y LiDAR da como resultado una imagen de profundidad que contiene pequeñas diferencias con respecto a una cámara estéreo clásica. El índice de error se concentra principalmente en los objetos que se encuentran a más de 100 metros de distancia, que es el máximo alcance configurado para el sensor LiDAR como se muestra en la comparación cualitativa de la Figura 5.8.



(e) Imagen a color. (f) Canal de profundidad.

Figura 5.8: Comparación cualitativa de la imagen a color y el canal de profundidad.

5.6.2. Clasificación

La evaluación de clasificación se divide en la etapa de entrenamiento y validación. Cada época de entrenamiento se intercala con una etapa de validación, la cual toma el 15% de los datos que no se usaron para entrenar. De cada modelo se registró la función de costo final de cada época de entrenamiento y el total en validación, los resultados se muestran en la Figura 5.9. En estas gráficas se visualiza la diferencia en la calidad del aprendizaje en la etapa de entrenamiento, donde el modelo mejorado Chauffeur RGBD obtiene desde el inicio el menor error y se aprecia un gradiente descendente estable, a diferencia de los otros dos modelos en los que el gradiente parece inestable.

En la etapa de validación se aprecia un error muy variable en todos los modelos, este fenómeno ocurre principalmente por la validación cruzada implícita en los entrenamientos. Debido a que en cada época se utiliza solo una porción del total de los datos de entrenamiento, ciertas situaciones que aparecen en la validación no se consideraron en el entrenamiento. Este se va estabilizando con el paso de las épocas de entrenamiento.

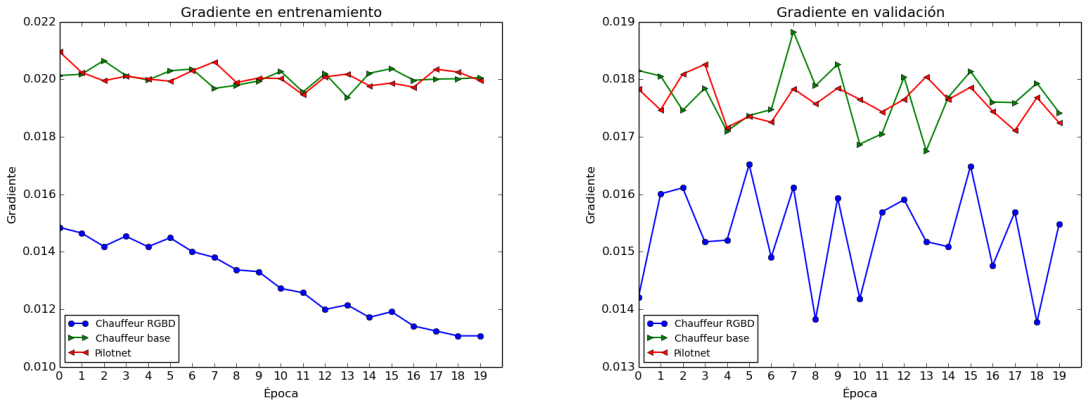


Figura 5.9: Función de pérdida en entrenamiento y validación.

Aún así se aprecia que el error del modelo mejorado es menor aunque más inestable. Esto indica que al manejar más información (el canal de profundidad que no usan los demás modelos), se requiere de un entrenamiento más extenso para obtener mejores resultados.

En el caso de la métrica *logcosh* mostrada en la Figura 5.10, presenta la misma tendencia ya que no se dan predicciones incorrectas muy dispersas, lo que no afecta la función de costo. Sin embargo si se observa el eje Y, los índices de la métrica se concentran en un mismo intervalo, esto es usado para comparar la validación con respecto a los entrenamientos (Zhu et al., 2016).

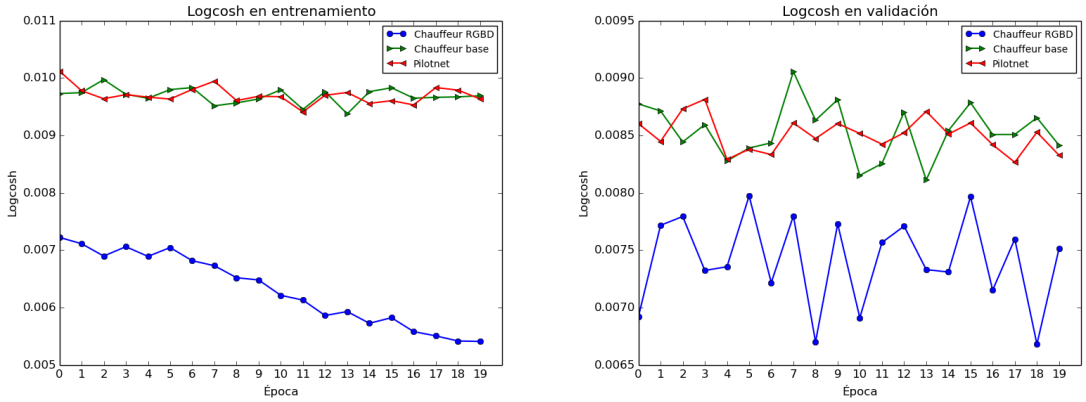


Figura 5.10: Métrica *logcosh* en entrenamiento y validación.

Tomando en cuenta lo antes mencionado, se observa en las líneas de color azul correspondientes al modelo mejorado como en la época 0 la validación y el

entrenamiento son muy similares, sin embargo en las posteriores épocas el error en validación es mayor que el entrenamiento, esto indica que el entrenamiento requiere de más datos por época. El caso contrario se observa con los otros dos modelos, que presentan errores menores en validación, lo que indica que la relación de datos por época es buena. Para mejorar esta métrica en el caso del modelo propuesto es necesario considerar una sintonización de hiperparámetros diferente que mejore estos resultados.

En cuanto a la distancia coseno mostrada en la Figura 5.11 se aprecia que en el entrenamiento la distancia se encuentra próxima al 0, que es lo ideal. Sin embargo con el paso de las épocas se dispersan los valores, esto debido al incremento de información que se procesa. En entrenamiento el modelo propuesto presenta los mejores resultados.

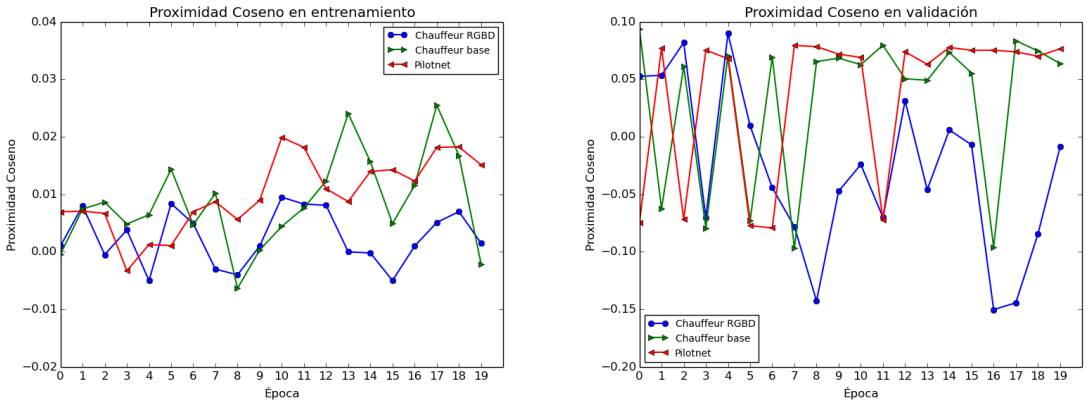


Figura 5.11: Distancia coseno en etapa de entrenamiento y validación.

En cuanto a la validación, es común que los valores oscilan en las primeras épocas ya que no se tiene un buen ajuste de los parámetros de la red. En el caso del modelo Pilotnet y Chauffeur base se observa que a partir de la época 7 los valores se estabilizan con ocasionales saltos, esto se puede deber a que el modelo se ajusta de forma rápida. Por otra parte el modelo propuesto sigue oscilando ya que procesa mayor información, sin embargo en algunas épocas muestra una distancia muy cercana a 0, lo que indica que el ajuste se está realizando con mayor precisión pero requiere de más entrenamiento.

Discusión

Un punto destacable es que el modelo Chauffeur mejorado tardó inicialmente 6.5 días en entrenar 20 épocas, en contraste al modelo Chauffeur base que tardó 2.4 días y el modelo Pilotnet que tardó 16 horas. La explicación es que en el modelo propuesto se opera con un canal de color extra y se expande la sucesión de imágenes a procesar. Se tuvo como limitante una sucesión de $t = 3$ *timesteps* con 5 instancias por lote de entrenamiento ya que más de esto desborda los 12 GB de memoria de la GPU. Los resultados obtenidos demuestran que el modelo requiere de más iteraciones y épocas de entrenamiento ya que está procesando una mayor cantidad de datos, sin embargo se ve limitado por los recursos de hardware disponibles para esta experimentación.

Sin embargo, aunque los resultados de clasificación no son óptimos, los resultados de autonomía presentan mejoría con respecto a los modelo de comparación.

5.6.3. Autonomía

Tomando los modelos entrenados antes mostrados, se realizaron los experimentos mencionados en la Sección 5.4. Internamente en el controlador, se capturaron los datos para realizar la evaluación con las métricas de autonomía, permitiendo al usuario desactivar el control autónomo para realizar ajustes y evitar colisiones. Los resultados mostrados en esta sección representan la media de los resultados obtenidos en todos los experimentos, a manera de representar los resultados de forma ordenada. Adicionalmente, se incluyó una columna de intervenciones, la cual indica las ocasiones (media) en que el usuario desactiva el control autónomo, con motivo de tener una referencia para relacionar cada métrica.

El primer bloque de experimentos es el de caminos libres, donde se realizaron 25 experimentos en total incluyendo rectas, curvas e intersecciones. Los resultados se resumen en la Tabla 5.1, en esta se aprecia que el modelo Pilotnet requiere en promedio 1.72 intervenciones por ocasión, lo que significa que requirió de un total de 43 intervenciones distribuidas en los 25 experimentos. Comparando la métrica de autonomía de (Bojarski et al., 2017) con la propuesta de tiempo de autonomía, esta última devuelve mejores resultados debido a que las intervenciones son breves, en el caso del modelo propuesto Chauffeur RGBD ambas métricas son muy similares ya que las intervenciones realizadas son aproximadas a los 6 segundos esperados

de la métrica de autonomía. En todo caso, para esta tarea el modelo propuesto obtuvo el mayor índice de autonomía.

Tabla 5.1: Resultados en camino libre.

Modelo	Intervenciones	Autonomía (%)	T. Autonomía (%)	Conducta
Pilotnet	*1.72*	*79.50*	*89.79*	0.0163
Chauffeur	0.96	84.44	92.36	*0.0294*
Chauffeur RGBD	**0.24**	**97.41**	**98.29**	**0.0086**

Los experimentos con obstrucciones se dividieron en dos evaluaciones, ya que en estos la complejidad es diferente para cada uno. Los resultados para la evaluación con obstrucción parcial del camino son mostrados en la Tabla 5.2.

Tabla 5.2: Resultados en camino con obstrucción parcial.

Modelo	Intervenciones	Autonomía (%)	T. Autonomía (%)	Conducta
Pilotnet	1.0	75.11	90.26	0.0622
Chauffeur	*1.16*	*53.68*	*80.85*	*0.2402*
Chauffeur RGBD	**0.33**	**87.48**	**91.58**	**0.0197**

A diferencia de los experimentos con caminos libres, en la prueba con obstrucción parcial el modelo Chauffeur base obtuvo los resultados más bajos. Esto debido a que se requirió de un número mayor de intervenciones y en un escenario fue incapaz de realizar la tarea exitosamente.

En cuanto a la conducción con un camino totalmente obstruido mostrado en la Tabla 5.3, los resultados de Pilotnet y Chauffeur base requieren de las mismas intervenciones y completan el recorrido en tiempo similar, por lo que las métricas son muy similares, donde el tiempo de autonomía difiere debida a que la intervención con Chauffeur base es más breve. En cuanto a la versión mejorada no requiere de intervenciones. Los resultados en esta tabla parecen ser mejores a comparación de las demás, esto se debe a que esta experimentación es la más corta con apenas 4 escenarios y rutas muy breves.

Tabla 5.3: Resultados en camino con obstrucción total.

Modelo	Intervenciones	Autonomía (%)	T. Autonomía (%)	Conducta
Pilotnet	*0.25*	*94.68*	*97.93*	*0.0479*
Chauffeur	*0.25*	94.75	99.33	0.0250
Chauffeur RGBD	**0.0**	**100.0**	**100.0**	**0.0035**

Finalmente se muestra en la Tabla 5.4 la prueba en el escenario completo *VRC Task 1*, el cual contiene la mayoría de las características de los experimentos antes evaluados. En esta evaluación al hacerse en una única ocasión son absolutos y no una media. El modelo mejorado claramente presenta los mejores resultados con una autonomía casi perfecta para esta combinación de tareas.

Tabla 5.4: Resultados en escenario *VRC Task 1*.

Modelo	Intervenciones	Autonomía (%)	T. Autonomía (%)	Conducta
Pilotnet	*5*	*75.13*	*85.23*	*0.0977*
Chauffeur	3	86.18	94.24	0.0684
Chauffeur RGBD	**1**	**95.14**	**97.30**	**0.0181**

Validación de resultados

Para tener la certeza de que los resultados de autonomía antes mostrados son producto de una correcta implementación del modelo propuesto y no de la aleatoriedad, es necesario hacer un análisis de validación sobre la totalidad de los mismos. El término estadísticamente significativo se percibe como una etiqueta que indica garantía de calidad, por lo que la significancia estadística fue aplicada a los resultados de los experimentos a partir de un nivel de significancia establecido en $\alpha = 0.05$, como recomienda (Kline, 2013).

La prueba de hipótesis es un test de significación estadística que cuantifica hasta qué punto la variabilidad de la muestra puede ser responsable de los resultados de un estudio en particular, donde H_o (hipótesis nula) representa la afirmación de que no hay asociación entre las variables estudiadas y la H_a (hipótesis alternativa) afirma que hay algún grado de relación o asociación entre las variables.

Por lo tanto, para esta validación se toman en cuenta las siguientes hipótesis:

- H_o: la autonomía es prácticamente igual.
- H_a: existe mejoría considerable con el modelo propuesto.

Es necesario considerar el total de experimentos realizados en los distintos escenarios con cada método. En total se realizaron 31 experimentos con cada uno de los 3 métodos, por lo que se considera $N = 93$. Otra consideración importante es observar que los resultados varían en intervalos indefinidos, por lo que es necesario

normalizar las premisas $P_i \mapsto [0,1]$, así como el resultado a evaluar $P_T \mapsto [0,1]$.

Para realizar la validación es necesario demostrar que la diferencia absoluta de las premisas y el resultado a evaluar es mayor al producto del error estándar de p y el nivel de significancia, de ser así se puede afirmar que los resultados son estadísticamente significativos. En primera instancia es necesario calcular la media:

$$\bar{a} = \frac{1}{d} \sum_{i=1}^{d} |P_i - P_T| \tag{5.10}$$

donde d es definido en 2 y representa el total de premisas (métodos de comparación). Para este caso el valor es de $\bar{a} = 0.279$, tomando en cuenta las 4 métricas de evaluación normalizadas. Posteriormente se realiza el calculo de p:

$$p = \frac{\sum_{i=1}^{d} P_i}{d} \tag{5.11}$$

obteniendo $p = 0.317$. El siguiente paso es calcular el error estándar de p:

$$\sigma(p) = \sqrt{p(1-p)(d \cdot \frac{1}{N})} \tag{5.12}$$

donde se obtiene un valor de $\sigma(p) = 0.083$. Este resultado es multiplicado por el nivel de significancia escalado por un valor Z definido por (Pita Fernández and Pértega Díaz, 2000) $Z_{\alpha=0.05} = 1.96$. El producto da como resultado la conclusión $p = 0.163$, con lo que se puede concluir que $\bar{a} > p \Rightarrow H_a$, por lo tanto se rechaza la hipótesis nula H_o y se acepta la hipótesis H_a: .^existe mejoría considerable con el modelo propuesto"; demostrando que los resultados **si son estadísticamente significativos**, a la vez que el error estándar $\sigma(p) < \alpha$ indica una seguridad del 95%.

Discusión

Tomando en cuenta la combinación de las métricas antes obtenidas en todos los experimentos, se obtiene que el modelo Pilotnet es autónomo en un 85.95% en este entorno de experimentación. Por otro lado el modelo Chauffeur base obtiene una media de autonomía de 85.72%, muy similar al modelo Pilotnet debido a que en ciertas tareas se comportaba de forma errática. Por último el modelo mejorado Chauffeur RGBD obtuvo una autonomía de 95.90%, por lo que se concluye que esta

propuesta mejora $\approx 10\%$ la autonomía que proporcionan los modelos evaluados. Los anteriores resultados se visualizan en la Figura 5.12.

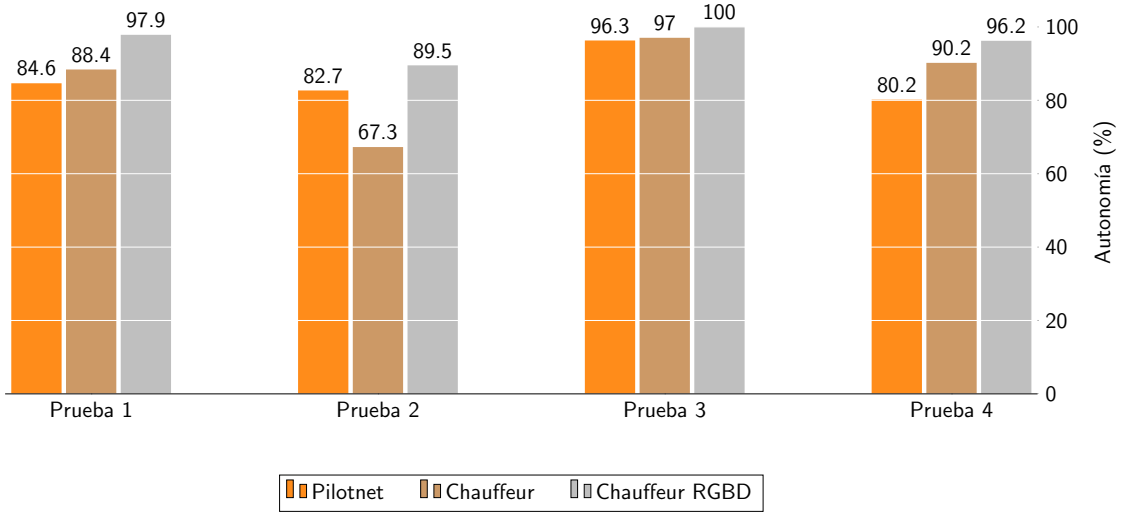


Figura 5.12: Resumen de los resultados de autonomía.

Con base en los resultados expuestos se puede concluir que el modelo híbrido propuesto realiza una conducción más precisa que los modelos reactivos. De la misma forma el agregar el canal de profundidad es de gran utilidad para evadir obstáculos, sobre todo en escenarios donde existe una obstrucción parcial del camino, el cual es la prueba más difícil ya que es necesario controlar el vehículo en un espacio más reducido.

Dada la evidencia, en el siguiente capítulo se expresan las conclusiones de la investigación. De la misma manera se muestra la comparación objetivos y alcances planteados al inicio frente a las actividades realizadas para cumplirlos, así como los productos derivados de la investigación.

Capítulo 6

Conclusiones

En este Capítulo se realiza un análisis detallado de las conclusiones a las que se llegaron después de la evaluación del algoritmo y de estudiar los resultados obtenidos, con respecto a los objetivos y alcances planteados. También se hace un análisis para trabajos futuros y sugerencias sobre el tema.

6.1. Objetivos y alcances logrados

En la Tabla 6.1, se muestran las actividades realizadas en relación a los objetivos y alcances planteados para el desarrollo de la tesis.

Tabla 6.1: Objetivos y alcances realizados.

Objetivo	Actividad
Desarrollar la simulación del vehículo en ROS y dotarlo de sensores disponibles.	Se integró el modelo para el vehículo y se le acoplaron cuatro tipos de sensores a este. Además se integraron diversos ambientes urbanos para pruebas.
Estudiar y comprender las Redes Neuronales Recurrentes basadas en aprendizaje profundo, sus variantes y sus aplicaciones en el reconocimiento de objetos y toma de decisiones.	Se realizó el estudio del marco conceptual y estado del arte de las Redes Neuronales de Aprendizaje Profundo, se redactó un reporte del estado del arte.
Utilizar bancos de imágenes y datos de sensores capturados durante la conducción de un automóvil real y/o simulado para el entrenamiento del modelo.	Se probaron diversas bases de datos, sin embargo los mejores resultados se dieron con una base de datos creada a partir de la simulación.

Integrar el modelo a la simulación del vehículo para permitir que este realice las acciones control de dirección y velocidad del vehículo.	Se realizó la implementación del modelo y la comunicación con el vehículo mediante librerías de Python y ROS.
Evaluar el desempeño del vehículo y su autonomía obtenida con el modelo.	Se dividió la evaluación en el dominio de clasificación para el modelo neuronal y de autonomía para el control del vehículo.
Alcances	**Actividad**
Desarrollar la simulación del vehículo en ROS.	Se diseño e integró un modelo de vehículo CUV Ford Escape basado en el de (CPS-VO, 2012)
Definir la arquitectura del modelo.	Se propuso una variante mejorada del modelo Chauffeur (Tian et al., 2017) que mejora al modelo base.
Entrenar el modelo para realizar las acciones de viraje y control de velocidad.	Se diseñó el modelo y se entrenó con la base de datos propuesta.
Experimentar con el vehículo en un ambiente simulado estático.	La evaluación de autonomía se realizó experimentando en los ambientes integrados.
Evaluar la autonomía del vehículo con base en las métricas definidas en la literatura.	Se evaluó la autonomía con la métrica propuesta por (Bojarski et al., 2017) y se propusieron dos evaluaciones de autonomía.

6.2. Resultados del trabajo

6.2.1. Productos

Durante el desarrollo de este proyecto se obtuvieron los siguientes productos:

1. Reporte del estado del arte: documento en el que se presenta un resumen de diversas publicaciones recientes que tratan los tópicos tratados en el proyecto como lo es redes neuronales profundas, autoconducción, fusión sensorial y bases de datos.

2. Simulación del vehículo autónomo y entornos: archivos fuente de los modelos de simulación en el sistema ROS y de integración del modelo de red neuronal. También se agrega una versión modificada del sistema Gazebo para trabajar las nubes de puntos LiDAR con GPU.

3. Diseño e implementación del modelo Chauffeur mejorado: códigos fuente para la construcción, entrenamiento y pruebas del modelo mejorado propuesto.

4. Base de datos de conducción en ambiente simulado con imágenes RGB-D.

5. Poster presentado en la Escuela de Inteligencia Artificial y Robótica 2018 de la UTEZ y congreso ICMEAE 2018 (Figura A.1 y A.3).

6. Artículos para la Segunda y Tercera Jornada de Ciencia y Tecnología Aplicada del CENIDET (Figura A.4 y A.6).

7. Artículo para la Escuela de Inteligencia Artificial y Robótica 2019 de la UTEZ (Figura A.5).

8. Artículo para el congreso CORE del CIC-IPN (Figura A.9).

9. Artículo para la revista *Applied Soft Computing* editorial Elsevier (Figura A.10).

6.2.2. Aportaciones

Como aportaciones al tema de los vehículos autónomos y al marco teórico se obtuvieron:

1. Sistema completo de simulación y control de un vehículo autónomo de libre acceso.

2. Base de datos con imágenes RGB-D de conducción.

3. Versión compatible con GPU del simulador Gazebo.

4. Mejora al modelo Chauffeur proponiendo una variante distribuida en el tiempo, la cual procesa secuencias de imágenes RGB-D y que otorga una autonomía 10.18 % mayor a su forma base.

6.2.3. Conclusiones

En este trabajo se presentó una versión de una Red Neuronal Convolucional Recurrente mejorada para trabajar con secuencias de imágenes con canal de profundidad. Este modelo de red neuronal se comparó con su forma base y con el modelo reactivo más citado en la literatura, la experimentación demostró que el modelo reactivo ofrece una autonomía media de 85.95%, el modelo base 85.72% y el propuesto 95.90%, proporcionando una mejoría 9.95% con respecto a la máxima. Las métricas de autonomía utilizadas son la propuesta por (Bojarski et al., 2016) y basada en esta se propone una variante que mide el tiempo de autonomía. Adicionalmente se propone un método de evaluación de la calidad de conducción, la cual penaliza cuando la autonomía presenta comportamientos diferentes a los que realiza un conductor humano promedio.

La evaluación de estos modelos neuronales se realizó en un entorno de simulación diseñado en el Sistema Operativo Robótico que incluye la simulación de un vehículo, sensores y entornos de prueba a escala de un entorno real. Adicionalmente se creó una base de datos de conducción en esta simulación a partir de la fusión de sensores para obtener imágenes a color con canal de profundidad, las cuales fueron utilizadas para entrenar los modelos. La fusión de datos presenta un error medio de 0.08 y máximo de 0.12, lo que indica que la información obtenida es confiable en un 92% en la mayoría de los casos y 88% en el peor de los casos.

La medición de los resultados se dividieron en clasificación de la red y autonomía del vehículo. Para evaluar la clasificación se utilizaron métricas que miden el error en lugar de los índices de acierto, ya que el dominio de la clase es basado en aproximaciones y no en clases discretas como en aplicaciones de detección. El resultado de estas aproximaciones basadas en error demostraron una reducción ≈ 0.01 con respecto a la mínima, que en términos de error cuadrático respalda la mejoría de autonomia del 10%.

El sistema fue probado de forma equitativa y el modelo propuesto obtuvo los mejores resultados en autonomía para la tarea de control de dirección. Los índices de autonomía se ubican por encima del 95%. Estos resultados dan evidencia de la comprobación de la hipótesis planteada en la propuesta del tema, la cual dice que la autoconducción debe ser tratada como un problema dependiente del tiempo y

no con una solución reactiva. También se observó que el uso de diversos sensores ayuda a mejorar la autonomía.

6.2.4. Trabajo futuro

Como trabajo futuro directo a este proyecto se propone ampliar el campo de visión del sistema mediante la fusión de varias cámaras en distintos ángulos y el uso del LiDAR en 360° para realizar maniobras como estacionar el vehículo y detectar objetos en un entorno dinámico. De forma indirecta se pretende investigar nuevas técnicas de entrenamiento para las Redes Neuronales de Aprendizaje Profundo que permitan realizar los entrenamiento con menos datos de entrenamiento y en menor tiempo, ésto basado en el mejoramiento de los optimizadores de propagación del error como *Adaptive Momentum Estimation* (Kingma and Ba, 2014), *Nesterov Accelerated Adaptive Moment Estimation* (Dozat, 2016) o *Root Mean Square Propagation* (Tieleman and Hinton, 2012) y la mejoría de las capas convolucionales agregando algoritmos inteligentes de extracción de información, como la extracción de características clave basado en agrupamiento jerárquico (Wan et al., 2019) o bien fusión mediante alienación basado en el método de (Li, 2015) expresado como una capa entrenable, o medidas difusas (Grabisch et al., 2010) entrenables.

Referencias

Alexeev, V., Staravoitau, A., Piskun, G., and Likhacheuski, D. (2018). End to end learning for a driving simulator. *foreignlanguage russian Reports of the Belarusian State University of Informatics and Radioelectronics*, (2 (112)).

Álvarez, C. (2016). Rambo model. url: https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/rambo.

Aziz, M. V. G., Hindersah, H., and Prihatmanto, A. S. (2017). Implementation of vehicle detection algorithm for self-driving car on toll road cipularang using python language. In *Electric Vehicular Technology (ICEVT), 2017 4th International Conference on*, pages 149–153. IEEE.

Bechtel (2018). Deeppicar v2. url:https://github.com/mbechtel2/DeepPicar-v2.

Bechtel, M. G., McEllhiney, E., and Yun, H. (2017). Deeppicar: A low-cost deep neural network-based autonomous car. *arXiv preprint arXiv:1712.08644*.

Bednar, J. and Watt, T. (1984). Alpha-trimmed means and their relationship to median filters. *IEEE Transactions on acoustics, speech, and signal processing*, 32(1):145–153.

Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., and Jenssen, R. (2017). *Recurrent Neural Networks for Short-term Load Forecasting: An Overview and Comparative Analysis*. Springer.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., and Muller, U. (2017). Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911*.

Bonet Cruz, I., Salazar Martínez, S., Rodríguez Abed, A., Grau Ábalo, R., and García Lorenzo, M. M. (2007). Redes neuronales recurrentes para el análisis de secuencias. *Revista Cubana de Ciencias Informáticas*, 1(4).

Chandra, B. and Sharma, R. K. (2017). On improving recurrent neural network for image classification. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 1904–1907. IEEE.

Chen, C., Seff, A., Kornhauser, A., and Xiao, J. (2015). Deepdriving: Learning affordance for direct perception in autonomous driving. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 2722–2730. IEEE.

Chen, S., Zhang, S., Shang, J., Chen, B., and Zheng, N. (2017a). Brain-inspired cognitive model with attention for self-driving cars. *IEEE Transactions on Cognitive and Developmental Systems*.

Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., and Urtasun, R. (2016). Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156.

Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017b). Multi-view 3d object detection network for autonomous driving. In *IEEE CVPR*, volume 1, page 3.

Chollet, F. (2017). Deep learning with python, vol. 1. *Greenwich, CT: Manning Publications CO*.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

CPS-VO (2012). Cat vehicle testbed. https://cps-vo.org/group/CATVehicleTestbed.

de Oliveira, G. H., da Silva, F. A., Pereira, D. R., de Almeida, L. L., Artero, A. O., Bonora, A. F., and de Albuquerque, V. H. C. (2018). Automatic detection and recognition of text-based traffic signs from images. *IEEE Latin America Transactions*, 16(12):2947–2953.

De Silva, V., Roche, J., and Kondoz, A. (2018). Robust fusion of lidar and wide-angle camera data for autonomous mobile robots. *Sensors*, 18(8):2730.

Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*.

Dozat, T. (2016). Incorporating nesterov momentum into adam.

Du, X., Ang, M. H., and Rus, D. (2017). Car detection for autonomous vehicle: Lidar and vision fusion approach through deep learning framework. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 749–754. IEEE.

Elman, J. L. (1995). Language as a dynamical system. *Mind as motion: Explorations in the dynamics of cognition*, pages 195–225.

Emirler, M. T., Uygan, İ. M. C., Aksun Güvenç, B., and Güvenç, L. (2014). Robust pid steering control in parameter space for highly automated driving. *International Journal of Vehicular Technology*, 2014.

Foote, T. (2017). Simulated car demo. https://www.ros.org/news/2017/06/simulated-car-demo.html.

Foundation, O. S. R. (2017). Vehicle and city simulation. http://gazebosim.org/blog/car_sim.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Grabisch, M., Sugeno, M., and Murofushi, T. (2010). *Fuzzy measures and integrals: theory and applications*. Heidelberg: Physica, 2000.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2017). Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232.

Haapala, J. et al. (2017). Recurrent neural networks for object detection in video sequences.

Han, J.-Y., Perng, N.-H., and Chen, H.-J. (2012). Lidar point cloud registration by image detection technique. *IEEE Geoscience and Remote Sensing Letters*, 10(4):746–750.

Higgins, M. (2016). Chauffeur model. url: https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/chauffeur.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kline, R. B. (2013). *Beyond significance testing: Statistics reform in the behavioral sciences*. American Psychological Association.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

Li, C., Song, D., Tong, R., and Tang, M. (2019). Illumination-aware faster r-cnn for robust multispectral pedestrian detection. *Pattern Recognition*, 85:161–171.

Li, J. (2015). Fusion of lidar 3d points cloud with 2d digital camera image. *Oakland University: Rochester, MI, USA*.

Liang, M. and Hu, X. (2015). Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375.

Lin, G., Milan, A., Shen, C., and Reid, I. D. (2017). Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Cvpr*, volume 1, page 5.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Litman, T. (2017). *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute.

Maqueda, A. I., Loquercio, A., Gallego, G., García, N., and Scaramuzza, D. (2018). Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5419–5427.

Matuz, M. (2017). Detección de anormalidades en mamografías utilizando la red neuronal convolucional alexnet. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Morales, M. (2018). Segmentación no paramétrica de tejidos cerebrales mediante unaarquitectura paralela de redes neuronales convolucionales. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET.

Morley, P., Warren, A., Rabb, E., Whitsitt, S., Bunting, M., and Sprinkle, J. (2013). Generating a ros/jaus bridge for an autonomous ground vehicle. In *Proceedings of the 2013 ACM workshop on Domain-specific modeling*, pages 13–18. ACM.

Nageshrao, S., Tseng, E., and Filev, D. (2019). Autonomous highway driving using deep reinforcement learning. *arXiv preprint arXiv:1904.00035*.

Novitskiy, K. and Boyarskaya, A. (2016). Self-driving cars.

Olson, E. A., Risso, N., Johnson, A. M., and Sprinkle, J. (2017). Fuzzy control of an autonomous car using a smart phone. In *Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), 2017 CHILEAN Conference on*, pages 1–6. IEEE.

Pattanayak, S. (2016). *Pro Deep Learning with TensorFlow*. Springer.

Pei, S., Tang, F., Ji, Y., Fan, J., and Ning, Z. (2018). Localized traffic sign detection with multi-scale deconvolution networks. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 355–360. IEEE.

Pita Fernández, S. and Pértega Díaz, S. (2000). Significancia estadística y relevancia clínica. *Cad Aten Primaria*, 8:191–195.

Pérez, J. (2001). Compresión de imágenes usando redes neuronales artificiales recurrentes. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET.

Ramos, S., Gehrig, S., Pinggera, P., Franke, U., and Rother, C. (2017). Detecting unexpected obstacles for self-driving cars: Fusing deep learning and geometric modeling. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1025–1032. IEEE.

Rödel, C., Stadler, S., Meschtscherjakov, A., and Tscheligi, M. (2014). Towards autonomous cars: the effect of autonomy levels on acceptance and user experience. In *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, pages 1–8. ACM.

Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76.

Salman, Y. D., Ku-Mahamud, K. R., and Kamioka, E. (2017). Distance measurement for self-driving cars using stereo camera. In *proceeding 6th Int. Conf. Comput. informations*, number 105, pages 235–242.

Sardinha, H. (2017). Darpa robotics challenge simulator. https://github.com/Hurisa/drcsim.

Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks–ICANN 2010*, pages 92–101. Springer.

Shalev-Shwartz, S., Shammah, S., and Shashua, A. (2016). Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*.

Spiegel, M. R. (1991). *Estadística*. McGraw-Hill Interamericana,.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Suárez (2018). Sintonización de una red totalmente conectada para segmentación de dos clases de objetos en imágenes. Master's thesis, Centro Nacional de Investigación y Desarrollo Tecnológico CENIDET.

Tian, Y., Pei, K., Jana, S., and Ray, B. (2017). Deeptest: Automated testing of deep-neural-network-driven autonomous cars. *arXiv preprint arXiv:1708.08559*.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*.

Venna, S. R., Tavanaei, A., Gottumukkala, R. N., Raghavan, V. V., Maida, A. S., and Nichols, S. (2018). A novel data-driven model for real-time influenza forecasting. *IEEE Access*, 7:7691–7701.

Virgo, M. (2016). Udacity self driving car challenge 2 dataset. url: https://github.com/udacity/self-driving-car/tree/master/datasets/CH2.

Wan, J., Zheng, P., Si, H., Xiong, N. N., Zhang, W., and Vasilakos, A. V. (2019). An artificial intelligence driven multi-feature extraction scheme for big data detection. *IEEE Access*, 7:80122–80132.

Widaa, A. H. and Talha, W. A. (2017). Design of fuzzy-based autonomous car control system. In *Communication, Control, Computing and Electronics Engineering (ICCCCEE), 2017 International Conference on*, pages 1–7. IEEE.

Xu, H., Gao, Y., Yu, F., and Darrell, T. (2016). End-to-end learning of driving models from large-scale video datasets. *arXiv preprint*.

Zhang, S., Wu, Y., Che, T., Lin, Z., Memisevic, R., Salakhutdinov, R. R., and Bengio, Y. (2016). Architectural complexity measures of recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1822–1830.

Zhang, S., Yang, J., and Schiele, B. (2018). Occluded pedestrian detection through guided attention in cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6995–7003.

Zhu, H., Long, M., Wang, J., and Cao, Y. (2016). Deep hashing network for efficient similarity retrieval. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Anexo A

Constancias de participaciones

UNIVERSIDAD TECNOLÓGICA
EMILIANO ZAPATA DEL ESTADO DE MORELOS
ORGANISMO PÚBLICO DESCENTRALIZADO DEL GOBIERNO DEL ESTADO DE MORELOS

otorga el presente

RECONOCIMIENTO

A: Jesús Antonio Luna Álvarez

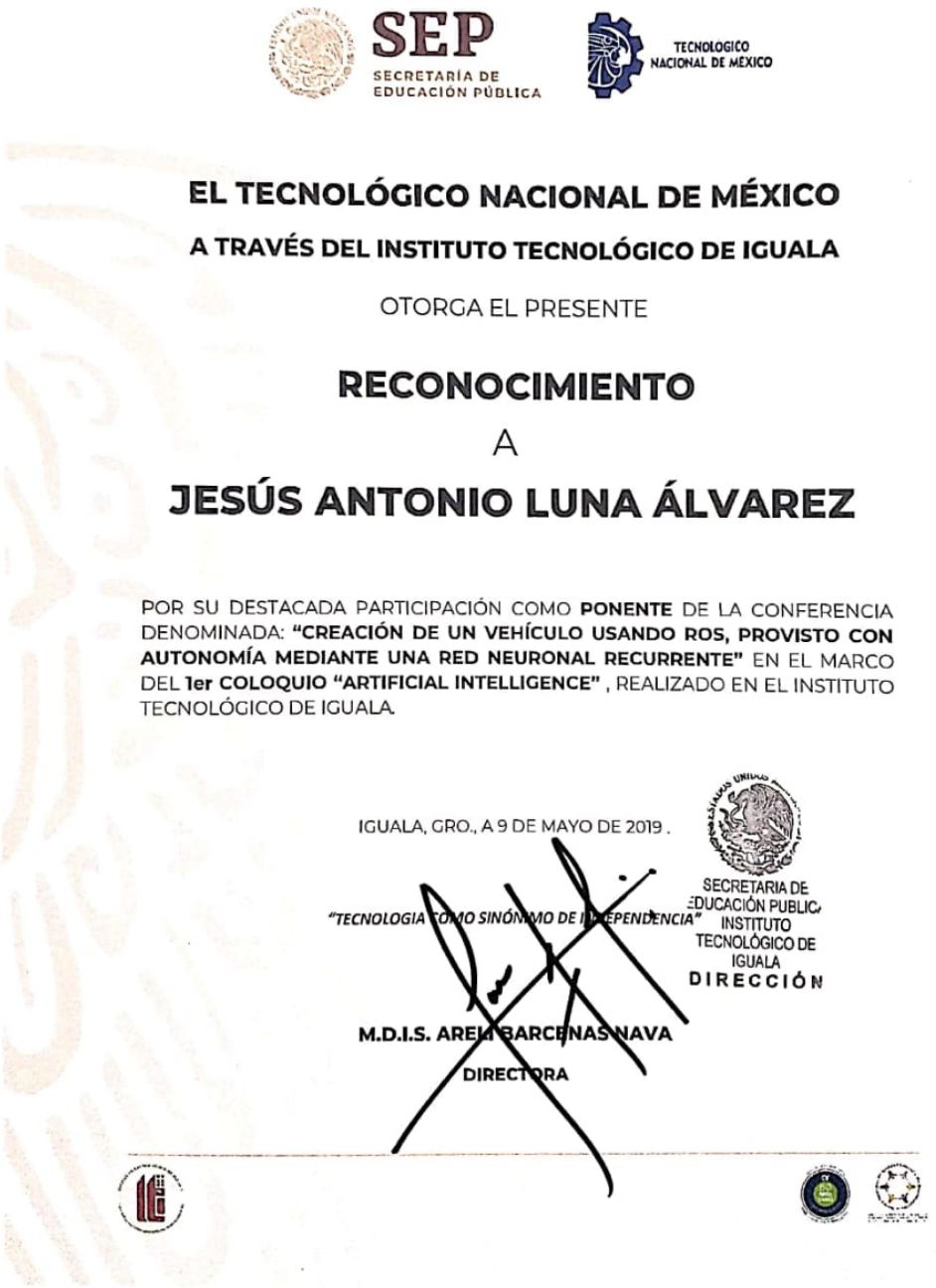
Por su participación como ponente de la conferencia
"Creación de un vehículo usando ROS, provisto con autonomía mediante una Red Neuronal Recurrente"
en el marco del evento: Escuela de Inteligencia Artificial y Robótica 2018,
llevado a cabo en las instalaciones de esta Universidad Tecnológica,
del 25 al 27 de octubre del presente año.

Emiliano Zapata, Mor, octubre de 2018

UTEZ
DIVISIÓN ACADÉMICA DE MECÁNICA INDUSTRIAL

M. en C. Jaime Vázquez Colín
Director de la División Académica de Mecánica Industrial

Figura A.1: Constancia como ponente en la Escuela de Inteligencia Artificial y Robótica 2018, UTEZ.



SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

EL TECNOLÓGICO NACIONAL DE MÉXICO

A TRAVÉS DEL INSTITUTO TECNOLÓGICO DE IGUALA

OTORGA EL PRESENTE

RECONOCIMIENTO

A

JESÚS ANTONIO LUNA ÁLVAREZ

POR SU DESTACADA PARTICIPACIÓN COMO **PONENTE** DE LA CONFERENCIA DENOMINADA: **"CREACIÓN DE UN VEHÍCULO USANDO ROS, PROVISTO CON AUTONOMÍA MEDIANTE UNA RED NEURONAL RECURRENTE"** EN EL MARCO DEL **1er COLOQUIO "ARTIFICIAL INTELLIGENCE"** , REALIZADO EN EL INSTITUTO TECNOLÓGICO DE IGUALA.

IGUALA, GRO., A 9 DE MAYO DE 2019 .

"TECNOLOGÍA COMO SINÓNIMO DE INDEPENDENCIA"

SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO TECNOLÓGICO DE IGUALA
DIRECCIÓN

M.D.I.S. ARELI BARCENAS NAVA

DIRECTORA

Figura A.2: Constancia como ponente en el 1er coloquio .Artificial Intelligence", Tecnológico de Iguala.

Figura A.3: Constancia como concursante en concurso de posters, ICMEAE 2018.

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

EL TECNOLÓGICO NACIONAL DE MÉXICO
A TRAVÉS DEL CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO

OTORGAN LA PRESENTE

CONSTANCIA

A

Ángel Arturo Rendón Castro, Jesús Antonio Luna Álvarez, Dante Mújica Vargas

POR SU PARTICIPACIÓN CON EL ARTÍCULO: **"FUZZY CONTROL OF STEWART PLATFORM WITH ARTIFICIAL VISION",** PRESENTADO EN LA SEGUNDA JORNADA DE CIENCIA Y TECNOLOGÍA APLICADA, QUE SE LLEVO A CABO LOS DÍAS 4 Y 5 DE ABRIL DE 2019, EN LA CIUDAD DE CUERNAVACA, MORELOS, MÉXICO.

CUERNAVACA, MOR., A 5 DE ABRIL DE 2019.

S. E. P.
CENTRO NACIONAL DE INVESTIGACION Y DESARROLLO TECNOLÓGICO

DR. VÍCTOR HUGO OLIVARES PEREGRINO
DIRECTOR

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

PREMIO ESTATAL AHORRO DE ENERGÍA 2015

Figura A.4: Constancia como coautor de artículo publicado en la Segunda Jornada de Ciencia y Tecnología Aplicada, CENIDET 2018.

Supresión de lluvia en imágenes mediante una Red Neuronal Convolucional

Antonio Luna Álvarez, Dante Mújica Vargas, Manuel Mejía Lavalle, Gerardo Reyes Salgado
Dept. Ciencias Computacionales
Tecnológico Nacional de México / CENIDET
Cuernavaca Morelos, México
{jesus.luna18ce, dantemv, mlavalle, greyes}@cenidet.edu.mx

***Abstract*—En este articulo se presenta una estrategia de supresión de lluvia presente en imágenes utilizadas para la autoconducción mediante un modelo de Red Neuronal Convolucional con una capa de Filtro Guiado entrenable. Se demostró la efectividad de la red para el mejoramiento de las imágenes frente a los métodos más utilizados en la literatura.**

***Index Terms*—Redes Neuronales Convolucionales, Supresión de lluvia, Autoducción de vehículos.**

I. Introducción

Dentro del dominio de los automóviles autoconducidos las redes neuronales de aprendizaje profundo es la técnica más utilizada y que se ha mostrado como la más efectiva [1], [2]. Existen una variedad de modelos prediseñados y preentrenados para realizar dicha tarea con un margen de error pequeño, estos modelos en su mayoría trabajan con la fusión de diversos sensores infrarrojos, GPS y sobre todo con el uso de diversas

En este artículo de propone una solución basada en el Filtro Guiado [15], [16] expresado como una capa de filtrado en una Red Neuronal Convolucional. Se ha demostrado que este filtro tiene la capacidad de realizar el mejoramiento de la imagen en vídeos [17], lo cual indica que es apto para operar en tiempo real.

El resto del documento está organizado de la siguiente manera: La Sección II describe los conceptos teóricos necesarios para entender el funcionamiento de los filtros utilizados para la supresión de lluvia. La descripción del sistema propuesto se detalla en la Sección III. Los datos utilizados para entrenamientos y pruebas, configuraciones del sistema propuesto y la forma de evaluación es descrita en la Sección IV. Finalmente los resultados y comparación del sistema es mostrado en la Sección V para finalizar con las conclusiones en el final del documento.

Figura A.5: Constancia como autor en la Escuela de Inteligencia Artificial y Robótica 2019, UTEZ.

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

EL TECNOLÓGICO NACIONAL DE MÉXICO
A TRAVÉS DEL CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO

OTORGA EL PRESENTE

RECONOCIMIENTO

a

Antonio Luna Álvarez and Dante Mújica Vargas

POR LA PRESENTACIÓN DEL ARTÍCULO TITULADO: **"RED NEURONAL CONVOLUCIONAL Y DE FILTRADO PARA LA SUPRESIÓN DE LLUVIA Y NIEBLA EN IMÁGENES DE CONDUCCIÓN"**, DURANTE LA **3ra JORNADA DE CIENCIA Y TECNOLOGÍA APLICADA.**

CUERNAVACA, MORELOS, MÉXICO, NOVIEMBRE 14 Y 15, 2019.

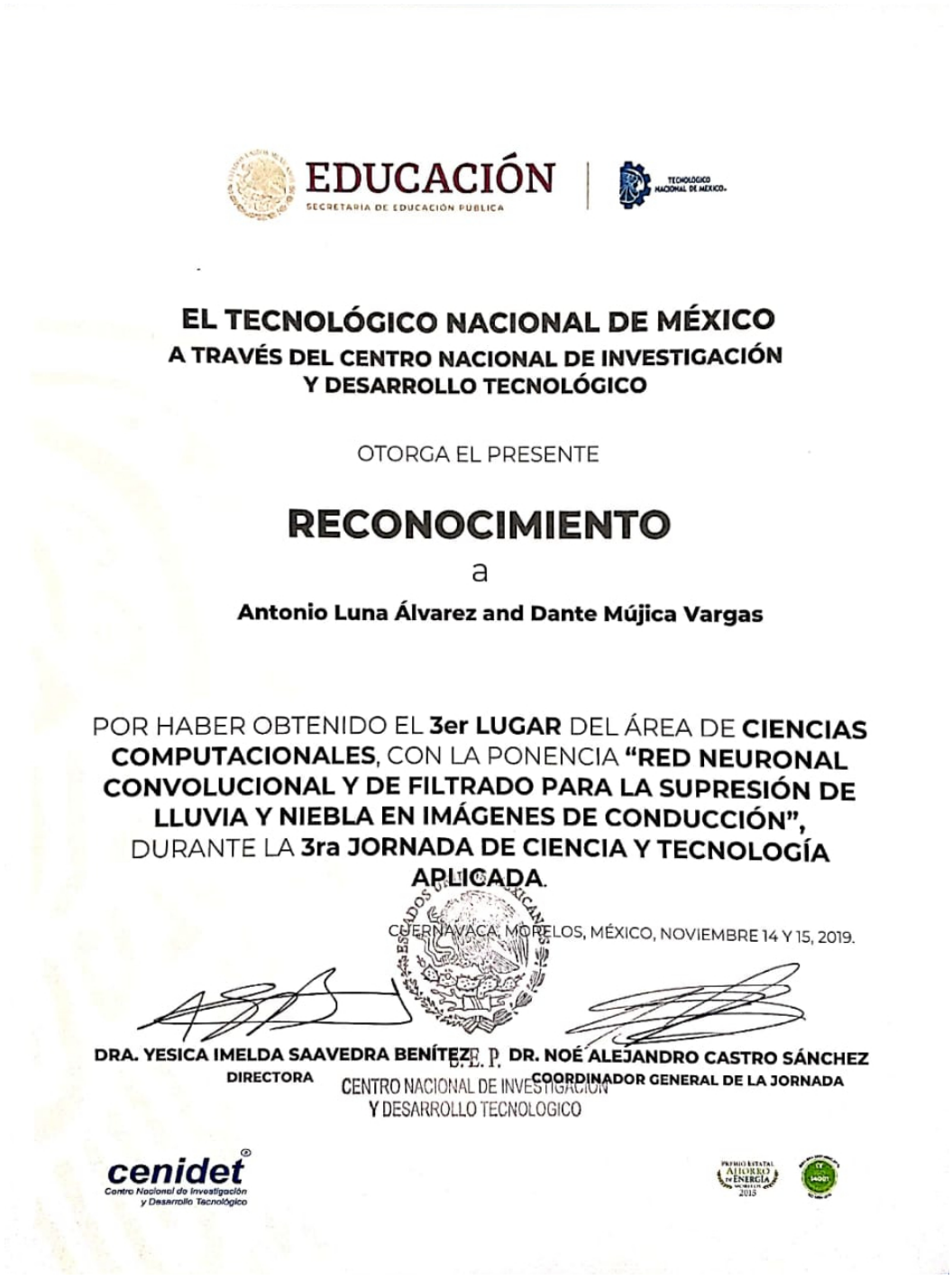
DRA. YESICA IMELDA SAAVEDRA BENÍTEZ
DIRECTORA

DR. NOÉ ALEJANDRO CASTRO SÁNCHEZ
COORDINADOR GENERAL DE LA JORNADA

S. E. P.
CENTRO NACIONAL DE INVESTIGACION Y DESARROLLO TECNOLOGICO

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

Figura A.6: Constancia como autor la Tercera Jornada de Ciencia y Tecnología Aplicada, CENIDET 2019.



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

EL TECNOLÓGICO NACIONAL DE MÉXICO
A TRAVÉS DEL CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO

OTORGA EL PRESENTE

RECONOCIMIENTO

a

Antonio Luna Álvarez and Dante Mújica Vargas

POR HABER OBTENIDO EL **3er LUGAR** DEL ÁREA DE **CIENCIAS COMPUTACIONALES**, CON LA PONENCIA **"RED NEURONAL CONVOLUCIONAL Y DE FILTRADO PARA LA SUPRESIÓN DE LLUVIA Y NIEBLA EN IMÁGENES DE CONDUCCIÓN"**, DURANTE LA **3ra JORNADA DE CIENCIA Y TECNOLOGÍA APLICADA**.

CUERNAVACA, MORELOS, MÉXICO, NOVIEMBRE 14 Y 15, 2019.

DRA. YESICA IMELDA SAAVEDRA BENÍTEZ
DIRECTORA

DR. NOÉ ALEJANDRO CASTRO SÁNCHEZ
COORDINADOR GENERAL DE LA JORNADA

C.E.N.I.D.E.T.
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLOGICO

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

Figura A.7: Constancia de 3^{er} lugar del artículo presentado en la Tercera Jornada de Ciencia y Tecnología Aplicada, CENIDET 2019.

Figura A.8: Constancia como coautor de artículo publicado en el congreso MCPR 2019.

Figura A.9: Constancia como autor de artículo publicado en el congreso CORE 2019, CIC-IPN.

Noise Gradient Strategy for an Enhanced Hybrid Convolutional-Recurrent Deep Network to control a Self-Driving Vehicle

Dante Mújica-Vargas[a,*], Antonio Luna-Alvarez[a], José de Jesús Rubio[b], Blanca Carvajal-Gámez[c]

[a] *Tecnológico Nacional de México/CENIDET, Cuernavaca, Morelos, México.*
[b] *Instituto Politécnico Nacional, SEPI-ESIME-Azcapotzalco,Ciudad de México, México.*
[c] *Instituto Politécnico Nacional, SEPI-ESCOM, Ciudad de México 07320, México.*

Abstract

In this paper a noise gradient strategy on the Adam optimizer is introduced, in order to reduce the training time of our enhanced Chauffeur hybrid deep model. This neural network was modified to take into account the time dependence of the input visual information, with the aim of increasing the autonomy of a self-driving vehicle. The effectiveness of this proposal was evaluated and quantified during the training and validation with a performance superior than comparative optimizers. In terms of the autonomy, it can be seen that our improved Hybrid Convolutional-Recurrent Deep Network was better trained, achieving autonomy greater than 95% with a minimum number of human interventions.

Keywords: Hybrid Convolutional-Recurrent Deep Network, Noise Gradient Strategy, Self-driving cars

Track your recent Co-Authored submission to ASOC Recibidos ×

Applied Soft Computing <eesserver@eesmail.elsevier.com>
para mí

inglés › español Traducir mensaje

*** Automated email sent by the system ***

Dear Dr. Antonio Luna-Alvarez,

You have been listed as a Co-Author of the following submission:

Journal: Applied Soft Computing
Title: Noise Gradient Strategy for an Enhanced Hybrid Convolutional-Recurrent Deep Network to control a Self-Driving Vehicle
Corresponding Author: Dante Mújica-Vargas
Co-Authors: Antonio Luna-Alvarez; José de Jesús Rubio; Blanca Carvajal-Gámez

To be kept informed of the status of your submission, register or log in (if you already have an Elsevier profile).

Register here: https://ees.elsevier.com/asoc/default.asp?acw=&pg=preRegistration.asp&user=coauthor&fname=Antonio&lname=Luna-Alvarez&email=jesus.luna18ce@cenidet.edu.mx

Or log in: https://ees.elsevier.com/asoc/default.asp?acw=&pg=login.asp&email=jesus.luna18ce@cenidet.edu.mx

If you did not co-author this submission, please do not follow the above link but instead contact the Corresponding Author of this submission at dantemv@cenidet.edu.mx.

Thank you,

Applied Soft Computing

Figura A.10: Artículo sometido a la revista *Applied Soft Computing, Elsevier*.