Camera-Centric Autonomous Driving: Progress in End-to-End Learning for Simulation and Real-World Deployment

Zhihui Lai

BSc (Hons) W. Aust., BEng Guangdong

This thesis is presented for the degree of Doctor of Philosophy of The University of Western Australia

Department of Electrical, Electronic and Computer Engineering

Electrical Engineering

2025

Acknowledgements

I am profoundly grateful to my supervisor, Professor Thomas Bräunl, whose mentorship shaped my academic journey. It was in his undergraduate robotics course that I first developed a fascination with robotics and mobile platforms, which gradually inspired me to pursue an honors degree and later a direct-entry PhD. I was extraordinarily fortunate to join his research group during my honors program, where I inherited the ModCar project—the very initiative that defined my doctoral research direction—and began exploring end-to-end autonomous driving. While the research journey was not without challenges, it yielded modest achievements and, most importantly, ignited my passion to delve deeper into this field. The subsequent award of the RTP Scholarship enabled me to continue this pursuit. Throughout my studies, Thomas provided the perfect balance of intellectual guidance and autonomy, empowering me to boldly test ideas and ultimately finish my PhD journey.

I would like to thank the Department of Transport (DoT) in Western Australia and Stockland Corporation Limited for the support of the Eglinton experiment.

My sincere appreciation extends to our REV sponsors, including Allkem, Tesla Slack, CD Dodd, Dyflex, Tesla Roadster Inc, Tesla Owners Club of WA, Tesla Owners Club of Australia, SBG Systems, and Altronics, as well as our university sponsors UWA School of Engineering, Business School, School of Physics, Mathematics and Computing, and the UWA Innovation Quarter. Without their generous support, this research would not have been possible.

I am equally indebted to my exceptional colleagues—Kieran Quirke-Brown, Xiangrui Kong, An Trung (Lee) Le, and Patrick Reidel—whose camaraderie made this endeavor rewarding. Special thanks to Kieran, our student leader, whose encyclopedic practical knowledge made him our first recourse when facing obstacles. His expertise accelerated our team's adaptation to shuttle bus development and troubleshooting.

To the anonymous reviewers of my work: your meticulous feedback elevated the quality of this dissertation, and for that I am deeply thankful.

To my parents: your unwavering support and sacrifices over the past two decades lifted me to where I stand today. This achievement belongs as much to you as it does to me.

To my friends: thank you for the shared meals, travels, and companionship that sustained me through this journey.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship.

This research was supported by a UWA fees scholarship.

Abstract

End-to-end (E2E) learning has emerged as an alternative paradigm to traditional modular autonomous driving pipelines by directly mapping sensory inputs to control commands. Despite recent progress, many E2E systems depend on large-scale datasets, extensive multi-modal sensor suites, and high-performance computing platforms, which limits their applicability to low-cost autonomous vehicles and pilot deployments. This thesis investigates how lightweight, camera-based E2E driving systems can be designed, trained, and deployed under constraints of limited data, limited computational resources, and embedded hardware.

The research is structured around four core questions: how compact monocular E2E architectures can be realized under resource constraints; how temporal memory and maneuver specialization can improve robustness in complex, multi-step navigation tasks; how simulation environments such as CARLA and AWSIM can be benchmarked and integrated to support scalable development and sim-to-real workflows; and to what extent camera-only E2E systems can achieve reliable closed-loop performance within constrained operational design domains (ODDs).

To address these questions, three complementary studies are conducted, forming a progressive research pathway from controlled experimentation to real-world deployment. The first study evaluates memory-augmented E2E models on a scaled indoor platform, demonstrating the importance of temporal reasoning for sequential maneuvers. The second study systematically benchmarks CARLA and AWSIM using common digital twins, clarifying their respective strengths for E2E training, ROS2 integration, and scenario-based testing. The third study presents a Mixture-of-Experts (MoE) E2E framework deployed on a full-scale autonomous shuttle bus, enabling context-specific control for maneuvers such as lane following, pull-ins, pull-outs, and roundabout navigation.

Closed-loop suburban driving trials show that lightweight, camera-only MoE architectures can achieve smooth and operationally consistent behavior within the evaluated routes and ODDs. These results demonstrate the practical feasibility of camera-centric E2E autonomy under realistic resource constraints. However, the findings are intentionally scoped and do not constitute general safety guarantees. Extending such systems beyond the tested ODDs would require comprehensive edge-case coverage, redundancy strategies, and formal safety assurance processes that are beyond the scope of this thesis.

Overall, this work contributes architectural insights, experimental methodologies, and empirical evidence for cost-effective E2E autonomous driving. By integrating simulation, scaled platforms, and real-world shuttle trials, the thesis clarifies both the potential and the limitations of lightweight, camera-based E2E systems, and situates their role within the broader evolution of autonomous driving research.

List of Acronyms

ACC Adaptive Cruise Control

AD Autonomous Driving

ADAS Advanced Driver Assistance Systems

ADS Automated Driving System

AHS Automated Highway Systems

AI Artificial Intelligence

ALVINN Autonomous Land Vehicle In a Neural Network

AM Autonomous Mode

API Application Programming Interface

AV Autonomous Vehicle

AWSIM Autoware Simulator

BEV Bird's-Eye View

BOM Bill of Materials

BRT Bus Rapid Transit

C-V2X Cellular Vehicle-to-Everything

CACC Cooperative Adaptive Cruise Control

CAN Controller Area Network

CARLA Car Learning to Act Simulator

CIL Conditional Imitation Learning

CNN Convolutional Neural Network

COVID Coronavirus Disease

CPU Central Processing Unit

ConvLSTM Convolutional Long Short-Term Memory

DPIA Data Protection Impact Assessment

DoT Department of Transport

DVS Dynamic Vision Sensor

E2E End-to-End

ESTOP Emergency Stop

EU European Union

FCN Fully Convolutional Network

FLIR Forward Looking Infrared (Point Grey cameras brand)

FMCW Frequency-Modulated Continuous Wave

FoV Field of View

FPS Frames Per Second

FSD Full Self-Driving

GDPR General Data Protection Regulation

GNSS Global Navigation Satellite System

GPS Global Positioning System

GPU Graphics Processing Unit

GUI Graphical User Interface

HD High Definition

HDR High Dynamic Range

HDRP High Definition Render Pipeline

HIL Hardware-in-the-Loop

HMI Human–Machine Interaction

IEC International Electrotechnical Commission

IMU Inertial Measurement Unit

INS Inertial Navigation System

IoT Internet of Things

ISP Image Signal Processor

KPIs Key Performance Indicators

LCA Life-Cycle Assessment

LiDAR Light Detection and Ranging

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MIC Manual Intervention Count

MIMO Multiple-Input Multiple-Output

MIL Model-in-the-Loop

MIIT Ministry of Industry and Information Technology (China)

ML Machine Learning

MM Manual Mode

MoE Mixture of Experts

MSE Mean Squared Error

NAV2 Navigation2 (ROS 2 navigation stack)

NHTSA National Highway Traffic Safety Administration

NN Neural Network

NOA Navigate on Autopilot

NPC Non-Player Character

NTRIP Networked Transport of RTCM via Internet Protocol

OEM Original Equipment Manufacturer

ODD Operational Design Domain

OPA Optical Phased Array

OpenCV Open Source Computer Vision Library

OS Operating System

OTA Over-The-Air (software update)

PLC Programmable Logic Controller

PS4 PlayStation 4

PTP Precision Time Protocol

REV Renewable Energy Vehicle

RGB Red-Green-Blue

RL Reinforcement Learning

ROS Robot Operating System

ROS2 Robot Operating System 2

RViz ROS Visualization

RSU Roadside Unit

RTF Real-Time Factor

RTK Real-Time Kinematic

SAE Society of Automotive Engineers

SCANeR Simulation of Connected and Autonomous Navigation on Roads

SDK Software Development Kit

SIL Software-in-the-Loop

SLAM Simultaneous Localization and Mapping

SOTIF Safety of the Intended Functionality

SUMO Simulation of Urban Mobility

TFLite TensorFlow Lite

TMP Traffic Management Plan

TRT TensorRT

UE4 Unreal Engine 4

UN R157 United Nations Regulation No. 157 (Automated Lane Keeping Systems)

UNECE United Nations Economic Commission for Europe

UWA University of Western Australia

V2I Vehicle-to-Infrastructure

V2V Vehicle-to-Vehicle

V2X Vehicle-to-Everything

VKT Vehicle-Kilometers Traveled

VLA Vision-Language-Action

VRU Vulnerable Road User

WP.29 World Forum for Harmonization of Vehicle Regulations

Contents

Chapter 1

Introduction

I. Background and Overview

Autonomous driving (AD) has emerged as one of the most transformative technological developments in modern transportation, integrating advances in artificial intelligence (AI), robotics, sensing, communication networks, and vehicle engineering [1–5]. Over the past two decades, both industry and academia have advanced from basic driver-assistance systems toward higher levels of automation, driven by breakthroughs in perception, planning, control, and embedded computing.

While traditional autonomous-driving systems rely on modular pipelines—separating perception, mapping, prediction, planning, and control—end-to-end (E2E) learning offers an alternative paradigm. Instead of constructing and maintaining multiple engineered modules, an E2E neural network learns a direct mapping from sensor inputs to driving actions.

Traditional systems often suffer from error accumulation across individual modules and struggle with long-tail scenarios, typically requiring laborious rule-based additions that are both time-consuming and difficult to scale. In contrast, end-to-end approaches enable richer information flow to downstream decisions and demonstrate stronger generalization in rare or unseen situations. However, E2E autonomy also faces notable challenges, including limited interpretability, high data requirements, and difficulties in verification and safety assurance.

Recent industrial and academic developments show growing interest in such E2E systems, but many existing approaches assume large-scale datasets, extensive sensor suites, and high-performance computing infrastructure that are often impractical in academic or low-cost deployment settings. This thesis investigates how *lightweight, camera-based* E2E architectures can be designed and deployed effectively under realistic constraints.

II. Historical Evolution of Autonomous Driving

The historical trajectory of autonomous driving can be divided into several distinct phases, each marked by characteristic technological capabilities, deployment models, and dominant research or commercial priorities.

1) Early Research and DARPA Challenges (1920s–2000s): The concept of autonomous driving has evolved over nearly a century, transitioning from early theoretical visions to the advanced AI-driven systems of today. The earliest notions of self-driving vehicles emerged as far back as the 1920s, when radio-controlled cars (Fig. 1) were publicly demonstrated in the United States to showcase wireless communication technologies [3]. However, these demonstrations were purely mechanical and lacked any form of environmental perception or decision-making capabilities.

The 1980s marked the emergence of computer vision in vehicle control, exemplified by Carnegie Mellon University's *Navlab* (Fig. 2) and the Bundeswehr University Munich's *VaMoRs* projects, which integrated onboard sensors, real-time computing, and early AI algorithms for road following [7, 8]. These systems relied on handcrafted perception pipelines and were limited by the computing power of the time, performing best in structured environments.

In the 1990s, programs such as the U.S. Automated Highway Systems (AHS) demonstrated cooperative vehicle–infrastructure systems capable of platooning [10, 11]. These initiatives emphasized reliable perception, longitudinal and lateral control, and vehicle-to-vehicle (V2V) communication, laying important groundwork for later advances.

As shown in Fig. 3, a major breakthrough came in the 2000s with the DARPA Grand Challenges (2004, 2005) and Urban Challenge (2007), which drove rapid innovation in unstructured and urban navigation [12–14]. Winning teams combined GPS, LiDAR, radar, cameras, and probabilistic planning into modular perception–planning–control stacks, successfully completing long-range autonomous missions without human intervention. These competitions demonstrated the viability of high-level autonomy and catalyzed industry adoption.

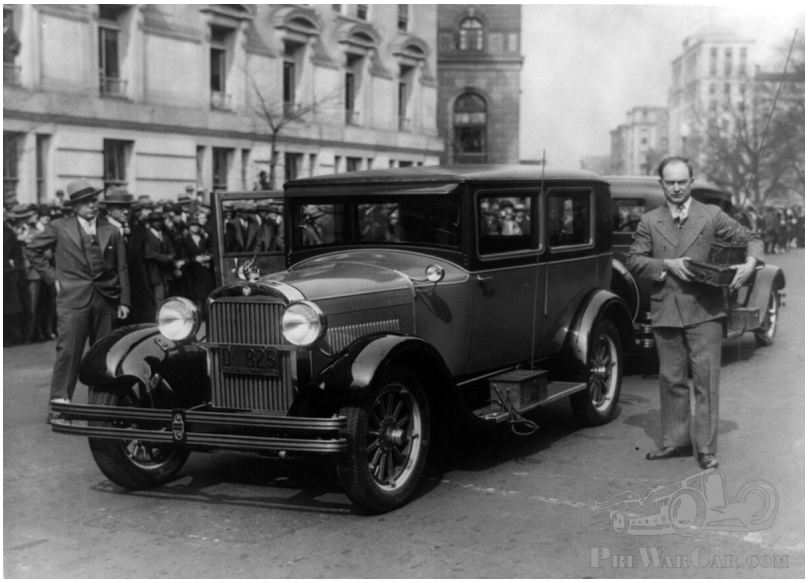


Fig. 1: American Wonder in 1920s, copied from [6]

Fig. 2: NavLab 2 vehicle using ALVINN in 1980s, copied from [9]

2) L1/L2 ADAS Commercialization (2010–2016): As depicted in Fig. 4, in the 2010s, advances in deep learning revolutionized perception, enabling end-to-end processing of high-dimensional sensory data [16, 17]. Combined with high-definition (HD) maps, simultaneous localization and mapping (SLAM), and increasingly sophisticated sensor fusion techniques [18, 19], autonomous vehicles achieved greater safety and robustness in complex traffic scenarios. Between 2010 and 2016, carmakers began introducing Level 1 and Level 2 Advanced Driver Assistance Systems (ADAS) into mass-market vehicles [2, 20], incorporating features such as adaptive cruise control, lane keeping assist, automated emergency braking, and automated parking [21]. These systems, which fused inputs from cameras, radar, and ultrasonic sensors, represented a gradual shift toward higher levels of automation as defined by SAE J3016 [22], while still requiring the driver to remain engaged.

3) Rise of L2+ and City NOA (2017–2021): The period from 2017 onward saw the expansion of Level 2+ systems, particularly in China and the US. "Navigate on Autopilot" (NOA) functions enabled semi-automated driving in both highway and urban settings. Chinese OEMs such as Xpeng (Fig. 5), Li Auto,

Fig. 3: Vehicle in the DARPA Grand Challenges in 2000s, copied from [15]

Fig. 4: NVIDIA-branded self-driving car at the NVIDIA GPU Technology Conference (GTC) 2018 in Silicon Valley, copied from [23]

and NIO deployed city NOA in multiple urban regions, supported by large-scale road testing and high-definition maps [2]. Tesla iteratively upgraded its Autopilot toward Full Self-Driving (FSD) capability, initially with a modular architecture but gradually shifting toward more unified, neural-network-based pipelines.

4) L3/L4 Pilots and Vehicle–Road–Cloud Integration (2021–Present): From 2021, multiple stakeholders initiated Level 3 and Level 4 pilots in geo-fenced areas. Honda and Mercedes-Benz obtained type approval for L3 traffic jam pilot systems under UN R157 in Japan and Germany. In China, Baidu Apollo, Huawei ADS, and other platforms launched commercial robotaxi services, while vehicle–road–cloud integration emerged as a key strategy for extending L4 operation to public transport, logistics, ports, and mining scenarios [25–27]. Cooperative perception using roadside LiDAR and cameras allowed reduced on-board sensor costs while maintaining safety.

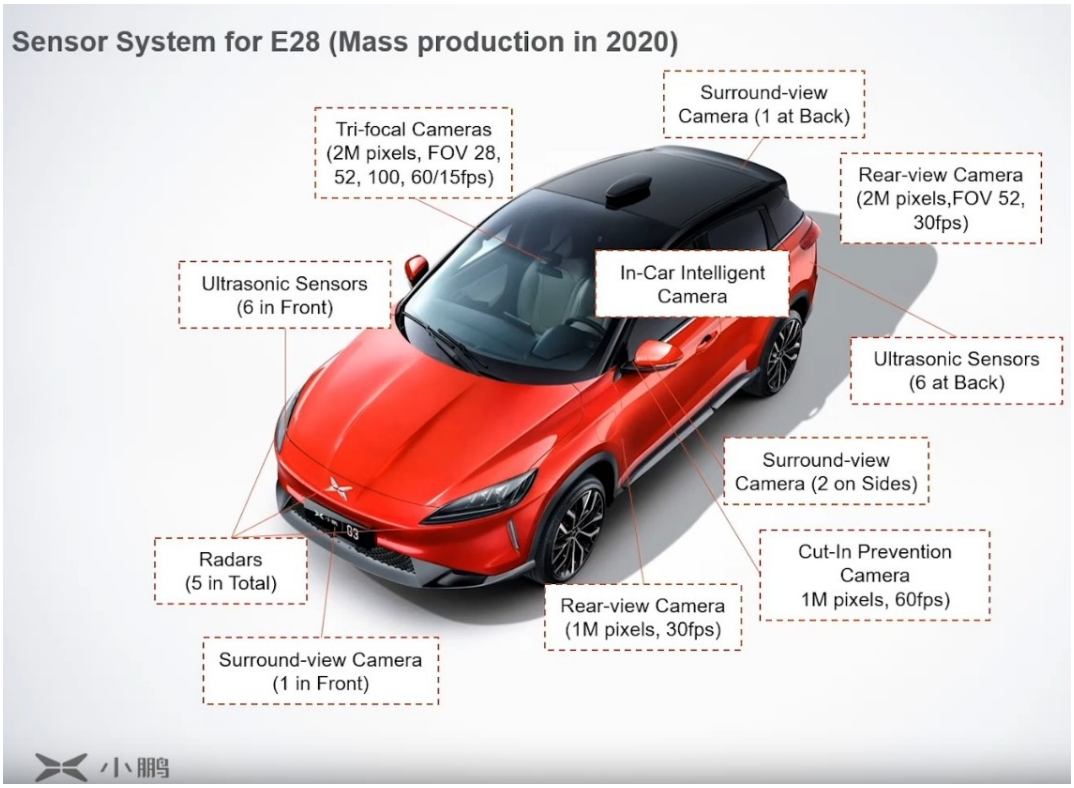


Fig. 5: Xpeng P7 sensor configuration in 2020, copied from [24]

III. Historical Evolution of Autonomous Driving Simulation

Autonomous driving (AD) simulation evolved from robotics testbeds and traffic micro-simulation to photorealistic, physics-aware digital twins tightly integrated with open-source AD stacks and large-scale data engines. Across this trajectory, three persistent drivers shaped the field: (i) *closed-loop fidelity* (vehicle dynamics, sensing, and multi-agent traffic), (ii) *scalability* (scenario generation, coverage, and automation for testing), and (iii) *integration* (ROS/ROS 2, Autoware, and learning pipelines).

A. 2000s Foundations: Robotics Platforms, Traffic Micro-simulation, and Vehicle Dynamics

Early AD simulation practice drew heavily from robotics platforms that enabled sensor–actuator loops and repeatable experiments. Player/Stage [28] established a client–server architecture and multi-robot abstractions, soon followed by Gazebo [29] which added 3D physics and extensible sensor plugins, creating a path to reproducible closed-loop experiments with ROS integration. In parallel, traffic micro-simulation matured, with SUMO [30] (Simulation of Urban Mobility) offering open, scalable multi-agent traffic models and network import pipelines. At the vehicle level, classical longitudinal/lateral dynamics models shaped controller-in-the-loop testing, connecting simulation results to real-world driving behavior [31]. Together, these ingredients—robotics simulators, traffic microsimulation, and vehicle dynamics—formed the basis for modular co-simulation and Hardware-/Software-/Model-in-the-Loop (HIL/SIL/MIL) workflows in the 2000s and early 2010s.

B. 2010s: AV-Specific, Photorealistic Simulators and Scenario-Based Testing

The 2010s marked a shift from robotics-first tools to AV-specific, photorealistic engines built on modern game technology. CARLA introduced an open, research-oriented simulator built on Unreal Engine with native support for camera/lidar/radar sensors, realistic urban layouts, and reproducible benchmarks; it

quickly became a de-facto standard for perception, planning, and reinforcement learning research [32]. AirSim expanded photorealism and physics-accurate sensing for both aerial and ground vehicles, facilitating fast experimentation with vision pipelines and control policies [33]. At the same time, learning-centric simulation matured: large-scale training in sim gained traction with *domain randomization* to bridge the sim-to-real gap (e.g., randomized textures, lighting, and object distributions), and deep RL surveys consolidated best practices for training policies and perception modules in virtual worlds [34, 35].

C. 2020–2022: Stack Integration, ROS/Autoware Co-Simulation, and Safety Workflows

The early 2020s emphasized tighter coupling between simulators and open-source autonomy stacks. LGSVL (later SVL) provided a Unity-based environment with high-fidelity sensors and city-scale maps, designed to run closed-loop with ROS/ROS 2 and Autoware, streamlining perception–planning–control evaluation and regression [36]. Surveys synthesized the broader landscape of intelligent testing and validation, while safety engineering reviews clarified scenario definition, oracle design, and measurable coverage for autonomy pipelines [37, 38]. These advances normalized workflows that combine physics-aware ego vehicle models, realistic multi-agent traffic (often via co-simulation with tools like SUMO), and reproducible automated test harnesses. The learning community continued to formalize RL in driving simulators while the classical stack benefited from increasingly standardized interfaces and artifacts (maps, scenarios, and sensor rigs) [35].

D. 2023–Present: Digital Twins, Autoware-Centric Pipelines, World Models, and Safety-at-Scale

Recent years accelerated *digital-twin* approaches and open-source pipelines aligned with production-grade stacks. AWSIM emerged as a Unity-based simulator designed for Autoware, emphasizing realistic lidar/camera pipelines, traffic co-simulation, and ROS 2-first interfaces; CARLA's ecosystem expanded with richer assets and improved sensor/physics fidelity for urban driving and benchmarking [32, 39]. Industry/consortia reports and whitepapers highlight safety-at-scale practices (scenario catalogs, structured coverage, and large-scale regression), the role of simulation in safety cases, and the increasing use of cloud resources to run millions of test kilometers virtually [40, 41]. Concurrently, learning-centric simulation embraced *data engines* and *world models*, unifying perception and prediction in generative simulators that can synthesize scenes, perturb agents, and assess policy robustness. vision-language-action (VLA) interfaces and world-model-based training further blur boundaries between simulated and real data, while industrial position papers emphasize high-fidelity sensor modeling and edge-case mining as critical to reducing deployment risk [42–44]. Regional ecosystem reports also document how municipal pilots, infrastructure support, and regulatory sandboxes feed back into scenario libraries for simulation, facilitating targeted stress tests and policy-aligned metrics [25, 26, 45].

E. Summary and Ongoing Challenges

Across eras, three themes persist. First, *fidelity vs. scale*: photorealistic sensing and accurate physics improve validity but are computationally costly; scalable coverage demands fast, scriptable scenario generation and efficient co-simulation. Second, *integration*: ROS/ROS 2, Autoware, and standardized artifacts enable reproducible, end-to-end regression from perception to control. Third, *learning in the loop*: domain randomization, synthetic data, RL, and emerging world models position simulation not only as a testbed but as a data *source* powering iterative improvement. The field continues to push on sensor realism (especially lidar/radar effects), standardized scenario semantics, measurable safety coverage, and robust sim-to-real transfer [25, 26, 28–33, 35, 38–44].

IV. Evolution and Focus on End-to-End Autonomous Driving

End-to-end (E2E) autonomous driving has evolved from early behavioral cloning to modern learning-centric stacks that output waypoints or controls directly from raw perception. This section presents a

timeline that emphasizes (i) representation choices (direct controls vs. waypoints/affordances), (ii) learning paradigms (pure imitation, RL fine-tuning, data engines), and (iii) evaluation and safety (open- vs. closed-loop, robustness, and industrialization).

A. 1980s–2005: Early Behavioral Cloning and Neural Control

Foundational work demonstrated that neural networks trained by behavioral cloning could map perception to steering on real roads. ALVINN learned a direct perception-to-control policy from camera input, establishing the core E2E idea—supervised learning from human demonstrations—decades before deep learning resurged [7].

B. 2006–2015: Affordance-Centric E2E and Modular Baselines

Before modern deep E2E, two currents ran in parallel. First, *affordance-based* E2E predicted mid-level semantics (e.g., heading angle, distance to lane, traffic light state) that a lightweight controller could track, trading full direct control for interpretable intermediate cues [17]. Second, industrial systems advanced largely modular pipelines (perception–prediction–planning) with HD maps and rigorous systems engineering; these provided robust baselines and clarified the integration challenges that E2E would later revisit [18].

C. 2016–2018: Deep Learning Revival (PilotNet, CIL, RL Fine-Tuning)

The deep-learning resurgence reignited E2E with large convolutional encoders and GPU training. PilotNet showed that a camera-only CNN trained by behavioral cloning could map pixels to steering, catalyzing a wave of E2E studies [16]. Conditional Imitation Learning (CIL) injected high-level commands (e.g., "turn left/right/straight") to disambiguate multi-modal policies; branched/gated decoders improved performance at decision points and clarified how intent should condition the policy [46]. Work in this period also explored sequence modeling and attention for temporal context [47, 48], reinforcement learning or DAgger-style fine-tuning to mitigate covariate shift [49], and the use of CARLA to standardize closed-loop evaluation and scenario diversity for learning policies [32].

D. 2019–2021: Waypoints vs. Direct Control, Closed-Loop & Safety

A central design fork emerged between *direct control* (regress steering/throttle) and *waypoint-based* E2E (predict future path points or trajectories). Direct control remains simple and data-efficient, but is sensitive to covariate shift; waypoint E2E decouples perception and low-level control, easing stability and allowing downstream trajectory tracking, while retaining end-to-end trainability. Studies examined closed-loop behavior in urban driving, data aggregation for on-policy corrections, and multi-task/multi-modal conditioning to cover diverse traffic behaviors and junction types [50–52]. At the same time, research connected E2E to safety engineering: measuring scenario coverage and failure modes, integrating interpretable signals, and stress-testing robustness to natural distribution shift and adversarial perturbations [37, 53, 54].

E. 2022–Present: Transformers, World Models, and Industrialization

Recent trends push E2E beyond pure behavioral cloning. Transformer backbones and sequence models unify perception and planning with longer temporal context; large-scale data engines emphasize targeted collection, re-labeling, and continuous evaluation under safety constraints. Industrial roadmaps describe how E2E components integrate with production safety cases (monitoring, fallback behaviors, ODD definitions) and with classical modules where needed [55]. Public briefings and technical notes describe production-flavored E2E stacks (e.g., surround camera transformers, BEV fusion, and waypoint heads coupled to classical tracking/control) and emphasize massive regression suites [56]. Emerging *vision-language-action* and related foundation-style approaches point to policies that reason over structured maps, agents, and textual constraints, while bridging training across sim and real data [42].

F. Evaluation: From Open-Loop MAE to Closed-Loop & Safety Metrics

Open-loop metrics (e.g., steering MAE) correlate weakly with closed-loop driving quality; modern practice emphasizes *closed-loop* evaluation in sim or proving grounds, multi-scenario benchmarking (junctions, roundabouts, merges), rare-event mining, and performance under perturbations. Safety-aligned evaluation integrates scenario coverage, interpretable diagnostics, and robustness tests (e.g., natural adversarial shifts), and reports regression across large scenario catalogs [32, 37, 54, 57].

V. Sensor Classification and Evolution

Sensors form the perceptual backbone of autonomous driving systems, enabling the detection, tracking, and classification of objects, as well as localization and mapping in dynamic environments. Over the past three decades, sensor technology has evolved from basic monocular cameras and short-range ultrasonic sensors to sophisticated multi-modal perception suites combining LiDAR, radar, cameras, GNSS, and inertial measurement units (IMUs).

A. Taxonomy and Roles

Autonomous driving relies on three sensing pillars: (i) exteroceptive sensors that perceive the environment (cameras, lidars, radars, ultrasonic); (ii) proprioceptive sensors estimating ego-state (IMU/INS, wheel odometry, GNSS/RTK); and (iii) cooperative sources (V2X and map priors) that extend perception beyond line of sight. This taxonomy clarifies complementary strengths: cameras excel at texture and semantics, lidar at geometry, radar at velocity and adverse-weather robustness, while localization sensors and HD maps stabilize state estimation and global context [4, 58–60].

B. Camera Sensors (RGB, HDR, Fisheye, Event, Thermal)

a) Principles and signals.: Cameras provide dense appearance cues (texture, color, edges) fundamental to lane, sign, and object understanding. Surround-camera rigs grew from narrow FOV front cameras to multi-focal, fisheye side/rear views for complete coverage, with HDR (High Dynamic Range) pipelines and global shutters to mitigate motion artifacts and glare [58]. Event cameras offer microsecond-level, HDR change detection; thermal cameras aid in low-light and through-glare scenarios [58, 61].

b) Evolution.: Advancements include: (1) *HDR imaging* for sun/glare; (2) migration from rolling to *global shutter* in critical channels; (3) *calibrated surround* rigs with tighter timebases; (4) better ISP pipelines and learned deglare/denoise; (5) exploration of *event/thermal* as complementary streams in night and adverse weather [58, 61]. Datasets (e.g., KITTI, Oxford RobotCar, KITTI-360) standardized evaluation under varied lighting and seasonal changes, pushing algorithms to be robust to photometric shifts [62–64].

c) Limits/mitigation.: Cameras degrade under low light, fog, and direct sun. HDR, polarization, thermal augmentation, and fusion with radar/lidar mitigate these effects; careful synchronization and accurate ego-motion compensation reduce rolling-shutter distortions in motion [58].

C. LiDAR (ToF, Solid-State, FMCW, Wavelengths)

a) Principles and signals.: LiDAR measures range/intensity via time-of-flight (ToF) returns, yielding metrically accurate 3D geometry. Rich point clouds support detection, freespace estimation, and map building; multi-echo returns and intensity help in partial occlusions and rain [58, 62, 65].

b) Evolution.: Systems progressed from spinning mechanical to *solid-state* (MEMS, OPA, flash) for lower cost/size, improved reliability, and easier integration. Debates around 905 nm vs. 1550 nm balance eye-safety, range, and weather performance. *FMCW LiDAR* adds instantaneous radial velocity and improved interference immunity, narrowing a traditional radar advantage. Adverse-weather modeling has improved (multi-echo, denoising, learned completion), and simulation now injects realistic LiDAR artifacts for training and testing [58, 59, 66].

c) Limits/mitigation.: Dense precipitation and soiling degrade returns; careful placement, cleaning/heating, multi-echo filtering, and fusion (occupancy/BEV) improve robustness [58, 66].

D. Automotive Radar (mmWave, 4D/Imaging, Doppler)

a) Principles and signals.: 76–81 GHz mmWave radar measures range and Doppler directly, offering resilience to rain/fog and giving velocity at long range. Modern MIMO arrays and sparse recovery lead to *4D/imaging radar* with elevation resolution and finer angular detail [67–70].

b) Evolution.: Radar evolved from low-resolution ACC sensors to dense grids approaching image-like angular resolution. Micro-Doppler signatures and learned radar perception improved small-object detection and cross-traffic awareness, while interference mitigation and calibration pipelines matured for large fleets [58, 67, 68].

c) Limits/mitigation.: Multipath/ghosting and limited angular resolution remain challenges; BEV fusion with camera/lidar, temporal filtering, and learned denoisers reduce false positives [66, 68].

E. Ultrasonic and Near-Field Proximity

Ultrasonic sensors remain valuable for low-speed maneuvers and near-field coverage around bumpers. They provide short-range, low-cost redundancy for parking and tight maneuvers, complementing cameras and radar at very close distances [58].

F. Localization: GNSS/IMU/INS and HD Maps

a) Principles.: GNSS (often RTK) and IMU/INS fusion stabilize global and short-term state estimation; wheel odometry further constrains drift. *HD maps* (lane-level topology, traffic controls, static 3D) provide priors for localization and planning; visual/lidar SLAM can reduce map dependence or maintain consistency under outages [4, 59].

b) Evolution and practice.: RTK corrections, multi-constellation receivers, and high-grade IMUs reduced drift; modern fleets maintain *map freshness* with continuous updates. Recent industry reports detail GNSS/IMU integration options, antenna placement, and practical considerations in urban canyons [71].

G. V2X Sensing as Cooperative Perception

C-V2X/DSRC extends perception via *cooperative awareness*, e.g., broadcasting trajectories/intent of nearby vehicles or RSU-based blind-spot warnings. While not a substitute for on-board sensing, V2X improves safety envelopes at occluded intersections and supports priority/signal cooperation in connected corridors [4, 60].

H. Sensor Fusion and Representations (Early/Mid/Late, BEV, Occupancy)

a) Design space.: Fusion choices span *early* (raw), *mid-level* (features), and *late* (decisions). Mid-level BEV representations became standard for unifying modalities and supporting downstream planning. Occupancy grids and neural fields integrate geometry (lidar/radar) and semantics (camera) with uncertainty, improving detection, freespace, and closed-loop stability [19, 66, 72].

b) Calibration, synchronization, timebase.: Accurate *intrinsic/extrinsic* calibration and stable time synchronization (hardware triggers/PTP, motion-compensated timestamping) are essential to avoid modality misalignment that degrades fusion. Practical pipelines emphasize on-vehicle re-calibration, clock monitoring, and dataset curation with precise timebases [58, 73].

I. Cost/Power/Compute Trends and System Iteration

Camera-first stacks dominate cost–performance, with lidar/radar selectively added for redundancy and ODD expansion. Solid-state lidar and imaging radar have reduced cost/size, while compute/bandwidth constraints drive choices in sensor counts, resolution, and compression. Industry whitepapers and regional reports track declining costs, port logistics for sensor supply, and fleet integration practices that balance Bill of Materials (BOM) with coverage and safety [26, 27, 74, 75].

J. Environmental Robustness and Failure Modes

Rain/fog/snow attenuate lidar and camera; glare saturates camera ISPs; radar handles weather but faces multipath and clutter. Robust systems employ *heterogeneous redundancy*, self-checks (soiling/fogging detection), and graceful degradation policies. Scenario catalogs and safety analyses emphasize sensing failure taxonomy and corresponding mitigations in design and test campaigns [5, 59, 66].

K. Historical Throughline and Outlook

From Dickmanns/ALVINN-era vision to today's multimodal, map- and V2X-aware stacks, sensor evolution reflects a steady push toward higher fidelity, wider coverage, and better *calibrated* fusion. Modern designs combine camera-centric semantics with lidar/radar geometry and velocity, stabilized by GNSS/IMU and augmented by cooperative perception—an architecture that aligns with both classical modular pipelines and modern end-to-end learning [7, 8, 58, 60].

VI. Legal and Regulatory Frameworks

This section contains the following subsections: (i) regulatory *archetypes* and scopes, (ii) testing and deployment governance, (iii) safety assurance and standards, (iv) cybersecurity, software updates, and data protection, (v) liability, insurance, and accountability, and (vi) regional perspectives and convergence.

A. Regulatory Archetypes and Scope

Global approaches cluster into two archetypes. The self-certification model (e.g., U.S.) relies on manufacturers/operators to certify compliance with federal safety principles and guidance, supported by post-market enforcement and supplemental state-level rules; NHTSA's policy frameworks outline expectations for safety assessment, data, and transparency without prescribing a single technical path [76, 77]. The type-approval model (e.g., UNECE/WP.29 in Europe and many adopting countries) defines approval regimes and technical regulations before market entry; ADS deployment hinges on conformity with harmonized regulations and approval of system updates [78, 79]. China's national guidance situates ADS pilots within administrative permits, safety supervision, and data governance requirements, with MIIT and local authorities issuing graduated rules for testing, demonstration, and commercial services [41].

B. Testing, Pilots, and Deployment Governance

Both sources emphasize staged progression from closed-course testing to public-road pilots and limited commercial service, bounded by an *operational design domain* (ODD) that specifies geography, road class, speed, weather, and lighting. Jurisdictions typically require safety drivers or remote assistance in early phases, incident reporting protocols, and clear disengagement/behavior taxonomies to support oversight and learning [54, 80]. Municipal playbooks and legal briefs highlight permit structures, data-sharing obligations, and community engagement, noting that emergency measures around COVID sometimes accelerated sandbox-like pilots and digital coordination between agencies and operators [81].

C. Safety Assurance, Standards, and the Safety Case

Modern frameworks converge on goal-based assurance: developers present a *safety case*—a structured, evidence-backed argument that risks are reduced to acceptable levels in the declared ODD. Evidence spans design-time analyses, scenario-based testing (sim, track, public road), performance monitoring, and update governance. Industry and research guidance elaborates scenario catalogs, coverage metrics, and toolchains for automated safety analysis, while legal scholarship stresses traceability and explainability in decisions that affect public risk [40, 54, 57, 82].

D. Cybersecurity and Software Updates

The UNECE/WP.29 regime introduced explicit approval requirements for cybersecurity management systems and software update management systems, operationalized through regulations adopted by many markets. These provisions require processes for threat analysis, patch management, secure over-the-air updates, and post-market incident handling—preconditions for ADS deployment and continued conformity [78, 79]. U.S. guidance likewise calls for lifecycle cybersecurity risk management and update safety assurances in the absence of pre-market type approval [76].

E. Data Protection, Privacy, and Localization

Large-scale AV operations collect sensitive telemetry and environmental data. Governance therefore spans privacy-by-design architectures, purpose limitation, retention/minimization, and access controls, alongside disclosure and redress mechanisms. In the EU, GDPR shapes consent, DPIAs, and data-subject rights for in-cabin and external sensing. In China and other jurisdictions, cybersecurity and data-export controls impose localization and cross-border transfer conditions that affect cloud pipelines and remote support [2, 41]. Legal analyses underscore the need to reconcile safety transparency with privacy and trade-secret protections [82].

F. Liability, Insurance, and Accountability

As control transitions from human to ADS, liability allocations evolve among driver/operator, manufacturer, and service provider [83]. Early U.S. case analyses and scholarly treatments proposed frameworks for crash responsibility, forensic data access, and evidentiary standards; state statutes (e.g., early California bills) enabled testing while preserving avenues for tort claims [84]. Contemporary guidance links *software update* provenance, *ODD compliance*, and *human oversight roles* (e.g., remote assistance) to fault attribution and insurance product design [82, 85]. Public-law instruments (incident reporting, safety case review) complement private-law remedies to sustain accountability [54, 80].

G. Regional Perspectives and Convergence

United States: federal guidance with post-market enforcement and state-level pilot regimes; emphasis on transparency, cybersecurity, and operational safety assurances rather than prescriptive design [76, 77]. Europe/UNECE: pre-market approvals for ADS and mandatory cybersecurity/update systems; structured conformity assessments and oversight of over-the-air updates [78, 79]. China: national guidance and local roadmaps for scaled pilots and commercialization under administrative permits, with integrated safety supervision and data controls [2, 41]. Cross-regionally, the trend is towards goal-based safety cases, standardized reporting, and lifecycle governance for software-intensive vehicles [40, 57, 82].

VII. Problem Statement and Research Questions

Despite notable progress in E2E driving, several challenges remain unresolved, particularly under resource constraints:

- Many E2E systems assume powerful computing platforms and large-scale curated datasets that are difficult to replicate in academic or pilot deployments.
- Multi-step maneuvers such as dead-end recovery, parallel parking, and roundabout navigation in E2E autonomous driving still face significant challenges.
- Simulation environments (e.g., CARLA and AWSIM) differ in realism, performance, ROS2 integration, and sensor modeling; their impact on E2E development and sim-to-real transfer is not fully understood.
- There is limited evidence on the feasibility and limitations of deploying lightweight, camera-only E2E architectures on real suburban shuttle routes within a constrained operational design domain.

These gaps motivate the following research questions:

1) **RQ1.** How can lightweight, camera-based end-to-end driving systems be designed, trained, and deployed under constraints of limited data, limited compute, and low-cost hardware?
2) **RQ2.** What architectural enhancements (e.g., temporal memory, spatiotemporal encoding, Mixture-of-Experts) are required to improve robustness in complex navigation tasks that involve sequential decision-making?
3) **RQ3.** How can simulation environments (CARLA and AWSIM) be systematically benchmarked and integrated to support scalable model development, scenario testing, and sim-to-real transfer?
4) **RQ4.** To what extent can compact, monocular camera–based end-to-end frameworks achieve safe, interpretable, and reliable closed-loop performance on real suburban shuttle routes?

These research questions provide the conceptual foundation for the remainder of this thesis.

VIII. Research Aims and Objectives

The overarching aim of this thesis is to investigate the practicality of deploying lightweight E2E driving systems on low-cost autonomous platforms, particularly shuttle buses operating in constrained suburban and campus environments. This aim is pursued through the following objectives:

- **O1:** Develop baseline and memory-enhanced E2E models to evaluate the role of temporal reasoning in sequential navigation tasks.
- **O2:** Establish a simulation-supported development workflow and evaluate CARLA and AWSIM, using common digital twins, for E2E training and ROS2 integration [32, 39, 86].
- **O3:** Design and deploy a Mixture-of-Experts driving framework on a real shuttle platform, including dataset construction, model training, and interpretability analysis.
- **O4:** Assess the performance, limitations, and safety-related behaviors of camera-only E2E models across simulation, scaled platforms, and real-world suburban shuttle trials.

IX. Research Scope and Significance

The research scope is intentionally focused on low-speed autonomous shuttle operations in constrained operational design domains. In particular, the thesis considers:

- campus-style and suburban residential routes with narrow roads, limited lane markings, and frequent occlusions;
- modest speeds and limited driving envelopes appropriate for shuttle services;
- camera-only or camera-dominant sensor configurations, with LiDAR and other sensors used primarily for safety curtaining and system monitoring rather than as primary inputs to the E2E network.

Within this scope, the thesis does *not* claim to provide general safety guarantees for all autonomous driving scenarios. Instead, it offers:

- architectural insights into lightweight E2E models that can be implemented on embedded platforms;

- a reproducible sim-to-real pipeline that connects EyeSim, CARLA, AWSIM, ModCar, and full-scale shuttle buses [32, 39, 86];
- empirical evidence on the benefits and limitations of temporal models and MoE architectures under realistic resource constraints;
- a nuanced discussion of what can reasonably be concluded about autonomy, robustness, and interpretability within the tested operational design domains, framed against broader safety and regulatory perspectives [21, 22].

X. Autonomous Driving Platforms in the REV Project and the RobLab

The autonomous driving platforms within the UWA Renewable Energy Vehicle (REV) Project and the Robotics and Automation Lab (RobLab) at the University of Western Australia (UWA) currently comprise three simulators and four autonomous shuttle buses. These platforms provide both virtual and physical testbeds for evaluating novel neural network architectures, validating perception and control algorithms.

A. Simulators

Given the uncertainty in performance and optimization time of newly designed neural network architectures, simulators serve as indispensable prototyping environments. They allow rapid iteration, safe validation of corner cases, and scalable testing before transferring models to physical vehicles.

1) EyeSim Simulator: EyeSim [86] is a mobile robot simulator originally developed for the EyeBot family of robots. It supports the simulation of mobile platforms equipped with cameras, 2D LiDAR, and position-sensitive detectors (PSDs), along with an LCD interface for data visualization. While more limited in fidelity compared to modern driving simulators, EyeSim remains valuable for algorithm prototyping and educational purposes due to its lightweight design and ease of deployment.

2) CARLA Simulator: CARLA [32] is an open-source autonomous driving simulator built on Unreal Engine. It provides high-fidelity rendering, multiple city-scale maps, and a wide range of vehicles and sensor modalities, including RGB and depth cameras, LiDARs, RADARs, GNSS and IMU units, collision and lane-invasion detectors, semantic and instance segmentation cameras, dynamic vision sensors (DVS), and optical flow sensors. Within this project, CARLA has been extended to include a detailed digital twin of the UWA campus, the Eglington route, and a 3D shuttle bus model. These additions enable testing of neural network models in both structured campus layouts and suburban road scenarios, providing a close approximation of real-world operational environments.

3) AWSIM Simulator: AWSIM [39] is a Unity-based, open-source simulator developed by Tier IV for use with the Autoware autonomous driving stack. It emphasizes extensibility and real-time integration with Autoware modules, making it well-suited for system-level validation. Similar to CARLA, custom digital replicas of the UWA campus, Eglington route, and shuttle bus have been developed for AWSIM. Comparative use of CARLA and AWSIM supports both model-level experimentation and integration-level evaluation, bridging research environments with industry-grade development pipelines.

B. nUWAy Shuttle Buses

The Renewable Energy Vehicle (REV) team operates four autonomous shuttle buses, branded as *nUWAy*, built on EasyMile hardware and customized with in-house software. These shuttles enable real-world validation of autonomous driving research in both campus and suburban environments.

The fleet currently includes:

- nUWAy1 (2020) and nUWAy2 (2022): EZ10 Gen 1 vehicles equipped with a front and rear grayscale camera, an IMU, GPS, four wheel speed sensors, and eight LiDAR units.
- nUWAy3 (2024) and nUWAy4 (2025): EZ10 Gen 2 vehicles with sensor suites similar to the Gen 1 but undergoing upgrades to a four-camera GMSL RGB configuration. These vehicles are not yet operational.

Due to the timing of this thesis, experiments are limited to nUWAy1 and nUWAy2.

1) Campus Shuttle — nUWAy1: nUWAy1 has been primarily deployed on the UWA campus. This setting introduces unique challenges such as:

- Shared pathways with pedestrians and cyclists.
- Lack of formal lane markings, complicating visual perception.
- Frequent occlusions from trees and tall buildings, which interfere with lighting and GPS signals.
- Pedestrians distracted by smartphones or headphones, resulting in unpredictable interactions with the shuttle.

These conditions create a demanding environment for evaluating perception robustness, pedestrian safety, and low-speed navigation.

2) On-Road Shuttle — nUWAy2: nUWAy2 has been deployed in the suburban residential area of Eglington. The primary test route connects Amberton Beach with the Stockland Sales Office, featuring:

- Narrow residential streets with parked vehicles.
- Roundabout-style intersections requiring give-way behaviors.
- Frequent junctions and non-orthogonal turns.
- Scenarios involving parallel parking and constrained maneuvering space.

These characteristics make the route ideal for evaluating common low-speed driving maneuvers, including pull-ins, pull-outs, and roundabout navigation, under conditions of constrained perception and limited localization accuracy.

XI. Chapter Synopses

The works comprising this dissertation describe the creation of the necessary components and methods for the development of three simulators (EyeSim, CARLA, AWSIM), one toy car (ModCar), and two autonomous shuttle buses (nUWAy1, nUWAy2). This dissertation is structured around four articles, each addressing a different aspect of autonomous driving research within the UWA Renewable Energy Vehicle (REV) Project and the Robotics and Automation Lab (RobLab). Together, these chapters trace the progression from early platform development and simulation-supported experimentation, through the design of memory-augmented end-to-end models, to comparative simulator analysis, and finally the deployment of a mixture-of-experts framework on a full-scale autonomous shuttle bus. Collectively, they showcase a research trajectory that integrates hardware platforms, simulation environments, neural network innovation, and closed-loop real-world validation, ultimately advancing the feasibility of lightweight, camera-only solutions for suburban autonomous mobility.

Chapter 2 Temporal Memory Models on the ModCar Platform extends end-to-end autonomous driving beyond PilotNet-style architectures by incorporating temporal memory for complex indoor tasks. While PilotNet can match LiDAR-based control in simple scenarios, its lack of temporal context limits robustness in environments requiring sequential decision-making, such as dead-end recovery or multi-step maneuvers. To address this, two memory-enhanced models were proposed: (1) a CNN+LSTM architecture that couples spatial feature extraction with temporal recurrence, and (2) a 3D-CNN that jointly encodes spatiotemporal dependencies from video sequences. These models were first validated in the EyeSim simulator to enable safe and repeatable indoor trials, and subsequently evaluated on the ModCar platform equipped with camera and LiDAR sensors, allowing controlled comparisons across manual, LiDAR-driven, and neural policies. Results showed that memory-based models significantly outperformed PilotNet in complex navigation tasks, demonstrating the necessity of temporal awareness for reliable end-to-end driving. This chapter directly addresses **RQ2** by demonstrating how temporal modeling improves robustness over purely frame-based E2E approaches, while continuing to respect the lightweight and camera-centric constraints of **RQ1**.

Chapter 2 Publication: Lai, Z., Bräunl, T. End-to-End Learning with Memory Models for Complex Autonomous Driving Tasks in Indoor Environments. J Intell Robot Syst 107, 37 (2023). https://doi.org/10.1007/s10846-022-01801-2 (published)

Chapter 3: Vision-Based and AI-Based Shuttle Navigation presents the earlier development of UWA's autonomous shuttle bus project, focusing on vision-based navigation, deep learning methods, and simulation-supported testing. Two complementary approaches were implemented: (1) a traditional computer vision pipeline using Canny edge and Hough line detection for lane tracking, and (2) an AI-based method using a modified PilotNet neural network trained in TensorFlow to map camera images to steering and throttle commands directly. In parallel, a CARLA-based simulation environment with digital replicas of the UWA campus and Amberton Beach, supporting multiple driving modes (manual, LiDAR, vision, neural network, and mirror mode) was developed for controlled experimentation. Results demonstrated that neural network navigation significantly outperformed classical vision algorithms in both simulation and real-world trials. On-campus tests compared GPS, GPS+RTK, SLAM, and neural networks, with the latter two showing the highest autonomy but different failure characteristics. Initial road trials at Amberton Beach validated the feasibility of end-to-end neural models, though further data collection and sorting are required for reliable intersection and parallel parking handling. This chapter primarily addresses **RQ1** and provides baseline evidence of the benefits and shortcomings of feed-forward E2E models.

Chapter 3 Publication: Quirke-Brown, K., Lai, Z., Kong, X., Tan, T., Du, Y., Bräunl, T. Developing an Autonomous Shuttle Service. Australasian Transport Research Forum 2023 Proceedings, Perth, 29 Nov-1 Dec 2023 (published)

Chapter 4 Comparative Analysis of Autonomous Driving Simulators presents a comprehensive comparative study of two leading open-source autonomous driving simulators: CARLA (Unreal Engine) and AWSIM (Unity). Using identical digital twins of the Amberton Beach route, this work evaluated the simulators across quantitative and qualitative metrics, including computational efficiency, realism of physics and sensors, scenario creation, user-friendliness, and extendability. Results show that CARLA offers high-fidelity rendering, extensive APIs, and rich scenario customization, making it strong for scalable end-to-end testing. In contrast, AWSIM demonstrates lower setup overhead, smoother real-time performance, native ROS2 Humble integration, and superior LiDAR simulation, positioning it as an effective platform for LiDAR-based algorithms and Autoware development. The study highlights how the underlying game engines—Unreal's realism versus Unity's flexibility—shape simulator strengths, and establishes a set of weighted comparison criteria that future researchers can adapt. This chapter primarily addresses **RQ3**, establishing a structured methodology for simulator selection and highlighting how simulation choices influence E2E model development and validation.

Chapter 4 Publication: Lai, Z., Le, L., Silva, V., Bräunl, T. A Comprehensive Comparative Analysis of Carla and Awsim: Open-Source Autonomous Driving Simulators. Robotics and Autonomous Systems (under review, submitted on 26 May 2025)

Chapter 5 Mixture-of-Experts End-to-End Driving on nUWAy2 develops and evaluates a lightweight, grayscale camera–based end-to-end driving system for an autonomous shuttle operating on a 4.1 km roundabout-style suburban loop. A Mixture-of-Experts (MoE) framework—comprising specialized experts for lane following/pull-out, pull-in (with an empty-bay classifier), and reverse—runs on a dual-computer nUWAy platform (industrial PC + Jetson Orin, ROS 2), with safety LiDAR curtaining and CAN integration. Using a 600 GB/40 h dataset curated via staged auto+manual sorting and targeted augmentation, this study benchmark EfficientNetV2-based models in both open-loop (MAE, latency, classifier accuracy) and closed-loop (autonomy rate, manual interventions, e-stops) trials. The best-performing MoE (front-camera C0F + hybrid classifier–regression C1F1 + front-camera C2F) balances autonomy, safety, and interpretability; dual- and rear-camera variants provide limited gains given latency and data scarcity. Saliency analyses confirm attention to bays, markings, and obstacles, and results underscore that closed-loop testing reveals compounding errors unseen in offline metrics. The chapter concludes that compact, camera-only MoE pipelines are viable for cost-effective suburban shuttles and outlines future work on unified branched models, temporal sequence learning, improved moving-vehicle detection, and broader deployment. This chapter directly addresses **RQ1**, **RQ2**, and **RQ4**, and also discusses the limitations of generalising safety claims beyond the tested conditions.

Chapter 5 Publication: Lai, Z., Quirke-Brown, K., Kong, X., Le, L., Bräunl, T. Shuttle Bus Performing Common Maneuvers in Roundabout-Style Suburban Residential Area Using Grayscale Camera-Based End-to-End Driving. IEEE Transactions on Intelligent Vehicles (under review, submitted on 11 Sep 2025)

Chapter 6 summarizes the contributions of this thesis and offers suggestions for potential future research avenues.

REFERENCES

[1] Dentons, "Global autonomous vehicle guide 2025," 2025. [Online]. Available: https://www.dentons.com

[2] C. Securities, "Industry strategy for autonomous driving: The singularity moment arrives?" 2023. [Online]. Available: https://www.cs.ecitic.com

[3] K. Bimbraw, "Autonomous cars: Past, present and future a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology," in *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, vol. 01, pp. 191–198. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7350466

[4] M. Yang, K. Jiang, B. Wijaya, T. Wen, J. Miao, J. Huang, C. Zhong, W. Zhang, H. Chen, and D. Yang, "Review and challenge: High definition map technology for intelligent connected vehicle," p. S2667325824000268. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2667325824000268

[5] D. Garikapati and S. S. Shetiya, "Autonomous vehicles: Evolution of artificial intelligence and the current industry landscape," vol. 8, no. 4, p. 42, publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2504-2289/8/4/42

[6] PreWarCar. [Online]. Available: https://www.prewarcar.com/the-mystery-of-the-radio-controlled-american-wonder

[7] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems*, vol. 1. Morgan-Kaufmann. [Online]. Available: https://proceedings.neurips.cc/paper/1988/hash/812b4ba287f5ee0bc9d43bbf5bbe87fb-Abstract.html

[8] E. D. Dickmanns and V. Graefe, "Dynamic monocular machine vision," vol. 1, no. 4, pp. 223–240. [Online]. Available: https://doi.org/10.1007/BF01212361

[9] Timeline: Historia de la robotica. [Online]. Available: https://www.timetoast.com/timelines/historia-de-la-robotica-9cc7de3d-b17a-4e4c-8fbb-958f1474bb7d

[10] P. Varaiya, "Smart cars on smart roads: problems of control," vol. 38, no. 2, pp. 195–207. [Online]. Available: https://ieeexplore.ieee.org/document/250509

[11] S. E. Shladover, "Automated vehicles for highway operations (automated highway systems)," vol. 219, no. 1, pp. 53–75, publisher: IMECHE. [Online]. Available: https://doi.org/10.1243/095440705X9407

[12] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. Van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA grand challenge," vol. 23, no. 9, pp. 661–692. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/rob.20147

[13] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the urban challenge," vol. 25, no. 8, pp. 425–466. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20255

[14] "The DARPA urban challenge: Autonomous vehicles in city traffic." [Online]. Available: https://link.springer.com/10.1007/978-3-642-03991-1
[15] The drive for autonomous vehicles: The DARPA grand challenge | HeroX. [Online]. Available: https://www.herox.com/blog/159-the-drive-for-autonomous-vehicles-the-darpa-grand
[16] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars." [Online]. Available: http://arxiv.org/abs/1604.07316
[17] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2722–2730, ISSN: 2380-7504. [Online]. Available: https://ieeexplore.ieee.org/document/7410669
[18] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 163–168, ISSN: 1931-0587. [Online]. Available: https://ieeexplore.ieee.org/document/5940562
[19] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6526–6534, ISSN: 1063-6919. [Online]. Available: https://ieeexplore.ieee.org/document/8100174
[20] P. A. Securities, "Policy and technology spiral progress: Emerging commercial loop of high-level autonomous driving," 2024. [Online]. Available: https://stock.pingan.com
[21] T. Litman, "Autonomous vehicle implementation predictions: Implications for transport planning." [Online]. Available: https://www.vtpi.org/avip.pdf
[22] J3016_202104: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles - SAE international. [Online]. Available: https://www.sae.org/standards/content/j3016_202104/
[23] D. Stine. BIM chapters: Recap of NVIDIA GTC 2018 in silicon valley. [Online]. Available: https://bimchapters.blogspot.com/2018/03/recap-of-nvidia-gtc-2018-in-silicon.html
[24] neuralmobile, "Xpeng - highway autopilot sensor suite." [Online]. Available: https://www.youtube.com/watch?v=rab8BUdlOt4
[25] C. P. T. Group, "Vehicle-road-cloud integrated l4 bus system joint commissioning guidelines," 2025. [Online]. Available: http://www.cdgjbus.com
[26] E. Intelligence, "China smart mining autonomous driving industry market research report 2024," 2024. [Online]. Available: https://www.iyiou.com
[27] ——, "China autonomous driving in port scenarios: Commercialization case studies," 2024. [Online]. Available: https://www.iyiou.com
[28] B. Gerkey, R. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," *Proceedings of the International Conference on Advanced Robotics*, 08 2003. [Online]. Available: https://www.researchgate.net/publication/2876085_The_PlayerStage_Project_Tools_for_Multi-Robot_and_Distributed_Sensor_Systems
[29] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154 vol.3. [Online]. Available: https://ieeexplore.ieee.org/document/1389727
[30] SUMO – simulation of urban MObility: An overview. [Online]. Available: https://www.researchgate.net/publication/225022282_SUMO_-_Simulation_of_Urban_MObility_An_Overview
[31] R. Rajamani, *Vehicle Dynamics and Control*, ser. Mechanical Engineering Series. Springer US. [Online]. Available: https://link.springer.com/10.1007/978-1-4614-1433-9
[32] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator." [Online]. Available: http://arxiv.org/abs/1711.03938
[33] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation

for autonomous vehicles." [Online]. Available: http://arxiv.org/abs/1705.05065

[34] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1082–10 828, ISSN: 2160-7516. [Online]. Available: https://ieeexplore.ieee.org/document/8575297

[35] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," vol. 23, no. 6, pp. 4909–4926. [Online]. Available: https://ieeexplore.ieee.org/document/9351818

[36] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta, E. Agafonov, T. H. Kim, E. Sterner, K. Ushiroda, M. Reyes, D. Zelenkovsky, and S. Kim, "LGSVL simulator: A high fidelity simulator for autonomous driving," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9294422

[37] S. Feng, X. Yan, H. Sun, Y. Feng, and H. X. Liu, "Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment," vol. 12, no. 1, p. 748, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41467-021-21007-8

[38] D. D. Tran, T. Tomita, and T. Aoki, "Safety analysis of autonomous driving systems: A simulation-based runtime verification approach," pp. 1–15. [Online]. Available: https://ieeexplore.ieee.org/document/10981812

[39] "tier4/AWSIM," 2024, original-date: 2022-10-14T00:48:24Z. [Online]. Available: https://github.com/tier4/AWSIM

[40] C. A. Technology, R. Center, and Huawei, "Safety models for autonomous driving 2024," 2024. [Online]. Available: https://www.catarc.ac.cn

[41] C. A. I. C. V. I. I. Alliance, "China intelligent and connected vehicle autonomous driving simulation testing white paper," 2023. [Online]. Available: http://www.caicv.org.cn

[42] H. Securities, "When autonomous driving meets robotics: Detailed analysis of vla and world models," 2024. [Online]. Available: https://www.huayuanstock.com/

[43] K. Securities, "World models: The next stop of autonomous driving," 2024. [Online]. Available: http://www.kaiyuansec.com

[44] Y. Securities, "In-depth report on the intelligent driving industry: The route to autopilot evolving toward end-to-end, data flywheel becomes crucial," Yongxing Securities, Tech. Rep., 2024. [Online]. Available: https://www.yongxingsec.com

[45] H. Securities, "Vehicle–road–cloud industry observation (3): Vehicle–road–cloud integration leads china's new infrastructure construction and supports the implementation of high-level autonomous driving," Haitong Securities, Tech. Rep., 2024. [Online]. Available: https://www.gtht.com

[46] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning." [Online]. Available: http://arxiv.org/abs/1710.02410

[47] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3530–3538, ISSN: 1063-6919. [Online]. Available: https://ieeexplore.ieee.org/document/8099859

[48] J. Kim and J. Canny, "Interpretable learning for self-driving cars by visualizing causal attention," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, pp. 2961–2969. [Online]. Available: http://ieeexplore.ieee.org/document/8237582/

[49] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," vol. 3, no. 4, pp. 4423–4430. [Online]. Available: https://ieeexplore.ieee.org/document/8458422

[50] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*,

pp. 8248–8254, ISSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/document/8793742

[51] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah, and A. Kndall, "Urban driving with conditional imitation learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 251–257, ISSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9197408

[52] A. Prakash, K. Chitta, and A. Geiger, "Multi-modal fusion transformer for end-to-end autonomous driving," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 7073–7083. [Online]. Available: https://ieeexplore.ieee.org/document/9578103/

[53] S. Casas, A. Sadat, and R. Urtasun, "MP3: A unified model to map, perceive, predict and plan," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 14 398–14 407. [Online]. Available: https://ieeexplore.ieee.org/document/9577532/

[54] P. Koopman and B. Osyk, "Safety argument considerations for public road testing of autonomous vehicles," vol. 1, no. 2, pp. 512–523. [Online]. Available: https://www.sae.org/content/2019-01-0123

[55] C. A. R. C. M. Baidu, "White paper on applications of high-level autonomous driving," 2023, white paper. [Online]. Available: https://www.caeri.com.cn

[56] L. Zheshang Securities Co., "Ai industry series: The large-model era of intelligent driving — toward end-to-end architectures," 2023, industry research report. [Online]. Available: https://www.stocke.com.cn

[57] T. Sever and G. Contissa, "Automated driving regulations – where are we now?" vol. 24, p. 101033. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2590198224000198

[58] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," vol. 21, no. 6, p. 2140, publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2140

[59] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," vol. 17, no. 4, pp. 1135–1145. [Online]. Available: https://ieeexplore.ieee.org/document/7339478

[60] S. Adnan Yusuf, A. Khan, and R. Souissi, "Vehicle-to-everything (v2x) in the autonomous vehicles domain – a technical review of communication, sensor, and AI technologies for road user safety," vol. 23, p. 100980. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2590198223002270

[61] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," vol. 44, no. 1, pp. 154–180. [Online]. Available: https://ieeexplore.ieee.org/document/9138762

[62] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, ISSN: 1063-6919. [Online]. Available: https://ieeexplore.ieee.org/document/6248074

[63] 1 year, 1000 km: The oxford RobotCar dataset. [Online]. Available: https://journals.sagepub.com/doi/epub/10.1177/0278364916679498

[64] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d." [Online]. Available: http://arxiv.org/abs/2109.13410

[65] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast segmentation of 3d point clouds: A paradigm on LiDAR data for autonomous vehicle applications," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5067–5073. [Online]. Available: https://ieeexplore.ieee.org/document/7989591

[66] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Gläser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," vol. 22, no. 3, pp. 1341–1360. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9000872

[67] S. M. Patole, M. Torlak, D. Wang, and M. Ali, "Automotive radars: A review of signal processing techniques," vol. 34, no. 2, pp. 22–35. [Online]. Available: https:

//ieeexplore.ieee.org/document/7870764
[68] L. Fan, J. Wang, Y. Chang, Y. Li, Y. Wang, and D. Cao, "4d mmWave radar for autonomous driving perception: A comprehensive survey," vol. 9, no. 4, pp. 4606–4620. [Online]. Available: https://ieeexplore.ieee.org/document/10477463
[69] T. Securities, "4d millimeter wave radar: The standard sensor balancing cost and performance," 2024. [Online]. Available: https://www.tfzq.com
[70] K. Securities, "4d imaging millimeter wave radar: New sensors empowering autonomous driving," 2024. [Online]. Available: http://www.kaiyuansec.com
[71] H. Navigation and H. Securities, "Surveying as the base: High-precision positioning opens growth space for autonomous driving," 2024. [Online]. Available: https://www.hazq.com
[72] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection." [Online]. Available: http://arxiv.org/abs/2012.10992
[73] Z. Wang, Y. Wu, and Q. Niu, "Multi-sensor fusion in automated driving: A survey," vol. 8, pp. 2847–2868. [Online]. Available: https://ieeexplore.ieee.org/document/8943388
[74] G. Securities, "Sensor cleaning technology: Towards standardization in l4 autonomous driving," 2024. [Online]. Available: http://www.gjzq.com.cn
[75] M. Securities, "Tesla full self-driving: Development timeline, technical principles and future outlook," 2024. [Online]. Available: https://www.mszq.com
[76] "Automated driving systems: A vision for safety."
[77] J. M. Anderson, N. Kalra, K. D. Stanley, P. Sorensen, C. Samaras, and T. A. Oluwatola, "Autonomous vehicle technology: A guide for policymakers." [Online]. Available: https://www.rand.org/pubs/research_reports/RR443-2.html
[78] UN regulation no. 157 - automated lane keeping systems (ALKS) | UNECE. [Online]. Available: https://unece.org/transport/documents/2021/03/standards/un-regulation-no-157-automated-lane-keeping-systems-alks
[79] Framework document for automated/autonomous vehicles (UPDATED) | UNECE. [Online]. Available: https://unece.org/transport/publications/framework-document-automatedautonomous-vehicles-updated
[80] N. E. Vellinga, "From the testing to the deployment of self-driving cars: Legal challenges to policymakers on the road ahead," vol. 33, no. 6, pp. 847–863. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0267364917301334
[81] H. Securities, "Strategic technology (communication) industry weekly report: Contactless delivery as a potential killer app for autonomous driving," 2020. [Online]. Available: https://www.hazq.com
[82] M. L. Roth, "Regulating the future: Autonomous vehicles and the role of government."
[83] M. L. Kubica, "Autonomous vehicles and liability law," vol. 70, pp. i39–i69. [Online]. Available: https://academic.oup.com/ajcl/article/70/Supplement_1/i39/6655619
[84] SB-1298 vehicles: autonomous vehicles: safety and performance requirements. [Online]. Available: https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201120120SB1298
[85] A. Jones, "Autonomous cars: Navigating the patchwork of data privacy laws that could impact the industry," vol. 25, no. 1. [Online]. Available: https://scholarship.law.edu/jlt/vol25/iss1/6
[86] T. Bräunl, "Eyesim vr," 2022. [Online]. Available: https://robotics.ee.uwa.edu.au/eyesim/

Chapter 2

End-to-End Learning with Memory Models for Complex Autonomous Driving Tasks in Indoor Environments

Preamble: *Building on the platform groundwork and simulation capability established in the prior work, this chapter moves from static, frame-based end-to-end control toward temporal reasoning. We introduce memory-augmented models (CNN+LSTM and 3D-CNN), first validating them in EyeSim for safe, repeatable experiments, then transferring to ModCar for controlled indoor trials with camera/LiDAR. This chapter demonstrates why sequence awareness is critical for dead-end recovery and multi-step maneuvers, setting up the transition to full-scale shuttle experiments in the next chapter, where we compare classical vision pipelines with learned end-to-end control. This chapter directly addresses* ***RQ2*** *by demonstrating how temporal modeling improves robustness over purely frame-based E2E approaches, while continuing to respect the lightweight and camera-centric constraints of* ***RQ1****.*

Publication: *Lai, Z., Bräunl, T. End-to-End Learning with Memory Models for Complex Autonomous Driving Tasks in Indoor Environments. J Intell Robot Syst 107, 37 (2023).* `https://doi.org/10.1007/s10846-022-01801-2` *(published)*

Journal of Intelligent & Robotic Systems (2023) 107:37
https://doi.org/10.1007/s10846-022-01801-2

SHORT PAPER

End-to-End Learning with Memory Models for Complex Autonomous Driving Tasks in Indoor Environments

Zhihui Lai[1] · **Thomas Bräunl**[1]

Received: 13 April 2022 / Accepted: 23 December 2022

Abstract
The interest in autonomous vehicles has increased exponentially in recent years. While Lidar is a proven autonomous driving technology, end-to-end learning approaches have become popular as computer performance has improved. A fully end-to-end method—NVIDIA's PilotNet has shown its ability to predict speed and steering angle with only camera images. This method achieved the Lidar-based methods' performance in simple driving tasks. However, a significant drawback was no past spatiotemporal information, imposing an error-sensitive performance, especially in complex driving tasks. Spurred by this deficiency, this paper introduces two novel models: CNN + LSTM and CNN3D, aiming for complex autonomous driving tasks in indoor environments.

Keywords Deep learning · CNN · End to end learning · Autonomous driving · LSTM · CNN3D

1 Introduction

The interest in autonomous vehicles (AVs) has increased exponentially in recent years. Even though AVs can not fully replace human drivers at the current stage, there is an irreversible trend towards full driving automation. AVs are an important research area because they have the potential to make transport safer and more efficient while also reducing carbon emissions. In 2018, the US Department of Transportation National Highway Traffic Safety Administration (NHTSA) reported that 94% of severe traffic accidents are caused by human error in the US [1]. AVs or computer-assisted driving systems can significantly reduce fatalities in accidents and thus are promising to solve this issue [2]. However, there are still many challenges to overcome, as the demand for AVs is not only to be better than human drivers but also to be economical. Hence, the technology that can solve these two problems simultaneously is still an open research case.

Currently, most AV companies are using Lidar to collect data. The Lidar sensor is accurate at sensing distances of the surrounding environment, but extremely expensive [3], while an alternative sensor is a camera. Since computers nowadays can learn from a large dataset, using a camera to collect self-driving data for deep learning algorithms is cheap and significantly reduces the time to program traditional driving algorithms. If camera-based deep learning algorithms can reach the performance of Lidar-based algorithms, then there is a high probability that we can afford to market AVs.

In 2016 NVIDIA introduced PilotNet, an end-to-end Convolutional Neural Network (CNN) widely used in image analysis [4]) that extracts raw pixels from a single front camera image as input and produces steering commands as output [5, 6]. This network has been proven incredibly powerful, but there is still much space for improvement, e.g., adding memory to increase driving consistency so it can deal with more complex scenarios and perform a sequence of actions smoothly. The hypothesis is that driving in a sequence of actions needs memory to perform well. The upgrade to PilotNet can be implemented in two ways: extend the 2D convolutional layers to 3D convolutional layers (CNN3D) or add Long Short-Term Memory (LSTM) after the 2D convolutional layers. Both methods have the potential to achieve better steering results with more profound spatiotemporal information. LSTM has been developed to solve common RNNs shortcomings.

✉ Zhihui Lai
zhihui.lai@research.uwa.edu.au

Thomas Bräunl
thomas.braunl@uwa.edu.au

[1] School of Engineering, The University of Western Australia, 35 Stirling Hwy, Crawley, Perth, 6009, WA, Australia

Published online: 11 March 2023

LSTM uses a cell, an input gate, an output gate, and a forget gate for its calculations. These features allow it to achieve long-term memory and eliminate the "Vanishing Gradient Problem" [7]. LSTM networks are usually used to classify and predict time information. If there is a lag of unknown duration between the basic features in the time series, the LSTM network can also capture it well. In addition, the relative insensitivity to gap length is a significant advantage of LSTM over RNN and other sequence learning methods in many fields. CNN3D is a neural network that uses 3D filters (kernels). This network is beneficial in image processing because 3D filters can extract spatial and temporal features from a sequence of image inputs hence obtaining action information [8].

In 2017, an RC car equipped with a Lidar sensor and a Raspberry Pi named ModCar (Fig. 1) was developed as a research project at the University of Western Australia (UWA). This project provided a Lidar-based autonomous driving baseline. Later, this project utilized NVIDIA's PilotNet and performed end-to-end deep learning research by modifying the vehicle with a wide-range camera. This project provided a camera-based autonomous driving baseline. The vehicle now has a Lidar drive mode, manual mode, and PilotNet-drive mode. These features satisfy the prerequisites for further autonomous driving research.

In simple driving tasks, such as lane following and driving between two walls, PilotNet works fine and has the same performance as the Lidar-based autonomous driving algorithm. However, for complex driving tasks in indoor environments, such as turning around at a dead-end or recovering from a malfunction, PilotNet fails. This paper continues the research on end-to-end visual navigation and proposes two novel models: CNN + LSTM and CNN3D, aiming for complex autonomous driving tasks in indoor environment. Our source code is available at: https://github.com/zhi-hui-lai/ModCar_Project.git.

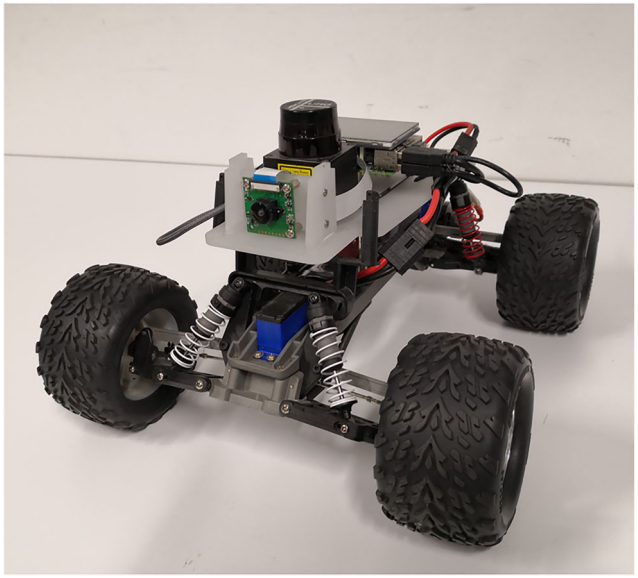


Fig. 1 ModCar autonomous driving robot

2 Related Work

Among the earliest groundwork for end-to-end driving is ALVINN [9], which maps images to direct steering control through a fully-connected network. Then NVIDIA extended this approach deeply with a Convolutional Neural Network (CNN) called PilotNet [5]. After PilotNet, researchers attempted numerous different approaches to improve end-to-end driving performance.

Different input configurations have been tried. Maqueda et al. employed an event camera to collect images over a specified time interval as an input to a CNN [10]. Codevilla et al. utilized conditional imitation learning and recorded commands as extra neural network input [11]. Wang et al. also implemented imitation learning and described a subgoal to guide their angle-branched architecture [12]. Yang et al. applied a multi-modal multi-task network that included feedback speed sequence as additional input [13]. Wang et al. proposed an auto-encoder for eliminating irrelevant roadside objects in images before feeding into a CNN [14]. Drews et al. integrated CNN, cost maps, particle filter, IMUs, and wheel speed measurement in their autonomous system [15].

Chi et al. developed a neural network architecture by combining spatio-temporal convolution, convolutional LSTM and LSTM [16]. Hou et al. proposed a novel fast recurrent fully convolutional network (F-RFCN) [17]. Okamoto et al. proposed a combined self-driving approach by converting a frontal-view image to a top-view image with a CNN and then feeding it to a dynamic Bayesian network (DBN) to compute affordances [18].

Different output configurations have been tried (not necessarily direct control output). These approaches are in the partially end-to-end learning category. Chen et al. replaced direct control commands with affordance information in their neural network output [19]. Chen et al. inserted a cognitive map between CNN and RNN in their neural network to detect the traffic conditions [20]. Weiss et al. developed a novel end-to-end deep learning model, AdmiralNet, combining the original PilotNet, LSTM, and 3D convolution. This network works well on the photo-realistic F1 racing simulator and the 1/10 scale racecar testbed. Unlike the original PilotNet predicting steering commands, they use AdmiralNet to predict Bezier Curves from pixels directly and compared four different deep learning methods: PilotNet (pixel-to-control), CNN-LSTM (pixel-to-control), CNN-LSTM (pixel-to-waypoint), and

CNN-LSTM (pixel-to-Bezier curve trajectory). The results are impressive, with AdmiralNet Bezier Curve Predictor outperforming all other methods [21]. This experiment shows that the performance of an autonomous driving system can be significantly improved by adding memory and predicting trajectories rather than control commands.

Some researchers added auxiliary tasks along with the control command prediction task. Xu et al. presented a novel FCN-LSTM network that takes advantage of semantic segmentation (as an auxiliary task) [22]. Chen et al. considered auxiliary tasks, including image segmentation, transfer learning, optical flow, and additional vehicle information [23].

Some researchers experimented with end-to-end methods on different tasks. Pierre proved the feasibility of robot-following with an end-to-end approach by comparing several models, including CNN, stacked CNN, Conv3D, and RNN [24]. Jeong et al. proposed a classifier to predict whether the next lane is free [25].

Jaritz et al. used deep reinforcement learning to drive a robot in the WRC6 racing game [26]. Liang et al. applied Controllable Imitative Reinforcement Learning (CIRL) to drive a robot in the CARLA simulator [27]. Sallab et al. implemented deep reinforcement learning to drive a robot in the TORCS racing game [28].

Nie et al. proposed an Integrated Multimodality Fusion Deep Neural Networks (IMF-DNN) framework that can handle both objection detection (as an auxiliary task) and driving behaviors (steering angle and speed) prediction by fusing camera and Lidar data [29]. Sobh et al. tested several camera lidar fusion methods in the CARLA simulator and found the fusion of Polar Grid Mapping (PGM) and semantic segmentation performed the best [30].

This paper aims to solve new tasks in an indoor environment with memory models, including driving in the center between two walls and making a three-point turn at a dead-end. Several models involving PilotNet, CNN+LSTM (ours), CNN+LSTM Weiss et al. [21], and CNN3D (ours) are investigated. Weiss' CNN-LSTM (pixel-to-Bezier curve trajectory) model has excellent performance in fast racing games, however, it requires positioning equipment on the actual car to collect waypoints that our ModCar doesn't have. So this paper only compares the pixel-to-control methods.

3 Design

3.1 ModCar

Figure 2 represents the simplified ModCar control flow, UWA's ModCar uses the RoBIOS GUI [31] as the low-level control program, so users can efficiently operate it via a touch screen or gamepad. The main interface is shown in Fig. 3a; there are three driving modes: manual, camera neural network, and Lidar. As presented in Fig. 3b, the manual drive mode uses the gamepad to control the speed and steer the robot, and image recordings are possible at a 30Hz frame rate. Lidar drive mode is visualized in Fig. 3d; it uses the Lidar sensor to measure the distance of the surrounding environment and then inputs these distances into a fine-tuned algorithm to calculate the speed and

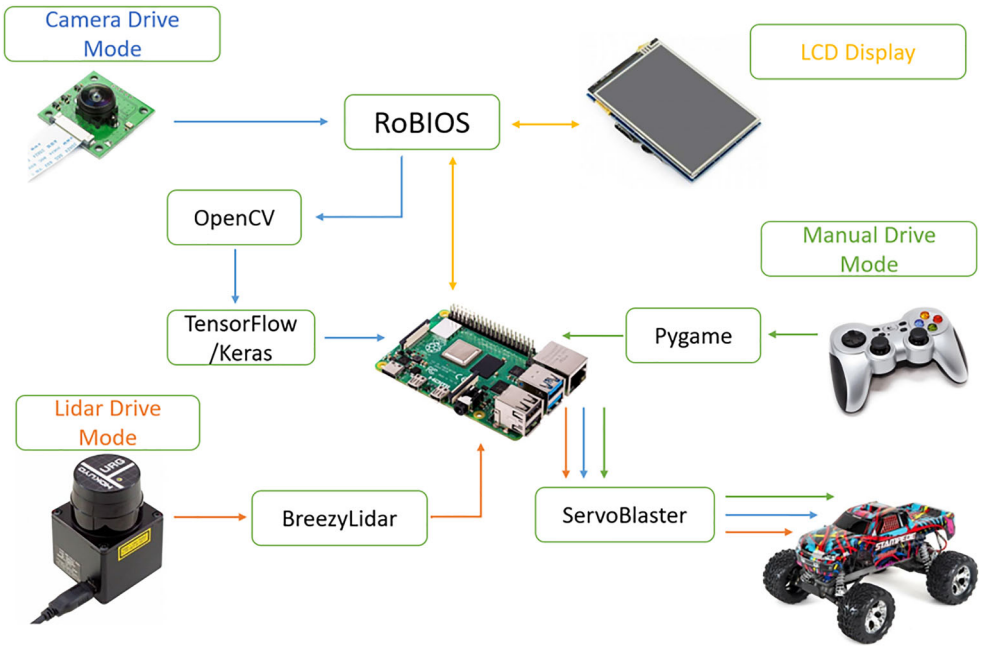


Fig. 2 Simplified control flow for ModCar, arrows of different colors indicate the control flow of the three driving modes

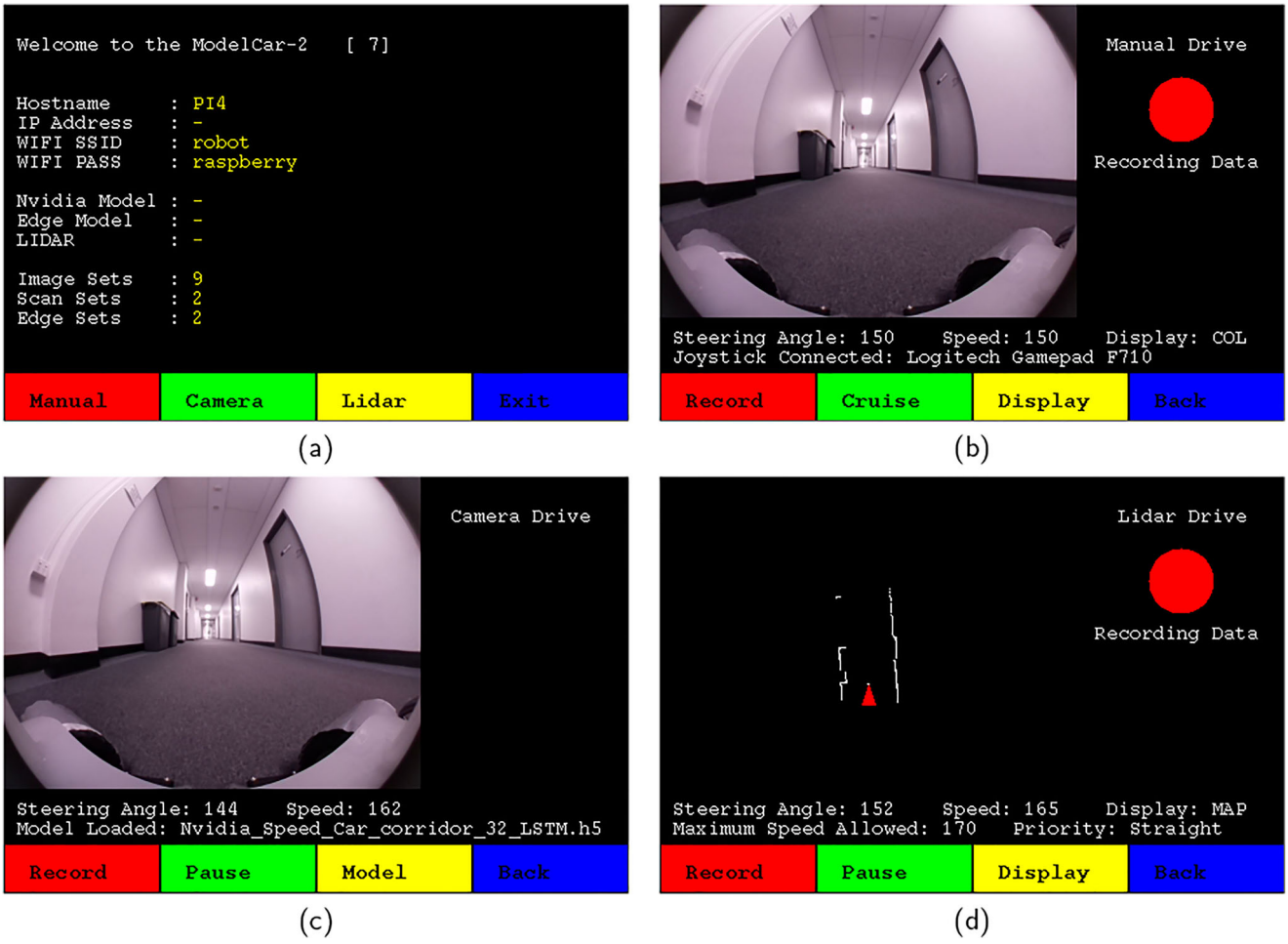


Fig. 3 ModCar program: (a) Main interface (b) Manual drive mode (c) Camera neural network drive mode (d) Lidar drive mode

steering of the robot. The camera neural network drive mode is illustrated in Fig. 3c; it uses the camera to capture images and then feeds these images into a trained neural network to generate the speed and steering of the robot.

3.2 Neural Network Models

This part introduces the neural network models implemented in the project. The previous work experimented with the original PilotNet using speed as additional output and dropout for the deep network's normalization (regularization). We adopt this architecture with extra batch normalization, which is illustrated in Fig. 4a. The input is an image in YUV color space with a height of 200 pixels, a width of 66 pixels and 3 color channels. Input normalization converts the image pixel values from [0, 255] 'uint8' type to [-1, 1] 'float' type. The activation function used by the model is 'ELU' instead of 'RELU' for better performance [32], and its output range is [-1, ∞] when alpha equals one. If the input is not normalized, the activation function will encounter errors [33], so the next two models also applied input normalization.

The CNN3D model (Fig. 4b) uses 3D convolutional layers instead of 2D, with the additional dimension employed as a timeline. Specifically, when the input is an image sequence, it obtains both spatial and temporal information. The GlobalAveragePooling3D layer is similar to the GlobalAveragePooling2D layer but has an additional pooling dimension.

The CNN+LSTM model (Fig. 4c) involves an LSTM layer between the fully connected layers (dense layer) and the 2D convolutional layers. Furthermore, the input changes from a single image to an image sequence (in this case, five images in a row), where the current image and the past four images are input into five convolutional layers separately, then input into the LSTM layer together. The LSTM layer can extract information along the timeline and assist the model in making better decisions. This model also uses GlobalAveragePooling2D layers to replace the Flatten layers, where the latter layer transforms a multi-dimensional

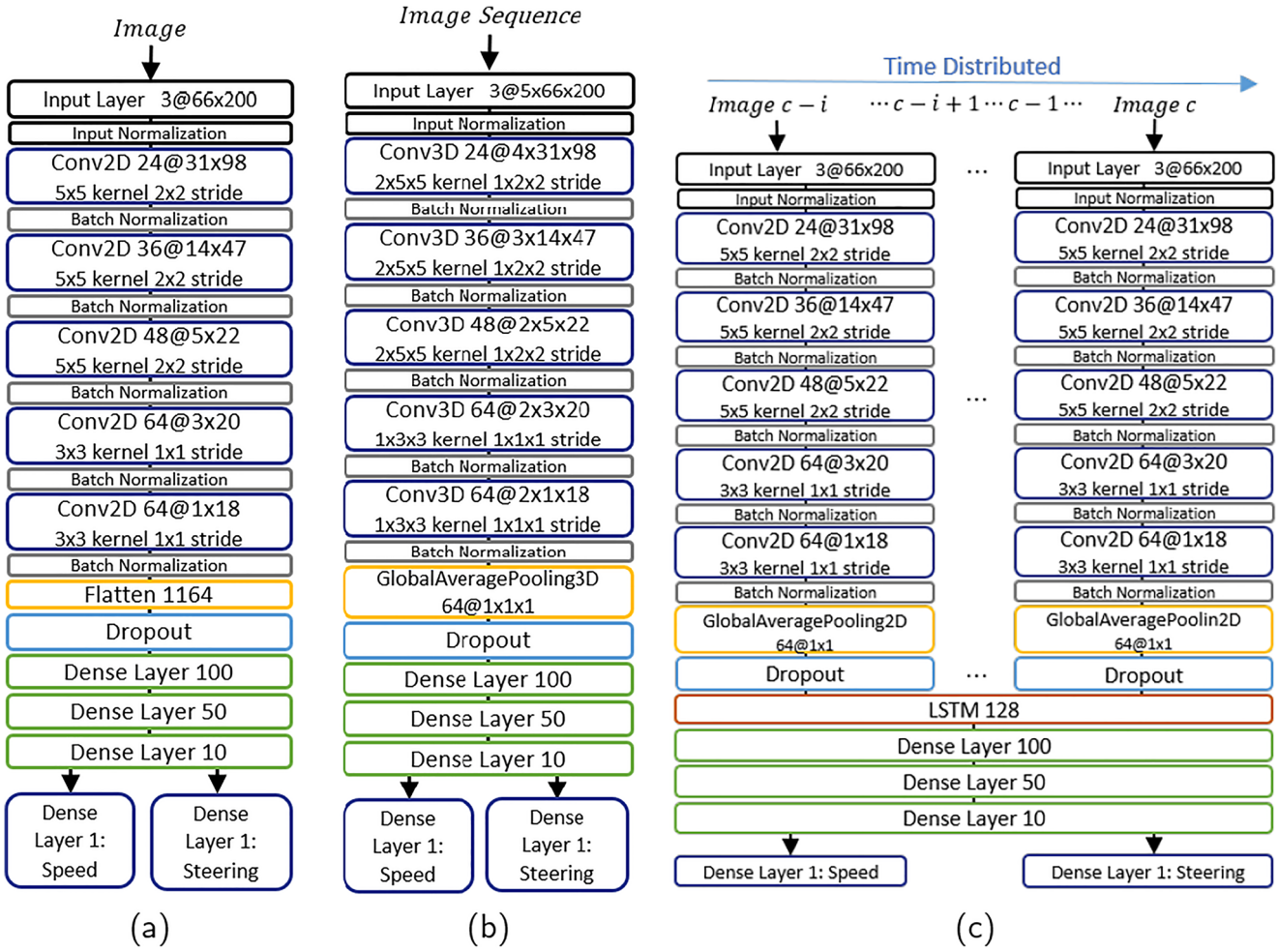


Fig. 4 Neural network architectures: (a) Original PilotNet (b) CNN3D (c) CNN+LSTM (image c: current image, i = sequence length-1)

tensor into a one-dimensional tensor. This transformation has a risk of overfitting, especially for complex models. However, the GlobalAveragePooling2D layer solves this problem because it sums up spatial information and has no parameters to optimize. Batch normalization transforms the data to a zero mean and a unit variance, reducing the "Vanishing or Exploring Gradient Problem" [33], affording a higher learning rate and speeding up the entire training process.

4 Methodology

4.1 Simulation

Before any practical experiments, it is necessary to try the neural network models in simulation, e.g., the EyeSim platform [34]. The latter is a mobile robot simulator for the EyeBot family that can simulate a mobile car, which has attached a camera, Lidar, a PSD sensor, and an LCD screen for the camera and data display.

4.1.1 Data Collection

A well-designed autonomous driving algorithm is preferred against a manual button-controlled drive to reduce human error when obtaining training data for the neural network. Thus, we apply a simulated Lidar-based algorithm to drive the ModCar in the maze map as depicted in Fig. 5. During this process, we exploit the camera to capture images and store them with a labeled steering angle and speed until adequate training images are captured; the speed and steering angle data distribution is illustrated in Fig. 6. The simulation experiment collected 5,110 clockwise driving images and 5,166 anti-clockwise driving images (10,276 images in total). It should be noticed that the Lidar drive mode captures 700 images per lap with a frame rate of 10Hz, and thus, 10,276 images require 15 laps to collect,

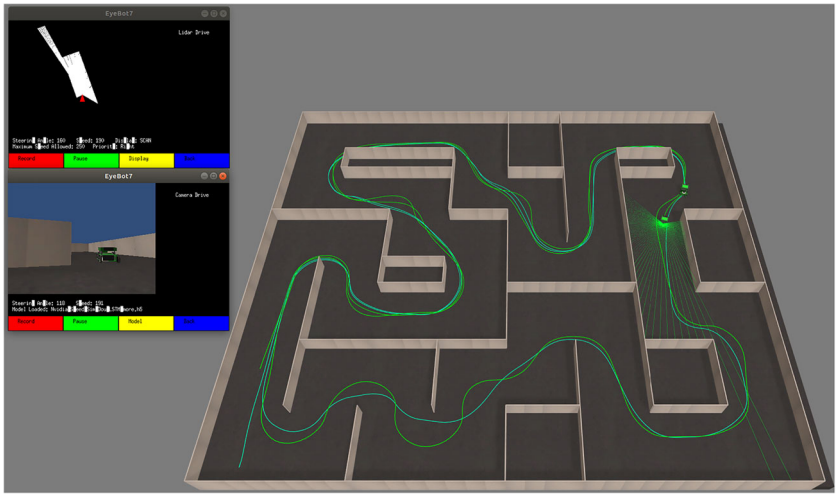


Fig. 5 EyeSim maze map

presenting an adequate training data density. The captured images with labeled steering angle and speed are stored in the npy format, a NumPy-array format that is small and expandable with additional information like labels. The speed, ranging from (100, 150, 250), corresponds to (backward, stop, forward). The steering angle, ranging from (100, 150, 200), corresponds to (right turning, straight, left turning). Before the neural network training, the images are randomly split into three datasets: training, validation, and testing, with the latter involving 0.4% of the total images, and the remaining imagery is split into a 4:1 training-to-validation ratio.

4.1.2 Data Preprocessing

The driving program using RoBIOS API captures 320x240 RGB images. As shown in Fig. 7 (left), each image is first converted to the YUV color space, then passes through a Gaussian blurring filter, and is finally resized to 200×66 pixels by the OpenCV resize function. The YUV color space helps adjust the brightness. The Gaussian blurring filter smoothes the resizing process and prevents image distortion [35]. Given that 200×66 is the input size of the original PilotNet, the entire PilotNet architecture is built based on this size, so there is no need to modify that. Additionally, a smaller size can help reduce the computational load, especially when generating image sequences for CNN+LSTM and CNN3D models.

4.1.3 Data Augmentation

Data augmentation increases the dataset's size by generating data variants. This can improve the model's generalization and robustness [36], but excessive augmentation generates much noise preventing the model from learning. Thus, the extent of data augmentation needs to be well considered. As illustrated in Fig. 7 (right), our simulation experiment considers two data augmentation types: blurring and

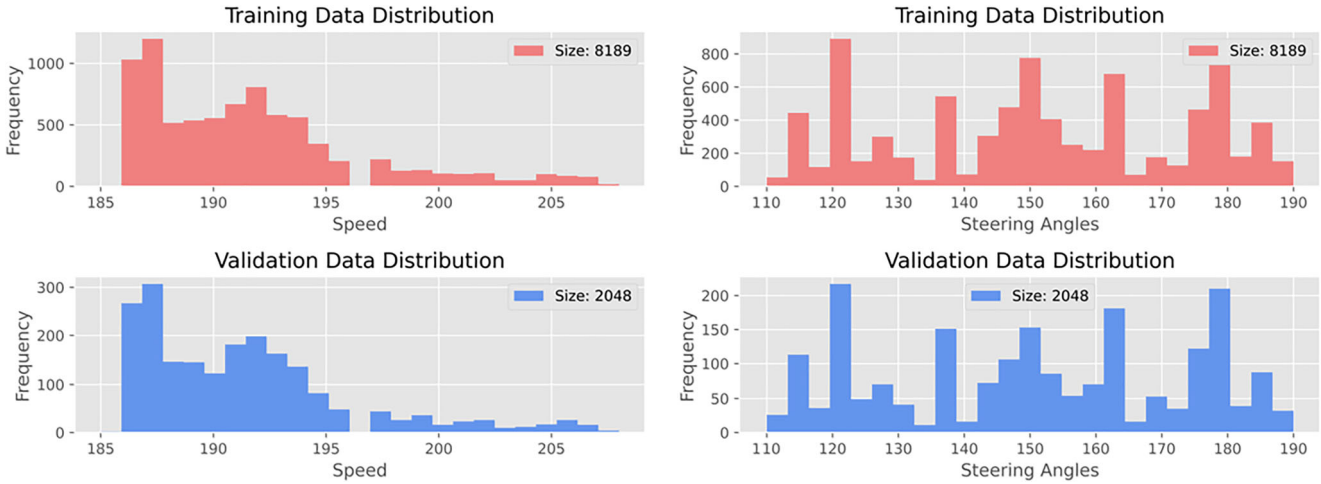


Fig. 6 EyeSim maze map: speed (left) and steering angle (right) data distribution

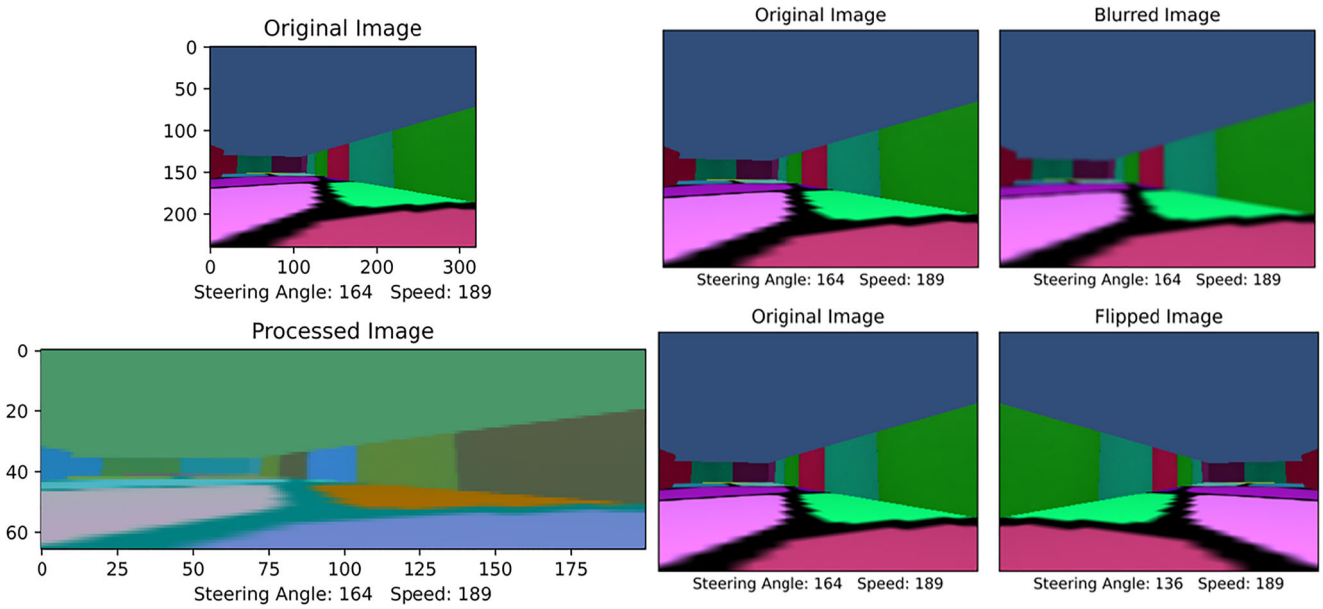


Fig. 7 Simulation image preprocessing (left), simulation image augmentation (right)

flipping. Blurring helps the model learn more holistic features without overlearning the minutiae [35], while image flipping balances the uneven distribution of left and right steering data, and eliminates the dataset's inherent bias for assisting the model's better learning [37]. Each augmentation is randomly applied to 5% of the training data in a single batch.

4.1.4 Image Sequence Generation

To generate an image sequence for the CNN+LSTM and CNN3D models, we arrange the collected pictures in order and use a sequence-length window to slide from the beginning to the end by sliding one image at a time. Each sequence uses the labels of the last image.

4.1.5 Model Training

The model training employs the bagging strategy. Bagging or bootstrap aggregation refers to randomly sampling the dataset with replacement in each batch [33], significantly reducing the dataset's variance. Besides, avoiding overfitting and saving training time can be achieved by including early stopping, learning rate decaying, batch normalization, and dropout. This project has four neural network models to train. Thus, we first fit the data into each model with its specific inputs and then train it on Ubuntu 18.04 with one NVIDIA TITAN X (Pascal) GPU. Once training finishes, we copy the models to the EyeSim model folder for further exploitation. Table 1 indicates basic parameter settings for model training.

Table 1 Basic parameter settings for model training

Hyperparameter	Value/Function
Batch size	32
Optimizer	Adam
Epochs	100
Dropout	0.2
Input normalization	f(x) = x/127.5 - 1
Activation	ELU function; $f(x) = \alpha*(e^x - 1)$ for $x < 0$; $f(x) = x$ for $x \geq 0$; α: Scale for the negative factor
Early stopping	If the validation loss does not improve within 10 epochs, stop
Learning rate schedule	If the validation loss does not improve within 4 epochs, reduce it to $\frac{1}{5}$ of the original. (Initial: 0.001; Min: 0.000001)
Training steps	Number of training image / batch size
Validation steps	Number of validation image / batch size

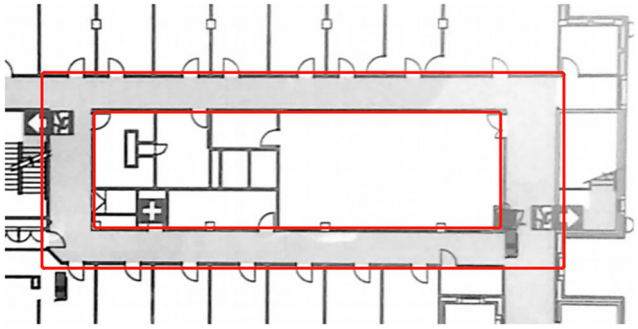


Fig. 8 UWA EECE 4th floor plan

4.1.6 Validation and Comparison Methods

Open-loop metrics consider the following:

1. Separate the training set and validation set with an 0.8 training ratio and then calculate the Mean Square Error (MSE) of each model.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left[\alpha \times (g_i - \hat{g_i})^2 + \beta \times (h_i - \hat{h_i})^2\right] \quad (1)$$

(MSE = Mean Squared Error;
n = number of data points;
g_i = true speed values;
$\hat{g_i}$ = predicted speed values;
h_i = true steering values;
$\hat{h_i}$ = predicted steering values;
α and β are coefficients, and the default value is one;)

The training loss (MSE) and validation loss (MSE) are calculated in each epoch during the model training by the Keras API.

Closed-loop metrics consider the following:

1. Mean Lap time [21]: if a lap is not finished, the label "did not finish" (DNF) is placed.
2. Autonomy [3]: The percentage of time the network model drives the vehicle without human intervention.

$$Autonomy = \left(1 - \frac{NoI \times HRT[S]}{ET[S]}\right) \times 100\% \quad (2)$$

(NoI = Number of Interventions;
HRT = Human Reset Time;
S = seconds; ET = Elapsed Time.)

The ModCar driving program has a pause function. Each time the robot hits a wall or stops moving, the pause button will be pressed and counted as an intervention. We set the manual reset time to 5 seconds. The elapsed time is the total time minus the pause time.

4.2 Practical Experiments

This project considers three practical experiments: driving in a rectangular loop and turning around at two corridors; one of the corridor is a non-trained environment.

4.2.1 Data Collection

We apply the Lidar-based algorithm to drive ModCar on the 4th floor of the EECE building and the ground floor of the CME building at UWA.

The 4th floor of the EECE building has a rectangular loop for the Lidar drive mode to run continuously (Fig. 8 red zone). We prioritized right-turning first in the Lidar drive mode and collected 30,679 clockwise driving images. Then we prioritized left-turning first and collected 30,518 anti-clockwise driving images (61,197 images in total). The Lidar driving mode captures 590 images per lap at a frame rate of 10Hz, and thus, 61,197 images require 104 laps to be collected. This data density far exceeds the simulated data density because the natural environment is complicated, and data is unevenly distributed. As illustrated in Fig. 9, the steering angle in most pictures is marked as 150, i.e., the car is driving in a straight line. This unbalance causes the model to ignore a few turns and fails a correct learning process.

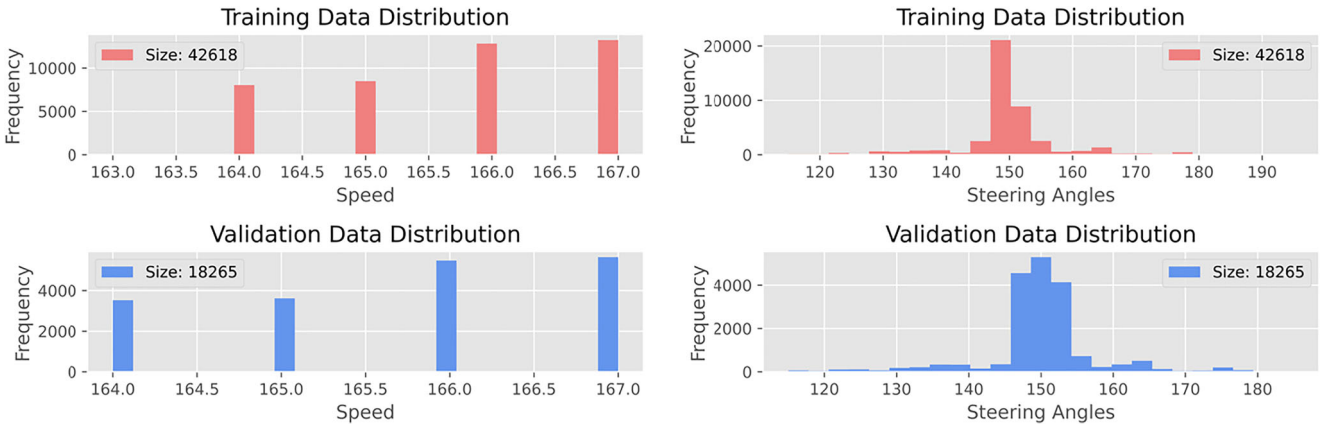


Fig. 9 EECE building: speed data (left) and steering angle data (right) distribution

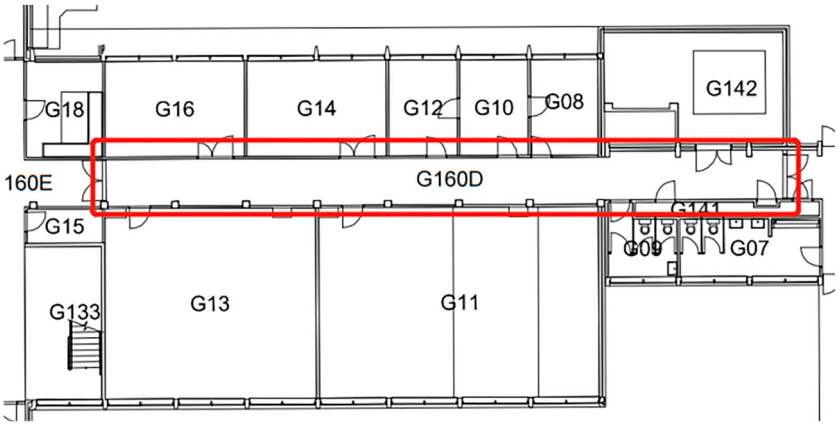


Fig. 10 UWA CME ground floor plan

Thus, more data and data augmentation are both needed to compensate for this bias.

The ground floor of the CME building has a corridor (Fig. 10 red zone) which is not wide enough for the robot to make a complete U-turn, so the robot uses a three-point turn strategy as shown in Fig. 11. It stays in the middle of the walls on both sides. If it encounters a dead-end, it turns left until it approaches the wall, turns right, reverses, and finally turns left to the center of the road to repeat the initial steps. For this scenario, we trained two versions of neural network models. For version one (V1), we collected 47,838 images. The Lidar drive mode captures 560 images in one round trip with a frame rate of 10Hz. Thus, 47,838 images need 86 rounds to be collected. The data density is sufficient, but due to the design defects of the Lidar's driving program, when encountering a dead end, the robot has a 50% chance to turn left and a 50% chance to turn right. This inconsistent behavior is not conducive to the training of the neural network models, with data cleaning reducing this effect. Hence, we manually delete the images where the car turns to the right and simply duplicate the images where the car turns to the left to compensate for the loss of images. For version two (V2), we collected 75,845 images. The Lidar drive mode captures 560 images in one round trip with a frame rate of 10Hz. Thus, 75,845 images need 135 rounds to be collected. In this version, we use a different strategy to eliminate the inconsistent behavior of the Lidar program, namely image flipping: flip all right turns to left turns. The collected speed and steering angle data distribution is presented in Fig. 12.

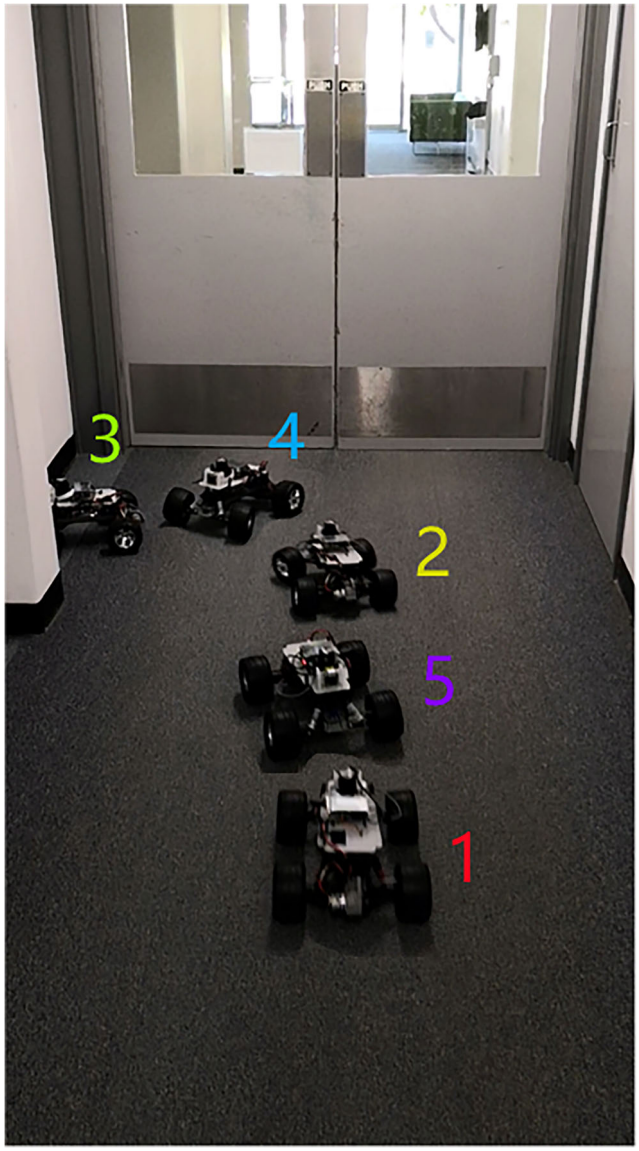


Fig. 11 ModCar 3-point-turn steps 1-5

4.2.2 Data Preprocessing

As shown in Fig. 13, the practical data preprocessing is mainly the same process as for simulation, except that the images are taken by a fisheye lens, requiring additional image processing to eliminate lens distortion. Precisely, the images are cropped 25% of the top and 31.25% of

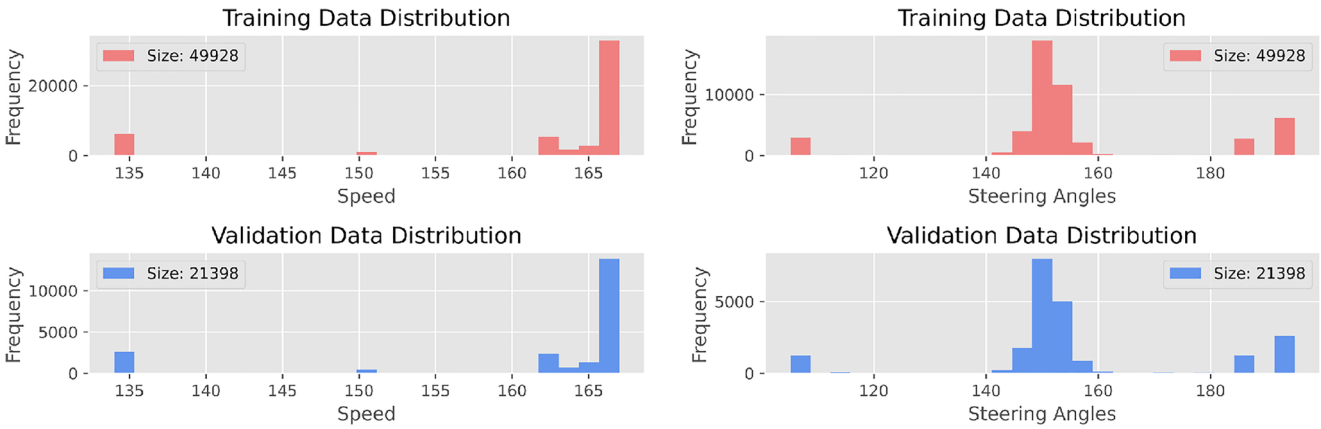


Fig. 12 CME building: speed data (left) and steering angle data (right) distribution

the bottom before being converted into YUV color space. This step removes the highly distorted parts and spinning wheels from the images. Yang et al. pointed out that the computational cost of neural network training can be greatly reduced by identifying important features and cutting out irrelevant features [38]. Apart from this step, the images are resized to 230x66 instead of 200x66 for additional data augmentation.

4.2.3 Data Augmentation

Practical experiments consider additional data augmentation process: brightness changing (Fig. 14), image shifting and rotating (Fig. 15). As the brightness of both places changes due to sunlight, we randomly modify the image brightness within the range of $[-20\%, +20\%]$ and utilize brightness modification on 5% of the training data in a single batch. Besides, the images collected by Lidar autonomous driving algorithm are always at the center of the road; there is not enough diversity of different lane positions. When the car deviates from such a trained trajectory, it cannot recover. While NVIDIA solved the diversity problem by image shifting and rotating in 2020 [39], this paper uses a simpler method to achieve the same effect. For the single image model (PilotNet), only image shifting is applied; the images are cropped into three categories in a batch: centered image (40%), shifted left image(30%), and shifted right image (30%) by a 200x66 sliding window. For the image sequence model (CNN+LSTM and CNN3D), the images are cropped into five categories in a batch: centered image (40%), shifted left image (15%), shifted right image (15%), rotated left image (15%) and rotated right image (15%) by a 200x66 sliding window. Once the image is cropped, change the steering angle correspondingly.

4.2.4 Model Training

Practical model training is almost identical to the simulation except that the CME models have the speed weight α=1, the steering weight β=2 and one more model (Weiss' CNN+LSTM) for training. Weiss' CNN+LSTM architecture is similar to our CNN+LSTM, but it connects the prediction layer directly after the LSTM layer. Once the model training is finished, we copy the models to the Raspberry Pi model folder for further manipulation.

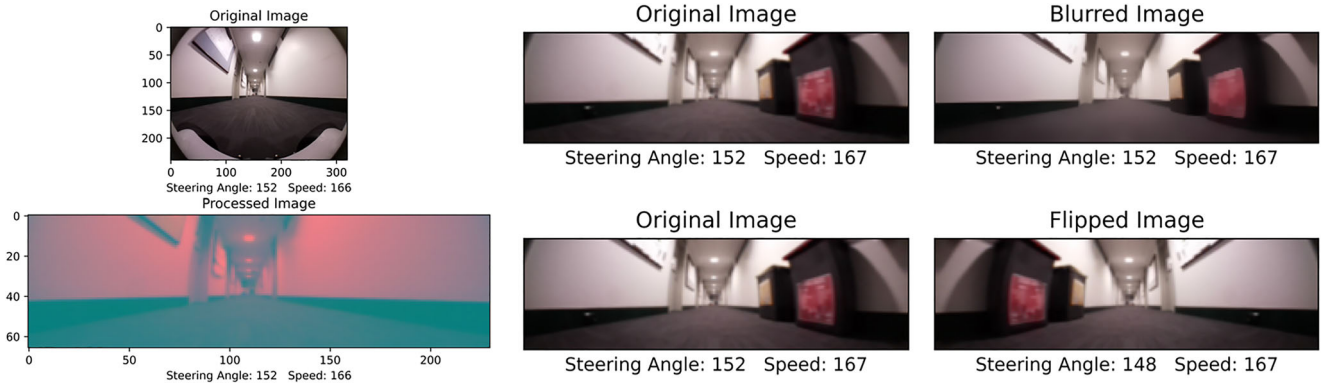


Fig. 13 Practical image preprocessing (left), practical image augmentation (right)

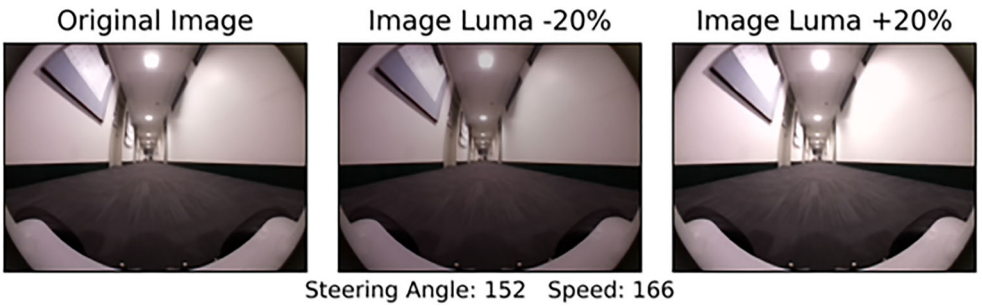


Fig. 14 Practical image augmentation: brightness changing

4.2.5 Validation and Comparison Methods

The practical validation and comparison methods are almost identical to the simulation ones except that the CME experiments are evaluated by NoI instead of Autonomy because NoI is more representative in this case. Because the assumption HRT=5s does not hold in this case, the robot may stop in place or shake back and forth. These cases should be counted as HRT but are difficult to measure, so the NoT here is more representative.

5 Results

5.1 Simulation Results

Table 2 indicates the simulation results.

5.1.1 Open-loop Test

The CNN+LSTM model affords the best performance in the open-loop test, but the metrics utilized do not necessarily correlate with the real driving performance. Because the Lidar drive mode used to collect data is not the optimal solution, the neural network model may surpass the performance of Lidar during the learning process. The more the neural network model mimics the Lidar drive mode, the more it is restricted by the Lidar driving performance.

5.1.2 Closed-loop Test

In the closed-loop test, each model is applied on six laps to calculate the mean lap time, and then each model is subjected to two 10-minute autonomy tests. The results show that the CNN+LSTM model achieves the

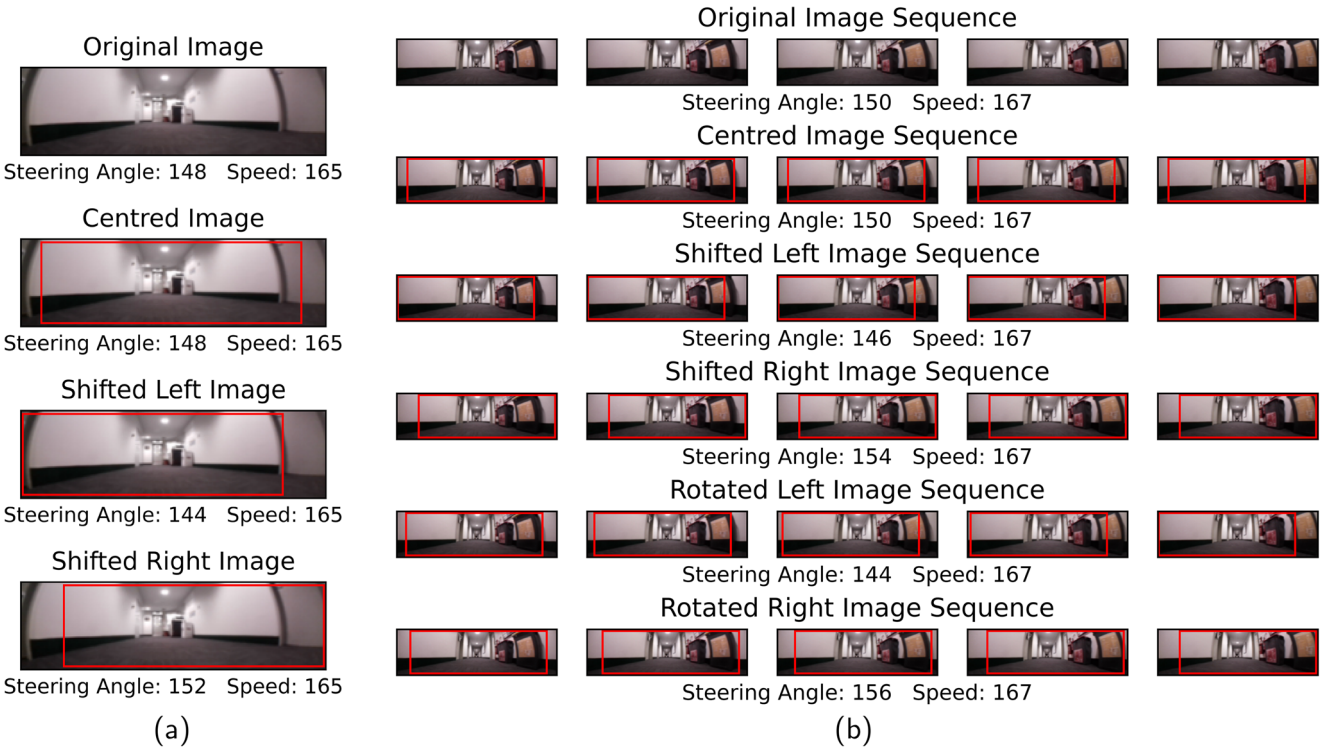


Fig. 15 Additional practical image augmentation: (a) for single image model (b) for image sequence model

Table 2 Simulation models comparison

	Neural Network Models				Reference Model
Test	CNN+LSTM	Weiss' CNN+LSTM	CNN3D	PilotNet	Lidar
Training Loss (MSE)	11.3484	9.7082	34.6803	29.1005	–
Validation Loss (MSE)	4.4548	11.7999	17.8612	8.1348	–
Mean Lap Time (s)	73.45	75.19	73.61	73.94	76.89
Autonomy	100%	98.30%	100%	100%	100%

shortest time per lap, but each model learns from the same speed samples, suggesting that the CNN+LSTM model attains a better solution and travels a shorter route. Overall, all neural network models can drive perfectly in a clockwise and a counterclockwise 10-minute autonomy test without requiring any human intervention except Weiss' CNN+LSTM model. Additionally, all neural network models are much faster than the Lidar model they learned from. Furthermore, the results indicate that the Lidar drive algorithm does not apply optimal route planning, deliberately keeping a distance from each obstacle that significantly limits its driving performance. It is worth noting that amending to this method optimal route planning requires significant effort, while in contrast, neural networks learn optimal route planning automatically.

5.1.3 Prediction

The prediction process of all neural network models is very similar. Here we use the CNN+LSTM model as an example. Although the last image of the sequence is illustrated, the corresponding predictions exploit five consecutive images as input. As presented in Fig. 16, the predicted steering angles are significantly different from the actual ones, which allows the neural network models to complete a lap faster than the Lidar model.

5.1.4 Obstacle Avoidance Test

In this trial, we add new walls as obstacles to test the model's ability to avoid obstacles in the simulation map. All neural network models react to obstacles and change their routes, indicating that the neural network models have successfully learned to recognize walls and plan routes based on them. For illustration purposes, the new walls (obstacles) in Fig. 17 are marked in red. Despite the Lidar model avoiding these obstacles, this model is sometimes too sensitive and forces the robot to stop and turn around. This performance suggests that to avoid these obstacles ideally, several adjustments have to be made to the Lidar algorithm every time it encounters a new environment limiting its effectiveness. Contrary to this, the neural network models adapt to the new environment well without any adjustment.

5.2 EECE Rectangular Loop Results

Table 3 illustrates the EECE rectangular loop results.

5.2.1 Open-loop Test

In the open-loop test, all models attain similar performance. CNN+LSTM model has the best validation loss, while PilotNet has the best training loss.

5.2.2 Closed-loop Test

In the closed-loop test, each model runs three laps clockwise and three laps anti-clockwise to calculate the mean lap time and autonomy. All neural network models can achieve nearly full autonomy except for some failures caused by environmental changes, e.g., broken light bulbs and new obstacles.

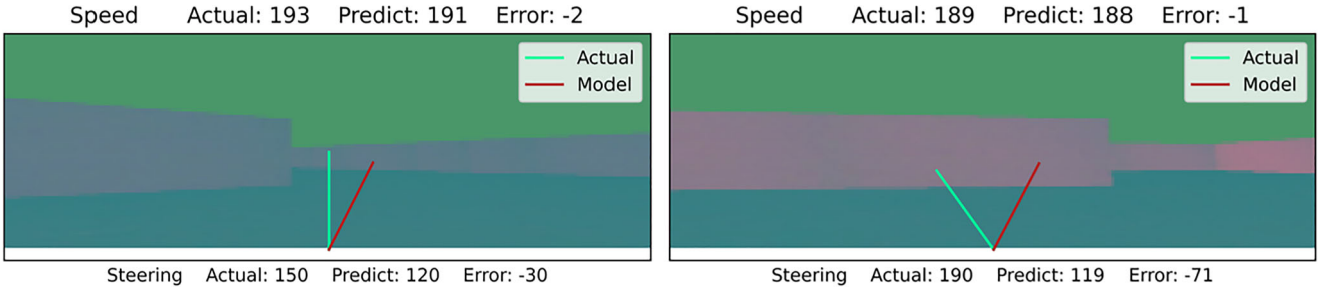


Fig. 16 CNN+LSTM prediction in EyeSim maze map

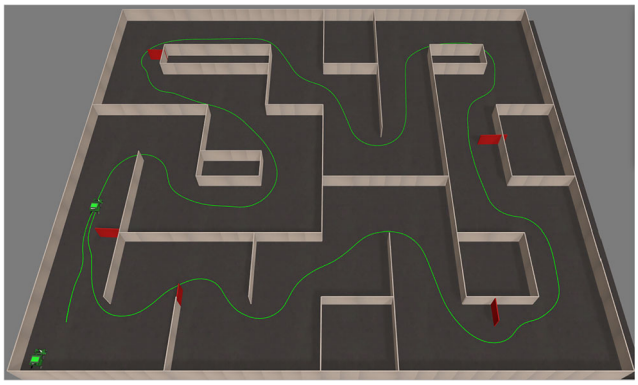


Fig. 17 Simulation models obstacle avoidance test

5.2.3 Prediction

All neural network models can drive autonomously, even with some incorrect predictions. Here we use the CNN+LSTM model as an example. In Fig. 18, the left prediction illustrates that the model drifts from the trained trajectory because of different steering angles. Still, image shifting or rotating give models the ability to recover. The right prediction illustrates that the robot drives with poor lighting conditions, but the models can identify the core feature and make the right decision.

5.3 CME Corridor Results

Tables 4 and 5 shows the CME corridor results.

5.3.1 Open-loop Test

In the open-loop test, the proposed CNN+LSTM model performs best, while Weiss' CNN+LSTM has the highest validation loss, matching the performance of the closed-loop tests.

5.3.2 Closed-loop Test

In the closed-loop test version one (V1), each model runs six laps to calculate the mean lap time and autonomy. Among the neural network models, only the CNN3D and CNN+LSTM models can complete a lap without human interventions, the PilotNet and Weiss' CNN+LSTM models keep the car in the middle between the walls when going straight, but it is jittering at the dead end and cannot turn around. In the closed-loop test version two (V2), the models are tested in a trained corridor with new obstacles and a non-trained corridor. Each model runs six laps in the trained corridor (C1) and three laps in the non-trained corridor (C2) to calculate the mean lap time and NoI. Among the neural network models, none of the models can complete a lap without human intervention. This is due to the fact that during the training phase, the corridor was free, but during the experiment phase there were some new unknown obstacles present. However, the proposed CNN+LSTM and CNN3D models have a lower mean lap time and NoI. The PilotNet model remains the car in the middle of the wall when going straight, but it is jittering at the dead-end and cannot turn around. Weiss' CNN+LSTM performs even worse than PilotNet.

5.3.3 Prediction

Figure 19 illustrates two predictions where the image is labeled with the wrong speed due to Lidar algorithm flaws, but the CNN3D model predicts it correctly. The left picture presents the case where the robot should drive forward, but the image is labeled as backward. The right picture presents the case where the robot should turn left, but the image is labeled as straight. These interferences increase the training difficulty, but models with image sequences manage a greater tolerance.

5.4 Saliency Map

As shown in Fig. 20, all neural network models have successfully highlighted the walls' edges in the saliency map.

6 Conclusion

This research verifies the significance of end-to-end learning for autonomous driving systems and the original PilotNet model improvement by adding LSTM or 3D

Table 3 Rectangular loop models comparison

	Neural Network Models				Reference Model
Test	CNN+LSTM	Weiss' CNN+LSTM	CNN3D	PilotNet	Lidar
Training Loss (MSE)	2.1591	1.0496	3.0968	1.7334	–
Validation Loss (MSE)	0.3375	0.6599	0.5586	0.5226	–
Mean Lap Time (s)	49.15	52.05	52.21	54.16	48.03
Autonomy	96.95%	88.93%	95.76%	93.90%	100%

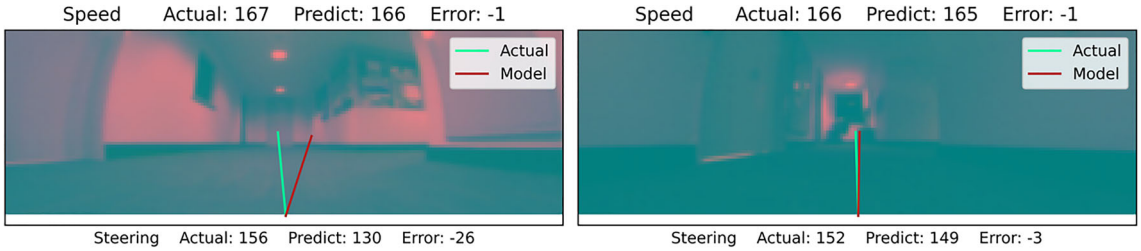


Fig. 18 EECE CNN+LSTM predictions

Table 4 Corridor models comparison in trained environment

	Neural Network Models V1 (47,838 images; α:β=1:1)				Reference Model
Test	CNN+LSTM	Weiss' CNN+LSTM	CNN3D	PilotNet	Lidar
Training Loss (MSE)	7.2576	8.5176	7.1375	14.9284	–
Validation Loss (MSE)	0.9752	2.4830	0.9427	2.2583	–
Mean Lap Time (s)	59.31	110.90	68.98	145.05	50.88
Autonomy	100%	90.21%	100%	87.50%	100%

Table 5 Corridor models comparison in trained environment with new obstacles (C1) and non-trained environment (C2)

	Neural Network Models V2 (75,845 images; α:β=1:2)				Reference Model
Test	CNN+LSTM	Weiss' CNN+LSTM	CNN3D	PilotNet	Lidar
Training Loss (MSE)	11.4965	12.5503	12.6672	14.8427	–
Validation Loss (MSE)	1.5202	5.1174	1.9751	1.5396	–
C1: Mean Lap Time (s)	84.24	112.01	87.61	97.68	50.88
C1: NoI (6 laps)	7	10	10	11	0
C2: Mean Lap Time (s)	190.85	318.20	214.42	270.34	77.26
C2: NoI (3 laps)	11	21	17	19	0

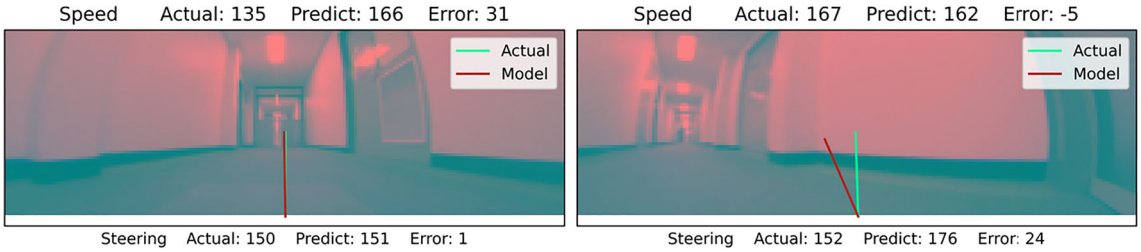


Fig. 19 CME V2 CNN3D predictions

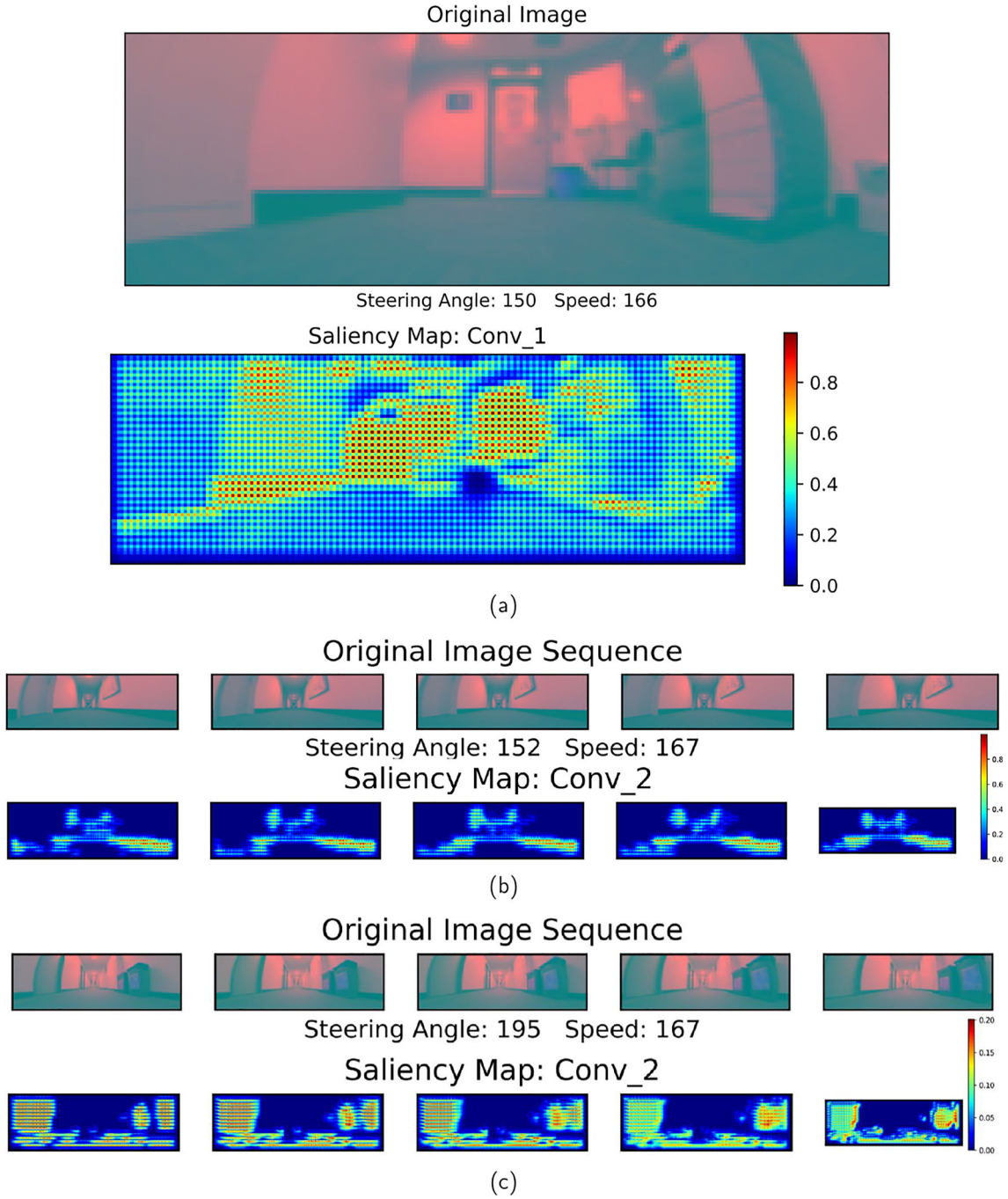


Fig. 20 Saliency map: (a) EECE PilotNet (b) EECE CNN+LSTM (c) CME V2 CNN3D

convolutional layers; however, the architecture must be well-designed. This research also verifies the feasibility of end-to-end driving in indoor environments. The improvements are:

- Complex motions:
  The model can create more complex motions like a three-point turn at a dead-end, in which the robot needs to make a series of moves to turn around at the dead-ends. These actions are in a temporal sequence, but PilotNet does not have access to temporal information, and for it, these actions are like a single input corresponding to multiple outputs. Both CNN+LSTM and CNN3D models solve this problem perfectly by virtue of their processing of temporal information.
- Recovery from failure:

Even if the robot may not succeed in a single turnaround at a dead-end, the CNN+LSTM and CNN3D models repeatedly try until they succeed while PilotNet stops there.

- Better driving performance:
  In all experiments, the proposed CNN+LSTM and CNN3D models have similar performance, the CNN+LSTM model is slightly better, but both models have a better closed-loop performance than PilotNet.

It should be noticed that in this experiment the performance of the CNN+LSTM and CNN3D models is limited by the Raspberry Pi's computing power, as with low FPS the model cannot turn in time if the speed is high. Additionally, the inconsistency of the gap between time steps due to speed decaying increases the difficulty of temporal information extraction.

Future work will include the following tasks:

- Waypoints prediction:
  Predicting waypoints using the Inertial Measurement Unit (IMU) to obtain future locations and optimize the path. Use of IMU-odometry fusion to derive the pose and direction of the attached vehicle.
- Ego-state input:
  Adding different types of input, i.e., past steering angles and speeds. One feature of Lidar drive mode that no model can currently emulate is braking. The robot in this project can simulate the braking effect by reversing its motor. Still, when using images with brakes as a dataset, even a model using image sequences as input cannot tell when to brake. When judging the brakes, the driver considers the immediate view as well as the speed and direction, so adding past speed and steering can help the neural network model learn more complex driving behaviors.
- Parking:
  Training neural network models to park. The action of turning around at a dead-end is similar to parking. It may be possible to learn parking actions for CNN+LSTM and CNN3D models. Still, it is more challenging to learn only with the front camera because the driver usually needs to look at the scene behind the vehicle and judge when parking. Installing an additional camera at the robot's rear, and using images from multiple cameras as input may solve this problem.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10846-022-01801-2.

Acknowledgements The authors would like to thank Anthony Ryan, Omar Anwar, and Michael Mollison for their previous works on the ModCar project. The authors would also like to thank Nyi MyoMaung for his assistance during the ModCar experiments. Finally, the authors would like to express their gratitude to Nvidia for donating two GPUs used for the neural network training.

References

1. Automated vehicles for safety. https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety
2. Petrović, D., Mijailović, R., Pešić, D.: Traffic accidents with autonomous vehicles: type of collisions, manoeuvres and errors of conventional vehicles' drivers. Transp. Res. Procedia **45**, 161–168 (2020). https://doi.org/10.1016/j.trpro.2020.03.003. Transport infrastructure and systems in a changing world, towards a more sustainable, reliable and smarter mobility. TIS Roma 2019 conference proceedings
3. Mugunthan, N., Balaji, S.B., Harini, C., Naresh, V.H., Prasannaa Venkatesh, V.: Comparison review on lidar vs camera in autonomous vehicle. In: International research journal of engineering and technology (IRJET), vol. 07, pp. 4242–4246 (2020). https://www.irjet.net/archives/V7/i8/IRJET-V7I8731.pdf
4. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J., Bottou, L., Weinberger, K.Q. (eds.) Advances in neural information processing systems, vol. 25. Curran Associates, Inc., New York. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (2012)
5. Bojarski, M., Testa, D.D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K.: End to end learning for self-driving cars. arXiv:1604.07316 (2016)
6. Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., Muller, U.: Explaining how a deep neural network trained with end-to-end learning steers a car. https://doi.org/10.48550. arXiv:1704.07911 (2017)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
8. Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. **35**(1), 221–231 (2013). https://doi.org/10.1109/TPAMI.2012.59
9. Pomerleau, D.A.: Alvinn: an autonomous land vehicle in a neural network. In: Touretzky, D. (ed.) Advances in neural information processing systems, vol. 1. Morgan-Kaufmann, Burlington (1988). https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf
10. Maqueda, A.I., Loquercio, A., Gallego, G., García, N., Scaramuzza, D.: Event-based vision meets deep learning on steering prediction for self-driving cars. arXiv:1804.01310 (2018)

11. Codevilla, F., Müller, M., Dosovitskiy, A., López, A.M., Koltun, V.: End-to-end driving via conditional imitation learning. arXiv:1710.02410 (2017)
12. Wang, Q., Chen, L., Tian, B., Tian, W., Li, L., Cao, D.: End-to-end autonomous driving: an angle branched network approach. IEEE Trans. Veh. Technol. **68**(12), 11599–11610 (2019). https://doi.org/10.1109/TVT.2019.2921918
13. Yang, Z., Zhang, Y., Yu, J., Cai, J., Luo, J.: End-to-end multi-modal multi-task vehicle control for self-driving cars with visual perceptions. In: 2018 24th international conference on pattern recognition (ICPR), pp. 2289–2294. https://doi.org/10.1109/ICPR.2018.8546189 (2018)
14. Wang, T., Luo, Y., Liu, J., Chen, R., Li, K.: End-to-end self-driving approach independent of irrelevant roadside objects with auto-encoder. IEEE Trans. Intell. Transp. Syst. **23**(1), 641–650 (2022). https://doi.org/10.1109/TITS.2020.3018473
15. Drews, P., Williams, G., Goldfain, B., Theodorou, E.A., Rehg, J.M.: Vision-based high-speed driving with a deep dynamic observer. IEEE Robot. Autom. Lett. **4**(2), 1564–1571 (2019). https://doi.org/10.1109/LRA.2019.2896449
16. Chi, L., Mu, Y.: Deep steering: learning end-to-end driving model from spatial and temporal visual cues. arXiv:1708.03798 (2017)
17. Hou, Y., Hornauer, S., Zipser, K.: Fast recurrent fully convolutional networks for direct perception in autonomous driving. arXiv:1711.06459 (2017)
18. Okamoto, K., Itti, L., Tsiotras, P.: Vision-based autonomous path following using a human driver control model with reliable input-feature value estimation. IEEE Trans. Intell. Veh. **4**(3), 497–506 (2019). https://doi.org/10.1109/TIV.2019.2919476
19. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: Deepdriving: learning affordance for direct perception in autonomous driving. In: 2015 IEEE international conference on computer vision (ICCV), pp. 2722–2730. https://doi.org/10.1109/ICCV.2015.312 (2015)
20. Chen, S., Zhang, S., Shang, J., Chen, B., Zheng, N.: Brain-inspired cognitive model with attention for self-driving cars. IEEE Trans. Cogn. Dev. Syst. **11**(1), 13–25 (2019). https://doi.org/10.1109/TCDS.2017.2717451
21. Weiss, T., Behl, M.: Deepracing: parameterized trajectories for autonomous racing. arXiv:2005.05178 (2020)
22. Xu, H., Gao, Y., Yu, F., Darrell, T.: End-to-end learning of driving models from large-scale video datasets. arXiv:1612.01079 (2016)
23. Chen, Y., Praveen, P., Priyantha, M., Muelling, K., Dolan, J.: Learning on-road visual control for self-driving vehicles with auxiliary tasks. In: 2019 IEEE winter conference on applications of computer vision (WACV), pp. 331–338. https://doi.org/10.1109/WACV.2019.00041 (2019)
24. Pierre, J.M.: End-to-end deep learning for robotic following. In: Proceedings of the 2018 2nd international conference on mechatronics systems and control engineering. ICMSCE 2018, pp. 77–85. Association for Computing Machinery, New York. https://doi.org/10.1145/3185066.3185084 (2018)
25. Jeong, S.-G., Kim, J., Kim, S., Min, J.: End-to-end learning of image based lane-change decision. https://doi.org/10.48550.arXiv:1706.08211 (2017)
26. Jaritz, M., de Charette, R., Toromanoff, M., Perot, E., Nashashibi, F.: End-to-end race driving with deep reinforcement learning. arXiv:1807.02371 (2018)
27. Liang, X., Wang, T., Yang, L., Xing, E.P.: CIRL: controllable imitative reinforcement learning for vision-based self-driving. arXiv:1807.03776 (2018)
28. Sallab, A., Abdou, M., Perot, E., Yogamani, S.: Deep reinforcement learning framework for autonomous driving. Electron. Imaging **2017**, 70–76 (2017). https://doi.org/10.2352/ISSN.2470-1173.2017.19.AVM-023
29. Nie, J., Yan, J., Yin, H., Ren, L., Meng, Q.: A multimodality fusion deep neural network and safety test strategy for intelligent vehicles. IEEE Trans. Intell. Veh. **6**(2), 310–322 (2021). https://doi.org/10.1109/TIV.2020.3027319
30. Sobh, I., Amin, L., Abdelkarim, S., Elmadawy, K., Saeed, M., Abdeltawab, O., Gamal, M.E., Sallab, A.E.: End-to-end multi-modal sensors fusion system for urban automated driving (2018)
31. Bräunl, T. Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems, 3rd edn. Springer Berlin, Heidelberg (2006)
32. Clevert, D., Unterthiner, T., Hochreiter, S.: Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). https://doi.org/10.48550/arXiv.1511.07289 (2015)
33. Géron, A.: Hands-on Machine Learning with Scikit-Learn and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, Sebastopol (2017)
34. Bräunl, T.: EyeSim VR. https://robotics.ee.uwa.edu.au/eyesim/ (2022)
35. Gedraite, E.S., Hadad, M.: Investigation on the effect of a gaussian blur in image filtering and segmentation. In: Proceedings ELMAR-2011, pp. 393–396 (2011)
36. Wong, S.C., Gatt, A., Stamatescu, V., McDonnell, M.D.: Understanding data augmentation for classification: when to warp? In: 2016 International conference on digital image computing: techniques and applications (DICTA), pp. 1–6. https://doi.org/10.1109/DICTA.2016.7797091 (2016)
37. Ungurean, D.: Deeprcar: an Autonomous Car Model. Czech Technical University, Prague (2018)
38. Yang, S., Wang, W., Liu, C., Deng, K., Hedrick, J.K.: Feature analysis and selection for training an end-to-end autonomous vehicle controller using the deep learning approach. arXiv:1703.09744 (2017)
39. Bojarski, M., Chen, C., Daw, J., Degirmenci, A., Deri, J., Firner, B., Flepp, B., Gogri, S., Hong, J., Jackel, L.D., Jia, Z., Lee, B.J., Liu, B., Liu, F., Muller, U., Payne, S., Prasad, N.K.N., Provodin, A., Roach, J., Rvachov, T., Tadimeti, N., van Engelen, J.E., Wen H., Yang, E., Yang, Z.: The NVIDIA pilotnet experiments. arXiv:2010.08776 (2020)

Zhihui Lai received the B.E. (Hons.) degree from The University of Western Australia, Perth, Australia in 2021 where he is currently pursuing a Ph.D. degree with a focus on deep learning methods for autonomous driving. His research interests include end-to-end self-driving based on deep neural networks, computer vision, object detection and tracking.

Thomas Bräunl is a Professor in the School of Engineering at The University of Western Australia, Perth, where he directs the Robotics & Automation Lab as well as the Renewable Energy Vehicle Project (REV). He has developed numerous robotics systems, including the EyeBot robot family and the EyeSim simulation system. On the automotive side, he has done research on electric drive and charging systems, and is developing AI solutions for autonomous driving. Professor Bräunl worked on Driver-Assistance Systems with Mercedes-Benz Stuttgart and on Electric Vehicle Charging Systems with BMW Munich and BMW Mountain View. He holds a Diploma from the University of Kaiserslautern, an M.S. degree from the University of Southern California, Los Angeles, and a Ph.D. and Habilitation from the University of Stuttgart.

Chapter 3

Vision and AI-based Navigation for Autonomous Shuttle Bus Systems

Preamble: *Having shown that temporal memory improves robustness on ModCar, we now scale up to shuttle buses (nUWAy1, nUWAy2) and deploy learned policies alongside a classical vision baseline. We also introduce a CARLA-based digital twin (UWA campus, Amberton Beach) to bridge simulation and road testing. The results reveal the limits of classical lane tracking and the advantages of end-to-end learning in closed-loop runs, motivating a deeper question addressed next: Which simulator best supports iterative autonomy development? Chapter 4 answers this through a head-to-head study of CARLA and AWSIM. This chapter primarily addresses* ***RQ1*** *and provides baseline evidence of the benefits and shortcomings of feed-forward E2E models.*

Publication: *Quirke-Brown, K., Lai, Z., Kong, X., Tan, T., Du, Y., Bräunl, T. Developing an Autonomous Shuttle Service. Australasian Transport Research Forum 2023 Proceedings, Perth, 29 Nov-1 Dec 2023 (published)*

Vision and AI-based Navigation for Autonomous Shuttle Bus Systems

ABSTRACT

This is an extended version of a portion of a published conference paper, entitled "Developing An Autonomous Shuttle Service." This chapter presents contributions to the development of an autonomous shuttle system, focusing on vision-based and AI-based navigation, simulation systems, and public road testing. Key components of the software stack and experimental setup were implemented to enable the shuttle to perceive and navigate complex environments using deep learning and computer vision. The work encompasses camera configuration, simulation, model design, and testing in both campus and suburban scenarios.

I. CAMERAS

The bus comes fitted with two Flir Point Grey cameras which produce good quality grayscale images. At this stage of development, grayscale images are sufficient for the Neural Network learning process and obstacle tracking features. Each camera is configurable to use different size images and publishing rates to avoid flooding networks and reducing processing times on images.

II. VISION-BASED NAVIGATION

Vision-based navigation can be implemented using traditional Engineering methods or by using more recent deep learning methods. The traditional Engineering method is based on programmed computer vision operations, e.g., software packages like OpenCV [1], to detect lane markings and curbs and then plan a driving path according to that.

Our current algorithm is applying Canny edge detection [2] and a subsequent Hough line detection [3] algorithm to identify the lane's left and right boundaries, to determine the shuttle's steering and throttle commands (Fig. 1). However, this approach only works on straight and curved road segments and cannot yet handle intersections or roundabouts. These need to be handled by the safety operator for the time being.

III. AI-BASED NAVIGATION

A deep-learning method is based on an artificial neural network approach, for which we use the TensorFlow [4] software package to train a neural network with image input data. We used a modified version of PilotNet [5] (Fig. 3) that takes the front camera image as input and outputs the corresponding steering angle and throttle control commands (Fig. 2). This method can handle intersections with traffic lights and is implemented in real shuttle buses.

IV. SHUTTLE SIMULATION SYSTEM

In addition to the real-world vehicle, the REV team has set up a simulation environment for an autonomous shuttle bus using Carla [6]. Initially developed as a hardware-in-the-loop (HIL) system [7], it was later found that the HIL component did not enhance the system's performance or accuracy, so it was converted into a pure software simulation. In the simulation environment, a digital version of the entire UWA university campus and Amberton beach was developed, along with a model of the autonomous shuttle bus (Fig. 4 and Fig. 5). The simulator has two manual control inputs: A steering wheel with accelerator and brake pedals and a keyboard input. With this system, users can easily set up and conduct driving experiments.

To operate the simulator, we incorporated five distinct modes:

1) Manual mode, where the vehicle is controlled manually with joystick or keyboard.
2) Lidar autonomous mode, where the Lidar-based navigation stack drives the vehicle.
3) Computer vision autonomous mode, which uses OpenCV-based vision algorithms to operate the shuttle.
4) Neural network autonomous mode, which employs a deep learning neural network model for control.

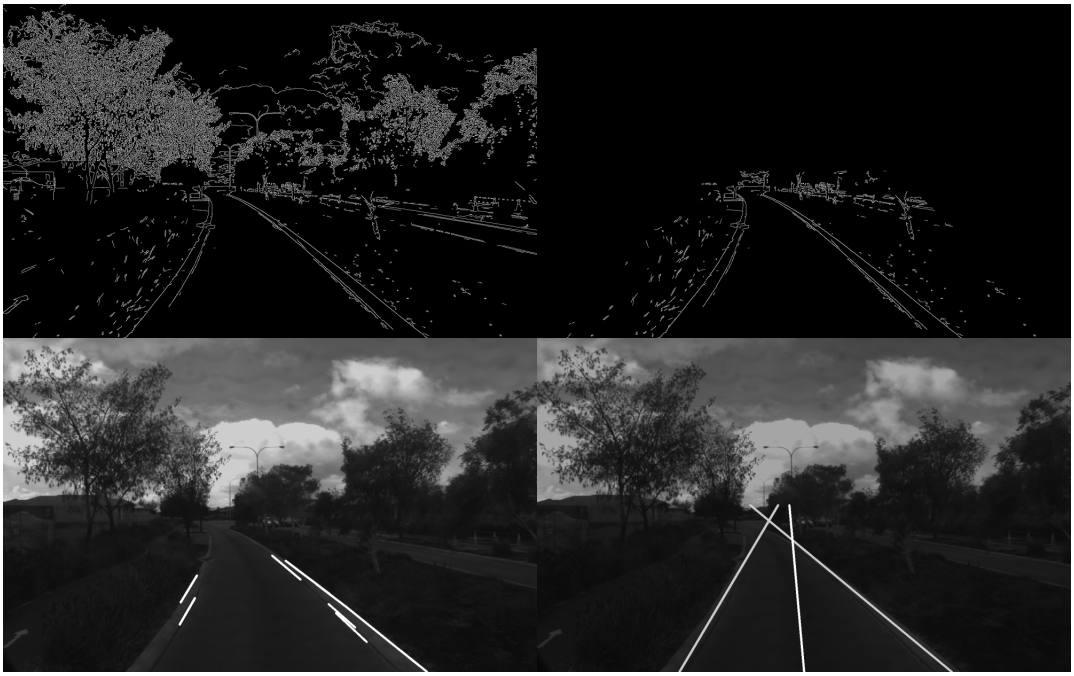


Fig. 1: OpenCV workflow showing: top left - edges, top right - region of interest, bottom left - superimposed lane markings, bottom right - smoothed lane borders and heading

5) Mirror mode, which utilizes the GPS coordinates of the real vehicle in the simulation.

V. Results and Evaluation

The autonomous shuttle bus has been tested with a variety of control methods. Driving data provides valuable insights into where current implementations are failing and how they might be improved. Road training in Amberton Beach began earlier this year with data collection from manual driving. This data provided the basis for extending a campus-based autonomous vehicle to a road-based solution. The following sections summaries the data collected so far, with a comparison of methods as well as the current limitations of the technology.

A. Simulated Driving

The simulation system provides a safe space for testing software before it is deployed in the real world, this is particularly important for AI solutions where the control output is not always known. The simulated shuttle bus has currently been set up to use either a neural network or computer vision for testing in various campus-based situations including turning corners and decision making at intersections. The results of the simulator are shown in Table I.

Limited testing has been done with GPS and SLAM [8] based solutions as they require additional sensor setup as well as random error additions. Looking at the results, it is clear that neural networks provide a sound solution for autonomous driving.

B. UWA Campus Driving

Currently the shuttle bus at the UWA campus has three different methods of navigation, which are being tested and compared to make future improvements. Table II comparing the results from deployment using pure GPS. GPS+RTK [9, 10], SLAM and neural networks. Multiple trips were taken following the main path from the Ocean Marine Centre to Business School and back as this is one of the major routes planned for future services (see Fig. 6).

The time spent in autonomous and manual mode is given as a percentage as each journey's average time can vary depending on the type of intervention and wait time. The neural network and SLAM implementations currently have the most stable runs, typically running in autonomous mode for much

Fig. 2: NN workflow showing steering angle and throttle control commands in the test dataset. (gray line on the left represents actual commands, white line on the right represents neural network commands, line length is controlled by throttle, line angle is controlled by steering)

TABLE I: Simulation results

	Neural Network	Computer Vision
% Time Autonomous Mode (AM)	98.73%	64.96%
% Time Manual mode (MM)	1.27%	35.04%
Number of interventions	1	17
Number of times out of bounds	5	22

TABLE II: Real driving results

	GPS	GPS+RTK	SLAM	NN
% Time AM	41.78%	56.19%	83.33%	73.32%
% Time MM	58.22%	43.81%	16.67%	26.68%
Number of interventions	5	9.5	30	18

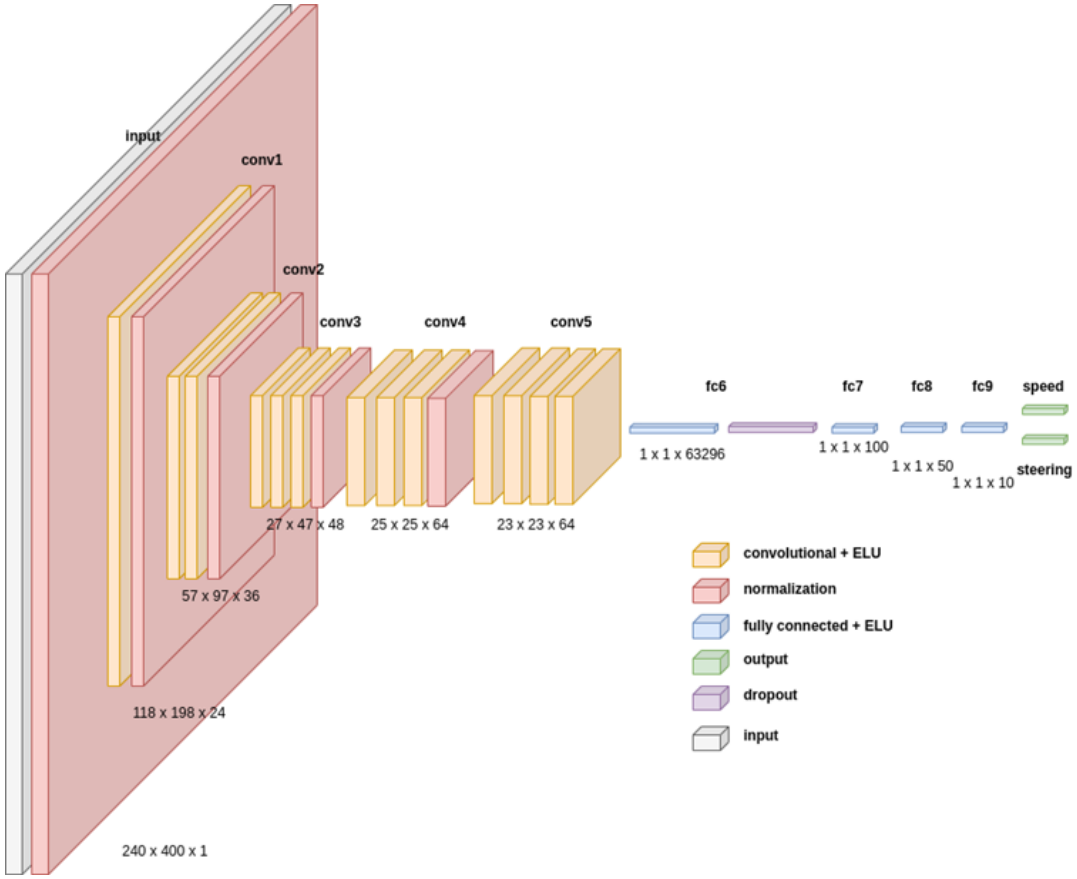


Fig. 3: End-to-end NN architecture

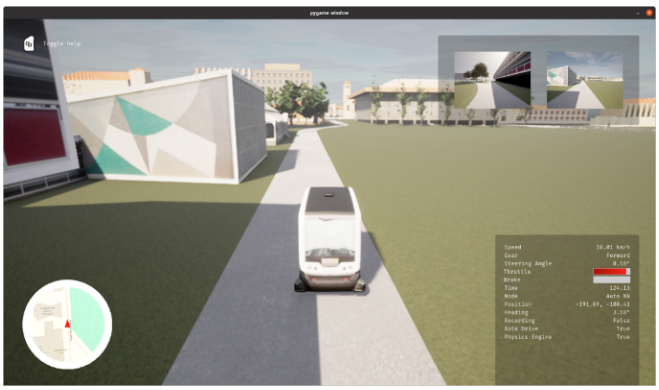


Fig. 4: Shuttle bus simulations on campus

Fig. 5: Shuttle bus simulations at Amberton Beach

of the journey. The neural network model is more reliable, however with a smaller number of interventions and is currently limited by its training around obstacle avoidance. The SLAM implementation has a high autonomous rate but with the current implementation has a higher failure rate. This is due to the SLAM implementation delocalizing from the map and moving towards obstacles or locations where the vehicle is not allowed to run. Fig. 7 shows a shorter journey between law and guild where the shuttle bus has performed well yet due to localization issue it will deviate from the planned path. At its largest the bus deviated approximately 1.5m from its planned path, which is the gray path shown, however the system was able to recover after the corner due to better map localization at this

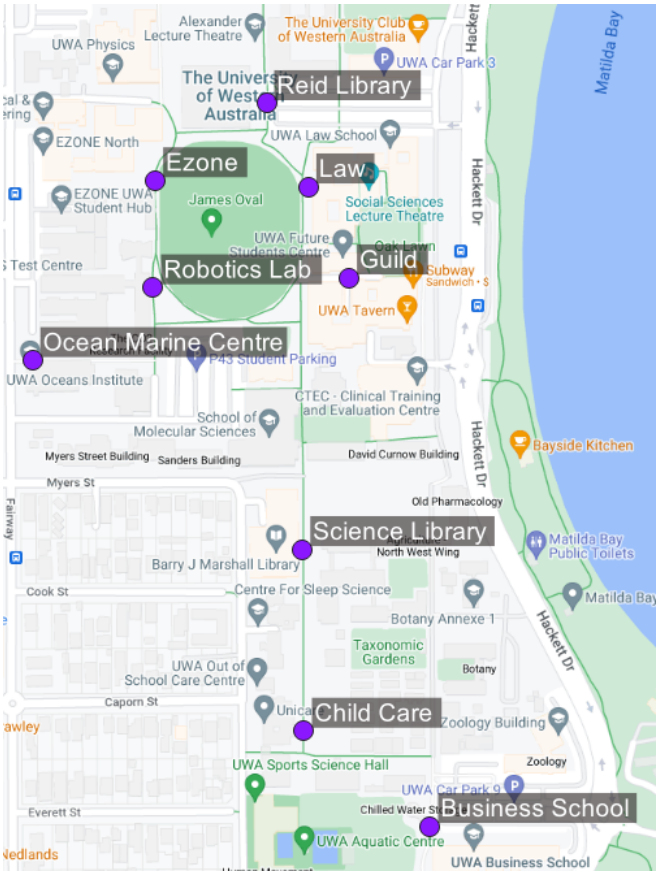


Fig. 6: Map showing major stops on the UWA campus

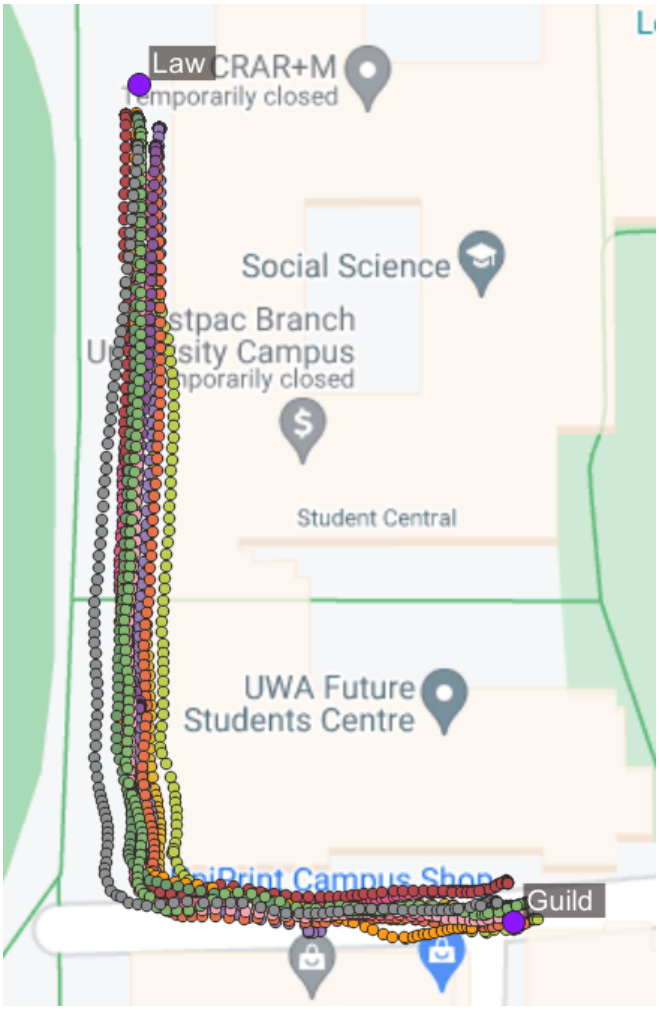


Fig. 7: Map showing path deviation using SLAM

point.

Comparatively GPS and GPS+RTK implementation has far fewer interventions, but this could also be attributed to the longer manual control where the operator is waiting for a clear path. The GPS model also struggles to stay on the planned path due to the error for the GPS coordinates. GPS+RTK shows an improved performance when compared to pure GPS due to the much higher coordinate accuracy allowing it to follow the path more rigidly. There are still, however, several locations along the path where the vehicle struggles to drive using GPS+RTK, such as UWA's Science Library, where buildings cause signal interference and the vehicle starts to veer away from the path, requiring intervention. The SLAM implementation has proven to overcome these issues when it can properly localize, which requires larger stationary structures nearby.

C. Amberton Beach Driving

During the initial manual driving stage, data has been recorded for path accuracy, image data for neural networks and the development of safety systems. At this stage only the neural network mode has been tested with the results shown in Table III.

TABLE III: Neural network driving accuracy

	Neural Network
Percentage of Time Autonomous Mode	43.96%
Percentage of Time Manual mode	56.04%
Number of interventions	11

The model developed for Amberton Beach is still in its infancy and requires more data for improvement. The system has been designed to drive up to intersections where on-board safety operators will determine when it is safe to proceed through. It is a

development goal to automate this, but more sensor data is required to do this safely. A GPS+RTK drive system has also been developed and will be combined with the neural network.

VI. Conclusions

My work significantly contributed to the software and experimental development of UWA's autonomous shuttle project. I focused on perception, end-to-end AI control, simulation, and public-road data infrastructure. These contributions form a foundation for future improvements toward safe, on-demand autonomous transport.

References

[1] OpenCV. [Online]. Available: https://opencv.org/

[2] J. Canny, "A computational approach to edge detection," vol. PAMI-8, no. 6, pp. 679–698. [Online]. Available: https://ieeexplore.ieee.org/document/4767851

[3] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," vol. 15, no. 1, pp. 11–15. [Online]. Available: https://dl.acm.org/doi/10.1145/361237.361242

[4] TensorFlow. [Online]. Available: https://www.tensorflow.org/

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

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

[7] C. Brogle, C. Zhang, K. Lim, and T. Bräunl, "Hardware-in-the-loop autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 375–384, 2019.

[8] S. Macenski and I. Jambrecic, "Slam toolbox: Slam for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, 2021.

[9] M. E. Cannon, G. Lachapelle, P. Alves, L. P. S. Fortes, B. Townsend, and N. Weston, "Gps rtk positioning using a regional reference network: theory and results," pp. 1842–1848, 2001.

[10] SubCarrier Systems Corp., "Rtk2go," http://rtk2go.com/, 2021, accessed: 2023-04-01.

Chapter 4

A Comprehensive Comparative Analysis of Carla and Awsim: Open-Source Autonomous Driving Simulators

Preamble: *The shuttle experiments relied heavily on simulation for risk-reduced iteration. This chapter systematically compares CARLA (Unreal) and AWSIM (Unity) using the same Amberton Beach digital twin to quantify trade-offs in sensor realism, real-time performance, ROS 2 integration, and extensibility. By clarifying when to prefer photorealistic APIs (CARLA) versus ROS-native, LiDAR-centric pipelines (AWSIM), this chapter provides a principled basis for picking tools that match the target autonomy stack. This chapter primarily addresses* ***RQ3****, establishing a structured methodology for simulator selection and highlighting how simulation choices influence E2E model development and validation.*

Publication: *Lai, Z., Le, L., Silva, V., Bräunl, T. A Comprehensive Comparative Analysis of Carla and Awsim: Open-Source Autonomous Driving Simulators. Robotics and Autonomous Systems (under review, submitted on 26 May 2025)*

A Comparative Analysis of Open-Source Autonomous Driving Simulators—CARLA and AWSIM

Zhihui Lai[c,*], Lee Le[c], Vihanga Silva[d], Thomas Bräunl[c]

[c] *The University of Western Australia, 35 Stirling Hwy, Crawley, 6009, WA, Australia*
[d] *Western Power, 363 Wellington St, Perth, 6000, WA, Australia*

1. Abstract

The number of open-source autonomous driving simulators has been increasing in recent years, and many papers have surveyed the most popular options. However, there is a lack of in-depth and comprehensive comparisons among the most advanced open-source simulators. This article provides a detailed quantitative and qualitative analysis of two of the most promising simulators for end-to-end testing: CARLA and AWSIM. The analysis focuses on various metrics, including the efficiency of simulation, the realism of physics simulation, scenario creation and testing, user-friendliness, and extendability. Based on our findings, we conclude that CARLA remains the best open-source simulator for end-to-end testing, while AWSIM shows advanced performance in specific areas, such as LiDAR-based algorithms and Autoware development.

2. Abstract

The number of open-source autonomous driving simulators has grown rapidly in recent years, yet comprehensive and quantitative comparisons between state-of-the-art platforms remain limited. This paper presents a detailed qualitative and quantitative comparison of two representative open-

*Corresponding author.
Email addresses: zhihui.lai@research.uwa.edu.au (Zhihui Lai), 23011365@student.uwa.edu.au (Lee Le), vihanga.silva@outlook.com (Vihanga Silva), thomas.braunl@uwa.edu.au (Thomas Bräunl)

source autonomous driving simulators: UE4-based CARLA and Unity-based AWSIM, with a primary focus on end-to-end autonomous driving research.

The evaluation covers simulation efficiency, physical realism, scenario creation and testing, user-friendliness, and extendability. A weighted scoring framework is adopted to reflect practical research priorities. Under this framework, CARLA achieves a weighted total score of **216.995 out of 251**, while AWSIM scores **145.031 out of 251**. It is important to note that safety-related factors are intentionally assigned a relatively low weight, as this study emphasizes simulator effectiveness for algorithm development rather than safety validation. This design choice represents a limitation of the study and should be considered when interpreting the results.

The results indicate that CARLA remains the most comprehensive open-source simulator for systematic end-to-end testing, benefiting from mature APIs and rich scenario customization. AWSIM, in contrast, demonstrates advantages in rapid iteration and LiDAR simulation fidelity, making it particularly suitable for LiDAR-based learning and integration with modern autonomy stacks such as Autoware.

3. Introduction

Autonomous driving simulations are crucial for supplementing and accelerating real-world testing, which can be difficult, dangerous, or even impossible in some cases [1, 2]. Over the past few years, the domain of simulators has witnessed a surge in the development of both commercial and open-source simulators. Interestingly, the number of open-source simulators has been on the rise [3]. The most popular commercial simulators currently include Waymo SimulationCity [4]; Cognata [5]; AVSimulation (SCANeR) [6]; Simcenter Prescan [7]; CarSim [8]; IPG CarMaker [9]; MATLAB RoadRunner Scenario [10]; and Nvidia Drive Sim [11]. In contrast, the open-source simulators that have gained popularity are Gazebo [12], Webots [13], AirSim [14], CARLA [15], AWSIM [16], and LGSVL [17]. While these simulators help researchers study autonomous driving tasks, including perception, localization, planning, vehicle control, and 3D scene reconstruction, there is a lack of in-depth and comprehensive comparison among the most advanced open-source simulators.

This work fills the gap by performing a comprehensive comparative analysis of two simulators, CARLA and AWSIM, where the former is prominent in the domain of autonomous driving, and the latter is new but has a lot of

potential. This analysis will involve performance testing (quantitative) and functionality comparison (quantitative and qualitative), offering insights for researchers and developers selecting simulators for autonomous driving applications.

When assessing the effectiveness of an autonomous driving simulator, it is essential to consider its efficiency, involving the degree of simplification and computational cost required to run the simulator and the time it takes to complete each simulation step. Simulators that need less computational power and memory are more accessible and cost-effective, but may sacrifice realism and accuracy. Another critical consideration is the level of realism and fidelity they provide in replicating real-world driving scenarios, including factors such as the accuracy of the physics engine, the realistic behavior of other vehicles and pedestrians, and the authenticity of the sensor, environment, and weather conditions. A high-fidelity simulator can shrink the "Reality Gap" [18, 19], so the trade-off between simulator efficiency and fidelity is important [20]. Support for the scenario creation and testing is also crucial. Simulators with rich scenario libraries, construction tools, and well-designed traffic management systems are preferred. Besides, it is necessary to take user-friendliness into account. UI functionality, setup & learning curve, community and support, and affordability & platform support are legit indicators of the difficulty involved in mastering a simulator. Finally, the extendability of a simulator is critical to reflect its bridging ability to AI algorithms, ROS [21], and other simulators. All of these criteria play a crucial role in determining the overall effectiveness of the simulation, which is essential for developing and testing autonomous vehicle systems.

The following sections are organized as follows: Section 2 reviews the existing literature on autonomous driving simulators. Section 3 identifies and describes the criteria for the end-to-end testing of autonomous driving simulators. Section 4 presents an analysis of the comparative results of the simulators. Section 5 introduces the conclusion and future works.

4. Related Works

Real-world self-driving experiments are expensive, time-consuming, and unsafe. It also requires a tremendous effort to achieve regulatory approvals with road agencies in most countries. Kalra et al. stressed that automated driving systems (ADS) require hundreds of millions of miles of driving to reach the same level of safety as human drivers [22]. However, this is im-

possible with real-world testing alone. Therefore, simulators are pivotal for safely speeding up real-world testing and reducing hardware and time costs. Open-source simulators, including Gazebo, Webots, CARLA, AWSIM, and LGSVL, are the preferred choices in academia due to their active communities and low barriers to entry. The advantages and disadvantages of these simulators have been extensively discussed and analyzed in recent publications.

Fadaie surveyed several simulators from an industrial perspective, including companies such as Toyota, Waymo (Google), Uber, NVIDIA, Microsoft, HERE, Mapper.ai, BMW, Ford, and GM; he also summarized the simulator characteristics comprising data generation, scenario controllability, software extendability, component-level interfacing, scenario parameterization, system observability, usability and sim management, software reusability, throughput (iteration speed), experiment replay, and behavior visualization [23]. Violate compared 18 simulators with filters, including team support, documentation, fidelity, ease of customization, extendability (especially with Apollo [24] and Autoware [25]), and HD maps support [26].

Kaur et al. surveyed six simulators—CARLA, LGSVL, CarSim, Gazebo, Prescan, and Matlab/Simulink—to explore their functionalities and performance on perception, multi-view geometry, path planning, vehicle control, map and scenario creation, 2D/3D ground truth, team support, API flexibility, compatibility of different OS, scalability via a server multi-client architecture, and open-source [27].

Meanwhile, Zhou et al. conducted a similar study, but with the main difference of replacing Gazebo with AirSim and using simpler criteria focusing on HD map and scenario generation, support for different sensors, non-player character (NPC) controlling, and vehicle dynamic model [28].

Luttkus et al. identified key requirements for simulators tailored to autonomous micromobility vehicles, emphasizing the need for accurate sensor modeling, realistic pedestrian behavior, customizability, comprehensive scenario and vehicle libraries, scalability, and user-friendliness. Their study compared three simulators—CARLA, AWSIM, and Arena-Rosnav-3D—highlighting their strengths and limitations in meeting these criteria [29].

Li et al. conducted a thorough survey of 46 simulators, classifying them into five primary categories: traffic flow, sensory data, driving policy, vehicle dynamics, and comprehensive simulators. This survey serves as a helpful guide for researchers to select the most appropriate simulator based on their specific requirements [3].

Although useful, these surveys only collected surface-level information and did not compare the simulators through experiments or analyze their documents in detail.

There is a need for an in-depth comparison between these simulators using the same experiments in the same environments. However, building the same environments and experiments in all the simulators is time-consuming. It is more convenient and reasonable to compare the most popular ones. Since the surveys from [26], [27], [28], and [3] all indicated that CARLA and LGSVL are the two most suitable simulators for end-to-end testing, it is ideal to do experiments within these two simulators. However, LGSVL has been suspended since 2022, prompting the Autoware team -TIER IV to develop a new simulator known as AWSIM, which we employ in this paper. Other open-source simulators, such as AirSim, have not received updates for several years, while options such as Gazebo and Webots lack sufficient realism and ADS support. AWSIM is an open-source, Unity-based simulator designed for Autoware and extensible for other tasks. It doesn't have a mature ecology as CARLA; however, it has several advantages, such as a smooth learning curve, low computational cost, and time acceleration, that are worth attracting more researchers to join and develop this community.

This paper aims to conduct an in-depth, comprehensive comparative analysis on the same map in Unreal Engine 4 (UE4)-based CARLA and Unity-based AWSIM to compare their simulator performance for autonomous driving. Hence, it compares the game engines behind the two simulators, UE4 and Unity.

5. Method and Experiment Setup

This section identifies a set of criteria that can serve as a metric to determine which simulators best suit the task at hand. The criteria for defining a superior autonomous driving simulator can be categorized into two primary domains: quantitative and qualitative assessments. Quantitative metrics, including simulation efficiency, scenario creation and testing, and sensor simulation realism, are derived from objective methodologies, whereas qualitative metrics, such as UI functionality, the ease of installation, and the level of community support, depend on subjective judgment.

As previously noted, the two simulations under comparison are high-end models that utilize advanced, high-quality engines to accurately replicate real-world scenarios. Consequently, this analysis is primarily centered on

evaluating the performance and complexity necessary to emulate real-life conditions. The focus will predominantly be on road scenarios rather than corner cases, such as environments with dense pedestrian activity or jungle terrains. Furthermore, safety factors are deemed low priority in this context, since it is assumed that all environmental actions and behaviors adhere to standard safety protocols, with no entities violating safety regulations—this decision aims to highlight the intrinsic performance capabilities of the simulations. The evaluation of the sections concerning simulation efficiency, vehicle dynamics, and sensor implementation will employ distinct weight distribution methodologies. Conversely, the remaining sections will solely assess their relative importance to the outlined comparison objectives, categorized from low to high priority as detailed in Table 1.

Level	Possible Weight
Low	1-3
Medium	4-6
High	7-10

Table 1: Level of Weight Distribution

5.1. Simulation platform setup

Since 2020, The Renewable Energy Vehicle (REV) Project at The University of Western Australia (UWA) has purchased two second-hand EasyMile shuttle buses with drive-by-wire and sensor systems installed without any software. This project aims to build a complete software stack and improve current algorithms for autonomous driving in a closed environment and on public roads. We use two test locations, one at the UWA Crawley campus and one in the northern Perth suburb of Eglinton, the new Amberton Beach residential development, connecting the Stockland Sales Center to Amberton Beach and then returning. The total length of the route is approximately 4.1 kilometers, with a maximum allowable speed of 50 kilometers per hour on the road. We built digital versions of the UWA campus and Amberton Beach in RoadRunner and then exported them to CARLA and AWSIM to test autonomous driving algorithms. As an experimental scenario, the Amberton Beach map in this paper compares CARLA and AWSIM's performance. Fig. 1 is the Amberton Beach map with a regular path. Hardware Setup: All experiments are conducted on an OMEN laptop with three Ubuntu systems

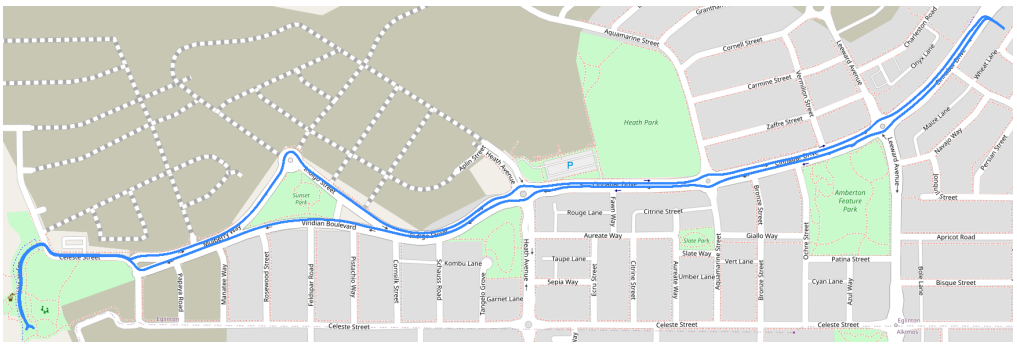


Figure 1: Amberton Beach map with a regular path (scale 100m)

(18, 20, 22). The CPU is an Intel i7-11800H, the RAM is 32 GB, and the GPU is an NVIDIA GeForce RTX 3070.

RoadRunner Setup: Using the Eglinton aerial image (see Fig.2 A) as a reference, we carefully constructed the road network, treating it as the primary focus of the map. To accelerate the overall development, vegetation was placed randomly, and similar template models were employed for residential buildings due to the absence of actual 3D assets. The resulting RoadRunner Eglinton map is shown in Fig.2 B, with a close-up view presented in Fig. 3 B. During the experiment, we used RoadRunner R2024b, the latest version at the time, although it exhibited no significant differences compared to earlier releases.

CARLA Setup: We use the CARLA 0.9.12 standalone package on Ubuntu 20 and the CARLA 0.9.12 build from source on Ubuntu 18. CARLA version 0.9.12 was chosen for comparison purposes because our scripts are designed for this version. Additionally, we aim to utilize the CARLA version that officially supports ROS. The most recent version supporting the official ROS bridge is CARLA 0.9.13, making CARLA 0.9.12 just one minor version older with no substantial differences. The graphics quality is configured to Epic, which is the default setting and offers the highest level of detail. Fig. 2 C is the CARLA Eglinton map, while Fig. 3 C is the CARLA Eglinton close-up.

AWSIM Setup: We use the AWSIM 1.3.1 in Unity editor 2021.1.7f1 and the AWSIM 1.3.1 executable on Ubuntu 22. AWSIM 1.3.1 is the most recent version available, while Unity editor 2021.1.7f1 is the only editor version AWSIM supports. The graphics quality—High Definition Render Pipeline (HDRP)—is set to High Fidelity, providing the most intricate and detailed visuals. Fig. 2 D is the AWSIM Eglinton map, while Fig. 3 D is the AWSIM

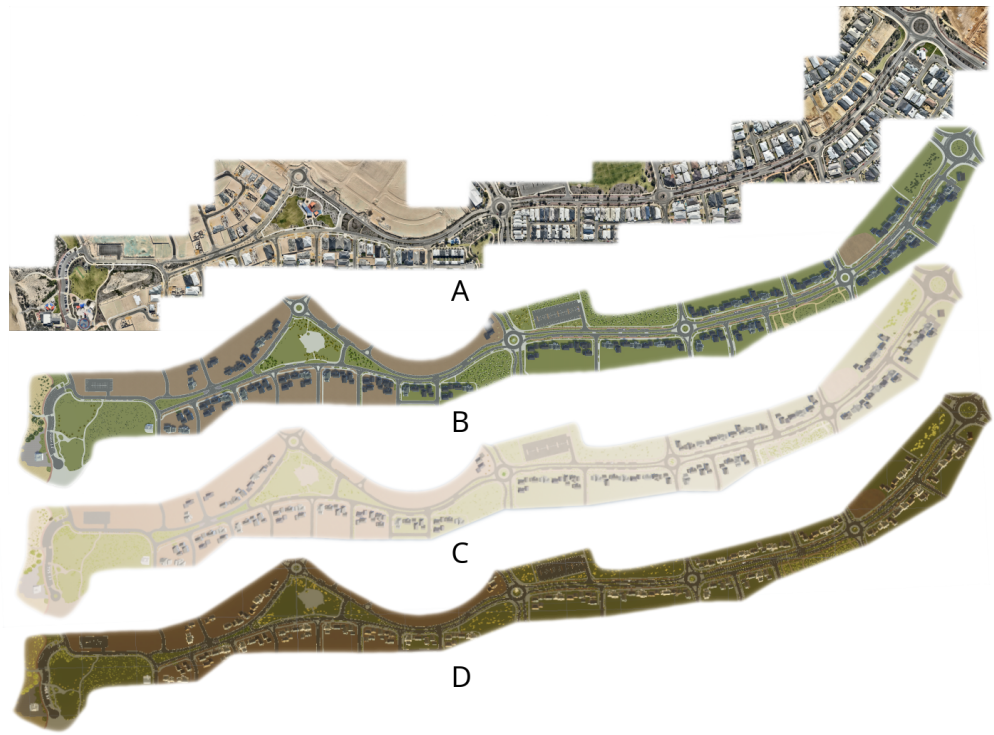


Figure 2: Eglinton maps

Eglinton close-up.

CARLA and AWSIM are tested under the same Level of Detail (LOD).

5.2. Efficiency of Simulation

The efficiency of simulators is evaluated through the analysis of simplification/computational cost and processing speed. Low weights (1-3) are used for efficiency of Simulation metrics in Table 5 since these factors do not impact the outputs of the simulator.

5.2.1. Simplification/Computational Cost

The simulator that requires less computational power for physics calculations has more spare capacity for running AI algorithms. We evaluate the computational cost by measuring the usage and memory consumption of both the CPU and GPU.

Figure 3: Eglinton close shots

Metrics. Monitor and record GPU utilization, GPU used dedicated memory, CPU utilization, and CPU memory by different sensors, including all sensors (cameras, LiDAR, GNSS, and IMU), cameras only, LiDAR only, and GNSS and IMU only, to evaluate each sensor's computational cost. These assessments occur for five minutes at 20 Hz during identical simulation scenarios with their optimal delta time using Python performance monitoring modules: psutil and GPUtil (see Table 3). CARLA's optimal delta time is 0.05, while AWSIM's is 0.017, so changing the delta time may result in bad performance. Reducing CARLA delta time (fixed delta seconds) will lower the ROS publishing rates of cameras and LiDAR and increase frame drops. Increasing AWSIM delta time (fixed time step) will reduce the ROS publishing rates of cameras and LiDAR.

The computational cost (GPU utilization, GPU used dedicated memory, CPU utilization, and CPU memory) scores are calculated for all sensors in Table 5 with CARLA in synchronous mode:

$$S_i = \frac{1}{C_i} \max_{i \in 1,2} C_i \tag{1}$$

where S_i is the computational cost score of simulator i, C_i is the computa-

tional cost of simulator i.

Weights. We assigned weights of 2 to GPU and CPU utilization, while allocating weights of 1 to the dedicated memory used by the GPU and the memory used by the CPU. This decision stems from our belief that the significance of GPU and CPU should be equal, with utilization being prioritized over memory.

5.2.2. Processing Speed

The simulator with a higher processing speed can accelerate the iteration of AI algorithms.

Metrics. Measure the real-time factor (RTF)—the ratio of simulation time to the real time with fixed delta time (CARLA delta time is 0.05, AWSIM delta time is 0.017). This metric is listed in Tables 3 to 5. Table 4 additionally presents the actual completion time for a single lap of the Eglinton map, along with the frame rates of the cameras and LiDAR.

$$R_i = \frac{A_i - B_i}{C_i - D_i} \tag{2}$$

where i is a given simulator, R_i is the RTF, A_i and B_i are the end and start simulation time, C_i and D_i are the end and start real time. The RTF score is calculated for all sensors in Table 5 with CARLA in synchronous mode:

$$F_i = \frac{1}{R_i} \min_{i \in 1,2} R_i \tag{3}$$

where i is a given simulator, F_i is the RTF score, R_i is the RTF.

Weights. We allocated a weight of 3 to the RTF score to ensure it matches the total weight of both the GPU and CPU.

5.3. Realism of Physics Simulation

Evaluate how accurately each simulator models real-world physics, including vehicle dynamics, sensor simulation, and environmental interactions (e.g., weather and road conditions).

5.3.1. Vehicle Dynamics

Accurately simulating vehicle dynamics is crucial for effective control algorithm development in autonomous vehicles. An ideal simulator must capture critical features such as tire-road interaction, multi-body dynamics, actuator modeling, and control systems to support advanced control strategies, vehicle design evaluation, and overall performance enhancement [29]. Each metric outlined below assigns a score of 0.3 to simulators that offer support for that metric via integration with other simulators; a score of 1 is awarded for native support of the metric; otherwise, a score of 0 is given.

1. **Tire-road Interaction:** Models how tires grip or slip on different road surfaces.
2. **Multi-body Dynamics:** Simulates how vehicle parts (e.g., chassis, suspension) move and interact.
3. **Actuator and Powertrain Modeling:** Mimics steering, brakes, and engine/transmission behavior.
4. **Integrated Control System Modeling:** Combines subsystems (e.g., ABS, traction control) with vehicle physics.
5. **Vehicle Parametrization and Customization:** Adjusts vehicle specs (size, weight, etc.) for different simulations.

Weights. We determined a weight of 1 for all vehicle dynamics scores, based on our assessment that these factors have a limited effect on the advancement of ADAS.

5.3.2. Sensor Simulation

Effective sensor modeling is critical for autonomous vehicles, which rely on sensors for environmental perception. Simulators must accurately model sensors such as LiDAR, cameras, radar, IMU, and GNSS to account for sensor noise. Cameras and LiDAR are the two most important and challenging sensors to simulate [30, 31].

Since sensor simulation is a significant section, the weights are assigned according to the criteria outlined in Table 2. With the comparison focusing on performance and efficiency, the sensors are heavily weighted on complexity. For each sensor, the weight is represented in the following format: R0C1I2 represents 0 in rarity, 1 in complexity, and 2 in importance.

Table 2: Sensors Weight Distribution

	Weight	Weight Description
Rarity	2	Based on how many simulations have successfully implemented the sensor. From common to rare.
Complexity	6	Based on the complexity of implementing the sensor on a graphical level. From simple to very complex.
Importance	2	Based on how useful the sensor is for simulation. From ineffective to compulsory.

GNSS. Evaluates whether the simulator includes basic GNSS functionality and how accurately it models position drift and satellite loss under different conditions. A simulator that provides fundamental GNSS capabilities is awarded a score of 0.5. If it additionally incorporates noise modeling alongside basic GNSS functionality, it earns a score of 1. In all other cases, the score is 0. Currently, most autonomous vehicle (AV) simulations integrate GNSS, which is relatively straightforward in complexity. Given that GNSS remains one of the most important sensors, it is assigned a weight of 3 (R0C1I2).

IMU. Assesses support for gyroscope and accelerometer measurements, including noise modeling and realistic drift behavior. A simulator that offers basic IMU functionality earns a score of 0.5. If it also incorporates noise modeling, the score rises to 1. Otherwise, the score is 0. Similar to GNSS, the IMU has been given a weight of 3 (R0C1I2).

RGB Camera. In simulations, particularly for autonomous driving, cameras can have post-effects that simulate real-world visual conditions. These effects enhance realism and make sensor data closer to what a real-world camera might capture. The RGB camera's score is determined by the number of supported post-effects divided by the total available. The evaluated post-effects are:

1. **Bloom:** Simulates light bleed from bright areas (e.g., headlights, sun glare).
2. **Motion Blur:** Simulates blurring caused by the camera's motion or objects in the scene.
3. **Depth of Field:** Adds focal blur to foreground or background.
4. **Vignette:** Darkens image corners, replicating lens light fall-off.
5. **Lens Flare & Glare:** Adds light streaks and glow around bright light sources (e.g., sun, lamps).
6. **Chromatic Aberration:** Simulates color fringing from lens distortion.

7. **Exposure & HDR:** Adjusts scene brightness and contrast dynamically (e.g., dark tunnels, sun).
8. **Noise and Grain:** Adds sensor noise for low-light realism.
9. **Color Grading:** Alters image tones to simulate film styles or weather (e.g., dusk, fog).
10. **Tone Mapping:** Converts HDR lighting to displayable values.
11. **Distortion:** Applies lens warping effects (e.g., fisheye).
12. **Sharpening & Anti-Aliasing:** Enhances or smooths image details and edges.

As previously mentioned, this sensor is among the most important and challenging to simulate; therefore, few simulations have been able to achieve this. Thus, the maximum weight assigned to it is 10 (R2C6I2).

Depth Camera. Depth cameras are important for ADAS algorithm development because they provide pixel-wise depth information. Simulators are scored 1 if supported, otherwise 0. The depth camera is assigned a weight of 2 (R1C0I1), as it does not require a complex graphical environment to simulate, and while it is rare for a simulator to have this, it is not equally important compared to the other sensors.

Event/DVS Camera. Event cameras provide advantages when capturing motion, making them essential for detecting fast-moving objects. Simulators receive a score of 1 if supported, otherwise 0. A weight of 2 (R1C0I1) is assigned to the event camera, with the same explanation as the depth camera.

LiDAR. High-fidelity LiDAR simulations should provide realistic point cloud data and reflect real-world challenges such as noise, range limitations, environmental conditions, and material reflections. The fidelity score of the LiDAR is determined by dividing the number of supported fidelity metrics by the total available. The metrics for LiDAR fidelity evaluation are:

1. **Total Configurable Parameters:** This metric is valued at 0.5 when the number of parameters ranges from 0 to 30, at 1 when it exceeds 30, and is 0 otherwise.
2. **Point Cloud Configuration:** Adjustable density, range, and field of view (FoV).
3. **Noise Modeling:** Gaussian noise that scales with distance and surface.

4. **Ghost Points and Multipath Effects:** Simulates false returns due to reflections, which occur in real LiDAR due to multipath propagation.
5. **Material Reflection:** Models how different materials (e.g., glass, asphalt, vegetation) reflect LiDAR.
6. **Weather Conditions:** Degrades LiDAR under rain, fog, or dust.
7. **Sunlight Interference:** Models distortion under direct sunlight.
8. **Clipping Range:** Proper near and far clipping planes.
9. **Occlusion Handling:** Realistic occlusion and shadow in point clouds.
10. **Dynamic Object Tracking:** Accurate moving object representation.
11. **Deformation Artifacts:** Models deformable objects (e.g., swaying trees).
12. **GPU Support:** Uses GPU to accelerate LiDAR simulation.
13. **Point Cloud Format:** Compatible with real LiDAR output formats for easy integration and analysis. It is assigned a value of 0.5 for support of the XYZI format, 1 for support of the XYZIRCAEDT format, and 0 otherwise.

Simulating LiDAR, particularly the Velodyne model with its multiple layers, presents significant complexity. Furthermore, it is essential to tailor the environment to accurately simulate LiDAR data capture as it occurs in the real world. Consequently, this sensor is assigned a maximum weight of 10 (R2C6I2), comparable to that of the RGB camera.

Radar. Checks support for radar functionality and noise modeling, including range, Doppler effects, object tracking under occlusion, and power level. A score of 0.5 is given to a simulator that offers basic radar functionality. If it further supports noise modeling along with the basic radar features, it achieves a score of 1. If neither is present, the score is 0. Since the Radar is being overshadowed by the LiDAR, a weight of 2 is given (R1C1I0).

V2X / V2I (Vehicle-to-Everything / Infrastructure). Assesses whether the simulator supports communication between the vehicle and infrastructure (e.g., traffic lights) or other vehicles, which is crucial for cooperative driving scenarios. Simulators receive a score of 1 if supported, otherwise 0. Given that few simulations are specifically designed for this purpose (e.g., SUMO), and in terms of graphical requirements, the sensor does not necessitate extensive detail. Rather, it focuses more on managing multiple infrastructures and external factors. Therefore, a weight of 1 (R1C0I0) is assigned.

5.3.3. Environmental Interactions

Simulators must support the controlled and reproducible simulation of environmental phenomena—such as lighting, weather, and road conditions—to ensure the transferability of perception algorithms from simulation to real-world deployment. In this evaluation, scores are derived from how well each simulator represents or supports these conditions, with 1.0 indicating full native support, 0.5 for partial/limited support, and 0.0 for absence. Where relevant, intermediate scores such as 0.8 or 0.3 are used to reflect visual quality, configurability, or extensibility. Weights are assigned based on two criteria: (i) the frequency and impact of these environmental effects in real-world driving, and (ii) the degree to which they challenge the robustness and generalization capability of autonomous systems.

Adverse weathers. Simulating a wide range of weather conditions is essential for testing the perception stack's resilience under varying visibility, reflection, and occlusion effects. The weights for weather conditions are determined by their prevalence in real driving scenarios, with more common phenomena such as night or cloudy weather receiving higher weights. Each condition is rated on a scale from 1 (very rare) to 4 (very common). Accordingly, night and cloudy conditions are weighted at 4 due to their regular occurrence. Rainy and foggy conditions, while less frequent, are still critical for perception testing and thus receive weights of 4 and 2, respectively. Snowy conditions, being geographically and seasonally constrained, are weighted at 1.

Lighting Conditions. Lighting variations pose a significant challenge to perception models, particularly camera-based systems. Accurate rendering of lighting effects—such as shadows, reflections, and glare—is essential for replicating real-world sensor performance. A medium weight of 5 is assigned to Lighting Conditions to reflect its critical role in sensor realism, while also acknowledging its limited direct impact on non-visual sensor modalities such as LiDAR and radar.

Road Conditons. Under road conditions, we evaluate traffic-related signaling infrastructure such as pedestrian lights and classic traffic lights, which are essential for decision-making modules in urban environments. Their presence enables validation of behavior planning and interaction rules in accordance with traffic laws. Given their importance, but relative simplicity and static nature, pedestrian lights are assigned a weight of 4, and classic traffic lights

are given a slightly higher weight of 5 due to their ubiquity and critical role in intersection handling.

NPCs. Dynamic traffic participants form the backbone of a realistic simulation ecosystem. Weights for different types of NPCs are based on their statistical frequency in real traffic and the level of behavioral complexity they introduce. Cars, being the most prevalent and behaviorally diverse actors on the road, receive the highest weight of 5. Pedestrians, while less frequent, interact closely with the ego vehicle and involve complex visual recognition and prediction tasks, justifying a weight of 4. Trucks are assigned a weight of 3 for their impact on occlusion and motion dynamics, despite being less common. Cyclists and motorcyclists, although less frequent and often more predictable in their movement, still introduce unique challenges, such as lateral instability and variable speeds, and are thus assigned a weight of 2.

5.4. Scenario Creation and Testing

Compare the ease of creating, configuring, and testing different driving scenarios, including the various scenarios supported (e.g., urban, highway, off-road) and the complexity of scenario setups.

5.4.1. Scenario Construction

Having a robust API for scenario generation is vital for expediting the ADS testing pipeline, as it allows efficient scripting and automation of diverse test cases. Each subcomponent of scenario construction is evaluated and scored based on its ease of implementation, configurability, and flexibility, with scores ranging from 0 to 1. A score of 1 represents full support with minimal effort (e.g., intuitive UI, extensive documentation, automated pipelines), while a score closer to 0 reflects either limited functionality or a high implementation burden.

Weights are assigned based on each element's practical importance and complexity, per Table 1.

New Map (Weight = 10). The map is the foundational asset for any scenario. Its complexity lies in generation, importation, customization, and navigation integration. Given its central role and high configuration burden, we assign it the highest weight.

- **Generation:** This step refers to creating a map using 3D modeling tools or geographic data sources. A user-friendly map editor or support for GIS datasets accelerates this process.

- **Importation:** The ability to import the generated map into the simulator's environment with minimal manual conversion is essential. Support for common formats (e.g., FBX, OBJ, OpenDRIVE) is a key consideration.
- **Customization:** After import, the map should be editable within the simulator. This includes annotating road elements (e.g., traffic lanes, stop lines) and placing dynamic actors (e.g., pedestrians, traffic lights).
- **Navigation:** A complete integration requires generating navigation meshes, waypoints, or HD maps that allow vehicles to localize and plan paths through the new environment.

New Vehicle (Weight = 8). While slightly less complex than maps, vehicles require careful configuration of both visuals and dynamics. Their prevalence and impact on simulation fidelity justify a high weight.

- **Prepare Vehicle Model:** A 3D model of the vehicle must be obtained from external libraries or designed in software such as Blender. The model should be segmented by components (e.g., chassis, wheels, lights) with correct materials and textures.
- **Import and Configure Physics:** The 3D model is imported into the simulator's engine from common formats such as FBX and OBJ, and vehicle dynamics must be configured. This includes defining physical properties such as mass, center of mass, and collision shapes.
- **Configure Wheels:** Accurate simulation of wheel behavior involves setting up parameters such as suspension stiffness, damping, and steering for front and rear wheels.
- **Configure Lights:** The simulator should allow functional integration of headlights, brake lights, and indicators, ideally controllable via API for behavior scripting.

New Sensors With Templates (Weight = 5). Simulators often provide predefined templates for commonly used sensors (e.g., cameras, LiDAR, radar). These templates streamline the process of adding sensors to vehicles by allowing users to configure basic parameters such as FoV, resolution, and update rate without coding from scratch. A user-friendly UI or YAML/JSON interface for configuring these templates improves usability. These sensors are

easier to implement due to existing templates, and their role is supportive rather than foundational. A medium weight reflects their moderate impact on scenario design.

New Sensors Without Templates (Weight = 3). For advanced or custom sensors, simulators should allow manual integration via APIs or plugin systems. This involves defining sensor behavior in code, attaching it to vehicle frames, and ensuring proper data streaming. The extensibility of the sensor framework (e.g., C++, Python, or plugin-based design) is crucial for research flexibility. These sensors represent edge cases where users go beyond built-in capabilities. Their rarity and the fact that default sensors are often sufficient justify a lower weight.

5.4.2. Scenario Library and Traffic Manager

A rich scenario library simplifies simulation, enabling quicker setup and collaboration. It should include predefined maps, vehicle models, and pedestrian models. We count the number of sample maps, vehicles, and pedestrians to test the richness of the scenario library. A simulator that provides a single sample map, vehicle, or pedestrian receives a score of 0.2. For each additional sample, the simulator's score increases by 0.1 for maps, 0.05 for vehicles, and 0.08 for pedestrians, up to a maximum score of 1. If no samples are provided, the score remains 0. For traffic scenarios, simulators should support traffic aids such as traffic lights and road signs, which help regulate traffic. Additionally, simulators must facilitate the motion planning for NPCs representing vehicles and pedestrians. The scoring of a simulator's traffic management effectiveness emphasizes the simplicity of managing traffic signals and the complexity of NPC motion planning.

The weights assigned to the scenario library metrics are also relevant to the discussion in the previous section. Given the challenges associated with map creation, having a selection of predefined maps is advantageous for reducing creation time or quick testing, resulting in a high weight of 7 for the number of maps. The vehicle category follows closely, sharing a similar rationale, and is assigned a medium weight of 5. In contrast, while having a few pedestrian models available is beneficial, they are not as convenient as the other two categories, plus given the considerable similarity among most pedestrian models, this category receives a low weight of 3.

As previously highlighted, effective traffic management is essential for autonomous vehicle (AV) simulations, as the ultimate goal is to develop

realistic traffic scenarios for testing purposes. Consequently, the weights assigned to this category are substantial, with a value of 6 given to traffic light management and vehicle motion. In comparison, pedestrian motion receives a low weight of 3 since our focus is not on pedestrian-intensive scenarios.

5.5. User-Friendliness

Indicators such as UI functionality, setup complexity, community support strength, affordability, and platform support are legitimate factors in assessing the difficulty of mastering a simulator. Scoring in this category is based on a qualitative evaluation along a scale from 0.0 to 1.0 for each subcategory, emphasizing the simplicity of installation, the depth of UI features and documentation, the community engagement, and the software and hardware prerequisites. While user-friendliness is important, this study primarily emphasizes simulation effectiveness and technical capabilities. Hence, this category is weighted less heavily in the overall evaluation. Specifically, each element in subcategories—UI functionality, setup, affordability and platform support—is assigned a low weight of 2. An exception is made for elements in community and support, which are assigned a slightly higher weight of 3 due to their long-term value in helping users overcome limitations, learn best practices, and extend functionality.

5.5.1. UI functionality

Every simulator needs to be able to run simulations in a user-defined environment. A robust UI allows for the easy creation of simulations by drag-and-drop controls on models. A good UI can also provide live data more visually, which can help researchers understand and debug simulation runs.

5.5.2. Setup

This includes the ease of installation, system configuration, and initial usage. Tools that offer pre-built binaries, package managers, or clear setup scripts are scored more favorably than those requiring complex build processes or extensive manual configuration.

5.5.3. Community and Support

Evaluate the strength of the community and the availability of documentation, tutorials, and support resources, including how active the user community is and the responsiveness of developers to issues and feature requests.

5.5.4. Affordability & Platform Support

Compare the cost structures (not all researchers can afford expensive and high-end devices), licensing models, platform support, and any potential restrictions on using each simulator for commercial or research purposes as listed in Table 8. Simulators that are free, open-source, and support a wide range of platforms with modest hardware needs score higher in this category.

5.6. Extendability

Extendability evaluates how easily a simulator can integrate with external tools and frameworks, such as ROS, ML libraries, and annotation sensors. These capabilities are essential for supporting the full development cycle of autonomous driving systems—from perception and planning to validation.

5.6.1. ROS Integration

ROS is a widely used robotics middleware that enables real-time communication and modularity in autonomous systems. While not strictly mandatory (e.g., CARLA supports a standalone Python API), native ROS integration—either built-in or supported through official plugins (e.g., Abdelhamed et al. manage to design a software-in-the-loop (SIL) framework based on the gazebo_ros_pkgs plugin [32])—greatly enhances workflow compatibility, especially in research and robotics applications. Simulators are scored from 0.0 to 1.0, indicating no support, partial or outdated support, and full ROS integration. Given its foundational role in AV development, ROS integration is assigned a high weight of 10, comparable to critical categories such as RGB cameras and LiDAR.

5.6.2. Integration with ML/AI Frameworks

An important capability of autonomous driving simulators is their ability to integrate seamlessly with modern ML and AI frameworks (e.g., TensorFlow, PyTorch). This integration supports the training and evaluation of perception, planning, and control algorithms, including tasks such as object detection, semantic segmentation, and behavior prediction.

Simulator Detectors. Simulators often provide virtual sensors—referred to as simulator detectors—that act as ground-truth sources for testing autonomous vehicle systems in controlled environments. These detectors are essential for validating safety and performance.

- **Collision Detectors:** Monitor contact between objects to assess safety.

- **Lane Invasion Detectors:** Track when a vehicle crosses lane markings, which is critical for evaluating lane-keeping and path-following behavior.
- **Obstacle Detectors:** Provide precise distance measurements and object presence information ahead of the ego-vehicle or actors. These are particularly useful for benchmarking algorithms such as obstacle avoidance, emergency braking, and safe path planning.

Each detector type contributes 1 point to the score if natively supported; otherwise, 0. We assign a medium weight of 6, as these detectors support ADAS evaluation but are not directly involved in training perception models.

Annotation Sensors. Simulators that provide ground-truth annotations significantly accelerate dataset creation for supervised learning. We distinguish between native support (score = 1), plugin-based support (score = 0.3), and no support (score = 0).

- **Camera Annotation Sensors:** Generate pixel-wise labels or bounding boxes for objects in rendered images. These labels are crucial for training and evaluating visual perception models such as image segmentation, object detection, and tracking.
- **LiDAR Annotation Sensors:** Provide labeled 3D point clouds, which are essential for training LiDAR-based neural networks in tasks such as 3D object detection, instance segmentation, and dynamic scene understanding.

For camera annotation sensors, we concluded that the camera semantic segmentation and the 2D bounding box metric should each carry a high weight of 10, as they are imperative for the advancement of ML/AI algorithms. On the other hand, the camera instance segmentation and the 3D bounding box, which represent more advanced forms of the previous metrics, were given a marginally lower weight of 8. The implementation of optical flow is relatively straightforward using the OpenCV package, making native support for this function less critical; thus, we allocated a weight of 5. For LiDAR annotation sensors, we assigned a weight of 10 to the LiDAR semantic segmentation score and a weight of 5 to the LiDAR intensity score. This weighting reflects our judgment that LiDAR semantic segmentation significantly enhances the labeling process for the LiDAR-based ML/AI algorithms,

thereby improving scene understanding, in contrast to the relatively minor contribution of the intensity score.

5.6.3. Programming Languages

Support for multiple programming languages improves a simulator's adaptability and broadens its user base. We score simulators 0.5 for support of either C/C++/C# or Python, and 1.0 for both. The programming language support is assigned a medium weight of 5, as it fosters custom tool development and long-term maintenance.

6. Results

All the results selected for scoring and weighting are listed in Table 5.

6.1. Efficiency of Simulation

6.1.1. Simplification/Computational Cost and Processing Speed

As shown in Table 3, AWSIM demonstrates lower GPU memory usage, CPU utilization, and CPU memory consumption than CARLA. Although AWSIM exhibits similar or higher GPU utilization, it generally requires less computing power overall. Both Tables 3 and 4 illustrate RTF performance on the Eglinton map. AWSIM consistently operates in real time, while CARLA faces issues in both execution modes: asynchronous mode is excessively fast and prone to frame drops, compromising control, whereas synchronous mode is too slow and results in low ROS topic publishing rates. Additionally, AWSIM supports dynamic time scaling, enhancing its temporal flexibility.

Regarding sensor load, RGB cameras are the most GPU-intensive, particularly in AWSIM. Therefore, scaling visual tasks (e.g., semantic segmentation, depth estimation) with AWSIM will require GPU-side optimization. LiDAR adds more CPU pressure, particularly in CARLA with asynchronous mode. GNSS/IMU sensors are negligible in resource cost, but CARLA doesn't throttle or batch them efficiently, leading to huge RTFs.

6.2. Realism of Physics Simulation

6.2.1. Vehicle Dynamics

Unity or Unreal game engines use NVIDIA PhysX pre-integrated, so CARLA, which is based on UE4, and AWSIM, which is based on Unity,

Table 3: CPU, GPU and RTF Comparison Table

	Metrics	CARLA Async	CARLA Sync	AWSIM
All Sensors	GPU Utilization	39.55%	**32.52%**	37.85%
	GPU Used Dedicated Memory	30.30%	30.21%	**16.98%**
	RTF	128.79%	65.50%	100.00%
	CPU Utilization	**25.13%**	29.56%	30.02%
	CPU Memory	22.70%	22.34%	**12.68%**
Cameras Only	GPU Utilization	40.16%	**39.22%**	66.36%
	GPU Used Dedicated Memory	30.27%	30.23%	**9.48%**
	RTF	133.17%	124.13%	100.00%
	CPU Utilization	**16.54%**	30.57%	31.05%
	CPU Memory	22.53%	22.31%	**12.68%**
LiDAR Only	GPU Utilization	38.75%	**30.19%**	38.12%
	GPU Used Dedicated Memory	29.28%	29.23%	**16.49%**
	RTF	251.12%	116.02%	100.00%
	CPU Utilization	36.76%	32.47%	**22.46%**
	CPU Memory	22.54%	22.24%	**11.07%**
GNSS and IMU Only	GPU Utilization	39.75%	38.79%	**38.66%**
	GPU Used Dedicated Memory	29.33%	29.24%	**8.94%**
	RTF	300.76%	265.69%	100.00%
	CPU Utilization	20.76%	33.51%	**14.20%**
	CPU Memory	22.38%	22.11%	**10.40%**

Table 4: One Lap Comparison Table

	CARLA Async	CARLA Sync	AWSIM
One Lap Real Finish Time (s)	848.8	1373	812.8
RTF	118.63%	59.61%	100.00%
Camera FPS	23.77	11.94	20
LiDAR FPS	23.77	11.94	20

have the same vehicle dynamics level. Trade-offs in NVIDIA PhysX prioritize visualization over precise physics behavior [29], so both have low vehicle dynamics performance. Still, CARLA supports integration with CarSim (a commercial software), which has accurate vehicle dynamics, and hence, CARLA obtains a higher score than AWSIM.

Both AWSIM and CARLA utilize NVIDIA PhysX through their respective game engines (Unity and UE4), leading to comparable baseline vehicle dynamics. However, PhysX is optimized more for visual fidelity than physical realism [29]. As a result, both simulators offer limited fidelity in vehicle

Table 5: CARLA vs. AWSIM Comparison Table

Criteria	Weight	CARLA			AWSIM		
		Raw	Score	Weighted	Raw	Score	Weighted
GPU Utilization	2	32.52%	1.000	2.000	37.85%	0.859	1.718
GPU Used Dedicated Memory	1	30.21%	0.562	0.562	16.98%	1.000	1.000
RTF	3	65.50%	0.655	1.965	100.00%	1.000	3.000
CPU Utilization	2	29.56%	1.000	2.000	30.02%	0.985	1.969
CPU Memory	1	22.34%	0.568	0.568	12.68%	1.000	1.000
Tire-Road Interaction	1	-	0.300	0.300	-	0.000	0.000
Multi-body Dynamics	1	-	0.300	0.300	-	0.000	0.000
Actuator and Powertrain Modeling	1	-	0.300	0.300	-	0.000	0.000
Integrated Control System Modeling	1	-	0.300	0.300	-	0.000	0.000
Vehicle Parametrization and Customization	1	-	0.300	0.300	-	0.000	0.000
GNSS	3	✓	1.000	3.000	✓	0.500	1.500
IMU	3	✓	1.000	3.000	✓	0.500	1.500
RGB Camera	10	✓	0.800	8.000	✓	1.000	10.00
Depth Camera	2	✓	1.000	2.000	-	0.000	0.000
Event/DVS Camera	2	✓	1.000	2.000	-	0.000	0.000
LiDAR	10	✓	0.500	5.000	✓	0.714	7.143
Radar	2	✓	0.500	1.000	✓	1.000	2.000
V2X	1	✓	1.000	1.000	✓	1.000	1.000
Night	4	✓	1.000	4.000	-	0.300	1.200
Cloudy	4	✓	1.000	4.000	-	0.300	1.200
Rainy	4	✓	0.800	3.200	-	0.200	0.800
Foggy	2	✓	0.800	1.600	-	0.200	0.400
Snowy	1	-	0.100	0.100	-	0.200	0.200
Lighting Conditions	5	✓	1.000	5.000	✓	0.800	4.000
Pedestrian Lights	4	✓	1.000	4.000	✓	1.000	4.000
Classic Traffclights	5	✓	1.000	5.000	✓	1.000	5.000
Pedestrians	4	✓	1.000	4.000	✓	1.000	4.000
Cyclist/Motorcyclist	2	✓	1.000	2.000	-	0.000	0.000
Cars	5	✓	1.000	5.000	✓	1.000	5.000
Trucks	3	✓	1.000	3.000	✓	1.000	3.000
New Maps	10	✓	0.700	7.000	✓	0.800	8.000
New Vehicles	8	✓	0.800	6.400	✓	1.000	8.000
New Sensors With Templates	5	✓	0.800	4.000	✓	1.000	5.000
New Sensors Without Templates	3	✓	1.000	3.000	-	0.000	0.000
Number of Sample Maps	7	8	1.000	7.000	1	0.200	1.400
Number of Sample Vehicles	5	36	1.000	5.000	6	0.400	2.000
Number of Sample Pedestrians	3	11	1.000	3.000	1	0.200	0.600
Vehicle Motion	6	✓	0.850	5.100	✓	0.700	4.200
Pedestrain Motion	3	✓	1.000	3.000	✓	0.500	1.500
Traffic Lights Management	6	✓	1.000	6.000	✓	1.000	6.000
Modular Editing	2	✓	1.000	2.000	✓	1.000	2.000
Data Visulization	2	✓	1.000	2.000	✓	1.000	2.000
Installation	2	Slow	0.200	0.400	Fast	1.000	2.000
Availability of documentation	3	High	1.000	3.000	Normal	0.600	1.800
Community Activity	3	High	1.000	3.000	Low	0.200	0.600
Licensing Fees	2	Open-source	1.000	2.000	Open-source	1.000	2.000
System Requirements	2	Normal	0.800	1.600	High	0.500	1.000
ROS Integration	10	Foxy	0.500	5.000	Humble	0.800	8.000
Collision Detector	6	✓	1.000	6.000	-	0.000	0.000
Lane Invasion Detector	6	✓	1.000	6.000	-	0.000	0.000
Obstacle Detector	6	✓	1.000	6.000	-	0.000	0.000
Camera Semantic Segmentation	10	✓	1.000	10.000	-	0.300	3.000
Camera Instance Segmentation	8	✓	1.000	8.000	-	0.300	2.400
2D Bounding box	10	✓	1.000	10.000	-	0.300	3.000
3D Bounding box	8	✓	1.000	8.000	-	0.300	2.400
Optical flow	5	✓	1.000	5.000	-	0.000	0.000
LiDAR Semantic Segmentation	10	✓	1.000	10.000	✓	1.000	10.000
LiDAR Intensity	5	-	0.000	0.000	✓	1.000	5.000
Programming Languages	5	Python, C++	1.000	5.000	C#	0.500	2.500
Total	251	-	48.435	216.995	-	32.158	145.031

Table 6: CARLA vs. AWSIM Camera Post-effects Comparison Table

Post-Effect	CARLA	AWSIM
Bloom	✓	✓
Motion Blur	✓	✓
Depth of Field	✓	✓
Vignette	✓	✓
Lens Flare	✓	✓
Chromatic Aberration	✓	✓
Exposure Adjustment	✓	✓
HDR	✓	✓
Noise and Grain	✓	✓
Color Grading	✓	✓
Tone Mapping	✓	✓
Distortion	✓	✓
Glare and Halo	-	✓
Sharpening	-	✓
Anti-Aliasing	-	✓

Table 7: CARLA vs. AWSIM LiDAR Fidelity Comparison Table

LiDAR Fidelity Metrics	CARLA	AWSIM
Total Configurable Parameters	13	32
Configurable Point Cloud Density and range	✓	✓
Configurable Field of View (FoV)	✓	✓
Noise Modeling	✓	✓
Ghost Points and Multipath Effects	-	-
Material Reflection	-	✓
Weather Conditions	-	-
Sunlight Interference	-	-
Clipping Ranges	✓	✓
Occlusion Handling	✓	✓
Dynamic Object Tracking	✓	✓
Deformation Artifacts	-	-
GPU Support	-	✓
Point Cloud Format	XYZI	XYZIR/XYZIRCAEDT

dynamics out of the box. CARLA gains an advantage through its optional integration with CarSim—a commercial-grade vehicle dynamics engine—which

enables high-accuracy simulations. This external integration justifies a higher score for CARLA in this category.

6.2.2. Sensor Simulation

CARLA provides native support for all listed sensors, while AWSIM lacks support for depth and event/DVS cameras. Sensor-specific evaluations are as follows:

1. **GNSS:** AWSIM outputs only positional data without rotation or covariance, whereas CARLA provides more complete sensor outputs.
2. **IMU:** AWSIM lacks a noise model, which limits its realism for motion estimation tasks.
3. **RGB Camera:** As shown in Table 6, both simulators implement essential post-processing effects. AWSIM supports a broader set of advanced visual effects. CARLA has documented issues with anti-aliasing and lacks support for glare and halo effects.
4. **LiDAR:** According to Table 7, AWSIM offers more configurable parameters (e.g., noise models, material reflectivity), and its GPU-accelerated processing supports dense point clouds efficiently. AWSIM also exports point clouds in the rich XYZIRCAEDT (XYZ position, intensity, return type (including big-endian), channel (or ring), azimuth, elevation, distance) format, closely aligning with real-world formats such as Velodyne's XYZIRT (XYZ position, intensity, ring (or channel), time). In contrast, CARLA uses a simpler XYZI (XYZ position, intensity) format, offering less fidelity.
5. **Radar:** AWSIM mirrors its LiDAR advantages here, with GPU support and extended configuration options, whereas CARLA offers only basic radar emulation.
6. **V2X/V2I:** AWSIM supports a V2I sensor for receiving nearby traffic light states. CARLA includes a more structured V2X framework with customizable messages, although this is limited to the development (DEV) version.

6.2.3. Environmental Interactions

CARLA provides a dynamic weather API capable of simulating various conditions such as night, cloudy skies, rain, and fog. However, it does not natively support snowfall. In contrast, AWSIM does not include a built-in weather API, but weather effects can be added using Unity editor assets or

third-party plugins. Unity's asset ecosystem is beginner-friendly for implementing basic weather conditions (e.g., particle-based rain or fog), whereas UE4, used by CARLA, includes a sophisticated built-in atmospheric system. While UE4 offers higher fidelity and lower dependence on third-party tools for advanced weather, it comes with a steeper learning curve.

It is important to note that, in both Unity and UE4, simulated weather conditions such as rain, snow, and fog currently do not affect LiDAR behavior, limiting their utility for testing perception under degraded environmental conditions.

Both simulators support configurable lighting conditions, including time-of-day changes, pedestrian signals, and traffic lights. CARLA uses circular pedestrian light indicators, while AWSIM uses human-shaped icons. Additionally, AWSIM's sun and lighting system is managed via third-party Unity assets, whereas CARLA leverages UE4's native atmospheric lighting.

Regarding road user diversity, CARLA includes pedestrians, cyclists/motorcyclists, cars, and trucks. AWSIM supports pedestrians, cars, and trucks but currently lacks support for cyclists or motorcyclists. This limitation reduces the realism of certain urban driving scenarios, such as shared roadways or dense city environments.

Summary: CARLA offers more comprehensive out-of-the-box support for environmental dynamics and actor diversity, but Unity's flexibility allows AWSIM users to rapidly prototype custom weather effects with lower technical barriers. For advanced and physically consistent weather systems, CARLA remains more robust, although neither simulator supports LiDAR degradation effects from weather.

6.3. Scenario Creation and Testing

6.3.1. Scenario Construction

New Map.

- **Generation:** Both simulators support map creation via RoadRunner (free for students) or TrueVison (free), with exports typically in FBX or OBJ formats.
- **Importation:** CARLA offers two primary approaches for map importation. The first involves building from source, which is time-consuming due to UE4's need to convert every mesh into a native asset. The second uses Docker, requiring approximately 700 GB of storage and

disallowing further edits post-import. In contrast, AWSIM supports direct importation of RoadRunner maps in FBX format, enabling a significantly faster and more flexible workflow.

- **Customization:** Both simulators allow collaborative map editing by multiple users. However, RoadRunner assets may require corrections (e.g., missing materials or meshes). Based on testing, Unity (used by AWSIM) exhibits greater compatibility with RoadRunner assets, resulting in reduced post-processing effort compared to UE4 (used by CARLA).

- **Navigation:** RoadRunner automatically generates OpenDrive files (in XODR format), which CARLA can directly utilize for vehicle and pedestrian navigation without manual configuration. AWSIM, on the other hand, supports Lanelet2 (in OSM format), which requires manual lane drawing. A tool by the Technical University of Munich (TUM), CommonRoad Scenario Designer [33], allows efficient conversion from OpenDrive to Lanelet2.

CARLA supports large maps by splitting them into 2 km $\times$ 2 km tiles, which optimizes resource usage. AWSIM currently lacks native support for large map tiling. Both simulators support OpenScenario for random and route-based vehicle traffic generation.

New vehicle.

- **Prepare Vehicle Model:** Vehicle integration in CARLA involves several steps, including skeletal bone binding, proper naming of parts and textures, and creating separate meshes for physical assets and ray-cast sensors (with more details).

- **Import and Configure Physics:** CARLA includes additional animation and physics setup steps not required by AWSIM.

- **Configure Wheels:** Wheel configuration is straightforward in both simulators.

- **Configure Lights:** CARLA provides templates for all vehicle lights (e.g., blinkers, hazard lights, reverse lights), though documentation is lacking. AWSIM also supports all standard lights and offers user-friendly implementation guides.

New Sensors With Templates. Both CARLA and AWSIM provide templates for standard sensors, including LiDAR, cameras, radar, IMU, and GNSS. AWSIM includes more LiDAR presets with corresponding 3D models and parameter configurations. CARLA lacks 3D sensor models and relies solely on functional implementations.

New Sensors Without Templates. For non-standard sensor integration, CARLA provides a complex tutorial, while AWSIM currently lacks any official documentation or guides.

6.3.2. Scenario Library and Traffic Manager

As summarized in Table 5, CARLA features a significantly richer scenario library compared to AWSIM. Both simulators support random and route-based traffic generation. CARLA leverages OpenDrive data—including lane types, junctions, and traffic light metadata—for both pedestrian and vehicle navigation. AWSIM requires more manual setup even when a Lanelet2 file is provided; it only extracts vehicle lane rules and mandates manual configuration for pedestrian motion (limited to back-and-forth linear movement), junction behavior, and traffic light control.

Despite these limitations, AWSIM offers a more nuanced intersection management script that supports yielding behavior (e.g., right-turning vehicles yielding to oncoming traffic). CARLA, in turn, supports driver behavior customization, allowing adjustment of NPC aggressiveness and error rates. Furthermore, CARLA improves performance on large maps using an auto-activation script that loads NPCs only within active tiles, reducing the simulation's computational load.

6.4. User-Friendliness

6.4.1. UI functionality

Both UE4 and Unity support modular editing through reusable components—Blueprints in UE4 and Prefabs in Unity. These tools enable developers to encapsulate logic, behaviors, and properties within customizable assets. UE4's Blueprint system offers robust support for complex logic, making it ideal for detailed and scalable systems. In contrast, Unity's Prefab system is generally more accessible for new users and easier to modify.

For data visualization, CARLA offers integration with Rviz and Carlaviz (a web-based visualizer), while AWSIM supports Rviz and native rendering within the Unity Editor.

6.4.2. Setup

Installing CARLA is time-intensive, particularly when building from source, which can take over four hours. AWSIM, by contrast, can typically be set up within one hour using the Unity Editor. The UE4 Editor used by CARLA requires more installation steps and is more prone to compilation failures, making AWSIM comparatively easier to install and configure.

6.4.3. Community and Support

Documentation: CARLA features comprehensive and well-organized documentation, including detailed API references and tutorials ranging from basic to advanced. This depth reflects its maturity and broader adoption in both academia and industry. In comparison, AWSIM's documentation is more limited, with fewer API references and only basic tutorials available.

Community and Resources: CARLA has a larger and more active community, evidenced by a higher number of GitHub stars and a broader base of supplemental resources (e.g., research papers, tutorials). AWSIM's community is still in its early stages but is growing steadily.

Ease of Information Access: Due to its mature documentation and active forums, solutions to CARLA-related issues are easier to find. AWSIM users may need to rely more on direct experimentation, although improvements are expected as its user base expands.

6.4.4. Affordability & Platform Support

CARLA and AWSIM are open-source; however, both rely on proprietary tools such as RoadRunner for map creation. Most university researchers can access RoadRunner through MATLAB campus licenses. Purchasing third-party assets for Unity or UE4 incurs additional costs.

Table 8 outlines system requirements. AWSIM demands a higher-end CPU and GPU and currently supports only Ubuntu 22.04 and Windows. CARLA supports a broader range of OS versions. Although both simulators run on Windows, ROS is not officially supported on that platform, making Linux the preferred choice for AV research and development.

6.5. Extendability

6.5.1. ROS Integration

CARLA supports ROS2 Foxy via an official ROS bridge, with version 0.9.13 being the latest stable release supporting it. AWSIM, on the other

Table 8: CARLA vs. AWSIM System Requirements

Minimum Requirements	CARLA	AWSIM
CPU	4 cores	6 cores and 12 threads
GPU	6 GB GPU	RTX 2080 Ti
Storage	130 GB	18 GB
Operating System	Ubuntu 18.04/20.04; Windows 10/11	Ubuntu 22.04; Windows 10/11

hand, natively supports ROS2 Humble, a newer and more feature-rich ROS distribution.

6.5.2. Integration with ML/AI Frameworks

CARLA supports all the simulator detectors and annotation sensors via predefined object labels, which are useful for evaluating AV systems. AWSIM does not currently offer built-in tools for AV evaluation. However, Unity provides a perception module that can be leveraged for label generation and other AI-related tasks. Both CARLA and AWSIM support LiDAR semantic segmentation.

6.5.3. Programming Languages

CARLA supports both Python and C++ for standalone executables. Its UE4 editor is C++-based, enabling low-level customization. AWSIM does not yet support scripting for standalone Unity builds but offers configurable GUI elements (e.g., function buttons in the game window). Unity editor primarily uses C# for all scripting and behavior logic.

7. Discussion and Conclusion

This paper presents a comparative study of UE4-based CARLA and Unity-based AWSIM, evaluating their suitability for autonomous driving research on identical maps. The analysis demonstrates that differences in the underlying game engines—Unreal Engine and Unity—lead to distinct trade-offs in realism, usability, and development efficiency.

From a quantitative perspective, the weighted evaluation framework shows that CARLA achieves a total score of **216.995 out of 251**, while AWSIM scores **145.031 out of 251**. The score gap is mainly driven by CARLA's advantages in scenario richness, scalability, traffic management, and API maturity. These results support the conclusion that CARLA is currently

better suited for large-scale, systematic end-to-end testing under controlled experimental settings.

AWSIM, despite its lower overall score, performs strongly in specific high-impact categories. In particular, it demonstrates advantages in rapid development cycles, native ROS2 Humble integration, and LiDAR simulation fidelity. These strengths explain its competitiveness for LiDAR-based learning approaches and modular autonomy frameworks such as Autoware and Apollo, even though they contribute less to the final weighted score.

It is important to emphasize that the quantitative scores are influenced by the weighting strategy adopted in this study. Safety-related factors were intentionally assigned relatively low weights, reflecting a focus on simulator effectiveness for algorithm development rather than safety validation or regulatory assessment. As a result, the higher total score of CARLA should not be interpreted as superior safety performance. This limitation highlights the need for alternative weighting schemes in future studies that prioritize safety-critical evaluation.

Regarding traffic rule configuration, both simulators are capable of supporting right-hand and left-hand driving scenarios through RoadRunner-based map generation. CARLA further provides runtime configuration for switching traffic handedness, whereas AWSIM typically requires manual adjustments. Consequently, both simulators can represent Australian driving environments, with CARLA offering greater flexibility for traffic behavior experimentation.

Overall, this work provides a structured comparison framework that combines qualitative insights with quantitative scoring. While the current weighting favors end-to-end research workflows, the proposed criteria can be adapted to evaluate traditional modular pipelines by adjusting category priorities. As both simulators continue to evolve, the reported scores should be viewed as a time-specific benchmark rather than absolute rankings.

Future work will explore alternative weighting strategies that emphasize safety and robustness, extend evaluation to camera-based and LiDAR-based end-to-end models, and assess real-time deployment of modular autonomy stacks such as Autoware and Apollo.

References

[1] W. Shi, L. Liu, Autonomous driving simulators, in: W. Shi, L. Liu (Eds.), Computing Systems for Autonomous Driving,

Springer International Publishing, 2021, pp. 143–156. doi:10.1007/978-3-030-81564-6_6.
URL https://doi.org/10.1007/978-3-030-81564-6_6

[2] J. Seymour, D.-T.-C. Ho, Q.-H. Luu, An empirical testing of autonomous vehicle simulator system for urban driving, in: 2021 IEEE International Conference on Artificial Intelligence Testing (AITest), 2021, pp. 111–117. doi:10.1109/AITEST52744.2021.00031.
URL https://ieeexplore.ieee.org/document/9564102

[3] Y. Li, W. Yuan, S. Zhang, W. Yan, Q. Shen, C. Wang, M. Yang, Choose your simulator wisely: A review on open-source simulators for autonomous driving (2024) 1–19Conference Name: IEEE Transactions on Intelligent Vehicles. doi:10.1109/TIV.2024.3374044.
URL https://ieeexplore.ieee.org/document/10461065

[4] Simulation city: Introducing waymo's most advanced simulation system yet for autonomous driving.
URL https://waymo.com/blog/2021/07/simulation-city

[5] Cognata | AV & ADAS real data & simulation tools for AI.
URL https://www.cognata.com/

[6] AVSimulation | AVS : Automotive simulation software and solutions (2024).
URL https://www.avsimulation.com/en/

[7] Simcenter prescan software simulation platform.
URL https://plm.sw.siemens.com/en-US/simcenter/autonomous-vehicle-solutions/prescan/

[8] Mechanical simulation corporation.
URL https://www.carsim.com/

[9] CarMaker | IPG automotive.
URL https://www.ipg-automotive.com/en/products-solutions/software/carmaker/

[10] RoadRunner scenario.
URL https://au.mathworks.com/products/roadrunner-scenario.html

[11] NVIDIA DRIVE sim.
URL https://developer.nvidia.com/drive/simulation

[12] N. Koenig, A. Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, in: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Vol. 3, 2004, pp. 2149–2154 vol.3. doi:10.1109/IROS.2004.1389727.
URL https://ieeexplore.ieee.org/document/1389727

[13] O. Michel, Cyberbotics ltd. webots™: Professional mobile robot simulation 1 (1) (2004) 5, publisher: SAGE Publications. doi:10.5772/5618.
URL https://doi.org/10.5772/5618

[14] S. Shah, D. Dey, C. Lovett, A. Kapoor, AirSim: High-fidelity visual and physical simulation for autonomous vehicles (2017). arXiv:1705.05065[cs], doi:10.48550/arXiv.1705.05065.
URL http://arxiv.org/abs/1705.05065

[15] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: An open urban driving simulator (2017). arXiv:1711.03938[cs], doi:10.48550/arXiv.1711.03938.
URL http://arxiv.org/abs/1711.03938

[16] tier4/AWSIM, original-date: 2022-10-14T00:48:24Z (2024).
URL https://github.com/tier4/AWSIM

[17] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta, E. Agafonov, T. H. Kim, E. Sterner, K. Ushiroda, M. Reyes, D. Zelenkovsky, S. Kim, LGSVL simulator: A high fidelity simulator for autonomous driving, in: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020, pp. 1–6. doi:10.1109/ITSC45102.2020.9294422.
URL https://ieeexplore.ieee.org/abstract/document/9294422

[18] X. Hu, S. Li, T. Huang, B. Tang, R. Huai, L. Chen, How simulation helps autonomous driving: A survey of sim2real, digital twins, and parallel intelligence 9 (1) (2024) 593–612, conference Name: IEEE Transactions on Intelligent Vehicles. doi:10.1109/TIV.2023.3312777.
URL https://ieeexplore.ieee.org/document/10242366?denied=

[19] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, D. Batra, Sim2real predictivity: Does evaluation in simulation predict real-world performance? 5 (4) (2020) 6670–6677. arXiv:1912.06321[cs], doi:10.1109/LRA.2020.3013848.
URL http://arxiv.org/abs/1912.06321

[20] M. C. Figueiredo, R. J. F. Rossetti, R. A. M. Braga, L. P. Reis, An approach to simulate autonomous vehicles in urban traffic scenarios, in: 2009 12th International IEEE Conference on Intelligent Transportation Systems, 2009, pp. 1–6, ISSN: 2153-0017. doi:10.1109/ITSC.2009.5309524.
URL https://ieeexplore.ieee.org/document/5309524

[21] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng, ROS: an open-source robot operating system.

[22] N. Kalra, S. M. Paddock, Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?, in: Driving to Safety, How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?, RAND Corporation, 2016, pp. 1–16.
URL https://www.jstor.org/stable/10.7249/j.ctt1btc0xw.1

[23] J. Fadaie, The state of modeling, simulation, and data utilization within industry: An autonomous vehicles perspective (2019). arXiv:1910.06075[cs], doi:10.48550/arXiv.1910.06075.
URL http://arxiv.org/abs/1910.06075

[24] ApolloAuto/apollo, original-date: 2017-07-04T19:03:31Z (2025).
URL https://github.com/ApolloAuto/apollo

[25] autowarefoundation/autoware, original-date: 2015-08-24T23:17:57Z (2025).
URL https://github.com/autowarefoundation/autoware

[26] M. Violante, A COMPLETE END-TO-END SIMULATION FLOW FOR AUTONOMOUS DRIVING FRAMEWORKS.

[27] P. Kaur, S. Taghavi, Z. Tian, W. Shi, A survey on simulators for testing self-driving cars, in: 2021 Fourth International Conference on Connected and Autonomous Driving (MetroCAD), 2021, pp. 62–70. doi:10.1109/

MetroCAD51599.2021.00018.
URL https://ieeexplore.ieee.org/document/9499331

[28] J. Zhou, Y. Zhang, S. Guo, Y. Guo, A survey on autonomous driving system simulators, in: 2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2022, pp. 301–306. doi:10.1109/ISSREW55968.2022.00084.
URL https://ieeexplore.ieee.org/document/9985152

[29] L. Luttkus, L. Mikelsons, Key requirements for autonomous micromobility vehicle simulators, in: 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2023, pp. 1506–1511, ISSN: 2577-1655. doi:10.1109/SMC53992.2023.10394587.
URL https://ieeexplore.ieee.org/abstract/document/10394587

[30] A. Stocco, B. Pulfer, P. Tonella, Mind the gap! a study on the transferability of virtual versus physical-world testing of autonomous driving systems 49 (4) (2023) 1928–1940, conference Name: IEEE Transactions on Software Engineering. doi:10.1109/TSE.2022.3202311.
URL https://ieeexplore.ieee.org/abstract/document/9869302

[31] H. Haghighi, M. Dianati, V. Donzella, K. Debattista, A unified generative framework for realistic lidar simulation in autonomous driving systems (2024). arXiv:2312.15817[cs], doi:10.48550/arXiv.2312.15817.
URL http://arxiv.org/abs/2312.15817

[32] A. AbdelHamed, G. Tewolde, J. Kwon, Simulation framework for development and testing of autonomous vehicles, in: 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), 2020, pp. 1–6. doi:10.1109/IEMTRONICS51293.2020.9216334.
URL https://ieeexplore.ieee.org/abstract/document/9216334

[33] M. Althoff, S. Urban, M. Koschi, Automatic conversion of road networks from OpenDRIVE to lanelets, 2018.
URL https://mediatum.ub.tum.de/1449005

8. Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the author(s) used Grammarly Pro AI and ChatGPT in order to improve writing. After using this tool, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

9. CRediT authorship contribution statement

Zhihui Lai: Conceptualization, Methodology, Software, Writing - Original Draft. Lee Le: Software, Writing - Original Draft. Vihanga Silva: Conceptualization, Software. Thomas Bräunl: Conceptualization, Writing - Review & Editing, Supervision.

10. Appendices (Unpublished works)

The graphical user interfaces (GUIs) of RoadRunner, UE4, and Unity are illustrated in Fig. 4, Fig. 5, and Fig. 6, respectively.

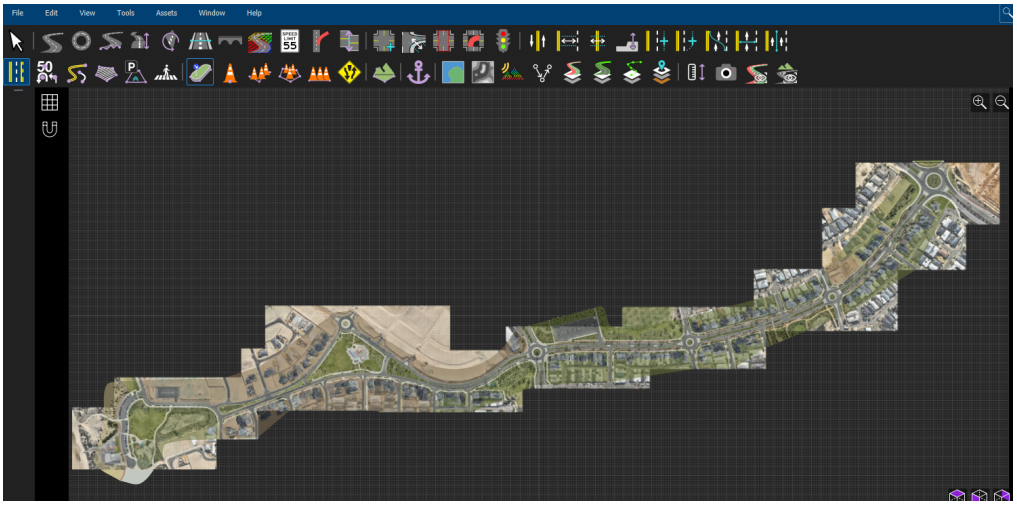


Figure 4: GUI of RoadRunner.

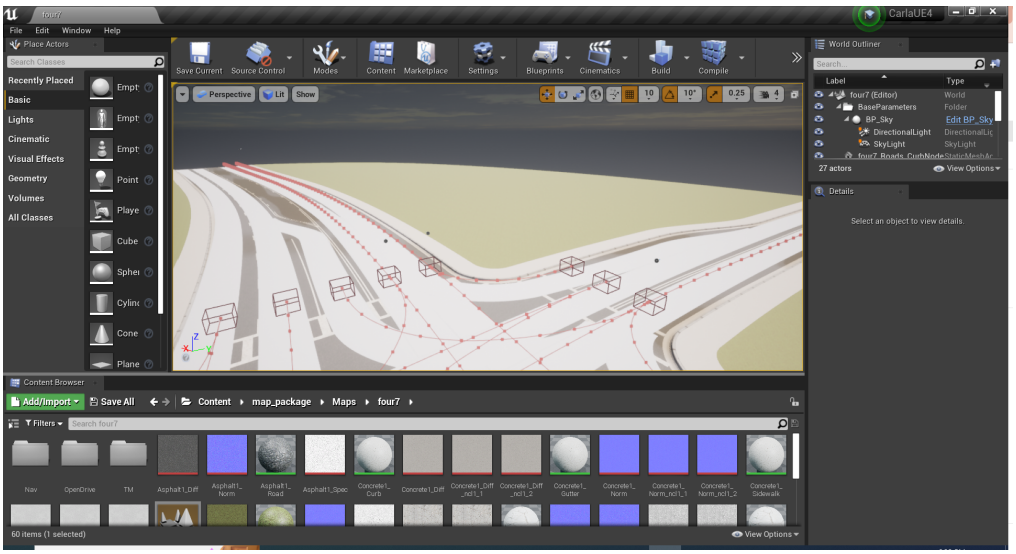


Figure 5: GUI of UE4.

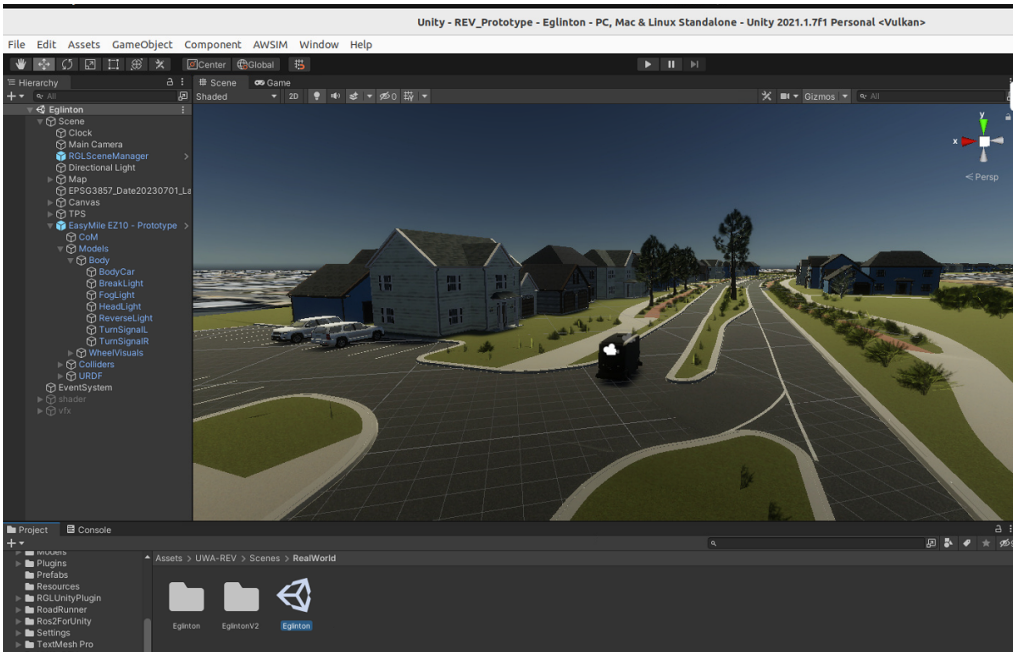


Figure 6: GUI of Unity.

Chapter 5

Shuttle Bus Performing Common Maneuvers in Roundabout-Style Suburban Residential Area Using Grayscale Camera-Based End-to-End Driving

Preamble: *Guided by the simulator study and the need for maneuver-specific competence observed in prior chapters, we develop a grayscale, camera-only MoE for real suburban routes. The system combines lane following/pull-out, pull-in (with an empty-bay classifier), and reverse experts on a dual-computer ROS 2 platform, and is evaluated in both open-loop and closed-loop trials on a 4.1 km loop. This chapter shows how expert specialization plus simple gating yields a practical autonomy/safety/latency balance, and it distills design lessons—data curation, latency control, and interpretability. This chapter directly addresses* ***RQ1****,* ***RQ2****, and* ***RQ4****, and also discusses the limitations of generalising safety claims beyond the tested conditions.*

Publication: *Lai, Z., Quirke-Brown, K., Kong, X., Le, L., Bräunl, T. Shuttle Bus Performing Common Maneuvers in Roundabout-Style Suburban Residential Area Using Grayscale Camera-Based End-to-End Driving. IEEE Transactions on Intelligent Vehicles (under review, submitted on 11 Sep 2025)*

Shuttle Bus Performing Common Maneuvers in Roundabout-Style Suburban Residential Area Using Grayscale Camera-Based End-to-End Driving

Zhihui Lai, Kieran Quirke-Brown, Xiangrui Kong, Lee Le, Thomas Bräunl, *Senior Member, IEEE*

***Abstract*—Interest in autonomous vehicles has grown exponentially in recent years. While the traditional mediated perceptron approach is a proven technology for autonomous driving, the end-to-end learning method has gained popularity due to its potential to reduce time, labor, and costs. Although many end-to-end methods have emerged recently, most require substantial data, complex neural network architecture, expensive equipment, and lengthy training periods. This paper presents an end-to-end Mixture of Experts (MoE) autonomous driving system for a shuttle bus operating in a roundabout-style suburban residential environment. Our system leverages front and rear grayscale cameras to execute a set of common driving maneuvers using deep learning models with minimal costs, reduced training times, and limited real-world driving data. Experimental results show successful automatic switching between expert navigation models through roundabouts, intersections, and parallel parking scenarios with high accuracy. The paper involves optimizing neural network architecture, refining data classification algorithms, enhancing data augmentation techniques, and evaluating different degrees of fusion among multiple sub-neural network models. More information is on our project website https://therevproject.com/vehicles/nuway.php**

***Index Terms*—End-to-End Driving, Grayscale Camera, Autonomous Shuttle Bus, Roundabout, Suburban Navigation, Sensor Fusion, Deep learning, Autonomous parallel parking.**

I. Introduction

AUTONOMOUS Vehicles (AVs) and Advanced Driver Assistance Systems (ADAS) have garnered significant attention from academia and industry alike. The U.S. National Highway Traffic Safety Administration (NHTSA) reports that 94% of severe traffic accidents are caused by human error [1], motivating the development of automation technologies to enhance safety. However, real-world deployment of AVs remains limited due to high costs, complex system requirements, and the challenge of outperforming human drivers while maintaining affordability.

Autonomous shuttle buses present a promising solution for last-mile mobility in suburban environments. The Renewable Energy Vehicle (REV) Project at The University of Western Australia (UWA) has developed and deployed a fleet of autonomous shuttle buses—nUWAy—to address limitations in conventional public transport systems. These electric vehicles are designed to fill first-mile and last-mile gaps with flexible, low-speed, on-demand transport. By repurposing EasyMile EZ10 GEN 1 shuttles and integrating them with open-source and proprietary control software, the project has established a testbed for campus mobility, public road trials, and advanced research on autonomy.

The Society of Automotive Engineers (SAE) defines six levels of driving automation, with Level 5 indicating full autonomy. Most commercial systems, such as those by Waymo and Tesla, fall below Level 5 and rely on either expensive LiDAR-based sensor fusion or monocular camera-based perception with limited generalization.

Traditional autonomous driving stacks follow a mediated perception approach, dividing the system into modular components such as perception, planning, and control. While interpretable and tunable, these systems are engineering-intensive, susceptible to error propagation, and require large-scale annotated datasets. In contrast, end-to-end methods learn a direct mapping from raw sensor input (e.g., camera or LiDAR) to driving commands via neural networks. This approach reduces system complexity, hardware requirements, and computational burden, though at the cost of reduced interpretability.

Popular end-to-end models, such as NVIDIA's PilotNet [2] and Conditional Imitation Learning (CIL) [3], have demonstrated the feasibility of learning driving behavior directly from data.

In this work, we propose a Mixture-of-Experts (MoE) framework for grayscale camera-based end-to-end autonomous driving. Designed for an electric shuttle bus operating in a roundabout-style suburban residential area, our system performs common maneuvers such as roundabout navigation, intersection handling, and parallel parking. The MoE architecture enables efficient rule-based expert model switching conditioned on the driving context, allowing for robust maneuver execution with limited training data and reduced hardware complexity.

We evaluate and select the optimal expert combination in terms of safety, smoothness, and maneuver completion. Our findings illustrate the promise of end-to-end MoE frameworks for developing lightweight and cost-effective autonomous driving solutions in fixed environments characterized by weak localization signals.

II. Related Work

A. Sensor Modalities in Autonomous Driving

Autonomous driving systems are traditionally categorized by their reliance on specific sensor configurations. LiDAR-

Manuscript received September 11, 2025; revised September 11, 2025. This work was supported by Stockland

Authors are with The UWA Renewable Energy Vehicle Project (REV) team, The University of Western Australia, 6009 Crawley, Australia (e-mail: zhihui.lai@research.uwa.edu.au).

centric systems, such as those used by Waymo, provide accurate localization and robust obstacle detection but are expensive and depend on richly annotated high-definition (HD) maps. Camera-based approaches, exemplified by Tesla, are more cost-effective and scalable but often struggle with robustness in varied lighting and weather conditions. Hybrid systems, as seen in Mobileye's "true redundancy" approach, seek to integrate independent LiDAR and camera pathways to increase safety and flexibility, but remain largely proprietary and under active development.

B. Limitations of Traditional Pipelines

Traditional autonomous driving systems, including those deployed on shuttles, typically adopt a modular architecture composed of discrete components for perception, localization, planning, and control. These systems often rely on high-precision LiDAR localization and Global Navigation Satellite System - Real Time Kinematic (GNSS-RTK) positioning, requiring multiple 3D LiDARs, differential GPS antennas, and a full autonomy software stack such as Autoware. While this mediated perception pipeline offers interpretability and robust performance, it introduces significant cost, integration overhead, and demands extensive engineering effort, large annotated datasets, and substantial computational resources—challenges that limit scalability in constrained or cost-sensitive environments [4, 5, 6].

C. Minimal-Sensor End-to-End Driving

Recent advances in deep learning have motivated a shift toward end-to-end architectures that learn control policies directly from raw camera data, potentially reducing hardware requirements and eliminating the need for detailed semantic maps or precise localization [4, 5, 6]. In this study, we adopt a camera-only setup to evaluate the feasibility of such models in suburban shuttle applications with limited sensor input.

D. End-to-End Driving Architectures

End-to-end driving approaches aim to directly learn control policies from raw sensory inputs, bypassing the complexity and modularity of traditional pipelines. Early systems such as ALVINN [7] and NVIDIA's PilotNet [2] demonstrated the feasibility of mapping front-facing camera images directly to steering commands using convolutional neural networks. However, these early models employed monolithic architectures that lack task-specific modularity and tend to underperform in complex or multi-maneuver driving scenarios.

To address maneuver ambiguity, Conditional Imitation Learning (CIL) [3] introduced high-level command conditioning, enabling shared visual features to be routed through task-specific output heads. Further advances such as FCN-LSTM [8], Deep-Steering [9], and C-LSTM [10] incorporated temporal sequences to improve prediction consistency, while multi-modal frameworks including Drive360 [11] and IMF-DNN [12] extended the input space to include route planning information and LiDAR signals. Angle-based command conditioning [13] further extended CIL by replacing discrete commands with continuous subgoal angles, enabling smoother transitions and improved maneuver disambiguation.

Other notable architectures focus on future trajectory prediction rather than low-level actuation. Models such as PilotNet-2020 [14] and AdmiralNet [15] generate planned motion paths, which can offer smoother control and improved interpretability. Additionally, auxiliary supervision techniques such as semantic segmentation [8, 16] or affordance prediction [17, 18] provide richer learning signals, but often require additional labeled data and introduce architectural complexity. Kendall *et al.* [19] demonstrated that vision-based end-to-end driving policies can be learned rapidly from real-world interaction using a combination of imitation learning and reinforcement learning, achieving route-level autonomy within a single day of training. While this significantly reduces manual labeling effort and development time, such approaches still depend on large volumes of high-frequency sensor data, online interaction, and substantial onboard computation, which can limit their applicability to low-cost or resource-constrained platforms.

Despite significant progress, multiple studies have highlighted persistent limitations in end-to-end behavior cloning. Codevilla *et al.* [20] show that imitation-based models tend to overfit and fail to generalize to dynamic, multi-maneuver settings. Ishihara *et al.* [21] report that conditional end-to-end models struggle with rare events such as traffic lights without explicit multi-task supervision. Survey work by Chen *et al.* [5] further confirms that most current end-to-end models lack maneuver-specific specialization, particularly in intersections and roundabouts, motivating the need for expert-based or gating architectures.

E. MoE for Camera-Based Driving

MoE architectures (as a popular ensemble learning method) have shown promise in enhancing the adaptability and robustness of end-to-end autonomous driving systems. By decomposing complex driving tasks into specialized expert modules, MoE frameworks can dynamically select the most appropriate expert based on real-time input, leading to improved performance in diverse driving scenarios. For example, Kim *et al.* proposed a Mixture of Domain-specific Experts (MoDE) that disentangles domain-general from domain-specific representations, significantly improving closed-loop success rates on CARLA benchmarks [22]. More recently, large-scale MoE systems like DriveMoE integrate Scene-Specialized Vision and Skill-Specialized Action routers in vision-language pipelines, achieving state-of-the-art performance on Bench2Drive simulations [23]. Another notable example is ARTEMIS, which performs autoregressive trajectory planning using expert-specialized modules. It generates sequential waypoints and dynamically routes to scene-specific experts, mitigating performance degradation in ambiguous guidance scenarios. ARTEMIS records strong closed-loop performance on the NAVSIM dataset using a ResNet-34 backbone [24].

Although these studies enhance model performance through expert specialization, they rely on complex sensing—such

as multi-view cameras, LiDAR, or trajectory prediction—and large-scale architectures. In contrast, our approach employs a compact, camera-only MoE tailored for real-world shuttle deployment, using grayscale images and a simple rule-based gating mechanism, not data-driven routers, to switch between three expert models. This design maintains both model interpretability and practical deployability, while delivering robust performance across common suburban-driving maneuvers.

III. System Overview

A. Vehicle Platform and Hardware Setup

Our primary vehicle platform is the EZ10 Gen 1 shuttle (4050 mm × 1892 mm × 2871 mm, Ackermann steering angle: 18°, max speed: 20 km/h), retrofitted with additional sensing and computing systems (see Fig. 1 (a) (c) (e) and Table I).

TABLE I: Hardware Configuration Summary

Component	Details
Cameras	FLIR Grasshopper3 (Front/Rear), 960×480 @ 24 FPS, Spinnaker SDK
IMU + GNSS	SBG Ellipse-D RTK-GNSS with IMU
LiDAR	4 SICK LMS-151 (safety), 2 Velodyne VLP-16 (3D), 2 SICK LD-MRS (long-range)
Power	Dual 48V $LiFePO_4$ (7.2kWh), 3 bus bars (48V, 24V, 12V), range around 40 km
Control	CAN-based traction/steering, hydraulic/emergency brakes
Charging	IEC Type-2 with TBS Pro monitoring
Safety	PLC with perimeter-based emergency stop

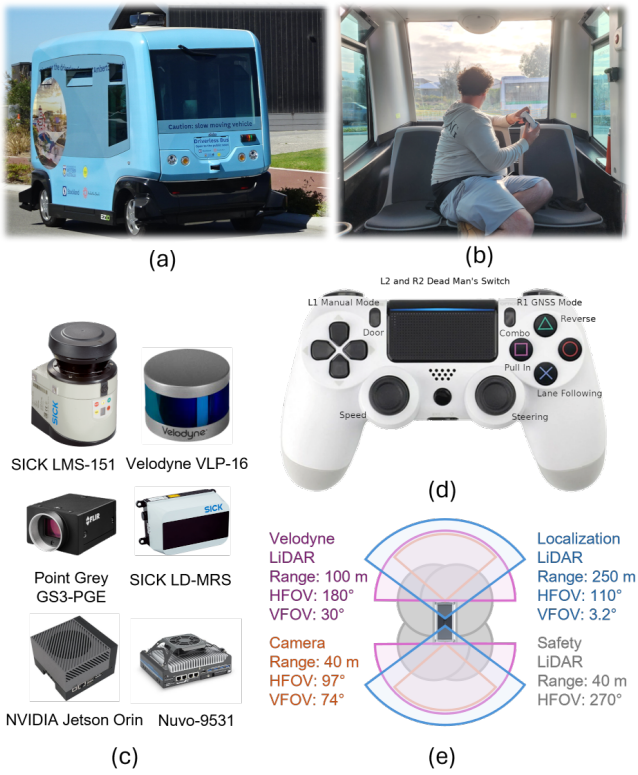


Fig. 1: nUWAy Shuttle Configuration: (a) nUWAy outside view, (b) nUWAy inside view, (c) sensors and PCs, (d) PS4, (e) sensor range

B. Computing Architecture and Software Stack

The shuttle uses a dual computer configuration to manage the computational load. Both systems communicate over an Ethernet network, and one PC can communicate over a shared Controller Area Network (CAN) network. Safety is enforced through a manufacturer-supplied Programmable Logic Controller (PLC), which independently triggers emergency stops based on LiDAR curtain inputs. The software stacks are built on ROS 2 Humble, locally or in Docker containers, integrating open source packages and custom modules for end-to-end navigation, shuttle bus manipulation, safety curtaining, and data visualization.

1) Main Computer: An industrial x86 PC (Nuvo-9531) for CAN bus communication, safety monitoring, and auxiliary I/O. The modules include:

a) Low-level module: The low-level module is designed to handle low-level communications with the SBG Ellipse-D combo IMU and GNSS+RTK mounted on the vehicle ceiling. The main pc gets precision GPS readings from the SBG device over serial USB communications and passes back RTK corrections from an RTK2GO NTRIP server. The low-level module is also responsible for displaying the vehicle's state and position on the map.

b) RViz: RViz is used for live visualizations of the system as shown in Fig. 2. It provides comprehensive details on the vehicle's speed, steering, steering mode (dual, front, or rear steering), error codes, and driving mode (i.e., NN lane following, manual, etc.). In addition, the display also shows

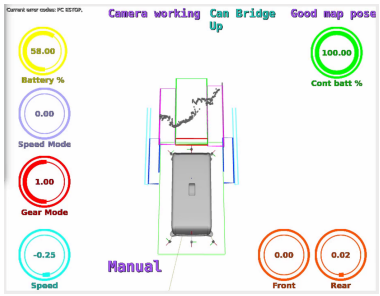


Fig. 2: nUWAy RViz Display

point cloud data that exists within the slow/stop node range, which is explained below, as well as the current state of the important nodes for driving. The three key nodes monitored on the display are cameras, the CAN bridge, and the map pose. If any of these nodes should fail, then the system will either lock out a particular driving mode (vision-based driving or RTK + GNSS driving), or lock out all driving if the low-level communications fail.

c) NAV2 Stack: The NAV2 stack comes from [25] and provides additional features to the existing ROS2 implementation. Currently, the NAV2 stack is used for deterministic control methods setting up global paths and implementing local control, which, based on previous work [26], used a regulated pure pursuit controller [27].

d) GNSS+RTK Navigation: The autonomous shuttle bus is fitted with an SBG Ellipse-D dual antenna IMU+GNSS device that can receive RTK corrections. The RTK corrections were sent to the autonomous shuttle bus via an IP-based NTRIP client using a cellular sim modem. During the initial testing phase it was found that the accuracy of the GNSS would be around 1 to 2 centimeters but there were periodic spikes where the accuracy would drop to greater than 2 meters. This was attributed to cellular dead zones and satellite interference from cloud cover, therefore AI methods, detailed below, were explored.

e) CAN Bridge: The PC communicates with low-level controllers using a CAN bus system. The CAN bridge node reads and sends commands directly to the CAN bus system. The messages are used to control the speed, acceleration and turning of the vehicle as well as some peripherals like door and bell operations. Messages read into the PC are used to check the ESTOP status, noting the different causes of potential ESTOPS, as well as speed and steering feedback to confirm correct operations. The node is designed to be lightweight, as heavy communications to the system can disrupt heartbeat operations between the PC and the PLC, resulting in an ESTOP.

f) Slow and Stop Node: As autonomous driving algorithms cannot always guarantee correct and safe movements when driving on the road, a number of low-level safety measures have been implemented to guide the vehicle safely along the route. We encapsulate these low-level safeties into a node called slow/stop node based on the actions it can perform. The space around the vehicle is broken up into a number of regions as shown in Fig. 2. Each region either behaves in a different way or creates a different action based on the most critical need of the vehicle. To start with, the region in front of the vehicle is used to slow it down as it approaches other vehicles or the vehicle is driving towards a static object. The region is further subdivided into a slow region and a stop region. This is because a failure to stop in time can result from a low level estop from the PLC controllers which then requires the system to be rearmed which can take upward of 30 seconds. In the slow sub-region the vehicle speed is reduced by how close the object in front of it is until it reaches the stop region or is no longer present. If the stop subregion is reached, then the vehicle will reduce its speed to 0 (without emergency brakes), but will not require a system rearm so the vehicle can proceed when the object is cleared.

g) PlayStation 4 (PS4) Controller: Currently, the selection process between the MoE models is performed using a PlayStation controller. As seen in Fig. 1 (b) (d), the shape buttons are used to select between vision-based models, the ”R1” button is used to run the GPS mode, and L1 is used to drive manually. For any driving to occur, a dead-man switch must be held in which an onboard safety officer can quickly stop the vehicle if necessary. Both the ”L2” and ”R2” buttons can be used as a dead-man switch, but the ”R2” additionally engages a higher speed mode (from 4.0 to 5.4 m/s) when the roads are more maneuverable. Manual control of the vehicle is achieved using the left and right paddles; the left paddle controls speed, and the right paddle controls steering.

h) Incident Recorder: In the event of an incident, a snapshot of all data from the past 5 seconds is collected into a ROSbag for post-drive analysis. An incident can occur for a number of reasons, but the primary reason is a safety LiDAR trip due to obstacles that come too close to the vehicle. The distance for this trigger varies depending on the speed of the shuttle bus. Other reasons for incidents are invalid commands to the PLC controllers, sensor failure (for sensors linked to the CAN bus system) and heartbeat failure between the on-board PC and the PLCs.

2) AI Supercomputer: An NVIDIA Jetson AGX Orin accelerates real-time inference for neural network-based control decisions. The modules include:

a) Regular Recorder: A subset of the messages in the ROS2 network are stored in an mcap ROSbag, which condenses the data. The topics chosen for recording are selected for training and evaluation purposes. The mcap files are converted to numpy files in post-processing but are also used with Foxglove studio [28] for quick replay and analyses to identify immediate issues.

b) Image Publisher: Sets the configuration of the two FLIR cameras and publishes the front and rear camera images at a rate of 20 Hz.

c) Camera-based NN Command Publisher: The camera-based NN command publisher consists of three navigation models, two decision-making expert models, and a gating mechanism to switch experts in real-time. Each navigation model consists of a fused set of behaviors as follows:

- **Navigation Model Cmd 0:** Lane following, pull-out from bays, straight passing along bays, straight at roundabouts and intersections.
- **Navigation Model Cmd 1:** Lane following, pull-in to bays, turnaround to the office at roundabouts and intersections.
- **Navigation Model Cmd 2:** Lane following, reverse in bays, turnaround to the beach at roundabouts and intersections.

The decision of which expert to use at any given point is made through manual control as specified before; however, certain flags and states can result in automatic triggers of state change as detailed:

- **Manual Switching:** Press triangle (Cmd 2), square (Cmd 1), or cross (Cmd 0) on PS4 controller.
- **Empty Bay Classifier:** Detect an empty bay as a pull-in trigger when a stop has been requested (up on the arrow keys).
- **Auto Switching:** While in lane following mode, the stop station request (up arrow on the controller) can be used to flag a mode change. The empty bay classifier will look for a car park with two empty bays available and when detected with a probability of 90%, or above, for 3 consecutive frames, the pull-in mode (Cmd 1) is triggered. After successfully completing a pull-in maneuver, the shuttle will stop at the end of the bay for 2 seconds before switching to reverse (Cmd 2). After reversing in the bay, the shuttle will again stop and wait 2 seconds before switching into pull-out mode (Cmd 0).

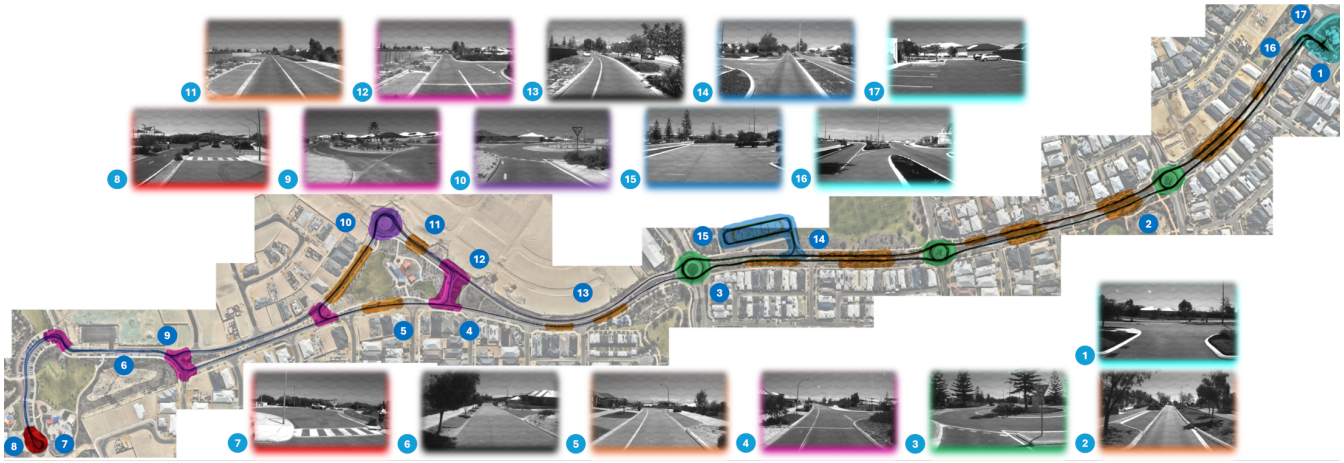


Fig. 3: Eglinton map with stage 1 automated sorting filters of driving behaviors highlighted in colors, including standard carpark (blue), right-turn roundabout (purple), standard roundabout (green), office carpark (cyan), cul-de-sac (red), intersection (pink), parallel parking (orange); and scenario images of driving behaviors from the start (Stockland Sales Centre) to the end (Amberton Beach Bar) numbered 1 - 17

IV. Methodology

A. Environment

As demonstrated in Fig. 3, the shuttle operates along a fixed 4.1 km loop in Amberton Beach, Eglinton, WA. Road types include single lane, two-way lane, cul-de-sac, roundabout, and parallel parking. Route characteristics are shown in Table II. Real-world trials involve repeated autonomous drives through residential zones and semi-structured environments, testing neural network models under variable conditions.

TABLE II: Route Characteristics

Criteria	Specification
Route Length	East–West: 2.0 km; West–East: 2.1 km
Destinations	Amberton Beach Bar, Stockland Sales Centre
Lane Width / Gradient	> 3.0 m / $< 2.3\%$
Network Coverage	4G (Three telecom providers)
Intersections	East–West: 3 roundabouts, 3 T-intersections; West–East: 4 roundabouts, 3 T-intersections
Parallel Parking Zones	East–West: 15 left, 2 right; West–East: 14 left, 4 right
Pedestrian Crossings	4 per route near Amberton Beach Bar
Speed Limit	50 km/h

B. Data Collection and Preprocessing

Our on-road data collection began in March 2023; however, only data collected after November 2023 was used for model training, due to a camera repositioning that affected image alignment. From November 2023 to January 2024, we recorded data using only the front camera, as running both front and rear cameras simultaneously caused stability issues. Starting February 2024, we successfully collected dual-camera data (front and rear) and concluded the data collection in February 2025 after obtaining sufficient data.

In total, the dataset spans from November 2023 to February 2025 and comprises approximately 600 GB of data, covering around 40 hours of driving. After filtering and sorting for quality and consistency, about half of this data was deemed suitable for training.

Data was recorded on an NVIDIA Orin platform using a rosbag recorder in MCAP format. The recorded topics include: front and rear camera images (20 Hz), front and rear Velodyne point clouds (20 Hz), CAN bus data (100 Hz), driving commands (50 Hz), driving mode, speed mode, gear position, GNSS, and IMU.

We developed a preprocessing script to extract data into NumPy arrays formatted as:

[Front image, Rear image, Linear velocity, Angular velocity, Driving mode, Gear position, Speed mode, Map position, Orientation, Scene category]

The timestamp of the front image is used to synchronize all other topics.

The Scene category field was manually labeled with the following classes: 0 for *unknown*, 1 for *pull-in trigger*, and 2 for *reverse and pull-out trigger*. These labels were applied throughout the dataset to support specialized classifier training.

The image shape undergoes several transformations: originally recorded at (832×520) pixels, we first crop 20% from the top, reducing it to (832×416). The image is then resized to (480×240) in the raw NumPy files. To simulate camera motion or augment the data, we apply horizontal shifting or rotation, yielding an intermediate resolution of (400×240). Finally, we crop an additional 25% from the top—removing the sky—resulting in a final input size of (400×180) used in the training batch generator.

C. Data Sorting

We sort and clean npy files in different behavior folders using both automatic sorting (based on map position) and manual sorting (correcting or removing bad behaviors). This multi-stage heuristic approach enables efficient and accurate segmentation of behavior-specific driving data, significantly reducing manual annotation effort and improving the structure of training datasets.

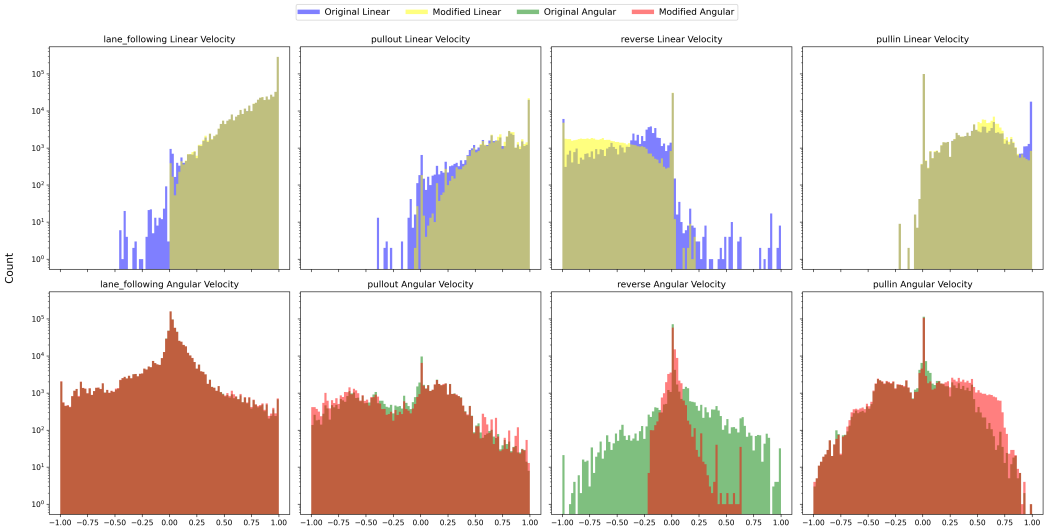


Fig. 4: Linear and angular distribution of original and modified data after manual annotation in driving behaviors, including lane following, pull-in, pull-out, and reverse

1) Stage 1—Automated Sorting of Driving Behaviors: The stage 1 method extracts meaningful trajectory segments from raw data using spatial binary filters (see Fig. 3 and Table III). By comparing consecutive vehicle positions on the map with region-specific masks, we detect entry and exit events into predefined zones. Velocity data between these events is buffered, and segments longer than 100 frames are saved into corresponding folders for the next stage.

2) Stage 2—Automated Sorting, Manual Sorting, and Manual Annotation of Driving Behaviors:

a) Automated Sorting: The stage 2 automated sorting method performs automatic categorization of previously filtered trajectory segments by identifying distinct driving behaviors through heuristic rules.

b) Manual Sorting: The stage 2 manual sorting method performs manual categorization of driving behaviors that can't be sorted or misclassified by the automated sorting method.

c) Manual Annotation: The stage 2 manual annotation method makes imperfect data caused by manual driving more suitable for neural network learning through correction, replication, and interpolation. Specifically, we have smoothed out some abrupt turns and the stopping process during pull-in and reverse maneuvers, while also improving the startup phases in pull-in, reverse, and pull-out actions (instead of gradually increasing from zero to the target linear and angular velocity, it will now initiate with the target commands). The linear and angular distribution of the data before and after the manual annotation is depicted in Fig. 4.

D. Data Offset

During our initial data collection, the camera driver (PySpin, the Python wrapper for Spinnaker) was operating with its default configuration, which introduced a significant camera latency of approximately 0.436 seconds. Unfortunately, this issue was only solved after all training data had been collected. Upon further investigation, we identified a PySpin parameter setting that allowed us to reduce the latency to 0.123 seconds. Consequently, a temporal offset of 0.313 seconds—equivalent to approximately 6 frames at 20 Hz—was applied to all training data to align the sensor streams correctly.

In early testing of our neural network models using the high-latency dataset, the system could only operate reliably at 6 Hz. Attempting to run the models at higher frequencies (e.g., 20 Hz) led to immediate instability and crashes. After reducing the camera latency and retraining the models on a corrected dataset without the high-latency offset, the models were able to run stably at 20 Hz.

E. Data Distribution

The lane following intensity is depicted in Fig. 5. The pull-in, reverse, pull-out frame counts are shown in Table XI. The roundabout/intersection turnaround to office/beach frame counts are shown in Table XII.

F. Data Augmentation

To improve the robustness of the model against visual variations and label noise, several data augmentation techniques were applied in the batch generator during training (see Fig. 6). Each augmentation is randomly applied to 5% of the training data in a single batch:

- **Blurring:** Random kernel-size in [2, 5] uniform blurring simulates motion blur and sensor noise. Blurring helps the model learn more holistic features without overlearning the minutiae.
- **Brightness Adjustment:** Brightness is scaled multiplicatively by a random factor in [0.5, 1.5] to mimic lighting changes.
- **Gaussian Noise:** Additive white Gaussian noise in [0, 0.1] simulates sensor-level distortions.
- **Shadow Overlay:** A random polygonal region of the image is darkened in [0.5, 1] to simulate cast shadows.

TABLE III: Hierarchical mapping from stage 1 (filtered segments) to stage 2 behaviors, along with definitions and frame counts (in thousands). S: Single-camera data, D: Dual-camera data

Stage 1 Category	Stage 2 Behavior Definitions
Cul-de-sac	**Cul-de-sac Dualsteering (S: 0.0, D: 14.5):** Navigating a dead-end loop in dual steering mode. **Cul-de-sac Lane Following (Merged with Main Lane following):** Navigating a dead-end loop in front steering mode.
Intersection	**Intersection Lane Following (S: 0.0, D: 57.2):** Driving through intersections following route. **Intersection Turnaround to Beach (S: 0.0, D: 0.0):** U-turn at an intersection heading to the beach. **Intersection Turnaround to Office (S: 0.0, D: 7.5):** U-turn at an intersection heading to the office.
Main	**Lane Following (S: 107.1, D: 294.7):** Remaining data not covered by other behaviors. **Others (S: 496.0, D: 1529.3):** Miscellaneous, bad behaviors, or uncategorized driving scenarios.
Parallel Parking	**Pull-in (S: 43.2, D: 41.9):** Entering bays. **Pull-out (S: 22.6, D: 46.9):** Exiting bays. **Reverse (S: 33.9, D: 35.1):** Driving backwards in bays. **Carpark Pass (S: 0.0, D: 73.5):** Passing occupied bays. **Empty Bay (1st/2nd Half) (S: 9.8, D: 53.1):** Passing empty bays. **Pull-in Trigger (S: 5.8, D: 22.7):** Speed/steering adjusted from Empty Bay for better pull-in. **Pull-in Stops (S: 16.6, D: 82.0):** Stops after pull-in. **Pull-out Stops (S: 22.0, D: 8.4):** Stops before pull-out.
Right-turn Roundabout	**Roundabout Right Turn (S: 0.0, D: 35.1):** Taking the third (rightmost) exit in a roundabout.
Office Carpark	**Shed In (S: 0.0, D: 32.3):** Entering the shed. **Shed Out (S: 0.0, D: 0.0):** Exiting the shed. **Startpoint In (S: 8.3, D: 10.6):** Entry from initial point. **Startpoint Out (S: 3.8, D: 10.1):** Exit to main road.
Standard Carpark	**Carpark Entry (S: 8.1, D: 4.0):** Entering the carpark. **Carpark Pass (S: 0.0, D: 73.5):** Passing inside the carpark. **Carpark Dualsteering (S: 46.1, D: 121.4):** Dual steering maneuvers inside carpark.
Standard Roundabout	**Roundabout Straight (S: 23.2, D: 62.9):** Going straight through a roundabout. **Roundabout Turnaround to Beach (S: 6.1, D: 42.8):** Full loop to redirect to the beach. **Roundabout Turnaround to Office (S: 20.4, D: 18.2):** Full loop to redirect to the office.

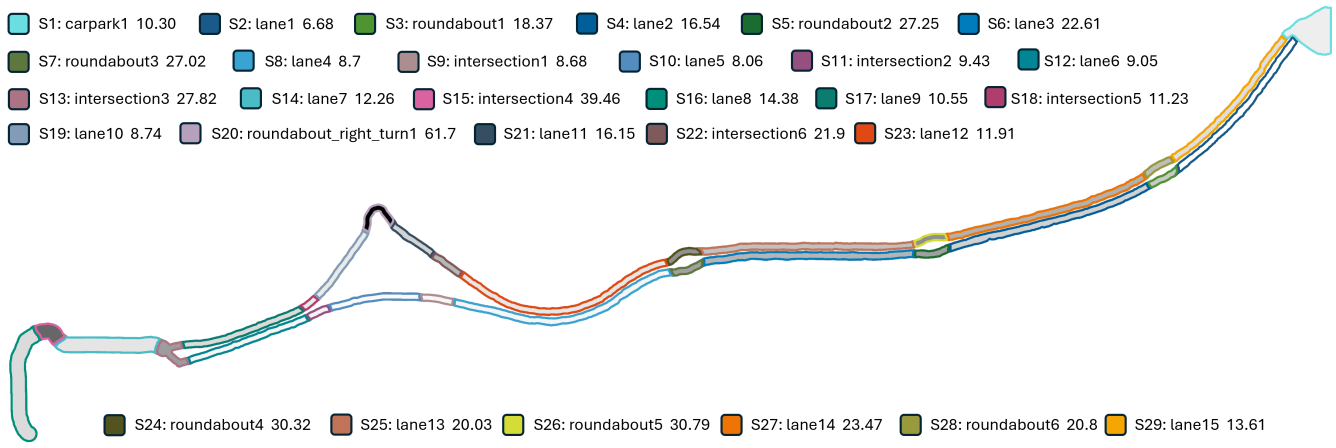


Fig. 5: Lane following data distribution: Sections are outlined by unique colors, numbered office → beach → office, and filled with a grayscale gradient (white → black) representing intensity (frame count per 100 pixels)

- **Horizontal Shift:** Steering is adjusted linearly based on the pixel shift (40 pixels on each side, pairing with up to 20% of the maximum steering angle), increasing the diversity of lateral lane positions.
- **Horizontal Rotation:** Steering is adjusted linearly based on the perspective warp and pixel shift (5° of rotation and 40 pixels on each side, pairing with up to 15% of the maximum steering angle), increasing the diversity of lateral lane positions.

G. Selection of Inputs/Outputs, Model Design, and Training/Testing Conditions

This section outlines our choices and rationale for model inputs, outputs, architectures, and experimental setup.

- **Pull-in Trigger via Classifier:** We use a lightweight classifier to detect empty parking bays, avoiding the complexity and annotation cost of object detection.
- **Direct Control Outputs:** Our model predicts steering and speed directly—eschewing waypoints—because we lack high-accuracy localization and because direct control has been shown to match or exceed trajectory approaches in end-to-end driving [29].
- **Separate Expert Models:** Each behavior expert is implemented as a standalone model—each with its own encoder and decoder—rather than sharing a vision backbone across behaviors. This avoids shared latent interference common in branched networks [3] and improves specialization.
- **Grayscale Monocular Input:** We leverage the shuttle's existing grayscale camera, eliminating the need for mechanical redesign and reducing computational load. Recent studies show grayscale models offer comparable performance with faster inference due to single-channel input [30]. We use monocular cameras, not stereo cameras, since humans can drive with one eye only, and many monocular vision-only models have achieved good performance [4].
- **No Ego-State or Temporal Stack:** Only the current camera frame is input; no vehicle state or image history is used. Prior work indicates that including ego feedback may introduce undesirable inertia in behavior cloning [20], while temporal stacking risks causal confu-

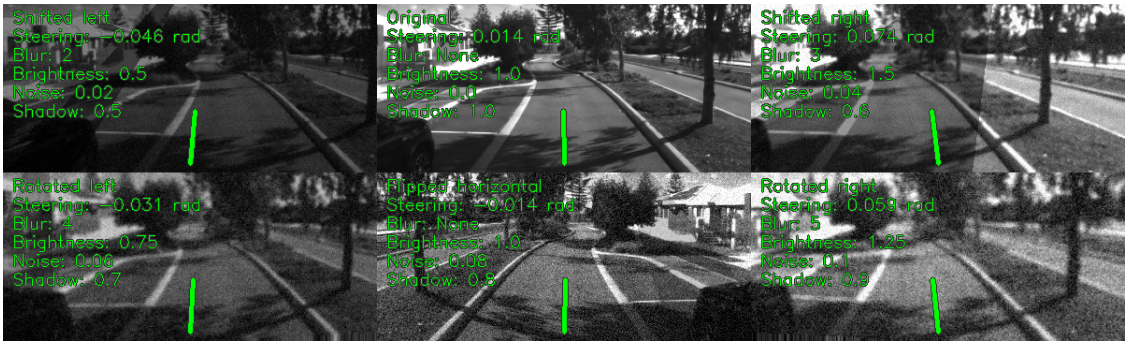


Fig. 6: Images with different augmentation levels

sion in time-varying environments [5].

- **Loss Function—MAE:** Models are trained with Mean Absolute Error (MAE), which better respects bounded [-1,1] command ranges and handles outliers more robustly than MSE [4].
- **Vision-Only Setup:** Omitting depth, LiDAR, or multi-sensor fusion to maintain system efficiency and employability. Multi-sensor integration involves considerable computational expenses and does not assure effective information extraction [5].
- **Data Balancing via sampling:** To address behavioral imbalance, we apply down-sampling or up-sampling across maneuver datasets, mitigating the long-tail distribution often observed in driving tasks [4, 5].
- **Domain-Specific Evaluation:** Acknowledge that end-to-end models struggle with full generalization [4, 5], so we focus on domain-specific validation under fixed conditions: daytime, clear weather, and our target suburban scenario. This approach yields strong performance within deployment constraints.

H. Model Architecture

We investigate a variety of expert model architectures and behavior fusion strategies to identify an optimal configuration. Our evaluation includes PilotNet, MobileNetV3 [31], and EfficientNetV2—where MobileNetV3 and EfficientNetV2 are selected from the Keras Applications suite [32] due to their strong performance-to-size ratio. Based on a comparative analysis of training loss and inference time (see Table IV), we selected EfficientNetV2Small (EfficientNetV2S) as the foundation for our expert models, offering a favorable trade-off between performance and computational efficiency.

Each expert model builds upon a customized EfficientNetV2S backbone, denoted SGrayEfficientNetV2SD2, and takes a single grayscale image of shape $(180 \times 400 \times 1)$ as input. To ensure compatibility with the EfficientNetV2S architecture, we apply a 1×1 convolution to expand the grayscale input to three channels. The resulting image is then processed through a pretrained EfficientNetV2S network with trainable weights serving as the visual encoder. The extracted feature maps are globally pooled and passed through a stack of dense layers, culminating in the simultaneous prediction of vehicle speed and steering angle.

We also propose a dual-camera model named DGrayEfficientNetV2SB3D2 for comparison. This architecture incorporates two separate visual encoders: EfficientNetV2S for the front camera and EfficientNetV2B3 for the rear camera. Given the relatively lower importance of the rear view, we employ a lighter encoder (EfficientNetV2B3) to reduce computational overhead. The architecture mirrors that of the single-camera model, with an added fusion layer following the individual pooling stages to integrate front and rear features before the final prediction layers.

For the empty bay classifier model, we chose PilotNetC3, which preserves all the dense layers of the original PilotNet, and EfficientNetV2SC1, which includes a global pooling layer and a dense layer with sigmoid activation for binary classification. This selection was made for further testing based on validation accuracy and inference time (refer to Table V).

TABLE IV: Comparison of different NN architectures in training, validation, and inference Metrics using the Cmd0 dataset

Metric	PN	MV3S	MV3L	EV2S	EV2SD2	DE2SB3D2
Tr-St	0.0308	0.0515	0.0405	0.0253	**0.0164**	0.0210
Tr-Sp	0.0322	0.0627	0.0457	0.0215	**0.0116**	0.0169
Tr-Loss	0.0630	0.1142	0.0862	0.0468	**0.0279**	0.0379
Va-St	0.0301	0.0486	0.0381	0.0222	**0.0172**	0.0195
Va-Sp	0.0313	0.0598	0.0423	0.0170	**0.0125**	0.0144
Va-Loss	0.0614	0.1083	0.0805	0.0392	**0.0296**	0.0339
Te-Loss	0.0584	0.1070	0.0763	0.0385	**0.0283**	0.0339
TFL-T	0.0225	**0.0112**	0.0320	0.3575	0.3563	0.5832
TRT-T	**0.0025**	0.0041	0.0055	0.0170	0.0174	0.0294
Tot-P (M)	4.26	**1.00**	3.10	20.47	20.61	33.85
Trn-P (M)	4.26	**0.99**	3.07	20.31	20.46	33.59
NonTrn-P (K)	**0.34**	12.1	24.4	153.9	153.9	263.1

PN: PilotNet-2016; M3S/L: MobileNetV3 Small/Large; E2S: EfficientNetV2-Small; E2SD2: E2S + Dense_1 layer (see Table VI); DE2SB3D2: Dual-cameras E2SD2 with EfficientNetV2-B3 for rear camera.Tr/Va/Te: Train/Validation/Test; St/Sp: Steering/Speed MAE; Loss: Combined MAE loss; TFL-T/TRT-T: TFLite/TensorRT Inference Time in seconds; Tot/Trn/NonTrn-P: Total/Trainable/Non-trainable Parameters in Millions (M) or Thousands (K).

I. Model Training

Models are trained on a few laptops and desktops equipped with NVIDIA GeForce RTX 3050, RTX 3070, RTX 3070 Ti, or RTX 5070 Ti. To balance general and scenario-specific behaviors, we construct each expert dataset as a mixture of two classes:

- **Main behaviors** subsampled by factor `MainDiv` to prevent dominance.

TABLE V: Comparison of Empty Bay Classifier Architectures

Metric	PNC1	PNC2	PNC3	EV2SC1
Tr-Acc	0.9993	0.9910	0.9993	**0.9995**
Tr-Loss	0.0022	0.0249	0.0021	**0.0012**
Va-Acc	0.9987	0.9915	0.9987	**0.9991**
Va-Loss	0.0035	0.0228	0.0036	**0.0028**
Te-Loss	0.0014	0.0287	**0.0010**	0.0020
TFL-T	0.0225	**0.0136**	0.0227	0.3626
TRT-T	0.0029	**0.0025**	0.0025	0.0168

PNC1/2/3: PilotNet-2016 with 2/1/4 dense layers; EV2SC1: EfficientNetV2-Small with 1 dense layers; Tr/Va/Te: Train/Validation/Test; Acc: Accuracy, TFL-T/TRT-T: TFLite/TensorRT Inference Time in seconds.

TABLE VI: SGrayEfficientNetV2SD2 Model Architecture

Layer Name	Output Size	Notes
Img_Input	(180, 400, 1)	Grayscale image input
Lambda_Normalisation	(180, 400, 1)	Normalize to [-1, 1]
GrayConv_1	(180, 400, 3)	Convert grayscale to RGB
EfficientNetV2S	(6, 13, 1280)	ImageNet pretrained, no top
GlobalAveragePooling2D	1280	Feature vector output
Dropout (p=0.2)	1280	Regularization
Dense_1 (ReLU)	200	Fully connected layer
Dense_2 (ReLU)	100	Fully connected layer
Dense_3 (ReLU)	50	Fully connected layer
Dense_4 (ReLU)	10	Fully connected layer
Speed (Linear)	1	Output: speed prediction
Steering (Linear)	1	Output: steering prediction

- **Branch behaviors** included in full or repeated (`BranchMulti` times) to emphasize rare maneuvers.

Tables VII and VIII summarize the training parameters and behavior lists used for each expert. In practice, each expert's dataset is built as:

$$\texttt{samples} = \frac{\texttt{MainList}}{\texttt{MainDiv}} + \texttt{BranchMulti} \times \texttt{BranchList}$$

We then shuffle and segment these for training and validation. This weighting ensures:

- Frequent, generic behavior (e.g., lane following) is subsampled to promote diversity.
- Rare but important maneuvers receive sufficient representation to learn effectively.

TABLE VII: Model Training Hyperparameters

Hyperparameter	Value / Description
Batch size	4–128 (device-dependent)
Optimizer	Adam
Epochs	200
Early stopping	Stop if validation loss does not improve in 3 epochs
LR schedule	Halve if no val. loss improvement in 1 epoch
Initial/Min LR	1×10^{-5} / 5×10^{-7}
Loss function	[MAE, MAE]
Loss weights	[1, 1]
Train/Val split	0.8 / 0.199
Train/Val coef.	1 / 1
Train/Val steps	Train/Val coef. × Train/Val Images / Batch size

V. Experiments

A. Design

We evaluate the expert models under both open-loop and closed-loop conditions, focusing on three key driving behaviors (see Table VIII):

- **Lane following:** Comparison between a front-camera model and a dual-camera model.

TABLE VIII: Expert Dataset Configurations

Expert	Behavior Selection (Main → Branch)
Cmd0	Lane Following → Bay Pass, Roundabout Straight, Intersection Lane Following, Startpoint In/Out, Carpark Pass, Roundabout Right Turn, Empty Bay (1st/2nd Half), Pull-out
Cmd1-1	Lane Following → Pull-in, Roundabout/Intersection Turn to Office, Startpoint In/Out, Carpark Entry, Roundabout Right Turn, Pull-in Stops, Pull-in Trigger, Empty Bay (2nd Half)
Cmd1-2	Lane Following → Bay Pass, Pull-in, Roundabout/Intersection Turn to Office, Startpoint In/Out, Carpark Entry, Roundabout Right Turn, Pull-in Stops, Pull-in Trigger, Empty Bay (2nd Half)
Cmd1-3	Pull-in Stops → Pull-in
Cmd2	Lane Following → Reverse, Roundabout/Intersection Turn to Beach, Startpoint In/Out, Carpark Entry, Roundabout Right Turn, Pull-out Stops

MainDiv/BranchMulti: Cmd0 = 1/1, Cmd1-1 = 10/1, Cmd1-2 = 10/1, Cmd1-3 = 5/1, Cmd1 = 10/1, Cmd2 = 10/1.

- **Pull-in:** Comparison between (i) a front-camera model without bay-pass data combined with an empty-bay classifier, (ii) a front-camera model trained with bay-pass (occupied bay) data, and (iii) a front-camera model trained exclusively on pull-in data.
- **Reverse:** Comparison between a front-camera model and a rear-camera model.

B. Evaluation Metrics

The evaluation is divided into two complementary settings: open-loop (offline prediction) and closed-loop (real-world deployment).

Open-loop evaluation measures offline prediction accuracy and model efficiency:

- **Prediction accuracy:** Speed and steering angle mean absolute error (MAE), as well as total loss (sum of speed and steering MAE) for training, validation, and test sets.
- **Inference time:** Model latency measured in both TensorFlow Lite and TensorRT runtimes. For safety-critical deployment, inference time must remain below 100 ms [33].
- **Pull-in trigger success rate:** Accuracy of lane empty-bay and occupied-bay classifications on an unseen test set.
- **Saliency map analysis:** Qualitative assessment of model attention on relevant scene features.

Closed-loop evaluation captures performance during autonomous operation:

- **Maneuver success rate:** Per-behavior success criteria, as defined in Table IX.
- **Manual interventions:** Number of takeovers required by a safety driver.
- **Emergency stops (e-stops):** Frequency of automatic emergency stop triggers.
- **Cautious driving behavior:** Duration of slow-driving and full-stop actions as indicators of safety conservativeness.
- **Autonomy rate:** Ratio of autonomous driving time to total driving time.

Unlike most end-to-end driving studies that evaluate models solely on open datasets such as nuScenes or Waymo Open,

our experiments extend to closed-loop testing on a full-scale autonomous shuttle. Open datasets are valuable for benchmarking perception and prediction accuracy in controlled, open-loop settings, but they cannot capture the feedback dynamics, actuation delays, or safety-critical interventions that emerge during real deployments. By contrast, our real-world evaluation exposes compounding errors in closed-loop driving, measures operational safety through metrics such as autonomy rate and emergency stop frequency, and incorporates vehicle-specific dynamics. This provides a more realistic assessment of deployability, bridging the gap between academic benchmarks and operational autonomous driving systems.

To complement this, we adopt a dual evaluation framework: open-loop metrics capture prediction accuracy, computational efficiency, and interpretability, while closed-loop metrics assess robustness, safety, and overall driving effectiveness. Together, these perspectives provide a comprehensive understanding of model performance across both controlled benchmarks and real-world conditions.

TABLE IX: Closed-loop success criteria for each expert driving behavior

Behavior	Success Criteria
Pull-in	(1) Correctly identify and fully enter the target parking bay; (2) Final heading aligned with the bay orientation; (3) Come to a complete stop without contacting curbs; (4) Complete the maneuver without requiring manual intervention.
Reverse	(1) Maintain alignment within the bay while reversing; (2) Come to a complete stop without requiring manual intervention.
Pull-out	(1) Fully exit the parking bay; (2) Merge into the main lane without contacting curbs; (3) Merge into the main lane without requiring manual intervention.
R/I to b/o	(1) Initiate the U-turn maneuver; (2) Exit at the correct designated point; (3) Avoid contacting curbs throughout the maneuver.

R/I to b/o: Roundabout / Intersection turnaround (to beach / office)

C. Results

We present the experimental outcomes in both open-loop and closed-loop settings. Open-loop evaluation covers prediction accuracy, model efficiency, and classifier performance, while closed-loop trials capture real-world deployability through autonomy rates, manual interventions, emergency stops, and cautious driving indicators.

The detailed results are summarized in Tables X to XIII, with qualitative attention analyses illustrated in Fig. 7. These tables and figures collectively provide a comprehensive view of model performance across lane following, pull-in, pull-out, reverse, and roundabout/intersection maneuvers, as well as saliency-based interpretability.

VI. Discussion

a) Lane following and pull-out (Cmd 0).: As shown in Table X, the front-camera model (C0F) and the dual-camera model (C0D) achieved similar performance in lane following and pull-out. C0D reached a slightly higher autonomy rate (94.7% $\pm$ 0.6%) with fewer manual interventions (MIC: 10.3 vs. 11.8) compared to C0F, but at the cost of higher inference latency (0.0294 s vs. 0.0174 s). Safety trade-offs were minor: both models maintained cautious driving percentages (Slow-P, Stop-P) near 5% and 3%, though C0D showed marginally more Estops (6.0 vs. 5.3). Overall, C0F is preferable for lower latency and simpler sensor setup, while C0D offers marginal gains in stability.

b) Pull-in (Cmd 1).: Within the pull-in group (Table X), the single-task model (C1F3) achieved the highest autonomy (82.5% $\pm$ 22.4%) but showed instability across bays, reflected in high standard deviations—suggesting that weather and traffic conditions strongly influence its robustness. The model trained without bay-pass data (C1F1) balanced autonomy (79.2% $\pm$ 20.8%) with moderate MIC (2.0) and occasional Estops (0.5). By contrast, the model trained with bay-pass data (C1F2) yielded the lowest autonomy (64.2% $\pm$ 36.8%) but the strongest safety indicators: MIC of 1.0, no Estops, and the lowest cautious-driving percentages (Stop-P 0.03%).

Data distribution was a decisive factor (Table XI): bays with more than six pull-in examples consistently achieved perfect scores (4.0/4 with C1F1), whereas remaining bays averaged just 3.4/4 with C1F1. Since testing occupied bays in the real world risks collisions, we evaluated this scenario using an unseen dataset (Table XIII). The hybrid model (C1F1+ECN1) reached 88.5% success on empty bays and 85.1% on occupied bays, closely matching the regression-only model trained with bay-pass data (C1F2: 87.9% / 85.8%). Crucially, the classifier also provided an interpretable binary signal that could alert the safety driver, making it more suitable for deployment despite similar raw performance. These results highlight complementary strengths: C1F3 maximizes autonomy, C1F2 emphasizes conservative safety, and C1F1 provides interpretable classifier outputs that improve both transparency and safety.

c) Reverse (Cmd 2).: For reversing maneuvers (Table X), the front-camera model (C2F) clearly outperformed the rear-camera model (C2R). C2F reached 91.1% $\pm$ 8.7% autonomy with only 3.2 MIC and negligible Estops (0.2). In contrast, C2R achieved just 56.6% $\pm$ 6.5% autonomy, required 7.3 MIC, and triggered more Estops (0.5). These failures are likely due to limited rear-camera training data and sensor issues (e.g., over-exposure), underscoring how data scarcity and sensor quality directly impact closed-loop safety.

d) Roundabout and intersection turnarounds.: As shown in Table XII, both C1F1 and C2F demonstrated strong performance in roundabouts and intersections when supported by sufficient training data. For instance, C2F scored 2.5/3 in R-1 (25 training examples) but dropped to 1.25/3 in R-2 (13 examples). These results confirm that success in complex maneuvers is highly data-dependent, with insufficient coverage leading to reduced generalization.

e) Saliency and interpretability.: Visual explanations through saliency maps (Figs. 7a to 7f and 8) further corroborate these findings. Pull-in models (states 1-1 and 1-2) focus strongly on bay regions, while the pull-in–only model (state 1-3) distributes attention more diffusely. The reverse model (state 2) also attends to the bay, while the lane-following and pull-out model (state 0) concentrates on the main road. In roundabout and intersection scenarios (Figs. 7g and 7h), pull-in models attend to environmental landmarks such as lamp posts and trees for exit identification, whereas lane-following

TABLE X: A comparison of expert models that have been trained on various command datasets. The Cmd 0 models are evaluated on lane following and pull-out tasks, while the Cmd 1 models are assessed on pull-in and roundabout/intersection turnarounds to the office. Additionally, Cmd 1 models are tested on reverse maneuvers and roundabout/intersection turnarounds to the beach (based on SGrayEfficientNetV2SD2/DGrayEfficientNetV2SB3D2)

Metric	C0F (6 laps)	C0D (4 laps)	C1F1 (6 laps)	C1F2 (4 laps)	C1F3 (4 laps)	C2F (6 laps)	C2R (4 laps)
Tr-St	**0.0164**	0.0210	0.0181	**0.0148**	0.0185	0.0168	**0.0118**
Tr-Sp	**0.0116**	0.0169	0.0116	**0.0101**	0.0126	**0.0145**	0.0176
Tr-Loss	**0.0279**	0.0379	0.0297	**0.0249**	0.0311	0.0313	**0.0294**
Va-St	**0.0172**	0.0195	0.0190	**0.0136**	0.0155	0.0132	**0.0103**
Va-Sp	**0.0125**	0.0144	0.0125	**0.0092**	0.0102	**0.0108**	0.0148
Va-Loss	**0.0296**	0.0339	0.0316	**0.0228**	0.0257	**0.0240**	0.0252
Te-Loss	**0.0283**	0.0339	0.0349	**0.0215**	0.0254	0.0228	**0.0226**
TFL-T	**0.3563**	0.5832	**0.3547**	0.3554	0.3562	**0.3547**	0.3573
TRT-T	**0.0174**	0.0294	0.0179	0.0179	**0.0174**	**0.0173**	0.0178
TD	1182.88	1106.81	88.68	26.64	86.28	106.10	137.74
MD	99.37	59.20	22.61	11.36	21.23	12.05	60.59
AD	1083.51	1047.61	66.08	15.28	65.05	94.05	77.15
AU	91.83% ± 5.02%	**94.66% ± 0.54%**	79.18% ± 20.76%	64.15% ± 36.75%	**82.54% ± 22.37%**	**91.10% ± 8.74%**	56.57% ± 6.45%
MIC	11.83 ± 5.27	**10.25 ± 1.26**	2.00 ± 1.90	**1.00 ± 0.82**	1.75 ± 1.26	**3.17 ± 2.79**	7.25 ± 1.26
Slow-D	75.13 ± 8.78	64.10 ± 13.39	7.40 ± 1.56	2.25 ± 1.80	6.43 ± 1.29	9.63 ± 4.03	5.97 ± 2.46
Slow-C	127.00 ± 12.63	107.33 ± 15.04	11.83 ± 2.56	2.75 ± 2.22	10.00 ± 3.61	13.33 ± 7.97	7.00 ± 3.00
Stop-D	42.60 ± 5.69	39.90 ± 9.07	2.95 ± 1.61	0.45 ± 0.42	2.43 ± 0.45	2.90 ± 1.64	1.40 ± 0.36
Stop-C	104.50 ± 31.39	103.33 ± 24.58	4.83 ± 2.93	0.75 ± 0.50	4.67 ± 0.58	4.00 ± 1.41	2.33 ± 0.58
Slow-P	5.45% ± 0.96%	**4.94% ± 1.03%**	0.54% ± 0.16%	**0.16% ± 0.14%**	0.50% ± 0.13%	0.68% ± 0.24%	**0.46% ± 0.22%**
Stop-P	**3.06% ± 0.30%**	3.06% ± 0.56%	0.22% ± 0.13%	**0.03% ± 0.03%**	0.19% ± 0.05%	0.21% ± 0.10%	**0.11% ± 0.04%**
Estops	**5.33 ± 1.21**	6.00 ± 2.45	0.50 ± 0.84	**0.00 ± 0.00**	0.25 ± 0.50	**0.17 ± 0.41**	0.50 ± 1.00

C0F/D: Cmd 0 Front/Dual cams; C1F1/2/3: Front cam with dataset Cmd1-1/Cmd1-2/Cmd1-3; C2F/R: Cmd 2 Front/Rear cams. Tr/Va, St/Sp, Loss, TFL-T/TRT-T same as Table IV. TD/MD/AD/AU: Total/Manual/Auto duration/Autonomy; MIC: Manual intervention count; Slow-D/Stop-D/SS-D: Slow/Stop/Slow and stop duration; Slow-C/Stop-C/SS-C: Slow/Stop/Slow and stop count; Slow-P/Stop-P/SS-P: Slow/Stop/Slow and stop percentage; Bold indicates best result within each command group; All durations in seconds.

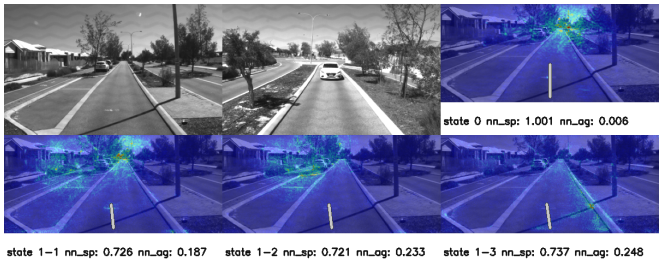
TABLE XI: Success statistics of pull-in (4 points maximum), reverse (2 points maximum), and pullout (3 points maximum) models across bays using criteria from Table IX with bay training data counts (PIC/RVC/POC)

Bay	PIC	C1F1	C1F2	C1F3	RVC	C2F	C2R	POC	C0F	C0D
1-1	1	–	–	–	1	–	–	1	–	–
1-2	11	–	–	–	9	–	–	6	–	–
1-3	4	–	–	–	4	–	–	4	–	–
2-1	12	4 ± – (n=1)	4 ± – (n=1)	4 ± 0 (n=2)	10	1.86 ± 0.38 (n=7)	2 ± 0 (n=2)	5	3 ± 0 (n=7)	3 ± 0 (n=3)
2-2	16	–	–	2 ± – (n=1)	16	–	2 ± – (n=1)	18	–	3 ± – (n=1)
3-1	2	–	3 ± – (n=1)	–	7	–	–	4	–	–
3-2	2	2 ± – (n=1)	–	–	4	2 ± 0 (n=2)	–	5	3 ± – (n=1)	–
4-1	2	4 ± 0 (n=3)	4 ± 0 (n=2)	4 ± 0 (n=3)	4	2 ± 0 (n=4)	0 ± 0 (n=3)	5	3 ± 0 (n=4)	3 ± – (n=1)
4-2	2	4 ± – (n=1)	–	–	3	2 ± – (n=1)	–	1	3 ± – (n=1)	–
4-3	10	4 ± – (n=1)	–	4 ± – (n=1)	10	–	1 ± 1.41 (n=2)	10	–	3 ± 0 (n=2)
5-1	2	–	–	–	1	2 ± – (n=1)	–	2	3 ± – (n=1)	–
5-2	1	–	–	–	0	–	–	0	–	–
6-1	6	3.33 ± 0.58 (n=3)	–	0 ± 0 (n=2)	3	1.5 ± 0.71 (n=2)	–	2	3 ± 0 (n=3)	3 ± – (n=1)
6-2	10	–	3 ± – (n=1)	–	2	–	–	6	–	–
7-1	1	–	0 ± – (n=1)	–	2	–	–	3	–	–
8-1	4	2.5 ± 2.12 (n=2)	0 ± – (n=1)	–	4	1 ± – (n=1)	–	4	3 ± – (n=1)	3 ± – (n=1)
8-2	4	4 ± – (n=1)	1.5 ± 2.12 (n=2)	4 ± 0 (n=3)	3	2 ± 0 (n=2)	0.6 ± 0.55 (n=5)	5	3 ± 0 (n=2)	3 ± 0 (n=3)
9-1	0	–	–	–	0	–	–	0	–	–
9-2	6	4 ± 0 (n=3)	4 ± 0 (n=2)	4 ± 0 (n=2)	0	1 ± 1 (n=3)	1 ± 0 (n=2)	1	3 ± 0 (n=2)	3 ± 0 (n=2)
9-3	5	4 ± 0 (n=2)	–	–	3	2 ± 0 (n=2)	–	6	3 ± 0 (n=2)	–
9-4	2	–	–	–	4	–	–	4	–	–
10-1	8	4 ± – (n=1)	–	–	2	0 ± – (n=1)	–	3	3 ± – (n=1)	–
11-1	5	3.6 ± 0.89 (n=5)	2.67 ± 0.58 (n=3)	4 ± 0 (n=2)	5	1.5 ± 0.55 (n=6)	0.5 ± 0.58 (n=4)	4	3 ± 0 (n=5)	3 ± 0 (n=3)
11-2	16	4 ± – (n=1)	0 ± – (n=1)	4 ± – (n=1)	18	0.5 ± 0.71 (n=2)	1 ± – (n=1)	16	3 ± – (n=1)	1 ± – (n=1)
11-3	7	4 ± – (n=1)	4 ± – (n=1)	4 ± – (n=1)	8	2 ± 0 (n=3)	0 ± – (n=1)	9	3 ± 0 (n=3)	–
11-4	4	–	4 ± – (n=1)	–	3	2 ± – (n=1)	–	2	3 ± – (n=1)	–
12-1	2	–	4 ± – (n=1)	–	1	2 ± – (n=1)	–	1	3 ± – (n=1)	–
13-1	3	0 ± – (n=1)	3.5 ± 0.71 (n=2)	3.5 ± 0.71 (n=2)	3	–	1.5 ± 0.71 (n=2)	3	–	2.33 ± 0.58 (n=3)
13-2	9	–	–	–	11	–	–	14	3 ± – (n=1)	–
14-1	2	–	–	3 ± – (n=1)	2	–	1 ± – (n=1)	1	–	3 ± – (n=1)
14-2	15	–	–	4 ± – (n=1)	12	2 ± 0 (n=2)	–	13	3 ± 0 (n=2)	–
15-1	2	–	–	–	1	–	–	0	–	–
15-2	6	–	–	–	9	–	–	4	–	–
15-3	0	–	–	–	1	–	–	0	–	–
16-1	0	–	–	–	0	–	–	0	–	–

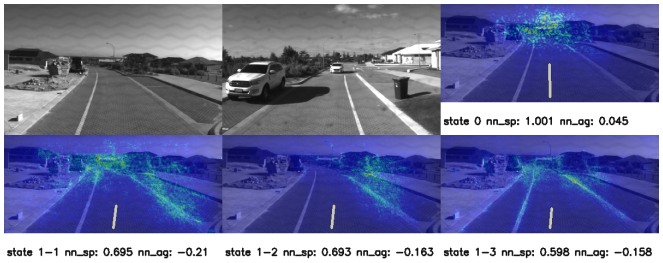
PIC: Pull-in count, RVC: Reverse count, POC: Pullout count, bays are numbered office → beach → office; Model columns indicate success rate for corresponding models (C1F1, C2R, etc.); Values are mean ± SD (sample standard deviation) (n=Count); "–" denotes unavailable values.

and reverse models rely more on road markings. The classifier model (ENC1) emphasizes obstacles such as vehicles and curbs, highlighting its role in safety monitoring.

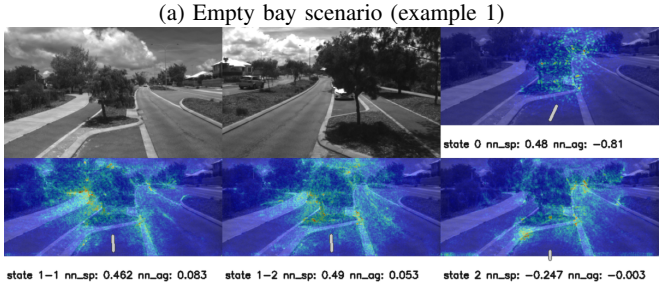
f) Classifier versus regression models.: The "empty bay" challenge is analogous to the red-light violation problem identified in [20, 21], where regression models struggle with discrete decision boundaries. Ishihara *et al.* demonstrated that



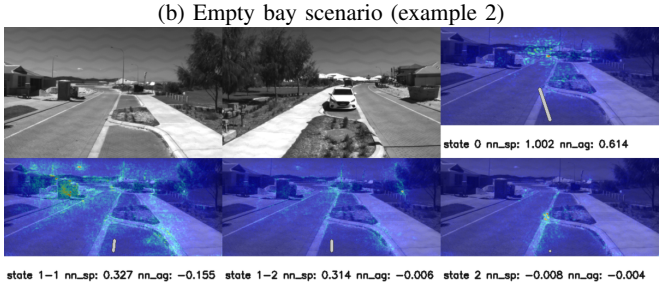
(a) Empty bay scenario (example 1)



(b) Empty bay scenario (example 2)



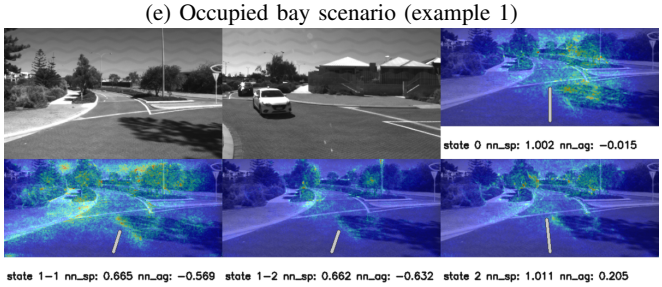
(c) In-bay maneuver (example 1)



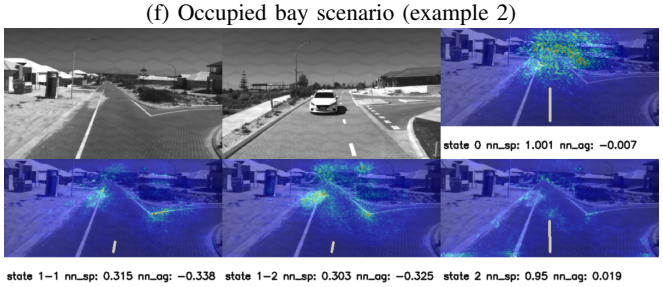
(d) In-bay maneuver (example 2)

(e) Occupied bay scenario (example 1)

(f) Occupied bay scenario (example 2)



(g) Roundabout scenario



(h) Intersection scenario

Fig. 7: Comparison of saliency maps across navigation neural network models in different driving scenarios. In each subfigure, the first two panels correspond to the front and rear input images. States 0, 1-1, 1-2, 1-3, and 2 denote expert models Cmd0, Cmd1-1, Cmd1-2, Cmd1-3, and Cmd2, respectively. `nn_sp` indicates the neural network speed output, while `nn_ag` denotes the neural network angular output.

classifiers are particularly effective in such contexts, capturing local pixel-level cues that regression-based action models often miss. Our results align with this observation, showing that classifier integration both improves interpretability and reduces safety risks.

g) Single-task versus multi-task models.: Finally, comparisons between single-behavior and multi-behavior models suggest that behaviors occurring in distinct contexts can be consolidated without major performance loss. For example, C1F3 achieved 82.5% autonomy on its narrow pull-in task, while multi-behavior models (C1F1, C1F2) attained more moderate autonomy levels (64–79%) but generalized across multiple tasks (e.g., lane following and roundabout turnarounds). High autonomy of lane following models suggests that behaviors in distinct spatial contexts (e.g., pull-in actions in left side bays compared to right side bays; lane following in a single lane versus a two-way lane; lane following on concrete surfaces as opposed to brick surfaces) can be effectively merged within a single model without major performance degradation. These comparisons indicate potential for consolidating expert models into fewer, more generalized architectures, while still maintaining task reliability across diverse maneuvers, as long as the behaviors are not situated in the same position.

TABLE XII: Success statistics of roundabout/intersection turnarounds to office/beach (3 points maximum) models using criteria from Table IX with roundabout/intersection training data counts (TOC/TBC)

Area	TOC	C1F1	TBC	C2F
R-1	1	–	25	2.5 ± 1 (n=4)
R-2	4	3 ± - (n=1)	13	1.25 ± 0.5 (n=4)
R-3	21	1.667 ± 1.155 (n=3)	35	3 ± - (n=1)
I-1	0	–	0	–
I-2	4	–	0	–
I-3	10	1 ± – (n=1)	0	–

TOC/TBC: Turnaround to office/beach counts; R/I: Roundabout/Intersection, numbered office → beach; Values are mean ± SD (sample standard deviation) (n=Count); "–" denotes unavailable values.

Fig. 8: Comparison of saliency maps across classification models in different driving scenarios. States ENC1 denotes EfficientNetV2SC1. `empty_bay` indicates the neural network empty bay classification probability output.

h) Summary.: Overall, these findings underscore the necessity of evaluating models under both open-loop and closed-loop conditions. While open-loop metrics capture accuracy and computational efficiency, they cannot reveal compounding errors, disengagements (MIC), excessive Estops, or overly cautious driving patterns (Slow-P, Stop-P). Closed-loop testing was therefore essential to uncover trade-offs between accuracy, robustness, and safety, providing a realistic measure of deployability in real-world autonomous driving.

We selected C0F + C1F1 + C2F as the best mixture-of-experts (MoE) combination because the single-camera model (C0F) exhibits performance comparable to that of the dual-camera (C0D). Additionally, the dual-camera model is less stable due to increased inference time, which also doubles the likelihood of camera failure. Furthermore, the rear camera frequently encounters over-exposure issues.

VII. Conclusion

This work demonstrates that a Mixture of Experts (MoE) approach can efficiently handle common suburban shuttle maneuvers using only grayscale camera input and limited training data. By evaluating models in both open-loop and closed-loop settings, we show that offline prediction accuracy alone is insufficient: real-world testing is essential to expose failure modes such as unstable pull-in autonomy and poor reverse generalization.

Among the tested models, the MoE combination of C0F (front-camera lane following and pull-out), C1F1 (hybrid classifier–regression pull-in), and C2F (front-camera reverse) achieved the best balance of autonomy, safety, and interpretability. The dual-camera model (C0D) offered only marginal gains while introducing higher latency and greater hardware risks, and the rear-camera model (C2R) suffered from poor performance due to limited data and sensor issues. These results underline the value of single-camera designs for lightweight and reliable deployment.

Our findings further highlight the importance of data distribution and task framing. Bays with sufficient pull-in samples achieved near-perfect success, whereas sparse coverage degraded performance. The hybrid classifier–regression design (C1F1) not only matched the regression-only alternative in accuracy but also provided interpretable binary signals that can be surfaced to safety drivers, improving transparency and trust. Saliency analyses confirmed that models attend to contextually meaningful cues such as bay regions, road markings, and environmental landmarks, while classifiers emphasized local obstacles relevant to safety-critical decisions.

Overall, this study shows that lightweight, camera-only systems—when combined with expert decomposition and classifier integration—can deliver practical and interpretable autonomy in suburban shuttle operations. Future work will extend these insights toward consolidating behaviors into more generalized models, improving classifier components, and scaling real-world evaluation under broader environmental and traffic conditions.

VIII. Future Work

All-in-One Model: While this paper uses separate expert models for each high-level driving command, we plan to explore an all-in-one model architecture similar to the branched conditional imitation learning approach proposed in [3], where command-specific branches follow a shared encoder. Although this architecture introduces additional complexity—particularly in training time, TensorFlow-to-TensorRT conversion, and limited branch-specific iteration—it holds the potential for improved efficiency and compact deployment.

Front Camera On-Road Moving Vehicle Detection: To better handle roundabout give-way scenarios, we aim to develop a vision-based module for detecting oncoming vehicles in the front camera feed. This will require a camera with a wider and longer field of view than our current setup, which only detects vehicles directly ahead. Detecting vehicles at roundabout entry points in advance would allow more proactive and safer decision-making.

Rear Camera On-Road Moving Vehicle Detection: We plan to enhance the detection of approaching traffic from the rear during pull-out maneuvers. The current rear camera setup has a limited sensing horizon, only capable of detecting vehicles approximately three seconds before they pass—insufficient for a safe 10-second pull-out operation. Integrating a longer-range rear-view sensor or an improved vision model will help facilitate safer merging behavior.

Cul-de-sac Autonomous Navigation: Future efforts will include training a specialized model for navigating cul-de-sac environments. This involves learning to perform tight turnarounds while avoiding static obstacles such as parked

TABLE XIII: Pull-in trigger success rates of lane empty-bay and lane occupied-bay scenarios in unseen test dataset

Metric	PC3	EC1	Cmd1-1	Cmd1-2	Cmd1-3	EC1+Cmd1-1
Lane Empty Bay (157 bays)						
Count of success	139	143	**148**	138	129	139
Success rate	88.54%	91.08%	**94.27%**	87.90%	82.17%	88.54%
Lane Occupied Bay (282 bays)						
Count of success	242	238	101	**242**	58	240
Success rate	85.82%	84.40%	35.82%	**85.82%**	20.57%	85.11%

PC3/EC1: Expert classifier models; Cmd1-1/2/3: pull-in command subsets; EC1+Cmd1-1: hybrid model; Values represent the number and percentage of successful trials. For empty bays (157 total), a success is defined as either: (i) classifier output exceeding 80% predicted probability for at least one frame, or (ii) regression model output with absolute steering angle larger than 0.2 sustained for at least four consecutive frames. For occupied bays (282 total), a success is defined as passing (i.e., not pulling in) under the same classifier and regression model criteria; Bold indicates the best success rate in each block.

vehicles, a challenging maneuver that requires fine-grained control and perception.

Temporal Sequence Modeling: To enhance behavioral consistency and decision accuracy, we intend to integrate temporal information into the driving model. Incorporating image sequences or recurrent architectures (e.g., ConvLSTM, Transformers) could allow the system to reason over time and better anticipate dynamic changes in the environment. This is particularly critical for empty-bay classification and pull-in maneuvers, where most models failed when only the first half of a bay was occupied. In such cases, the front camera often loses sight of the parked vehicle when approaching the second half of the bay, leading to misclassification or incorrect pull-in behavior.

Generalization and Deployment in Unseen Environments: We aim to evaluate the model's generalization capabilities in novel and untrained environments. However, this is currently constrained by the limitations imposed by the Traffic Management Plan (TMP) and the Department of Transport (DoT) regulations. Future work will seek to relax these constraints, either through simulation-based testing or by negotiating expanded testing areas.

ACKNOWLEDGMENTS

The authors would like to thank the Department of Transport (DoT) in Western Australia and Stockland Corporation Limited for the support of the Eglinton experiment. The authors appreciate all our REV sponsors, including Allkem, Tesla Slack, CD Dodd, Dyflex, Tesla Roadster Inc, Tesla Owners Club of WA, Tesla Owners Club of Australia, SBG Systems, and Altronics, as well as our university sponsors UWA School of Engineering, Business School, School of Physics, Mathematics and Computing, and the UWA Innovation Quarter.

REFERENCES

[1] "Automated vehicles for safety." [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[2] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[3] F. Codevilla, M. Müller, A. Dosovitskiy, A. M. López, and V. Koltun, "End-to-end driving via conditional imitation learning," *CoRR*, vol. abs/1710.02410, 2017. [Online]. Available: http://arxiv.org/abs/1710.02410

[4] A. Tampuu, M. Semikin, N. Muhammad, D. Fishman, and T. Matiisen, "A survey of end-to-end driving: Architectures and training methods," *CoRR*, vol. abs/2003.06404, 2020. [Online]. Available: https://arxiv.org/abs/2003.06404

[5] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers." [Online]. Available: http://arxiv.org/abs/2306.16927

[6] P. S. Chib and P. Singh, "Recent advancements in end-to-end autonomous driving using deep learning: A survey," vol. 9, no. 1, pp. 103–118, conference Name: IEEE Transactions on Intelligent Vehicles. [Online]. Available: https://ieeexplore.ieee.org/document/10258330

[7] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 1. Morgan-Kaufmann, 1988. [Online]. Available: https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf

[8] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *CoRR*, vol. abs/1612.01079, 2016. [Online]. Available: http://arxiv.org/abs/1612.01079

[9] L. Chi and Y. Mu, "Deep steering: Learning end-to-end driving model from spatial and temporal visual cues," *CoRR*, vol. abs/1708.03798, 2017. [Online]. Available: http://arxiv.org/abs/1708.03798

[10] H. M. Eraqi, M. N. Moustafa, and J. Honer, "End-to-end deep learning for steering autonomous vehicles considering temporal dependencies," *CoRR*, vol. abs/1710.03804, 2017. [Online]. Available: http://arxiv.org/abs/1710.03804

[11] S. Hecker, D. Dai, and L. V. Gool, "Learning driving models with a surround-view camera system and a route planner," *CoRR*, vol. abs/1803.10158, 2018. [Online]. Available: http://arxiv.org/abs/1803.10158

[12] J. Nie, J. Yan, H. Yin, L. Ren, and Q. Meng, "A multimodality fusion deep neural network and safety test strategy for intelligent vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 310–322, 2021.

[13] Q. Wang, L. Chen, B. Tian, W. Tian, L. Li, and D. Cao, "End-to-end autonomous driving: An angle branched network approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 11 599–11 610, 2019.

[14] M. Bojarski, C. Chen, J. Daw, A. Degirmenci, J. Deri, B. Firner, B. Flepp, S. Gogri, J. Hong, L. D. Jackel, Z. Jia, B. J. Lee, B. Liu, F. Liu, U. Muller, S. Payne, N. K. N. Prasad, A. Provodin, J. Roach, T. Rvachov, N. Tadimeti, J. E. van Engelen, H. Wen, E. Yang, and Z. Yang, "The NVIDIA pilotnet experiments," *CoRR*, vol. abs/2010.08776, 2020. [Online]. Available: https://arxiv.org/abs/2010.08776

[15] T. Weiss and M. Behl, "Deepracing: Parameterized trajectories for autonomous racing," *CoRR*, vol. abs/2005.05178, 2020. [Online]. Available: https://arxiv.org/abs/2005.05178

[16] Y. Chen, P. Praveen, M. Priyantha, K. Muelling, and J. Dolan, "Learning on-road visual control for self-driving vehicles with auxiliary tasks," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 331–338.

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

[18] S. Chen, S. Zhang, J. Shang, B. Chen, and N. Zheng, "Brain-inspired cognitive model with attention for self-driving cars," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 11, no. 1, pp. 13–25, 2019.

[19] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J. Allen, V. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," *CoRR*, vol. abs/1807.00412, 2018. [Online]. Available: http://arxiv.org/abs/1807.00412

[20] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," *CoRR*, vol. abs/1904.08980, 2019. [Online]. Available: http://arxiv.org/abs/1904.08980

[21] K. Ishihara, A. Kanervisto, J. Miura, and V. Hautamaki, "Multi-task learning with attention for end-to-end autonomous driving," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, pp. 2896–2905. [Online]. Available: https://ieeexplore.ieee.org/document/9523129/

[22] I. Kim, J. Lee, and D. Kim, "Learning mixture of domain-specific experts via disentangled factors for autonomous driving," vol. 36, no. 1, pp. 1148–1156, number: 1. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/20000

[23] Z. Yang, Y. Chai, X. Jia, Q. Li, Y. Shao, X. Zhu, H. Su, and J. Yan, "DriveMoE: Mixture-of-experts for vision-language-action model in end-to-end autonomous driving." [Online]. Available: http://arxiv.org/abs/2505.16278

[24] R. Feng, N. Xi, D. Chu, R. Wang, Z. Deng, A. Wang, L. Lu, J. Wang, and Y. Huang, "ARTEMIS: Autoregressive end-to-end trajectory planning with mixture of experts for autonomous driving." [Online]. Available: http://arxiv.org/abs/2504.19580

[25] S. Macenski, F. Martin, R. White, and J. Ginés Clavero, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[26] T. H. Drage, K. Quirke-Brown, L. Haddad, Z. Lai, K. L. Lim, and T. Bräunl, "Managing risk in the design of modular systems for an autonomous shuttle," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 5, pp. 555–565, 2024.
[27] S. Macenski, S. Singh, F. Martin, and J. Gines, "Regulated pure pursuit for robot path tracking," *Autonomous Robots*, 2023.
[28] Foxglove documentation. [Online]. Available: https://docs.foxglove.dev/docs
[29] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, "Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline." [Online]. Available: http://arxiv.org/abs/2206.08129
[30] D. Liu and A. Moch, *Optimizing Object Detection in Autonomous Vehicles Using Grayscale Computer Vision Models*. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-351093
[31] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3." [Online]. Available: http://arxiv.org/abs/1905.02244
[32] K. Team. Keras documentation: Keras applications. [Online]. Available: https://keras.io/api/applications/
[33] L. Chen, Y. Li, C. Huang, Y. Xing, D. Tian, L. Li, Z. Hu, S. Teng, C. Lv, J. Wang, D. Cao, N. Zheng, and F.-Y. Wang, "Milestones in autonomous driving and intelligent vehicles—part i: Control, computing system design, communication, HD map, testing, and human behaviors," vol. 53, no. 9, pp. 5831–5847, conference Name: IEEE Transactions on Systems, Man, and Cybernetics: Systems. [Online]. Available: https://ieeexplore.ieee.org/document/10138317

IX. Biography Section

Zhihui Lai received the B.E. (Hons.) degree from The University of Western Australia, Perth, Australia, in 2021, where he is currently pursuing a Ph.D. degree with a focus on vision-based deep learning methods for autonomous driving. His research interests include end-to-end self-driving based on deep neural networks, autonomous driving simulators, computer vision, object detection and tracking.

Kieran Quirke-Brown has received a Master of Professional Engineering in Electrical & Electronics engineering in 2017 from The University of Western Australia. Between 2017 and 2021 Kieran worked as a BMS (Building Management Services) Engineer with Schneider Electric while earning his second master's degree in Information Technology from The University of Queensland. In 2021 he returned to UWA as a Ph.D. student in robotics and automation; his research focus' on how autonomous vehicles react to various scenarios and how proactive driving can be achieved. He also holds a strong passion for teaching the next generation of students.

Xiangrui Kong is a PhD candidate at the University of Western Australia since 2022, researching autonomous public vehicle, object detection, and large language model. He previously worked on CAD/CAM software at UDS China and optimized autonomous underwater systems at the China Electronics Technology Group Corporation Research Institute. He holds a master's degree from Ocean University of China, where he focused on path planning for autonomous underwater vehicles.

Lee Le obtained his Master of Professional Engineering degree from The University of Western Australia, Perth, Australia, in 2024. He is currently a Ph.D. candidate who concentrates on LiDAR-based algorithms and deep learning technique for autonomous driving. His research interests encompass SLAM, autonomous driving simulators, object detection and avoidance.

Thomas Bräunl is a Professor in the School of Engineering at The University of Western Australia, Perth, where he directs the Robotics & Automation Lab as well as the Renewable Energy Vehicle Project (REV). He has developed numerous robotics systems, including the EyeBot robot family and the EyeSim simulation system. On the automotive side, he has done research on electric drive and charging systems, and is developing AI solutions for autonomous driving. Professor Bräunl worked on Driver-Assistance Systems with Mercedes-Benz Stuttgart and on Electric Vehicle Charging Systems with BMW Munich and BMW Mountain View. He holds a Diploma from the University of Kaiserslautern, an M.S. degree from the University of Southern California, Los Angeles, and a Ph.D. and Habilitation from the University of Stuttgart.

APPENDIX A
ADDITIONAL DISCUSSION AND THESIS-RELATED CLARIFICATIONS

This appendix provides supplementary discussion addressing reviewer questions raised during the author's PhD thesis evaluation. The content is included for completeness and transparency, and reflects exploratory analysis and design considerations that are not central to the primary experimental contributions of this paper.

A. Use of Grayscale Cameras

A reviewer raised concerns regarding the reliance on grayscale cameras and their potential limitations under challenging lighting or occlusion conditions. In practice, the performance degradation in such scenarios is dominated less by color information and more by camera exposure dynamics, motion blur, and sensor saturation. In our deployments, overexposure caused by rapid illumination changes (e.g., entering open areas from shaded regions) affected both grayscale and RGB sensors similarly.

Grayscale imaging was selected primarily due to hardware constraints of the EasyMile EZ10 platform and to reduce computational load. Prior work has shown that many driving-relevant cues—such as lane boundaries, vehicle contours, and road geometry—are largely encoded in luminance rather than chromatic information. While color can be beneficial in certain semantic tasks (e.g., traffic light recognition), the targeted suburban shuttle environment and maneuver set considered in this study did not rely on color-specific cues. Future work may explore adaptive exposure control or HDR sensing to further improve robustness.

B. Train–Validation–Test Data Splits

All datasets used in this study were collected on a fixed suburban route approved by local authorities. Due to regulatory and safety constraints, testing on unseen public roads was not permitted. Consequently, training, validation, and test splits were generated from temporally disjoint driving sessions within the same operational area.

Specifically, driving logs were partitioned chronologically to ensure that test sequences were not temporally adjacent to training data, reducing frame-level correlation. While the physical locations remain identical, environmental variations such as traffic presence, pedestrian activity, and minor illumination changes introduce a degree of variability. This evaluation protocol reflects realistic deployment conditions for fixed-route autonomous shuttles and aligns with the intended operational design domain (ODD).

C. Driving Smoothness and Ride Comfort

Another reviewer questioned the smoothness of the generated driving behavior, particularly with respect to trajectory continuity and passenger comfort. The proposed end-to-end models output continuous steering and speed commands and inherently produce smooth control signals due to the supervised imitation learning paradigm, which mimics human driving behavior.

Trajectory smoothness was qualitatively assessed by projecting vehicle paths onto a map and visually inspecting curvature continuity through roundabouts and intersections. In practice, passenger discomfort during testing was primarily caused not by the neural network outputs, but by redundant safety mechanisms—specifically LiDAR-based emergency stops—which introduce abrupt decelerations when conservative safety thresholds are triggered. These interruptions are external to the learned policy and reflect safety-layer constraints rather than control instability. Quantitative jerk or acceleration measurements were not logged in this study but are planned for future evaluations.

D. Safety Considerations and Operational Design Domain

The proposed system does not explicitly address advanced safety use cases such as minimum risk maneuvers (MRMs), blocked-road recovery, or emergency path planning. These scenarios typically require precise localization, semantic understanding, and explicit planning layers, which fall outside the scope of the lightweight end-to-end framework explored in this work.

Given the limited sensor suite and absence of high-definition maps or redundancy, a trained safety driver was required during all trials. Addressing safety-critical behaviors and formally defining system performance across multiple ODDs remains an important direction for future research. Potential extensions include integrating rule-based fallback behaviors, formal safety monitors, or hybrid architectures combining end-to-end control with constrained planning for emergency scenarios.

Chapter 6

Conclusion

I. Summary

This thesis investigated the feasibility, limitations, and practical deployment of lightweight, camera-based end-to-end (E2E) autonomous driving systems under constrained computational, data, and operational conditions. Motivated by gaps in existing E2E research—particularly its reliance on large-scale datasets, powerful hardware, and limited real-world validation—the work was structured around four research questions (RQ1–RQ4) that collectively address architectural design, temporal robustness, simulation-supported development, and real-world deployment.

To address **RQ1**, the thesis demonstrated that compact, monocular camera-based E2E systems can be designed, trained, and deployed on low-cost embedded platforms when supported by careful data curation, staged training pipelines, and task-specific architectural choices. Starting from PilotNet-style baselines, the work showed that acceptable closed-loop performance can be achieved without large-scale fleet data or high-end computing infrastructure, provided that the operational design domain (ODD) is clearly defined and respected.

RQ2 examined architectural enhancements required for complex, sequential maneuvers such as dead-end recovery, pull-ins, pull-outs, roundabout navigation, and parallel parking. Through a progression from frame-based CNNs to memory-augmented models (LSTM and 3D-CNN) and ultimately Mixture-of-Experts (MoE) architectures, the research demonstrated that temporal reasoning and maneuver specialization significantly improve robustness and behavioral consistency. The MoE framework, in particular, enabled smoother and more interpretable decision-making by decomposing complex driving tasks into context-specific experts.

RQ3 focused on the role of simulation in scalable E2E development. By benchmarking CARLA and AWSIM using common digital twins, this thesis clarified their complementary strengths in realism, performance, ROS2 integration, and sensor modeling. The results highlight that no single simulator is sufficient for all development stages; instead, a layered simulation strategy—combining EyeSim, ModCar, CARLA, and AWSIM—provides a practical and reproducible pathway for model development, validation, and controlled risk reduction prior to real-world trials.

Finally, **RQ4** evaluated the extent to which lightweight, camera-only E2E frameworks can achieve reliable closed-loop performance on real suburban shuttle routes. Through deployments on the nUWAy1 and nUWAy2 autonomous shuttles, the research demonstrated that such systems can operate safely and smoothly within constrained ODDs, including campus environments and suburban residential areas. Importantly, these findings are not presented as universal safety guarantees, but as empirical evidence of feasibility under explicitly defined operational boundaries.

Across all studies, a key contribution of this thesis is the establishment of a complete research pipeline—from simulation and scaled platforms to full-scale shuttle deployment—that enables systematic investigation of E2E autonomy under realistic constraints. The work situates camera-based E2E driving within the historical evolution of autonomous driving, illustrating how modern learning-based systems can complement, rather than replace, classical modular safety mechanisms when deployed in carefully engineered contexts.

II. Limitations and Future Work

While the results presented in this thesis are encouraging, several limitations must be acknowledged.

First, all experimental evaluations were conducted within **explicitly constrained operational design domains**. The deployed systems were tested at modest speeds, on predefined routes, and under limited environmental variability. As such, the results should not be interpreted as evidence of general-purpose autonomy. Edge cases outside the tested ODDs—such as dense urban traffic, extreme weather, or highly dynamic interactions—remain open challenges.

Second, the driving scenarios investigated in this work were **largely fixed and curated**, both in simulation and real-world trials. Although this controlled approach was necessary to ensure safety and repeatability during development, it limits exposure to rare or adversarial events. Future work should

incorporate broader scenario diversity and longer-term deployments to better characterize failure modes and performance drift.

Third, the thesis deliberately focuses on **image-only E2E perception**. Unlike classical modular systems, end-to-end learning does not guarantee deterministic or repeatable behavior in the traditional control-theoretic sense. However, this work argues that strict repeatability is not a prerequisite for safe autonomy. Instead, the objective is to achieve behaviors that are consistently correct, smooth, and safe within the defined ODD, supported by external safety layers such as LiDAR curtaining, emergency stops, and human oversight. This framing aligns with emerging regulatory perspectives that emphasize outcome-based safety rather than deterministic internal representations.

A further limitation concerns the **sim-to-real gap**, which remains one of the most significant challenges in deep learning for robotics. While simulators were extensively used for training, testing, and benchmarking, the thesis intentionally avoided claims of seamless sim-to-real transfer. Domain mismatch in appearance, dynamics, and sensor noise continues to affect deployment performance. Addressing this gap represents a major opportunity for future research. In particular, recent advances in large generative models for image translation and domain randomization offer promising directions for sim-to-real adaptation, enabling more realistic synthetic data generation and improved robustness across domains.

Looking ahead, several avenues for future work emerge. Architecturally, more unified models that combine maneuver specialization with shared representations—potentially using transformer-based temporal encoders—could further improve scalability and interpretability. From a systems perspective, extending evaluations to longer deployment periods, denser traffic, and multi-vehicle interactions would strengthen confidence in real-world robustness. Finally, deeper integration with traffic management systems and regulatory frameworks could support broader pilot deployments of low-cost, camera-centric autonomous shuttles.

III. Concluding Remarks

This dissertation demonstrates that lightweight, camera-based end-to-end autonomous driving systems can achieve practical levels of autonomy when developed and deployed within carefully defined constraints. By integrating simulation, scaled platforms, and real-world shuttle deployments, the research bridges the gap between algorithmic development and operational reality, offering both technical contributions and a reproducible methodology for future work.

Within the broader autonomous driving landscape, this thesis occupies a distinct position. While industrial leaders such as Tesla pursue large-scale vision-based autonomy and others such as Waymo rely on LiDAR-intensive sensor stacks and high-definition mapping, this work explores an alternative design point: low-cost, interpretable, and camera-centric systems tailored for campus and suburban mobility. The results suggest that meaningful autonomy does not require either massive fleets or expensive sensor suites, provided that system design, evaluation, and deployment are aligned with realistic operational goals.

More broadly, the thesis highlights the continuing relevance of academic research in autonomous driving. By operating outside purely commercial incentives, university-led efforts can explore constrained, transparent, and sustainable approaches that complement industrial development. In doing so, this work contributes to the evolving narrative of autonomous driving—one in which end-to-end learning, classical safety mechanisms, and regulatory awareness coexist to enable practical and responsible deployment.