Reactive and Proactive Autonomous Driving Techniques

Kieran Scott Quirke-Brown

BSc MPE MIT

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

School of Engineering
Department of Electrical, Electronic and Computer Engineering

2025

Acknowledgements

I express my deepest gratitude to my supervisor, Professor Thomas Bräunl, for his incredible support throughout my project and beyond. He has provided me with numerous opportunities, from serving as team manager on the REV team to gaining teaching experience across various units, and has encouraged me to pursue a faculty position. Working with him allowed me to explore a wide range of vehicles and cutting-edge automation technology, including the nUWAy shuttle buses. I am also grateful to my other supervisors and academics with whom I consulted along the way, whose insights and perspectives I will carry forward in my career.

I extend my thanks to the past and present members of the REV team, starting with Zhihui Lai, Xiangrui Kong, An Trung Le, and Patrick Riedel, for their invaluable assistance with the nUWAy shuttles and collaborative publications. In particular, I would like to acknowledge my colleague Zhihui Lai who has been there the whole way of this journey, helping me generate ideas, keeping me sane during trials and providing his invaluable knowledge around AI models, data cleaning, and training pipelines without might have made this journey so much harder. I am also grateful to Lemar Haddad, Jai Castle, Zack Wong, Tim Tan, and others who dedicated countless hours refining the shuttles to their current state. Special thanks to Aaron Delandgrafft, Sebastian Williams, Travis Ryan, Fraser Loneragan, and Jason Seotis, who ensured I could take breaks and enjoy my time at UWA.

I would like to acknowledge the generous support of our sponsors, CD DODD and SBG, for providing funds and equipment essential to this work, as well as Tim Brunner from EV Works for his ongoing assistance with repairs and fit-outs.

Finally, I am deeply thankful to my family for their unwavering encouragement and support during my studies, particularly my daughter Ciara, whose smile made the hardest days worthwhile.

I also gratefully acknowledge the Stockland team for providing financial support and the operational base for our road trials at Amberton Beach, Eglinton.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship doi.org/10.82133/C42F-K220.

Abstract

The rapid development of faster and smaller computing systems has fundamentally changed the way driving is conceptualised. Many modern vehicles are already equipped with features that reduce the need for driver input. In recent years, there has been an acceleration in autonomous vehicle research, driven both by the increasing number of vehicle-related fatalities and the broader societal demand for safer and more efficient transport solutions. This thesis investigates how autonomous driving systems perceive and proactively respond to their environment to achieve SAE Level 4 autonomy. Although current vehicles predominantly operate at Level 3 autonomy, this work seeks to advance real-world autonomous driving capabilities through systematic experimentation.

The early chapters of this thesis focus on deterministic approaches and their application to a university campus environment. However, as artificial intelligence techniques rapidly surpass traditional deterministic systems, later chapters explore how advanced AI models can improve driving performance on suburban roads and illuminate their inherent limitations. Specifically, this work examines state-of-the-art sequence-based models, including Transformers and state-space models, with a focus on the Mamba SSM variant. These models are evaluated across multiple sensor modalities and feedback mechanisms to improve road performance while maintaining computational efficiency.

Additional safety systems are implemented to promote user comfort and trust in autonomous vehicle systems. The findings indicate that while deterministic systems can achieve reliable on-campus performance, more complex algorithms can exhibit behaviours that are unpredictable to nearby pedestrians, which may reduce overall driving reliability. Furthermore, the open terrain of a campus environment constrains the effectiveness of localisation methods such as 2D SLAM. State-space models (SSMs) demonstrate strong performance, maintaining or exceeding the accuracy of transformer-based models while reducing resource requirements. LiDAR-based models also show promise but require specialised training to achieve robust performance. Simple safety interventions have been shown to improve driving time and user comfort, while temporal and feedback models require careful architectural and training design to mitigate lag and bias in real-time operations.

Finally, the thesis identifies several avenues for future work to further enhance autonomous driving performance, advocating a more proactive approach to decision making and system adaptation. Collectively, the results provide insights into balancing performance, safety, and resource efficiency in autonomous vehicle systems.

List of Abbreviations

ACO	Ant Colony Optimisation
Adam	Adaptive Moment Estimation
AdamW	Adam Optimiser with Decoupled Weight Decay
AI	Artificial Intelligence
ALVINN	Autonomous Land Vehicle In a Neural Network
AMCL	Adaptive Monte Carlo Localisation
APF	Artificial Potential Field
AV	Autonomous Vehicle
AWSim	Autoware Simulator
BEV	Bird's Eye View
BFO	Bacterial Foraging Optimisation
CAN	Controller Area Network
CARLA	Car Learning to Act (Autonomous Driving Simulator)
CE	Cross Entropy
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
DAVE	DARPA Autonomous Vehicle Experiment
DDS	Data Distribution Service
DeiT	Data-Efficient Image Transformer
DNN	Deep Neural Network
DoF	Degrees of Freedom
DRAMA	Driving with Mamba (end-to-end motion planner)
DWA/DWB	Dynamic Window-Based Approach
EKF	Extended Kalman Filter
ELU	Exponential Linear Unit
EyeSim	UWA-designed simulation system
FFN	Feed Forward Network
FPS	Frames per Second
FPS	Farthest Point Sampling
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GUI	Graphical User Interface
HD	High Definition
HiPPO	High-order Polynomial Projection Operator
IMU	Inertial Measurement Unit
ISEAUTO	Intelligent Self-Driving Electric Autonomous Transport
JetPack	NVIDIA JetPack Software Environment for Orin Modules
KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute Dataset

KNN	k-Nearest Neighbours
LAN	Local Area Network
LiDAR	Light Detection and Ranging
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAE	Masked Autoencoder
ML	Machine Learning
MLP	Multi-Layer Perceptron
MPC	Model Predictive Control
MPPI	Model Predictive Path Integral
MSE	Mean Squared Error
NAS	Neural Architecture Search
NAVSIM	Navigation Simulation Dataset
Nav2	Navigation 2 (Framework for ROS 2)
NN	Neural Network
nUWAy	UWA Autonomous Shuttle Platform
ONNX	Open Neural Network Exchange
Orin	NVIDIA Orin Module
PBM	Portable Bitmap
PLC	Programmable Logic Controller
PPO	Proximal Policy Optimisation
Q, K, V	Query, Key, and Value
RAM	Random Access Memory
RANSAC	Random Sample Consensus
ReLU	Rectified Linear Unit
ResNet	Residual Network
RL	Reinforcement Learning
RMSProp	Root Mean Square Propagation Optimiser
RNN	Recurrent Neural Network
RPP	Regulated Pure Pursuit
ROS 1	Robot Operating System Version 1
ROS 2	Robot Operating System Version 2
RRT	Rapidly Exploring Random Tree
RTK	Real-Time Kinematic
RTOS	Real-Time Operating System
SAE	Society of Automotive Engineers
SLAM	Simultaneous Localisation and Mapping
SMAC	Search-based Motion and Control Planner
SSD	Solid-State Drive
SSM	State Space Model
S4	Structured State Space Sequence Model
TEB	Timed Elastic Band
TTS	Text-To-Speech system
UWA	University of Western Australia
ViM	Vision Mamba
ViT	Vision Transformer
ViT-Traj	Vision Transformer for Trajectory Prediction
VMamba	Vision Mamba
VSS	Visual State Space (block in VMamba)
YOLO	You Only Look Once

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This dissertation investigates the application of autonomous driving in real-world scenarios. Two primary domains are identified as potential applications for autonomous shuttle bus technology: campus environments and suburban first- and final-leg services. The growing interest in autonomous technology has led to the classification of automation into five distinct levels, as shown in 1.1. Considerable commercial and academic research exists in this area, but as far as the author can tell, to date, no system has demonstrated level 5 capability, and most deployed systems remain at a level 3. By first understanding how the vehicle reacts in both domains, this work aims to expand and build upon those systems in order to work towards a level 4 SAE system, currently avoiding poor weather conditions and night time driving. To encourage the uptake and acceptance of these systems, they must be shown to be safe and robust in nature, while minimising impact on other road users. The exploration in this thesis starts with common mathematical model based approaches before moving into an end-to-end learning approach that will be discussed more in Chapter 5. In general, the research aims to improve autonomous driving through real-world testing.

FIGURE 1.1: 5 SAE levels for autonomous driving [1].

1.1 Background and Motivation

The development of autonomous systems has progressed rapidly over the past two decades, driven by advancements in sensor technology, computing power, communication speed, and algorithmic efficiency. These improvements have enabled increasingly sophisticated systems capable of analysing complex environments in real time. Autonomous driving has emerged as one of the most prominent applications, with major companies such as Google and Tesla developing self-driving technologies for on-road use. Modern vehicles now feature a range of semi-autonomous safety functions, such as adaptive cruise control, lane keeping, and automatic braking, that represent transitional steps toward fully driver less systems.

Early driver-assistance technologies, such as power steering and anti-lock braking systems, were designed to support drivers rather than replace them. However, the introduction of drive-by-wire systems in the early 21st century marked a significant shift by replacing mechanical linkages with electronically controlled actuators. These innovations allowed for precise electronic control of steering, braking, and acceleration, which is the basis for modern autonomous driving systems. The overarching goal has been to improve safety by reducing the role of human error, the main cause of road accidents worldwide [2, 3].

Despite these advances, fully autonomous driving remains a challenging problem, particularly in dynamic and unstructured environments. While technology continues to mature, the number of road fatalities worldwide remains high, underscoring the need for systems that can operate safely in complex real-world conditions. Research and commercial development efforts are now expanding beyond private vehicles to include autonomous shuttle buses, which are emerging as promising solutions for urban, commercial, campus, and first- and last-mile transport.

In 2020, the University of Western Australia (UWA) acquired its first autonomous shuttle bus from EasyMile, a former leader in this field. The vehicle arrived as a shell with no internal software or limited hardware control that was completely reconfigured in-house by the nUWAy research team. An additional NVIDIA-based computing unit was integrated to provide enhanced GPU processing capabilities and distribute the computational load, enabling support for machine learning and computer vision applications. Since its deployment, the shuttle has been tested on the UWA campus and regularly interacted with students, providing real-world data used to improve perception, planning, and control algorithms. A second EasyMile shuttle was obtained in 2022 for suburban deployment in northern Perth,

allowing testing in more complex mixed-traffic environments. The primary research objective of this work is to develop an intelligent and adaptive system capable of safely navigating pedestrian-heavy and vehicle-dominated areas without disrupting existing users.

The technology has also inspired interest beyond academic research. During an outreach tour of regional Western Australia in 2022, the nUWAy team showcased the shuttle in several towns, where it received enthusiastic support from local communities. In one example, a taxi operator expressed interest in replacing his entire fleet with autonomous shuttles due to a shortage of drivers, highlighting the potential of this technology to benefit remote and regional communities. Although such use cases are currently limited by regulations that require an on-board driver, remote supervision solutions, such as those demonstrated in Norway [4], suggest future pathways toward scalable autonomous transport networks.

Autonomous shuttle development is an active area of research and commercialisation worldwide. The major vendors, including EasyMile, Baidu Apollo, Navya Arma and Olli, have each released fully electric autonomous vehicles aimed at deployment in urban and controlled environments (Figure 1.2). Beyond these commercial efforts, several academic groups are also developing their own systems, such as the ISEAUTO project [5], which integrates research-grade autonomy into custom-built shuttles for deployment in smart-city testbeds.

FIGURE 1.2: Examples of autonomous shuttle buses, from left to right, top to bottom: EasyMile [6], Baidu Apollo [7], Navya [8], and Olli [9].

Overall, the convergence of research and industry efforts demonstrates both the promise and the ongoing challenges of autonomous shuttle technology. The following sections describe the key hardware and sensing components of the UWA shuttle platform, including the driving interface and perception sensors that enable autonomous operation.

1.2 Problem Statement and Research Gap

Although there are many existing systems deployed around the world, the technology seems far from achieving level 5 SAE autonomy level. Many company applications focus on pure GNSS+RTK-based systems (explained further in Chapter 3) which limits their ability to be deployed in unknown or changing environments. Even in research using end-to-end AI models, the deployment of these systems is typically confined to a simulation environment. Other problems with current technology are their response to system failure. In most cases, the vehicle stops and waits for a human operator to resolve the issue before continuing.

These limitations in the current literature and application present gaps for research exploration. There is a need to explore real-world autonomous systems that can present a robust design platform that is not dependent on external sensor data or information for driving. The system also needs to manage the dynamic environment so that in the event of unforeseen hazards, the vehicle can continue to operate. In the context of autonomous shuttle buses, where the potential for deployment for campus and suburban last-leg transportation driving is desirable, addressing these challenges is needed to achieve level 5 SAE autonomy.

This thesis attempts to bridge that gap by exploring how autonomous systems operate in real-time real-world scenarios and enhancing them using modern sequential model architectures in a mixture of experts approach for end-to-end driving that incorporates both vision and LiDAR sensors. To strengthen robustness, deterministic safety checks are added which mitigate unsafe manoeuvres and attempts to eliminate downtime while driving, and reinforcement learning is used to better optimise the training data. Finally, an exploration of temporal and feedback data models is conducted, with the goal of determining if the trade-offs of model size and inference time improve the model significantly. Together, these contributions aim to design a system that is more reliable, reduces downtime, and improves safety to improve the broader societal acceptance of autonomous shuttle technology.

1.3 Chapter Synopsis and Contributions

This thesis is structured into nine chapters, each exploring key aspects of autonomous driving systems with an emphasis on real-world performance, proactive safety, and system reliability. The overarching goal of this research is to investigate methods and architectures that contribute to achieving level 4 SAE autonomy. Each chapter contributes uniquely to this objective, as outlined below.

Chapter 2 details the experimental setup used throughout this thesis. It establishes the physical and virtual environments necessary for experimentation, including the drive system, sensor suite, simulation tools, and data collection, cleaning, and training pipelines. Together, these components form a robust testbed for evaluating autonomous driving systems.

Chapter 3 provides a comprehensive literature review covering foundational concepts in autonomous driving. It discusses several control algorithms, compares mathematical model and learning-based approaches, and examines artificial intelligence methods and their integration into autonomous driving systems.

Chapter 4 presents a preliminary investigation into mathematical model autonomous drive systems for autonomous shuttle operations in dynamic campus environments. The study focuses on safety, reliability, and robustness of the software when navigating crowded real-world settings. A novel ROS2-based software stack was developed, integrating multi modal sensors, navigation algorithms, and vehicle actuation modules with recovery mechanisms to ensure fault tolerance and continuous operation. *Contribution:* Development of a robust ROS2-based software stack that integrates multi modal sensors, navigation algorithms, and vehicle actuation for a campus-based autonomous shuttle bus. The platform enabled system recovery and facilitated real-world benchmark testing. **[Chapter 4]**

Chapter 5 introduces sequential AI models for vision-based end-to-end autonomous driving and compares them to established convolutional models. These models were trained and validated in simulation before being deployed in a real-world suburban environment. The results are assessed in terms of performance, safety, robustness, and user comfort. The investigation identifies limitations related to environmental variation and system generalisation. *Contribution:* Determined suitable autonomous driving approaches for suburban environments. Redesigned existing Transformer and SSM architectures for end-to-end control. Integrated hardware and software for data collection, training and deployment, including safety checks and fail-safe modes for system recovery. **[Chapter 5]**

Chapter 6 addresses limitations observed in the vision-based models of Chapter 5, specifically their sensitivity to lighting and camera failure. A LiDAR-based end-to-end model is introduced as a fallback to improve reliability, along with a joint LiDAR vision model that improves performance under variable lighting conditions and increases the robustness of the system. *Contribution:* Introduced LiDAR-based end-to-end models as fallback systems for vision-based approaches. Developed a novel PointMamba-based LiDAR architecture and a joint LiDAR–vision model to enhance resilience to lighting variation and sensor failure. **[Chapter 6]**

Chapter 7 expands on the temporal modelling of vision-based systems by introducing five new architectures designed to capture feedback and spatiotemporal context. These include feedback and temporal variants of the Transformer and SSM-based vision models. The models introduced a perceived lag due to temporal aggregation. *Contribution:* Developed temporal architectures for end-to-end driving based on the VideoMAE and VideoMamba frameworks. Adapted existing vision-based models for image sequence processing and investigated feedback mechanisms to improve prediction stability. **[Chapter 7]**

Chapter 8 focuses on addressing safety and reliability issues arising from the stochastic nature of end-to-end AI controllers. Two deterministic safety modules are proposed: a LiDAR-based system for rapid obstacle response and a vision-based system for low-level obstacle detection, road position refinement, and user interaction management. In addition, a reinforcement learning framework (PPO) is introduced to refine ground truth trajectories for future retraining. *Contribution:* Designed and implemented deterministic safety systems to mitigate failure states in end-to-end models. Developed and tuned LiDAR and vision-based subsystems for improved safety. Proposed a reinforcement learning framework for trajectory optimisation and improved model generalisation. **[Chapter 8]**

Chapter 9 concludes the thesis by summarising the major findings and discussing pathways toward achieving Level 4 autonomy. The research demonstrates that a hybrid approach combining expert-based systems, multi-sensor fusion, and deterministic safety mechanisms significantly improves reliability and safety. However, temporal models remain challenging because of the increased latency and computational cost. Recommendations for future work include further generalisation testing and refined behaviour decomposition to enhance scalability and robustness.

Chapter 2

Experimental Design and Data Collection

2.1 Introduction

The development of autonomous systems for testing and research requires careful planning and preparation. This chapter summarises the experimental prework completed to make the autonomous shuttle bus operational. It details the vehicle's control system, onboard sensors, driving environment, and simulation systems. In addition, there is a discussion on the data collection, sorting, and training procedures used throughout the research. The goal is to provide the reader with a contextual background for the decisions made in later chapters.

The chapter is organised into six sections:

1. Description of hardware onboard.
2. Overview of the drive system and the interface controls.
3. Introduction to the two driving environments used in experiments.
4. Breakdown of on-board sensors.
5. Description of the simulation systems available.
6. Explanation of the data collection and preprocessing process.

2.2 Vehicle Hardware Setup

2.2.1 PC specifications

The nUWAy vehicles are fitted with two dedicated PCs. The first PC is a NUVO-9531 with an internal CANbus extension card installed. The function of this PC is to communicate to the

low-level control systems on the CANbus as well as any serial devices on the vehicle, and to receive commands from a separate controller. It has a 12th Generation i7 processor, 16GB of DDR4 RAM, multiple gigabit ethernet ports, 6 USB ports, and a Bluetooth / WiFi adapter.

The second PC was originally an Nvidia Jetson Xavier but was upgraded during the project to an Nvidia Orin for a boost in performance and heat management. The Nvidia Orin employs a unified CPU + GPU architecture optimised for mobile autonomous platforms; however, it does come with several limitations including limited internal storage requiring an external M.2 storage module to be installed. In addition, the combined CPU + GPU architecture presents a problem with custom kernel model implementations and difficulties installing libraries that are built for most standard architectures.

2.2.2 High level software stack separation

The primary functions of the software stack are split between the two PCs. The Nvidia Orin handles all GPU operations focusing primarily on the AI models, but also handles the regular data recording, camera drivers, and Velodyne LiDAR communications. The NUVO PC handles all communications with the CANbus system, monitors the remaining LiDARs, and runs the low-level checks and safety features implemented in software, graphical interface, and wireless controller communications. In Chapter 4, much of the software stack for the mathematical model based systems is run on the NUVO PC to reduce system communications as much as possible.

2.2.3 Network architecture

Two network switches handle the network communications between all the IP based devices. The network is connected to a wireless access point TP-Link CPE210 for remote local development and a sierra wireless modem for internet access, required by the RTK+GNSS system and vehicle tracking software.

2.2.4 Control interfaces

Finally, there are two primary ways to control the vehicle. First, an analogue controller onboard, which sends commands directly to motor controllers for speed and steering control. The second control method is through a PS4/PS5 controller, which communicates with the NUVO PC which converts the input to CANbus commands to the system. The Playstation controllers allow for finer control, including a dead-man switch to ensure safe operations at

all times. This interface additionally allows the selection of models behaviours, which are detailed in Chapter 5.

2.3 NUWAy Drive system

The nUWAy autonomous shuttle buses are driven using SICK PLC controllers using the CANbus protocol. The PLCs control motor speeds, accelerations, and steering, as well as provide feedback on a variety of monitored variables. The whole system is powered by 2 48V lithium ion batteries and can reach a maximum speed of 40km/h; however, the current system is limited to a maximum of 20km/h for safety reasons. Two separate interfaces have been developed to communicate with the drive system, an interface board built at UWA by a former student, and CANbus communications from the on board PC.

2.3.1 Interface Board

Upon the acquisition of the first autonomous shuttle vehicle, no official documentation was provided on the communication protocols required to interface with the low-level control system via CANbus. This restriction, due to intellectual property limitations, prevented direct access to the system's internal communication network. As a result, the original research team developed a custom interface board, shown in Figure 2.1, to enable vehicle control without relying on proprietary CANbus commands [10, 11].

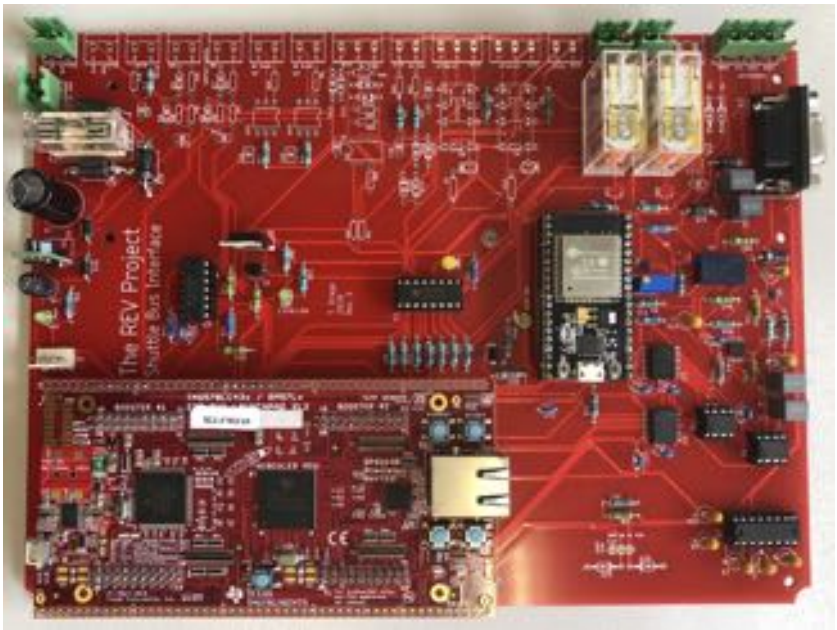


FIGURE 2.1: Interface board developed by Thomas Drage to control the autonomous shuttle bus [11].

The interface board connected directly to the manual control port located in the rear of the vehicle and replicated the analogue control signals generated by the manufacturer's original handheld controller. A serial interface provided communication between the onboard PC and the board, allowing for the transmission of steering and speed commands. The board was built around two micro controllers, an Infineon Hercules and an ESP32, which jointly managed low-level signal generation, rearming functionality, wireless controller input, and a basic state display for system feedback.

Although this solution provided an effective temporary means to achieve autonomous operation, it introduced several limitations and potential failure points. The functionality of the board was limited to the capabilities of the original wired controller, without providing feedback from vehicle systems. This meant that higher-level control software could issue commands, but could not verify their execution or detect hardware faults in real time.

Operational reliability also proved challenging. If the micro controllers did not initialise in the correct sequence, a process with limited visual indicators, the system required a full reboot. Moreover, as the hardware aged, intermittent communication errors occasionally resulted in the generation of stray analogue signals, which could cause unintended vehicle movement even when no drive command was issued. This failure mode was particularly concerning, as it could leave the vehicle unresponsive to stop commands while still outputting control signals.

Due to these safety and reliability concerns, transitioning to a native CANbus communication interface became a priority. Once CANbus access was established, a new software-based control layer was developed to replace the analogue interface system. This updated approach eliminated many of the hardware-related risks, provided robust two-way communication with the vehicle's low-level control unit, and allowed for more precise monitoring and diagnostic capabilities. The details of this updated control system are discussed in the following chapters.

2.3.2 CANbus

In 2022 the team gained access to the related CAN interface by signing an NDA with EasyMile. CAN, Control Area Network protocol, is one of the most popular communication systems in vehicles. First developed in 1986 by Robert Bosch [12], the protocol has become a standard in modern vehicles after a number of revisions. It is commonly referred to as CAN bus, as it uses a network bus line to communicate between multiple different controllers. Access to CAN bus system enabled a redesign of the low-level control architecture, necessitated by the failure

of the interface boards at that time. Greater control of the drive system was achieved with access to the wheel feedback values and acceleration control, if required. It also improves the external agent interactions with control over a number of utilities such as signal lights and bells, which may be useful in a mix of vehicle-human or on-road environments. Due to reduced risk and improved information collection, the interface boards have been removed from all vehicles as of 2023.

2.4 Driving Environment

The performance of an autonomous driving system is highly dependent on the characteristics of the environment in which it operates. Consequently, a comprehensive understanding of the driving environment is essential before developing and running the vehicle's software stack. The experiments in this research were conducted in both simulated and real-world settings in two distinct locations: the University of Western Australia (UWA) campus in Crawley, Perth, and Amberton Beach, Eglington, development in the north of Perth. These two environments provide variable real-world operating conditions that together enable a robust evaluation of the system's performance. This section outlines the key characteristics, constraints, and challenges associated with each site.

2.4.1 UWA Campus

The Crawley campus of the University of Western Australia (UWA) spans approximately 65 hectares and features relatively flat terrain, making it well suited for controlled autonomous vehicle testing. Vehicles are permitted to operate on campus, provided that they adhere to the 5 km/h speed limit and give way to pedestrians. The campus is a dynamic and complex environment that accommodates thousands of students, staff, and visitors daily. Regular pedestrian and cyclist activity, as well as outdoor events and recreational use of open spaces, create continuously changing conditions that challenge perception, planning, and control modules within the autonomous system.

A map of the UWA campus that highlights key locations for the nUWAy autonomous shuttle service is shown in Figure 2.2.

FIGURE 2.2: Key locations for the nUWAy shuttle service on the UWA campus.

The variability of the campus environment provides a prime test ground for autonomous navigation. Temporary signage, event infrastructure, and construction works frequently alter the physical landscape and can obstruct expected travel paths. Furthermore, the presence of dense vegetation, trees, and wildlife such as peacocks introduces dynamic and unpredictable elements to the visual and LiDAR data. Seasonal variations also influence operating conditions: the hot, dry summers can create glare, while the wet winters increase reflections and noise in both camera and LiDAR sensors.

As such, the UWA campus serves as an ideal environment for the development and refinement of autonomous driving algorithms. It combines the safety and accessibility of a semi-controlled space with the complexity of real-world interaction, allowing the evaluation of perception, planning, and control modules under diverse conditions. Furthermore, the deployment of the autonomous shuttle has potential societal benefits by improving campus accessibility and supporting individuals with limited mobility.

2.4.2 Suburban Driving – Amberton Beach

The suburban environment of Amberton Beach presents a different set of challenges compared to those encountered on the UWA campus. In this setting, the autonomous vehicle operates on public roads alongside other road users, which requires the system to manage interactions with live traffic while maintaining safety and efficiency. Although the vehicle's

maximum speed is limited to 20 km / h, maintaining a smooth traffic flow remains a key operational requirement.

Amberton Beach is a rapidly developing suburb with increasing residential density and infrastructure. New schools, parks and commercial areas are being built, leading to increased pedestrian and road user activity. Figure 2.3 shows the route taken by the autonomous shuttle bus as it travels from the Stockland Sales Office at the entrance to the suburb to the beachfront and back. A summary of the key characteristics of this route is presented in Table 2.1.

FIGURE 2.3: Route for the autonomous shuttle service at Amberton Beach.

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

TABLE 2.1: Amberton Beach - Route characteristics [13].

The road environment varies considerably along the route, ranging from narrow single-lane streets to wider, marked, and unmarked dual-lane roads with bidirectional traffic. Several notable features, including roundabouts, intersections, and car parks, are highlighted in Figure 2.4. These elements reflect the characteristics of many suburban areas in Australia, providing a realistic and diverse environment to evaluate autonomous driving performance under real world conditions.

FIGURE 2.4: Eglinton driving path with key landmarks highlighted [13].

To support safe interaction with human drivers, dedicated car bays, highlighted in brown in Figure 2.4, are available for the shuttle to pull over, allowing faster-moving vehicles to pass. These bays are parallel parking spaces, and developing a model capable of autonomously identifying and entering them has been included as a key training objective.

In summary, the suburban environment of Amberton Beach complements the campus setting of the UWA by providing an open, dynamic, and less predictable driving environment. Together, these two test sites present contrasting but equally valuable use cases for evaluating the robustness, adaptability, and safety of the autonomous driving technology.

2.5 Sensors

Sensors are at the core of an autonomous vehicle's ability to perceive and interact with its environment. They define not only what the vehicle can detect but also the range of driving approaches that are possible. The quality, placement and reliability of these sensors directly influence safety, navigation accuracy, and autonomy robustness.

The sensors on the shuttle bus have two primary roles:

1. **Localisation** - Determine the current position and heading of the vehicle in both known and unknown environments.

2. **Perception** - Detecting and understanding the environment around the vehicle as it moves to allow safe and reliable driving.

The nUWAy shuttle buses are equipped with a number of sensors, including wheel encoders, GNSS, IMUs, LiDARs, and cameras. Figure 2.5 provides an overview of the autonomous shuttle bus and the sensor locations, which serves as a reference for the following

subsections. Note that the IMU + dual antenna RTK GPS is located internally inside the roof of the vehicle.

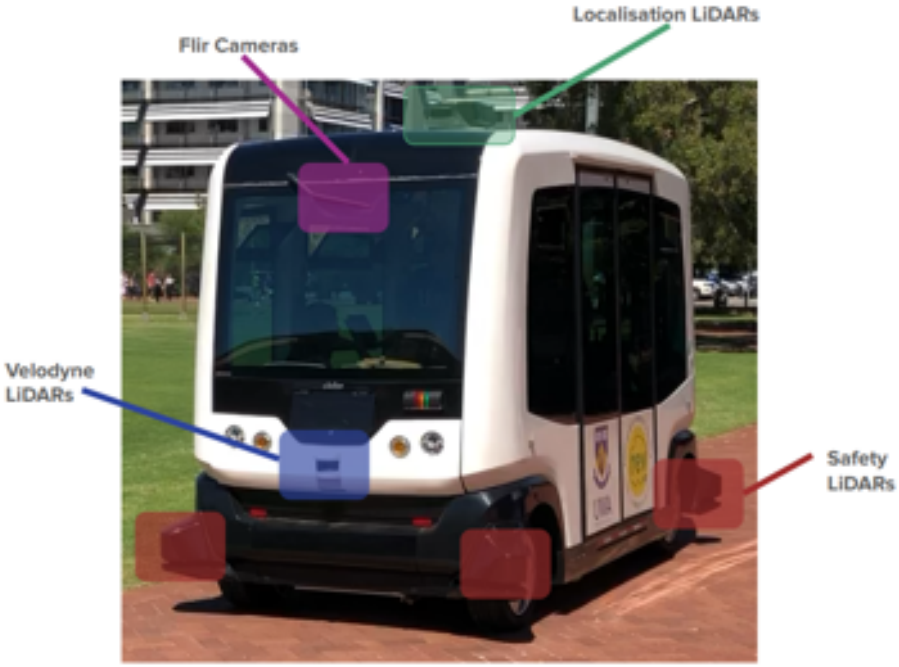


FIGURE 2.5: Autonomous shuttle bus sensor locations.

2.5.1 Wheel encoders

The autonomous shuttle bus is equipped with wheel encoders that determine the speed at which the wheels rotate. Wheel encoders achieve this by counting the number of "ticks" that occur as the wheel spins. In figure 2.6 a simple wheel encoder is shown. In principle, as the wheel rotates, the beam of light from the LED is periodically blocked, which is detected at the photo diode receiver. The number of "ticks" in one full rotation of the wheel can be determined experimentally, and then the number of "ticks" per second can be converted into wheel rotations per second.

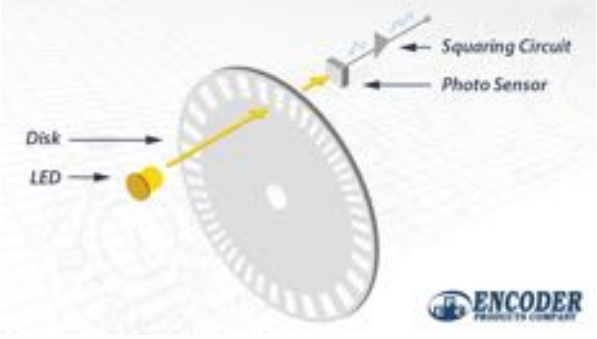


FIGURE 2.6: Example of wheel encoder [14].

There is a relationship between the physical dimensions of a vehicle, its wheel speeds, and its resulting linear and angular velocities. The following equations describe the kinematics of a vehicle with front wheel Ackermann steering, which is the standard model for most road vehicles [15]. The forward velocity of the vehicle is expressed as:

$$v = 2\pi r\dot{\theta} \tag{2.1}$$

where v denotes the forward velocity in metres per second, r is the radius of the wheel in metres, and $\dot{\theta}$ represents the rotational speed of the wheel in revolutions per second. The value of $\dot{\theta}$ can be obtained from the wheel encoder by dividing the number of ticks per second by the total number of ticks per revolution.

The angular velocity of the vehicle is given by:

$$\omega = \frac{v \cdot \sin(\alpha)}{e} \tag{2.2}$$

where ω is the vehicle's angular velocity, in radians per second, α is the steering angle, and e denotes the wheelbase in metres, i.e., the distance between the front and rear axles.

In this case a separate more complex encoder is needed for the steering rack to determine the current steering angle, but the principle is still the same as the previous.

This method provides a basis for autonomous driving but is susceptible to drift errors. As the wheel rotates, small errors in the encoder readings or slippage of the wheel can introduce errors in the linear and angular velocities. These errors will accumulate over time and will result in worse position estimates over time [16]. Figure 2.7 shows the effects of drift over time for a vehicle.

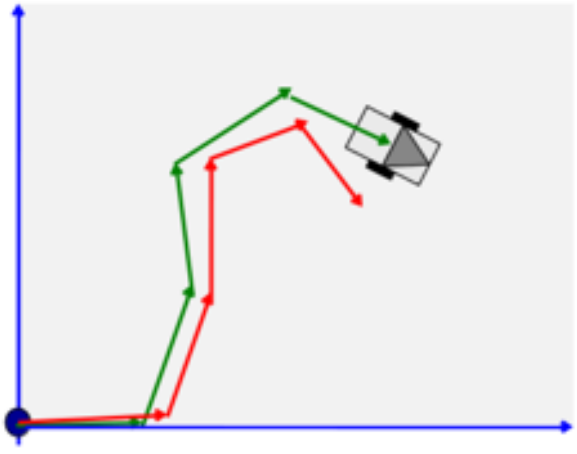


FIGURE 2.7: Dead reckoning with drift error [16].

2.5.2 GNSS + RTK

Most modern devices, from smart phones to vehicles, rely on GNSS (Global navigation satellite system) to determine their position on Earth. A GNSS device receives signals from a collection of overhead satellites launched by many different nations. The signals contain information about the location of the satellite and when the message was sent [17]. The difference in time between when the signal was sent and when it was received can be used to determine the distance from the satellite. One satellite provides enough information to determine where the receiver is on a sphere around the satellite, in order to achieve an accurate local position four satellites are required. Due to inaccuracies in the internal clocks and distances, the system will only be accurate to a few metres. Additional satellite signals can help reduce this error for internal clocks, providing a better measurement accuracy, but will still be limited to a few metres due to propagation delays from various environmental factors.

The accuracy of this measurement can be further improved using RTK (real time kinematic) corrections to give a centimetre-level accuracy [18]. RTK corrections are performed using a second GNSS receiver called the base station [19]. This base station has a known fixed position and does not move. After recording data received from satellites over a period of hours to a day, the base station can determine its fixed location with high precision. The base station can then determine signal corrections for incoming satellite data and send those corrections to local GNSS receivers to improve the precision of their on board systems. Two RTK base stations have been set up at sites where the nUWAy shuttle buses operate and present a viable option for autonomous driving which is discussed in further detail in the following section. Each shuttle bus is equipped with an Ellipse D RTK GNSS + IMU unit shown in figure 2.8

FIGURE 2.8: Ellipse D RTK GNSS + IMU unit used in nUWAy shuttle buses [20].

Despite advances in GNSS + RTK technology, there are still drawbacks to the technology that can impact autonomous driving:

1. When close to large buildings or steel structures, the accuracy of the GNSS will rapidly decrease [21].
2. If the system loses connection to the base station for some reason, the device will no longer receive corrections and the system accuracy will diminish.
3. Poor internet connection or visibility to the base station can result in latency for RTK corrections.
4. GNSS cannot perceive surrounding obstacles by itself, which means that a second source of information is required.

2.5.3 IMU

As mentioned previously, the nUWAy shuttle buses are equipped with a combination RTK GNSS + IMU (inertial measurement unit) for position and heading accuracy. An inertial measurement unit is used to gather information about velocity, acceleration, and heading from external forces as shown in figure 2.9. It uses a three-axis accelerometer and a three-axis gyroscope to give accurate measurements in each of the directions [22].

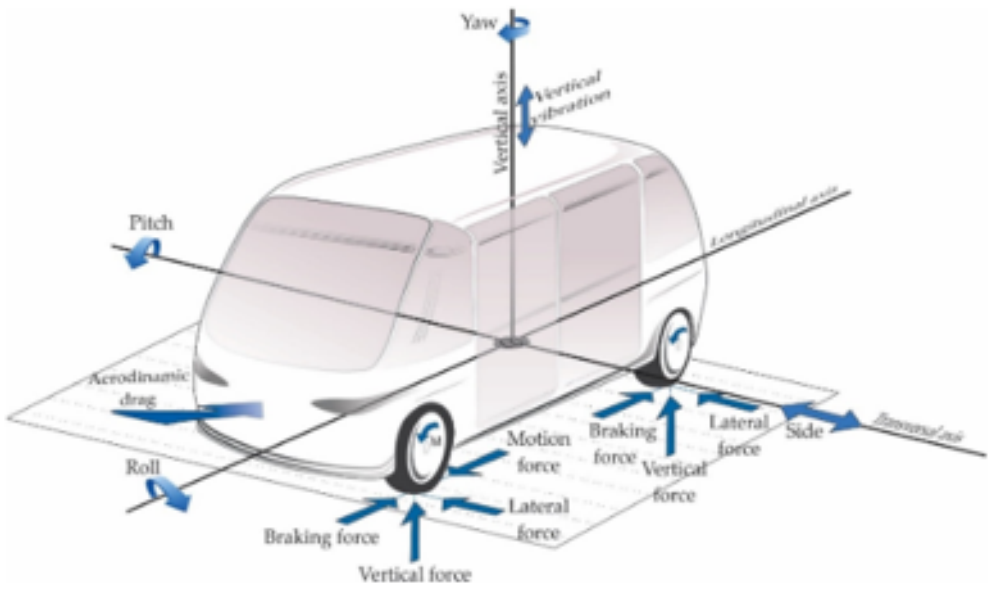


FIGURE 2.9: IMU sensor determining forces on vehicle [22]

However, IMUs do suffer from drift and noise issues; due to inherent sensor biases and noise, they might continue to produce acceleration values when the vehicle is stationary. This results in increasing errors in odometry, leading to system failures in autonomous driving.

To overcome these errors introduced by the IMU unit sensor, information is typically fused together using algorithms such as Kalman filters [23] which corrects for drift issues. The fused information can be used to give a good estimate of position and odometry which is vital in mapping and localisation tasks of autonomous driving.

2.5.4 LiDAR

Light Detection and Ranging (LiDAR) sensors generate point-cloud representations of the surrounding environment using time-of-flight (TOF) measurements. The sensors emit a laser beam, which reflects back off surrounding surfaces and is detected again back at the sensor. The distance to obstacles can be computed as

$$d = \frac{tc}{2} \tag{2.3}$$

Where t is the round trip time of the light pulse and c is the speed of light. The value is divided by two to account for the outbound and return distance.

There are a number of different styles of LiDARs, including single-layer and multilayer sensors [22]. Multilayer LiDARs are the most commonly used in autonomous vehicles for their ability to generate 3D features. Figure 2.10 shows an example of the 16-layer Velodyne Puck that is commonly used in driving applications.

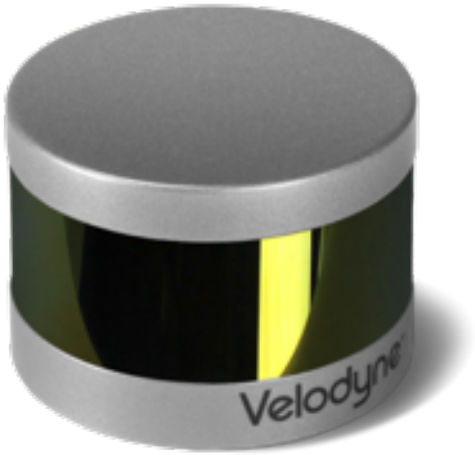


FIGURE 2.10: Example of Velodyne puck used on the nUWAy Shuttle buses [24].

The field of view of these sensors can range widely, the vertical view ranging from 20°to 45°, the horizontal view ranging between 270°and 360°and operating distances between 0.5 m and 100 m [22, 24].

The nUWAy shuttle buses use a combination of LiDAR sensors for different applications. Four single-layer SICK LiDARs at each corner, which trigger an estop on the local PLCs and disable driving when obstacles approach too closely. Two 16-layer Velodynes at the front and back of the shuttle bus which allow the vehicle to generate detailed point clouds of close obstacles and two 4-layer SICK LiDARs at the top of the vehicle for localising to objects up to 100 metres. A point cloud example from the autonomous shuttle bus is shown in figure 2.11.

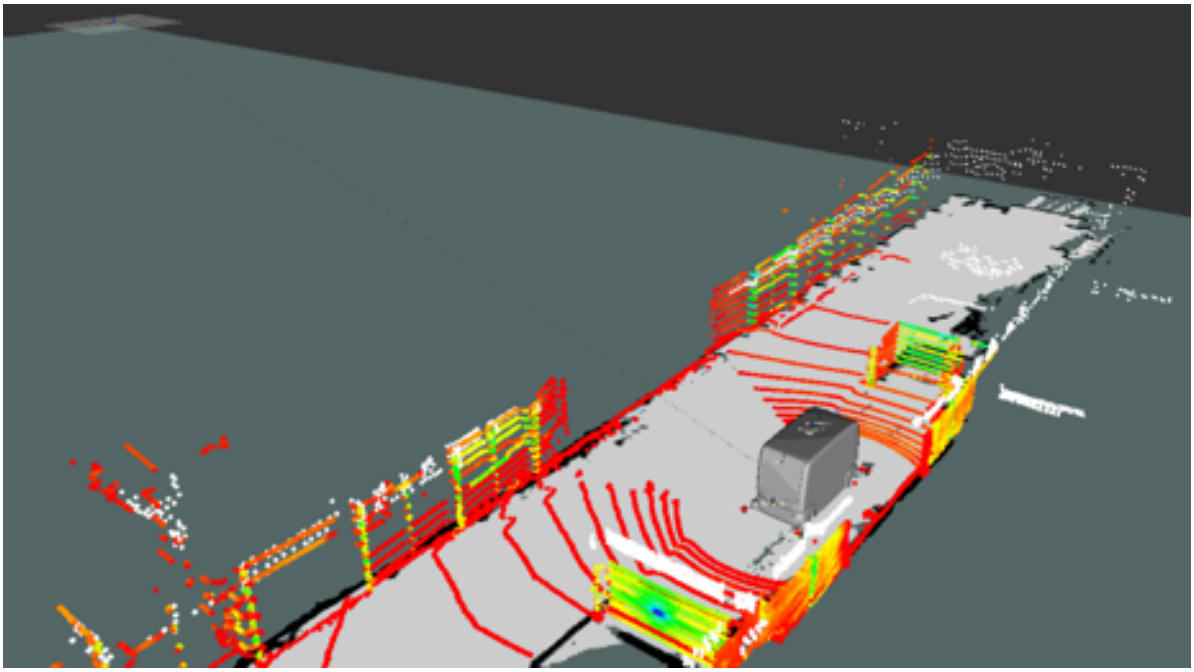


FIGURE 2.11: LiDAR image from nUWAy shuttle bus in corridor [25], white dots are from the localisation LiDARs while the coloured values are from the Velodyne LiDARs.

LiDARs offer a number of advantages in autonomous driving, including robustness for fluctuating lighting conditions, fast distance calculations, and a wide range of vision (depending on the mounting) [26]. However, these sensors also face a number of challenges: multilayer LiDARs produce large amounts of data, especially when you get 64 and 128 layers, which can make them computationally expensive for real-time applications. Their performance also degrades in heavy rain and fog, and increasing distance and/or poor mounting can cause sparse feature maps [27].

2.5.5 Cameras

Cameras present an alternative method of perception of the world for autonomous vehicles (AV). The visual information from cameras makes it easier to recognise unique obstacles in the field of view that can be used in conjunction or in place of a LiDAR sensor. There are two types of cameras typically used in the design of AV, monocular and stereo [22].

Monocular cameras capture 2D images in either colour or grey scale. Using computer vision techniques, edge detection, colour histograms, and erosion / dilation obstacles within the image can be detected. A limitation of monocular cameras is that they do not have depth information, which is needed to determine how far away something is. A reference value, with known size and shape, can be used to estimate the size of surrounding obstacles in an image frame using the techniques presented in [28]. However, these methods are typically inaccurate and not suitable for driving tasks.

Stereo cameras provide a stronger alternative solution as they mimic the binocular vision of humans to gain a sense of depth by placing two cameras a fixed distance apart. The disparity between the two images from each camera is compared, and a depth estimate can be made with higher accuracy [29]. Due to how these cameras operate, there is usually a significant trade-off with computational power in order to process these images. In addition, the accuracy of the stereo camera drops in low-texture or poor-light environments. Although most modern stereo cameras process the information locally before sending it to the host device, there may be delays in the image data.

As cameras are cheaper they have become a popular choice, but there is a trade-off between using cameras over LiDARs when it comes to autonomous driving which typically comes down to processing speed, image detail and field of vision. The delay for stereo cameras, as well as the extra processing power, to determine obstacle distances can limit their usefulness in safety critical tasks at high speeds. Alternatively, their ability to pick out unique obstacles quickly and accurately makes them a great choice in a number of complex driving scenarios. Therefore, many companies are working on autonomous systems that incorporate both LiDAR and camera-based systems to build a stronger overall solution.

The nUWAy shuttle buses are equipped with two FliR Point Gray Grasshopper cameras as shown in Figure 2.12. These cameras produce simple grey scale images, shown in figure 2.13, at a rate of 20 Hz at both the front and rear of the vehicle.

FIGURE 2.12: Flir point grey cameras used in nUWAy shuttle buses [30]

FIGURE 2.13: Sample image from the nUWAy shuttle bus in Amberton Beach.

The selection of sensors presented on the nUWAy shuttle buses provides an opportunity to explore a number of autonomous driving approaches. The following sections discusses how the previously discussed sensors can be used for localisation and perception tasks that are critical for autonomous driving.

2.6 Simulation

Since this research focuses on autonomous vehicles intended for deployment in real-world environments, it is essential that all algorithms and driving systems undergo testing before field use to prevent potential incidents. The initial stage of this testing pipeline typically involves evaluation within a controlled simulation environment, where system behaviour can be safely examined.

This project employed two simulation systems for this purpose: EyeSim [31] and AWSIM [32]. The EyeSim simulator provides a simplified testing environment that enables rapid verification of model performance. Although it lacks the complexity and realism of advanced driving simulators, EyeSim remains useful for confirming model viability. A visual representation of the system is shown in Figure 2.14.

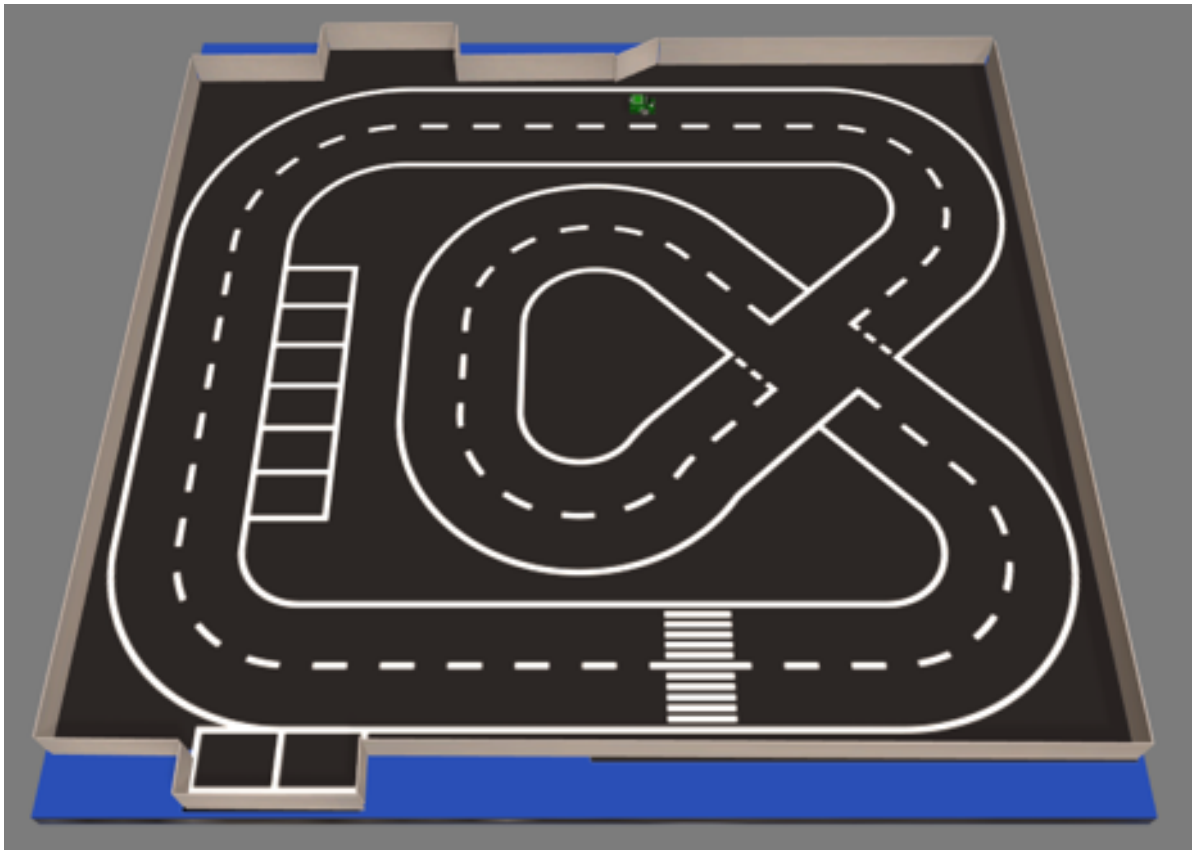
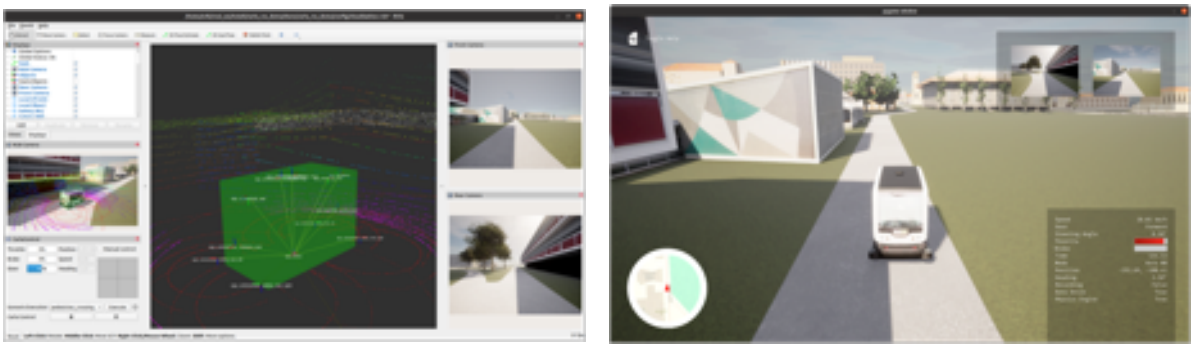


FIGURE 2.14: EyeSim simulation system developed by Professor Thomas Bräunl [31].

A more sophisticated simulation environment, such as CARLA [33], shown in Figure 2.15, could alternatively be used to validate image-based models under more realistic conditions. However, given that the performance of these models were already evaluated through other

complementary methods, an additional simulation stage using CARLA was deemed unnecessary.



(A) RVIZ view of CARLA simulation.

(B) CARLA world view of the UWA campus.

FIGURE 2.15: CARLA simulation system.

Although the EyeSim simulator includes a LiDAR module, substantial modification would be required to replicate the 3D LiDAR point clouds produced by the nUWAy shuttle buses. Furthermore, the virtual environment would need to be manually reconstructed to represent the Eglinton testing site.

As an alternative, the UWA team has already developed a map of Eglinton and the associated vehicle in AWSIM [32]. This simulation system is also more sophisticated with a stronger physics engine that already supports the existing 3D LiDAR sensors needed, making it ideal to test the LiDAR based models. A visualisation of the AWSIM environment is shown in Figure 2.16.

FIGURE 2.16: AWSIM simulation system with nUWAy shuttle operating on the Amberton Beach map.

2.7 Data Collection

The development of robust autonomous driving systems is highly dependent on the availability of high-quality, multi modal data sets that capture a wide range of environmental conditions and driving behaviours. The data collected for this work was used to train, validate and evaluate vision- and LiDAR-based models, as well as to support future reinforcement learning and mapping tasks. All data was collected manually driving the vehicle using a PlayStation controller to replicate the same speed and steering commands expected from the AI models. Existing datasets predominantly use colour-based images from a car perspective and 32 or higher layer LiDAR datasets, which are incompatible with the dual 16-layer perspective of the current vehicle. Early testing showed that maintaining the same camera position was imperative for current imitation-based models. This section describes the data collection framework, the processes used for data sorting and cleaning, and the composition of the resulting datasets.

2.7.1 Data Capture Framework

The ROS (Robot Operating System) architecture provides an effective and flexible framework for data acquisition through the use of `rosbag` [34]. This library enables recording of all messages that arrive from system topics, either completely or as a user-defined subset. In addition, the system can be configured to record data based on specific trigger events, allowing the targeted capture of relevant scenarios. This functionality proved particularly useful for analysing safety-critical events such as emergency stops (`E-stops`). A "snapshot mode " was configured to capture the preceding ten seconds of data from all active topics when a trigger event occurred, providing a detailed temporal context of the incident while keeping file sizes manageable.

Once recorded, the data could be reviewed and visualised using tools such as Foxglove Studio, which the REV research team extensively used, as noted in [35]. Foxglove facilitated efficient inspection of LiDAR trigger points, verification of sensor synchronisation, and general data quality assessment.

2.7.2 Data Sorting and Behavioural Categorisation

Beyond event analysis, the collected data formed the foundation for training autonomous driving models. The initial sorting was performed manually using a Python script developed by Zhihui Lai, who organised the raw recordings into behaviour-specific categories that could

then be used to define lane following, parking (pull-in), and reversing manoeuvres. Each data point was converted into a NumPy array containing the image and related metadata before being stored in the corresponding directories for model training. The breakdown of behavioural categories is shown in Table 2.2. Certain specialised behaviours, such as dual steering, were collected but excluded from the set of models due to mechanical limitation at the time of testing.

Lane-Following Behaviours	**Pull-in Behaviours**	**Reverse Behaviours**
lane_bay_pass	pullin	reverse
roundabout_straight	pullin_stops	pullout_stops
lane_following		reverse_manual
intersection_lane_following		
startpoint_out		
startpoint_in		
carpark_pass		
roundabout_right_turn		
lane_empty_bay		
lane_empty_bay_first_half		
lane_empty_bay_second_half		
pullout		

TABLE 2.2: Unique driving behaviours across lane-following, pull-in, and reverse datasets.

2.7.3 Data Cleaning and Quality Control

Given that the vehicle was manually driven during data collection using a PS4 / PS5 controller, the input of steering and speed from the controller served as ground truth control values for supervised learning. However, this approach introduced potential noise due to variations in driver performance. Instances involving sharp turns, inconsistent speed control, or emergency stops were identified and removed to ensure that the models did not learn unsafe or erratic behaviours. Additional data surrounding these events were also excluded to prevent contamination of otherwise valid driving sequences.

Lighting variations presented another significant challenge for image-based perception. Figure 2.17 illustrates the impact of changing light conditions, caused by cloud cover, time of day, and camera exposure, on the captured frames. To improve robustness, the dataset

intentionally incorporated data collected across different times and seasons, ensuring that the trained models could generalise to a wide range of illumination conditions.

FIGURE 2.17: Examples illustrating how cloud cover and poor exposure can lead to darkened image frames.

2.7.4 Dataset Composition and Applications

The curated datasets served multiple purposes. They supported end-to-end learning for lane-following and manoeuvring tasks, enabled the construction of high-resolution maps via LiDAR-based SLAM, and provided a foundation for reinforcement learning and GPS + RTK-based navigation. The mapping data required additional care due to the accumulated odometry drift inherent in SLAM processes. To mitigate this, smaller local maps were generated and subsequently manually aligned to produce a globally consistent and geometrically accurate map suitable for outdoor driving experiments.

In total, the image dataset comprised 1,121,757 samples, divided into 807,665 for training, 201,916 for validation, and 112,176 for testing. The video-based models were trained on 546,445 eight-frame sequences, divided into 393,440 training, 98,360 validation and 54,645 testing samples. The LiDAR dataset comprised 327,971 samples, divided into 236,139 training samples, 59,035 validation samples, and 32,797 test samples.

A detailed breakdown of parking-bay data is presented in Table 2.3, showing the number of valid pull-in, reverse, and pull-out manoeuvres available per bay after data cleaning which is used in later chapters to show general model performance in seen and unseen scenarios.

Bay	Pull-in Counts	Reverse Counts	Pull-out Counts
bay_1-1	1	1	1
bay_1-2	11	9	6
bay_1-3	4	4	4
bay_2-1	12	10	5
bay_2-2	16	16	18
bay_3-1	2	7	4
bay_3-2	2	4	5
bay_4-1	2	4	5
bay_4-2	2	3	1
bay_4-3	10	10	10
bay_5-1	2	1	2
bay_5-2	1	0	0
bay_6-1	6	3	2
bay_6-2	10	2	6
bay_7-1	1	2	3
bay_8-1	4	4	4
bay_8-2	4	3	5
bay_9-1	0	0	0
bay_9-2	6	0	1
bay_9-3	5	3	6
bay_9-4	2	4	4
bay_10-1	8	2	3
bay_11-1	5	5	4
bay_11-2	16	18	16
bay_11-3	7	8	9
bay_11-4	4	3	2
bay_12-1	2	1	1
bay_13-1	3	3	3
bay_13-2	9	11	14
bay_14-1	2	2	1
bay_14-2	15	12	13
bay_15-1	2	1	0
bay_15-2	6	9	4
bay_15-3	0	1	0
bay_16-1	0	0	0

TABLE 2.3: Counts of pull-in, reverse, and pull-out manoeuvres by bay to be used as training data after data cleaning.

2.7.5 Summary

Overall, the data collection and preprocessing pipeline established a solid foundation for subsequent model development and evaluation. The combination of diverse environmental conditions, structured behavioural labelling, and careful quality control ensured that the resulting datasets were well-suited for both perception and control learning tasks. This data set formed the foundation for experiments presented in later chapters, enabling a consistent and reproducible framework for autonomous driving research in unstructured outdoor environments.

Chapter 3

Literature Review

3.1 Introduction

This chapter reviews the current state of autonomous driving technologies, encompassing both mathematical and AI-based approaches. It begins by describing the fundamental perception-planning-control framework that underpins most autonomous systems, followed by an overview of the commonly used software framework, ROS 2. Subsequently, the chapter explores key mathematical models applied in mapping, localisation, and control, providing insight into their practical use and limitations.

The discussion then transitions to modern AI-driven techniques, with particular focus on neural networks and their role in advancing autonomous vehicle performance. Recent developments in deep learning architectures are also examined, highlighting how these methods are increasingly replacing or augmenting traditional algorithms.

By combining theoretical foundations with a review of established and emerging methods, this chapter provides the necessary context to understand the models and methodologies developed in the subsequent chapters, which form the main contributions of this thesis.

The chapter is organised into the following sections:

1. Overview of autonomous driving frameworks and a description of the ROS 2 software environment and its relevance to autonomous systems.
2. A summary breakdown of autonomous driving approaches.
3. Expansion of navigation-based approaches.
4. Review of mathematical algorithms for mapping, localisation, and control.
5. Examination of machine learning and AI-based methods, including neural networks.

6. A detailed review of neural networks and their application to autonomous driving.

3.2 Autonomous Driving Concepts and Software

The development of autonomous systems is commonly organised into hierarchical layers, each responsible for a specific function within the overall driving pipeline. The most widely adopted framework is the *perception–planning–control* architecture, as reflected in the ROS2 Navigation Stack [36, 22].

In the first stage, **perception**, the autonomous system collects data from the sensors onboard to interpret its environment and estimate its state. Sensor input may originate from individual devices such as LiDAR, GNSS, or cameras, or be produced through sensor fusion techniques. The sensor data is then processed into meaningful representations, including pose estimates, occupancy grids, or high-definition maps.

The second stage, **planning**, uses this environmental information to generate a feasible and safe path from the current pose to a defined goal. This may take the form of a sequence of way points or a continuous trajectory. Classical algorithms such as Dijkstra's and A* are widely used for global path planning due to their reliability and predictable outputs.

Finally, the **control** layer converts the planned trajectory into executable commands, typically steering angles and velocity set points, which are transmitted to the drive-by-wire actuators to control the vehicle's motion. A visual representation of this hierarchical architecture is shown in Figure 3.1.

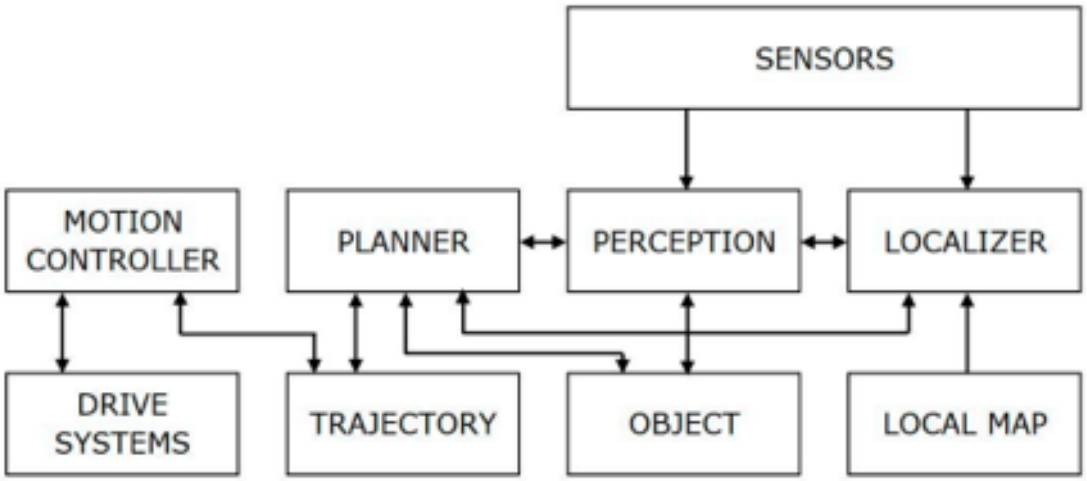


FIGURE 3.1: Visual representation of the autonomous driving pipeline [22].

ROS and ROS2 provide a **modular framework** that enables each layer of the pipeline to be independently designed, tested, and replaced. This modular architecture, in conjunction with pre-built packages and development tools, reduces engineering effort and accelerates system deployment. However, ROS2 does not natively offer the real-time guarantees required for safety-critical applications.

3.2.1 Navigation Stack - Perception, Planning and Navigation

The navigation stack is the core architecture that enables autonomous vehicles to perceive, plan, and act within their environment. Its key components include localisation, mapping, global planning, local planning, obstacle detection, and recovery. Localisation estimates the vehicle's position using sources such as GNSS coordinates, inertial measurements, or onboard sensors. Although GNSS can provide accurate outdoor positioning, indoor environments require sensor-based techniques such as Simultaneous Localisation and Mapping (SLAM), which builds a map and localises the vehicle simultaneously.

Autonomous systems vary widely, from proprietary software developed by companies such as Tesla, Google, and EasyMile, to open-source frameworks like ROS (Robot Operating System), which is widely used in research for its modularity and flexibility. In known environments, the navigation stack typically progresses from mapping and localisation to global path planning, then local motion control, and finally recovery strategies in the event of unexpected events. Figure 3.2 illustrates a typical navigation stack and how its components interact.

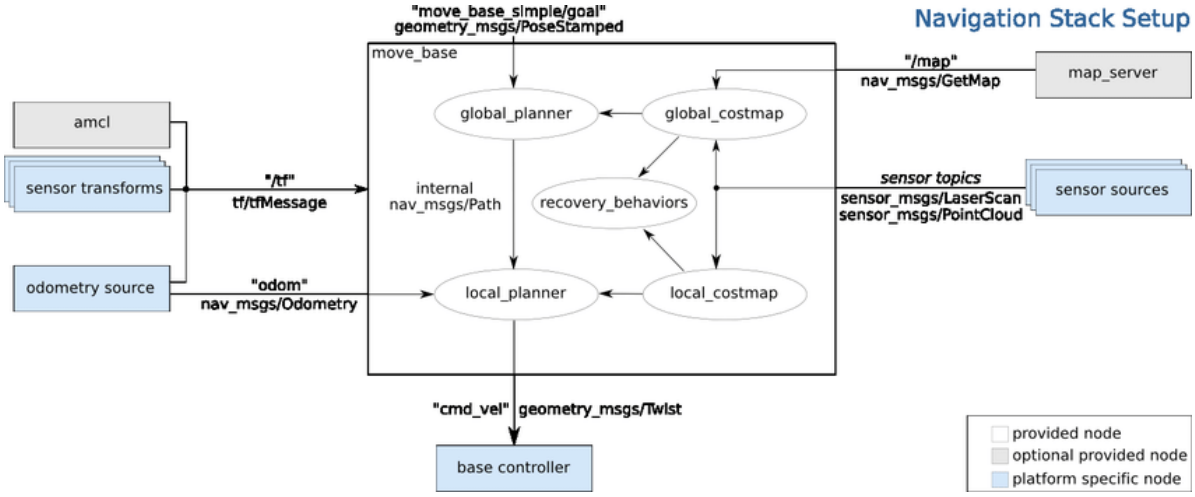


FIGURE 3.2: Typical navigation stack used in ROS 2 for autonomous driving [37].

Once the vehicle knows its position, the "global planner" determines an overall path from start to goal [38]. The global plan defines the high-level route without taking into account dynamic obstacles or shifts in the static environment. The "local planner" converts this high-level path into actionable motion commands, continuously adjusting the vehicle's trajectory based on real-time sensor input and the local occupancy grid. Local control algorithms optimise motion for safety, smoothness, and efficiency.

The "occupancy grid" (or cost map) supports local planning by representing free and occupied space around the vehicle [39]. Layers typically include the global map from the mapping stage, a dynamic obstacle layer based on sensor input, and an inflation layer that adds safety margins around obstacles. These layers collectively allow the local planner to maintain collision-free motion while respecting the vehicle's kinematic constraints.

The overall performance of autonomous navigation depends on the seamless interaction of these subsystems. Proper integration and tuning of mapping, localisation, planning, and obstacle avoidance modules are essential for robust and reliable operation [40].

3.2.1.1 Object Detection

The first step in object detection involves processing raw sensor data into a form suitable for perception and decision making. For LiDAR sensors, this process is relatively straightforward. LiDARs emit light pulses and measure the reflected signals to determine the distance and direction of surrounding obstacles. On larger vehicles, multiple LiDAR sensors can be used simultaneously, with their data fused and projected into a two-dimensional representation of the environment. From this two-dimensional view, the distances to surrounding objects can be used to determine a collision-free path. This is typically achieved by modelling the vehicle as a single point and evaluating whether its planned path intersects any detected obstacles. Since real vehicles occupy a physical volume, an "inflation zone" — illustrated in Figure 3.3 — is introduced around each obstacle to approximate the vehicle's size and safety buffer. The inflation zone is then used to assess potential collisions during motion planning.

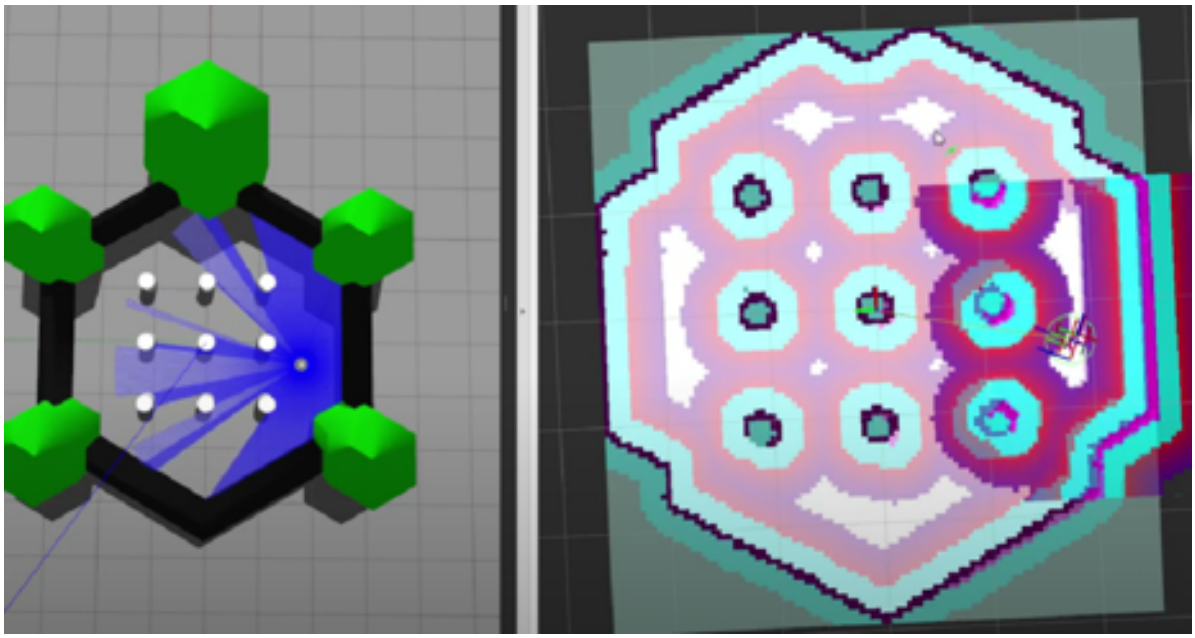


FIGURE 3.3: Getting started example from NAV2 showing inflation zone [41]. The inflation values decrease from light blue to purple, following a Gaussian distribution that models the safety buffer around detected obstacles.

Three-dimensional LiDAR systems, by contrast, generate dense point clouds that provide richer spatial detail and enable more precise characterisation of surrounding objects. However, the high density of these point clouds demands significant computational resources, which currently exceed the capabilities of the autonomous shuttle bus when accounting for all concurrently running systems.

Cameras also serve as an important source of environmental information. However, estimating object distances from monocular (single-lens) cameras is inherently challenging, as it often relies on reference objects of known dimensions to infer depth. Stereo cameras overcome this limitation by employing two lenses to compute depth through triangulation, similar to human binocular vision. Many modern stereo camera systems are capable of generating point clouds directly from captured imagery; however, such systems are not currently available on the vehicle and are considered unnecessary due to their additional computational overhead in image processing.

Regardless of whether LiDAR or camera data are employed, detected objects are typically represented within an occupancy grid. This grid serves as a critical input to the local planning and mapping algorithms, enabling the vehicle to make informed navigation and control decisions based on its perception of the surrounding environment.

3.2.1.2 Occupancy Grid

Although not a novel concept, both the Robot Operating System (ROS) and the navigation stack for ROS2, NAV2, implement the concept of an "occupancy grid" (or "cost map") to support autonomous navigation. The occupancy grid provides a spatial representation of the vehicle's environment, distinguishing between 'occupied' and 'free' areas within the robot's sensing and planning range [39].

The cost map plays a central role in determining how the vehicle perceives and interacts with its surroundings. It is typically composed of multiple layers, each encoding different aspects of the environment. The primary layers include the "global map", constructed during the mapping phase, and the "obstacle map", which dynamically updates based on real-time sensor data. An "inflation layer" is also applied, expanding occupied regions to create safety margins around obstacles and prevent potential collisions.

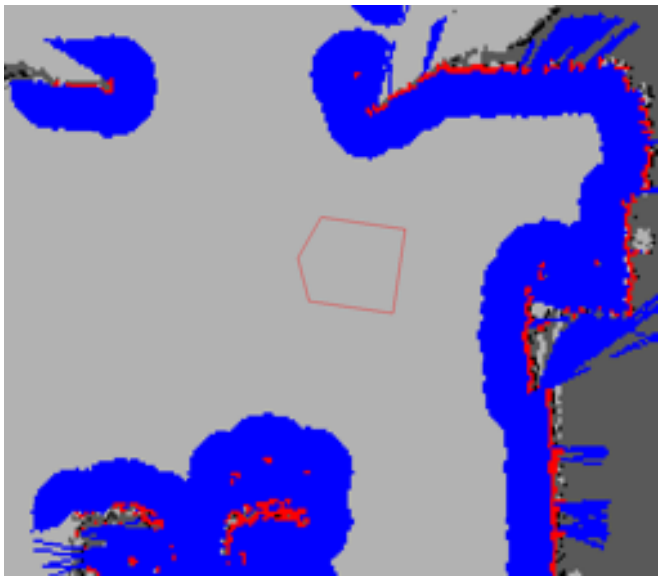


FIGURE 3.4: Example of a cost map from ROS 1. Red pixels indicate occupied (lethal) obstacles, while blue pixels represent inflated safety margins around those obstacles [42].

Although this research primarily focuses on the motion of autonomous vehicles and how they interact with the environment, it is important to recognise that overall navigation performance depends on the interaction of multiple components within the navigation stack. Each subsystem, from mapping and localisation to global and local planning, contributes to the overall driving behaviour, and their combined tuning can significantly improve or degrade autonomous navigation performance [40].

Building upon this standard navigation architecture, the Navigation2 (NAV2) framework in ROS2 extends these capabilities by providing modular, flexible, and high-performance

implementations of mapping, localisation, planning, and control. The following section explores NAV2 and two of its core tools that enhance autonomous navigation.

3.2.1.3 Mapping and Localisation

As mentioned earlier, an effective technique for autonomous navigation involves mapping an environment and subsequently localising within that map by integrating multiple data sources, such as LiDAR scans, GPS, and wheel odometry, into a unified and consistent estimate of position. This process is commonly achieved through a method known as "Simultaneous Localisation and Mapping" (SLAM), which performs both mapping and localisation concurrently.

The SLAM process begins with the robot or vehicle capturing an initial scan from its starting position. This scan is typically compiled into a two-dimensional map, as the storage and computational demands of three-dimensional mapping grow exponentially with area coverage. The vehicle then moves through the environment, taking additional scans at regular intervals based on elapsed time or distance travelled [43]. Although each new scan differs slightly from the previous one, they share overlapping features. By combining "scan matching" techniques with an "Extended Kalman Filter" (EKF) [44] to estimate positional changes, the system incrementally aligns each new scan with the existing map, much like fitting together pieces of a puzzle.

To further enhance the robustness of the mapping, the concept of "loop closure" is introduced [45]. When the vehicle revisits a previously explored location, accumulated odometry drift may cause current map features to become misaligned. However, the features observed in this region should closely correspond to those recorded earlier. By aligning these features, the system corrects for drift throughout the map, tightening, and refining the final representation (Figure 3.5).

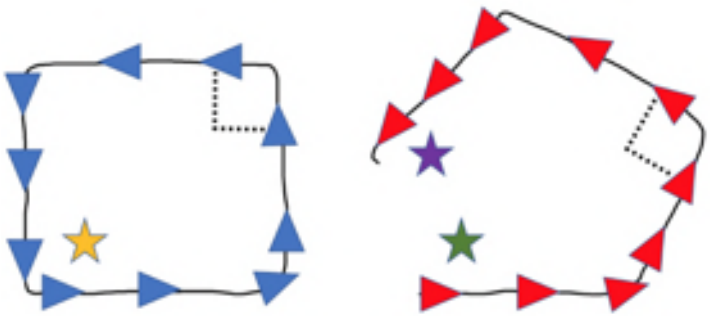


FIGURE 3.5: Illustration of loop closure [46], on the right is the pose graph with the two stars representing the same feature before closure and on the left is the result of loop closure, joining the previously seen feature.

As the robot or vehicle continues to explore its surroundings, the global map becomes increasingly detailed and robust. Once completed, the map can be stored and later reloaded for use in subsequent navigation tasks. During localisation, live sensor data from the EKF and LiDAR (or other onboard sensors) are compared against the stored map to determine the vehicle's current position, enabling continuous localisation as it moves through the mapped environment.

SLAM has become a widely adopted technique for indoor mapping and navigation. Many domestic robots, such as vacuum cleaners, rely on SLAM to efficiently map and navigate household environments. Figure 3.6 shows an example of an Ecovacs T8 robotic vacuum that has mapped a residential space using SLAM.

The next stage in autonomous driving builds upon this foundation through "global planning", which determines the optimal route a vehicle or robot should take from its current location to a specified goal.

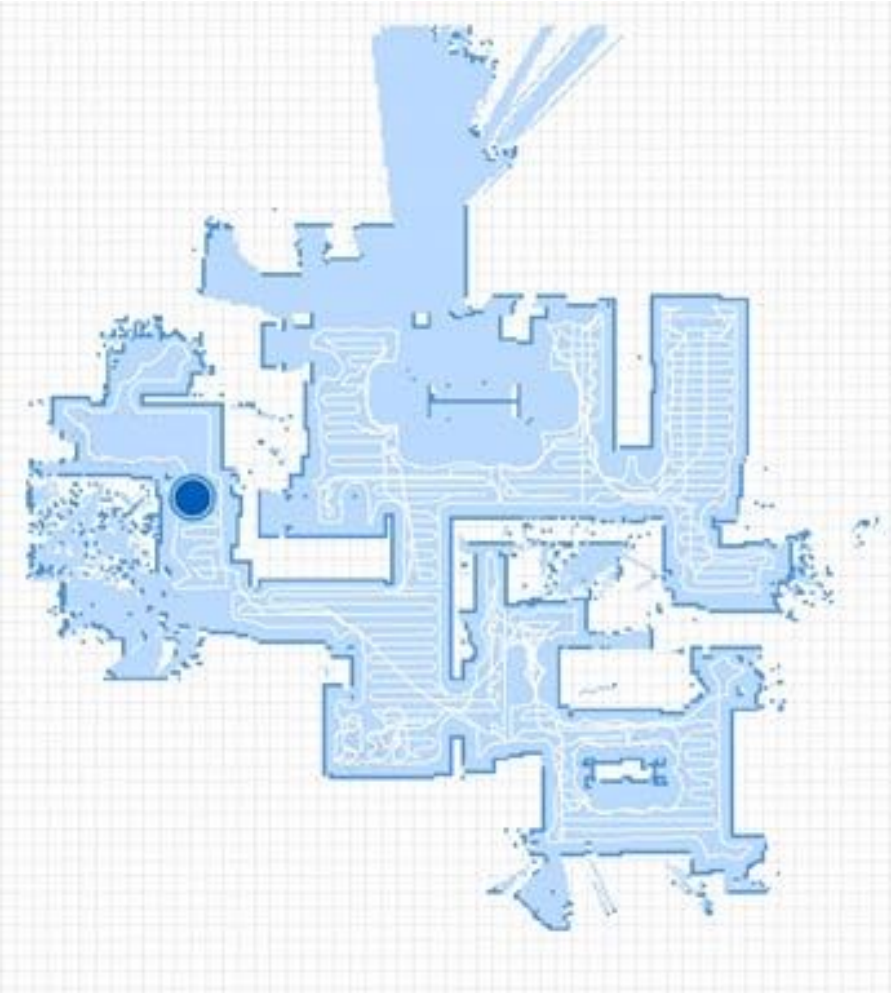


FIGURE 3.6: Example of an Ecovacs T8 robotic vacuum using SLAM to map a household environment.

3.2.1.4 Global Planner

Once a map has been generated and the system can reliably localise within it, a "global planner" is used to determine the optimal path between a start and goal position. This global path defines, at a high level, how the robot or vehicle should travel from point A to point B based on the static environment represented in the map, without accounting for dynamic obstacles or temporary changes in the environment. A variety of algorithms can be employed for global path planning, with two of the most commonly used being "Dijkstra's algorithm" [47] and the "A-star (A) algorithm" [48], or variations of them. Both methods operate by discretising the map into a set of nodes, where each node corresponds to a free (navigable) location in space. The planner then computes the optimal route between the nodes by minimising a cost function that represents the distance or time of travel.

Dijkstra's algorithm systematically explores all possible nodes to find the shortest path to the goal, guaranteeing an optimal solution but at the expense of computational efficiency. In contrast, A * introduces a heuristic, typically the Euclidean distance to the goal, that first guides the search for the most promising nodes. This heuristic enables A* to converge more quickly while still ensuring optimality under admissible heuristic conditions. As a result, A* is generally preferred for real-time navigation tasks, as it achieves comparable path quality with significantly reduced computational cost.

3.2.1.5 Local Planner

Once the global path has been generated, the vehicle must execute this route by using a "local planner", which translates the high-level path into real-time motion commands. The local planner continuously adjusts the vehicle's trajectory to follow the global path while avoiding obstacles and ensuring smooth, safe movement. Unlike the global planner, which operates on a static map, the local planner must respond to both static and dynamic obstacles that may not have been present in the original global map.

To accomplish this, the local planner relies on sensor-based perception systems—typically LiDARs or cameras—to detect nearby objects. These detections are then transformed into obstacle representations within the local frame. Depending on the level of sophistication of perception, these obstacles can be further classified into specific categories, such as pedestrians, vehicles, or traffic signs, often through computer vision or machine learning-based object recognition methods.

The processed sensor data is integrated into a "local occupancy grid", which represents the immediate environment around the vehicle. This grid provides the local planner with a cost-aware view of free and occupied space, enabling path adjustments to maintain collision-free navigation. Various local planning algorithms, such as the dynamic window approach (DWA), time elastic band (TEB), and regulated pure pursuit (RPP), build on this grid to optimise motion based on criteria such as safety, smoothness, and efficiency. Chapter 3 discusses several of these local planning methods in greater detail.

3.2.2 ROS - Robot Operating System

The robot operating system, often referred to as ROS, is a middleware framework that deals primarily with inter-process communication, but also provides resources management and a large library of pre-built packages [49], [50]. Together, these packages and standardised tools in communication, scheduling, and execution enables fast development of complex robot applications.

The core concepts in ROS are nodes, topics, services, and actions. The useful code runs within a node and can be started at any time during operation without disrupting other functionality of the system. The nodes can then interact with each other through the three different messaging services provided. Figure 3.7 shows the configuration of the first two messaging systems, topics, and services. When a node wants to make certain information publicly available it will setup a topic to publish to, other nodes can then subscribe to this topic to receive the information. If multiple nodes need to do the same calculation or require a certain physical action to be accomplished then a service can be used. For a service, a client node will send a request to an available service and then wait for a response from the server node.

FIGURE 3.7: ROS messaging system runs user code on nodes and provides topics and services for communication [49].

The final messaging system is the action server, as illustrated in Figure 3.8. It combines the concepts of topics and services to provide constant feedback while a request is being completed. For example, if a vehicle is navigating from A to B, it may provide constant feedback to the client node on how close it is to the goal. This feedback can then be used to determine whether satisfactory progress is being made or if the request has failed, i.e., an obstacle blocking the path.

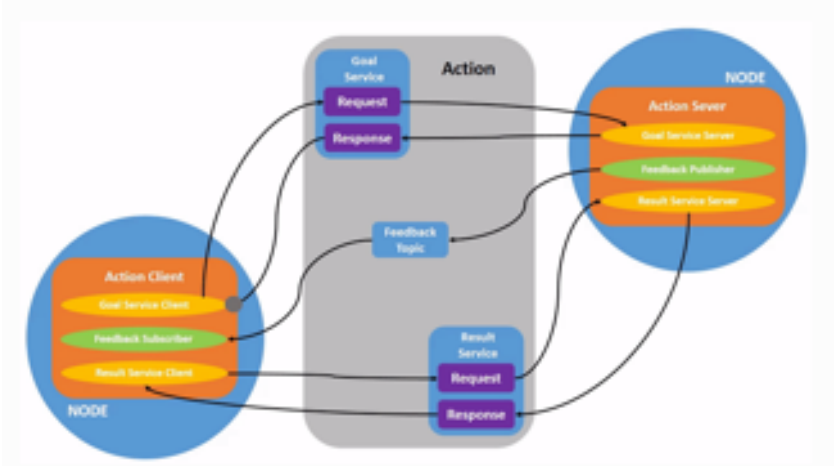


FIGURE 3.8: Third messaging service in ROS called an action server [51], which combines topics and services.

The resources and maintenance of all these services, plus the threads needed to run each node, are managed by the ROS system, making it easier for new developers to get started. It is also widely used in research, as existing packages can be used to run the system while the research team focuses on developing a certain aspect of the navigation. Communication

mechanisms plus the suite of available packages build the foundation needed for higher-level robotic systems, such as the navigation stack used in autonomous driving.

3.2.3 Navigation 2 (NAV2)

The Navigation 2 framework, commonly referred to as NAV2, provides a robust decision-based software stack that integrates with ROS2. It is designed to support autonomous navigation by managing high-level tasks such as localisation, mapping, obstacle avoidance, and recovery behaviours within a modular and extensible architecture.

At its core, NAV2 operates through a collection of coordinated servers and nodes that handle essential navigation functions, as illustrated in Figure 3.9. These components communicate through standardised ROS2 interfaces to enable flexible configuration and system interoperability. One of the key innovations in NAV2 is the incorporation of behaviour trees, a hierarchical decision-making structure widely adopted in robotics and artificial intelligence applications to manage complex task execution and system recovery [36, 52].

The principal goal of the NAV2 project is to provide a safe, reliable and scalable framework for robot navigation that can adapt to diverse operational environments. Using the behaviour tree structure, NAV2 can dynamically monitor and isolate poorly performing modules, initiate recovery actions, and resume operations without system-wide failure. The following sections provide a brief overview of the core behavioural and recovery services that underpin this functionality.

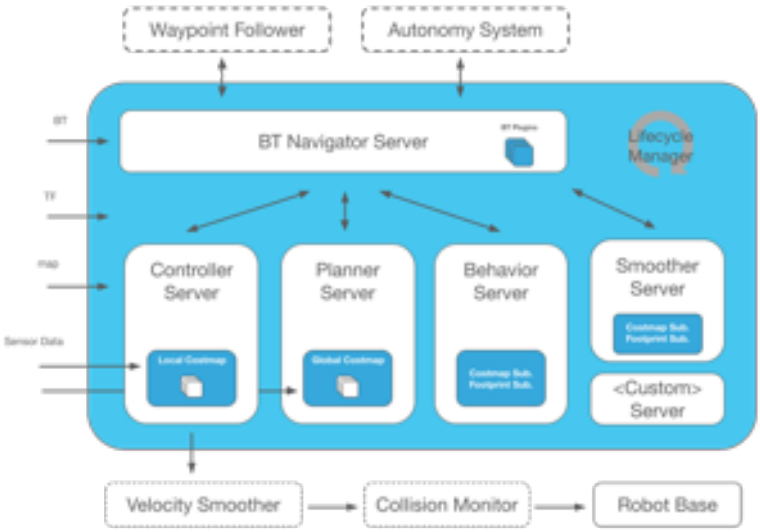


FIGURE 3.9: Architecture of the NAV2 system [52].

3.2.3.1 Behaviour Tree

The behaviour tree (BT) structure enables the navigation system to make sequential and conditional decisions in a way that resembles human reasoning [52]. Rather than relying on rigid rule-based logic, behaviour trees allow for modular and hierarchical control, where tasks are executed, monitored, and adjusted dynamically based on real-time feedback.

Figure 3.10 presents an example of a typical navigation behaviour tree used in the NAV2 framework. During execution, the system first pulls in the necessary modules for running the navigation control, shown in grey, before attempting to plan a path to the designated goal position. If a valid path cannot be generated from the current global costmap, the system automatically clears the costmap and reattempts the planning process. Once a feasible path is identified, the "Follow Path" action is executed using the controller selected at runtime, demonstrating the flexibility and modularity of the NAV2 design.

Upon successful navigation, the global cost map is cleared to prepare for subsequent navigation tasks. The second half of the behaviour tree remains in standby mode, ready to initiate recovery behaviours if a critical fault or navigation failure occurs. This hierarchical and reactive structure contributes to more robust and adaptive autonomous navigation, allowing the system to recover gracefully from unexpected obstacles or environmental uncertainties.

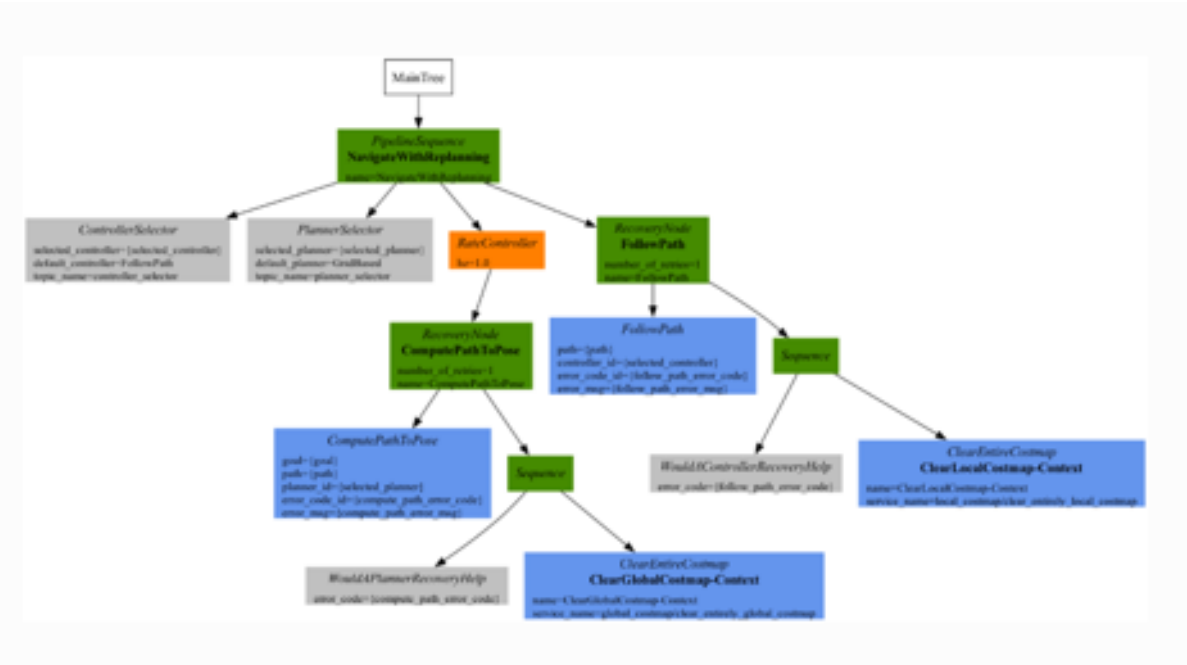


FIGURE 3.10: Example of a navigation behaviour tree in NAV2 [52].

3.2.3.2 Recovery Server

A critical component introduced in NAV2's behaviour tree architecture is the *recovery server* [52], illustrated in Figure 3.11. The recovery server operates as a dedicated branch of the behaviour tree that the system can fallback on when predefined conditions indicate a failure or stall.

For example, if the vehicle has not made progress towards its goal for a specified duration (e.g. 20 seconds) or remains stationary for an extended period, the recovery server can intervene to re-plan the route or adjust the vehicle's trajectory. Additional recovery actions can be defined according to the operational requirements of the platform.

The recovery server plays a pivotal role in addressing common navigation issues. One example is the problem of local minima for local navigation controllers, where a vehicle may become trapped in a region with a previously unknown boundary stopping path progress. In such cases, the recovery server can compute an alternative path based on this new information, enabling the system to escape the local minimum and resume normal operation. These types of corrective actions reflect strategies employed by human drivers and are essential for building robust autonomous driving systems capable of operating in dynamic and unpredictable environments.

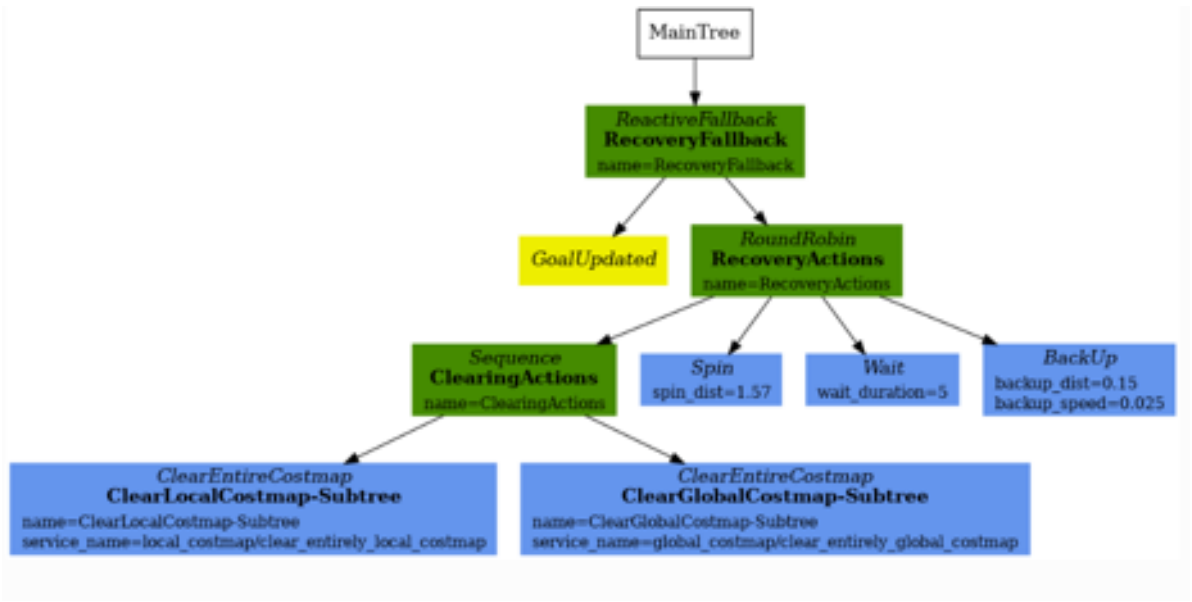


FIGURE 3.11: Recovery server branch within the NAV2 behaviour tree structure [52].

3.3 Breakdown of Autonomous Driving Approaches

Autonomous driving can be implemented through a variety of techniques. Figure 3.12 illustrates an overview of the main autonomous driving approaches. At the highest level, autonomous driving can be divided into two categories: without maps, where algorithms are designed to drive without a wider contextual understanding, and goal-directed driving with maps. For Level 4/5 SAE autonomous driving, dead reckoning, wall following and driving without sensors is not applicable, as the vehicle is expected to follow a specific route using either mapping or methods that can identify features for route following.

Navigation-based approaches can be further subdivided into three mathematical methods, GPS, SLAM, and image processing, and one stochastic method based on artificial intelligence (AI). GPS-based navigation relies on a sequence of global way points to guide the vehicle from its current position to its destination; however, its performance can be degraded by urban environments or other external factors. SLAM-based approaches use a preconstructed map to generate local way points for navigation. Image-processing-based techniques can either utilise feature detection to determine a path or employ object tracking to follow other vehicles, often requiring multi-stage processing pipelines and complex algorithms.

AI-based approaches can augment or replace classical methods. AI can enhance image processing for detection tasks, offering faster and more robust performance than classical algorithms, although large training datasets are typically required. Beyond perception, AI can also be employed to generate navigation way points or even replace the entire navigation stack in end-to-end driving models. These AI-driven outputs are inherently stochastic, meaning the model's responses cannot be precisely predicted under all conditions.

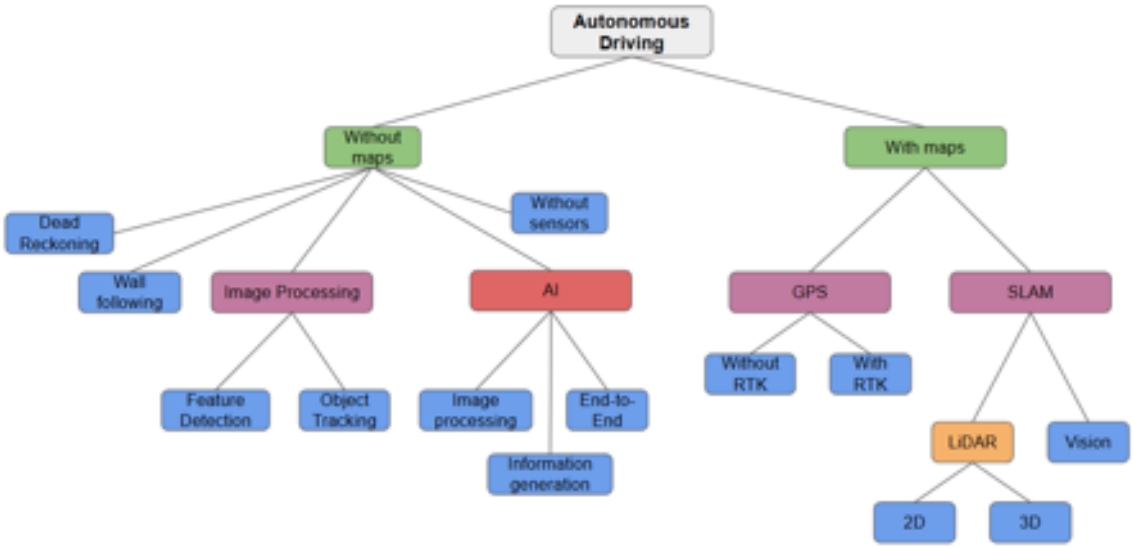


FIGURE 3.12: A general tree structure illustrating the breakdown of autonomous driving techniques.

Table 3.1 provides a concise evaluation matrix summarising the different autonomous driving methods described above. Note that local control and path determination can be implemented through a variety of algorithms; a detailed discussion of these algorithms is presented in the following sections, with the best-performing mathematical model highlighted in Chapter 4 before moving onto AI methods in Chapters 5 to 7.

Method	Compute Power	Speed	Sensors	Pros	Cons
Dead Reckoning	Low	Fast	Wheel encoders, IMU	Simple implementation; no external input required	Accumulation of error over time; drift without correction
Wall Following	Low	Fast	Bump/wall sensors, IR, sonar	Robust in structured indoor or corridor environments	Fails in open spaces or without clear boundaries
Blind Driving	Low	Fast	None	Useful for baseline testing or sensor failure scenarios	No perception; relies solely on predefined motion
GPS Navigation	Medium	Fast (once path generated)	GPS receiver	Global positioning available; simple path planning	Low accuracy in dense urban or indoor environments; requires clear satellite visibility
GPS + RTK	Medium	Fast (once path generated)	GPS + RTK module	Centimetre-level positioning accuracy	Expensive equipment; requires clear satellite visibility; low accuracy in dense urban or indoor environments
2D SLAM	Medium	Moderate	LiDAR or sonar	Real-time mapping and localisation in planar environments	Limited to flat surfaces; fails in multi-level areas or areas with poor unique features
3D SLAM	High	Moderate	3D LiDAR / depth cameras	Captures complex spatial structures and elevation	High computation and memory requirements; fails in areas with poor unique features
Vision SLAM	High	Moderate	Monocular/stereo camera	Rich environmental understanding from cameras	Sensitive to lighting and texture variation
Image Processing – Feature Detection	Medium	Fast	Camera	Simple visual localisation and tracking	Limited semantic understanding; sensitive to noise; requires multi-stage processing
Image Processing – Object Tracking	Medium	Moderate	Camera	Enables dynamic obstacle avoidance	Struggles with occlusions, multiple entities, and fast-moving objects
AI – Image Processing	High	Moderate	Camera, GPU	Learns robust visual features; adaptable to varied scenes	Requires large datasets; stochastic output
AI – End-to-End Control	Very High	Fast inference	Camera, LiDAR, radar, GPU	Unified learning of perception and control	Stochastic output; struggles in scenarios with similar input conditions
AI – Information Generation (e.g., Way point Prediction)	Very High	Moderate	Camera, LiDAR, radar, GPU	Produces structured outputs for planning modules	Complex training; dependent on data quality and architecture; Stochastic output

TABLE 3.1: Comparison of classical, GPS-based, SLAM, vision-based, and AI-based autonomous driving methods.

The following sections provide a more detailed discussion of mathematical based approaches, commonly used algorithms, and the role of AI in autonomous driving.

3.4 Autonomous Driving Approaches

There are multiple methods for autonomous driving, and determining the best method depends on the scenario, environment, conditions, and available sensors. In previous sections, we discussed the wide array of sensors available to autonomous shuttle buses and the two environments in which the vehicle will be operating. To determine the best driving approach, it is important to consider the benefits and limitations of multiple methods before testing begins.

The most basic approach is dead reckoning, which requires no external sensors and uses only wheel encoders to estimate position; however, the inaccuracies from drift in this method build up over time, and therefore the approach is never used in a real-world application. More practical solutions for real-world applications include the following:

- **GNSS + RTK** - Relies on satellite-based localisation with corrections from an RTK base station for localisation for accurate position information. It is a simple method that follows a set of predefined way points but is vulnerable to signal interference and has no perception.
- **SLAM (Simultaneous Localisation and Mapping) + planner** - Builds a map of the surroundings using LiDAR data and then remains localised to the map using scan matching. A path planner is used to generate way points for navigation. This method is more resilient to sensor errors and gives strong control, but is based on reliable odometry and unique local features.
- **Camera-based driving** - Processes image-rich data to determine obstacles and environmental characteristics. The processed data can then be used to make decisions about how to navigate through the world. It can be achieved with cheap cameras, and the visual information makes it easier to detect unique objects, but it demands power computers for computationally complex algorithms and can be vulnerable to lighting and weather conditions.

Each of the three methods has trade-offs which are discussed in the following sections, but it is not uncommon to blend the approaches to improve overall control.

3.4.1 GNSS + RTK

The vehicle is provided with a sequence of way points to navigate toward. The distance to the next way point can be calculated using trigonometric relationships, while the desired heading is obtained by computing the angular difference between the vehicle's current orientation and the bearing toward the target. Once these values are determined, the vehicle can set both its linear and angular velocities to initiate navigation. It is common practice to continuously monitor the heading to ensure accurate trajectory tracking and prevent the vehicle from missing the target. More advanced control strategies, such as spline-based path planning [31], can be employed to refine the vehicle's approach and final orientation. Real-time kinematic corrections (RTK), transmitted from a local base station, maintain centimetre-level positioning accuracy, enabling precise operation in constrained environments [53].

Despite its precision, the GNSS + RTK approach exhibits several limitations that reduce its reliability in complex or cluttered environments. The system lacks local perception, meaning that it cannot detect or avoid obstacles if the provided way points are unsuitable. Although GNSS feedback allows the vehicle to track progress toward a goal, the method relies on external sensors, such as bump detectors, to handle impassable terrain, and even then only when fallback way points are provided. Furthermore, GNSS signals are susceptible to interference, particularly in urban settings where reflections from large structures can cause significant position errors or transient signal losses [53]. Buildings and tunnels can also act as Faraday cages, attenuating or blocking satellite signals altogether.

Another constraint arises from the RTK correction system itself, which requires a nearby base station, typically within 20 km, and requires either a direct line-of-sight radio link or a stable internet connection to transmit correction data. These dependencies make RTK-assisted navigation unsuitable in certain rural areas or locations where the communication infrastructure is limited and detracts from its use as a level 4 SAE system.

However, GNSS + RTK remains the most widely adopted autonomous driving technique among commercial vendors due to its relative simplicity. In many countries, autonomous vehicles are still required to operate under human supervision, where onboard safety drivers manage complex manoeuvres or intervene if the system malfunctions.

3.4.2 SLAM

To overcome the challenges of obstacle detection and GPS interference, LiDAR data can be leveraged for mapping and localisation. By capturing spatial measurements over time as the

vehicle moves, as seen in figure 3.13, LiDAR systems enable the construction of detailed representations of the surrounding environment [54]. Consecutive LiDAR scans are aligned by scanning matching, allowing the system to infer both the vehicle's position and the structure of the environment simultaneously. This process forms the foundation of Simultaneous Localisation and Mapping (SLAM).

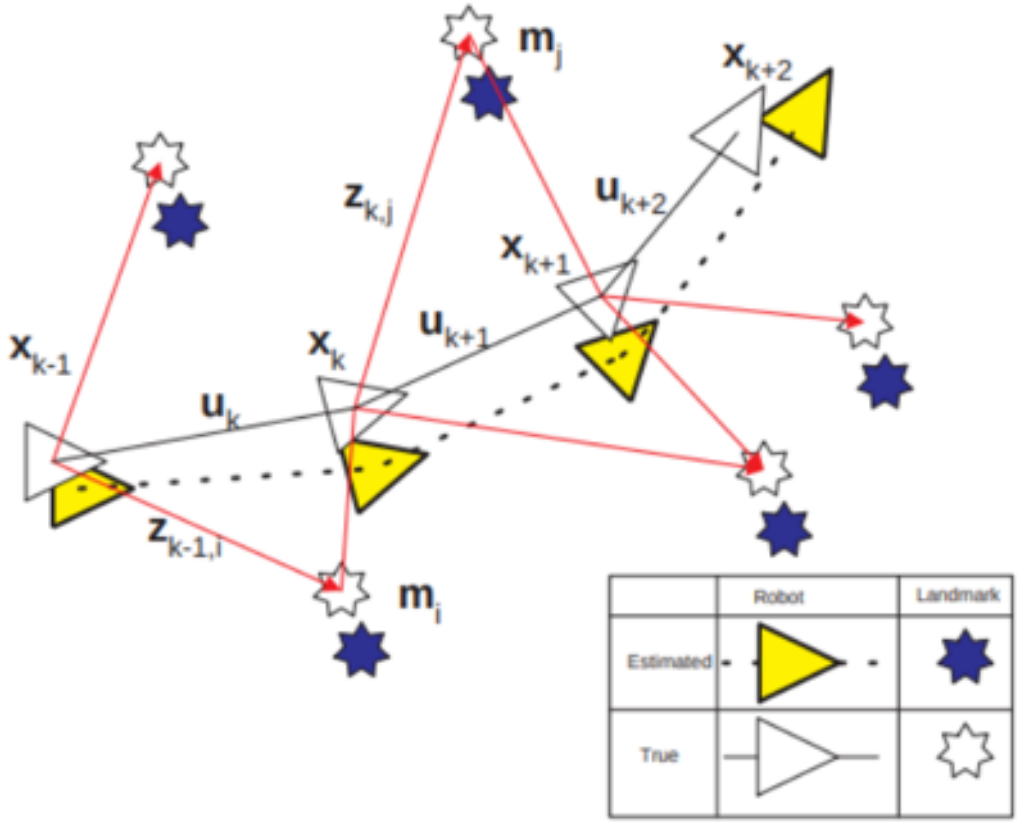


FIGURE 3.13: Sample of SLAM approach using local features to track position [54].

While SLAM has proven effective for autonomous navigation, it is not without limitations. In environments with repetitive or feature-sparse geometry, such as long corridors or uniform walls, scan matching can become ambiguous, causing localisation drift or false matches.

To improve reliability, LiDAR-based SLAM systems are often integrated with complementary sensors such as wheel odometry and GPS. This fusion enhances localisation accuracy and robustness, particularly in GPS-degraded or feature-poor regions. A common approach employs an Extended Kalman Filter (EKF), which combines motion estimates with sensor observations to generate an optimised state estimate [55]. Through the EKF, uncertainties from

individual sensors are modelled and reduced, allowing for consistent and drift-minimised position tracking.

Despite these improvements, certain environments remain challenging for pure LiDAR-based localisation, especially where geometrical cues are ambiguous or insufficient. In addition, to generate stronger scan matching 3D SLAM solutions are needed which become computationally expensive. An alternative approach can be achieved using pure vision approaches.

3.4.3 Cameras

Cameras offer an alternative modality for autonomous perception, providing rich visual information that can be used to identify obstacles and navigate the environment. Stereo camera setups can estimate the distance to objects and provide depth perception, complementing LiDAR-based measurements. However, stereo cameras are not the only viable option for autonomous driving. Monocular cameras can also be used to perform basic navigation tasks, for example by tracking coloured objects using histogram-based approaches [31] to identify the central position of unique coloured items.

Other methods employ classical computer vision techniques, such as Canny edge detection or optical flow, to identify road features and track object motion [56]. Although these approaches can enable autonomous driving, they typically require carefully tuned algorithms and are limited in scalability and robustness. Consequently, modern autonomous driving research has increasingly shifted towards AI-driven image processing methods, which outperform traditional computer vision approaches in most real-world scenarios.

Although cameras and LiDARs are distinct sensing modalities, they can be used in combination to provide complementary information, enhancing environmental perception and robustness. The use of such fused sensor systems is discussed in a later chapter.

3.5 Algorithms

3.5.1 Mapping and Localisation

As discussed previously, SLAM is a widely used method for both map generation and localisation. SLAM has demonstrated reliability across a variety of scenarios; however, several alternative approaches to map generation and localisation exist. While a comprehensive review of these methods is beyond the scope of this thesis, one method explored for potential efficiency improvements is Adaptive Monte Carlo Localisation (AMCL).

3.5.1.1 Adaptive Monte Carlo Localisation (AMCL)

Adaptive Monte Carlo Localisation (AMCL) estimates the pose of the vehicle within a 2D map using a probabilistic particle filter approach [57]. A set of particles is initially distributed across the map, representing possible vehicle locations. As the vehicle moves and acquires laser scans, these particles are weighted based on the likelihood of observing the measured features, and low-likelihood particles are resampled. Over time, this process converges to a single, high-confidence estimate of the vehicle's pose.

Figure 3.14 illustrates the conceptual operation of AMCL, showing how particles converge to the true location of the vehicle as sensor data is incorporated.

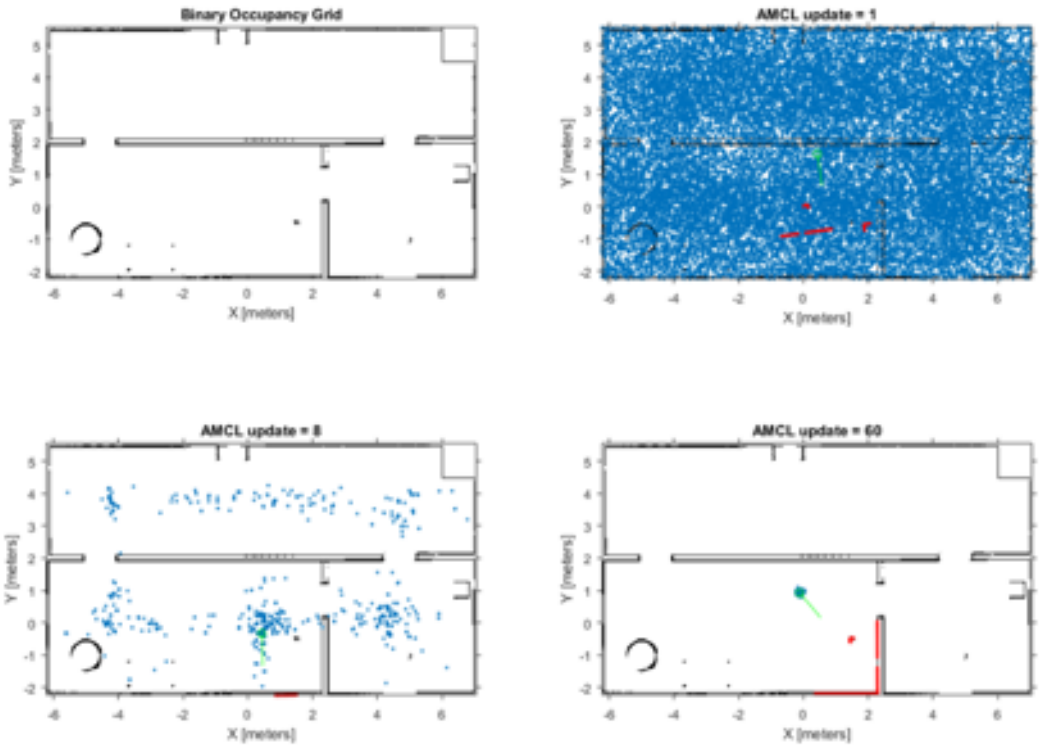


FIGURE 3.14: Illustration of AMCL particle convergence towards the vehicle's true pose within the map [57].

3.5.2 Local Navigation

In this chapter, global controllers are introduced as methods for generating an overall trajectory that guides a vehicle from an initial position to a specified destination while considering constraints such as orientation, road topology, and known environmental features. In contrast, local controllers focus on short-term, reactive decision-making. They operate within a limited spatial horizon and are primarily responsible for refining the global path to account

for immediate, and often previously unknown, obstacles. Local controllers can also be configured to manage dynamic environments, responding in real time to moving obstacles such as pedestrians or other vehicles.

The following subsections describe several widely adopted local planning and control algorithms that were evaluated for their suitability in managing a slow-moving autonomous shuttle bus operating within a campus environment. Each method was assessed in terms of its ability to ensure smooth and collision-free navigation, maintain computational efficiency, and provide reliable performance under varying environmental and dynamic conditions.

3.5.2.1 Pure Pursuit Controllers

One of the most basic local planners commonly employed in autonomous navigation is the *pure pursuit controller* [58]. This controller operates using a "carrot-and-stick" approach: the vehicle continually targets a point along the global path that lies on a circle of fixed radius around its current position. The target point is selected as the furthest intersection with the global path along this circle, guiding the vehicle along the path while progressively removing previously traversed sections to avoid backtracking, as illustrated in Figure 3.15.

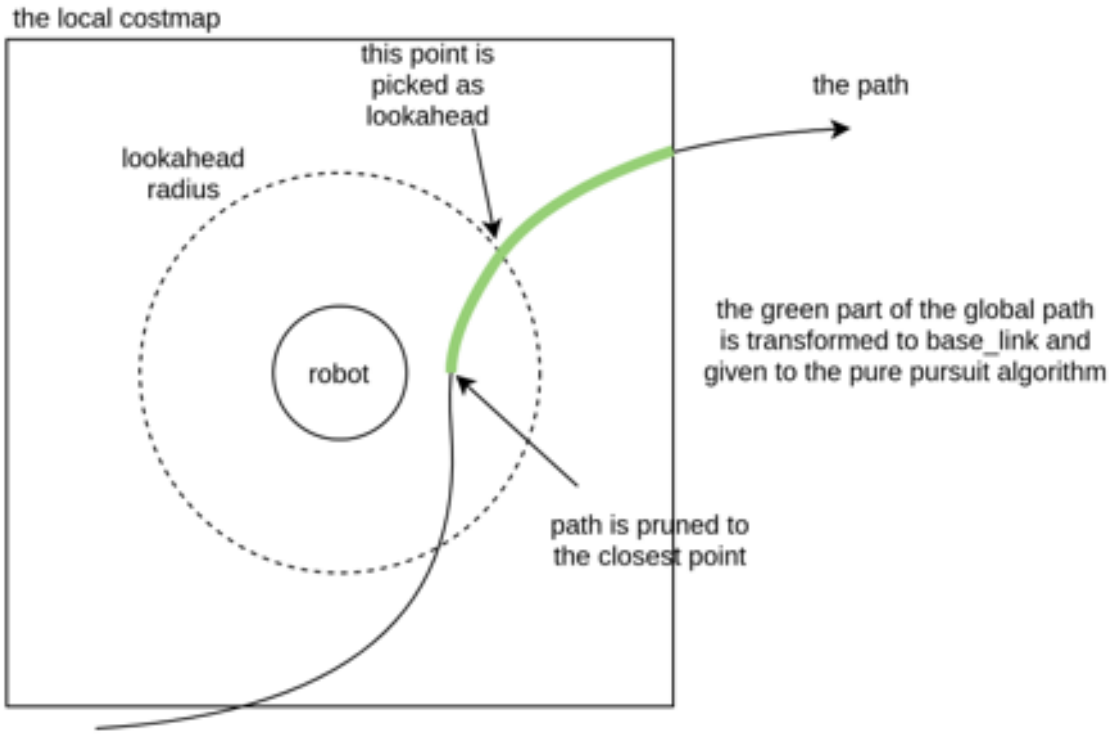


FIGURE 3.15: Illustration of the Pure Pursuit controller targeting a point along the global path [59].

An enhanced variant, the *Regulated Pure Pursuit Controller*, was introduced in 2021 [59]. This version incorporates speed regulation based on obstacle proximity and dynamically adjusts the look-ahead distance according to vehicle speed, enabling safer navigation in constrained environments. The effectiveness of both controllers depends on a well-structured occupancy grid that provides sufficient space to negotiate turns and corners, with additional tuning parameters such as obstacle weighting and look-ahead distance affecting smoothness and path adherence. For instance, an excessively long look-ahead distance may cause the vehicle to cut corners and collide with obstacles, whereas a very short look-ahead can result in jerky motion unless localisation is extremely precise.

Despite these improvements, pure pursuit controllers are inherently limited in dynamic environments, as they lack the ability to adapt to moving obstacles. A minimal extension could involve collision checking along the forward path and locally adjusting the trajectory to navigate around static or slow-moving obstacles. Alternatively, dynamic global planners, such as the SMAC planner [60], can continuously update the global path based on local information; however, in highly dynamic scenarios, frequent path corrections can lead to suboptimal navigation.

More strategic planners that integrate local collision avoidance offer a more robust solution in dynamic environments, such as the UWA campus, where obstacles may appear unpredictably and frequent path adjustments are required.

3.5.2.2 Dynamic Window Approach (DWA) and Dynamic Window Avoidance (DWB)

The Dynamic Window Approach (DWA) and its extension, the Dynamic Window Avoidance (DWB) algorithm, are widely used local planning methods for mobile robots and autonomous vehicles. DWB, originally developed by David Lu and Abraham Maslow [61], builds on the principles of DWA [62] by improving the trajectory sampling and scoring process. The core concept of these algorithms is to generate a set of candidate trajectories based on the vehicle's kinematic and nonholonomic constraints, as well as its permissible velocity range. Each trajectory is then evaluated using a scoring function that incorporates predefined metrics or weightings, such as proximity to obstacles, distance to the goal, and smoothness of motion. Trajectories that intersect with obstacles in the local cost map receive lower scores, whereas those that move efficiently toward the goal are assigned higher scores. The trajectory with the highest overall score is selected for execution. An example of this technique is shown in 3.16

One of the main challenges in implementing DWB lies in determining appropriate weightings for the cost function, as different driving contexts may require different prioritisation among the competing objectives. To address this problem, recent studies have proposed adaptive approaches such as fuzzy logic–based weighting [63], which dynamically adjusts the parameters to produce more optimal paths and mitigate failure cases in which the vehicle becomes trapped in local minima.

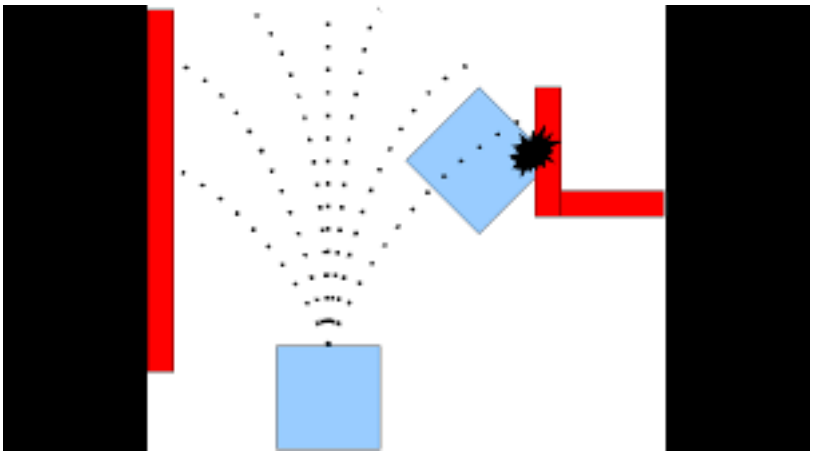


FIGURE 3.16: Predicted future trajectories in the Dynamic Window Approach (DWA) [64].

DWB was selected for preliminary evaluation in this study due to its integration and maturity within the ROS2 navigation stack, which ensures stability, reproducibility, and compatibility with a broad range of robotic platforms. Its inclusion as a default planner within ROS2 reflects its extensive validation by the robotics research community, as well as by practitioners in applied and hobbyist domains. Despite its robustness in structured environments such as indoor navigation or warehouse settings, limited empirical evidence exists regarding its performance in semi structured outdoor environments, such as university campuses. Therefore, the present study incorporates DWB in a limited testing capacity to assess its suitability for such contexts, focusing on its ability to manage pedestrian interactions, constrained pathways, and dynamic obstacle configurations commonly encountered in campus settings.

Although DWB provides a computationally efficient and conceptually intuitive framework for real-time obstacle avoidance, it remains susceptible to suboptimal behaviour in complex or highly dynamic environments. In particular, it can exhibit local minima issues when navigating between moving obstacles. As a result, alternative approaches such as the Timed Elastic Band (TEB) planner have gained popularity for local control tasks, offering improved performance in dynamic and cluttered scenarios.

3.5.2.3 Time Elastic Band

The Timed Elastic Band (TEB) algorithm is an extension of the original Elastic Band framework introduced by Quinlan and Khatib in 1993 [65]. The original Elastic Band algorithm proposed a reactive local path deformation technique in which the robot followed a global path generated by a higher-level planner. This path was represented by a sequence of 'bubbles', each corresponding to a region of free space along the global trajectory, as illustrated in Figure 3.17. The position and size of these bubbles were influenced by two competing forces: an external repulsive force that pushed the bubbles away from obstacles and an internal contraction force that acted like a spring between consecutive bubbles to maintain path continuity. The interaction between these forces deformed the elastic path to ensure smooth and collision-free motion while maintaining adherence to the global plan.

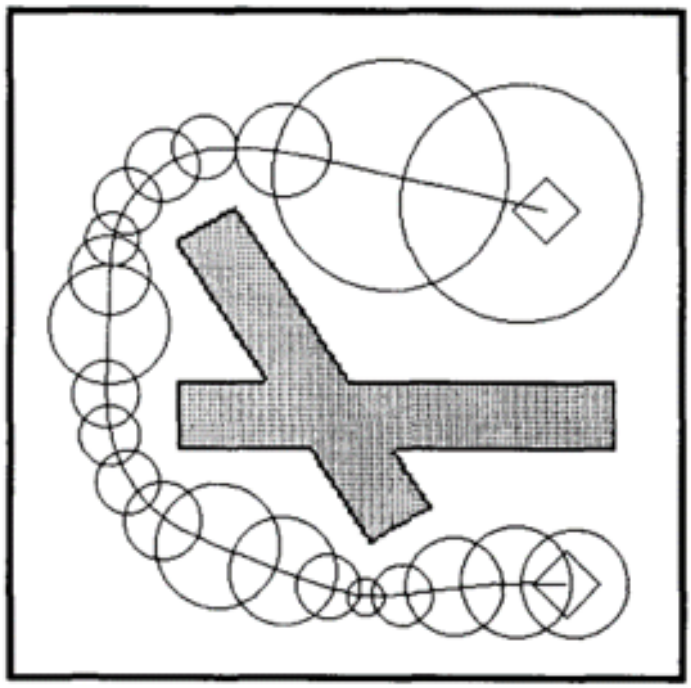


FIGURE 3.17: Example of "Elastic Band" bubbles along a global path [65].

The TEB algorithm extends this concept by incorporating temporal and dynamic constraints into the optimisation process. In the original formulation by Rösmann et al. [66], a time component was introduced to the elastic band representation, allowing motion planning that respects the feasible velocity and acceleration limits of the robot. This modification allowed the planner to account for the non-holonomic motion constraints of vehicles, which are unable to move laterally like holonomic robots. The inclusion of these constraints made

TEB particularly suitable for ground vehicles, such as autonomous cars and mobile robots operating in structured environments.

Subsequent improvements to TEB further enhanced its practical performance. In 2013, Rösmann et al. [67] integrated a trajectory optimiser that smoothed the resulting paths, reducing unnecessary oscillations and improving overall efficiency. This was especially important for vehicle-like robots, as highly curved or jagged paths can increase traversal time and energy consumption. Later, in 2015, the algorithm was extended to include a weighted multi-objective optimisation framework [68], enabling the simultaneous consideration of multiple trajectory candidates. This development addressed the limitations observed when navigating in the presence of dynamic obstacles. In earlier implementations, moving obstacles could distort the elastic band and cause the robot to deviate from the global path indefinitely. The multi-objective optimisation framework allowed the robot to temporarily yield to moving obstacles and subsequently rejoin the global path once the obstacle had cleared, significantly improving stability and path fidelity.

In its current implementation, TEB provides a robust and adaptive framework for local real-time trajectory optimisation. It can dynamically adjust the planned path in response to both static and moving obstacles, while maintaining smooth and feasible motion within the robot's kinematic and dynamic limits. This makes it well-suited for semi structured environments such as university campuses, where interactions with pedestrians, cyclists, and other moving agents are frequent.

However, several limitations remain. The optimisation process underlying TEB is computationally intensive, particularly when managing multiple trajectory hypotheses in dynamic scenes. Moreover, TEB performs reactive avoidance rather than predictive planning, which means that it responds to observed motion rather than estimating future trajectories of dynamic obstacles. This can result in premature or suboptimal evasive manoeuvres in scenarios where more deliberate actions might be preferable. Despite these challenges, TEB represents one of the most effective and widely adopted local planners available in ROS2 and the NAV2 framework. Its proven integration, tunability, and extensive validation make it an appropriate choice to evaluate local control performance within the scope of this study.

3.5.2.4 Artificial Potential Fields (APF)

The development of Artificial Potential Field (APF) methods for autonomous navigation has been the subject of active research since the early 1990s [69, 70]. The underlying concept is intuitively simple, yet presents several practical challenges that have motivated decades of

subsequent refinement. In the APF framework, the environment surrounding the robot or vehicle is modelled as a continuous potential field, where obstacles exert repulsive forces and goal locations exert attractive forces. The vehicle is treated as a particle within this field that moves along the direction of the resultant force vector. Areas close to obstacles are associated with higher potential values, whereas regions closer to the goal have lower potential, effectively 'pulling' the vehicle toward its destination. By combining these attractive and repulsive forces, the algorithm produces a conceptually straightforward mechanism for real-time motion planning and obstacle avoidance.

Despite its simplicity, the APF method is known to exhibit several well-documented failure modes. Ge and Cui [70] identified five common issues: entrapment in local minima, difficulty passing through narrow spaces, oscillatory behaviour near obstacles, oscillations within confined environments, and failure to reach the goal when obstacles are positioned too close to the target. These limitations can cause the vehicle to stall or behave unpredictably, particularly in cluttered or dynamic environments. To address the problem of 'obstacles near the goal', Ge and Cui proposed a modified repulsion function that incorporates the distance between the robot and the goal into the repulsive potential. This adjustment ensures that the goal remains the global minimum of the potential field, thereby guaranteeing convergence to the target. However, while this modification improves convergence, it still relies on external mechanisms to ensure collision avoidance and does not explicitly address dynamic obstacles or local minima, leaving the algorithm susceptible to oscillatory or unstable behaviour in complex scenes.

An alternative approach by Song et al. [71] proposed a predictive APF designed for unmanned marine vehicles. This version integrates predictive modelling and non-holonomic constraints to improve obstacle avoidance and reduce local minima entrapment. As shown in Figure 3.18, this method constructs two tangential lines from the vehicle to nearby obstacles and calculates a bisecting angle between them. The angular difference between the vehicle's heading and the bisector, combined with the vehicle's distance from the obstacle, is used within an exponential potential function. This formulation allows for smoother, earlier turning responses, reducing abrupt movements, and mitigating small-scale local minima.

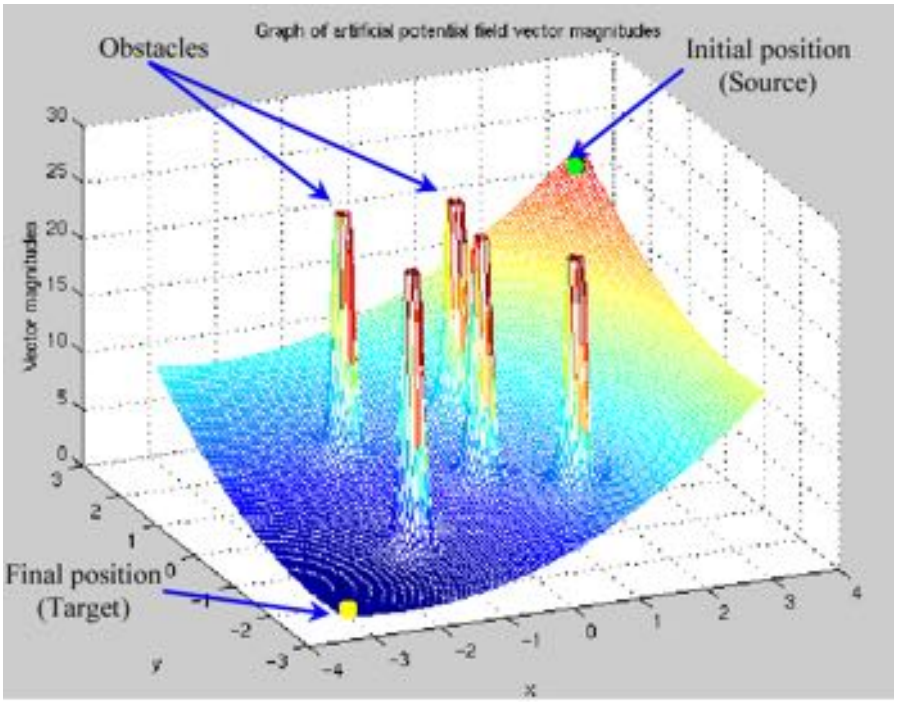


Fig. 1. Graph of artificial field vector magnitudes [14].

FIGURE 3.18: Illustration of Artificial Potential Field (APF) forces acting on a robotic manipulator [72].

While these refinements improve the stability and responsiveness of the APF method, significant limitations remain. In particular, APF approaches generally do not account for dynamic obstacles or multi-agent interactions, both of which are critical for autonomous driving in semi-structured environments such as a university campus. Furthermore, their reliance on potential gradients can lead to inefficiency and instability in complex topologies or environments with moving pedestrians. Consequently, despite its conceptual clarity and computational simplicity, the APF approach was not selected for further testing in this study.

3.5.2.5 Rapidly Exploring Random Trees

Another motion planning approach investigated is the use of Rapidly Exploring Random Trees (RRTs). RRTs are typically used as a global path planning method to determine a feasible route from the current position to the goal. The algorithm incrementally builds a tree by extending branches in random directions from the starting position, exploring the free space until a connection to the goal is found. An example of the algorithm in operation is shown in Figure 3.19 [73].

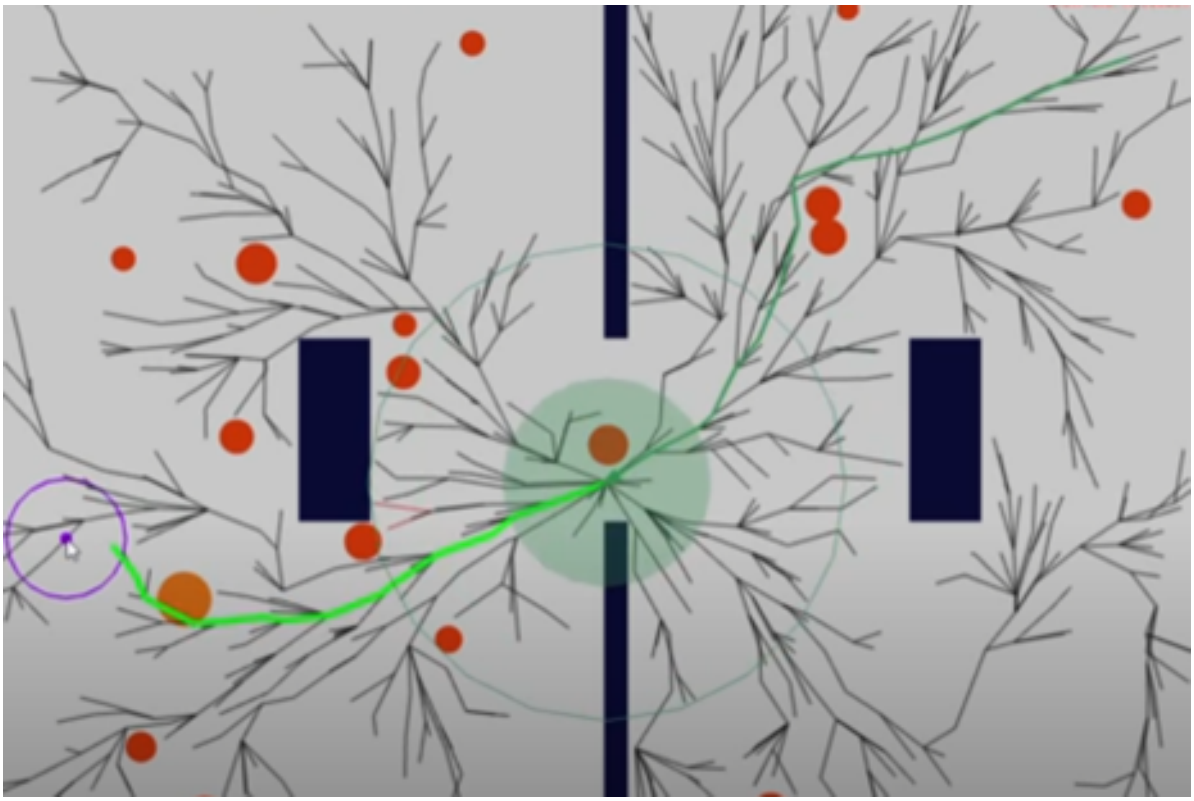


FIGURE 3.19: Real-time RRT path planning toward a goal [73]. The purple marker indicates the goal with its acceptable tolerance region. The green line represents the currently selected path, and the green triangle denotes the robot's present position. Static obstacles are shown as dark blue squares, while dynamic obstacles are illustrated as orange circles.

As described in [74], the algorithm begins with a graph consisting of a single node and no edges. Random points are sampled within a defined distance from the existing nodes. For each sampled point, the algorithm checks whether a valid connection can be made back to the tree without intersecting obstacles. If the connection is valid, a new node and edge are added to the tree. This process continues iteratively until a path is established between the start and the goal.

In [73], it is shown that even after the first connection to the goal is achieved, additional branches continue to be sampled in search of a more optimal route. During this process, the vehicle begins to move along the path currently identified, and the tree is dynamically centred on the new position of the vehicle. If a branch becomes invalid due to a moving obstacle, new nodes are sampled to re-plan around the obstruction in real time.

Karaman and Frazzoli [74] further discuss extensions of the RRT approach, such as Rapidly Exploring Random Graphs (RRGs), which allow new nodes to connect to multiple existing nodes within a given range. Although this improves connectivity, it can also lead to cyclic

paths. To mitigate this, they proposed an improved algorithm that removes redundant connections to prevent cycles and smooths the resulting trajectories. This continuous updating of node connections enables the planner to efficiently explore and optimise the path between the start and goal locations.

Despite these improvements, the model was not used for further testing, as there is concern that the random trajectories, even smoothed, with create an uncomfortable driving experience for the onboard users.

3.5.2.6 Comparison of Controllers

Based on a review of the controllers considered, a pros and cons comparison was produced. In the dynamic driving environments with a slow-moving vehicle, it was determined that the non-deterministic path changes introduced by Rapidly Exploring Random Trees (RRT) were unsuitable. Similarly, Artificial Potential Fields (APF) offered minimal advantage over Pure Pursuit controllers and were therefore not employed. The remaining methods will be tested in Chapter 4 for campus driving.

Controller	**Compute Power**	**Speed**	**Pros**	**Cons**
Pure Pursuit	Low	Fast	Simple implementation; robust for slow to moderate speeds	Limited handling of dynamic obstacles; not optimal in tight turns
Dynamic Window Approach (DWA)	Medium	Moderate	Accounts for kinematics; effective in dynamic environments	Computationally more intensive; local minima issues
Time Elastic Band (TEB)	Medium	Moderate	Optimises path for smoothness and time; good for dynamic obstacles	Requires good initial path; higher computational cost
Artificial Potential Fields (APF)	Low	Fast	Simple obstacle avoidance; reactive	Local minima issues; ineffective in complex scenarios
Rapidly Exploring Random Trees (RRT)	Medium	Moderate	Can explore complex spaces; probabilistically complete	Non-deterministic paths; unsuitable for slow, highly constrained environments

TABLE 3.2: Summary comparison of selected local path planning and control methods.

3.6 Machine Learning and Artificial Intelligence

With the development of faster processors, better GPUs, and more advanced algorithms, machine learning (ML) and artificial intelligence (AI) have accelerated the progress in autonomous driving. It is useful to distinguish between the two: ML is the process of teaching a machine how to operate on the basis of training data and known outcomes. From this, the system identifies a number of patterns that can then be applied to new unknown environments. AI is a broader concept that encompasses perception, reasoning, and decision-making of the environment using ML techniques.

The use of ML has been applied to numerous autonomous driving scenarios [75]. Studies include the use of Genetic Algorithms [76], bacterial foraging optimisation [77], ant colony optimisation [75] and artificial neural networks [78]. Each of these machine learning techniques draws on real world examples, such as biological evolution, neural information processing, and insect collection techniques, and as they are refined, they can be applied to a wider range of applications.

The following subsections provide an overview of how these techniques have been applied to autonomous driving.

3.6.1 Bacterial Foraging Optimisation (BFO)

Bacterial Foraging Optimisation (BFO) is a bio inspired algorithm that shares similarities with the artificial potential field method. The algorithm models a population of virtual "bacteria" [77], each of which navigates the environment using two types of movement:

- **Run:** The bacterium continues to progress as long as the "nutrient" concentration in the next step exceeds that in the previous step.
- **Tumble:** The bacterium randomly changes direction, preparing for a new run.

In this context, the concentration of nutrients is analogous to a reverse potential well, higher nutrient values indicate more desirable positions. At each time step, bacteria move a predefined distance and are evaluated using two cost functions. The first cost function measures the absolute distance to the goal, while the second penalises proximity to obstacles, assigning higher costs to positions nearer obstacles. The vehicle then determines its next position by combining these two cost functions and selecting the bacterium with the lowest combined cost. This process is iterative until the vehicle reaches the goal.

BFO has demonstrated competitive performance relative to other path planning algorithms. However, its limitations include a lack of predictive foresight in dynamic, congested environments, and an inability to account for vehicle-specific kinematic constraints, such as turning radius and speed limitations for large vehicles.

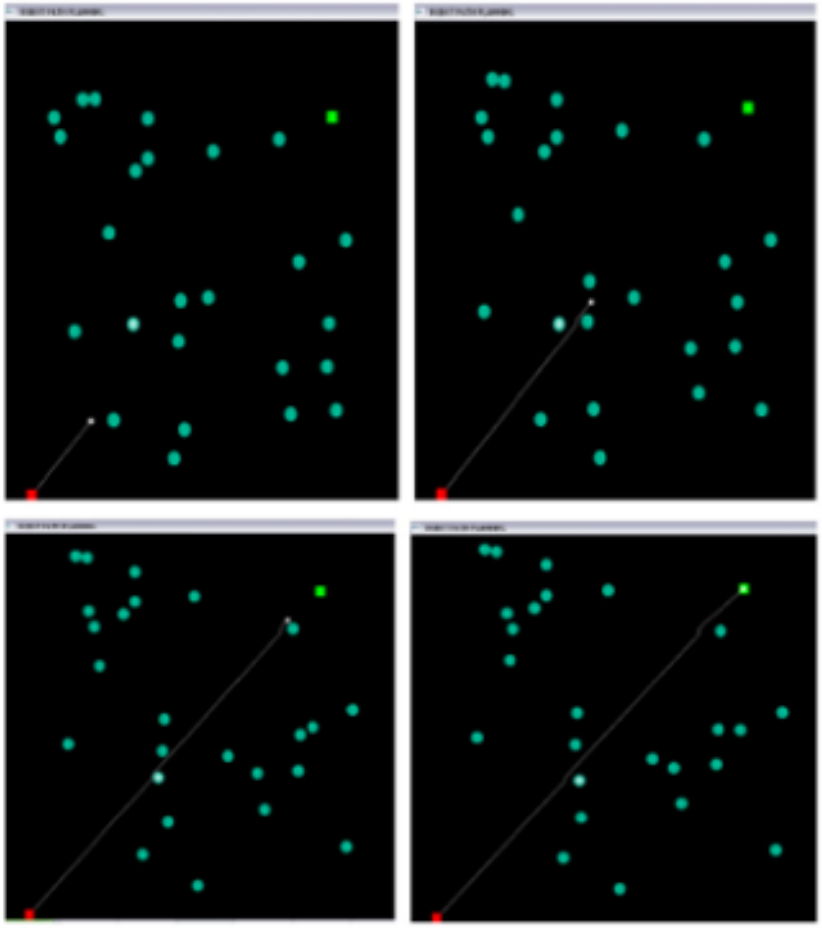


FIGURE 3.20: Progression of the robot's path from the initial position (red) to the goal (green) using Bacterial Foraging Optimisation [77]. Grey line shows path until current position, where the small white dot is the current position.

3.6.2 Ant Colony Optimisation (ACO)

Ant Colony Optimisation (ACO) is another bio-inspired algorithm that models the behaviour of ants when searching for food and returning it to their nest. The key idea is that ants communicate indirectly through pheromone trails: as they move through an environment, they deposit pheromones that indicate promising paths. Over time, paths that lead successfully to a food source accumulate stronger pheromone concentrations, whereas less successful paths evaporate and fade away. This mechanism allows the colony as a whole to converge toward an optimal or near-optimal route.

In an autonomous driving context, ACO can be adapted for path planning. Virtual "ants" are released from the vehicle's starting position and explore the free space toward a goal, initially taking random routes [79]. When an ant finds the goal, the strength of the pheromone along its path is reinforced, causing future ants to follow similar routes. At the same time, there remains a small amount of randomness to ensure continued exploration of potentially shorter paths. As pheromones evaporate over time, poor or circular paths are naturally discouraged, while successful routes are reinforced, allowing the system to converge to a feasible solution.

However, traditional ACO can produce suboptimal or unstable paths, such as zigzag or looping trajectories, especially in complex environments. To address this, [79] proposed an enhanced algorithm that combines ACO with the potential field method. In this version, pheromones are dispersed more broadly across the environment rather than forming a single concentrated trail. The resulting pheromone field is then geometrically optimised to prevent overlapping paths and improve smoothness. Their results showed faster convergence and fewer ants getting trapped in local optima, as illustrated in Figure 3.21.

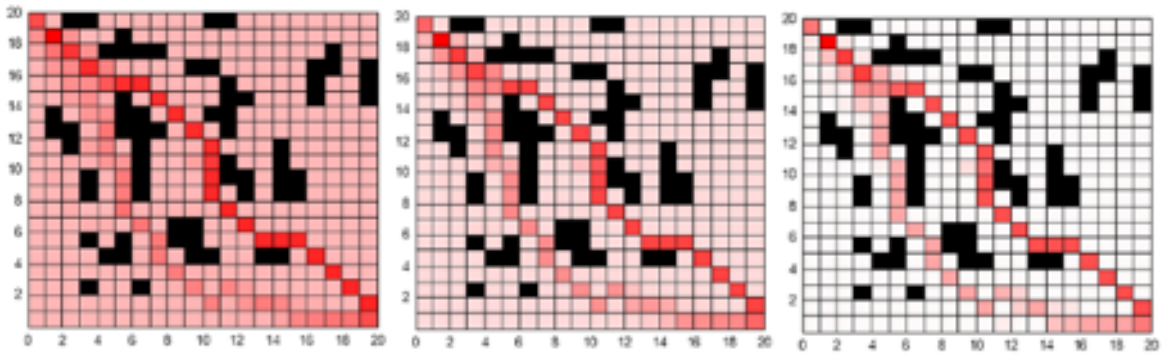


FIGURE 3.21: Ant pheromone distribution using the improved ACO algorithm from [79] after 6, 8, and 10 iterations, showing rapid convergence.

Further advancements were proposed by [75], who built upon the improved ACO algorithm from [79] by dividing the global path into smaller sub-paths. As the robot moves, LiDAR data are used to update the environment dynamically, allowing the algorithm to re-optimise local routes in real time. This hybrid approach reduces computational load and improves adaptability to environmental changes while maintaining efficient route planning.

Although these improvements significantly enhance classical ACO, the method still faces challenges in scalability and smoothness, particularly for larger autonomous platforms that require continuous, stable trajectories. For this reason, modern approaches increasingly turn to machine learning techniques, especially neural networks, which offer greater adaptability

and perception-driven decision-making capabilities.

3.6.3 Neural Networks

Neural networks were first proposed in the 1940s [80], but their practical application was limited at the time due to computational and data constraints. However, in recent years, advances in hardware and the availability of large datasets have enabled neural networks to become central to many modern autonomous driving systems. Given their demonstrated effectiveness in the field of artificial intelligence, the alternative methods described previously are not evaluated in this thesis. As neural networks form a foundational component of the work presented in chapters 5 to 8, a more detailed discussion is provided in the following sections.

3.7 Understanding Neural Network Architectures

AI architectures are evolving at a rapid pace, substantial research is being conducted to explore the limits of AI looking at different architectures and approaches. This section provides the foundational background needed to understand the selection of transformer and state space models (SSMs) used in this and the following chapters.

There are different approaches that can be explored when using AI models. A modular approach, for example, uses a data driven NN to replace specific mathematical model subtasks for autonomous driving, such as path planning [81, 82], which is then fed into a global or local controller, as described in the previous chapter. Other approaches seek to integrate NNs with visual SLAM techniques [83] to enhance their overall performance. The blended approach ensures that some portions of the autonomous driving is deterministic, reducing the risk of poor model output negatively impacting the system. A widely explored alternative is to create an end-to-end network that takes a set of data and outputs the drive commands directly. This has been shown to reduce the work load on modular components, as noted in [84], by consolidating the perception, planning, and control aspects of autonomous driving. The focus of this work is on end-to-end solutions in real-life driving scenarios. To contextualise these approaches, this section will introduce the fundamental elements of NNs as well as sequence-based architectures.

3.7.1 Neural networks

Artificial neural networks were first postulated in 1943 by McCulloch and Pitts [80] with ongoing research throughout the 1950s and 1960s - for example, Rosenblatt's perceptron model [85] and Widrow and Hoff's ADALINE [86] - but found limited use. At the time, they required large datasets that were not readily available, as well as a lot of memory, and compute power that could not be feasible obtained at the time.

Interest in NN returned as computers progressively became more powerful, new dedicated hardware solutions were built, and improvements to neural network algorithms were made, such as back propagation which enabled efficient training in layered networks [87]. These improvements allowed NNs to solve more complex tasks in a number of areas, particularly in image, speech, text, and video processing [88]. Today, the ease of use for large-scale NNs, such as ChatGPT and copilot, have revolutionised how people work and operate in the modern world, significantly impacting research and industry.

3.7.1.1 Fundamentals of Neural Networks

A neural network (NN) is a computational model inspired by biological neurons [89]. It comprises multiple layers of interconnected neurons, through which information is propagated from the input to the output layer. Neural networks can be applied to diverse tasks, including classification, regression, and prediction, depending on the structure and activation of the output layer.

For instance, in image classification tasks, the output layer typically consists of a vector with length equal to the number of classes, such as cars, dogs, or pedestrians. Each element represents the network's confidence in the corresponding class, generally ranging from 0 to 1. Neural networks have demonstrated substantial improvements over classical image processing methods, which rely on hand-crafted feature extraction, such as Sobel edge detection and shape matching [90]. Traditional methods are often less robust and computationally intensive, particularly in complex and variable environments.

Neural networks have since been applied to a wide range of domains, including speech and text recognition, fraud detection, content generation, and predictive modelling [91, 92, 93, 94]. A clear understanding of basic neural network principles is essential before exploring advanced architectures such as transformers and state-space models.

A neural network consists of multiple layers, each containing a set of neurons (Figure 3.22).

The input layer receives raw data, while the output layer produces the final prediction. Intermediate layers, called hidden layers, contain the learnable parameters of the network—weights and biases. Networks with more than three layers are commonly referred to as deep neural networks (DNNs).

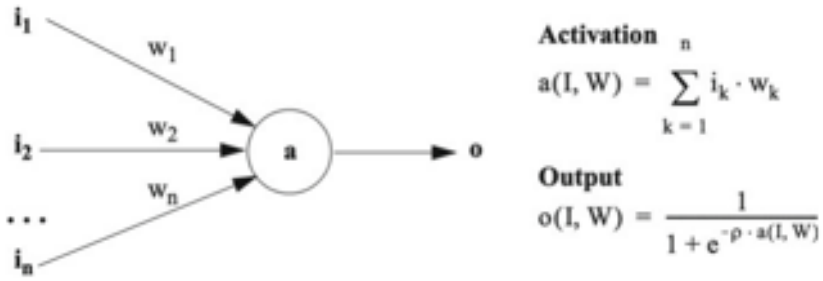


FIGURE 3.22: Single neuron in a neural network [15].

Mathematically, the output h of a neuron in a hidden layer is defined as:

$$h = \sigma(Wx + b) \tag{3.1}$$

where x is the input vector, W is the weight matrix, b is the bias vector, and σ is a nonlinear activation function. Each neuron computes a weighted sum of its inputs, optionally shifted by a bias, which is then passed through an activation function to produce its output.

Conceptually, each neuron comprises three components: a set of weighted inputs, a bias term, and an activation function. In its simplest form, a neuron may use a threshold-based step function, where the output is set to 1 if the sum of inputs exceeds the threshold, and 0 otherwise (Figure 3.23) [89].

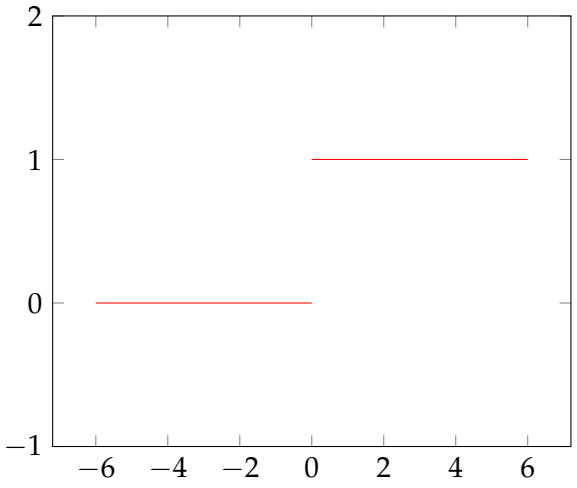


FIGURE 3.23: Step function used in simple neuron models.

However, such step functions are unsuitable for training deep networks, as back propagation requires differentiable and monotonic functions [95, 96]. Consequently, modern neural networks typically employ nonlinear activation functions, which enable effective gradient-based learning. Several commonly used activations are described below.

Sigmoid – Maps inputs to the range $(0, 1)$, commonly used for probability predictions. The function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

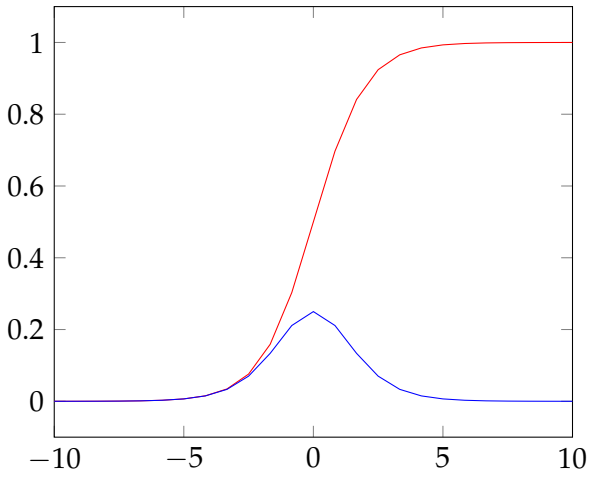


FIGURE 3.24: Sigmoid activation function and its derivative.

ReLU – The Rectified Linear Unit activation, defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{3.3}$$

ReLU avoids gradient saturation for positive inputs and is computationally efficient. However, negative inputs produce zero gradients, which can impede learning. Variants such as Leaky ReLU address this limitation [97].

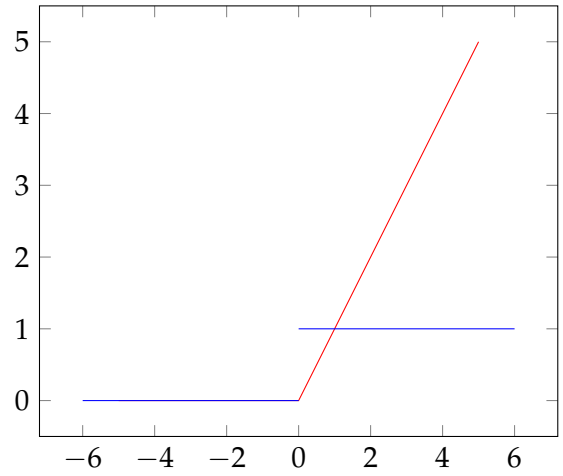


FIGURE 3.25: ReLU activation function and its derivative.

ELU – The Exponential Linear Unit improves learning by pushing negative values towards zero mean, defined as:

$$\text{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{if } x \leq 0 \end{cases} \tag{3.4}$$

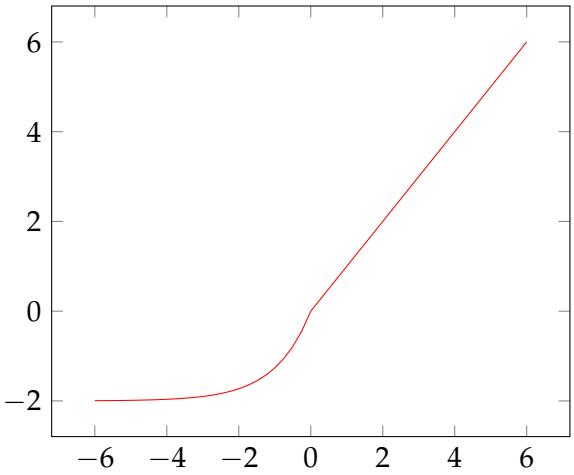


FIGURE 3.26: ELU activation function.

Tanh – Similar to sigmoid but zero-centred, mapping inputs to $(-1, 1)$:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.5}$$

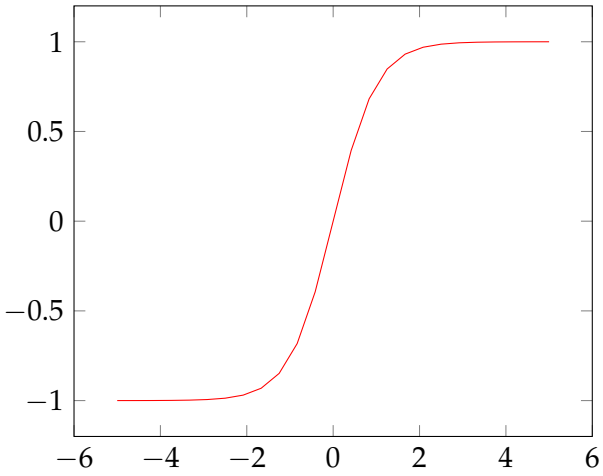


FIGURE 3.27: Tanh activation function.

Softmax – Commonly used in multi class classification, normalises output to sum to 1:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.6}$$

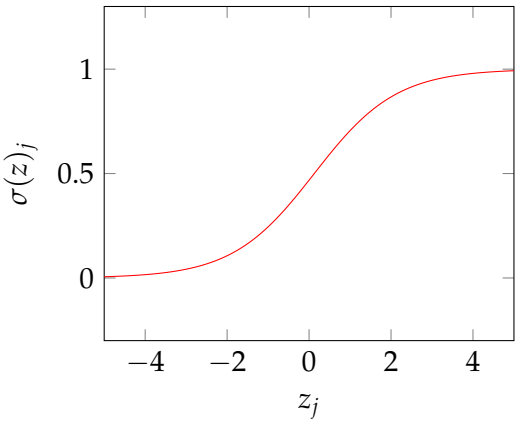


FIGURE 3.28: Softmax activation function for a single output neuron with three fixed logits.

3.7.1.2 Linear Layer

Neural network layers are connected to the preceding and subsequent layers in various ways, the most common being *linear* and *convolutional* layers. A linear layer, also referred to as a fully connected layer, connects every neuron in the previous layer to every neuron in the current layer. These layers can be used independently, though they are more commonly incorporated into larger architectures, particularly near the output stage, where the model generates its final prediction. Figure 3.29 illustrates a simple two-layer network that includes a fully connected layer.

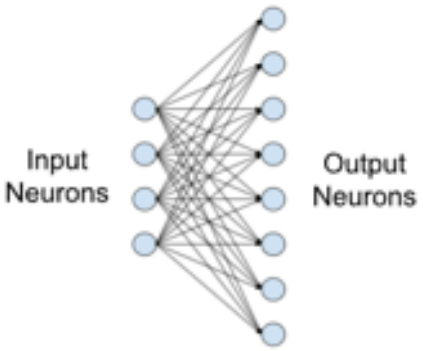


FIGURE 3.29: Example of a simple two-layer neural network using one fully connected layer [98].

In addition, a *bias* neuron is typically added to the input and hidden layers of the model. From a mathematical perspective, the bias allows the activation function of a neuron to shift along the x-axis. Without it, the system would have difficulty modelling functions that do not pass through the origin. The bias neuron is assigned a constant value of 1, and its corresponding weight is adjusted during training to scale the activation function appropriately. Figure 3.30 shows how neuron weighting and bias influence the output.

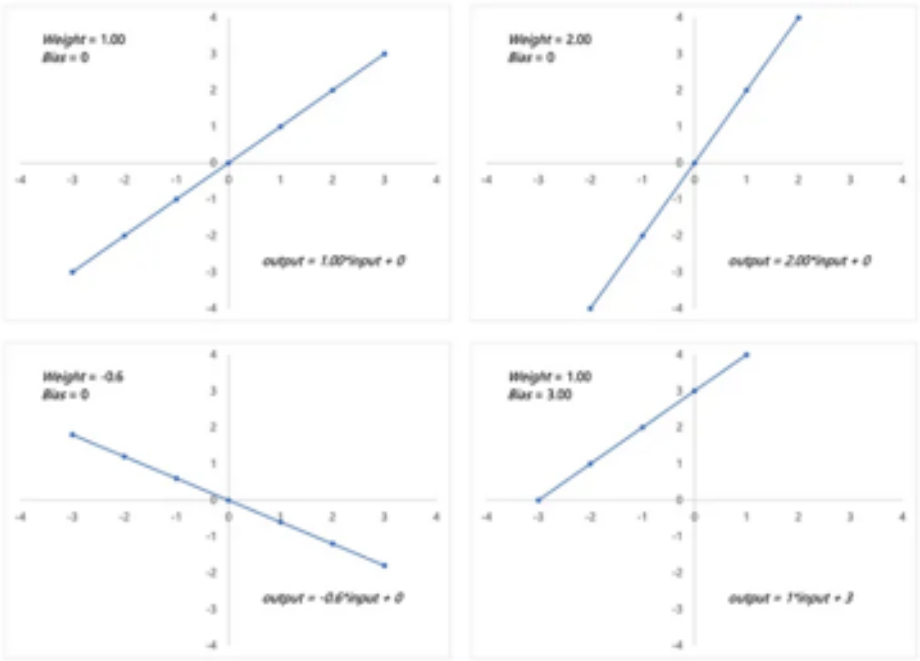


FIGURE 3.30: Effect of neuron weightings and bias on output [99].

3.7.1.3 Convolutional Neural Networks and Additional Layers

Convolutional layers, first introduced by LeCun et al. in 1989 [95], were specifically designed to process data with inherent spatial structure rather than data that can be represented as a simple vector. For example, images are typically represented as two-dimensional arrays, where the spatial arrangement of pixels is essential to their interpretation; shuffling the pixel order would destroy the meaningful content. A key advantage of convolutional layers is that not all neurons are fully connected, which reduces the number of parameters and, consequently, training time. This sparse connectivity also helps mitigate over fitting, as discussed later [100]. These characteristics make Convolutional Neural Networks (CNNs) particularly well-suited for image processing tasks, which are fundamental to autonomous driving.

Convolutional layers preserve spatial relationships by sliding a *kernel* (or *filter*) across the input image [95, 101]. The kernel, typically small and multidimensional, performs an element-wise multiplication and summation at each position, producing a *feature map* that captures local patterns such as edges or textures, as seen in figure 3.31. For colour images, a three-dimensional kernel can be used to account for the multiple colour channels. The values within each kernel are learnt during training, allowing the network to automatically identify the most relevant features for the task [102]. Applying multiple kernels produces several feature maps, each sensitive to different features (e.g. horizontal edges, vertical edges, or textures). CNNs have demonstrated strong performance across a range of image-based applications, including classification, object detection, semantic segmentation, and scene understanding [103, 104, 105, 106].

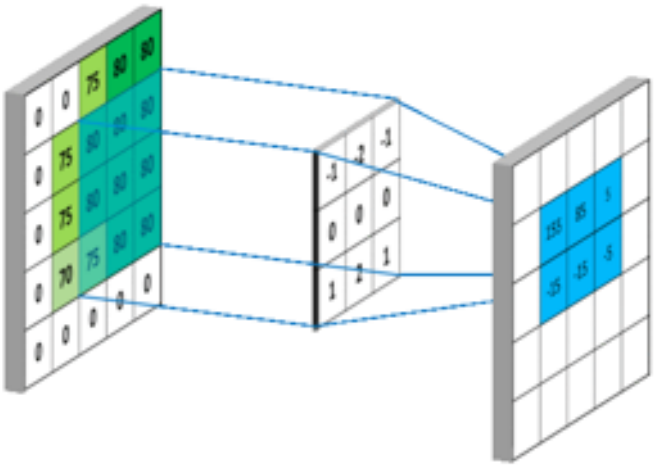


FIGURE 3.31: Example of a CNN creating a feature map using a 3×3 kernel [107].

Beyond convolutional layers, several additional layer types are commonly incorporated into neural network architectures to enhance training stability, reduce over fitting, and improve generalisation. These include pooling, flattening, normalisation, and dropout layers, each of which provides a specific functional benefit as described below.

Pooling Layer Convolutional operations often generate numerous feature maps, leading to a large number of trainable parameters. Although multiple convolutional filters are beneficial for learning diverse features, an excessive number can significantly increase computational cost and inference time. Pooling layers address this by reducing the spatial dimensions of the feature maps through local aggregation functions such as *max pooling* or *average pooling*. Max pooling is particularly common, it selects the maximum value within a local region (e.g., a 2×2 window) to create a down sampled feature map, as illustrated in Figure 3.32.

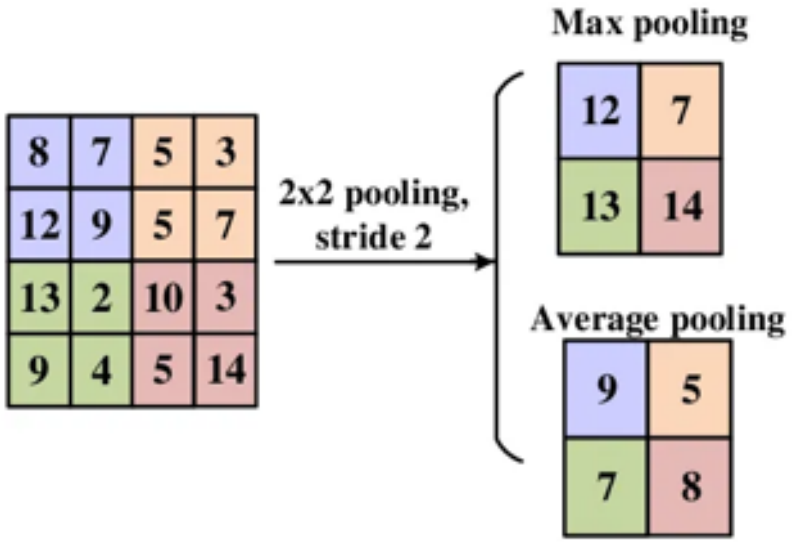


FIGURE 3.32: Examples of max and average pooling, reducing a 4×4 feature map to a 2×2 map [107].

Flattening Layer Following the convolution and pooling stages, the resulting feature maps contain rich spatial information, but it cannot be directly processed by the fully connected (linear) layers used for classification or regression. To bridge this gap, a *flattening layer* converts the multidimensional feature maps into a one-dimensional vector. This vector can then be fed into linear layers for subsequent learning stages, as shown in Figure 3.33.

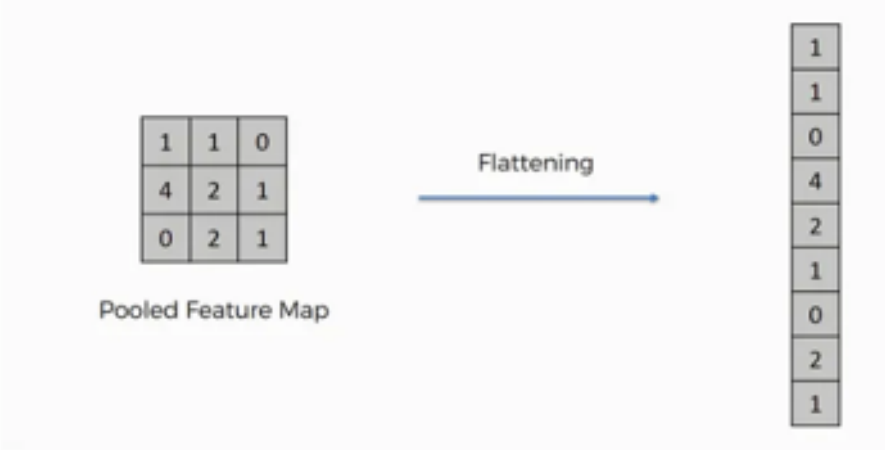


FIGURE 3.33: Flattening layer transforming feature maps into a one-dimensional vector for input to a linear layer [107].

Normalisation Layer In deep networks, weights across layers can vary widely in magnitude during training, sometimes causing certain neurons to dominate the learning process. This imbalance can slow convergence or even lead to issues such as vanishing or exploding gradients, particularly when using bounded activation functions. Normalisation techniques mitigate this by standardising outputs, ensuring that learning remains stable and balanced across the network [108].

Normalisation is also useful in multi-task learning contexts where outputs have different scales. By rescaling outputs to a comparable range, the learning process can balance the loss contributions from each task. Two common approaches are *batch normalisation* and *layer normalisation*. Batch normalisation computes the mean and variance across mini-batches and uses these statistics to normalise activations [109]. Its performance depends on batch size and is less effective for sequential data, as the relationships are disrupted between batches [110]. Layer normalisation, in contrast, normalises across all features within a single layer, making it more suitable for sequence-based models such as those used in natural language processing. However, it tends to be less efficient in CNNs and can introduce additional computational overhead. Figure 3.34 compares these two normalisation strategies.

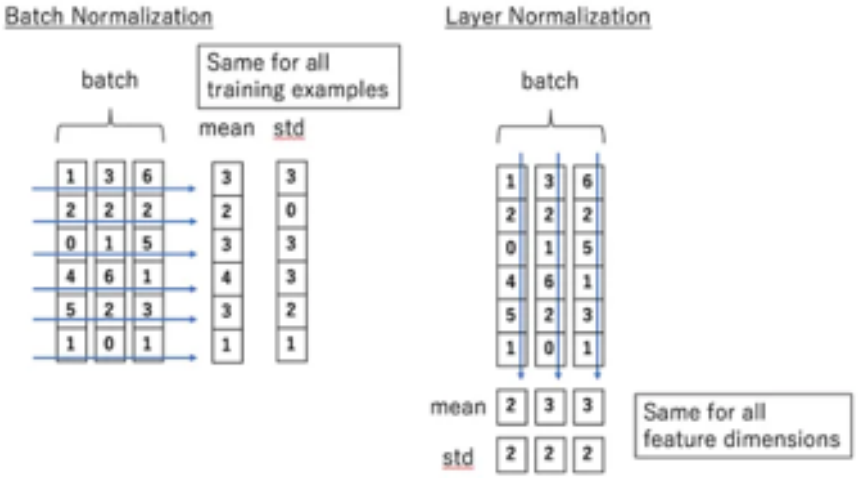


FIGURE 3.34: Comparison of batch and layer normalisation calculations [108].

Dropout Layer In 2014, Srivastava *et al.* [111] proposed *dropout* as an effective technique to combat over fitting, a common issue in neural networks where the model learns the training data too specifically and fails to generalise to unseen data. Earlier approaches attempted to address over fitting by adding regularisation penalties to weight magnitudes [112] or by averaging over ensembles of networks with different configurations [113, 114]. However, these methods were often computationally expensive and impractical for large models.

Dropout provides a more efficient solution by randomly deactivating a subset of neurons during training (Figure 3.35). This prevents the network from becoming overly reliant on specific connections and promotes redundancy in learnt representations, thereby improving generalisation without substantially increasing computational cost.

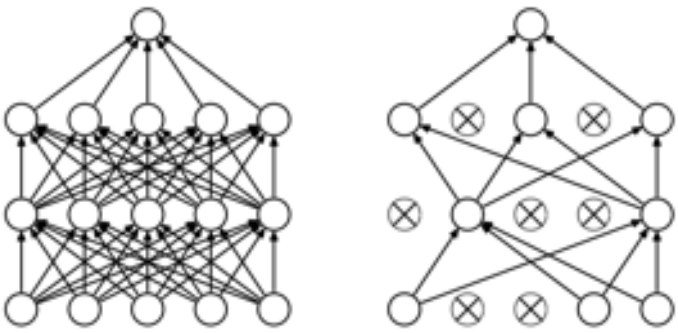


FIGURE 3.35: A standard fully connected neural network (left) compared with the same network after applying dropout (right) [111].

A simple example of a neural network that combines these various layers is shown in Figure 3.36. A handwritten digit image is first processed by a convolutional layer that generates n_1 feature maps of size 24×24. Each feature map is down sampled using max pooling over

a 2×2 window, followed by another convolutional and pooling layer. The resulting n_2 feature maps are flattened into a single vector and passed through two fully connected layers. The first employs a ReLU activation function, while the final output layer includes dropout regularisation to improve generalisation and prevent over fitting.

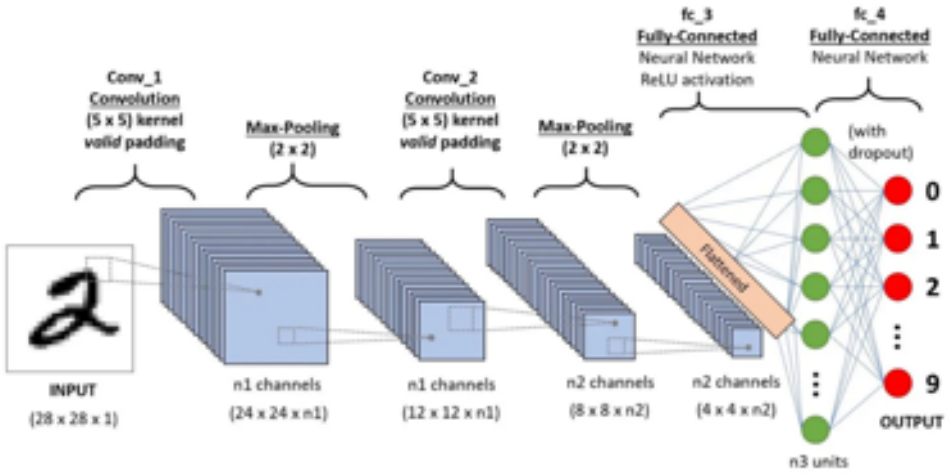


FIGURE 3.36: A simple neural network combining convolutional and linear layers for image classification [115].

Residual Networks As neural networks increase in depth, training can become increasingly challenging due to issues such as vanishing and exploding gradients [116]. Residual networks (ResNets) provide an effective solution to these challenges by introducing *skip connections*, which allow the input of a given layer to bypass one or more subsequent layers and be added directly to the output, as illustrated in Figure 3.37. These connections help preserve essential information across layers, facilitating the flow of gradients during back propagation. Consequently, residual networks improve training efficiency by enabling the network to learn residual mappings, rather than attempting to learn entirely new transformations at each layer.

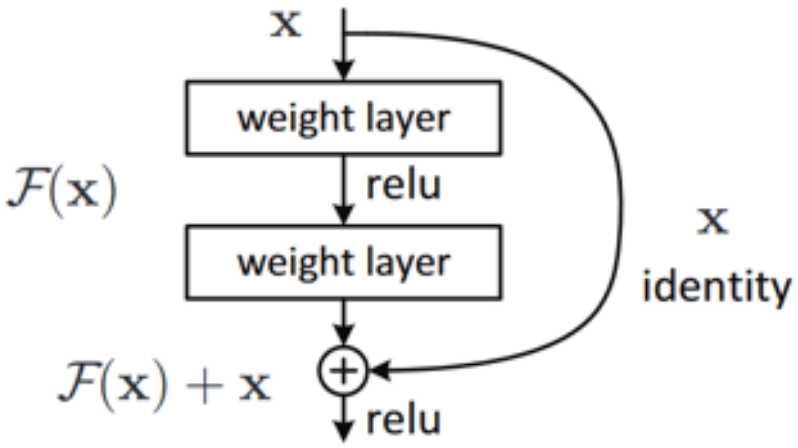


FIGURE 3.37: Illustration of a skip connection in a residual network, showing how inputs bypass intermediate layers and are added to the output [117].

3.7.1.4 Training

When training a neural network, the weights between each of the connected layers are initially assigned random values. The model is then trained using a dataset that is divided into batches to allow for faster and more stable learning. During training, the input data is passed through the network, typically in a single forward direction from the input layer to the output layer. Networks that follow this structure are called feed forward models. Other architectures, such as recurrent networks, pass outputs from later layers back to earlier ones to retain temporal information; these will be discussed later.

At the end of the network, an output is produced and compared with the ground truth (i.e., the correct answer) using a loss function. The loss function quantifies the difference between the predicted and actual values, and this difference is used to update the weights of the network through back propagation. The choice of loss function plays a critical role in the learning process, as it determines how the model interprets and corrects its errors. Some common loss functions are described below.

Mean Squared Error (MSE) Loss

The MSE loss is widely used due to its simplicity and smooth convergence properties when the errors are small and consistent [118]. It amplifies larger errors by squaring them, which encourages the network to prioritise reducing large deviations. However, this same amplification can hinder performance in datasets with outliers or in multi-output problems, where the model may focus disproportionately on reducing one error over others.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.7}$$

Mean Absolute Error (MAE) Loss

The MAE loss is less sensitive to outliers, as it treats all errors equally by taking their absolute difference [118]. This makes it useful when the data contain noise or large variations. In multi-output regression tasks, applying weighted MAE losses can help balance the contribution of each output and improve the overall performance.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.8}$$

Cross-Entropy Loss - Cross-entropy loss is commonly used for classification problems, particularly when the output of the model represents a probability distribution over multiple classes [119]. It measures the divergence between the predicted probability distribution and the true distribution (usually represented as a one-hot encoded vector). The loss increases as the predicted probability diverges from the actual class label, which encourages the model to assign higher probabilities to correct classes.

$$\text{CE} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{3.9}$$

Once the loss value is determined, it is used in the back propagation process to update the model's weights. Back propagation works by calculating how much each weight contributed to the overall loss and adjusting it accordingly to reduce future errors. The goal of training is to learn a function that maps the input space R^n to the output space R^m, where n and m represent the input and output dimensions respectively [96].

The key steps in the back propagation process are as follows:

- The model output is compared with the ground truth using a loss function to determine the prediction error. This error is typically averaged across the training batch.
- The error is propagated backward through the network and the derivatives of the activation functions are calculated. During this process, the gradient of the loss with respect to each weight is computed.
- Each weight is then updated based on an optimisation function thereby moving the parameters in the direction that minimises the loss.
- The model output is compared with the ground truth using a loss function to determine the prediction error. This error is typically averaged across the training batch.

- The error is propagated backward through the network using the chain rule. During this process, the derivatives of the activation functions are applied, and the loss gradient is computed with respect to each weight.
- Each weight is then updated based on an optimisation function, thereby moving the parameters in the direction that minimises the loss.

Optimisers define how the model updates its weights during back propagation to minimise the loss function [120]. Numerous optimisers have been developed, each designed to improve convergence speed, stability, or generalisation. The most commonly used optimisers are summarised below.

Gradient Descent Gradient descent updates the model parameters by applying a scaled negative gradient of the loss function, adjusting the parameters to minimise the loss function. The update rule is given by:

$$W_{t+1} = W_t - \alpha \nabla_\theta f(\theta_t) \tag{3.10}$$

where α is the learning rate and $\nabla_\theta f(\theta_t)$ is the gradient of the loss function with respect to the parameters at iteration t.

Stochastic Gradient Descent (SGD) A variation of gradient descent, Stochastic Gradient Descent (SGD) updates the model parameters after evaluating a single point of the training samples which reduces memory usage working better with large datasets.

Adaptive Gradient Descent (AdaGrad) The Adaptive Gradient (AdaGrad) optimiser assigns each parameter an individual adaptive learning rate based on its past gradients, making it particularly effective for sparse data. The update rule is given by:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot \nabla_\theta J(\theta_t) \tag{3.11}$$

where G_t represents the sum of the elements in squared past gradients and ϵ is a small constant added to prevent division by zero.

Adaptive Moment Estimation (Adam) The Adam optimiser combines the benefits of both momentum and RMSprop techniques to adaptively adjust learning rates. It maintains exponential moving averages of past gradients and their squares, referred to as the first and second moment estimates. The parameter update rule is:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{3.12}$$

where $\hat{m}_t$ is the bias-corrected first moment estimate (mean of past gradients) that helps accelerate convergence, and $\hat{v}_t$ is the bias-corrected second moment estimate (variance of past gradients).

AdamW AdamW is a modification of the Adam optimiser that decouples weight decay from the gradient update, improving both optimisation stability and generalisation performance [121]. The update rule is expressed as:

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta_t \right) \tag{3.13}$$

where λ represents the weight decay coefficient.

Repeating this process over many epochs enables the network to gradually minimise the loss function using a gradient descent approach. To improve computational efficiency, weight updates are often averaged over a batch of samples rather than the entire dataset, a process known as stochastic gradient descent (SGD) [122].

The training process described above falls under supervised learning, where the model learns from labelled input–output pairs. An alternative approach is reinforcement learning (RL), where the model learns through interaction with an environment rather than predefined labels [123]. In RL, the model receives a state as input, performs an action, and observes the resulting change in the environment. A reward is then assigned based on the desirability of the outcome, for example, in autonomous driving, the model may receive a higher reward for making progress along a track and a lower reward for deviating or stopping. The model updates its parameters to maximise cumulative rewards over time, thereby learning behaviours that achieve long-term goals while still exploring alternative actions that might yield better outcomes.

Model architectures are initially defined during setup through a set of **hype parameters**. These hyper parameters specify key attributes of the model, such as the number of layers, the dimensionality of embeddings, the input size, and other architectural or training configurations, but they are not modified during the training process [124]. Careful selection and tuning of these hyper parameters is critical, as they directly influence the model's capacity, generalisation ability, and convergence behaviour.

Following initial training, models can be further refined through **fine-tuning**. Fine-tuning involves adapting a pretrained model, initially trained on a large and general dataset, to a specific downstream task. For example, a model trained for general image classification can subsequently be fine-tuned to detect road markings in autonomous driving scenarios [125].

The pretraining phase allows the model to develop general feature representations and discriminate between object classes broadly, while fine-tuning enhances its ability to accurately detect and classify task-specific features.

During training, models can converge to regions of the loss landscape corresponding to **local minima** that are too narrow for continued effective learning. To address this, **learning rate schedulers** are used to dynamically adjust the learning rate over time, allowing the model to escape shallow local minima and continue effective optimisation [126]. Common scheduler strategies include:

- **StepLR**: Reduces the learning rate by a fixed multiplicative factor (γ) after a specified number of training steps.
- **ReduceLROnPlateau**: Decreases the learning rate when a monitored metric, such as validation loss, ceases to improve over a defined number of epochs.
- **CosineAnnealing**: Modulates the learning rate according to a cosine function over training epochs, promoting exploration while gradually converging towards local minima.

These strategies improve both the convergence speed and the likelihood of reaching a better performing solution in the parameter space.

3.7.1.5 Over fitting and Exploding Gradients

Training deep neural networks introduces several challenges, among which over fitting and vanishing or exploding gradients are the most significant [127]. Over fitting occurs when a model learns patterns that are overly specific to the training dataset, resulting in poor generalisation to unseen data. This issue often arises when the size of the training data set is insufficient relative to the number of trainable parameters in the network.

Various strategies have been proposed to mitigate over fitting. As discussed previously, dropout layers randomly deactivate a subset of neurons during training, encouraging the model to learn more generalisable representations. Regularisation techniques incorporated into optimisers, such as weight decay, further enhance generalisation by constraining the magnitude of network weights. Additionally, recurrent neural network (RNN) architectures reduce the number of parameters by reusing weights across time steps, which can help alleviate over fitting. These will be explored in more detail in the following section.

Vanishing and exploding gradients are another common problem during training. These phenomena occur when the gradients used to update the model's parameters either shrink

to near zero or grow uncontrollably, particularly during back propagation through deep or recurrent structures. This results in either extremely slow learning or unstable divergence. Techniques such as gradient clipping, which constrains the gradient values within a specified range, have been shown to reduce exploding gradients. Normalisation methods, such as batch normalisation, help stabilise training by standardising the activations within each layer. Carefully chosen activation functions, particularly those that preserve gradient flow, also play a key role in mitigating vanishing and exploding gradients.

3.7.1.6 Recurrent Neural Networks (RNNs)

The neural network architectures discussed so far operate only on the current input, without any inherent understanding of temporal relationships within the data. To address sequential dependencies, Rumelhart et al. introduced the **Recurrent Neural Network (RNN)** in 1986 [87]. RNNs enable models to maintain a temporal context by storing information from previous inputs in a *hidden state*, which is then propagated forward through recurrent connections at subsequent time steps, as illustrated in Figure 3.38.

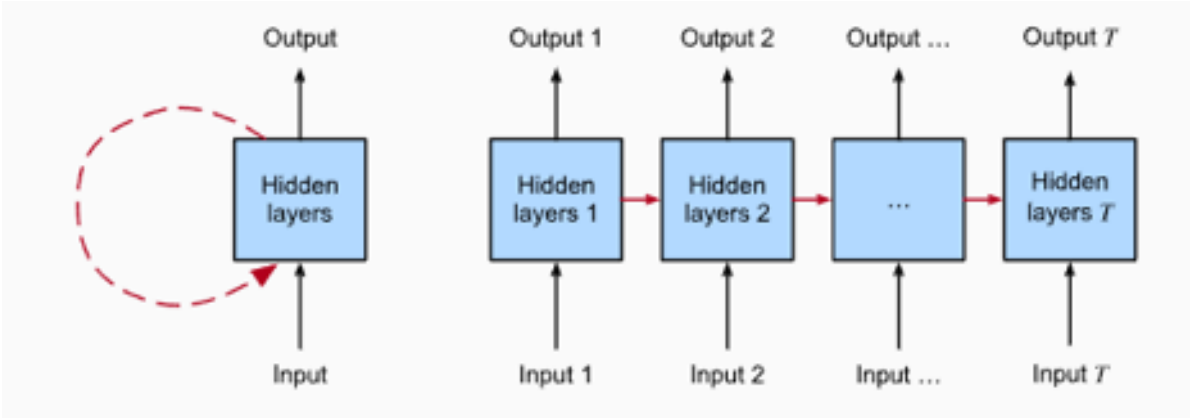


FIGURE 3.38: Recurrent connection over T time steps [128].

RNNs are particularly well-suited to sequential data where the order of inputs is critical, and where the relevance of information may decay over time. For example, in predicting housing prices, sales data from several years ago may be less relevant than recent transactions, while still providing context for long-term trends. The hidden state allows the RNN to capture such dependencies without requiring a fixed-length input window, in contrast to traditional feed forward networks.

Mathematically, the hidden state h_t at time step t is computed as:

$$h_t = \phi(W_h x_t + U_h h_{t-1} + b_h) \tag{3.14}$$

Despite their advantages, *vanilla RNNs* struggle to capture long-term dependencies due to the vanishing gradient problem, where the influence of earlier inputs diminishes as it propagates through the network.

To overcome this limitation, **Long Short-Term Memory (LSTM)** networks were developed. An LSTM cell contains three primary gates: *forget*, *input*, and *output* gates, which regulate the flow of information through the cell [129] (Figure 3.39). The gates control how much of the previous hidden state is retained, how much new information is incorporated, and how much of the current cell state is exposed to the next layer or output.

$$gate_{input} = W_i \cdot x_t + U_i \cdot h_{t-1} + b_i \tag{3.15}$$

$$gate_{input} = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \tag{3.16}$$

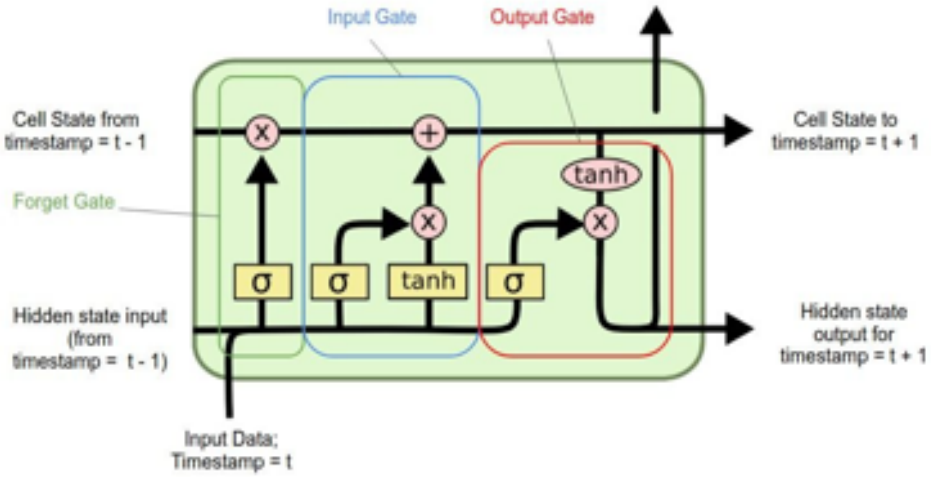


FIGURE 3.39: LSTM cell architecture [129].

The forget gate selectively removes information from the previous cell state, while the input gate determines which new information is added. The output gate generates the hidden state used for subsequent time steps and as the model's output. By explicitly controlling memory updates, LSTM cells mitigate the vanishing gradient problem and enable the network to capture long-term dependencies in sequential data.

Another popular variant is the **Gated Recurrent Unit (GRU)**, which simplifies the LSTM architecture by combining the forget and input gates into a single *update gate* and using a reset gate to control the incorporation of previous states. GRUs achieve performance comparable

to that of LSTMs while being computationally more efficient, making them suitable for tasks with limited computational resources [130].

Overall, RNNs and their variants, such as LSTM and GRU, provide powerful mechanisms for modelling sequential data, enabling temporal context to be retained and leveraged in tasks ranging from time series forecasting to natural language processing. They still, however, exhibit problems with vanishing and exploding gradients. In addition, they cannot be trained in a parallelisable way, expanding their training times.

3.7.1.7 Autoencoders

To improve model generalisation and task accuracy, autoencoders are often introduced. Autoencoders enable models to learn compact and meaningful representations of input data by extracting key features and encoding them into a lower-dimensional latent space. These representations capture the most critical information needed for downstream tasks while reducing redundancy and noise in the input data.

An autoencoder typically consists of three main components: an encoder, a bottleneck, and a decoder, as shown in Figure 3.40. The encoder contains learnable parameters that compress the input data into a lower-dimensional representation. The bottleneck, also referred to as the latent space, stores this compressed information and serves as a compact summary of the input. The decoder then reconstructs the original input data from the latent representation, allowing the model to learn how to efficiently encode and decode information [131].

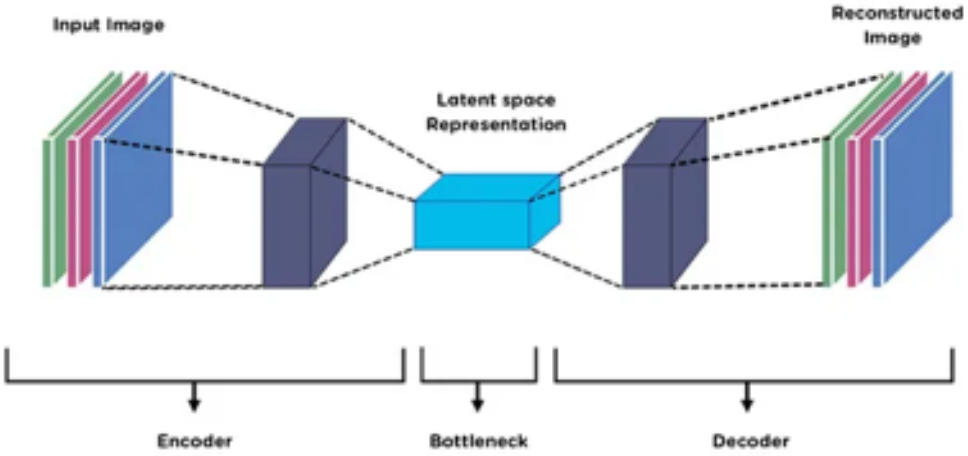


FIGURE 3.40: Structure of an autoencoder [132].

Once the autoencoder is trained to accurately reconstruct its inputs, the decoder component can be removed. The encoder and its bottleneck representation can then be used as a

feature extractor for other models or downstream tasks. This approach allows the model to take advantage of the compressed latent features, which retain the most important characteristics of the original data, to improve performance in tasks such as classification, prediction, or regression.

3.7.2 Transformers

In 2017, the development of transformers marked a significant paradigm shift in neural network architectures. In the seminal paper *Attention is All You Need* [133], researchers at Google proposed a novel approach to constructing neural networks capable of retaining contextual information over long sequences, similar to RNNs, while enabling full parallelisation during training. This architecture introduced the attention mechanism, which models relationships between all elements of the input sequence, selectively weighting the contribution of each element based on its relevance to the task. The transformer architecture demonstrated superior performance compared to prior models and generalised effectively across both large and limited datasets [133]. Today, transformers form the backbone of many large language models (LLMs) such as ChatGPT [134, 135] and Deepseek AI [136], illustrating their significant impact on information processing and natural language understanding.

Unlike recurrent architectures, transformers avoid sequential processing in favour of parallel computation through self-attention. The attention mechanism allows each output representation to consider all input elements, assigning higher weights to the most relevant tokens. Formally, the scaled dot-product attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.17}$$

where Q, K, and V denote queries, keys, and values, respectively, and d_k is the dimensionality of the keys. Each input token is linearly projected onto these three vectors via learnable weight matrices. The *query* represents the token for which the model computes attention, the *keys* represent all tokens to which the query may attend, and the *values* contain the aggregate information according to the attention weights. Attention scores are calculated as the scaled dot product of queries and keys, followed by a softmax normalisation, which produces a weighted sum over the values, generating a context-aware representation that captures the dependencies between tokens.

Multi-head attention extends this concept by applying multiple parallel attention operations to the same set of Q, K, and V projections. Each head operates in a distinct subspace,

capturing different relationships within the input sequence. The outputs of all heads are concatenated and linearly transformed to maintain the original dimensionality, allowing the model to simultaneously attend to multiple aspects of the sequence [133] (Figure 3.41).

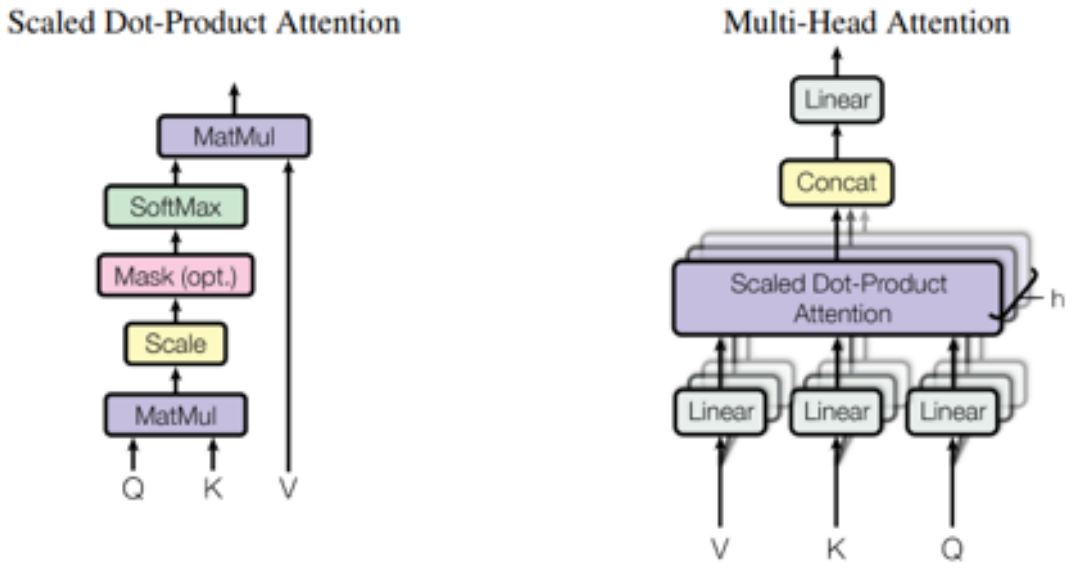


FIGURE 3.41: Multi-head attention mechanism, where multiple self-attention heads operate in parallel to learn diverse relationships among tokens [133].

The transformer architecture builds on the encoder-decoder paradigm, widely adopted in earlier sequence-to-sequence models [137]. The encoder maps an input sequence to a sequence of vectors that represent both positional and contextual information. The decoder then processes this encoding to generate the output sequence. Positional embeddings are added to the input sequence to provide information about the token order, ensuring that the model retains the relative positions of the sequence elements [138].

During decoder training, the model employs *teacher forcing*, where the actual output tokens are provided as inputs to subsequent time steps rather than the model's own predictions. This approach accelerates convergence and stabilises learning [139]. The encoder and decoder blocks, as illustrated in Figure 3.42, can conceptually be viewed as stacked LSTM units, although the transformer replaces sequential recurrence with self-attention mechanisms [139].

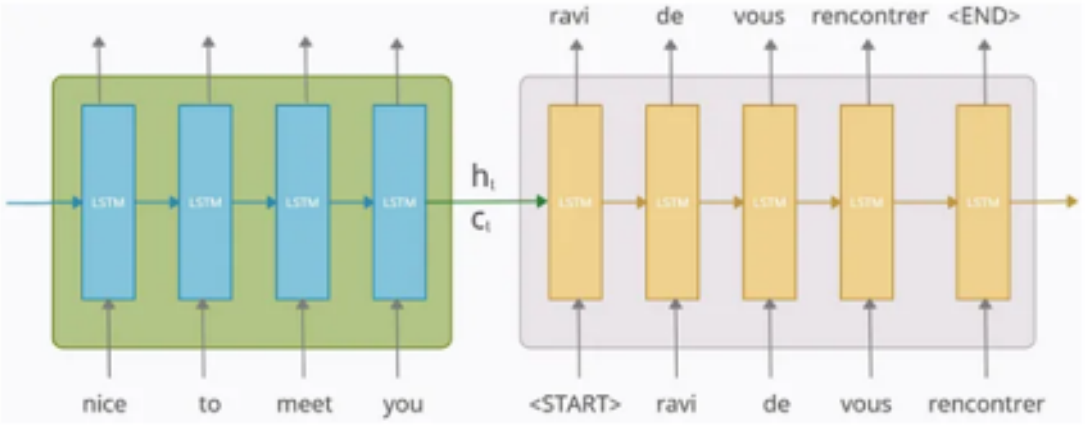


FIGURE 3.42: LSTM view of an encoder and decoder block [139].

Each encoder and decoder block incorporates multi-head attention, residual connections, layer normalisation, and feed forward networks (Figure 3.43). These elements collectively enhance gradient flow, stabilise training, and enable the model to learn complex global dependencies.

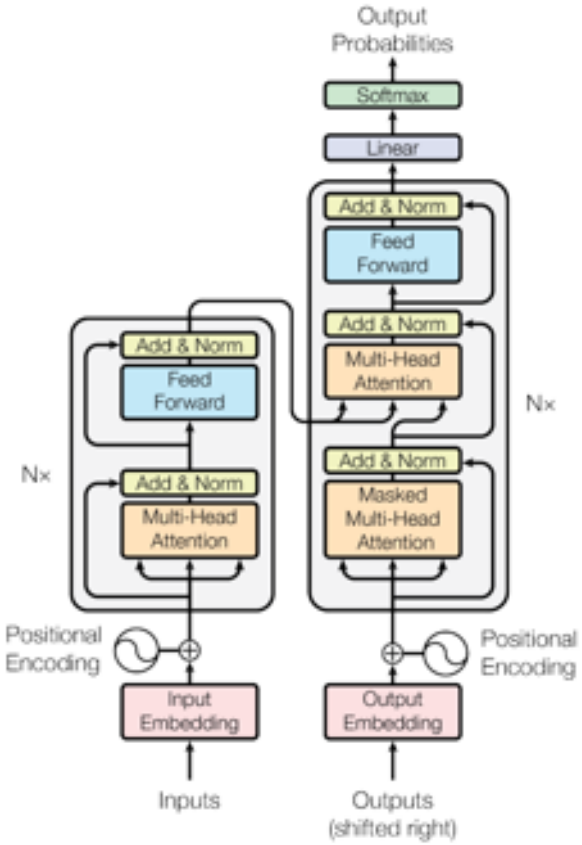


FIGURE 3.43: Complete transformer architecture [133].

A key advantage of transformers lies in their ability to capture both local and global contextual relationships. In contrast, convolutional neural networks have receptive fields limited by kernel size and depth, constraining their capacity to model long-range dependencies. Transformers, through self-attention, establish relationships between all tokens, enabling a more comprehensive understanding of the input sequence.

Despite their strengths, transformer models exhibit a notable limitation: the self-attention mechanism scales quadratically with sequence length, imposing significant computational and memory requirements. Various adaptations, such as the Swin Transformer [140], mitigate this issue by restricting attention to local windows, while emerging approaches such as state-space models aim to retain global contextual understanding with linear computational complexity.

Additional transformer variations include attention masks, which facilitate batching sequences of different lengths, and linear transformers, which approximate self-attention to reduce computational complexity while maintaining performance, albeit with some trade-offs in stability and accuracy.

3.7.3 State Space Models

While transformers have demonstrated exceptional performance across multiple AI tasks, their self-attention mechanism exhibits a key limitation: the computational and memory complexity scales quadratically with the length of the sequence $\mathcal{O}(n^2)$. On high-performance computing systems, this is often manageable, but for embedded systems or resource-constrained hardware, this scaling presents significant practical challenges. To address this limitation, researchers have developed *State Space Models* (SSMs), which aim to maintain long-range contextual awareness while reducing computational complexity. SSMs draw inspiration from classical control theory, representing model states and their updates based on sequential input data [141]. Figure 3.44 illustrates the general structure of an SSM.

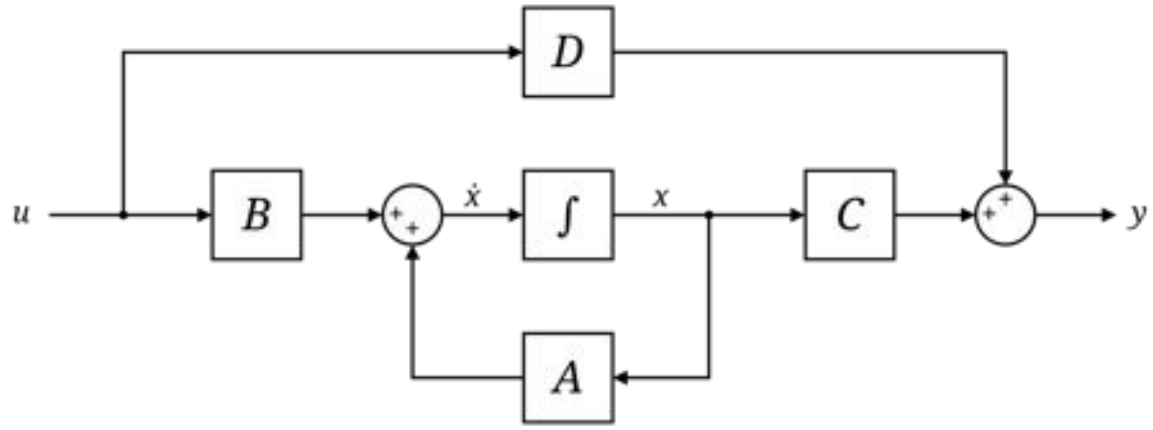


FIGURE 3.44: Typical state space model [142].

At the core of an SSM are four matrices: A, B, C, and D. In neural network implementations, the hidden state is commonly denoted h, analogous to the integral in continuous-time systems. The state evolves according to:

$$h_t = Ah_{t-1} + Bu_t, \tag{3.18}$$

where u_t is the input at time t and h_t encodes a compressed representation of the past sequence. The output is then given by:

$$y_t = Ch_t + Du_t, \tag{3.19}$$

with D often set to zero, as residual or skip connections can capture direct input-output contributions. Continuous-time dynamics can be discretised at intervals Δt, enabling efficient computation via Fourier transforms and allowing the model to be implemented either recurrently or as a convolutional architecture, as shown in Figure 3.45.

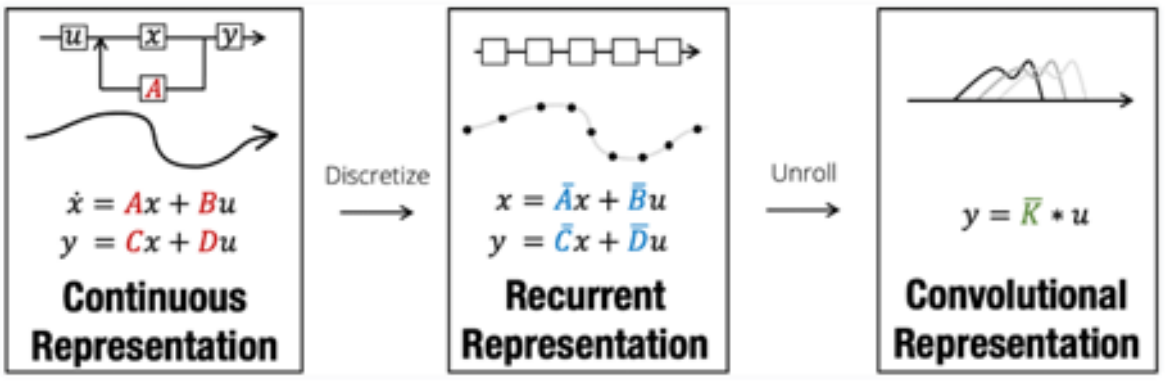


FIGURE 3.45: Recurrent and convolutional representations of a state space model [141].

The linear structure of SSMs allows parallelisation using techniques such as *associative scan* [143, 144], and hidden state updates can be efficiently converted into convolution kernels for frequency-domain computation using FFTs.

The initialisation of the state transition matrix A is critical to the performance of the model. Randomly initialised matrices allow fast training and inference, but often yield poor long-term memory. Structured initialisations, such as those provided by the *HiPPO* framework [145], significantly improve memory retention and mitigate vanishing or exploding gradient issues. In HiPPO-based SSMs, A is designed to optimally project continuous-time signals into a compressed polynomial basis, enabling effective retention of historical information over long sequences. This underlies the S4 SSM, which combines HiPPO-based state representation with discrete-time recurrent and convolutional implementations.

In 2024, Gu and Dao introduced improvements to the S4 model [146], including a selective scan algorithm to filter irrelevant data and hardware-aware optimisations to improve the storage efficiency of parallel scans and kernel fusion [147]. Previous S4 models were limited by fixed linear time-invariant matrices (A, B, C), which restricted the model's ability to focus on or ignore certain inputs and reproduce complex patterns. By adapting the matrices based on input, the models gained flexibility and expressiveness, though at the cost of increased training complexity. Hardware-specific optimisations restored efficiency, resulting in Mamba models that achieve equivalent or better performance than transformers at lower computational cost and scale linearly with sequence length.

In summary, state space models provide an alternative to attention-based architectures, offering near-linear time and memory complexity while retaining the ability to model long-range dependencies. Through careful design of the transition matrices and exploitation of parallel computation, SSMs have emerged as a powerful tool for sequential modelling in resource-constrained environments.

3.7.4 Applications in Autonomous Driving

The first notable implementation of an end-to-end autonomous driving system was ALVINN (Autonomous Land Vehicle In a Neural Network) in 1988 [78]. This system employed a simple neural network to generate steering commands from image inputs. While it demonstrated that neural networks could control vehicle steering, the hardware limitations of the time prevented reliable real-time operation. In 2006, the DARPA Autonomous Vehicle Experiment (DAVE) implemented convolutional neural networks using two camera inputs for off-road

navigation [148]. Again, hardware constraints limited learning capabilities, and the vehicle's operational range before failure remained restricted [149].

Real-time end-to-end control became feasible in 2016 when Nvidia introduced a novel architecture capable of executing autonomous driving commands in real time [150]. Subsequent studies, such as the survey by Apoorv (2023), highlight how this work inspired further end-to-end approaches, including Wayve's 2018 reinforcement learning method for navigation command generation [151] and Intel's 2019 conditional policy model for intersection navigation [152].

Research has also explored AI applications addressing specific sub-tasks in autonomous driving. Paniego et al. demonstrated that even relatively simple AI models could control speed and steering effectively in the CARLA simulator [153]. Ozaibi provided a comprehensive review of CARLA-based end-to-end methods, emphasising the need for standardised benchmarks and evaluation metrics [154]. Separate studies on off-road mobile platforms, such as those by LeCun, show that AI models can detect and navigate around obstacles in real time [155].

With the exception of the off-road experiments, much of the research to date has relied on simulated environments, where conditions are idealised. The majority of models evaluated are convolutional neural networks (CNNs), although Ozaibi notes the emerging application of Transformer-based architectures, which have become prominent for complex AI tasks.

Transformers leverage self-attention mechanisms to achieve a global understanding of the input data. In autonomous driving, Transformers have been applied in several contexts. For example, a Vision Transformer (ViT) model developed at Binus University predicted steering angles from a publicly available dataset [156, 157], outperforming CNN and CNN+LSTM baselines in simulation. Similarly, Vyas et al. (University of Turku) applied a ViT to generate visual odometry data using the KITTI dataset [158], demonstrating improved accuracy over classical geometry-based and prior AI methods. Transformers have also shown promise in traffic sign recognition using German and Persian datasets [159]. Hybrid models combining ViTs and CNNs have been proposed to compensate for the ViT's weaker local feature extraction, achieving high success rates in classification tasks. Extensions of ViT for lane detection [160] and trajectory prediction (ViT-Traj) [161] further illustrate their versatility.

Despite their demonstrated potential, most Transformer-based research has been conducted in simulated environments or benchmark datasets, highlighting a gap in real-world evaluation that can be explored. To address computational limitations of Transformers, recent work has explored State Space Models (SSM). Mamba, a recent SSM variant, reduces

time and resource complexity while maintaining comparable performance. Applications of Mamba in autonomous driving include DRAMA (**Dr**iving with M**amb**a), an end-to-end motion planner integrating camera and LiDAR bird's-eye-view data to predict future ego trajectories [162], outperforming other existing models on the NAVSIM dataset [163]. Other studies have applied Mamba to predict vehicle and pedestrian movements using the Argoverse dataset [164, 165], combining selective attention mechanisms with the Transformer framework to reduce computational overhead [166]. These studies indicate the advantages of Mamba over traditional CNN and Transformer models in real-time autonomous driving tasks.

Given the novelty of Mamba, there are only a limited number of studies, and most have focused on task-specific or driver-assistance applications using real-world datasets. The work presented in this thesis investigates Mamba models in end-to-end real-world driving scenarios, aiming to bridge the gap between simulation-based research and practical deployment.

Recent surveys further emphasise the limitations of simulation-based evaluation. Jia et al. highlight the need for closed-loop benchmarking, noting that many simulation environments do not fully capture real-world driving conditions [167]. Jaeger et al. discuss the potential biases inherent to simulation systems such as CARLA [168]. Industry perspectives, including work by Kirkham, Shirouzu, and Levy (2025), also indicate a stronger emphasis on real-world deployment testing over simulation, as seen in the development processes at Waymo and other companies [169].

Collectively, these studies underscore the importance of evaluating modern network architectures in real-world conditions. The subsequent chapters of this thesis explore the performance of contemporary models in real-time autonomous driving, addressing the gap between simulated environments and practical deployment.

Chapter 4

Autonomous Driving in a Campus Environment

4.1 Introduction

This chapter examines current autonomous driving approaches, particularly on their impact for large-scale vehicles such as the EasyMile shuttle bus, with the goal of identifying methods suitable for real-world shuttle deployment. The selected system must operate safely in a dynamic university campus environment, managing both high and low pedestrian traffic while avoiding collisions with buildings, vehicles, and pedestrians, and maintain predictable and comfortable movement patterns.

The acceptance of passengers and pedestrians is critical for the mass adoption of autonomous vehicles. Achieving SAE Level 4 autonomy requires operation in both known and unknown environments, with the ability to handle all scenarios safely. Current systems, limited to SAE Level 3, require onboard safety operators for nonstandard events. The testing of current systems has been limited to low pedestrian movement areas, so this chapter's focus is on local control in more dynamic pedestrian environments.

The autonomous shuttle bus is equipped with three primary sensors that can be used for autonomous driving, which have been discussed in the previous chapter. GNSS+RTK was excluded due to infrastructure limitations and interference near campus buildings. Given the current limitations in the network and computing resources, an AI approach was also not feasible at the time. This left a LiDAR-based approach as the most practical starting point. SLAM toolbox [170, 171], was adopted for mapping due to its popularity while several different local control and localisation methods are tested for driving on campus.

With these constraints in mind, the specific objectives of this chapter are:

1. Building accurate, robust environment maps and maintaining localisation despite sensor errors.
2. Enabling smooth navigation between key campus locations under pedestrian and environmental variability.
3. Minimising speed disruptions and preserving passenger comfort.
4. Hardening the software stack against failures and enabling safe, automatic recovery.

The chapter is organised into five sections:

1. Brief background on autonomous driving following from Chapter 3.
2. Description of the monitoring safety system and control methods tested.
3. Results from real-world testing on campus environment.
4. Discussion of findings.
5. Conclusion and suggestions for future work.

4.2 Related Work

Commercial autonomous vehicle developers often implement proprietary software frameworks to satisfy strict **Real-Time Operating System (RTOS)** requirements [172]. Alternatives include the DDS middleware standard [173], VxWorks [174], and other RTOS platforms. While these systems provide superior timing performance and reliability, ROS remains popular in research and prototyping due to its modularity and open-source ecosystem. ROS 1 lacks guaranteed real-time performance, whereas ROS 2 introduces improved communication and RTOS integration [175]. Open-source platforms such as *Autoware* have further encouraged the adoption of ROS2-based frameworks within the automotive industry [176].

Safety in autonomous driving remains a fundamental challenge, particularly in urban environments with high pedestrian density, unpredictable behaviour, and complex road geometries [177]. To mitigate operational risks, many commercial systems adopt conservative safety strategies, including:

- Maintaining a human safety driver onboard,
- Operating at reduced speeds, and

- Performing immediate emergency stops rather than dynamic re-routing or evasive manoeuvres.

Recent studies have highlighted the critical role of safety systems in AV performance and reliability. Dixit et al. reviewed AV disengagements and driver responses, showing that increased user confidence may lead to delayed human intervention during system failures, underscoring the need for fail-safe operation and self-recovery capabilities [178]. Banerjee et al. analysed over 144 AVs tested between 2014 and 2017, categorising system failures such as overload, sensor malfunction, limited network bandwidth, and software crashes [179]. Their findings emphasise the importance of robust software architectures and improved safety mechanisms to support higher levels of SAE automation.

Despite inherent risks, commercial deployment of autonomous systems continues. Mercedes-Benz recently introduced its SAE Level 3 vehicle to the U.S. market, implementing advanced safety mechanisms and defining strategic objectives for the expansion of autonomous vehicles [180]. Other companies, such as Tesla and Google, continue to develop proprietary autonomous systems, demonstrating sustained demand for commercial AV technologies.

These studies collectively highlight the importance of integrating safety-oriented design strategies during the early stages of development. They also reinforce the need for real-world testing to identify potential failure points and implement effective mitigation strategies. The software stack and monitoring framework proposed in this work contribute to establishing a robust and risk-aware foundation for autonomous shuttle deployment in complex and dynamic environments.

4.3 System Design

The development of an autonomous vehicle safety system can be conceptualised as a hierarchy of layers, each contributing to overall system reliability. At the foundational level, all nUWAy shuttle buses are equipped with a hardwired safety system implemented via PLC controllers. These controllers typically require a continuous heartbeat signal to ensure safe operation within distributed computing systems when using CANbus commands for control. However, in this study, the interface board described in Chapter 2 was used, rendering the heartbeat signal unnecessary.

Surrounding the shuttle bus is a dynamic safety curtain, composed of four single-layer LiDAR sensors mounted 30 cm above the ground, designed to prevent collisions in the event

of software control failures. The protective range of this curtain varies with vehicle speed, but it does not account for steering angle or vehicle trajectory.

The PLC system also enforces operational limits by constraining requested speeds and steering angles, maintaining vehicle stability and preventing hazardous scenarios such as roll-over. Since the interface board uses the limits of the manual control system as a guide, additional control signal constraints were not required. Additionally, the braking system is designed to fail-safe, automatically engaging in the event of power loss to prevent unintended motion. These hardware-based safety mechanisms are standard across autonomous shuttle platforms and were retained without modification for this study.

Building upon these fundamental safeguards, a second layer of safety is implemented at the software level, providing additional intelligence and adaptability to the autonomous vehicle's control system.

4.3.1 Monitoring Software

During initial testing of the high-level autonomous driving system, it was observed that the software stack was often too unstable to operate the vehicle reliably. The visual display systems frequently failed during operation, requiring either a full software stack restart or, in some cases, a complete vehicle restart. Furthermore, even when the full software stack was running, interactions such as loading maps or setting the global pose estimate frequently resulted in hardware crashes. These critical failure modes exceeded the capabilities of the existing ROS 2 life cycle manager, which primarily monitors individual node inconsistencies.

To address these issues, Lemar Haddad and the author developed a dedicated system monitoring node [181]. The monitoring node is designed to manage the software stack independently of ROS 2 and to recover the system where possible.

The monitoring node operates as a script that sequentially launches nodes in separate `tmux` sessions. This separation prevents interference between nodes and allows precise control over the startup sequence, which was crucial during development, as race conditions could otherwise lead to low-level communication failures. As each subsystem is launched, the primary script verifies node health by checking for active messages on expected topics. Nodes that continue to publish messages are considered healthy. Once the full software stack is operational, the monitoring node continuously checks for failures. If a node failure is detected, the system terminates the corresponding `tmux` session and restarts the node in a new session.

The monitoring node also manages the preservation of critical system data, such as the map. In standard ROS 2 deployments, maps are transmitted only at startup; thus, if a node fails and is restarted, it will lose access to this information. The monitoring node ensures that essential data is resent during recovery, maintaining continuity of operation.

In the event of more severe system failures, the monitoring node is capable of performing a full software stack restart, sequentially shutting down and relaunching all nodes to restore stable operation.

4.3.2 Localisation and Mapping

A critical requirement for the autonomous shuttle bus is the ability to construct and localise within an accurate map of its operating environment. Reliable localisation and mapping are essential for enabling autonomous navigation, ensuring that the vehicle can correctly position itself and plan safe trajectories along predefined routes. In the context of this system, several mapping and localisation frameworks were evaluated to identify suitable solutions compatible with ROS 2.

For map construction, only a limited number of mature ROS 2 packages are available. Among these, the *SLAM Toolbox* [170] was selected due to its robust scan matching and map optimisation capabilities, which allow it to produce smooth and consistent 2D occupancy maps. The *Cartographer* package [182] was also considered; however, it was found to be only partially implemented in ROS 2 and lacked several features required for deployment on the shuttle bus.

This work presents a comparison between the *SLAM Toolbox* and the *Adaptive Monte Carlo Localisation* (AMCL) implementations for localisation in large outdoor environments. While both frameworks are capable of supporting autonomous navigation, they differ significantly in their underlying approaches, scalability, and performance under dynamic conditions. The results presented later in this chapter discuss the relative advantages of each method and identify which approach provides more reliable localisation performance for the autonomous shuttle bus.

4.3.2.1 SLAM toolbox

Accurate localisation is a critical prerequisite for mathematical model based autonomous driving. Although GNSS+RTK provides high-accuracy positioning in open areas, several regions on the UWA campus were identified as dead zones of the GNSS, which substantially degraded positional accuracy. Consequently, Simultaneous Localisation and Mapping

(SLAM) was adopted as a robust alternative. Specifically, the `slam_toolbox` package from ROS 2 was employed due to its proven performance in a variety of applications.

A detailed description of the SLAM algorithm has been provided in Chapter 2; here, the focus is on the key parameter adjustments used for this work. Only non-default parameters are listed; all other parameters were maintained at their default values. These parameters were tuned to balance computational efficiency, map resolution, and scan matching robustness for large-scale campus navigation.

Parameter	Value
Solver Parameters	
solver_plugin	solver_plugins::CeresSolver
ceres_loss_function	HuberLoss
ROS Parameters	
transform_publish_period	0.1
resolution	0.2
max_laser_range	50.0
transform_timeout	0.5
tf_buffer_duration	30.0
stack_size_to_use	10000000000
General Parameters	
use_scan_matching	True
use_scan_barycenter	True
minimum_travel_distance	0.1
minimum_travel_heading	0.1
scan_buffer_size	10
scan_buffer_maximum_scan_distance	5.0
link_match_minimum_response_fine	0.1
link_scan_maximum_distance	2.5
loop_search_maximum_distance	5.0
do_loop_closing	True
loop_match_minimum_chain_size	10
Correlation Parameters (Scan Matching)	
correlation_search_space_dimension	1.0
correlation_search_space_resolution	0.1
correlation_search_space_smear_deviation	0.1
Correlation Parameters (Loop Closure)	
loop_search_space_dimension	4.0
loop_search_space_resolution	0.1
loop_search_space_smear_deviation	0.05
Scan Matcher Parameters	
distance_variance_penalty	0.1
angle_variance_penalty	0.2
minimum_angle_penalty	0.1

TABLE 4.1: Parameters used with SLAM Toolbox

The parameters shown in table 4.1 were selected to optimise scan matching performance, facilitate loop closure detection, and ensure consistent pose estimation while maintaining computational feasibility on the onboard hardware. The tuning process was guided by the specific challenges of the campus environment, including varying pedestrian density, open spaces, and structural features that affect LiDAR measurements.

4.3.2.2 Adaptive Monte Carlo Localisation (AMCL)

AMCL was selected for its computational efficiency and robustness in large-scale outdoor environments. Compared to pose-graph SLAM approaches, AMCL requires fewer resources for map loading and localisation, contributing to improved system stability and reduced mean time between failures (MTBF) during shuttle operation. Most of the default parameters were used with the exception of the alphas which were changed to 0.005.

4.3.3 Local Planners

Although a variety of local planners are available within ROS 2, only a select few were chosen for testing on the shuttle bus in the campus environment. Initially, the TEB and DWB planners were selected, as both have demonstrated strong performance on smaller mobile robots. However, experimental trials revealed that these planners exhibited overly reactive behaviour, which limited their suitability for larger vehicles operating in semi-structured environments. Consequently, the Regulated Pure Pursuit (RPP) controller was also evaluated to determine whether a more stable and anticipatory driving response could be achieved.

As the theoretical operation of these planners has been discussed in previous chapters, the following subsections present only the final parameter configurations used during testing for each model.

4.3.3.1 TEB

Table 4.2 lists the modified parameters used for the TEB local planner configuration.

Category	Parameter	Modified Value
Footprint	footprint_model.type	polygon
	footprint_model.vertices	[[-2.3, -1.2], [-2.3, 1.2], [2.3, 1.2], [2.3, -1.2]]
Obstacles	costmap_converter_plugin	CostmapToLinesDBSRANSAC
	min_obstacle_dist	1.0
	inflation_dist	0.25
Planning	enable_homotopy_class_planning	False
	enable_multithreading	True
	max_number_classes	3
	teb_autoresize	True
	min_samples	5
	feasibility_check_no_poses	10
Optimisation	max_global_plan_lookahead_dist	12.0
	no_inner_iterations	5
	no_outer_iterations	4
	weight_acc_lim_x	1.0
	weight_acc_lim_y	0.5
	dt_ref	0.3
Goal Tolerances	xy_goal_tolerance	2.0
	yaw_goal_tolerance	1.5
Kinematics	weight_kinematics_forward_drive	1.0
	allow_init_with_backwards_motion	False
Steering / Ackermann	cmd_angle_instead_rotvel	True
	wheelbase	2.4
Velocity & Acceleration	max_vel_x	3.0
	max_vel_x_backwards	3.0
	max_vel_theta	1.5
	acc_lim_x	50.0
	acc_lim_theta	0.13635
	min_turning_radius	5.5

TABLE 4.2: Modified parameters for the TEB Local Planner configuration.

4.3.3.2 DWB

Table 4.3 lists the parameters used for autonomous driving with the DWB local planner.

Category	Parameter	Modified Value
General	debug_trajectory_details	True
	short_circuit_trajectory_evaluation	True
	stateful	True
Velocity Limits	min_vel_x	-3.0
	max_vel_x	3.0
	max_vel_theta	1.0
	max_speed_xy	1.0
	min_speed_theta	-1.0
Acceleration / Deceleration	acc_lim_x	1.0
	acc_lim_theta	3.2
	decel_lim_x	-2.5
	decel_lim_theta	-3.2
Sampling	vx_samples	20
	vtheta_samples	20
Transform & Goal	transform_tolerance	0.2
	xy_goal_tolerance	1.5
	trans_stopped_velocity	0.25
Critics	RotateToGoal.scale	32.0
	RotateToGoal.slowing_factor	5.0
	RotateToGoal.lookahead_time	-1.0
	BaseObstacle.scale	0.02
	GoalAlign.scale	24.0
	GoalAlign.forward_point_distance	0.1
	PathAlign.scale	32.0
	PathAlign.forward_point_distance	0.1
	PathDist.scale	32.0
	GoalDist.scale	24.0

TABLE 4.3: Modified parameters for the DWB Local Planner configuration.

4.3.3.3 Regulated Pure Pursuit

The regulated pure pursuit controller used the parameters listed in Table 4.4. The algorithm was slightly modified to stop the vehicle when imminent collisions were detected, rather than allowing the controller to fail. This modification prevented the persistence of stale ROS 2 messages in the `cmd_vel` topic, which had previously caused emergency stop (*e-stop*) activations.

Category	Parameter	Modified Value
Velocity & Acceleration	desired_linear_vel	3.0
	max_angular_accel	20.0
Lookahead Settings	lookahead_dist	4.0
	min_lookahead_dist	3.0
	max_lookahead_dist	6.0
	lookahead_time	0.5
	use_velocity_scaled_lookahead_dist	False
Velocity Scaling & Regulation	use_approach_linear_velocity_scaling	False
	use_regulated_linear_velocity_scaling	True
	use_cost_regulated_linear_velocity_scaling	True
	cost_scaling_dist	6.0
	cost_scaling_gain	0.8
	inflation_cost_scaling_factor	4.0
	regulated_linear_scaling_min_radius	5.0
	regulated_linear_scaling_min_speed	0.15
Rotation & Reversing	use_rotate_to_heading	False
	rotate_to_heading_angular_vel	20.0
	rotate_to_heading_min_angle	1.8
	allow_reversing	True
Collision Checking	max_allowed_time_to_collision	1.5
Goal & Transform	transform_tolerance	2.0
	goal_dist_tol	2.0
Computation	use_interpolation	True

TABLE 4.4: Modified parameters for the Regulated Pure Pursuit Controller configuration.

4.3.4 Global Planners

Nav2 provides several grid-based global planners as interchangeable plugins, primarily utilising either Dijkstra's or the A* algorithm to compute paths. In this work, the *NavFn* and *Smac* [36] planners were compared to evaluate the potential benefits of feasibility-based planning approaches in large outdoor environments. Both planners were integrated within the same navigation framework to ensure consistent conditions, and their performance was analysed in terms of path smoothness, computational efficiency, and robustness to map complexity.

4.3.4.1 NavFn

The NavFn global planner provides a computationally efficient implementation of classical graph-based planning algorithms, including A* and Dijkstra, under the assumption of a circular robot footprint [183]. It operates over a weighted cost map to generate collision-free global paths between start and goal positions. The simplicity of NavFn makes it suitable for large vehicles in structured or semi-structured environments where computational overhead must be minimised.

The configuration parameters used in this study were limited and largely self-explanatory, as summarised in Table 4.5. These parameters were tuned to balance path optimality, computational efficiency, and the tolerance required for accurate way point tracking in the campus environment.

Parameter	Value
`tolerance`	1.5
`use_astar`	`True`
`allow_unknown`	`False`

TABLE 4.5: Configuration parameters for the NavFn (GridBased) planner.

This setup ensures that the global planner provides stable and predictable paths for the autonomous shuttle while remaining robust to minor environmental uncertainties and map errors.

4.3.4.2 SMAC

The SMAC global planner implements a cost-aware, holonomic A* algorithm that continuously updates the global path to enable dynamic obstacle avoidance [184]. Unlike static planners, SMAC actively considers obstacles in the environment and adapts the planned trajectory in real-time, making it suitable for environments with moving pedestrians or vehicles.

Table 4.6 summarises the key configuration parameters used for SMAC in this study. Parameters such as `max_iterations` and `max_planning_time` were selected to balance computational load and responsiveness, while smoothing weights (`smoother.w_smooth` and `smoother.w_data`) were tuned to produce feasible, continuous paths for the shuttle bus.

Parameter	Value
tolerance	1.5
allow_unknown	False
cost_travel_multiplier	2.0
use_final_approach_orientation	False
smoother.max_iterations	1000
smoother.w_smooth	0.3
smoother.w_data	0.2
smoother.tolerance	1e-10

TABLE 4.6: Parameters used with SmacPlanner2D

4.4 Evaluation of the nUWAy Autonomous Shuttle Bus

The results presented in this chapter are structured to provide a comprehensive evaluation of the nUWAy autonomous shuttle bus system. The chapter focuses on three primary areas:

1. Optimisation of shuttle path accuracy and local planner performance.
2. Reliability of mapping and localisation.
3. System reliability and driving performance, including the impact of the monitoring node.

Data for this study was primarily collected along the route between the Law Building and the Science Library over multiple driving attempts. This route presents a variety of

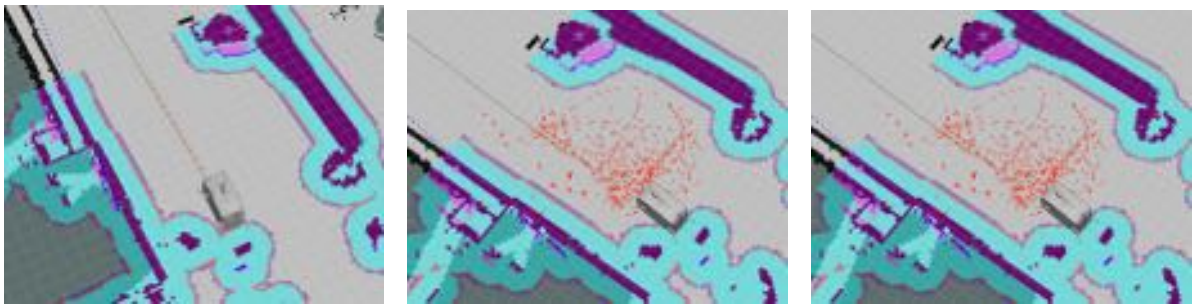
operational challenges, including high pedestrian density during class transitions, dynamic changes in static environmental features such as display signs used for local food vendors, and a combination of open and constrained areas for evaluating localisation and navigation performance. The diversity of the route allows for a detailed assessment of system robustness, path planning effectiveness, and autonomous driving behaviour under realistic conditions.

The following sections present results from these experiments, highlighting system limitations, observed failure modes, and improvements introduced through software and supervisory interventions.

4.4.1 Campus Shuttle Path Optimisation

Initial experiments revealed several limitations in the selection and tuning of the local planner. The Navigation 2 (Nav2) implementation of the Timed Elastic Band (TEB) planner was first adopted due to its widespread use in mobile robotics. However, the performance evaluation showed that TEB frequently produced incoherent or unstable trajectories following slight disturbances, such as pedestrian interference (Figure 4.1). These behaviours resulted in manual interventions and abrupt course corrections, which negatively influenced pedestrian predictability and confidence around the vehicle. The Dynamic Window Approach (DWB) was briefly evaluated as an alternative; however, it exhibited similar tendencies to generate erratic trajectories in crowded environments, often leading to disengagements.

Both TEB and DWB are designed for smaller, highly agile robotic platforms operating in confined spaces. When these planners were deployed on a larger autonomous shuttle operating in pedestrian-dense environments, their behaviour proved suboptimal. Observations indicated that, in potential collision scenarios, pedestrians typically yielded to the shuttle, stepping aside to create space. However, the two local planners did not model this implicit human–vehicle negotiation and consistently attempted to avoid pedestrians, resulting in oscillatory or indecisive motion.



(A) Before interaction (B) During interaction (C) After interaction

FIGURE 4.1: TEB-generated paths become incoherent when disrupted by pedestrians and do not simplify once the disturbance is removed.

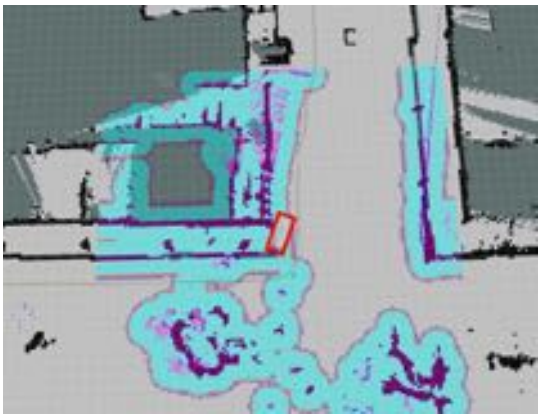
To improve robustness and pedestrian interaction, a simpler local planner, the Regulated Pure Pursuit (RPP) algorithm, was adopted. Although RPP does not support dynamic obstacle avoidance, it adjusts velocity based on proximity to obstacles, resulting in more predictable and human-comprehensible behaviour. To compensate for the lack of dynamic obstacle avoidance, RPP was initially paired with the SMAC global planner, which provides continuously updated global paths. However, this caused excessive replanning, minimal forward progress, and unstable steering. Consequently, NavFn was reinstated as the global planner.

Experimental deployment demonstrated that in campus environments, conservative speed reduction and stopping behaviour when approaching pedestrians was sufficient to ensure safe navigation (Figure 4.2).

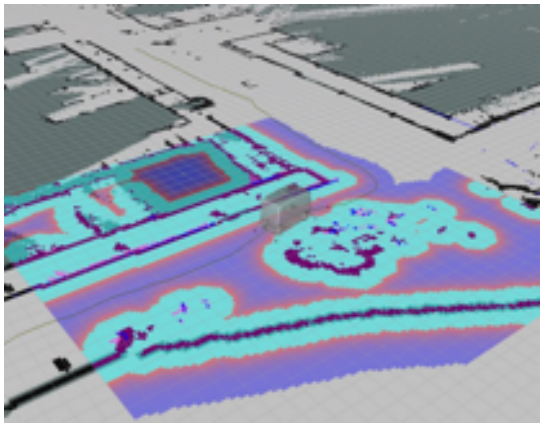
FIGURE 4.2: Vehicle deceleration as pedestrians approach the planned path.

Early trials demonstrated that RPP produced stable trajectories on straight segments but encountered difficulties during cornering (Figure 4.3a). This was attributable to the vehicle's physical footprint not being adequately represented within the planning pipeline. In addition, imminent collision events caused the local planner to terminate rather than recover. To

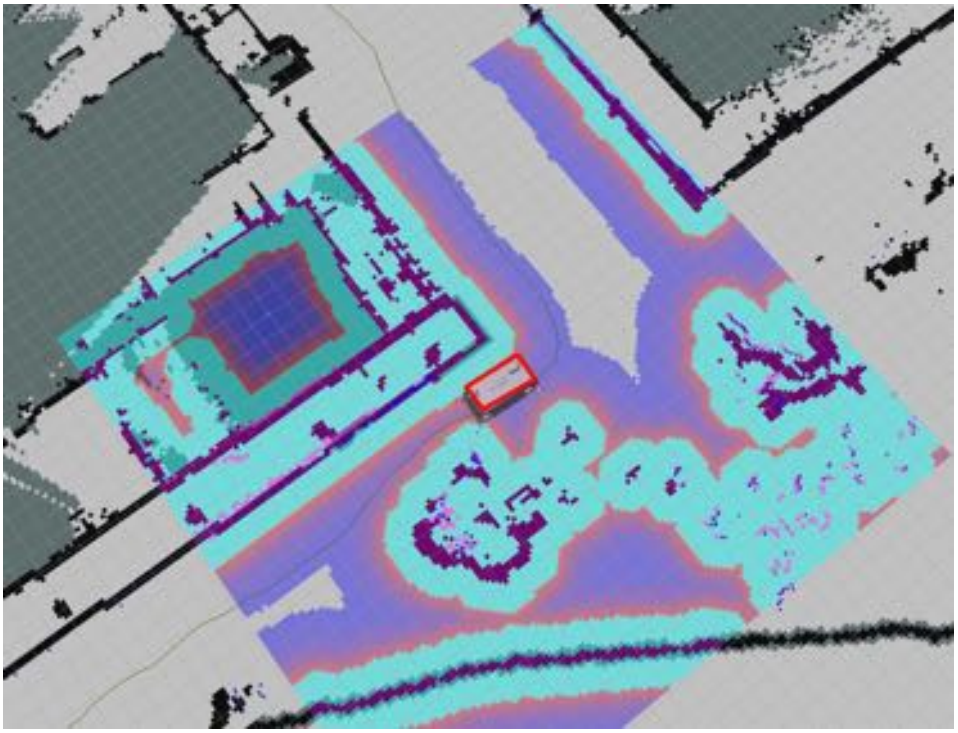
address this, collision handling logic was updated to trigger controlled deceleration and stopping instead of planner failure. The inflation zone around static obstacles was also expanded to exceed the dimensions of the shuttle, allowing safer and smoother cornering (Figures 4.3b and 4.3c).



(A) Corner cutting observed with initial RPP configuration.



(B) Revised global path with extended inflation zone.



(C) Corrected corner navigation following inflation zone tuning.

FIGURE 4.3: Navigation behaviour improvements following inflation zone adjustments.

The nUWAy shuttle was evaluated across nine key campus stops characterised by high pedestrian density, particularly during peak class transitions. Representative trajectories between the Law School and Student Guild stops are shown in Figure 4.4. Minor variations

in initial pose were observed, primarily due to GPS drift and inconsistent Real-Time Kinematic (RTK) corrections. Despite these perturbations, the system demonstrated consistent and repeatable path-tracking performance using the refined planning approach.

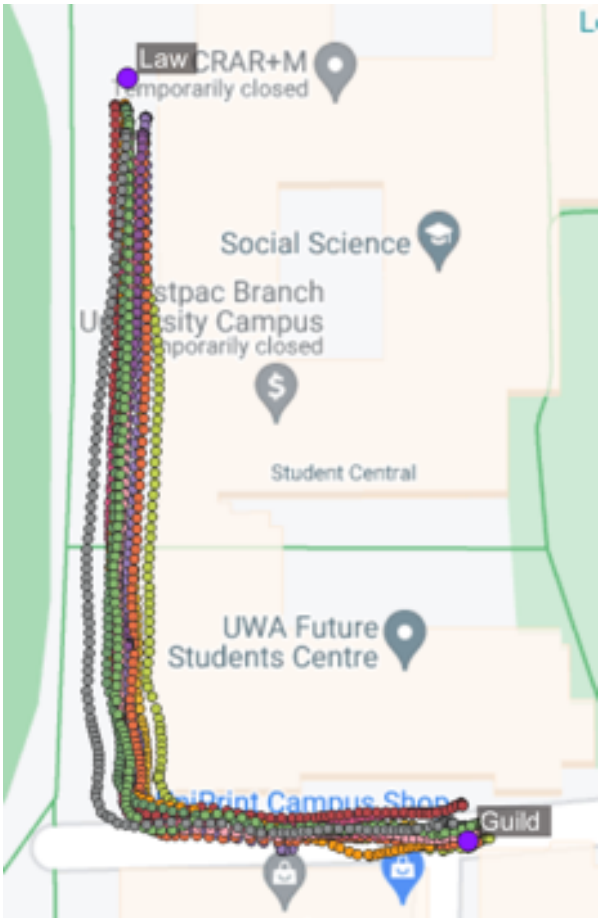


FIGURE 4.4: Multiple recorded shuttle trajectories between two campus stops using the final RPP configuration.

In summary, although the regulated pure pursuit planner does not provide complete dynamic obstacle avoidance, its deterministic and interpretable behaviour reduced the frequency of disengagements and improved pedestrian acceptance compared to more complex local planners. These findings highlight the importance of selecting planning strategies that account for human behavioural patterns in shared urban environments.

4.4.2 Analysis of Disengagement Events

To evaluate the reliability of the system prior to the implementation of the monitoring node, a black-box testing methodology was adopted. System failures were categorised by frequency, severity, and whether a restart was required. During a three-week observation period, totalling 57 hours of autonomous operation, 686 failures were recorded, corresponding to a

mean time between failures (MTBF) of 4.98 minutes. The most prevalent causes of failure were low-level motor driver communication faults, system initialisation errors, localisation failures, and issues in vehicle driving behaviour (Table 4.7).

Source	Before (%)	After (%)
Initialisation	107 (15.6)	1 (1.61)
Driving	154 (22.45)	37 (59.68)
Localisation	413 (60.2)	22 (35.48)
Low-level	12 (1.75)	2 (3.23)

TABLE 4.7: Sources of failure before and after system improvements.

Low-level failures were attributed to unsuccessful data exchanges between the interface board and the PLC controller. Initialisation failures occurred during system startup, typically due to missing or poorly configured node dependencies. Driving-related failures were largely caused by local planner instability, while localisation failures occurred when the system was unable to load a map or maintain a reliable position estimate. Failures were further classified by severity level: *low* — recoverable by non-technical users; *medium* — requiring restart of specific software nodes; and *high* — requiring full system restart or power cycling of the shuttle.

Given that the shuttle is intended for operation by non-technical personnel, efforts focused on mitigating medium- and high-severity failures. Localisation and initialisation failures accounted for 75.8% of total failures before system improvements. Analysis revealed that the primary causes were excessive computational load, large map size, and the complexity of the SLAM solution.

SLAM Toolbox stores maps using a pose-graph structure containing high-resolution pose and feature information. For large-scale outdoor environments, this resulted in unstable map loading and frequent failures. Furthermore, during localisation, sparse environmental features and open spaces degraded scan-matching performance, leading to frequent delocalisation after approximately 50 metres of travel. This also affected map generation, where accumulated drift caused distortions, as illustrated in Figure 4.5.

FIGURE 4.5: Distorted SLAM map resulting from scan-matching failure and drift.

To address this, the map was constructed in smaller sections and combined offline. This approach improved stability, but also highlighted the inherent limitations of 2D SLAM in large outdoor environments. The resulting composite map is shown in Figure 4.6.

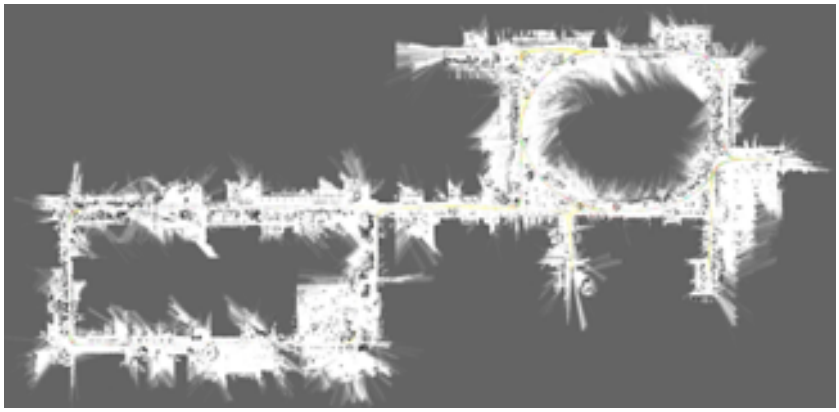


FIGURE 4.6: Improved map generated using sectional mapping and offline stitching.

Localisation reliability was significantly improved by replacing SLAM Toolbox with the Adaptive Monte Carlo Localisation (AMCL) package. AMCL uses a PBM occupancy grid rather than a pose-graph structure, enabling faster loading, reduced memory usage, and more stable runtime performance. This transition led to fewer localisation-related service crashes and more consistent operation (Table 4.8). The MTBF increased from 8.24 minutes

with SLAM Toolbox to approximately 55.91 minutes with AMCL, representing a substantial improvement in system stability.

	SLAM Toolbox		AMCL	
Operation	**Failure**	**Success**	**Failure**	**Success**
Map load (per hour)	5.727	2.009	0.098	3.610
Pose estimations (per hour)	1.004	2.132	0.293	6.244
MTBF (minutes)	8.24		55.91	

TABLE 4.8: Comparison of SLAM Toolbox and AMCL performance.

Low-level communication errors were resolved by improving load balancing, optimising timeout values, and implementing robust fault recovery logic. These enhancements contributed to improved overall operational reliability and safer system performance.

4.4.3 Improvements via Monitoring Node Implementation

The introduction of a supervisory monitoring node within the ROS 2 framework yielded substantial improvements in system robustness and operational reliability. A managed start-up sequence was implemented to ensure that all low-level hardware drivers were initialised prior to the activation of dependent software nodes. This reorganisation of package dependencies reduced initialisation errors and improved system readiness.

Table 4.9 shows a marked shift in the severity distribution of failures, with low-severity failures increasing from 21.7% to 80.7%. This indicates that the majority of failures could now be resolved by non-technical operators without requiring system restart or specialist intervention. Medium-severity failures were reduced from 76.5% to 16.1%, while high-severity failures remained minimal.

Severity	**Pre-Implementation (%)**	**Post-Implementation (%)**
Low	21.7	80.7
Medium	76.5	16.1
High	1.8	3.2

TABLE 4.9: Failure severity levels pre- and post-monitor node implementation.

The mean time between failures (MTBF) improved in all categories after the deployment of the monitoring node and the restructuring of the system (Table 4.10). In particular, localisation failures exhibited a significant increase in MTBF, demonstrating enhanced map stability and reduced node crashes. Launch- and initialisation-related failures also became less frequent because of the controlled startup process. While some categories such as GUI failures persisted, they no longer resulted in system shutdowns or disengagement events.

Category	**Level**	**Initial**	**Final**
Severity	Low	23	25
	Medium	7	123
	High	284	615
Failure Source	Low-level	284	615
	Launch	32	1230
	Localisation	8	56
	Driving	22	33

TABLE 4.10: MTBF comparison for severity levels and failure sources (minutes).

In addition to reducing failure frequency, the monitoring node improved the resilience of the system by introducing partial recovery capabilities. For example, GUI crashes—commonly caused by compatibility issues between visualisation tools and the ROS 2 distribution, no longer resulted in disengagements. Instead, the vehicle continued to follow its planned trajectory while the watchdog process detected the failure and automatically restarted the affected nodes. This significantly enhanced operational stability and safety.

Although upgrades to the ROS 2 software stack have reduced GUI-related issues, the persistence of such failures underscores the importance of modular system design and supervisory fault-recovery mechanisms in safety-critical autonomous systems.

4.5 Discussion

This chapter demonstrated the implementation and evaluation of a mathematical model based autonomous driving system on a large-scale shuttle vehicle operating in a dynamic university campus environment. The system was constrained by limited computing resources and the unavailability of reliable GNSS+RTK signals near campus buildings, which precluded the use of AI-based approaches at this stage. Within these constraints, the results

show that classical local and global planners, such as Timed Elastic Band (TEB) and Dynamic Window-Based (DWB), are not well-suited for environments with high pedestrian density. These planners often produced incoherent or overly complex paths, resulting in manual interventions and unpredictable vehicle behaviour that could negatively affect pedestrian perception and acceptance.

In contrast, the adoption of the regulated pure pursuit (RPP) planner with carefully tuned inflation zones provided smoother and more predictable paths, enabling safer interactions with pedestrians while maintaining continuous vehicle motion. Although RPP lacks full dynamic obstacle avoidance, slowing and stopping in proximity to pedestrians was sufficient to navigate the campus reliably. This finding aligns with the chapter's objectives of maintaining predictable and comfortable movement patterns under variable pedestrian and environmental conditions.

Localisation performance was a critical factor affecting overall system reliability. The use of SLAM Toolbox for mapping initially resulted in frequent localisation failures due to computational limitations and map drift over the large-scale campus environment. Dividing maps into smaller sections improved mapping stability but highlighted inherent limitations of 2D SLAM in open, unstructured outdoor areas. Transitioning to AMCL using PBM maps significantly improved reliability and MTBF, demonstrating that system performance is strongly dependent on efficient map representation and computational load distribution.

The introduction of a supervisory monitoring node within the ROS2 framework further enhanced system reliability by enforcing a managed start sequence and enabling safe recovery behaviours in response to failures. After implementation, the majority of failures were low-severity and recoverable by general users, with substantial increases in MTBF across all failure categories. This confirms that system robustness can be effectively increased through software stack hardening and supervisory monitoring, addressing one of the key objectives outlined in the introduction.

Despite these improvements, limitations remain. Localisation failures still occur intermittently, and the RPP planner assumes slow vehicle speeds and predictable pedestrian behaviour, which may not generalise to high-speed or highly dynamic environments such as public roads. These limitations highlight the need for future methods that integrate more advanced planning, sensor fusion, or AI-based perception while maintaining the reliability and safety demonstrated in this study.

Overall, the findings of this chapter demonstrate that mathematical based approaches, when combined with robust monitoring and careful parameter tuning, can provide a reliable

baseline for large-scale autonomous shuttle operation in complex campus environments. The work establishes foundational knowledge for subsequent exploration of AI-based methods and on-road deployment in less structured environments.

4.6 Conclusion

The primary goal of this chapter was to develop a reliable, deterministic autonomous driving system for a large-scale shuttle bus operating in a dynamic university campus environment. Key findings include:

- Classical planners such as TEB and DWB, while suitable for small robotic platforms, exhibit poor real-world performance on large-scale vehicles in pedestrian-rich environments due to incoherent or overly conservative path generation.
- The regulated pure pursuit planner, with tuned inflation zones, provided predictable and safe navigation suitable for campus conditions.
- AMCL, combined with efficient map segmentation, significantly improved localisation reliability and reduced computational load, increasing MTBF.
- Implementation of a supervisory monitoring node enhanced software stack robustness, enabling safe recovery from failures and improving overall system reliability.

4.6.1 Novel Contribution

Although this chapter does not introduce new algorithmic methods, it contributes a systematic evaluation and application of existing mathematical model navigation tools on a large-scale autonomous shuttle in a challenging and dynamic environment. In particular, the study establishes a baseline framework for reliable autonomous operation, integrates a supervisory monitoring node for software resilience, and identifies practical limitations of commonly used planners and localisation methods in pedestrian-rich outdoor environments.

4.6.2 Future Work

The limitations observed suggest several directions for future research.

1. Integration of AI-based perception and planning methods to handle dynamic pedestrian interactions and unstructured environments at larger scales.

2. Development of 3D or visual SLAM solutions to complement LiDAR-based localisation and mitigate drift in open areas.

3. Exploration of predictive or Model Predictive Control (MPC/MPPI) planners [185, 186] that balance path optimality with real-time safety constraints [187, 188].

4. Deployment on public roads with higher-speed traffic to evaluate system robustness under conditions where human interactions are less predictable.

By addressing these areas, the work can extend beyond deterministic campus driving to generalised, SAE Level 4 autonomous operation, building upon the reliable baseline established in this chapter.

Chapter 5

Vision-based Autonomous Driving on Suburban Roads

5.1 Introduction

Chapter 4 examined the deployment of mathematical model based driving methods for on-campus navigation. The primary limitation of such approaches is that they typically require a highly engineered and well-defined model of the environment, which is time-consuming to develop and often fails to account for all real-world variations [189]. An alternative is to use artificial intelligence (AI) techniques that allow models to learn behaviours through data-driven feedback during training. As discussed previously, AI methods are not new, having first been proposed in the 1940s [80], but only in recent years has computing power advanced sufficiently to make them practical for large-scale autonomous systems.

Currently, the autonomous shuttle bus operates using two convolutional neural network (CNN) models: PilotNet and EfficientNet. Although CNNs are still widely used and continue to perform well, newer architectures have emerged that focus on capturing global dependencies within the input data to improve prediction accuracy. Among these, Transformers have shown remarkable performance across domains such as natural language processing and image classification. However, their quadratic computational complexity makes them less suitable for resource-constrained real-time environments, such as autonomous vehicles. To address this, the emergence of State Space Models (SSMs), including the recently developed Mamba architecture, has generated growing interest for their ability to model long-range dependencies efficiently, offering a potential alternative to Transformer-based approaches.

This chapter investigates the effectiveness of sequence-based vision models for real-world autonomous driving. In 2023, a second autonomous shuttle bus was deployed on suburban roads in Amberton Beach, Eglinton, Perth. As described in Chapter 2, this environment

serves as a test site for future public transport solutions aimed at bridging the "last leg" gap in daily commutes. The autonomous system developed for this setting employs a mixture-of-experts framework (described later in this chapter), designed to achieve specific micro goals within the driving task.

The limited computational resources and future objectives of shuttle bus platforms constrain the model size and necessitate low-latency inference for real-time operation. The models developed here are evaluated against existing CNN-based systems and manually collected benchmark data across several key metrics. The primary evaluation criteria are defined as follows:

1. **Model accuracy and spread** - Measure of the accuracy of the model and its distribution of output accuracies.
2. **Autonomous driving time** - the percentage of time that the model remains in autonomous mode.
3. **Interventions/disengagements** — the frequency of human or hardware interventions required.
4. **Landmark performance** — evaluation of model behaviour at key locations such as roundabouts and intersections.
5. **Ride comfort** — assessed through speed transitions and percentage change in steering angle over time.

The first four metrics are quantitative and objectively measured, while ride comfort, though more subjective, is approximated through variations in speed and steering smoothness. Secondary metrics, including lap time, deviation from the optimal path, and performance in common suburban manoeuvres such as parking and turning, are also analysed.

The chapter is organised into six sections:

1. Background on current driving performance and challenges encountered when transitioning from campus to suburban environments.
2. Description of the sequential models used.
3. Preliminary testing results.
4. Results from vehicle simulation and real-world driving.
5. Discussion of findings.

6. Conclusion and suggestions for future work.

Chapter 3 established the theoretical foundation for AI in autonomous driving. It highlighted that while CNN and Transformer-based models have been applied in this domain, most studies are confined to simulation or static datasets, limiting their generalisation to real-world scenarios. In addition, given the novelty of SSMs, particularly Mamba, their use in autonomous driving remains largely unexplored. This chapter therefore aims to bridge that gap by evaluating SSM-based architectures on their suitability in constrained, real-world environments such as those found in autonomous shuttle systems. Before presenting the models and experiments, the following section outlines the distinct challenges of suburban driving and summarises the performance of the current system.

5.2 Challenges of Suburban Driving Versus Campus Driving and Current Driving Performance

This chapter explores the application of autonomous driving systems in suburban environments. As discussed in Chapter 2, suburban driving presents a markedly different set of challenges compared to campus environments. Campus driving typically involves fewer formal traffic rules, lower vehicle speeds, and a greater number of unpredictable actors such as pedestrians and cyclists. These movements, though less predictable, occur within relatively wide pathways where simple safety mechanisms, such as slowing down and stopping, are often sufficient.

In contrast, suburban driving demands clearer and more deliberate vehicle behaviour. The environment is characterised by higher speeds, narrower lanes, limited visibility, and the presence of more aggressive or inattentive human drivers. In addition, suburban areas often feature a wider demographic range, including children playing near roads, introducing a wider variety of potential hazards. Unlike on campus, abrupt stopping on a public road can cause confusion and frustration for other drivers, which means that safety responses must be more aware of the context and take into account surrounding traffic. Although suburban environments are governed by stricter road rules, they also introduce a greater number of high-risk scenarios that require careful management. The following sections summarise the initial exploration of systems trialled in this context.

As outlined in [84, 190], there are two principal paradigms for developing autonomous driving systems: the *modular* approach and the *end-to-end* approach. In the modular approach, individual subsystems handle specific functions such as perception, localisation, and

control, typical of deterministic designs. The end-to-end approach, by contrast, maps sensory inputs directly to control outputs through a learned model. Chapter 4 detailed the deployment of a modular system using LiDAR-based SLAM for mapping and localisation, coupled with traditional planning algorithms for control in a campus environment. Although this system successfully navigated campus routes, it required extensive parameter tuning and exhibited sensitivity to localisation failures. In a suburban setting such as Eglinton, where the consequences of system failure are more severe, this approach was deemed insufficiently robust.

Subsequently, two main strategies were evaluated in Eglinton: a mathematical model based system and an end-to-end learning model. The mathematical model method used a GNSS + RTK signal for precise localisation. Although this technique was unsuitable for the UWA campus, where signal reflections and electromagnetic interference from nearby structures reduced accuracy, it was initially promising in the open suburban environment. The global path planner was replaced by a single, predefined route stored as a set of way points, passed to the local controller on system startup. Regulated Pure Pursuit, as used in Chapter 4, was selected as the local controller. Due to the narrow geometry of the road, no map-based planning was used, as any buffer zone around the vehicle would have restricted viable navigation space.

During early testing, GNSS covariances showed strong initial precision, ranging between 0.01 and 0.03 m. However, during continuous operation, signal degradation and Wi-Fi dead zones produced intermittent spikes in localisation error. Figure 5.1 illustrates the variation in GNSS accuracy along the X-axis for the 4 km driven route. This approach cannot be considered a true SAE Level 4 system, as it depends on external data sources for localisation and requires highly accurate pre-generated paths. Given these limitations and the risk of localisation errors, an alternative approach was investigated.

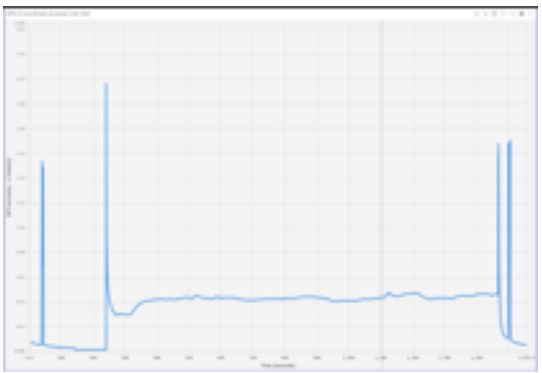


FIGURE 5.1: Accuracy of GNSS+RTK system over time, tested in Amberton Beach.

As an alternative, a convolutional neural network (CNN) based on the PilotNet architecture [191] was developed using imitation learning to follow the main route between the beach and the base of operations. Initially, the model was designed as a monolithic system but exhibited confusion when handling similar visual inputs corresponding to different required actions. To address this, a mixture-of-experts framework was introduced, allowing the system to select specialised sub-models for distinct behavioural tasks. This end-to-end model proved to be more effective for autonomous driving than the GNSS+RTK approach; however, its performance remains limited. Frequent disengagements and manual course corrections indicate instability and sensitivity to environmental variation.

To address these challenges, two avenues of improvement are under investigation. The focus of this thesis is the development of alternative sequence-based models, while complementary work (outside the scope of this thesis) examines the influence of data quality and augmentation on model performance. Before training new models, the dataset was further refined to remove examples of poor driving behaviour, ensuring that all architectures were trained on a consistent and representative dataset prior to applying simple augmentation techniques.

5.3 Models

This study evaluates the performance of seven end-to-end driving models, two baseline CNN models (PilotNet, EfficientNet) and five sequential models. The sequential models consist of two different transformer models (ViT, SwinTransformer), two different state-space models (ViM, VMamba), and one transformer model (DeiT) using an SSM model as a teacher.

The first CNN model was adapted from the PilotNet model developed by Nvidia in 2016 [191], as it is widely used as a benchmark for autonomous driving models. The second CNN model replaces the PilotNet model with EfficientNet [192], which has shown higher levels of performance in image processing while maintaining a lightweight framework.

To determine the impact of sequential-based models, two transformer image classifier models were adapted. The first model uses ViT, introduced by Dosovitskiy et al. [193], as it represents a baseline level of transformer performance. The second model is based on the Swin Transformer [140], which improves the performance of a ViT model and makes it more efficient to overcome the computational demands related to transformer models. Both of these transformer models have shown good results for classification tasks and are repurposed for end-to-end driving.

Recent advances to state-space models have made them more competitive, with a more streamlined architecture that overcomes the issues of transformers. As discussed previously, the new Mamba structure is capable of performing at or above the level of transformer models at a reduced bandwidth which is vital to resource-limited applications such as autonomous driving. Currently, there are two vision classification models based on the new Mamba SSM structure, which have been adapted for use in autonomous shuttle buses. Although both models are called vision mamba, this work will refer to them by the shortened names presented in their respective papers, namely ViM and VMamba. The first model, ViM, is described in the paper by Lianghui Zhu et al. [194], the model uses selective scans forward and backward with a limited state representation to describe the image context for prediction. The second mamba model, VMamba, is described in [195], and presents a similar method to understand image context using an improved cross-sectional scan that has both forward and backward selective scans, as well as up and down scans.

The final model is based on DeiT (data-efficient image transformers) [196], a transformer model that uses a distillation token and a teacher-student learning approach to improve overall transformer performance. Typically, a DeiT model uses a CNN model as a teacher to improve performance, but for this investigation VMamba has been used to see if better performance can be achieved with a SSM teacher.

5.3.1 Behaviour based modular system

Early testing of CNN models revealed that a single monolithic model struggled to predict the correct actions for the variety of driving tasks required to autonomously navigate public roads. A key challenge for the monolithic model was separating the parking in the bays behaviours, where the model needed to parallel park in two bays, from reverse and pullout behaviours. Because the model operated on a single image, it lacked the contextual information required to infer correction actions reliably.

Two methods were explored to create unique contexts for the models. The first method provides additional information at the input of the models to state the correct driving intention. However, additional details for this model do not guarantee that the system will act in the intended manner, and if there are errors in the data labelling it will have a large impact on the model performance.

The second method, which was adopted, uses a behaviour-based modular system. Each behaviour has a model trained on the specific data set for that navigation task, reducing the likelihood of conflicting behaviours introducing errors in the model.

There are a number of benefits to the modular system:

1. **Rapid prototyping** - Smaller models are faster to train and test.
2. **Real-time performance** - Smaller model sizes have reduced inference times, which is critical for real-world deployment.
3. **Mathematical model fallbacks** - Approaches based on rules or classic control systems can be more easily integrated for safety and robustness.
4. **Heterogeneous system** - Different model architectures can be used for each navigation task based on their performance.
5. **Robustness** - Separated models reduce ambiguity in decision making.
6. **Fault tolerance** - System failures due to sensor faults can be mitigated by swapping to running model that uses different sensors while the sensor is restored.

Based on the designated route for the autonomous shuttle bus and the described system, five navigation behaviours are introduced:

1. Lane following - Maintain driving path along the road, including through roundabouts, and maintain lane position on dual-lane roads.
2. Pull-in - Park vehicle in unoccupied parallel bay, ensuring vehicle stops at the end of the bay, is fully contained within the bay and be parallel to the curb without hitting it.
3. Reverse - Vehicle reverses to the end of the bay and comes to a complete stop. The vehicle should remain parallel to the curb without hitting it and still be fully in the bay.
4. Pull-out - Vehicle proceeds into the driving lane smoothly without hitting any curbs or vegetation.
5. Turning at intersections and roundabouts - Move the vehicle through intersection with either a left- or right-hand turn.

Subsequent testing of the initial five behaviours revealed that pull-out and intersection data was unique compared to lane following; therefore, the behaviours were merged into a single behaviour for training. Other behaviours could have been defined, but were deemed not necessary at this stage of the experiment. In future models with a larger driving area, additional and separated behaviours will be needed.

The resulting modular system is shown in figure 5.2. Currently, the choice of model behaviour is controlled purely by controller input by an on-board safety officer. However, future interaction methods, such as speech commands, are planned to take over this system at a later date.

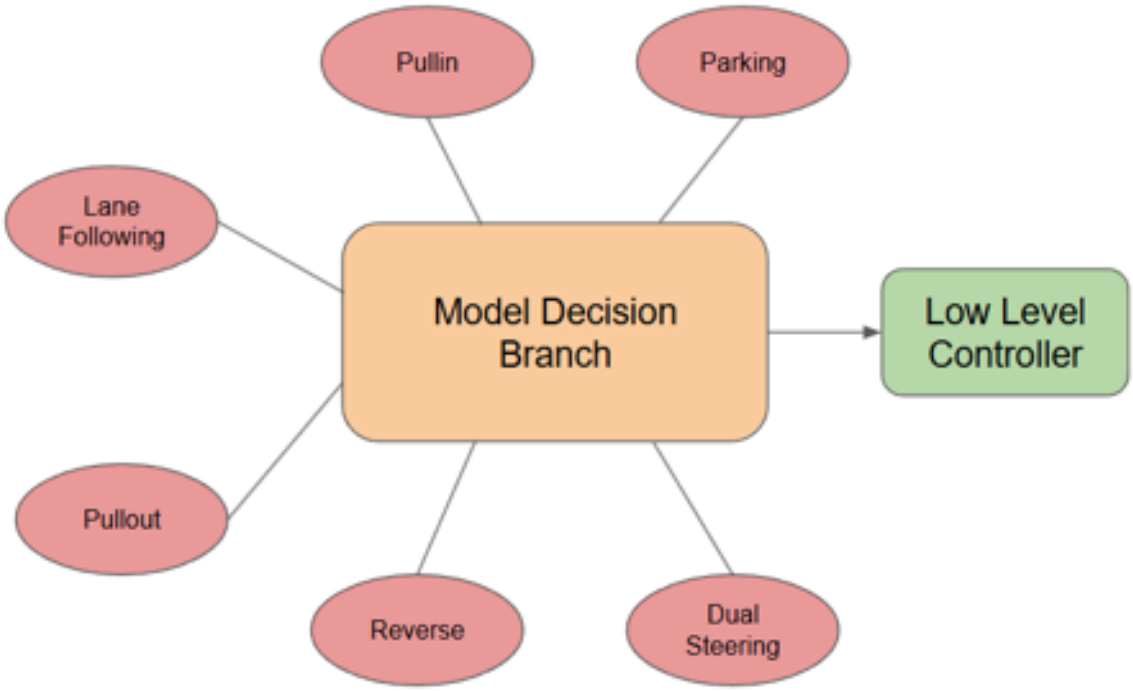


FIGURE 5.2: Autonomous shuttle bus modular AI system.

5.3.2 Dual Linear Regression Model

The general structure of the dual-branch regression model is illustrated in Figure 5.3. The purple block represents the vision processing component, which is modular and can be replaced with different feature extraction backbones. The extracted feature maps are flattened and passed through a linear projection layer (yellow) before being fed into two separate branches of multilayer perceptrons (MLPs, green), each responsible for regressing one control signal: vehicle speed and steering angle, both normalised to the range $[-1, 1]$. Each linear layer (except the final output layer) is followed by layer normalisation and an ELU activation function to promote stable learning and nonlinearity.

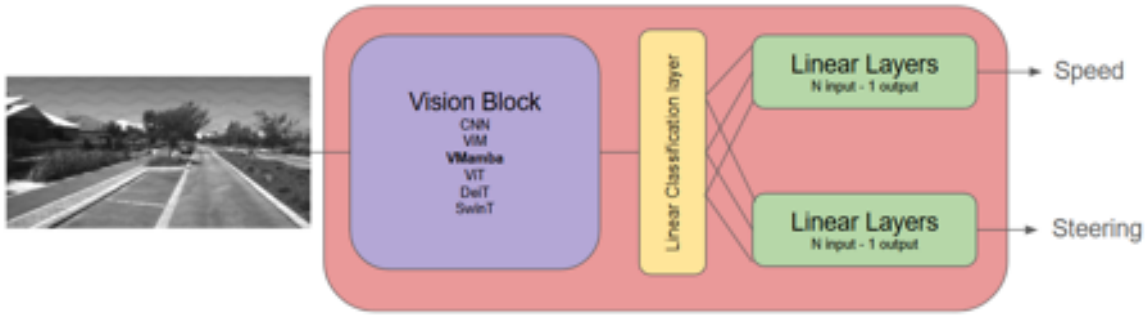


FIGURE 5.3: Example AI module used in the modular system.

The design follows a multitask learning paradigm, where task-specific branches allow independent tuning of the two outputs while still leveraging shared visual features. This approach has been shown to improve performance in autonomous driving tasks [197, 198, 199], and preliminary testing by the REV team confirmed its suitability for end-to-end driving scenarios. Moreover, separating the outputs reduces computational overhead, which is important for deployment on resource-limited hardware or in multi-model systems that may include large language models for human–vehicle interaction.

The dual linear regression design also extends the prior work of PilotNet, which only estimated steering. In contrast, the current implementation predicts both speed and steering. Although speed and steering are inherently coupled in driving dynamics, initial experiments indicated that separating the regression heads produced more accurate predictions. A detailed investigation of the trade-off between joint and separate regression is left for future work.

5.3.3 Classical Model Architectures

Two models based on classical convolutional neural network (CNN) architectures were developed and selected for this study due to their established use in autonomous driving and image recognition tasks. These models were implemented in TensorFlow by a collaborating student who investigated the effects of data set composition and data enhancement on autonomous driving performance. They are included in this thesis as baseline comparisons to evaluate whether the performance gains from modern architectures justify their increased complexity and computational cost. These models do not use the dual linear regression method discussed above. Instead they pass the the classification head through multiple MLP layers and the end with 2 classification outputs, one for speed and one for steering.

5.3.3.1 PilotNet

The PilotNet model was first introduced in 2016 by NVIDIA [191], building on the earlier ALVINN system [78] to demonstrate a purely end-to-end learning approach to autonomous driving. The model employs a deep convolutional neural network (CNN) that receives RGB images from a single front-facing camera and outputs steering commands through a drive-by-wire interface.

The network architecture, shown in Figure 5.4, begins by normalising the three-channel input image. The data is then processed through a series of five convolutional layers, followed by flattening and three fully connected layers, ultimately producing a single steering output.

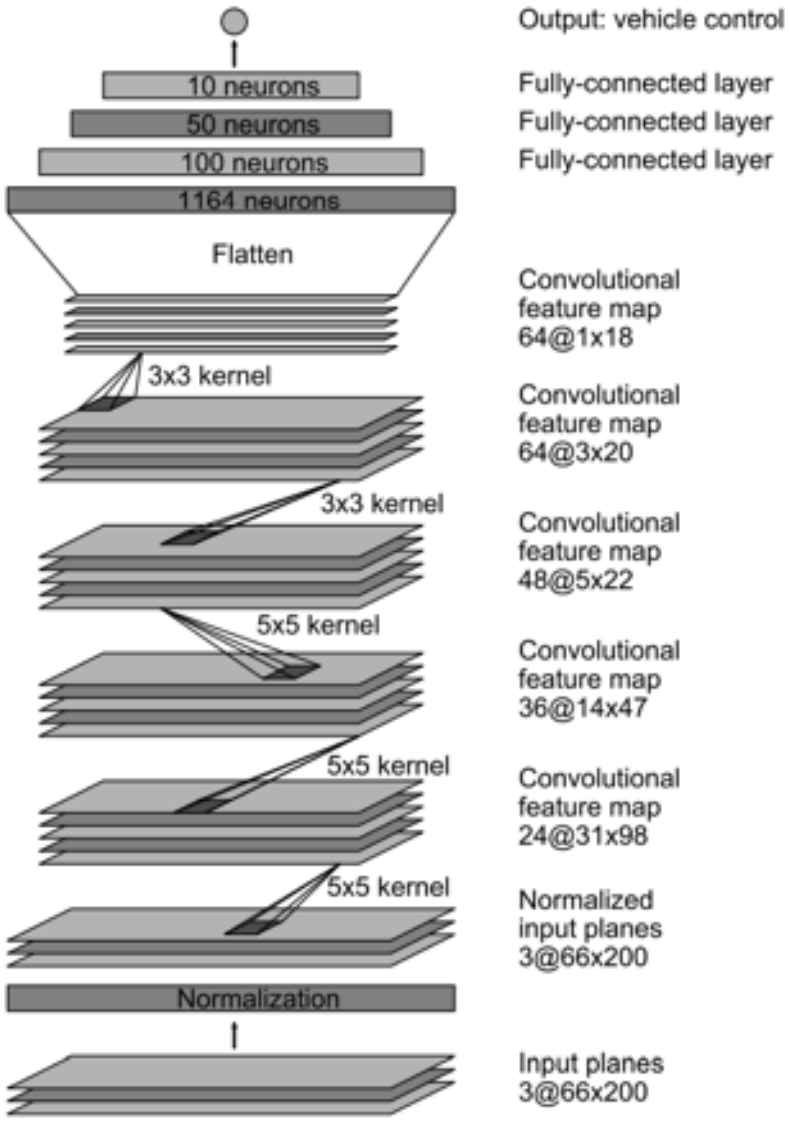


FIGURE 5.4: CNN model architecture used in PilotNet [191]

The final model contained approximately 250,000 parameters, with its primary focus on lane-following tasks. To address dataset imbalance, the training data was up sampled to reduce bias towards driving straight. Model performance was assessed using an autonomy metric, defined in Equation 5.1, which estimates the percentage of autonomous operation as a function of the elapsed time and number of human interventions (assuming six seconds of recovery time per intervention). Using this metric, the PilotNet team reported that the model achieved 98% autonomy in lane-following scenarios.

$$autonomy = \left(1 - \frac{number\ of\ interventions \times 6}{elapsed\ time}\right) \cdot 100 \tag{5.1}$$

The strength of CNN-based approaches lies in their ability to extract hierarchical image features, which is evident from the feature activation maps shown in Figure 5.5. These results highlight PilotNet's ability to detect road edges and other lane-keeping cues directly from raw image data.

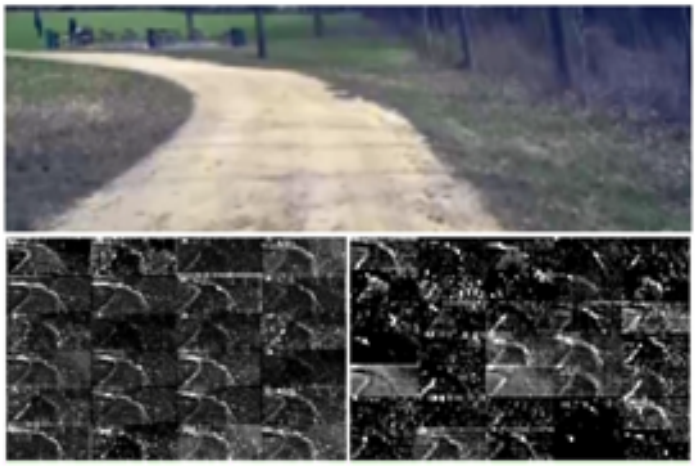


FIGURE 5.5: Feature map from PilotNet illustrating road edge detection [191]

In a later study [149], the same research team described how the model was further adapted in 2016 for deployment on a Lincoln MKZ equipped with adaptive cruise control, allowing a complete end-to-end driving solution. The paper continues to discuss the development of the PilotNet system including data collection and HMI tools developed for user interactions.

For the nUWAy shuttle bus project, the model implemented was based on the original 2016 PilotNet architecture. Given its simplicity, the later variant described in the 2020 paper was not adapted, instead, alternative models were explored as potential successors due to their high levels of performance in image base tasks.

5.3.3.2 EfficientNet

Based on the Keras documentation [200], several high-performance convolutional models could be selected for use on the autonomous shuttle bus. Among these, the EfficientNetV2-S model was chosen as the representative convolutional baseline due to its high accuracy and relatively low parameter count.

Tan and Le introduced the EfficientNet family of models with the goal of developing an optimised suite of CNN architectures that scale uniformly across three key dimensions: depth, width and resolution, while maintaining a balance between accuracy and computational efficiency [201]. This design enables users to select an EfficientNet variant that best matches their hardware and performance constraints.

The baseline EfficientNet model is first determined using neural architecture search (NAS), a guided search method that optimises key model hyperparameters under given resource constraints. Once the baseline is defined, the family of models is generated using the compound scaling rule shown in Equation 5.2.

$$d = \alpha^{\phi}\ w \quad = \beta^{\phi}\ r = \gamma^{\phi} \qquad \text{subject to} \qquad \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \tag{5.2}$$

Here, α, β, and γ are scaling coefficients derived from the baseline model via grid search, while ϕ is a user-defined compound scaling factor that controls the model's size. This approach produces a consistent family of models that achieve near-optimal accuracy–efficiency trade-offs across scales.

In their subsequent work, Tan and Le extended this idea with EfficientNetV2 [202], introducing a new model architecture optimised for both accuracy and training efficiency. The authors redesigned the architecture using a new Fused-MBConv block, which improves computational efficiency in the early network stages (where feature maps are smaller) while retaining the original MBConv structure in deeper layers. This hybrid design reduces training time and memory consumption, making the model better suited for practical deployment in real-time systems.

5.3.4 Sequential Models

Recent advances in deep learning have shifted towards architectures capable of modelling sequential dependencies, allowing the network to capture global relationships and contextual information across the entire image. The following sections describe the sequential models developed for comparison with the baseline CNN architectures. The primary objective is to

evaluate whether these more complex models provide tangible benefits to autonomous driving performance and control stability. All sequential models were implemented in PyTorch, as modern sequence-based architectures are more readily supported within this framework.

5.3.4.1 Vision Transformer (ViT)

The Vision Transformer (ViT) [193] extends the transformer architecture to image data, leveraging its ability to model long-range dependencies through self-attention. Unlike sequential text data, images are inherently two-dimensional, so the first step is to convert the image into a sequence. An input image of dimension $x \in \mathbb{R}^{H \times W \times C}$ is divided into non-overlapping patches using a convolutional operation where the kernel size and stride match the patch size. The patches are then flattened and arranged sequentially in a raster-scan order (left-to-right, top-to-bottom), resulting in a sequence of shape $x \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where P is the patch size and N is the number of patches, as illustrated in Figure 5.6.

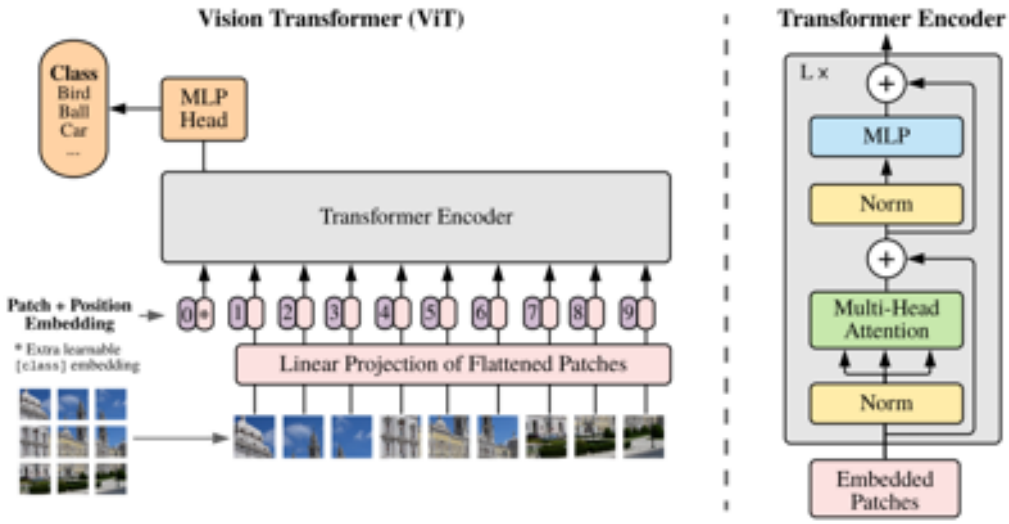


FIGURE 5.6: Vision Transformer architecture [193]

Each patch is embedded into a vector of fixed dimension, forming a token representation that captures local image information. Positional embeddings are added to the tokens to retain spatial information; without them, the model cannot distinguish the original patch positions, potentially producing incoherent reconstructions. ViT uses only one-dimensional positional embeddings, as higher-dimensional embeddings were found unnecessary due to the self-attention mechanism capturing relationships among all tokens.

The key advantage of ViT is its ability to model global dependencies across all patches via self-attention, in contrast to convolutional neural networks (CNNs), which rely on local

receptive fields that expand with network depth. However, ViT typically requires large-scale datasets to achieve strong performance and can struggle on smaller datasets without data augmentation or pretraining.

In this study, ViT serves as a baseline, representing the unmodified vision transformer architecture. The implementation was adapted from the repository [203] and modified according to the general procedure described in Section 4.5.2.

5.3.4.2 Swin Transformer V2

The Swin Transformer [140] was introduced to address the computational challenges of standard Transformers in vision tasks, particularly the quadratic time and memory complexity associated with self-attention. The model divides an image into non-overlapping windows, which are further partitioned into patches and embedded using the same approach as ViT. Each window's patch embeddings are processed through multi-head self-attention, producing feature maps for each local window. This windowing mechanism reduces computational complexity and enables parallelization; however, it limits the global receptive field.

To capture long-range dependencies across the entire image, the Swin Transformer employs a *shifted window* scheme, in which windows are offset at alternating layers (Figure 5.7). This shift allows attention to cross the original window boundaries, effectively modelling relationships between distant patches while preserving computational efficiency.

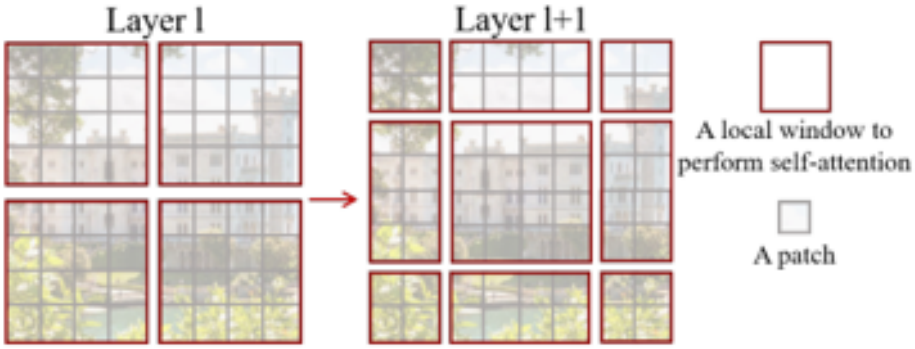


FIGURE 5.7: Shifted window mechanism in Swin Transformer for long-range dependency modelling while maintaining computational efficiency [140].

The architecture is organized into multiple stages, with patch merging layers applied between stages to progressively reduce the number of tokens and increase the feature dimension, improving both efficiency and representation quality. Figure 5.8 illustrates the overall Swin Transformer architecture.

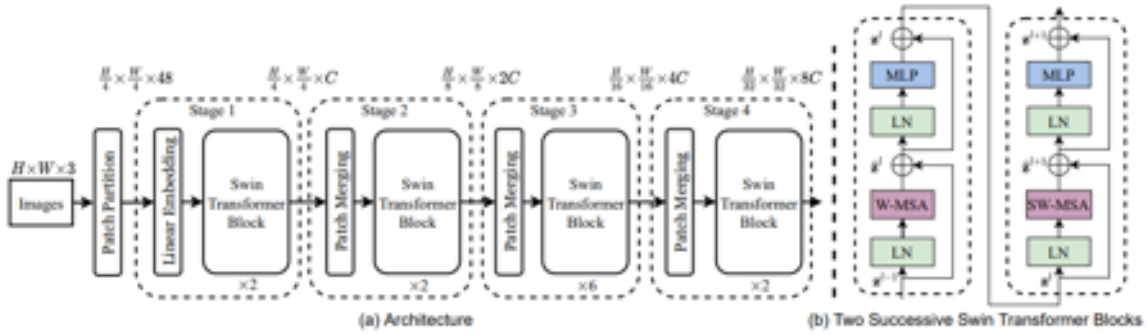


FIGURE 5.8: Swin Transformer architecture [140].

While the original Swin Transformer demonstrated strong performance on image classification and downstream tasks such as COCO object detection [204] and ADE20K semantic segmentation [205], the model exhibited instability and performance degradation when scaled to higher resolutions and larger window sizes. To address these limitations, Swin Transformer V2 [206] introduced three key modifications (Figure 5.9):

1. **Post-attention Layer Normalisation:** Applied after attention and MLP layers to improve training stability at large scales.

2. **Scaled Cosine Attention:** Replaces dot-product attention to prevent numerical instability in high-resolution inputs.

3. **Log-spaced Continuous Relative Position Bias:** Substitutes the original parameterised relative position bias to better generalise across varying window sizes.

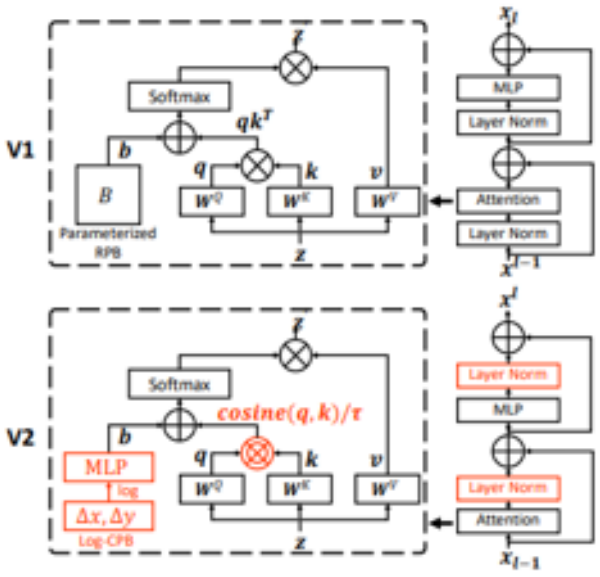


FIGURE 5.9: Key updates in Swin Transformer V2 [206].

The Swin Transformer V2 offers an effective compromise between accuracy and efficiency, making it suitable for real-time vision applications in resource-constrained settings such as autonomous vehicles. In this study, the V2 variant was selected for evaluation and adapted according to the general procedure described in Section 4.5.2.

5.3.4.3 ViM

Vision Mamba (ViM) [194] was developed to address limitations of Transformer-based models in vision tasks, particularly in scenarios requiring rapid decision-making such as autonomous driving. Standard Transformers scale quadratically in time and memory with respect to input sequence length, which can limit their responsiveness to high-resolution images and constrain deployment on resource-limited hardware. Mamba models, including ViM, mitigate these issues by leveraging state-space modeling to build contextual representations with linear computational complexity.

Following the standard Mamba approach, image inputs are first partitioned into patches, as in the ViT model, and tokenised. The tokens are then processed sequentially by the state-space model, but only a fixed-size context is maintained at each step, ensuring linear scaling. Whereas traditional Mamba blocks process sequences in a single direction (suitable for natural language processing), images benefit from bidirectional context due to the two-dimensional structure of visual data. To address this, ViM introduces a *dual-scan* mechanism, processing the patch sequence both forward (left-to-right, top-to-bottom) and backward (right-to-left, bottom-to-top), as illustrated in Figure 5.10. This approach ensures that the state representation for each patch encodes both forward and reverse contextual dependencies.

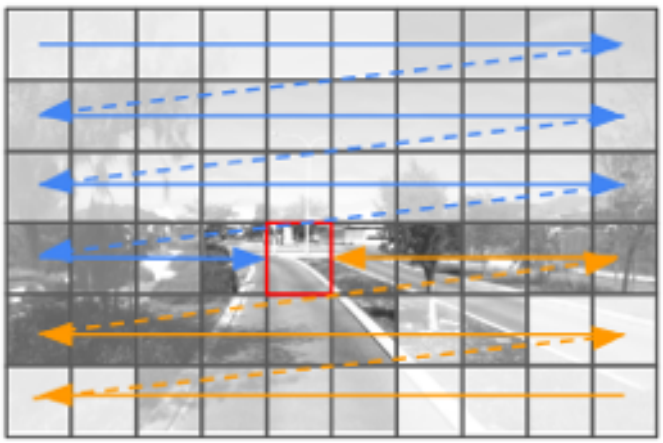


FIGURE 5.10: ViM dual scan mechanism adapted from [194].

The ViM block architecture, depicted in Figure 5.11, adapts the original Mamba block to image data and incorporates the dual-scan mechanism. In addition, a gating mechanism, z, is applied to the state space outputs to balance contributions from the previous and updated states. Multiple ViM blocks are stacked to form the full model, allowing deeper contextual integration. The gating ensures that the model dynamically controls the influence of new versus existing information at each stage, which improves learning efficiency and stability.

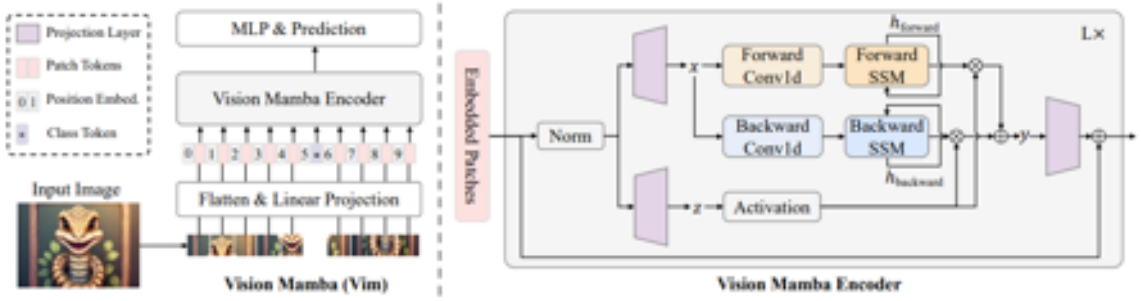


FIGURE 5.11: Vision Mamba (ViM) architecture [194]

ViM represents the baseline for contemporary vision-based state-space models, analogous to ViT in the Transformer domain. Benchmark evaluations in the original study demonstrate that ViM outperforms standard vision Transformers such as ViT and DeiT, while maintaining lower resource usage due to its linear complexity. In this study, ViM was selected as a representative model for evaluating efficient, high-accuracy vision models. The code used for adaptation was obtained from [207] and modified following the general procedure described in Section 4.5.2.

5.3.4.4 VMamba

VMamba [195], also known as Vision Mamba, was introduced as a vision-specific extension of the Mamba state-space framework. Like ViM, it leverages recurrent state updates rather than self-attention, thereby reducing computational complexity from quadratic in Transformers to linear with respect to sequence length. Its central innovation is the *2D Selective Scan* (VSS) mechanism, which generalises the one-dimensional scanning strategy used in Mamba. As shown in Figure 5.12, VMamba performs four directional passes across an image: (i) left-to-right and top-to-bottom, (ii) right-to-left and bottom-to-top, (iii) top-to-bottom and left-to-right, and (iv) bottom-to-top and right-to-left. This design enables the model to capture dependencies both horizontally and vertically, ensuring that contextual information from all spatial directions is encoded efficiently.

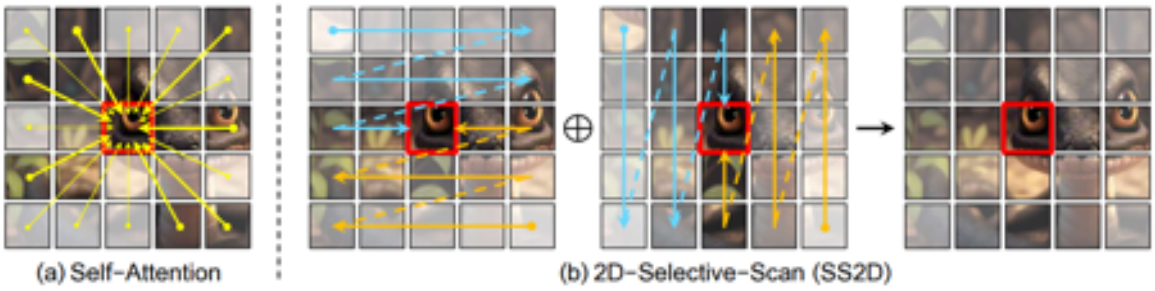


FIGURE 5.12: Vision Mamba (VMamba) 2D Selective Scan [195].

The overall architecture of VMamba is shown in Figure 5.13. Each stage is composed of stacked Visual State Space (VSS) blocks, which adapted the original Mamba block to image data. A VSS block consists of three main components:

1. **Input Projection:** The image patches are linearly projected into an embedding space, where their sequential representation is defined by the selective scan order.

2. **State Update Module:** Recurrent dynamics from the Mamba block are applied to the sequence, updating hidden states while maintaining linear complexity. VMamba integrates the gating mechanism directly into the selective scan step, rather than applying it separately, reducing redundant computation and accelerating the processing.

3. **Output Projection:** The updated hidden states are projected back into the patch embeddings, which are aggregated across scans.

Because spatial relationships are encoded implicitly by the scanning process, VMamba does not require explicit positional embeddings, further reducing memory consumption and model complexity. The design balances computational efficiency with expressive capacity, allowing the model to handle larger images while retaining fine-grained context.

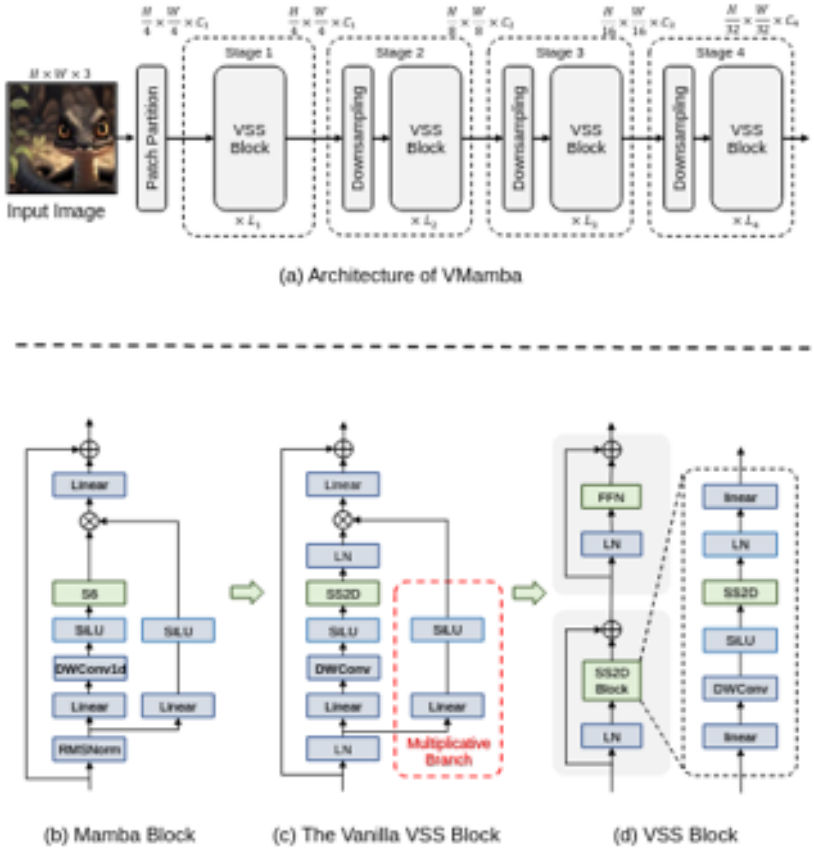


FIGURE 5.13: Vision Mamba (VMamba) architecture [195]

VMamba and ViM were published during the same time period and share the same state-space foundation. However, VMamba's cross-sectional 2D selective scan has consistently demonstrated stronger performance across vision benchmarks, albeit with a larger parameter count. This trade-off reflects a central design philosophy: increased model size and complexity are offset by richer spatial modelling.

In this study, VMamba was selected as one of the baseline architectures to evaluate whether its enhanced capacity translates to improved prediction performance in autonomous driving tasks. The official implementation [208] was adapted to meet the requirements of the shuttle bus control. As with the other models evaluated in this study, VMamba was adapted to the regression-based control task following the procedure outlined in Section 4.5.2.

5.3.4.5 DeiT

The Data-Efficient Image Transformer (DeiT) model [196] builds upon the Vision Transformer (ViT) architecture by introducing a *distillation token*, which enables the model to learn from an external teacher network. During training, each patched image sequence is augmented with a classification token at the beginning and a distillation token at the end before being passed

through the transformer encoder. In the output stage, the classification token is compared with the ground truth labels, while the distillation token is compared to the predictions of the teacher model. Separate loss terms are computed for each, and the mean of these losses are used as the error for back propagated through the network.

Touvron et al. [196] demonstrated that this distillation strategy allows the student network to surpass its teacher, as the two tokens capture complementary learning dynamics. Figure 5.14 illustrates how the tokens are integrated into the model architecture.

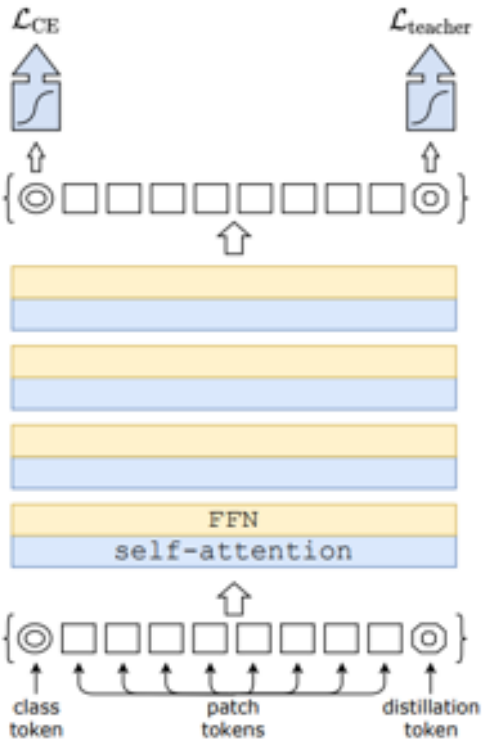


FIGURE 5.14: Distillation token added to the image sequence for training and inference [196].

In practice, the teacher model is typically a convolutional neural network (CNN), as it learns feature representations differently from a ViT model, thus providing complementary supervision. In this work, VMamba has been selected as the teacher network as it captures the temporal and spatial relationships differently. This allows the DeiT model to integrate the advantages of both Transformer-based and state-space architectures.

At inference time, the outputs of the classification and distillation tokens are averaged to produce a final prediction. The DeiT model was adapted from [196] for use within the dual linear regression output structure described earlier, enabling direct comparison to the other vision-based architectures.

5.4 Testing

Prior to full-scale deployment on the shuttle bus, all models were initially evaluated using a small-scale simulator to verify their ability to operate autonomously along a controlled track. Once the models demonstrated reliable performance in this simulated environment, they were adapted and trained using data collected from the full-scale autonomous shuttle bus. Throughout this development process, additional refinements and improvements to the models were identified, which are also described in this section.

5.4.1 EyeSim

The initial models were first tested using the UWA-designed simulation system, EyeSim. A virtual track was developed based on the Carolo Cup layout [31], as shown in Figure 5.15, using an S4 Eyebot designed by Thomas Bräunl. Image data was collected by manually driving the track multiple times in both directions, with each sample stored as a NumPy array containing the captured image and corresponding steering and speed commands.

The data set collected was used to train the models without data augmentation or manipulation, providing an initial check of model viability, training pipeline and establishing the scale of data required for autonomous driving performance. During this phase, key hyperparameters, such as the number of attention heads and model depth—were tuned for each model to best fit the simulation data.

To assess performance beyond standard loss metrics, a custom accuracy function was developed. Since the model could not be expected to predict exact steering and speed values for every frame, this function measured the proportion of predictions falling within a defined threshold of the ground-truth values for both speed and steering angle.

Initial training of the ViM model yielded poor performance until the hyperparameters were properly tuned. The model initially used three MLP layers based on previous work by another student, but was updated to include four layers, which produced improved learning results. After these adjustments, the ViM model successfully trained on just 10,000 image samples, without any augmentation or up sampling to compensate for the imbalance between turning and lane-following data. The trained model was then deployed in EyeSim and successfully completed the Carolo Cup track without error, as shown in Figure 5.15.

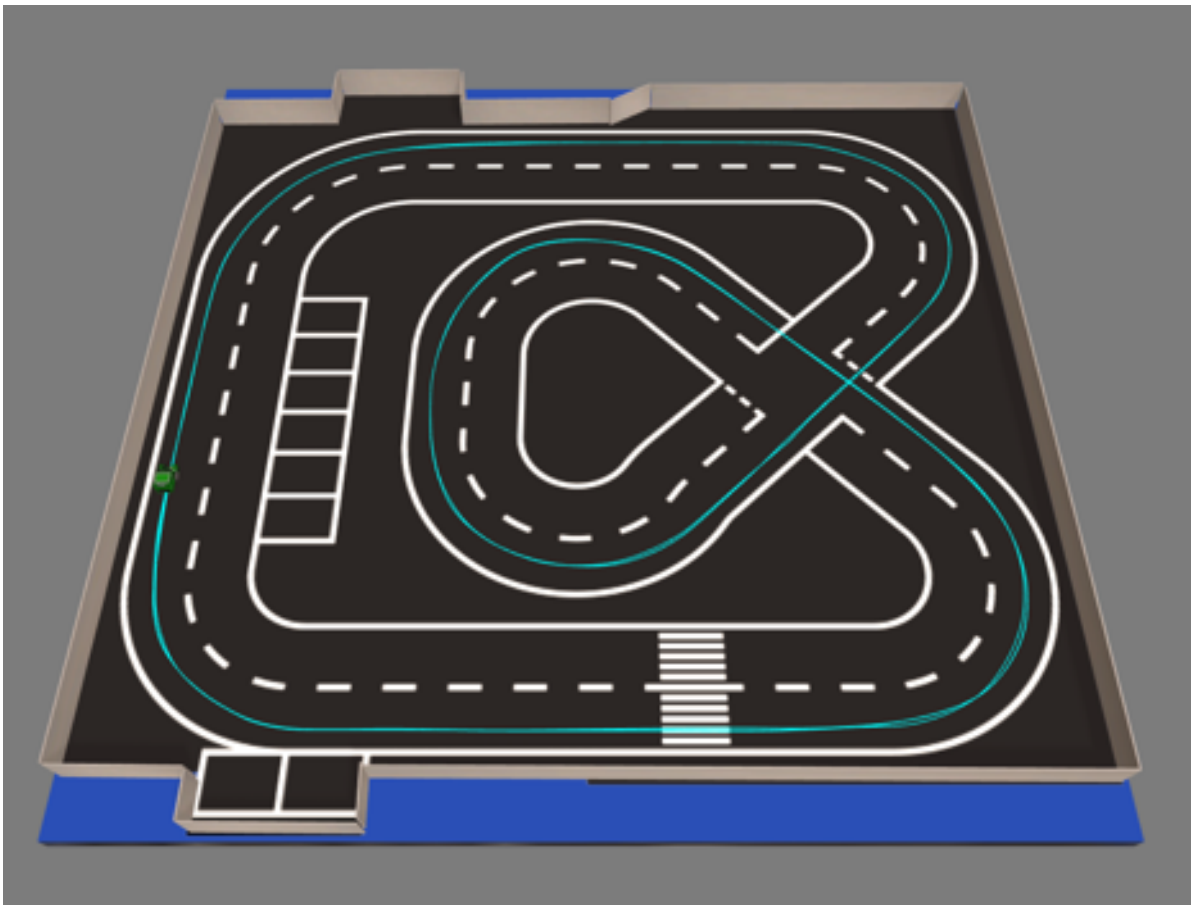


FIGURE 5.15: ViM model driving autonomously in EyeSim.

Of the original 10,000 images, approximately 3,000 were collected while driving in the reverse direction. Despite this limited reverse data, the model generalised sufficiently from the forward-driving data to complete the reverse route without error.

Other models trained on the same dataset initially produced less satisfactory results. Most failed to learn stable lane-following and turning behaviours, and those that partially succeeded exhibited unstable driving, often failing to complete a full lap. To address this, additional forward-direction data was collected, increasing the dataset size to approximately 100,000 samples. In addition, the steering and speed data were normalised to values between -1 and 1 for steering and 0 to 1 for speed (as no reversing data were required at this stage). Normalisation ensured that the larger speed values did not dominate the steering values during backpropagation, as the calculated error for speed is typically greater than that for steering. Finally, the loss function was changed from mean squared error (MSE) to mean absolute error (MAE), further preventing the speed component from dominating during backpropagation. Although the retrained models demonstrated good training and validation performance, testing in the simulator revealed that they primarily learned to drive

straight, with limited steering response.

This behaviour was traced to an imbalance between lane-following and turning samples. The ratio of large steering-angle samples to zero-angle samples was calculated, and turning data was up sampled to match the lane-following distribution. Although this represents a relatively simple approach to addressing over fitting, the goal at this stage was only to confirm that the models could operate autonomously. After up sampling, the models successfully navigated the Carolo Cup track without intervention, demonstrating their ability to learn fundamental driving behaviours.

Note that the larger up sampled models were only able to complete the track in one direction. Additional data would be required for reliable bidirectional operation. However, since this experiment served primarily as a proof-of-concept, further data collection was deemed unnecessary at this stage.

After further data collection and augmented to allow for an even sampling of turning and straight data, the key hyper parameters were tuned and are shown in tables 5.1 and 5.2 any parameters not mentioned are retained as the defaults from the original models. For the EyeSim models the channel size is 3 and a stepped learning rate scheduler was used.

Parameter	**ViT**	**DeiT**	**Swin**
Image Size	(66, 200)	(66, 200)	(64, 192)
Patch Size	(8, 8)	(8, 8)	(4, 4)
Features extracted	1000	1000	200
Embedding dimensions	512	768	96
Depth	4	12	[2, 6, 2]
Heads	16	16	[6, 12, 24]
Drop rate	0.1	0.1	0.05
Window size	-	-	4
MLP ratio	-	4.0	4.0
MLP dimensions	1024	-	-
Dim head	64	-	-
Learning rate	1e-4	1e-4	1e-4
Scheduler factor	0.7	0.7	0.7
Scheduler step size	5	5	5

TABLE 5.1: Model parameters for ViT, DeiT, and Swin for running on EyeSim simulator.

Parameter	ViM	VMamba
Image Size	(66, 200)	(66, 200)
Patch Size	(4, 4)	(4, 4)
Features extracted	1000	1000
Embedding dimensions	24	[96, 192, 384, 768]
Depth	24	[2, 2, 9, 2]
state space size	24	16
Drop rate	0.0	0.0
SSM activation function	-	"gelu"
Learning rate	1e-3	1e-3
Scheduler factor	0.1	0.1
Scheduler step size	10	10

TABLE 5.2: Model parameters for ViM and VMamba for running on EyeSim simulator.

5.4.2 Camera improvements

During the data collection stage, several updates were made to the camera setup to improve the quality and reliability of the collected data. The first modification involved adjusting the orientation of the front-facing camera to be more aligned with the centre line of the road. Following this adjustment, preliminary models were trained to validate the new configuration. However, it was observed that the vehicle consistently deviated to one side of the lane, rendering previously collected data incompatible with the new perspective. Although this setback temporarily delayed progress, it provided an opportunity to re-evaluate the data collection strategy. In particular, it highlighted the importance of capturing consistent and representative driving behaviour.

This revision also led to improvements in manual driving behaviour during data collection. More emphasis was placed on maintaining a stable trajectory within the lane and avoiding crossing the white line separating the main roadway from the road shoulder. This contributed to a cleaner and more consistent dataset for training.

Additionally, a noticeable latency was identified between the time an image was captured and when the corresponding metadata for speed and steering were recorded. After adjusting the camera and system settings, this latency was reduced from *0.436 seconds* to *0.123 seconds*. Since the training data had already been collected over several months prior to this improvement, the training pipeline was modified to shift the image stream by six frames, aligning

each image at time t with the control metadata recorded at time $t - \Delta t$, where $\Delta t = 0.313$ seconds (approximately six frames at 20 Hz). This adjustment restored temporal consistency between the visual inputs and control signals, enabling the models to perform stably at 20 Hz, compared to the original 6 Hz before camera optimisation, without noticeable disruptions in driving behaviour.

5.4.3 Training and running process

Efficient use of available computing resources was essential for model training, as training time varied substantially depending on architecture and model size. Two primary machines were used throughout this process: a laptop running Ubuntu 24.04 with an NVIDIA RTX4070 GPU, and a desktop workstation running Ubuntu 20.04 with an RTX3070Ti. Although the two GPUs offer broadly comparable performance, the 4070 system supported larger batch sizes during training. This advantage is attributed to the newer Ada Lovelace architecture of the 4070, which provides improved memory management and higher throughput compared to the Ampere-based 3070Ti. However, all models were successfully trained on both systems, with the 3070 Ti requiring smaller batch sizes to accommodate its lower memory bandwidth.

For deployment, the autonomous shuttle buses are equipped with NVIDIA Orin modules, which integrate CPU and GPU cores on a single system-on-chip (SoC). These devices require specific CUDA and PyTorch driver combinations, and the JetPack 6.0 (formerly JetPack26) environment used onboard was not directly compatible with the trained model configurations. To address this, Docker containers were created with matched CUDA, Torch, and driver versions to ensure runtime compatibility. While containerisation should not affect inference speed or model performance, it introduces additional layers of abstraction that can influence latency in subtle ways.

Following the completion of the main experiments, a post-hoc test was conducted to assess native deployment performance. The trained models were exported to the ONNX format and subsequently converted into TensorRT (TRT) engines optimised for the Orin hardware. These TRT-converted models ran successfully in native mode, achieving improved inference times. This suggests that future work could explore direct deployment using TRT-optimised models, potentially enabling the use of higher frame-rate or colour camera inputs for more responsive autonomous driving.

5.4.4 Image Shifting and Data Blurring

As discussed in Chapter 2, part of the data collection process involved removing undesirable or erroneous driving behaviours. While this cleaning process ensured that the models were trained on correct driving data, it also removed many examples of recovery behaviour, such as those occurring when the vehicle drifted toward the lane edge. As a result, the models initially struggled to learn how to recover from off-centre positions. To address this, the image data were randomly shifted horizontally during training, with the steering values adjusted accordingly to reflect the new viewpoint. After introducing this data shifting, the models were able to perform recovery actions more effectively.

It is also common practice to apply data augmentation during training to improve the robustness of the model. However, since the focus of this thesis is on model performance rather than the effects of augmentation, only limited augmentation techniques were employed. Due to camera performance and network latency, some recorded frames exhibited motion blur, while overexposure from lighting changes occasionally affected camera focus. To account for these real-world imperfections, additional blurred images were synthetically generated and included in the training dataset.

5.5 Results

The EyeSim simulation testing verified that all models were capable of learning and performing basic navigation tasks. Following successful validation, the models were adapted for deployment on the autonomous shuttle bus using the cleaned real-world dataset. The selected training parameters for each model are presented in Tables 5.3–8.1. The input image resolution for all models was set to 180×400 pixels (height × width). All sequence-based models employed the ReduceLROnPlateau learning rate scheduler to promote fine-grain training as local minima are reached.

Parameter	ViT	DeiT	Swin
Patch Size	(9,20)	(9,20)	(9,20)
Features extracted	1024	1000	1024
Embedding dimensions	768	768	240
Depth	6	6	[2, 6, 2]
Heads	12	12	[6, 12, 24]
Window size	-	-	5
MLP ratio	-	3.0	4.0
MLP dimensions	2048	-	-
Dim head	64	-	-
Drop rate	0.1	0.05	0.05
Learning rate	1e-4	1e-4	1e-4
Weight decay	0.05	0.05	0.05
Scheduler factor	0.7	0.8	0.01
Scheduler patience	5	3	5

TABLE 5.3: Model parameters for ViT, DeiT, and Swin for running on nUWAy shuttle bus.

Parameter	ViM	VMamba
Patch Size	(9, 20)	(9,20)
Features extracted	1024	1024
Embedding dimensions	512	[256, 512]
Depth	8	[3, 4]
state space size	16	32
SSM activation function	-	"gelu"

TABLE 5.4: Model parameters for ViM and VMamba for running on nUWAy shuttle bus.

A comparison of model size, inference time (on the NVIDIA Orin platform), and total parameter count is presented in Table 5.5. The CNN-based models were converted to the optimised TensorRT file format, providing measurable speed gains (shown in brackets) on

NVIDIA hardware. For this study, the sequential models were not converted, as their inference times were already sufficiently low for real-time operation. However if higher levels of performance are needed they too can be converted at a future date. Given the 20 Hz frame rate of the onboard camera, all models should achieve inference rates appropriate for real-time control.

Model	**File size (MB)**	**Inference (sec)**	**Parameters (10^6)**
PilotNet	0.003	0.023 (0.003)	4.26
EfficientNet	0.017	0.356 (0.017)	20.61 (20.46)
VMamba	281	0.030	23.46
ViM	191	0.028	15.90
ViT	427	0.022	35.62
Swin	539	0.046	44.95
DeiT	481	0.025	40.07

TABLE 5.5: Model file size, inference time, and parameter comparison. Inference time in brackets indicates post-TensorRT conversion; parameters in brackets indicate trainable subset.

Table 5.6 summarises the training and validation loss in three behavioural tasks: lane following, pull-in, and reversing. Swin demonstrated the shortest training times, even compared to SSM, attributed to its efficient windowed attention mechanism. However, this advantage may diminish when scaling to higher resolutions or colour image inputs, where computational complexity increases rapidly.

	Training Time (min)			**Training Loss**			**Validation Loss**		
Model	**Lane**	**Pull-in**	**Reverse**	**Lane**	**Pull-in**	**Reverse**	**Lane**	**Pull-in**	**Reverse**
PilotNet	–	–	–	0.0580	0.0703	0.0675	0.0557	0.0692	0.0665
EfficientNet	–	–	–	0.0133	0.0096	0.0141	0.0167	0.0122	0.01659
VMamba	88.38	26.98	12.68	0.0009	**0.0005**	**0.0002**	0.0009	**0.0005**	**0.0002**
ViM	94.98	27.57	13.44	0.0009	0.0005	0.0003	0.0009	**0.0005**	0.0003
ViT	123.24	30	15.5	0.0043	0.0045	0.0021	0.0043	0.0045	0.0022
Swin	**50.58**	**11.64**	**8.58**	**0.0005**	**0.0005**	0.0003	**0.0006**	**0.0005**	0.0003
DeiT	202.8	49.2	25.8	0.0090	0.0101	0.0047	0.0091	0.0104	0.0047

TABLE 5.6: Training duration and corresponding loss values for each model across lane following, pull-in, and reverse tasks.

The Speed and Accuracy values from the training are presented in Table 5.7. During the training of the VMamba model, the output values were compared with the ground truth to determine whether they fell within 5%. Both the training and validation results were consistently close throughout the training cycle, indicating stable and reliable model convergence. After this, the validation accuracy threshold was relaxed to within 10% of the ground truth. This adjustment served as an additional check to assess whether the models exhibited greater output variance. When training and validation losses remain close in value, it can be assumed that the corresponding accuracy metrics are also closely aligned.

From the table, it is evident that the VMamba model achieves superior training and validation accuracy across all three model types and for both speeding and steering outputs. looking at the remaining model, ViT demonstrates a significantly larger difference between the accuracy thresholds 5% and 10%, suggesting a higher output variance and less consistent performance.

	Speed Accuracy (%)						**Steering Accuracy (%)**					
	Training			**Validation**			**Training**			**Validation**		
Model	**Lane**	**Pull-in**	**Reverse**	**Lane**	**Pull-in**	**Reverse**	**Lane**	**Pull-in**	**Reverse**	**Lane**	**Pull-in**	**Reverse**
VMamba	**91.67**	**97.32**	**98.75**	**98.20**	**99.87**	**99.86**	**89.31**	**92.85**	**99.83**	**98.31**	**99.29**	**99.98**
ViM	88.08	94.69	96.55	95.94	99.06	99.43	86.68	89.05	99.64	96.89	97.37	99.95
ViT	75.57	76.91	81.64	88.46	88.21	93.84	75.86	71.41	97.82	91.68	84.64	99.71
Swin	89.85	91.54	93.59	96.7	98.43	99.18	87.44	85.39	99.77	97.01	95.89	99.96
DeiT	75.81	77.20	83.36	90.62	91.34	95.95	75.48	69.86	96.66	91.63	84.65	99.67

TABLE 5.7: Training and validation accuracy for speed and steering across lane following, pull-in, and reverse tasks.

A further breakdown of each model's accuracy and error was conducted using a previously unseen dataset. The results are presented in Figures 5.16 to 5.21 and Tables 5.8 to 5.15. Histograms illustrate the distribution of prediction accuracy, where a score of 1.0 represents an exact match to ground-truth output and a score of 0.0 corresponds to an error greater than or equal to 100% of the full output range. The accompanying tables present the discrete cumulative distribution functions (CDFs) of the error, separated by speed and steering outputs, and further categorised by the lane following, pull-in, and reversing models.

For the lane following task, only marginal differences are observed between model outputs for both speed and steering predictions. As shown in Tables 5.9 and 5.10, VMamba achieves the highest performance for speed prediction, while Swin slightly outperforms the other models in steering prediction. In both cases, the performance differences are small, with

DeiT and ViT exhibiting the lowest overall performance. Notably, ViM consistently ranks as the second-best performing model for both outputs and exhibits the lowest standard deviation, suggesting more stable and consistent predictions. The CDF results further indicate that ViT and DeiT produce a wider spread of errors, implying reduced reliability compared to the other models.

Across all models, steering predictions appear more accurate than speed predictions. This discrepancy may arise from the manner in which speed modes are adjusted in practice, as different operational modes scale speed differently and are not explicitly represented within the training data.

Analysis of the reversing task shows that all models produce steering predictions predominantly within 10% of the ground-truth values, which aligns with expectations. However, speed predictions demonstrate a substantially larger spread compared to the other evaluated tasks. During reversing, speed control is typically the more critical output, and the increased variability is likely attributable to differences in visual context. Because the reversing model relies on the front-facing camera, it initially observes the area ahead of the parking bays; therefore, variations in bay length can make it difficult for the model to accurately determine the appropriate deceleration timing. This behaviour is supported by the CDF results, where speed exhibits the largest error distribution, with errors exceeding 50% occurring more frequently than in other model outputs.

The pull-in model displays the greatest variation in steering predictions, which is expected given the importance of steering control for smoothly entering a parallel parking bay. This variability may be attributed to inconsistencies in the training data regarding the initiation point of turning manoeuvres, which may differ across recorded datasets.

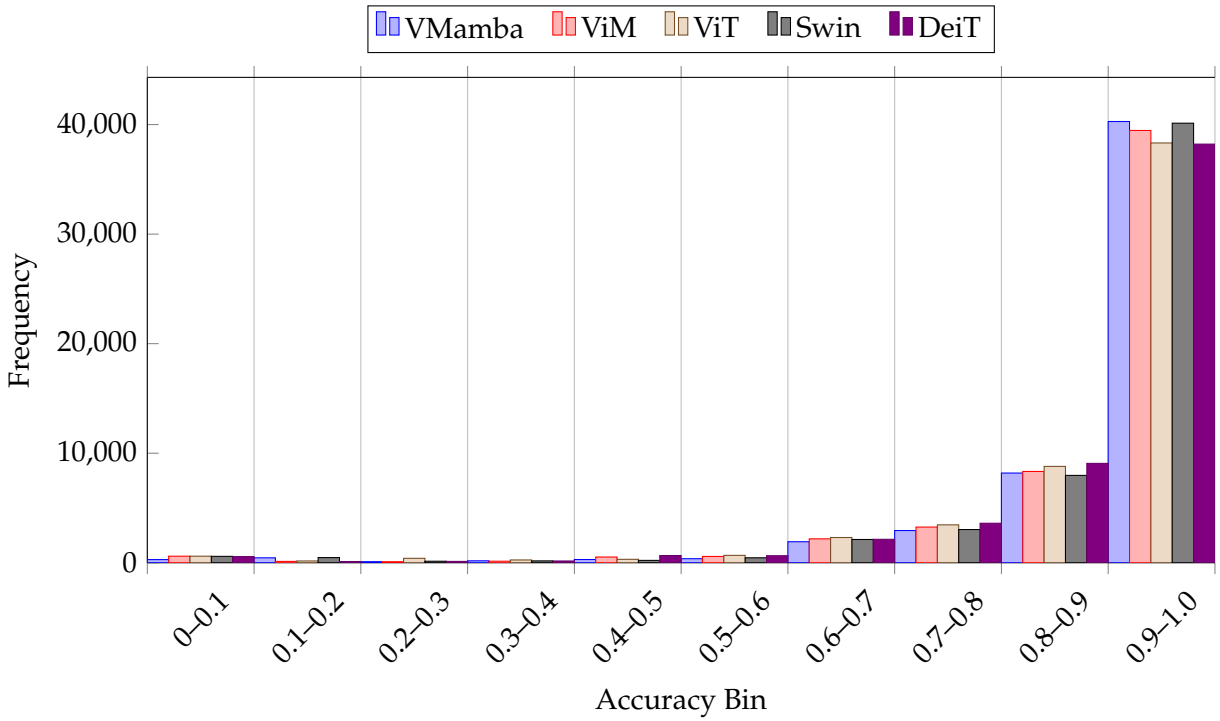


FIGURE 5.16: Lane following bin frequencies for five models - speed.

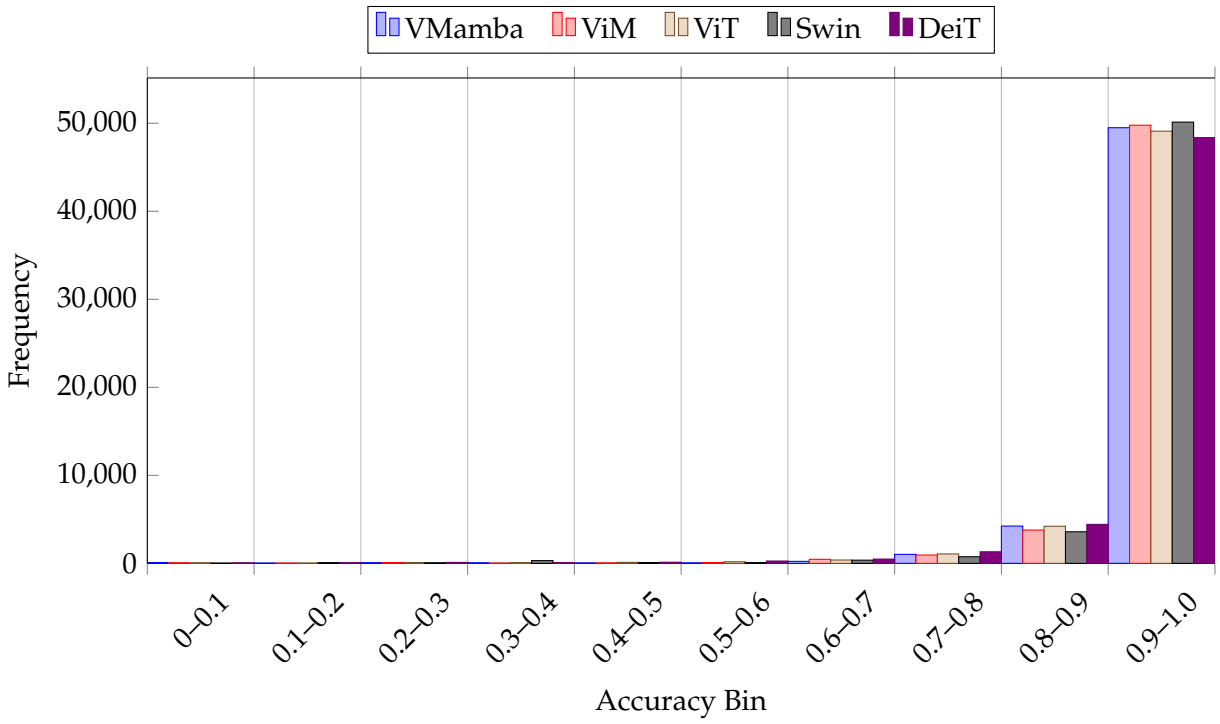


FIGURE 5.17: Lane following bin frequencies for five models - steering.

Speed Error (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
VMamba	34.34	55.85	71.11	80.44	85.80	89.23	92.67	93.89	94.40	4.90
ViM	34.40	53.89	68.91	78.52	84.47	88.16	90.34	93.64	94.72	4.49
ViT	29.75	50.48	66.24	76.46	82.96	86.75	90.60	92.63	93.97	5.38
Swin	35.34	54.97	69.77	79.16	85.14	88.29	90.37	93.68	94.50	4.83
DeiT	28.24	50.69	67.79	77.29	83.89	87.78	90.17	92.98	94.07	5.00
Steering Error (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
VMamba	54.67	82.18	91.61	95.36	97.80	98.53	98.98	99.22	99.29	0.63
ViM	60.06	83.54	91.44	94.84	96.44	97.71	98.91	99.20	99.38	0.51
ViT	52.93	81.27	89.45	93.95	96.02	97.26	98.05	98.50	98.85	0.91
Swin	60.57	84.37	92.09	95.27	96.75	97.44	97.86	98.11	98.24	1.69
DeiT	48.50	79.03	88.29	92.10	94.02	96.32	97.22	97.88	98.40	1.24

TABLE 5.8: CDF Lane following Error Comparison for Speed and Steering

Speed Error					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
VMamba	**13.69**	0.00	100.00	**8.55**	18.69
ViM	13.74	0.00	100.00	8.75	17.59
ViT	15.08	0.00	100.00	9.83	18.71
Swin	14.01	0.00	**99.96**	8.66	19.00
DeiT	14.59	0.00	100.00	9.78	**17.58**

TABLE 5.9: Summary of Lane Following Model Error for Speed and Steering

Steering Error					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
VMamba	6.30	0.00	100.00	4.09	8.31
ViM	**6.02**	0.00	100.00	3.40	**8.30**
ViT	6.93	0.00	100.00	4.48	9.30
Swin	6.17	0.00	**96.04**	**3.22**	9.80
DeiT	7.80	0.00	100.00	5.14	10.56

TABLE 5.10: Summary of Lane Following Model Error for Speed and Steering

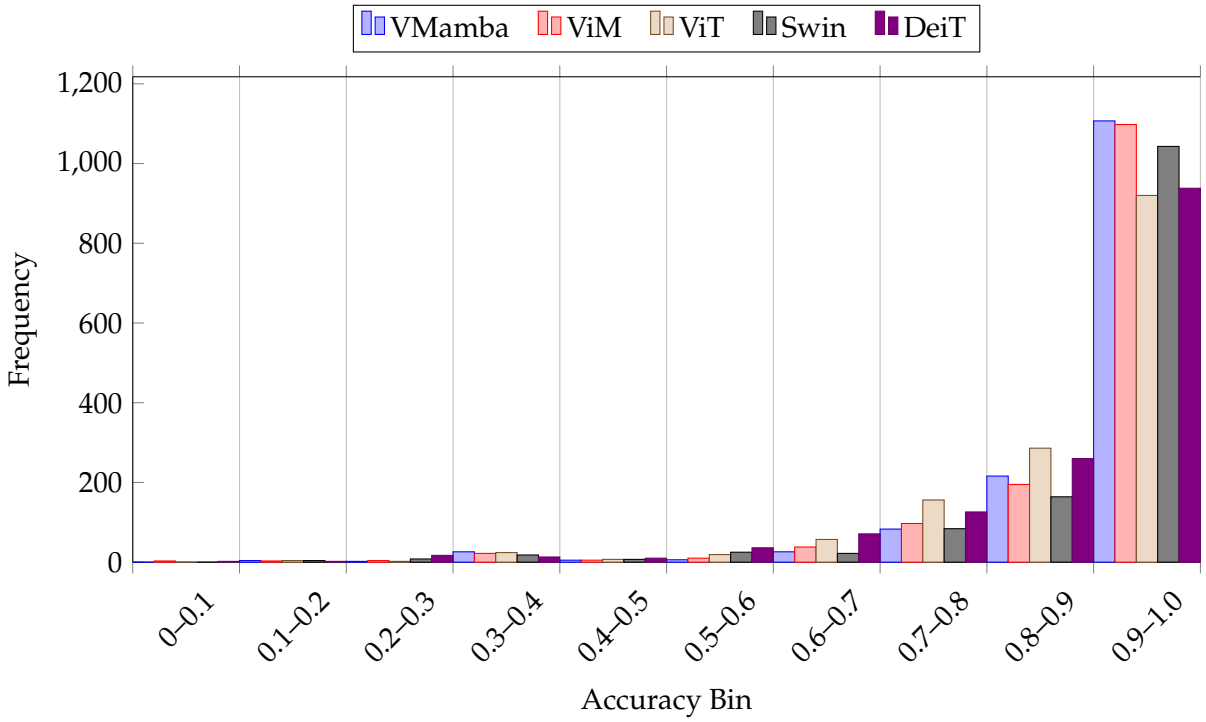


FIGURE 5.18: Pullin bin frequencies for five models - speed.

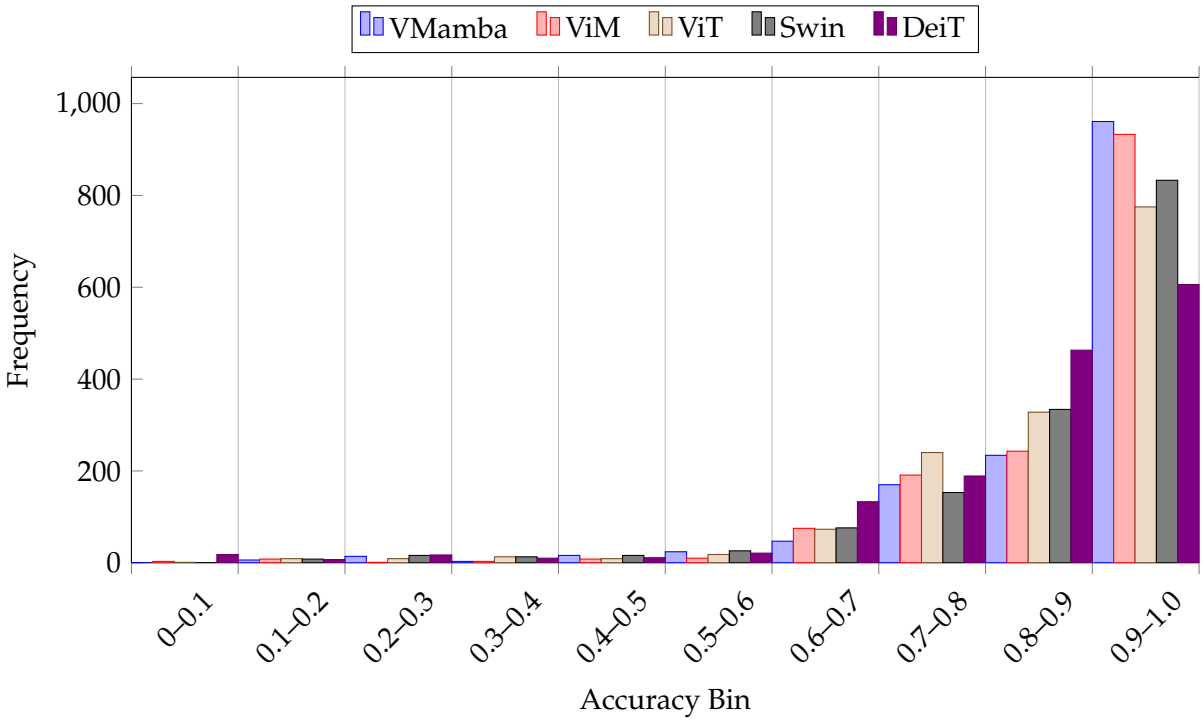


FIGURE 5.19: Pullin bin frequencies for five models - steering.

Speed Accuracy (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
VMamba	57.08	75.05	84.00	89.69	92.95	95.32	96.34	97.08	97.49	2.51
ViM	50.17	74.44	83.05	87.66	91.73	94.24	95.59	96.81	97.36	2.51
ViT	40.27	62.37	73.56	81.76	88.61	92.34	94.98	96.20	97.02	2.51
Swin	48.47	70.71	83.86	88.61	92.07	94.31	95.25	95.80	97.08	2.51
DeiT	36.41	63.59	74.17	81.22	96.31	89.76	92.20	94.58	96.34	2.98
Steering Accuracy (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
VMamba	48.61	65.15	74.64	81.02	88.54	92.54	94.71	95.73	96.34	2.64
ViM	41.36	63.25	73.56	79.80	86.71	92.75	96.27	97.69	98.37	1.56
ViT	30.71	52.20	64.88	74.85	83.53	91.05	94.78	95.93	96.81	2.85
Swin	32.68	56.41	70.24	78.85	84.81	89.49	93.42	94.64	95.05	3.66
DeiT	20.75	41.08	58.85	72.47	80.61	85.29	92.14	94.31	94.98	4.27

TABLE 5.11: CDF Pullin Accuracy Comparison for Speed and Steering

Speed Accuracy					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
VMamba	**8.14**	0.00	89.68	**3.65**	**12.24**
ViM	9.22	0.01	99.30	4.99	12.92
ViT	11.38	0.01	**89.60**	6.71	13.00
Swin	9.31	0.00	89.68	5.34	12.54
DeiT	12.31	0.00	90.44	7.21	14.20
Steering Accuracy					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
VMamba	**10.84**	0.00	87.30	**5.28**	13.95
ViM	11.26	0.00	97.45	7.02	**12.64**
ViT	13.79	0.01	94.55	9.29	13.96
Swin	13.25	0.01	**85.11**	8.46	14.80
DeiT	16.70	0.06	100.00	12.04	16.76

TABLE 5.12: Summary of Pullin Model Accuracy for Speed and Steering

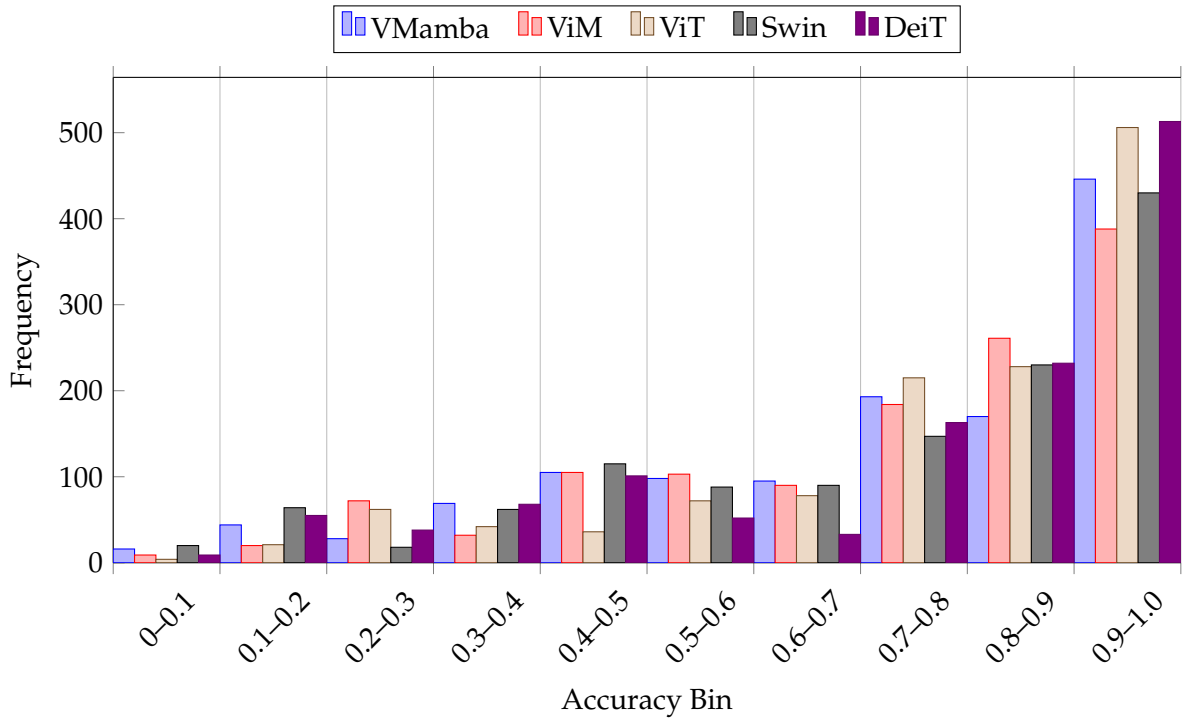


FIGURE 5.20: Reverse bin frequencies for five models - speed.

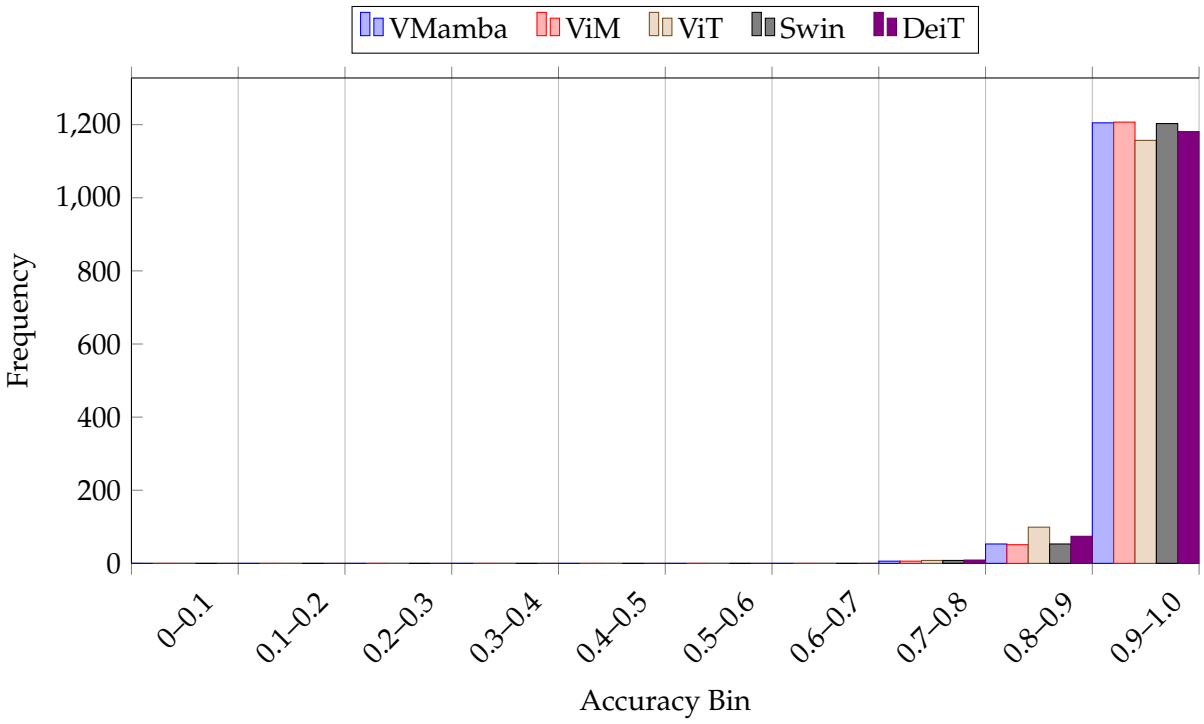


FIGURE 5.21: Reverse bin frequencies for five models - steering.

Speed Accuracy (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
VMamba	24.53	35.28	40.43	48.73	58.62	64.00	66.85	71.52	75.79	20.73
ViM	22.63	30.70	42.41	51.34	60.52	65.90	68.20	73.02	77.77	18.83
ViT	26.42	40.03	48.02	58.07	68.51	75.08	78.24	81.25	83.78	13.05
Swin	24.68	34.02	41.93	52.22	60.28	63.84	67.09	70.97	73.18	22.07
DeiT	20.41	40.59	50.00	58.94	68.28	71.84	73.18	74.45	76.34	21.44
Steering Accuracy (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
VMamba	83.78	95.33	98.18	99.53	99.76	100.00	100.00	100.00	100.00	0.00
ViM	86.63	95.41	98.18	99.53	99.60	100.00	100.00	100.00	100.00	0.00
ViT	73.50	91.46	98.10	99.37	100.00	100.00	100.00	100.00	100.00	0.00
Swin	74.68	95.09	98.26	99.37	99.92	100.00	100.00	100.00	100.00	0.00
DeiT	75.63	93.43	97.78	99.29	99.60	100.00	100.00	100.00	100.00	0.00

TABLE 5.13: CDF Reverse Accuracy Comparison for Speed and Steering

Speed Accuracy					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
VMamba	26.99	0.04	97.64	20.81	24.63
ViM	26.13	0.03	92.81	19.04	23.67
ViT	**22.14**	0.00	**91.71**	16.20	**22.48**
Swin	27.48	0.02	98.70	18.61	25.82
DeiT	24.90	0.00	98.71	**14.98**	25.13

TABLE 5.14: Summary of Reverse Model Accuracy for Speed and Steering

Steering Accuracy					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
VMamba	**2.93**	0.01	28.35	2.14	3.51
ViM	3.00	0.00	28.27	**1.88**	**3.44**
ViT	4.10	0.01	**22.22**	3.00	3.80
Swin	3.58	0.01	25.84	2.15	3.60
DeiT	3.98	0.00	28.37	3.16	3.63

TABLE 5.15: Summary of Reverse Model Accuracy for Speed and Steering

Each model was evaluated on autonomy time, disengagements, manual interventions, and deviation from the optimal path. Qualitative assessments such as speed control and steering smoothness were also performed.

Several driving scenarios were not represented in the original dataset and, consequently, not learned by the models. Four representative untrained events are shown in Figure 5.22. These include pedestrian crossings, passing vehicles at corners, road obstructions from parked vehicles, and overtaking scenarios. Such events significantly affect the measured autonomy and intervention rates; therefore, adjustments were made in post-analysis to account for their occurrence during test runs.

(A) Pedestrian crossing

(B) Passing vehicle on corner

(C) Vehicle parked on road

(D) Vehicle overtaking

FIGURE 5.22: Scenarios which the models are currently not trained to handle.

5.5.1 Saliency Maps and Ground Truth Comparison

Before deploying the model on the road, a final evaluation was performed using unseen test data. The model predictions were compared against the ground truth values, which were derived from manual outputs during data collection. In addition, saliency maps were generated to visualise the regions of the input image that most strongly influenced the model's predictions. Figures 5.23–5.29 show the linear and rotational saliency maps as well as the ground truth comparison for a variety of common scenarios encountered by the vehicle. These maps were particularly useful at key landmarks where the lane-following behaviour changed or where the model's output appeared inconsistent with the expected driving behaviour.

During model training, sky regions were occasionally observed to influence network inference results, as illustrated in Figure 5.29b. To mitigate this issue, input images were cropped to reduce the sky portion. However, this approach is not always feasible, particularly in dynamic driving environments. Future work could explore the use of masking to reduce the influence of irrelevant visual elements such as the sky.

In the ground truth comparison images, the green line represents the ground truth control output, while the purple line indicates the prediction of the model. The two lines are spatially separated to improve visual clarity. The length of each line corresponds to the desired vehicle speed, while the angle reflects the intended steering direction, being perpendicular to the lower image boundary when no steering is required and tilting in the direction of the intended turn when steering corrections are necessary.

(A) Carpark Saliency

(B) Carpark Ground Truth (green) vs Model Output (purple)

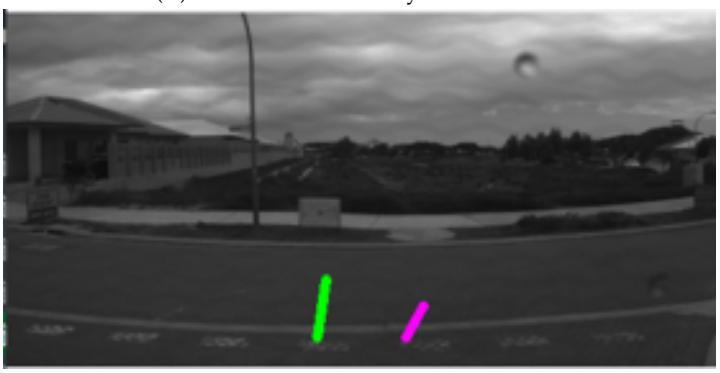
FIGURE 5.23: Ground truth and saliency map of carpark image.

(A) Lane Follow Saliency - Bending Road

(B) Lane Follow Ground Truth (green) vs Model Output (purple) - Bending Road

FIGURE 5.24: Ground truth and saliency map of lane following image - bending road.

(A) Lane Follow Saliency - Intersection



(B) Lane Follow Ground Truth (green) vs Model Output (purple) - Intersection

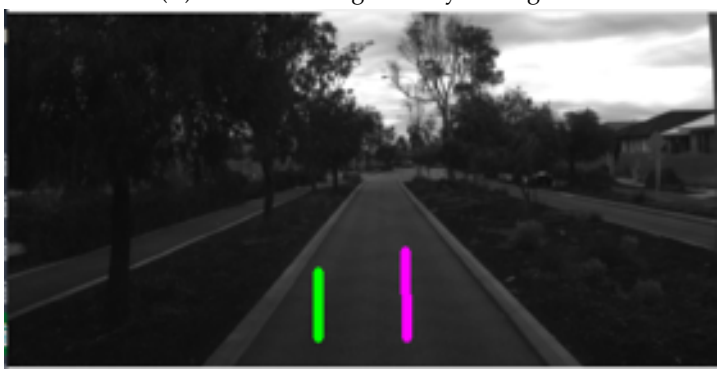
FIGURE 5.25: Ground truth and saliency map of lane following image - Intersection.

(A) Lane Following Saliency - Round About

(B) Lane Following Ground Truth (green) vs Model Output (purple) - Round About

FIGURE 5.26: Ground truth and saliency map of lane following image - Round About.

(A) Lane Following Saliency - Straight



(B) Lane Following Ground Truth (green) vs Model Output (purple) - Straight

FIGURE 5.27: Ground truth and saliency map of lane following image - Straight.

(A) Pull in Saliency

(B) Pull in Ground Truth (green) vs Model Output (purple)

FIGURE 5.28: Ground truth and saliency map of pull in image - parallel parking.

(A) Lane Following Saliency - Dual Lane

(B) Lane Following Saliency - Partial Obstruction

FIGURE 5.29: Additional saliency maps for common scenarios.

5.5.2 Driving Performance - Overview

The driving performance of each model is summarised in the following three tables. Table 5.16 presents the raw results.

Table 5.17 shows the results after removing periods when the vehicle speed was approximately zero, which usually corresponded to waiting while parked or in traffic. Such instances disproportionately affect the autonomy percentage depending on traffic conditions and are therefore excluded.

Finally, Table 5.18 shows the adjusted performance results, where the disengagements and interventions occurring in untrained scenarios were removed. This adjustment ensures a fair comparison between models by normalising for differing exposure to scenarios outside their training domain.

Model	PilotNet	EfficientNet	ViT	Swin	DeiT	ViM	VMamba
Drive time (min)	24.71 ± 1.01	24.67 ± 1.77	24.75 ± 3.18	26.26 ± 2.54	21.68 ± 2.02	23.14 ± 2.19	24.03 ± 2.23
Autonomy (%)	82.16 ± 4.87	89.45 ± 3.18	80.94 ± 4.70	81.09 ± 7.25	89.90 ± 4.84	86.37 ± 9.70	84.76 ± 5.40
Disengagements	6.33 ± 1.52	3.67 ± 3.06	3.00 ± 3.00	3.33 ± 1.15	2.00 ± 1.00	3.67 ± 2.31	1.00 ± 0.00
Interventions	18.67 ± 6.50	11.33 ± 3.06	17.33 ± 3.21	13.33 ± 1.53	9.33 ± 2.08	12.67 ± 7.77	14.33 ± 4.16

TABLE 5.16: Raw results for Eglinton Driving averaged across all drives (mean $\pm$ standard deviation).

Model	PilotNet	EfficientNet	ViT	Swin	DeiT	ViM	VMamba
Drive time (min)	20.73 ± 0.46	21.35 ± 1.80	19.84 ± 0.71	20.89 ± 0.32	18.80 ± 1.40	19.32 ± 0.88	19.40 ± 1.76
Autonomy (%)	89.02 ± 3.66	91.98 ± 2.42	87.23 ± 4.72	91.01 ± 0.23	93.53 ± 2.65	93.17 ± 4.15	90.77 ± 3.62
Disengagements	6.33 ± 1.52	3.67 ± 3.06	3.00 ± 3.00	3.33 ± 1.15	2.00 ± 1.00	3.67 ± 2.31	1.00 ± 0.00
Interventions	18.67 ± 6.50	11.33 ± 3.06	17.33 ± 3.21	13.33 ± 1.53	9.33 ± 2.08	12.67 ± 7.77	14.33 ± 4.16

TABLE 5.17: Results for Eglinton Driving averaged across all drives, excluding stationary waiting periods (mean $\pm$ standard deviation).

Table 5.18 presents the final performance metrics for each model after correcting for duplicate counts (disengagements and interventions) and excluding untrained scenarios. Among the models tested, DeiT, ViM, and VMamba achieved the highest overall performance, with comparable results across autonomy, disengagements, and interventions. All three Transformer-based models outperformed the classical architectures (PilotNet and EfficientNet) across all key metrics.

DeiT achieved the highest average autonomy percentage at 95.87%, although it exhibited a higher standard deviation compared to VMamba, indicating less consistent performance across runs. VMamba, while marginally lower in mean autonomy, demonstrated greater stability and achieved the lowest number of disengagements, highlighting its superior reliability during driving. ViM also performed competitively, showing strong autonomy and low intervention rates. Overall, these results indicate that the SSM models, particularly VMamba, offer the most consistent and robust driving performance on suburban roads.

Model	PilotNet	EfficientNet	ViT	Swin	DeiT	ViM	VMamba
Drive time (min)	20.26 ± 0.53	20.89 ± 1.70	18.64 ± 0.98	20.12 ± 0.15	18.34 ± 1.43	$\mathbf{18.76 \pm 0.61}$	18.42 ± 1.46
Autonomy (%)	90.07 ± 2.76	93.93 ± 2.15	92.40 ± 2.00	94.51 ± 1.72	95.87 ± 3.21	95.67 ± 3.27	$\mathbf{95.77 \pm 2.73}$
Disengagements	6.00 ± 1.00	3.33 ± 2.52	2.00 ± 2.00	3.33 ± 1.15	2.00 ± 1.00	2.67 ± 2.52	$\mathbf{0.67 \pm 0.58}$
Interventions	11.33 ± 5.13	6.67 ± 3.06	6.67 ± 0.58	5.67 ± 2.89	4.33 ± 2.62	$\mathbf{3.33 \pm 1.15}$	5.33 ± 1.53

TABLE 5.18: Final performance metrics for models tested on suburban roads. Results are adjusted for duplicate counts (disengagements and interventions) and excluding untrained scenarios (mean $\pm$ standard deviation).

Interventions were categorised into four primary types and two secondary (minor) types:

1. Poor driving behaviour in lane-following mode requiring intervention.
2. Untrained driving scenario intervention (removed in adjusted results).
3. Acceptable driving behaviour but with degraded passenger comfort (e.g., weaving, driving close to parked vehicles).
4. Poor driving behaviour in parking, reversing, or pulling out, requiring manual intervention.

Of these, only Category 2 interventions are removed in the adjusted results, along with their associated time.

The two minor categories are:

- **Estops:** Vehicle disengaged due to low-level hardware triggers. Manual intervention is removed as duplicate data, but time penalties are retained.
- **Invalid interventions:** Rare occurrences, typically at the start of data collection before runs began. These were excluded from both counts and time metrics.

	Category 1		Category 2		Category 3		Category 4		Estops		Invalid	
Model	**Count**	**%**	**Count**	**%**	**Count**	**%**	**Count**	**%**	**Count**	**%**	**Count**	**%**
PilotNet	26	49.06	12	22.64	1	1.89	7	13.21	6	11.32	1	1.89
EfficientNet	11	35.48	10	32.26	0	0.00	8	25.81	1	3.23	1	3.23
ViT	12	26.09	17	36.96	2	4.35	6	13.04	6	13.04	3	6.52
Swin	5	13.51	14	37.84	3	8.11	10	27.03	3	8.11	2	5.41
DeiT	8	32.00	12	48.00	0	0.00	4	16.00	1	4.00	0	0.00
ViM	9	25.00	14	38.89	0	0.00	4	11.11	6	16.67	3	8.33
VMamba	11	29.73	15	40.54	1	2.70	4	10.81	4	10.81	2	5.41

TABLE 5.19: Breakdown of manual interventions by category. Percentages represent the proportion of interventions per model.

From Table 5.19, the breakdown of manual interventions highlights distinct performance patterns across model architectures. The key categories of interest are 1, 3, and 4, which correspond to the most critical driving errors. The CNN-based models (PilotNet and EfficientNet), along with the Transformer models (ViT and Swin), show the highest number of interventions in these categories, close to or exceeding 20 interventions across all drives, indicating lower robustness in complex driving scenarios.

Among these, Swin demonstrates comparatively better on-road stability, with a large proportion of its interventions attributed to poor parking performance rather than general driving errors. In contrast, the remaining models—DeiT, ViM, and VMamba—show more balanced performance across categories. DeiT and ViM achieve similar results, while VMamba exhibits marginally higher overall intervention counts. However, all three models perform comparably when it comes to parking-related interventions. A more detailed breakdown of the parking results is provided in Section 4.6.4.

The data collected for each model is presented in the following sections as a series of maps. The first section shows the average performance across all drives for each model, with the exception of interventions, where all instances are displayed to highlight specific areas where the model encounters difficulties.

5.5.2.1 Manual Driving

Each model is trained on data collected from manual driving, which serves as the gold standard for how the models should learn to drive. To provide context to the following results, an optimal manually driven trajectory is included as a benchmark. Figure 5.30a shows the complete driving path; however, for evaluation purposes, the results are reported only from

the first roundabout to the parking to minimise the effects of congestion on the main road during the testing phase. The corresponding speed profile (Figure 5.30b) is slightly lower than that observed in the training data, as the focus during this run was to generate an ideal reference trajectory. The steering behaviour (Figure 5.30c) and the derived smoothness metric (Figure 5.30d) provide additional measures of driving quality. In particular, smoothness is used as a proxy for ride comfort, as rapid steering changes result in a less comfortable driving experience.

(A) Path

(B) Speed

FIGURE 5.30: Reference measurements from the manually driven optimal route (part 1). The plots show (a) the complete driving path and (b) the corresponding speed profile.

(C) Steering

(D) Smoothness

FIGURE 5.30: Reference measurements from the manually driven optimal route (part 2). The plots show (c) steering angles and (d) the smoothness metric.

5.5.2.2 PilotNet

The PilotNet model demonstrates relatively strong performance, as indicated in Table 5.18. It maintains reasonable path accuracy and speed regulation, as shown in Figures 5.31b and 5.31c. The steering behaviour does not exhibit any major anomalies overall; however, the smoothness profile in Figure 5.31f reveals periods of instability that correspond to an increased frequency of interventions and disengagements prior to the third roundabout (Figure 5.31g). In addition, a substantial number of interventions occur within the parking area, suggesting difficulties in navigating typical parking environments. Additional interventions are also observed at the right-hand roundabout, indicating limited generalisation capability in more complex driving scenarios.

(A) Driving mode distribution.

(B) Deviation from the optimal path.

(C) Average speed.

(D) Average speed relative to optimal.

FIGURE 5.31: On-road performance of the PilotNet model, showing driving mode, path accuracy, average speed and average speed relative to optimal.

(E) Steering relative to optimal.

(F) Trajectory smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.31: Additional PilotNet performance metrics including steering relative to optimal, smoothness, and intervention counts.

5.5.2.3 EfficientNet

The EfficientNet model demonstrates modest improvements over PilotNet, which is expected given its more complex architecture. A key difference is its ability to successfully execute right turns at the right-hand roundabout; however, this behaviour was not observed consistently. In terms of path following, the model exhibits a greater deviation from the optimal trajectory, as shown in Figure 5.32b. This aligns with the unstable steering patterns illustrated in Figures 5.32e and 5.32f, which in turn correspond to the disengagements and interventions observed around the first three roundabouts (Figure 5.32g). Overall, while EfficientNet shows some capacity for more complex manoeuvres, its reduced stability and increased deviation indicate limitations in reliable driving.

(A) Driving mode distribution.

(B) Deviation from the optimal path.

(C) Average speed.

(D) Average speed relative to optimal.

(E) Steering relative to optimal.

FIGURE 5.32: On-road performance of the EfficientNet model showing driving mode, path deviation, average speed , average speed relative to optimal and average steering relative to optimal.

(F) Trajectory smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.32: Additional performance metrics for EfficientNet including smoothness and intervention counts.

5.5.2.4 ViT

The ViT model demonstrates fewer disengagements compared to the previous models. As with earlier approaches, most of the disengagements occur in narrower sections of the road where over steering is observed (Figures 5.33g and 5.33f). In addition, it was able to consistently move through the right hand turn round about without intervention. Manual interventions are concentrated around the parking lot and the unmarked dual-lane road, indicating that the model struggles to stay on the left-hand side in multilane unmarked contexts. Compared to earlier models, the ViT model maintains a more consistent speed profile, particularly in the higher-speed section following the third roundabout, which improves ride comfort by reducing abrupt acceleration changes (Figure 5.33c). There is also some correlation between reduced autonomy and a greater deviation from the optimal path when comparing Figures 5.33a and 5.33b, although this relationship does not hold in all cases.

(A) Driving mode distribution.

(B) Deviation from the optimal path.

(C) Average speed.

(D) Average speed relative to optimal.

(E) Steering relative to optimal.

FIGURE 5.33: On-road performance of the ViT model showing driving mode distribution, path deviation, average speed, average speed relative to optimal and steering relative to optimal.

(F) Trajectory smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.33: Additional ViT performance metrics including smoothness and intervention counts.

5.5.2.5 Swin

The Swin model demonstrates some improvement at the right-hand roundabout, but still exhibits reduced levels of autonomy when deviating further from the optimal path, as shown in Figures 5.34a and 5.34b. The resulting disengagements and interventions, illustrated in Figure 5.34g, correspond to areas of greater path deviation as well as steering instability observed in Figure 5.34f. The model shows fewer interventions in the parking lot and along the unmarked multilane road, indicating improved ability to maintain road position. However, the speed fluctuations, depicted in Figure 5.34c, suggest reduced comfort on the road due to abrupt acceleration changes.

(A) Driving mode distribution.

FIGURE 5.34: On-road performance of the Swin model showing driving mode distribution.

(B) Deviation from the optimal path.

(C) Average speed.

(D) Average speed relative to optimal.

(E) Steering relative to optimal.

(F) Trajectory smoothness.

FIGURE 5.34: Additional Swin performance metrics including path deviation, average speed, average speed relative to optimal, average steering relative to optimal and smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.34: Swin performance metrics including intervention counts.

5.5.2.6 DeiT

The DeiT model demonstrates fewer disengagements compared to previous models, as shown in figure 5.35g, although some steering instability is still observed in figure 5.35f. This reduction in interventions suggests that the model can correct poor steering quickly, improving overall driving performance, but may result in jerky steering. The speed profile in figure 5.35c indicates the model often drives at lower speeds, which may help compensate for steering deviations. Figure 5.35a shows that the model is capable of autonomous navigation around the roundabout, while Figure 5.35b illustrates that the vehicle deviates further from the optimal path when more space is available.

(A) Driving mode

(B) Distance from optimal

FIGURE 5.35: Break down of DeiT model average on road performance including driving mode and path deviation.

(C) Average speed

(D) Average speed relative to optimal

(E) Average steering relative to optimal

(F) Average smoothness

(G) All interventions (blue manual, red disengagement)

FIGURE 5.35: Break down of DeiT model average on road performance including average speed, average speed relative to optimal, average steering relative to optimal, smoothness and intervention count.

5.5.2.7 Vim

The ViM model demonstrates more consistent driving performance, with fewer interventions as shown in figure 5.36g and stable speed profiles in figure 5.36c. Although the model drifts further from the optimal path than some previous models, as indicated in figure 5.36b, these small but consistent deviations do not appear to substantially impact performance. Disengagements are more evenly distributed, suggesting that variations in the road environment are not the primary factor influencing the model's behaviour.

(A) Driving mode distribution.

(B) Deviation from the optimal path.

(C) Average speed.

(D) Average speed relative to optimal.

FIGURE 5.36: On-road performance of the ViM model showing driving mode, path deviation, average speed and average speed relative to optimal.

(E) Steering relative to optimal.

(F) Trajectory smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.36: ViM performance metrics including steering relative to optimal, smoothness, and intervention counts.

5.5.2.8 VMamba

The VMamba model demonstrates the lowest number of disengagements among all models, as shown in Figure 5.37g. However, it exhibits a higher frequency of manual interventions, primarily in the parking lot area. The smoothness of the steering, illustrated in Figure 5.37f, indicates stable control throughout the route, with a localised reduction in stability between the second and third roundabouts, corresponding to the majority of manual interventions. The speed profile, presented in Figure 5.37c, shows a consistent velocity, which contributes to smoother ride quality. In general, Figure 5.37a demonstrates that VMamba maintains a consistent driving behaviour throughout the route.

(A) Driving mode distribution.

(B) Deviation from the optimal path.

(C) Average speed.

(D) Average speed relative to optimal.

(E) Steering relative to optimal.

FIGURE 5.37: On-road performance of the VMamba model showing driving mode, path deviation, average speed, average speed relative to optimal and average steering relative to optimal.

(F) Trajectory smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.37: VMamba performance metrics including smoothness and intervention counts.

5.5.3 Driving Performance – Best Drive per Model

For each model, the best performing drive was selected for a detailed analysis of driving performance, highlighting the respective strengths and weaknesses. Performance was assessed in several key categories, including smoothness of the journey (speed stability and steering control), manual interventions, and disengagement. As shown in Table 5.20, the DeiT model achieved the highest overall performance, driving with 99.45% autonomy, zero disengagements, and only one manual intervention. In particular, this also resulted in a substantially shorter overall drive time, completing the route nearly a full minute faster than the next best performing models.

Model	PilotNet	EfficientNet	ViT	Swin	DeiT	ViM	VMamba
Drive time (min)	20.08	21.12	18.37	20.00	**17.06**	18.06	18.01
Time autonomy (%)	93.47	95.69	91.91	95.84	**99.45**	98.85	98.23
Disengagements	5	1	0	4	**0**	**0**	**0**
Interventions	7	6	7	4	**1**	**1**	4

TABLE 5.20: Comparison of best driving performance across models.

The following sections present the driving metrics mapped for each model's best run. Unlike the earlier analysis, all driving modes are included, and the metrics for speed, steering

smoothness, and path deviation are reported across all autonomous and manual modes. Ideally, the steering smoothness values remain low except during expected manoeuvres such as turning at intersections, moving through round abouts, or parking. Similarly, vehicle speed should vary gradually to enhance passenger comfort, while deviations from the optimal path should remain minimal outside of parking manoeuvres.

5.5.3.1 PilotNet

Figure 5.38a illustrates the driving mode during PilotNet's best-performing run, in which the vehicle maintained autonomous control for the majority of the route. Figures 5.38b and 5.38c demonstrate that the journey was generally smooth, both in terms of velocity and steering control. However, Figure 5.38e highlights several disengagements and manual interventions, which occur most frequently on the narrow road segment and also following the third roundabout during parking.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

FIGURE 5.38: Performance metrics of PilotNet during its best driving run, including driving mode and speed.

(C) Steering smoothness expressed as angular velocity (rad/s).

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.38: Additional performance metrics of PilotNet during its best driving run, including smoothness, distance from optimal path and interventions.

5.5.3.2 EfficientNet

The EfficientNet model demonstrates improved performance compared to PilotNet, with fewer interventions overall (Figure 5.39e). Most interventions occurred within the parking lot, while the single disengagement was recorded during a parking manoeuvre (Figure 5.39c). The speed and steering smoothness profiles (Figures 5.39b and 5.39c) indicate a consistent stable motion throughout most of the drive.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

(C) Steering smoothness expressed as angular velocity (rad/s).

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.39: Performance of EfficientNet during its best driving run, including driving mode, speed, steering smoothness, path deviation, and interventions.

5.5.3.3 ViT

The best performing ViT model shows variable speed profiles across several locations, as shown in Figure 5.40b, along with minor fluctuations in steering smoothness (Figure 5.40c). Although no full disengagements were recorded, several manual interventions occurred (Figure 5.40e). These interventions often coincide with sharp steering adjustments, particularly at pull-in points, where steering changes are expected, indicating that the model still struggles to maintain smooth control transitions in such regions.

As shown in Figure 5.40d, the vehicle occasionally deviates from the optimal path for extended periods, though these deviations typically occur in wider sections of the road. Consequently, the frequency of manual interventions remains relatively low, as these deviations do not immediately result in unsafe driving behaviour.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

(C) Steering smoothness expressed as angular velocity (rad/s).

FIGURE 5.40: Performance of ViT during its best driving run, including driving mode, speed and steering smoothness.

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.40: Performance of ViT during its best driving run, including path deviation, and interventions.

5.5.3.4 Swin

The results for the best-performing Swin model demonstrate strong performance in steering smoothness, as illustrated in Figure 5.41c. Some of the discrepancies in distance observed in Figure 5.41d correspond to the intervention points shown in Figure 5.41e, although they do not fully account for all disengagements. The variations in speed presented in Figure 5.41b suggest potential discomfort from a passenger perspective but similarly fail to explain the frequency of disengagement events.

Based on qualitative observations from ROSbag playback, the disengagements are primarily attributed to suboptimal approach angles when navigating narrow or twisting road sections. This indicates that while the Swin model maintains generally smooth control under standard driving conditions, its ability to generalise to complex spatial constraints remains limited.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

(C) Steering smoothness expressed as angular velocity (rad/s).

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.41: Performance of the Swin Transformer during its best driving run, including driving mode, speed, steering smoothness, path deviation, and interventions.

5.5.3.5 DeiT

The best-performing DeiT model exhibited a single disengagement at the third roundabout, as shown in Figure 5.42e. This event corresponds with a notable fluctuation in steering smoothness (Figure 5.42c) and a greater deviation from the optimal trajectory (Figure 5.42d). Despite further path deviations, the model maintained relatively consistent speed throughout the test (Figure 5.42b), demonstrating stable longitudinal control. The single intervention occurred at the right-hand roundabout, a location where several models also experienced difficulties, suggesting that this may represent a data generalisation limitation in the training environment rather than a model-specific failure.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

(C) Steering smoothness expressed as angular velocity (rad/s).

FIGURE 5.42: Performance of DeiT during its best driving run, including driving mode, speed, and steering smoothness.

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.42: Performance of DeiT during its best driving run, including path deviation and interventions.

5.5.3.6 ViM

The best-performing ViM model highlights the strengths of State Space Models (SSMs). Figure 5.43b shows smooth and consistent speed control, while Figure 5.43c demonstrates stable steering performance. These characteristics resulted in only two manual interventions (Figure 5.43e), both associated with the common right-hand roundabout and variations in the parking lot. Figure 5.43d shows larger deviations from the optimal path, which may be due to GPS signal fluctuations or indicate that alternative driving trajectories can achieve comparable performance to the defined optimal path.

(A) Driving mode throughout the best-performing route.

FIGURE 5.43: Performance of ViM during its best driving run, including driving mode.

(B) Vehicle speed profile (m/s).

(C) Steering smoothness expressed as angular velocity (rad/s).

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.43: Performance of ViM during its best driving run, including speed, steering smoothness, path deviation, and interventions.

5.5.3.7 VMamba

The best-performing VMamba run demonstrates stable and reliable control, achieving zero disengagements and four manual interventions (Figure 5.44e). These interventions occurred primarily within the carpark and at the roundabout, suggesting that the model could benefit from improved generalisation to complex or irregular environments.

As shown in Figures 5.44b and 5.44c, the VMamba model exhibits smooth and consistent control of both speed and steering throughout the drive. Minor deviations from the optimal

path are visible in Figure 5.44d, primarily in wider road segments where greater freedom in trajectory selection exists. Overall, VMamba demonstrates robust performance with only limited instances requiring human intervention.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

(C) Steering smoothness expressed as angular velocity (rad/s).

(D) Distance from optimal path during the route (m).

FIGURE 5.44: Performance of VMamba during its best driving run, including driving mode, speed, steering smoothness and path deviation.

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 5.44: Performance of VMamba during its best driving run, including interventions.

5.5.4 Parking, Reversing, and Exiting Parallel Bays

The models were further evaluated on their ability to perform complex manoeuvres, including parallel parking and exiting along roadways, as illustrated in Figure 5.45.

FIGURE 5.45: nUWAy2 shuttle bus performing a parking manoeuvre.

Table 5.22 summarises the success rates for both known and unknown parking bays, based on the data distribution presented in Chapter 2. Bays are considered *known* if they

contain at least five data points from prior data collection. The scoring criteria for each manoeuvre are presented in Table 5.21, and provide a standardised method to evaluate performance across models.

Behaviour	**Success Criteria**
Pull-in	(1) Correctly identify and fully enter the target parking bay; (2) Align final heading with the bay orientation; (3) Come to a complete stop without contacting curbs; (4) Complete the manoeuvre 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 or causing disruption (2 points).

TABLE 5.21: Closed-loop success criteria for each expert driving behaviour.

Model	PilotNet	EfficientNet	ViT	Swin	DeiT	ViM	VMamba
Pull-in Score	2.42 ± 1.31	2.00 ± 1.63	$\mathbf{3.2 \pm 0.86}$	2.2 ± 1.08	2.83 ± 1.34	3 ± 0.68	2.58 ± 1.24
Pull-in left (%)	83.33	87.5	73.33	93.33	66.67	85.71	66.67
Pull-in right (%)	16.67	12.5	26.67	6.67	33.33	14.29	33.33
Pull-in known bays (%)	66.67	75	66.67	66.67	58.33	57.14	75
Pull-in unknown bays (%)	33.33	25	33.33	33.33	41.67	42.86	25
Reverse Score	1.33 ± 0.89	$\mathbf{1.93 \pm 0.27}$	1.63 ± 0.62	1.71 ± 0.61	$\mathbf{1.92 \pm 0.28}$	1.85 ± 0.38	1.58 ± 0.67
Reverse left (%)	76.92	93.33	75	92.86	69.23	84.62	66.67
Reverse right (%)	23.08	6.67	25	7.14	30.77	15.38	33.33
Reverse known bays (%)	46.15	64.29	50	57.14	38.46	46.15	63.64
Reverse unknown bays (%)	53.85	35.71	50	42.86	61.54	53.85	36.36
Pull-out Score	3.50 ± 1.08	3.78 ± 0.43	3.33 ± 1.11	3.85 ± 0.38	3.69 ± 0.85	3.93 ± 0.27	$\mathbf{4.00 \pm 0.00}$
Pull-out left (%)	81.82	93.33	73.33	92.31	69.23	85.71	63.64
Pull-out right (%)	18.18	6.67	26.67	7.69	30.77	14.29	36.36
Pull-out known bays (%)	45.45	85.71	53.33	53.85	46.15	50.00	54.55
Pull-out unknown bays (%)	54.55	14.29	46.67	46.15	53.85	50.00	45.45

TABLE 5.22: Performance results for parking and reversing tasks across all models.

From the results in Table 5.22, the ViT model achieved the highest overall score for the

pull-in manoeuvre (3.20/4.00). However, the ViM model demonstrated nearly equivalent performance (3.00/4.00) while handling a greater proportion of unknown bays, suggesting a stronger generalisation capability. In reversing tasks, performance across models was comparable, with DeiT and EfficientNet achieving the best results (1.92/2.00 and 1.93/2.00, respectively), closely followed by ViM (1.85/2.00). For pull-out manoeuvres, all models performed consistently, though ViT achieved the lowest score (3.33/4.00), while VMamba achieved a perfect score (4.00/4.00). This outcome aligns with its strong on-road performance, as pull-out behaviour is an extension of the on road control. Overall, these results highlight the advantages of a modular behavioural framework, where distinct models can be leveraged for specialised driving tasks to achieve optimal system-level performance.

5.5.5 Lane Following – Road Features

To further evaluate the robustness of lane-following behaviour, each model was tested across fourteen key driving events that introduce additional environmental and road complexities. These locations are highlighted in Figure 5.46.

FIGURE 5.46: Landmarks and road features along the Eglinton drive route.

Table 5.23 summarises the performance of each model at these critical landmarks. Each driving feature was scored according to the behavioural success criteria outlined below, which quantify the quality of model control, stability, and adherence to safe driving behaviour.

- **Roundabouts** – maximum 3 points: 1 point for successfully entering, 1 point for avoiding curbs, and 1 point for exiting without intervention.

- **Snaking road** – maximum 2 points: -1 point for taking an excessively wide path (activating the slow node; see Chapter 8), and -2 points for an emergency stop (e-stop) or manual override.
- **Intersections** – maximum 2 points: -1 point for entering too wide or too shallow (activating the slow node), and -2 points for e-stop or manual override.
- **Dual-lane (unmarked)** – maximum 2 points: -1 point for partially crossing into the opposite lane, and -2 points for fully crossing or hitting a curb.
- **Dual-lane (marked)** – maximum 2 points: -1 point for partially crossing into the opposite lane, and -2 points for fully crossing the line, contacting a curb, or performing an e-stop when lanes narrow.
- **Carpark section** – maximum 5 points: -1 point per manual intervention, -1 point per lane crossing, and -2 points for any e-stop.

Table 5.23 presents the detailed results across all models and road features.

Model	PilotNet	EfficientNet	ViT	Swin	DeiT	ViM	VMamba
Roundabout 1 (R1)	2.00 ± 1.00	$\mathbf{3.00 \pm 0.00}$	$\mathbf{3.00 \pm 0.00}$	$\mathbf{3.00 \pm 0.00}$	$\mathbf{3.00 \pm 0.00}$	2.67 ± 0.58	$\mathbf{3.00 \pm 0.00}$
Snaking Road	1.33 ± 0.58	1.00 ± 1.00	1.33 ± 0.58	1.00 ± 0.00	1.00 ± 1.00	$\mathbf{1.67 \pm 0.58}$	$\mathbf{1.67 \pm 0.58}$
Roundabout 2 (R2)	$\mathbf{3.00 \pm 0.00}$	2.67 ± 0.58	2.33 ± 0.58	2.67 ± 0.58	$\mathbf{3.00 \pm 0.00}$	$\mathbf{3.00 \pm 0.00}$	$\mathbf{3.00 \pm 0.00}$
Intersection 1 (I1)	1.33 ± 0.58	$\mathbf{2.00 \pm 0.00}$	$\mathbf{2.00 \pm 0.00}$	1.67 ± 0.58	1.67 ± 0.58	$\mathbf{2.00 \pm 0.00}$	1.33 ± 0.58
Dual lane unmarked 1	1.67 ± 0.58	$\mathbf{2.00 \pm 0.00}$	0.33 ± 0.58	1.67 ± 0.58	1.33 ± 1.15	1.33 ± 1.15	1.67 ± 0.58
Intersection 2 (I2)	0.67 ± 0.58	0.33 ± 0.58	0.00 ± 0.00	$\mathbf{1.33 \pm 0.58}$	1.33 ± 1.15	$\mathbf{1.33 \pm 0.58}$	$\mathbf{1.33 \pm 0.58}$
Carpark	2.67 ± 1.53	2.67 ± 0.58	2.67 ± 1.15	$\mathbf{3.67 \pm 0.58}$	2.67 ± 0.58	3.67 ± 1.15	1.67 ± 0.58
Intersection 3 (I3)	0.33 ± 0.58	1.33 ± 0.58	$\mathbf{1.67 \pm 0.58}$	0.67 ± 1.15	$\mathbf{1.67 \pm 0.58}$	1.33 ± 1.15	$\mathbf{1.67 \pm 0.58}$
Dual lane unmarked 2	1.33 ± 0.58	$\mathbf{2.00 \pm 0.00}$	1.33 ± 1.15	$\mathbf{2.00 \pm 0.00}$	1.33 ± 0.58	1.67 ± 0.58	1.67 ± 0.58
Intersection 4 (I4)	1.67 ± 0.58	$\mathbf{2.00 \pm 0.00}$	$\mathbf{2.00 \pm 0.00}$	1.67 ± 0.58	$\mathbf{2.00 \pm 0.00}$	$\mathbf{2.00 \pm 0.00}$	$\mathbf{2.00 \pm 0.00}$
Round about right	1.33 ± 0.58	1.00 ± 1.00	$\mathbf{3.00 \pm 0.00}$	2.33 ± 0.58	1.33 ± 0.58	2.00 ± 0.00	2.33 ± 0.58
Dual lane marked	$\mathbf{2.00 \pm 0.00}$	1.00 ± 1.00	1.00 ± 1.00	1.33 ± 1.15	$\mathbf{2.00 \pm 0.00}$	$\mathbf{2.00 \pm 0.00}$	$\mathbf{2.00 \pm 0.00}$
Roundabout 3 (R3)	2.67 ± 0.58	2.67 ± 0.58	2.33 ± 0.58	2.33 ± 0.58	$\mathbf{3.00 \pm 0.00}$	2.67 ± 0.58	2.67 ± 0.58
Roundabout 4 (R4)	2.67 ± 0.58	2.67 ± 0.58	2.00 ± 0.00	2.33 ± 0.58	$\mathbf{3.00 \pm 0.00}$	2.33 ± 1.15	2.67 ± 0.58
Total Score (/36)	$\mathbf{24.67 \pm 0.59}$	$\mathbf{26.33 \pm 0.46}$	$\mathbf{25.00 \pm 0.44}$	$\mathbf{27.67 \pm 0.54}$	$\mathbf{28.33 \pm 0.44}$	$\mathbf{29.67 \pm 0.54}$	$\mathbf{28.67 \pm 0.41}$

TABLE 5.23: Landmark driving results with total performance scores across all events.

Reviewing the results in Table 5.23, it is evident that most models successfully navigated roundabouts, with DeiT achieving a perfect success rate across all instances. The most challenging manoeuvre was the snaking road, which required maintaining a smooth, central

trajectory to avoid disengagements. Here, the two pure SSM models (ViM and VMamba) achieved the highest scores of 1.67/2.00, demonstrating superior control stability.

Intersections 1 and 4 represented simpler scenarios involving single turns, where EfficientNet, ViT, and ViM each achieved perfect scores. In contrast, Intersections 2 and 3 required multiple turns, presenting greater difficulty; VMamba achieved the highest overall performance, though not a perfect score. Although DeiT also scored the highest at Section 2, its higher standard deviation indicates less consistent behaviour compared to VMamba.

For the unmarked dual-lane sections, EfficientNet achieved perfect scores for both instances, suggesting strong lane-centreing behaviour in the absence of markings. In the marked dual-lane scenario, PilotNet, DeiT, ViM, and VMamba each achieved perfect performance. The carpark environment, with its variable vehicle density, pedestrian movement, and lack of lane markings, proved most challenging. Among all models, Swin and ViM achieved the highest carpark scores (3.67/5.00); however, Swin demonstrated greater consistency with a lower standard deviation.

These results highlight that while all models exhibit strong baseline lane-following behaviour, specific scenarios expose unique weaknesses. Overall, the SSM-based models (ViM and VMamba), along with DeiT, outperform the classical CNN-based approaches. ViM achieved the highest total score (29.67/36) but with moderate variability ($\pm$0.54), whereas VMamba and DeiT demonstrated slightly lower overall scores yet greater consistency, reflected in their lower average standard deviations. This suggests that SSMs and hybrid attention state-space architectures not only generalise well to complex driving conditions, but also offer stable, predictable behaviour in diverse road contexts.

5.5.6 Lighting Impacts

During testing, it was observed that varying lighting conditions and time-of-day effects influenced the performance of the vision-based models. To further investigate this, an additional evaluation was conducted using the VMamba model during an early-morning drive, when lighting and shadows differed significantly from the standard test conditions. The results of this trial, along with the corresponding baseline performance under normal lighting, are presented in Table 5.24.

Metric	VMamba (morning)	VMamba (standard)
Drive time (min)	20.21	18.42 ± 1.46
Time autonomy (%)	91.82	95.77 ± 2.73
Disengagements	4	0.67 ± 0.58
Interventions	6	5.33 ± 1.53

TABLE 5.24: Performance comparison of the VMamba model under morning and standard lighting conditions.

Although this comparison represents a single case, similar patterns were observed across other models, where rapid transitions between bright sunlight and cloud cover degraded performance. As shown in Table 5.24, the VMamba model experienced a notable increase in disengagements and interventions, along with a reduction in autonomy time of approximately 4%. These findings highlight the sensitivity of purely vision-based models to illumination variability and underscore the potential need for either illumination-robust training strategies or the integration of complementary sensor modalities such as LiDAR or infrared imaging.

5.6 Discussion

From Table 5.6, it can be seen that sequence-based models achieved substantially lower training and validation losses using the same MAE metric. However, it should be noted that this metric can be influenced by batch size. The PilotNet and EfficientNet models were trained with batch sizes between 4 and 30, while the sequence models used larger batches ranging from 20 to 80. Nevertheless, with the exception of the DeiT and ViT models, most sequential models achieved losses an order of magnitude smaller, highlighting the strength of transformer and State-Space Model (SSM) architectures.

Examining the accuracy values in Table 5.7, the VMamba model consistently outperformed all other sequential models in both speed and steering accuracy during training, exceeding the next best model by approximately 2–3%. Interestingly, the ViT model exhibited higher variance between the 5% and 10% thresholds, suggesting variability in model behaviour. However, as shown in Table 5.18, ViT maintained stable performance in autonomy, disengagements, and interventions compared to other models.

Examination of the CDFs and histograms reveals that it is challenging to establish a direct relationship between these evaluation metrics and the final driving performance on the road.

The relative performance of the models across the lane following, pullin, and reversing tasks on the unseen data set would suggest that the model performance during lane following should be similar across architectures, that VMamba or ViM would dominate during pullin manoeuvres, and that Swin or ViT would achieve the strongest performance during reversing. However, these expectations are not fully reflected in real-world driving results.

One possible explanation is that the unseen data set was limited in size and may not adequately represent the range of scenarios encountered during a real-world operation. The driving results presented in Table 5.18, which incorporate all three behavioural modes, provide additional insight. For example, Swin exhibits a comparatively higher standard deviation across model outputs, which may contribute to the increase in the number of disengagements and interventions observed during road tests, despite achieving competitive mean and median performance metrics.

Furthermore, DeiT demonstrates on-road performance comparable to ViM and VMamba, despite consistently under performing in evaluations conducted on the unseen data set. This discrepancy suggests that the evaluation based on a dataset alone may not fully capture the characteristics that influence the robustness and driving reliability of the real-world. Although a larger and more diverse unseen data set would be required to draw definitive conclusions regarding the relationship between offline evaluation and on-road performance, these findings support the original hypothesis presented in the introduction: that evaluation based solely on test datasets or simulation does not necessarily correspond to real-world performance.

Section 4.6.1 presented the ground truth comparisons and saliency maps for each model. The majority of models demonstrated strong alignment with the ground truth, consistent with the high training accuracy results. For example, VMamba achieved 91.58% and 89.19% accuracy within a 5% threshold for speed and steering, respectively, on unseen data, implying that even when predictions were incorrect, they remained close to the true value. Given the inference interval of approximately 0.05 seconds, the system is capable of rapid correction. The saliency maps, however, revealed a tendency for the models to focus on irrelevant features such as the sky, particularly under bright conditions. A more robust masking strategy or the inclusion of an autoencoder could help the model learn to ignore non-salient regions. Additionally, since only grey scale images were used, switching to colour images may help distinguish between bright sky and road surfaces, which often appear similar under certain lighting conditions.

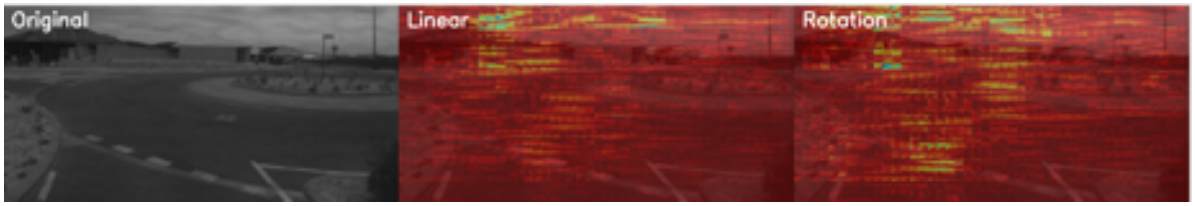
As shown in Table 5.18, all models were able to drive reliably on public roads. VMamba,

ViM, and DeiT achieved the best performance, with autonomy times of 95.77%, 95.67%, and 95.87% respectively. Although the Swin model also performed well, it exhibited a higher variance in disengagements and interventions, suggesting lower stability. In general, these four models outperformed the baseline CNN models, PilotNet and EfficientNet, while ViT's performance was closer to EfficientNet. Despite strong performance, further improvements are required to reduce interventions and increase reliability to ensure user comfort and readiness for deployment in fully autonomous systems.

Table 5.5 highlights the parameter breakdowns, which are critical considerations for deployment in resource-limited vehicles. Despite its small parameter count, PilotNet performed poorly compared to newer architectures. EfficientNet and VMamba had comparable parameter counts, while ViM had fewer parameters overall. As expected, Transformer-based models such as Swin and DeiT had significantly larger parameter sizes, approximately double that of EfficientNet and VMamba. These results position VMamba and ViM as the most promising candidates for deployment in real-world autonomous systems.

Table 5.19 presents the breakdown of interventions by category. Category 1 interventions (lane-following failures) are the most critical due to their direct impact on other road users, while Category 4 interventions (parking failures) are less severe. Swin exhibited a higher proportion of Category 4 interventions, suggesting weaker parking-related performance. In contrast, DeiT, ViM, and VMamba showed more Category 1 interventions, indicating a need to improve the reliability of the lane follow. Although all models achieved high levels of autonomy, the persistence of Category 2 interventions highlights that many real-world situations remain unaccounted for. To achieve SAE Level 4 autonomy, the models must be further trained or augmented with fallback mechanisms to handle unseen scenarios.

Sections 4.6.2 and 4.6.3 provided averaged and best-drive map metrics for each model. The results suggest that behaviours such as 'right turn at roundabouts' and 'roundabout straight' should be separated, as these remain challenging. Construction around the right-hand roundabout also increased difficulty due to environmental changes. Figure 5.47 illustrates this issue: a newly constructed house altered the scene, leading VMamba to misinterpret the scenario and fail to turn correctly. This underscores the need for behaviour-specific models to improve generalisation in changing environments.



(A) Saliency map with training data.

(B) Saliency map during inference

FIGURE 5.47: Saliency maps from VMamba at right hand round about showing the models failure to generalise.

Furthermore, performance in the car park was consistently weaker across models (Table 5.23) due to inconsistent data and high variability in parked vehicles. Future training using masked inputs and autoencoders may improve the consistency of features in these areas.

In a mixture-of-experts configuration, different models can be assigned to specific tasks. From Table 5.22, VMamba is the best choice for lane-following, ViT for parallel parking, and either EfficientNet or DeiT for reversing. Since EfficientNet has fewer parameters and faster inference, it is the preferred choice for this subsystem.

Lighting conditions and time of day also significantly affected performance. Morning drives produced reduced autonomy and increased disengagements, as shown in Table 5.24. This was attributed to limited morning training data and strong sunlight causing overexposure of the image. These findings highlight the importance of addressing variable lighting through data augmentation or multi modal sensing.

Finally, several practical limitations were identified that may have impacted the final results:

- Environmental changes, such as new construction, can reduce model generalisation, especially at key intersections.
- Increasing traffic density and new local developments affect driving behaviour.
- Mechanical issues (e.g., steering rack replacements and wheel alignment offsets) may have introduced inconsistencies in the steering output.

- The current data augmentation assumes that only lateral corrections are needed to improve driving behaviour; incorporating rotational augmentations may improve robustness.

In summary, this work has shown that the global relationships generated by sequence-based models enhance autonomous driving performance in suburban environments using a limited dataset of grey scale images. The models demonstrate superior driving performance compared to classical CNN based models. In particular, the state-space models have been able to achieve the best results while maintaining reasonable parameter sizes and fast inference times. The smooth speed and steering performance shown in the map metrics demonstrate model behaviour consistent with the needs for ride comfort. Further integration with complementary sensing modalities is essential to overcome the inherent limitations of monocular vision. The following chapter builds upon these findings by exploring alternative sensor approaches and multi modal fusion strategies to enhance robustness under varied environmental conditions.

In summary, this work has shown that the global relationships generated by sequence-based models enhance autonomous driving performance in suburban environments using a limited dataset of grey scale images. The models demonstrate superior driving performance compared to classical CNN based models. In particular, the state-space models have been able to achieve the best results while maintaining reasonable parameter sizes and fast inference times. The smooth speed and steering performance shown in the map metrics demonstrate model behaviour consistent with the needs for ride comfort. Further integration with complementary sensing modalities is essential to overcome the inherent limitations of monocular vision. The following chapter builds upon these findings by exploring alternative sensor approaches and multi modal fusion strategies to enhance robustness under varied environmental conditions.

Beyond empirical results, these findings contribute to a broader theoretical understanding of the application of sequence-based vision models and how they encode sequential dependencies in visual driving data. The results suggest that State Space Models (SSMs) implicitly capture spatial relationships in a confined state matrix, allowing them to approximate attention-like global reasoning with lower computational overhead. This highlights a fundamental advantage of SSMs as efficient spatial models that can sustain long-range dependencies without the quadratic complexity of transformer models. Consequently, this work bridges the conceptual gap between simulated and real-world driving performance.

5.7 Conclusion

The results demonstrate that sequential models, particularly recent State Space Models (SSMs), significantly outperform conventional CNN-based architectures for end-to-end autonomous driving. Their scalability and efficiency make them well-suited to embedded platforms with limited compute capacity. VMamba emerged as the most capable model for on-road driving, although achieving full SAE Level 4 autonomy will require further refinement. From the discussion, several key directions are identified:

1. A more comprehensive behavioural breakdown is needed in the mixture of experts system to improve performance across diverse environments.
2. Models should focus on key road features through improved training architectures that incorporate autoencoders and masking strategies.
3. The integration of complementary sensors, such as LiDAR, is necessary to overcome the lighting limitations of purely vision-based systems.
4. Safety mechanisms must be implemented to reduce disengagements and improve stability in unseen scenarios.
5. The use of colour imagery may enhance generalisation, though this comes at the cost of increased model complexity.
6. Maintaining hidden state continuity between the inference steps in SSMs could improve the predictions and consistency of the models.
7. The conversion of PyTorch models to TensorRT files should be investigated for deployment efficiency and real-time performance gains.

The following chapters explore alternative strategies to enhance vision-based driving models, including sensor fusion and multi modal systems. Although testing in this chapter was limited to the defined Eglinton route, the results confirm the feasibility of sequence-based models for real-world autonomous operation and provide a foundation for further generalisation studies.

5.7.1 Novel Contribution

To the author's knowledge, this study represents one of the first systematic evaluations of State Space Model architectures, including VMamba and ViM, applied to real-world end-to-end driving scenarios. The work demonstrates that these architectures achieve strong on road

performance from a limited dataset, while maintaining a lower computational footprint compared to transformer-based baselines. This establishes an important foundation for future research into SSM-driven perception and control in autonomous vehicles.

5.7.2 Future Works

Several observations during vehicle operation point to valuable future work directions. The current system relies on a single hardware controller, limiting usability in public service applications. A web-based interface and a voice-command system would allow on-demand access and multi-user interaction, supported by an onboard language models for dialogue.

The hyper parameter tuning was performed manually, which proved inefficient. Automated tuning frameworks such as Ray Tune [209] could be adopted to perform systematic grid searches for optimal configurations.

Finally, both ViM and VMamba employ raster-like scanning (left-to-right or top-to-bottom) to process images. Although effective, this method neglects spatial adjacency between rows. Alternative scanning strategies such as Hilbert curves could preserve 2D spatial locality more effectively, as illustrated in Figures 5.48 and 5.49 improving the model performance [210, 211].

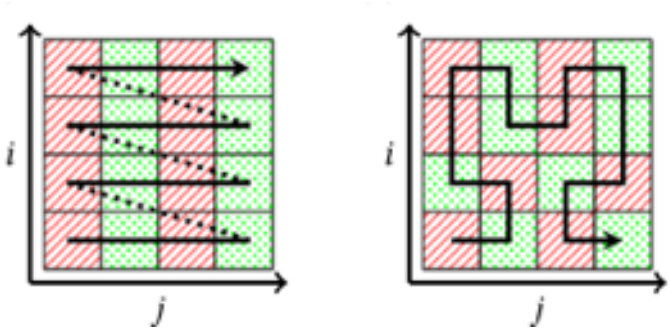


FIGURE 5.48: Hilbert curve [210].

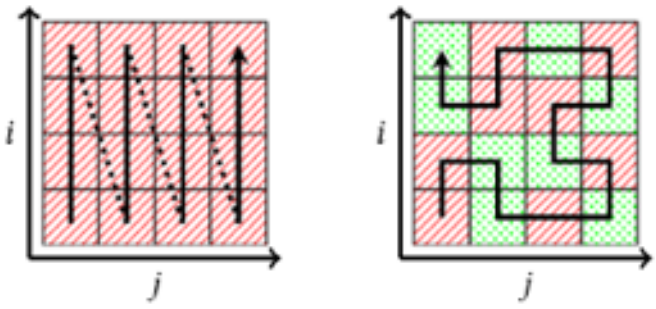


FIGURE 5.49: Transposed Hilbert curve [210].

Chapter 6

Combined LiDAR and Vision Models for Autonomous Driving on Suburban Roads

6.1 Introduction

Chapter 4 reviewed popular driving techniques, particularly those using LiDAR mapping with a mathematical model for local planning. The deterministic nature of the driving provides users with greater confidence in the system's ability to function as the output is always known; however, it can be hard to develop a single algorithm to work across all scenarios. In Chapter 5, we examined end-to-end sequence-based vision models operating in suburban environments. Although vision-based models were able to effectively drive on the road, their ability to perform was subject to variations in lighting conditions. This highlights the need to find an alternative method for driving that does not rely solely on vision and can operate robustly in a range of scenarios.

Initial testing of the autonomous shuttle bus in Eglinton revealed that a GNSS+RTK system would not be appropriate for the given domain. Fluctuations in the GNSS+RTK localisation, shown in the previous chapter, would result in poor driving performance. A 2D SLAM based approach was also rejected, as localisation relies heavily on strong uniquely identifiable features. Much of the route that nUWAy2 drives, either lacks any features, such as ovals and parking lots, or has repetitive features, where many houses share similar external fencing and garden options. 3D SLAM is also rejected as a viable method, due to its high computational requirements, which are not met with the current onboard resources. In addition to this, both GNSS+RTK and SLAM require complex local planners that take time to develop and extensive pre-processed data that align with live sensors.

As an alternative to these approaches, this chapter investigates sequence-based neural networks to predict speed and steering based purely on the LiDAR input. The SSM from the previous chapter have been adapted, by teams like [212], to make predictions based on point-cloud data. If these LiDAR-based models can reliably predict the control signals that enable safe and progressive lane following in a suburban environment, then they can be used as an alternative driving method.

It is expected that the LiDAR models may not perform to the same level as the vision-based models due to the limited resolution of the 16 layer LiDARs and the limited features seen in a point-cloud of a suburban environment. Therefore, any LiDAR based model is expected to be used as a fallback method for the vision-based models until they can be restored. In addition, a combined vision + LiDAR based model is designed to see if the additional information can be used to strengthen the performance of a vision-based model in variable lighting conditions enough to justify an increase in model size and inference time on the constrained resources of the nvidia Orin platform.

Since the focus is on maintaining driving autonomy during camera failures, only lane following models are developed for comparison to the best-performing vision models in the previous chapter. The same key metrics found in Chapter 5 are used to evaluate the performance of these models.

1. **Autonomous driving time** – the percentage of time the model remains in autonomous mode.
2. **Interventions/disengagements** – how often human or hardware interventions are required.
3. **Ride comfort** – assessed by speed transitions and percent change in steering angle over time.

The chapter is broken down into the following five sections:

1. Background on LiDAR-based AI and end-to-end driving models.
2. Description of sequential models.
3. Results from real-world driving.
4. Discussion of findings.
5. Conclusion and suggestions for future work.

6.2 Related Works

Lighting variability has been shown to significantly affect the performance of vision-based driving models [213]. Carvalho et al. demonstrated that flickering AC lighting severely degrades CNN performance in detecting obstacles, with longer exposure times offering limited improvements at the cost of motion blur in dynamic environments [214]. Similarly, Silwal et al. examined lighting effects in agricultural applications and found that using a camera flash before image capture mitigated the impact of natural light, although only at short distances (0.5–1.5 m). In longer ranges, longer exposures reintroduced blurring and reduced detection accuracy [215]. Other studies have explored specific lighting challenges for autonomous driving. Alam et al. investigated glare from bright light sources, introducing a calibration-based glare reduction method that performed well in the HDR4CV dataset but required offline calibration, limiting real-time applicability [216]. In general, these works focus on controlled environments or static datasets, limiting real-world relevance. Many proposed solutions, such as exposure control or offline preprocessing, are impractical for fully autonomous real-time systems, underscoring the need for alternative sensing modalities less affected by light variation.

GNSS+RTK and LiDAR sensors are both robust to changes in lighting. However, as discussed previously, GNSS+RTK depends on external signals, which can introduce reliability issues such as covariance spikes during operation, and also moves away from the objective of a fully self-contained system. LiDAR, in contrast, offers an onboard sensing alternative, but algorithms such as SLAM typically require mapped environments, which conflicts with the goals of Level 5 autonomy. In 2020, McCrae and Zakhor used recurrent neural networks with voxelised LiDAR point clouds to detect pedestrians and vehicles in the nuScenes dataset, achieving accuracies of 52–56% for pedestrians and 67% for vehicles [217, 218]. Similarly, Sun et al. introduced PointMoSeg, a sparse convolutional encoder–decoder architecture operating on sequential point clouds, which achieved 69.9% accuracy for vehicles and 52.9% for pedestrians, but required 126 ms inference time, too slow for real-time deployment [219]. These studies demonstrate the feasibility of LiDAR-based perception, but highlight ongoing challenges in achieving real-time performance.

Cai et al. proposed a deep reinforcement learning model using LiDAR input for end-to-end intersection navigation within the CARLA simulator [189]. Their approach achieved 90% success rates in both dense and regular traffic conditions, though performance degraded at

roundabouts. Liu et al. later developed Fast-LiDARNet, a CNN tested on real-world suburban roads using a 64-layer LiDAR sensor on a Toyota Prius [220]. Their fused model combined odometry and epistemic uncertainty to enhance robustness, outperforming PointNet and demonstrating practical on-road reliability. These studies collectively show the potential of LiDAR for end-to-end driving, but also reveal the computational and data throughput challenges that still constrain real-time performance. Importantly, only a limited number of works have demonstrated on-road validation, leaving a gap between simulation and real-world deployment that this work seeks to address.

With the development of advanced model architectures, LiDAR-based learning has increasingly incorporated Transformer and State Space Model (SSM) backbones. The introduction of Transformer-based point cloud encoders led to Point-BERT and Point-MAE, two leading architectures in this area. Point-BERT integrates bidirectional BERT-style pre-training [221] with geometric tokenisation of 3D point data using a discrete variational autoencoder (dVAE) [222], achieving improved classification accuracy in ModelNet40 [223] and ScanObjectNN [224] [225]. Pang et al. later introduced Point-MAE, a masked autoencoder variant with a more efficient token-masking strategy, achieving state-of-the-art performance on the same benchmarks [226].

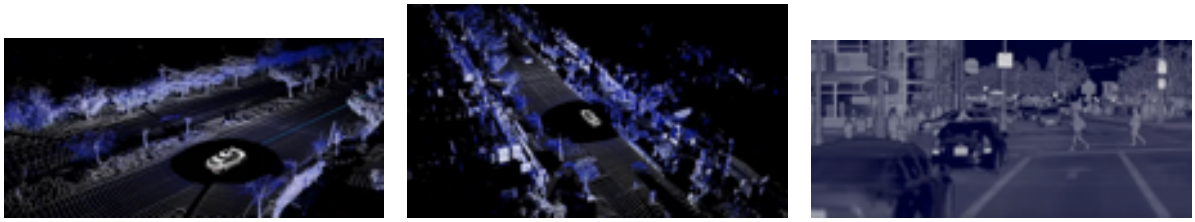
Research combining SSMs with point cloud data remains limited. Wang et al. recently proposed a hybrid Transformer–Mamba architecture, where Transformer blocks model local (intra-group) features and Mamba layers capture global (inter-group) dependencies [227]. This bidirectional design reduced computational complexity compared to pure Transformers and achieved superior results in ModelNet40, ScanObjectNN, and ShapeNetPart [228]. Although these architectures mark significant progress, most work remains focused on static classification datasets, rather than dynamic, real-world driving tasks. This gap motivates the use of the PointMamba model in the present study for end-to-end driving with real LiDAR data.

Performance can often be improved by integrating multiple sensing modalities. Numerous studies have explored sensor fusion, particularly between LiDAR and camera data, to enhance robustness in autonomous systems. Zhao et al. fused 3D-LiDAR and RGB camera data using fuzzy logic for scene parsing, followed by a Markov random field to establish temporal consistency, demonstrating improved detection reliability compared to single-sensor systems [229]. Zha et al. proposed a multi source fusion approach based on YOLO, aligning LiDAR and RGB data through calibration-based projection and introducing a gating mechanism to

suppress unreliable regions [230]. Their evaluation on the KITTI dataset [231] showed notable gains in detection accuracy, though only under controlled dataset conditions. Thu et al. further extended fusion to motion planning, employing CNN encoders for LiDAR image and bird's-eye view data, merged using a Transformer fusion head for waypoint prediction [232]. Their model achieved good performance in CARLA simulations, but suffered from overfitting, indicating the need for larger, real-world datasets.

Although multimodal fusion has proven to be effective, it often incurs higher latency and parameter costs. This work therefore examines whether these trade-offs are justified by the resulting performance improvements, particularly in real-world autonomous driving scenarios.

Commercial systems have also adopted LiDAR-centric approaches. While Tesla favours vision-only autonomy, companies such as Google's Waymo employ LiDAR-based perception with high-definition mapping [233]. As illustrated in Figure 6.1, these maps provide highly detailed 3D environmental representations, allowing precise object separation and scene understanding. However, such systems require significant computational resources and storage, limiting scalability. Consequently, there remains a need for lightweight and efficient LiDAR-based models capable of operating in real time without relying on extensive map data.



(A) Before interaction (B) During interaction (C) After interaction

FIGURE 6.1: Google Waymo's high-definition maps generated using multi-layer LiDARs [233].

In summary, the existing literature demonstrates that most academic research relies on simulation or controlled datasets, with limited real-world validation. This work aims to address this gap by evaluating the effectiveness of State Space Models (SSMs) with LiDAR input in real-world driving conditions. Furthermore, it explores the development of a fused model to mitigate lighting variability and improve the robustness of autonomous driving in unstructured outdoor environments.

6.3 Models and Testing

Building on the demonstrated advantages of State Space Models (SSMs) over other architectures, a SSM-based LiDAR model was selected as the foundational model. This was further combined with the best-performing image-based model from Chapter 5 to create a multi-modal system that leverages the strengths of both LiDAR and visual inputs.

6.3.1 PointMamba Model

The PointMamba model [212] represents an adaptation of the Mamba state-space model architecture for LiDAR-based autonomous driving tasks. Liang et al. demonstrated that PointMamba achieved high precision on the ScanObjectNN dataset, comparable to top-performing transformer models, but with a significantly reduced computational load. In this work, the model structure from Chapter 5 is applied to point cloud data, while maintaining the same dual linear regression setup for predicting vehicle speed and steering.

Point cloud data is collected from two 16-layer Velodyne LiDAR sensors mounted at the front and rear of the autonomous shuttle bus, as illustrated in Figure 6.2. This configuration provides a lightweight high-resolution input suitable for real-time processing. As the incoming data from each LiDAR is referenced in its own coordinate frame, a rotational transform is applied to both point clouds to align them with the shuttle bus frame before fusion. The data was collected separately to allow for future comparative studies between front-only and fused front–rear point cloud models.

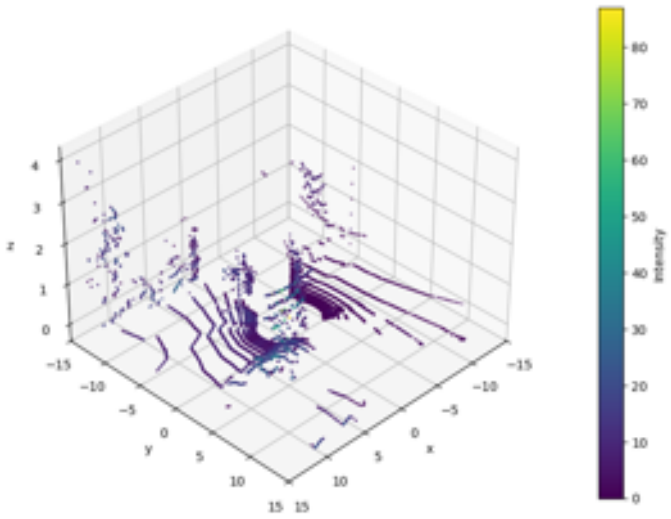


FIGURE 6.2: Point cloud data captured by two 16-layer velodynes on nUWAy shuttle bus.

Similarly to the vision-based Mamba models, PointMamba operates on sequences of data, requiring the point-cloud data to be organised into structured patches. The preprocessing pipeline consists of several stages:

1. **Farthest Point Sampling (FPS):** A subset of points is selected from the original point cloud by iteratively choosing the point farthest from all previously selected points, ensuring maximum coverage with minimal redundancy [212].

2. **Sequence Construction:** The sampled points are ordered using a space filling curve (Hilbert curve) and its transposed variant to preserve spatial locality throughout the sequence [234].

3. **Patch Formation:** The sampled points serve as key points for a k-nearest neighbours (KNN) search within the original point cloud [235]. Each key point forms a local patch consisting of its k nearest neighbours, analogous to the patch-based processing used in image-based models from Chapter 5.

4. **Normalisation and Embedding:** Points within each patch are normalised relative to the keypoint, and a lightweight PointNet [236] is applied to generate patch embeddings. The two sequences from the Hilbert and transposed Hilbert curves are combined and passed to the Mamba blocks for sequential processing.

Figure 6.3 provides an overview of the PointMamba architecture, illustrating the flow from raw LiDAR data through patch construction, embedding, and sequential state-space processing, culminating in the "Task head" which is equivalent to feature extraction. The task head output is then passed to the dual linear regression architecture, as outlined in Chapter 5, where the pointMamba architecture replaces the vision model in the purple block.

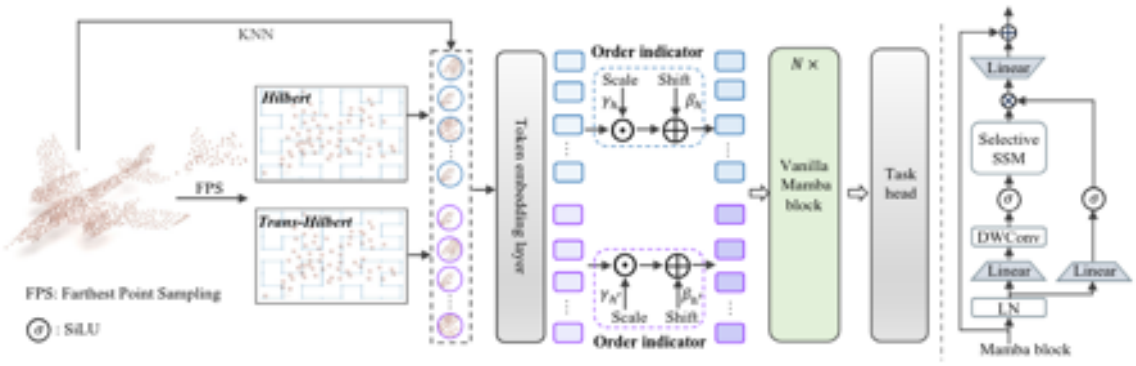


FIGURE 6.3: PointMamba architecture [212].

PointMamba thus extends the Mamba framework to three-dimensional spatial data, enabling efficient and accurate prediction, which can then be extended to real-time autonomous driving applications while maintaining the benefits of linear computational complexity inherent to state-space models.

6.3.2 Joint LiDAR–Vision Model

The second proposed model for this chapter is a fused LiDAR–vision architecture. This is achieved by combining the best-performing vision model from Chapter 5, VMamba, with the PointMamba model discussed in the previous subsection. As noted in the related works, model performance typically improves when combining data from multiple sensor modalities, since each provides complementary environmental information. The goal of this model is therefore to evaluate whether a joint fusion approach can produce more stable and smooth control outputs under variable lighting conditions compared to purely vision-based models (such as VMamba), without significantly increasing model size or inference time, which must remain around 0.05 seconds for real-time operation.

To achieve this, the two feature extraction models are executed in parallel, each receiving its respective input data. Using `torch.jit`, both models are forked to run simultaneously, generating two independent feature representations. These extracted feature maps are then concatenated and passed into the dual linear regression module designed in Chapter 5. The overall architecture is shown in Figure 6.4.

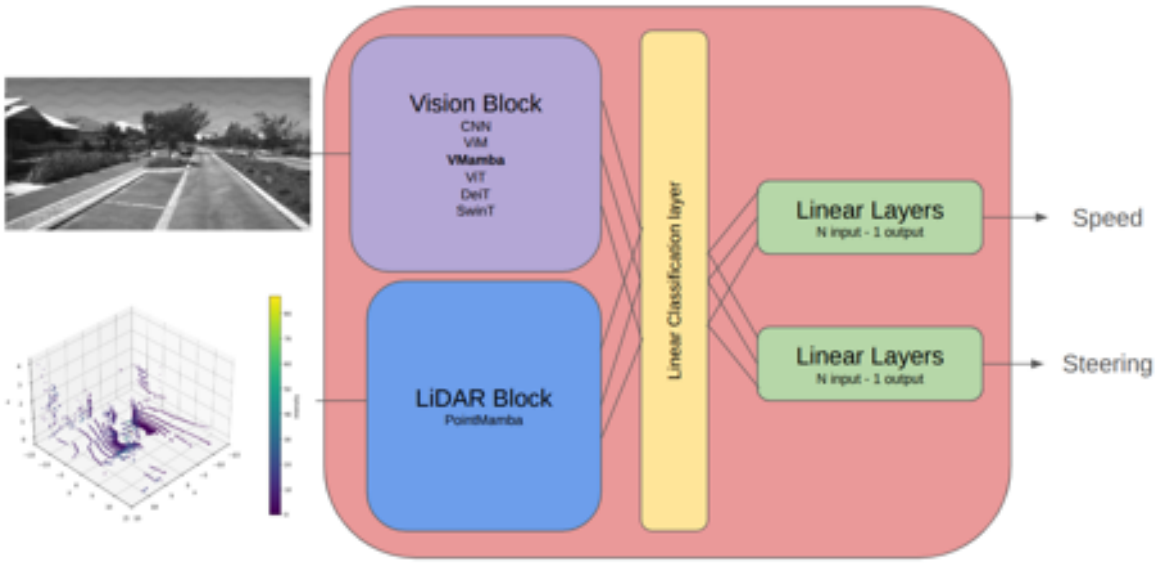


FIGURE 6.4: Joint VMamba and PointMamba model.

The parameters of both feature extraction models are frozen, as they are assumed to have already learned the optimal representations for their respective sensor domains. This strategy also reduces training time, as only the dual linear regression component requires optimisation. During runtime, the model receives continuous image and point cloud streams; LiDAR transforms are trigger with new front LiDAR frames and stored locally until an image is received which triggers the model prediction, enabling synchronised multimodal inference.

6.3.3 Testing

The PointMamba model was evaluated on the Eglinton map within the AWSim simulator. The model successfully completed a full autonomous lap of the route without any external intervention or manual correction, indicating stable learning and effective generalisation of driving behaviour.

During the test the model maintained good road position as well as steering and speed control. This demonstrated the LiDAR models ability to pick on key point cloud elements for autonomous driving.

6.4 Results

Following simulation testing, the models were trained and evaluated on real-world data collected from Eglinton. The selected driving parameters for the PointMamba model are listed in Table 6.1. For the Joint LiDAR-Vision model, both the PointMamba and VMamba components were frozen, and thus the same parameters were retained for each respective submodel.

Parameter	Value
trans_dim	512
depth	12
cls_dim	1000
group_size	32
num_group	64
encoder_dims	512
drop_out	0.1
lr	1e-4
weight_decay	0.05
gamma	0.8
npoints	3072
points_all	4300
padding	4300

TABLE 6.1: PointMamba model parameters for Eglinton dataset training.

Table 6.2 summarises the key deployment metrics. Although the inference time of the Joint LiDAR-Vision model slightly exceeds the target of 0.05 seconds, this only results in skipping approximately every fourth input frame. However, the model remains suitable for real-time driving applications. Future work could focus on streamlining the model to reduce inference time, potentially through parameter pruning or model compression. The PointMamba model achieves a parameter count comparable to previous vision-based models such as VMamba and EfficientNet. For the Joint model, it has approximately double the number of parameters but only a subset of parameters are trainable, as both the LiDAR and vision feature extractors are assumed to be pre-optimised.

Model	File size (MB)	Inference (sec)	Parameters (10^6)
PointMamba	273.3	0.043	22.75
Joint LiDAR-Vision	238.9	0.069	48.89 (5.39)

TABLE 6.2: Model file size, inference time, and parameter count for LiDAR-based models. Brackets indicate trainable parameters.

Tables 6.3 and 6.4 present the training results. Both models achieve training and validation losses comparable to those reported in Chapter 5. These models were trained exclusively

for lane-following scenarios, which provides a good baseline for on-road performance. The accuracy metrics show a slight decline in steering and speed accuracy compared to VMamba and ViM, but remain close to those achieved by the ViT model. This consistency suggests that the LiDAR-based models generalise effectively to real-world data and are suitable for integration into the real-world driving platform.

Model	Training Time (min)	Training Loss	Validation Loss
PointMamba	**67**	0.0009	0.0010
Joint LiDAR-Vision	186	**0.0004**	**0.0003**

TABLE 6.3: Training duration and corresponding losses for LiDAR-based models.

	Speed Accuracy (%)		Steering Accuracy (%)	
Model	**Training**	**Validation**	**Training**	**Validation**
PointMamba	**76.66**	**90.03**	**83.09**	**94.6**
Joint LiDAR-Vision	71.86	76.44	79.92	82.88

TABLE 6.4: Training and validation speed and steering accuracy for LiDAR-based models (lane-following only).

The results presented in Figures 6.5 and 6.6, together with Tables 6.5–6.6, illustrate model performance on the unseen dataset for the lane following task. Both models demonstrate strong overall performance, particularly for steering predictions, where more than 90% of outputs from each model fall within 10% of the ground-truth values.

In comparison with the vision-based models presented in Chapter 5, the LiDAR-based models appear capable of achieving improved driving performance, as indicated by lower mean and median error values. However, it should be noted that PointMamba exhibits a higher standard deviation in steering predictions, which may negatively influence overall driving stability and consequently affect final on-road performance.

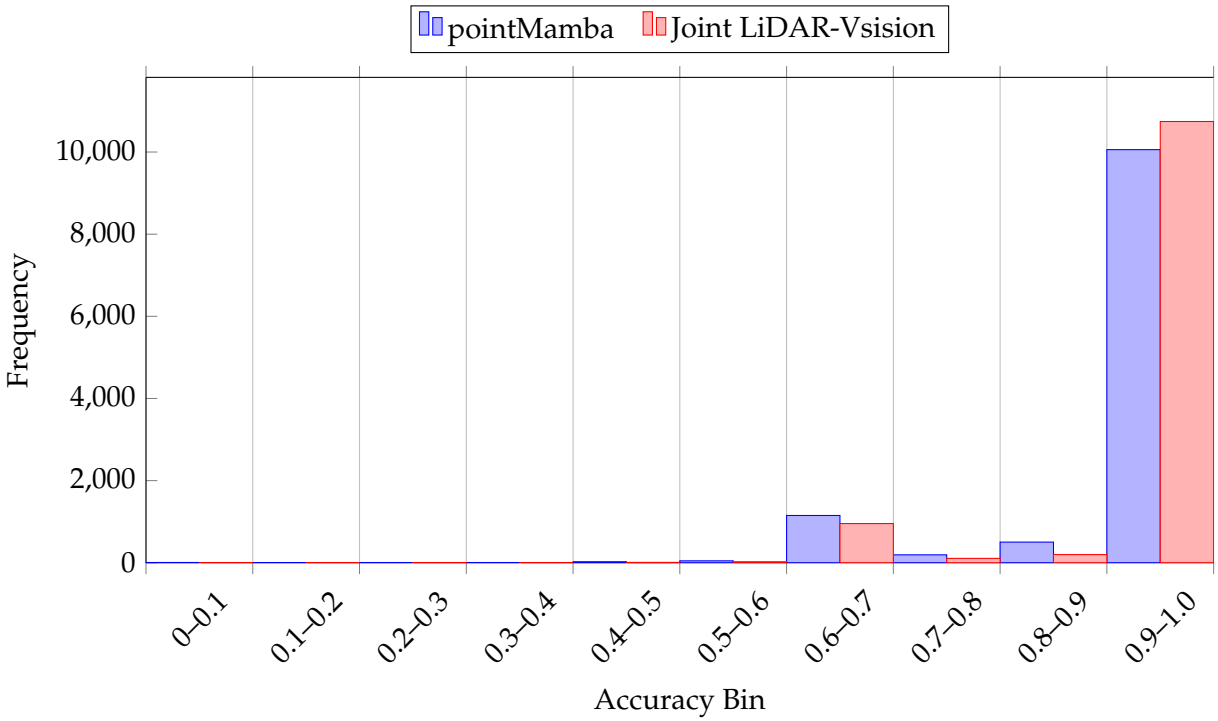


FIGURE 6.5: Lane following bin frequencies - speed.

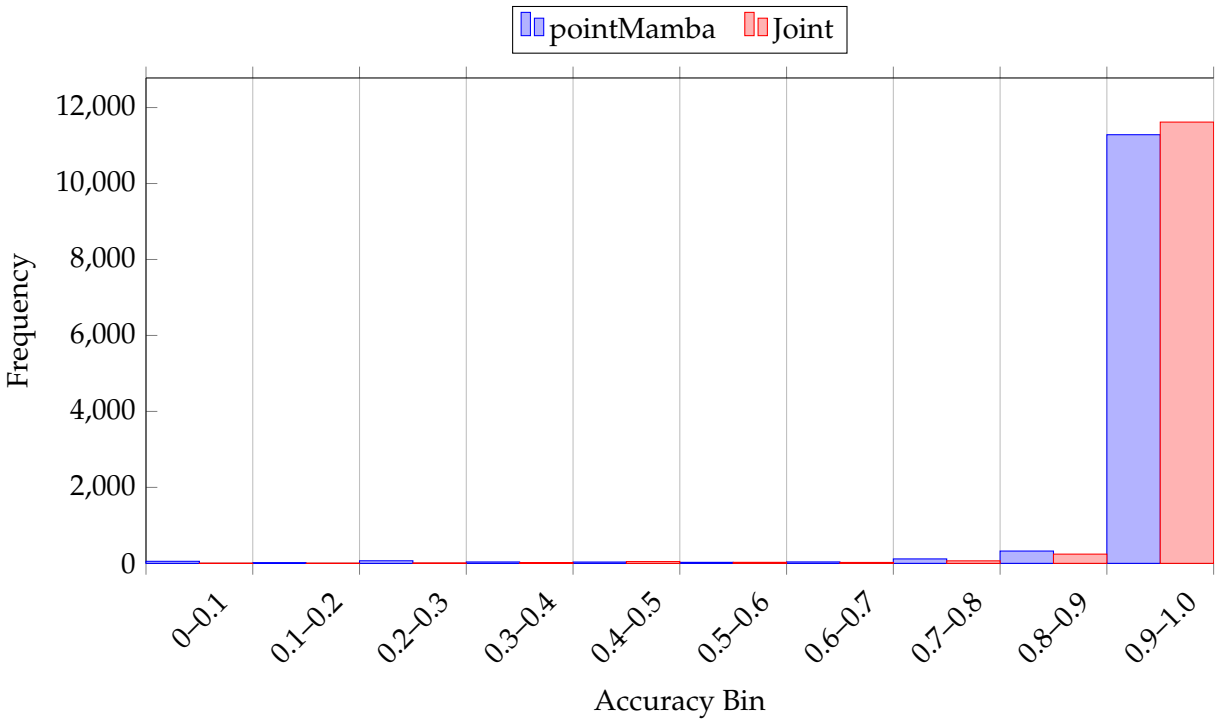


FIGURE 6.6: Lane following bin frequencies for five models - steering.

Speed Error (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
pointMamba	58.68	70.09	75.29	77.43	79.29	80.40	97.08	98.95	99.30	0.47
Joint L-V	85.74	89.15	90.22	90.80	91.97	91.97	91.97	99.61	99.75	0.19
Steering Error (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
pointMamba	79.27	92.01	94.83	95.91	96.53	97.09	97.33	97.59	97.73	2.01
Joint L-V	90.27	96.41	97.98	98.42	99.09	99.09	99.09	99.17	99.28	0.61

TABLE 6.5: CDF Lane following Error Comparison for Speed and Steering

Speed Error					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
pointMamba	9.93	0.00	100.00	1.92	14.08
Joint L-V	**4.60**	0.00	**96.94**	**0.75**	**10.19**
Steering Error					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
pointMamba	4.71	0.00	100.00	1.63	11.56
Joint L-V	**2.56**	0.00	**93.53**	**1.18**	**5.94**

TABLE 6.6: Summary of Lane Following Model Error for Speed and Steering

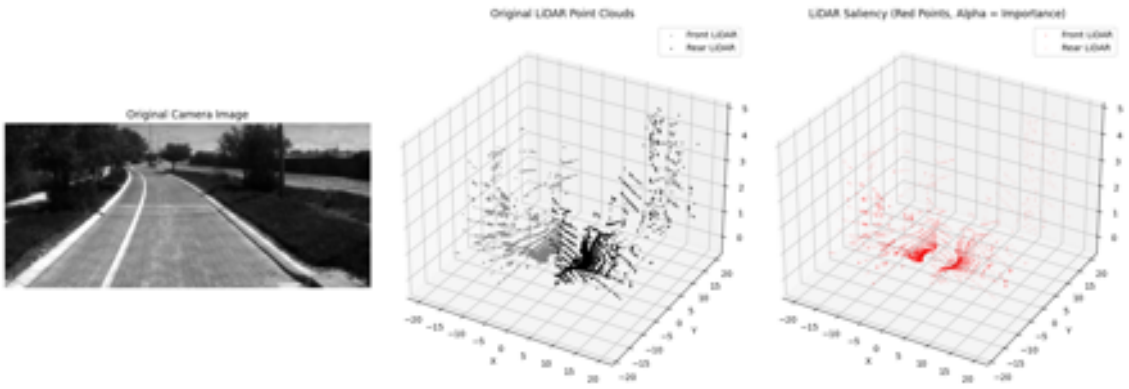
6.4.1 Saliency and ground truth comparison

Before deployment on public roads, the models were evaluated using unseen test data to verify the reliability of the prediction. As in Chapter 5, both saliency maps and ground truth comparisons were generated to assess the focus and precision of the model output.

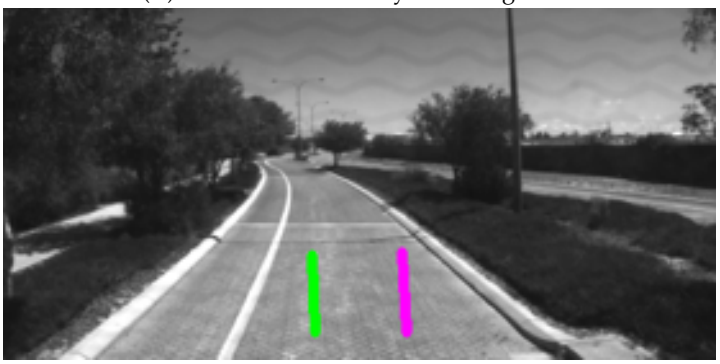
In the following figures, the saliency visualisations are shown alongside the corresponding camera image for human interpretation as pointclouds are not easy to read without further context. The original LiDAR point cloud is also presented with the saliency map itself to

give a visual reference point for the reader. The saliency map uses the alpha (transparency) value of each point to indicate its relative importance, points with higher opacity correspond to regions the model focuses on more strongly during inference. The corresponding ground truth visualisations show the manual driving output in green and the model's prediction in purple following the same speed and steering metrics from Chapter 5.

Although the saliency maps do not always provide a clear indication of attention due to the high point density, general patterns can still be observed. Points farther from the vehicle tend to have minimal influence, whereas points nearer the vehicle, particularly in the forward and rear regions, contribute more significantly to the model's decision process. The ground truth comparisons confirm that the LiDAR-based model closely aligns with the ground truth outputs, demonstrating strong adherence to the desired steering and speed behaviours across a range of driving scenarios.

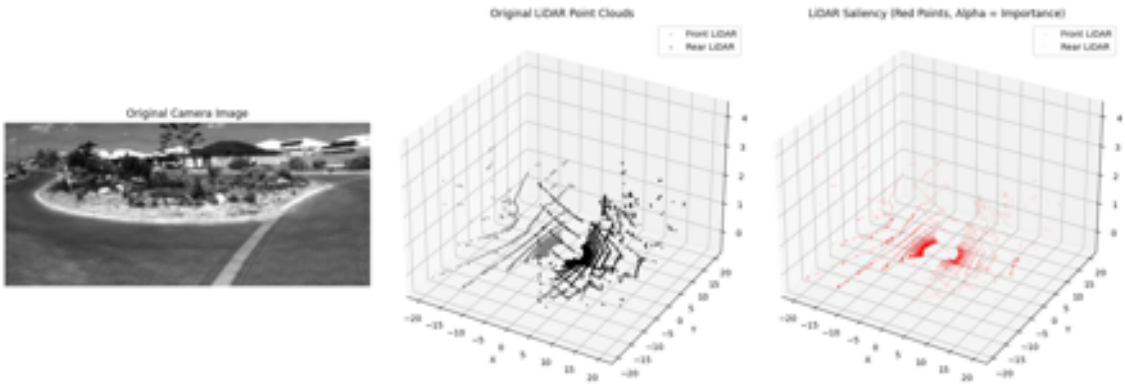


(A) Lane Follow Saliency - Bending Road

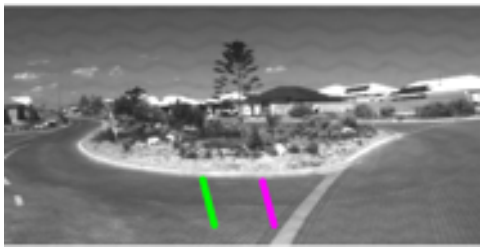


(B) Lane Follow Ground Truth (green) vs Model Output (purple) - Bending Road

FIGURE 6.7: Ground truth and saliency for lane following point cloud - bending road.

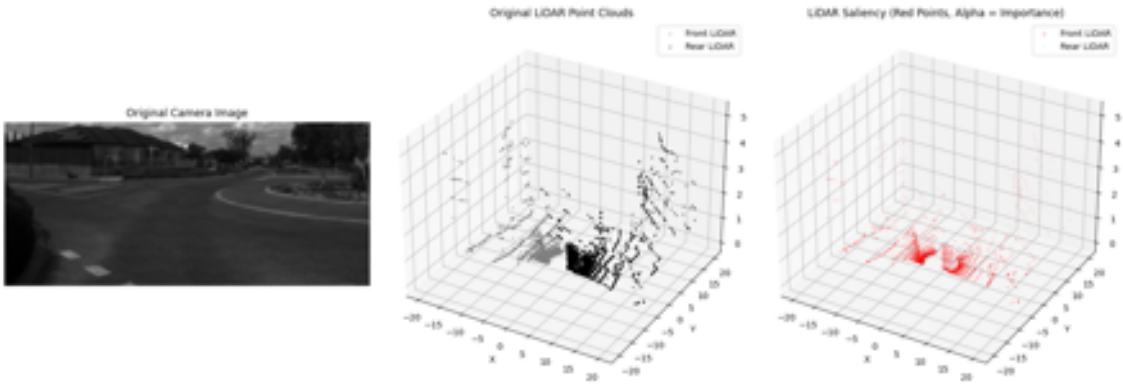


(A) Lane Follow Saliency - Intersection



(B) Lane Follow Ground Truth (green) vs Model Output (purple) - Intersection

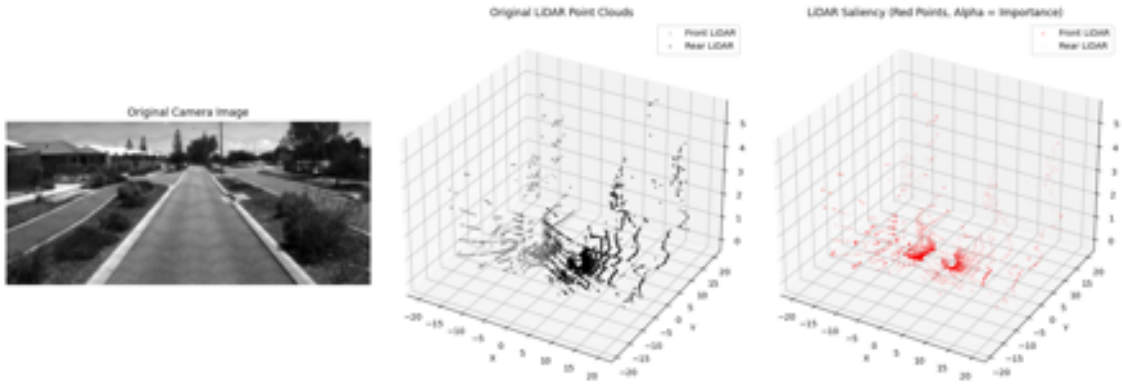
FIGURE 6.8: Ground truth and saliency for lane following point cloud - Intersection.



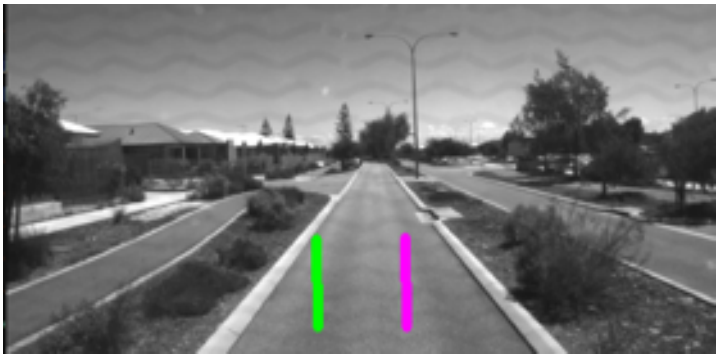
(A) Lane Following Saliency - Round About

(B) Lane Following Ground Truth (green) vs Model Output (purple) - Round About

FIGURE 6.9: Ground truth and saliency for lane following point cloud - Round About.



(A) Lane Following Saliency - Straight



(B) Lane Following Ground Truth (green) vs Model Output (purple) - Straight

FIGURE 6.10: Ground truth and saliency for lane following point cloud - Straight.

6.4.2 Driving performance – overview

The driving performance of each model is summarised in the following three tables. Table 6.7 presents the raw results. PointMamba showed poor road performance, requiring constant intervention and, therefore, only one trial run was completed. The VMamba results from Chapter 5 are also included for comparison with the Joint LiDAR-Vision model, which augments VMamba with PointMamba data.

Table 6.8 shows the results after removing periods when the vehicle speed was approximately zero, which usually corresponded to waiting while parked or in traffic. Such instances disproportionately affect the autonomy percentage depending on traffic conditions and are therefore excluded.

Finally, Table 6.9 shows the adjusted performance results, where the disengagements and interventions occurring in untrained scenarios were removed. This adjustment ensures a fair comparison between models by normalising for differing exposure to scenarios outside their training domain.

It should also be noted that the LiDAR models were trained exclusively for lane-following tasks, which constitute the majority of driving time. In contrast, VMamba data include lane-following, parking, reversing, and pull-out manoeuvres. VMamba was previously shown to perform strongly in reversing and pull-out tasks, while being less effective in parking. These additional behaviours may slightly influence the results, but in general the presented data is expected to closely approximate VMamba's pure lane following performance.

Model	PointMamba	Joint LiDAR–Image	VMamba
Drive time (min)	25.7	24.85 ± 1.97	24.03 ± 2.23
Autonomy (%)	74.59	85.45 ± 1.35	84.76 ± 5.40
Disengagements (#)	7	5.33 ± 3.21	1.00 ± 0.00
Interventions (#)	51	17.33 ± 5.86	14 ± 4.16

TABLE 6.7: Raw driving performance results for LiDAR models on suburban roads.

Model	PointMamba	Joint LiDAR–Image	VMamba
Drive time (min)	22.10	21.43 ± 1.25	19.40 ± 1.76
Autonomy (%)	74.59	85.45 ± 1.35	90.77 ± 3.62
Disengagements (#)	7	5.33 ± 3.21	1.00 ± 0.00
Interventions (#)	51	17.33 ± 5.86	14.33 ± 4.16

TABLE 6.8: Driving performance excluding stationary periods (speed ≈ 0).

Model	PointMamba	Joint LiDAR–Image	VMamba
Drive time (min)	19.53	18.99 ± 1.11	$\mathbf{18.54 \pm 1.32}$
Autonomy (%)	84.49	$\mathbf{97.31 \pm 2.23}$	95.13 ± 2.79
Disengagements (#)	4	3 ± 2.65	$\mathbf{0.67 \pm 0.58}$
Interventions (#)	32	$\mathbf{2.67 \pm 2.08}$	6.67 ± 2.31

TABLE 6.9: Final performance metrics for models tested on suburban roads. Results are adjusted for duplicate counts (disengagements and interventions) and excluding untrained scenarios (mean $\pm$ standard deviation).

Interventions were categorised into four primary types and two secondary (minor) types:

1. Poor driving behaviour in lane-following mode requiring intervention.
2. Untrained driving scenario intervention (removed in adjusted results).
3. Acceptable driving behaviour but with degraded passenger comfort (e.g., weaving, driving close to parked vehicles).
4. Poor driving behaviour in parking, reversing, or pulling out, requiring manual intervention.

Of these, only Category 2 interventions are removed in the adjusted results, along with their associated time.

The two minor categories are:

- **Estops:** Vehicle disengaged due to low-level hardware triggers. Manual intervention is removed as duplicate data, but time penalties are retained.
- **Invalid interventions:** Rare occurrences, typically at the start of data collection before runs began. These were excluded from both counts and time metrics.

	Category 1		Category 2		Category 3		Category 4		Estops		Invalid	
Item	Count	%	Count	%	Count	%	Count	%	Count	%	Count	%
PointMamba	32	64	13	26	0	0	–	–	4	8	1	2
Joint LiDAR–Image	7	16.67	29	69.05	1	2.38	–	–	4	9.52	1	2.38
VMamba	13	35.14	13	35.14	1	2.70	6	16.22	3	8.11	1	2.70

TABLE 6.10: Breakdown of manual interventions.

The data collected for each model is presented in the following sections as a series of maps. Each map shows the average performance across all drives for each model, with the exception of interventions, where all instances are displayed to highlight specific areas where the model encounters difficulties.

6.4.2.1 PointMamba

Figures 6.11a and 6.11g show the autonomy percentage with multiple gaps as a result of constant manual interventions. Figure 6.11b illustrates how PointMamba struggles to maintain close proximity to the optimal path, while figure 6.11f highlights the high instability in steering throughout much of the driven route. Due to this instability and poor path adherence, the vehicle's speed, shown in Figure 6.11c, had to be reduced for most of the drive.

(A) Driving mode distribution.

(B) Deviation from the optimal path.

(C) Average speed.

(D) Average speed relative to optimal.

(E) Steering relative to optimal.

FIGURE 6.11: On-road performance of the PointMamba model showing driving mode distribution, path deviation, average speed, average speed relative to optimal and steering relative to optimal.

(F) Trajectory smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 6.11: Additional PointMamba performance metrics including smoothness and intervention counts.

6.4.2.2 Joint LiDAR Vision

Combining the VMamba model with PointMamba shows a clear improvement over the standalone PointMamba model. However, when compared to the VMamba results in Chapter 5, the joint model exhibits a greater number of disengagements, as seen in Figure 6.12g. These disengagements are predominantly isolated to two narrow sections of the driving path. At these same locations, Figure 6.12f highlights notable steering instability. The speed profile in Figure 6.12c indicates reduced average speed in the high-speed section after the third roundabout, suggesting a more cautious driving pattern.

(A) Driving mode distribution.

FIGURE 6.12: On-road performance of the Joint LiDAR–Vision model showing driving mode distribution.

(B) Deviation from the optimal path.

(C) Average speed.

(D) Average speed relative to optimal.

(E) Steering relative to optimal.

(F) Trajectory smoothness.

FIGURE 6.12: On-road performance of the Joint LiDAR–Vision model showing path deviation, average speed, average speed relative to optimal, steering relative to optimal and smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 6.12: Additional Joint LiDAR–Vision performance metrics including intervention counts.

6.4.3 Driving performance – Best

Table 6.11 presents the results from the best-performing runs of each model. The Joint LiDAR–Vision model achieved the highest autonomy time (99.82%) with the lowest number of manual interventions (1), outperforming both the PointMamba and VMamba models in this regard. However, this came at the cost of a higher number of disengagements compared to VMamba.

It should also be noted that the LiDAR-based models were only evaluated on lane-following scenarios, whereas the VMamba vision model was tested more broadly. This distinction should be considered when comparing overall performance.

Model	PointMamba	Joint LiDAR–Vision	VMamba
Drive time (min)	21.72	18.32	**18.01**
Time autonomy (%)	80.60	**99.82**	98.23
Disengagements	7	2	**0**
Interventions	22	**1**	4

TABLE 6.11: Best driving performance for each model.

The following sections present the detailed map-based performance metrics for the best drive of each LiDAR-based model. These include distributions of driving modes, deviations from the optimal path, speed control, steering behaviour, and intervention counts. Together, these plots provide a more comprehensive view of how each model handled the route beyond the summary statistics in Table 6.11.

The following sections present the driving metrics mapped for each model's best run. Unlike the earlier analysis, all driving modes are included, and the metrics for speed, steering

smoothness, and path deviation are reported across all autonomous and manual modes. Ideally, the steering smoothness values remain low except during expected manoeuvres such as turning at intersections, moving through round abouts, or parking. Similarly, vehicle speed should vary gradually to enhance passenger comfort, while deviations from the optimal path should remain minimal outside of parking manoeuvres.

6.4.3.1 PointMamba

Figure 6.13a shows that manual interventions occurred frequently, particularly in narrow road sections. Steering behaviour was observed to be unstable (Figure 6.13c), and the vehicle was unable to maintain higher speeds for the majority of the route (Figure 6.13b). This unstable driving performance contributes to rider discomfort and results in a high number of interventions and disengagements, as shown in Figure 6.13e. Additionally, deviations from the optimal path (Figure 6.13d) highlight areas where the model struggled with precise lateral control, particularly in constrained sections of the course.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

FIGURE 6.13: Performance of PointMamba during its best driving run, including driving mode and speed.

(C) Steering smoothness expressed as angular velocity (rad/s).

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 6.13: Performance of PointMamba during its best driving run, including steering smoothness, path deviation, and interventions.

6.4.3.2 Joint LiDAR–Vision

The multi-modal model, which integrates both LiDAR and camera data, demonstrates strong overall driving performance. However, as shown in Figure 6.14b, it exhibits variable speed fluctuations and multiple disengagements (Figure 6.14e). These disengagements do not correspond to steering instability or path deviation, as indicated in Figures 6.14c and 6.14d.

Reviewing the recorded ROSbag footage suggests that the joint model has inherited aspects of the LiDAR-only model's behaviour, particularly a tendency to follow the road at a slight lateral offset. On narrow roads, this misalignment can lead to boundary violations and subsequent disengagements.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

(C) Steering smoothness expressed as angular velocity (rad/s).

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 6.14: Performance of the Joint LiDAR–Image model during its best driving run, including driving mode, speed, steering smoothness, path deviation, and interventions.

6.4.4 Lane Following – Road Features

Using the same metrics from Chapter 5, the models performance on various road features is tested. Table 6.12 summarises the performance of each model at these critical landmarks.

Landmark	PointMamba	Joint LiDAR Vision	VMamba
Roundabout 1	1	**3.00 ± 0.00**	**3.00 ± 0.00**
Snaking road	1	1.33 ± 0.58	**1.67 ± 0.58**
Roundabout 2	2	**3.00 ± 0.00**	2.67 ± 0.58
Intersection 1	1	**1.67 ± 0.58**	1.33 ± 0.58
Dual lane unmarked 1	1	**2.00 ± 0.00**	1.67 ± 0.58
Intersection 2	**2**	1.00 ± 1.00	1.33 ± 0.58
Carpark	1	**4.67 ± 0.58**	1.67 ± 0.58
Intersection 3	2	1.00 ± 1.00	**1.67 ± 0.58**
Dual lane unmarked 2	1	1.33 ± 0.58	**1.67 ± 0.58**
Intersection 4	0	**2.00 ± 0.00**	**2.00 ± 0.00**
Roundabout right	1	**2.33 ± 0.58**	**2.33 ± 0.58**
Dual lane marked	0	**2.00 ± 0.00**	**2.00 ± 0.00**
Roundabout 3	**3**	**3.00 ± 0.00**	2.00 ± 0.00
Roundabout 4	1	**3.00 ± 0.00**	2.00 ± 0.00
Total Score (/36)	17	**31.33 ± 0.35**	**28.67 ± 0.41**

TABLE 6.12: Performance of each model across key landmark driving event.

From Table 6.12, it is evident that the Joint LiDAR–Vision model consistently outperforms the other models across multiple landmarks. Its higher average total score (31.33) compared to VMamba (28.67) and significantly reduced standard deviation demonstrate improved reliability and robustness. However, the PointMamba model struggled throughout the route, reflected in its substantially lower total score of 17. With further refinement and improved LiDAR feature representation, these results suggest that future SSM-based LiDAR architectures could further enhance performance across complex driving environments.

6.4.5 Variations of PointMamba and Joint LiDAR Vision

Based on testing of the PointMamba model, several variations were explored to improve performance. Table 6.13 summarises the modifications, training time, model file size, and

parameter count.

Model	Training (min)	File size (MB)	Parameters (10^6)
PointMamba - Shrunk	57.54	273.4	22.75
PointMamba - Unstructured	63	254.6	21.19
PointMamba - Lateral Shift	68.64	273.4	22.75
PointMamba - Intensity	67.14	254.6	22.75
Joint - Residual	204	244.5	50.28(5.39)

TABLE 6.13: Training time, file size, and parameter count for each model.

Table 6.14 shows the training and validation loss, as well as the speed and steering accuracy for each model.

	Loss		Speed Accuracy (%)		Steering Accuracy (%)	
Model	**Train**	**Val**	**Train**	**Val**	**Train**	**Val**
PointMamba - Shrunk	0.0011	0.0012	70.20	84.06	71.98	91.72
PointMamba - Unstructured	0.0011	0.0012	73.27	87.76	81.18	93.49
PointMamba - Lateral Shift	0.0017	0.0018	64.36	81.09	51.16	75.05
PointMamba - Intensity	0.0015	0.0015	64.72	80.41	75.35	89.89
Joint - Residual	**0.0004**	**0.0004**	70.90	72.74	78.59	80.04

TABLE 6.14: Training and validation loss, speed accuracy, and steering accuracy for PointMamba and Joint models.

Several strategies were explored to improve PointMamba's performance. First, the point cloud was "shrunk" to focus on the area immediately surrounding the vehicle, reducing the influence of points above road edges (Figure 6.15). This saw no improvement over the original model and in fact the accuracy went down.

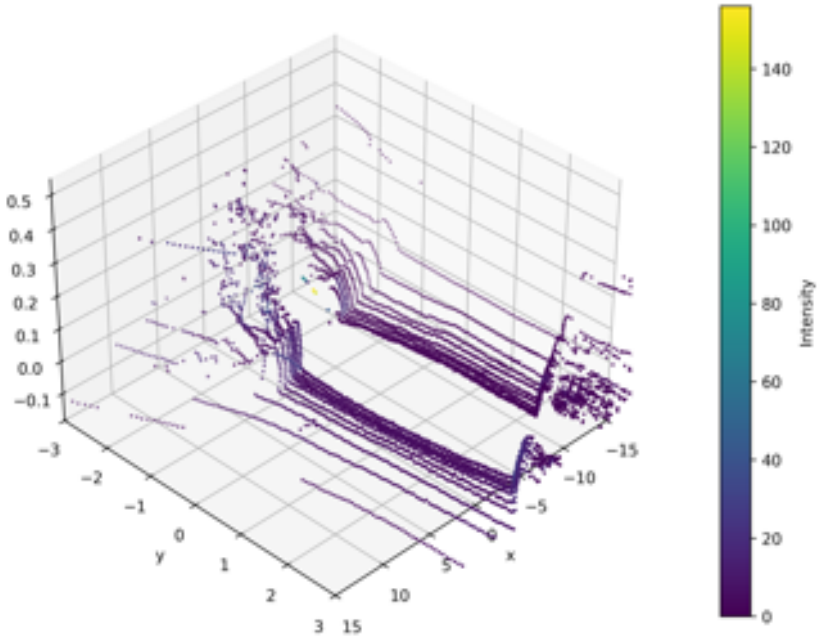


FIGURE 6.15: Shrunk point cloud.

Next, an unstructured point cloud was tested to see whether the model could establish relationships between inputs regardless of orientation (Figure 6.16). Although the validation loss and accuracy remained close to the original model, the outputs were inconsistent when applied multiple times to the same point cloud.

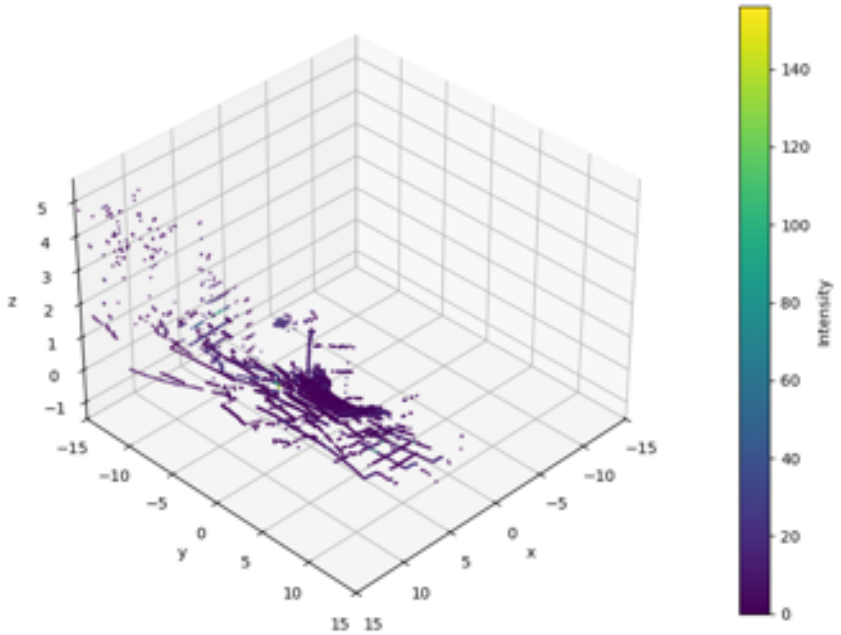


FIGURE 6.16: Unstructured point cloud.

A lateral shift was also applied to improve recovery behaviours on straight road sections. However, this modification degraded performance, likely because occlusions and beam angles were not accurately accounted for. Implementing lateral shifts correctly would require a full 3D map to generate realistic LiDAR views.

A final variation of the PointMamba model incorporated the intensity values from the LiDAR point cloud. Intensity represents the amount of reflected light, and during testing, it was observed that the system could detect road markings using this information. Despite this potential advantage, the overall performance of the model degraded significantly after including intensity data, suggesting that the model is not equipped to deal with the extra channel.

Finally, a joint model with a residual connection was developed to reduce the impact of a poorly performing PointMamba on VMamba. The residual connection corrects VMamba by combining the two model outputs and using it as an offset to the original prediction (Figure 6.17). The VMamba model weights are frozen so only the correction value is learnt. An alternative method would be to use just the LiDAR data as a correction but the LiDAR model performance was deemed not good enough to explore this further.

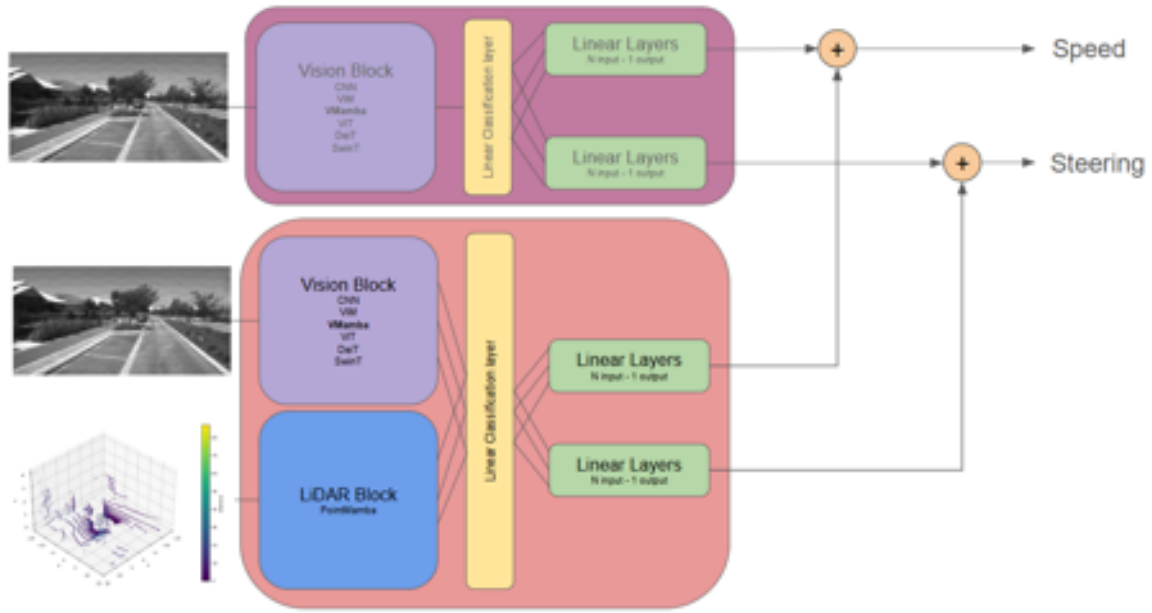


FIGURE 6.17: Residual model layout.

After training, the residual model maintained similar speed and steering accuracy to the original VMamba model. Interestingly, the validation results were slightly worse, suggesting that while the residual connection preserves the baseline performance of the original joint model, it does not necessarily improve generalisation.

6.5 Discussion

This chapter provides a critical interpretation of the experimental results presented in this chapter, focusing on the performance, behaviour, and implications of the developed models. The discussion is organised into two main sections. The first section examines the *PointMamba* model, which relies solely on LiDAR-based perception for end-to-end driving control. The second section considers the *Joint LiDAR–Vision* model, which integrates visual and spatial modalities to improve scene understanding and driving accuracy. This part of the discussion evaluates how sensor fusion contributes to improved robustness, as well as the limitations and trade-offs that arise from increased model complexity.

6.5.1 PointMamba

From the results presented in Table 6.9, it is evident that the current implementation of the *PointMamba* model is not yet sufficient for fully autonomous driving in suburban environments. The model required 32 manual interventions during testing, indicating that it cannot currently be trusted to operate independently. However, as observed in Section 6.4.1 and the driving metric maps in Section 6.4.2.1, the model demonstrated the ability to effectively control steering at intersections and negotiate roundabouts to a limited extent. Empirically, when the vehicle remained in the centre of the lane, autonomous control was maintained for longer periods. However, small perturbations that led the vehicle to the edge of the road often triggered interventions or disengagements.

These findings suggest that the model has successfully learned the general driving behaviour observed in the training data but lacks the capacity to generalise to real-world perturbations. This highlights a key limitation of purely dataset-driven imitation learning approaches, which tend to reproduce training behaviours without robust recovery mechanisms. As shown in Table 6.3, the *PointMamba* model achieved low training and validation losses comparable to vision-based models, and Table 6.4 indicates strong steering and speed accuracy. This is further reinforced by the CDF and histogram tables and figures presented which show strong model performance. However, unlike the vision-based models in Chapter 5, this model was not trained using augmented data that introduced recovery behaviours. Consequently, it struggles to recover once it deviates from the nominal driving path. This again highlights that on-road performance is not always equivalent to dataset testing and simulation results which can be idealised by ignoring things like data drift.

Attempts to address this limitation through lateral data shifting were discussed in Section 6.4.5. Although this augmentation was intended to simulate recovery behaviour, the resulting accuracy and loss metrics (Table 6.14) degraded significantly. This decline can be attributed to the inherent limitations of the 16-layer LiDAR sensor, which captures only sparse spatial detail. The lateral shift of the LiDAR data does not accurately model the geometric perspective or account for occlusion. A more appropriate approach would involve reconstructing a full 3D LiDAR environment map from the aggregated training data. This 3D representation could then be used to synthetically generate realistic point clouds from new viewpoints, thereby enabling effective learning of recovery behaviours.

Further experiments with structured and unstructured data revealed that unstructured data improved performance to some degree. However, during ground truth testing, the model produced unstable outputs for identical inputs, indicating sensitivity to stochastic sampling. This instability arises from random cropping during point cloud preprocessing, which should be removed or standardised to ensure deterministic model behaviour. Incorporating LiDAR intensity information was also explored, but this addition resulted in decreased overall performance. However, intensity data could prove beneficial in future work in identifying lane markings or other surface characteristics relevant to driving decisions.

An additional observation is that the LiDAR-based model tends to under steer during gradual turns, performing well on sharp corners and roundabouts but failing to match the nuanced steering of the vision-based models. This limitation is likely due to the restricted training data and sparse sensor input. Collecting additional data across varied driving scenarios may help mitigate this effect.

Overall, the results indicate that a 16-layer LiDAR provides sufficient spatial information for basic lane-following control in theory, but the absence of recovery data and temporal context constrains generalisation and robustness. Future work could explore the integration of temporal sequences of LiDAR frames to enhance the stability and continuity of predictions. Although this temporal component may increase the complexity of the model, it has shown promise in other LiDAR-based driving systems and represents a logical next step in improving the PointMamba model.

6.5.2 Joint LiDAR–Vision

The *Joint LiDAR–Vision* model demonstrated greater potential compared to the standalone *PointMamba* implementation, largely due to the integration of the robust visual features learned

by the *VMamba* model presented in Chapter 5. Initial tests showed promising driving behaviour, particularly in maintaining lane-centred trajectories. However, performance degradation was observed during extended testing, primarily attributed to the weaker LiDAR component, which negatively influenced the overall stability of the fused model.

As shown in Table 6.2, the joint model exhibits a considerable increase in both parameter count and inference time relative to the single-modality models. Although the inference time slightly exceeds the desired 0.05-second threshold, it remains within an acceptable range for real-time driving applications. Further optimisation is possible by unfreezing and fine-tuning the feature extraction layers of both the LiDAR and vision backbones, which could improve fusion quality while potentially reducing redundant parameters.

Training metrics in Tables 6.3 and 6.4 show low training and validation losses, indicating effective learning, though accuracy values are slightly lower than those achieved by the standalone vision and LiDAR models. Despite this, the accuracy remains within acceptable operational limits. The real world performance, summarised in Table 6.9, demonstrates that the fused model achieved a higher level of autonomy than the *PointMamba* model, with fewer manual interventions. However, an increased number of disengagements were observed, suggesting that while the model is capable of sustaining autonomy for longer periods, it still struggles with sudden or ambiguous environmental conditions. Improving the LiDAR feature extraction component is expected to significantly enhance these results.

It is also important to note that the current results for the joint model are based solely on lane-following data, whereas the *VMamba* model from Chapter 5 was trained across multiple driving behaviours, including parking and reversing. Expanding the dataset to include these additional behavioural classes will allow further model testing.

The analysis of Table 6.12 further reveals notable improvements in the recognition of road characteristics, indicating that the inclusion of LiDAR data contributes positively to spatial awareness and environmental understanding. This improvement suggests that even a relatively low-resolution LiDAR sensor can enhance scene interpretation when properly fused with visual information. Further improvements are anticipated with the addition of more varied data and improved model training strategies.

Despite the current limitations of the LiDAR component, the joint model demonstrates clear benefits in perception and control performance. From a systems perspective, the fusion of modalities appears to justify the added computational complexity. The mixture-of-experts approach used here offers an efficient framework in which improvements can be targeted at individual behaviour reducing the impact to the overall system. Future optimisation should

explore partial unfreezing of the vision and LiDAR encoders to allow adaptive feature alignment during joint training. Additionally, parameter efficiency could be improved by reducing redundant feature dimensions, as the complementary nature of the two modalities may allow for more compact model representations.

6.6 Conclusion

The results presented in this chapter demonstrate that while a LiDAR-only approach to autonomous navigation is feasible, substantial improvements are still required to achieve consistent and reliable performance. As discussed previously, a more effective method for modelling path discrepancies is needed to improve trajectory accuracy. The lateral shift training experiments showed minimal improvement, which can be attributed to occlusion effects and the inherent radial geometry of the LiDAR point cloud. To address these limitations, a more coherent and physically grounded training dataset is necessary. Constructing a full 3D representation of the driving environment would enable the synthesis of augmented point clouds through realistic data transformations, such as shifts and rotations, that more accurately account for occlusion, viewpoint variations, and geometric distortion.

The investigation into the multi-modal system, which combined the *VMamba* vision model (Chapter 5) with the LiDAR-based *PointMamba* model, demonstrated strong performance, particularly in complex driving environments and landmark navigation. However, the limited performance of the LiDAR component led to an increase in the number of disengagements, indicating that fusion effectiveness remains constrained by the weakest modality.

At this stage, additional refinement is necessary to develop a reliable LiDAR-only fallback system capable of maintaining autonomous operation under visual degradation or sensor failure. However, the fused vision-LiDAR model, with targeted architectural and data-level corrections, has proven viable for lane-following tasks. Before a broader deployment, further validation is required to assess model generalisation across unseen environments, different lighting, and weather conditions.

6.6.1 Novel contributions

This chapter contributes several novel insights into the integration of LiDAR-based perception within autonomous driving frameworks. First, it provides a critical evaluation of LiDAR-only models under real-world conditions, highlighting the discrepancies between synthetic

training environments and their physical counterparts. Second, it demonstrates the practical limitations of current dataset augmentation techniques and the necessity of physically grounded 3D modelling for robust generalisation. Finally, this work reinforces the potential of state space models (SSMs) when applied to point cloud data, illustrating that coherent spatial relationships can be learned directly from LiDAR inputs to support autonomous control behaviours.

6.6.2 Future work

Although current findings define a clear trajectory for future research, several specific directions are worth pursuing. Developing a more robust training and inference framework that incorporates spatial masking could improve the resilience of the model in unfamiliar or partially occluded environments. Furthermore, deploying and testing the system in new real-world contexts, such as the University of Western Australia (UWA) campus, would provide valuable insight into its adaptability. However, the relatively low density and limited coverage of LiDAR data in such environments may constrain performance, underscoring the continued need for dataset enrichment and adaptive fusion strategies. Finally, applying a confidence element to modality inputs that impacts the model decision making could lead to stronger variable models that can account for periods of modality weakness.

Chapter 7

Incorporating Temporal and Feedback Mechanisms in Vision-Based Models for Autonomous Driving on Suburban Roads

7.1 Introduction

As seen in Chapters 5 and 6, many model architectures have been adapted to process sequential data for various tasks like large language model and image processing tasks. The natural ability of transformers in processing sequential data has made them one of the dominant architectures; however, the more streamline SSM like mamba are slowly starting to make an impact with their reduced memory and resource consumption algorithms.

The models so far have been able to achieve good results using only spatial data; however, they have room for improvement, as they are still unable to achieve the same level of driving as a human operator. This chapter introduces new models that combine further information, either from alternative sources such as steering and acceleration, and temporal data such as a sequence of images collected over time.

The objective of this chapter is to evaluate whether temporal and feedback models can provide the additional information required to achieve human-level driving while maintaining low resource consumption on the GPU.

The models are evaluated against existing vision-based end-to-end approaches described in Chapter 5, using the following metrics:

1. **Autonomous driving time** – the percentage of time the model remains in autonomous mode.
2. **Interventions/disengagements** – how often human or hardware interventions are required.
3. **Ride comfort** – assessed by speed transitions and percent change in steering angle over time.
4. **Resource consumption** – assessed by the size of the model on the GPU plus the inference time.

The chapter is organised into five sections:

1. Background on temporal and feedback models.
2. Description of the proposed models used and preliminary testing results.
3. Results from vehicle simulation and real-world driving.
4. Discussion of findings.
5. Conclusion and suggestions for future work.

7.2 Related Works

To contextualise the development of autonomous driving using video data, it is essential to review the progression of video understanding methods. This section presents key milestones in video-based perception, beginning with handcrafted spatiotemporal features and advancing to deep learning architectures, including convolutional, transformer-based, and state space models.

Early research extended image recognition into the temporal domain to model motion and sequential events. One of the earliest notable contributions was made by Schüldt et al. in 2004, who developed a method for human motion recognition using space–time interest points and local feature descriptors tracked across consecutive frames, such as limb movements [237]. This approach demonstrated moderate success in classifying activities such as walking, jogging, running, and various hand gestures.

In 2012, Soomro et al. introduced the UCF101 dataset, which became a widely adopted benchmark for video action recognition [238]. Their method used Harris3D spatiotemporal

features and histogram-based descriptors to generate 'video words', followed by nearest-neighbour classification. Despite achieving an accuracy of only 44.5% across 101 action categories, this work highlighted the challenges of capturing temporal dynamics using handcrafted features.

The introduction of convolutional neural networks (CNNs) marked a turning point. Karpathy et al. demonstrated how CNNs could be extended to video by processing frame sequences for large-scale classification tasks using YouTube video datasets [239]. Whereas earlier studies applied CNNs to smaller datasets such as UCF101 [240], Karpathy's work explored large-scale training and proposed multiple fusion strategies to integrate spatial and temporal information. These included:

- **Early Fusion:** Applying 3D convolutional kernels across spatial and temporal dimensions, $H \times W \times C \times T$.
- **Late Fusion:** Processing initial and final frames independently and combining features at the fully connected stage.
- **Slow Fusion:** Gradually merging temporal information through successive convolutional layers to capture both local and global motion cues.

A two-stream design was also introduced to balance spatial detail and computational efficiency by combining a high resolution central crop with a lower resolution full frame stream (Figure 7.1).

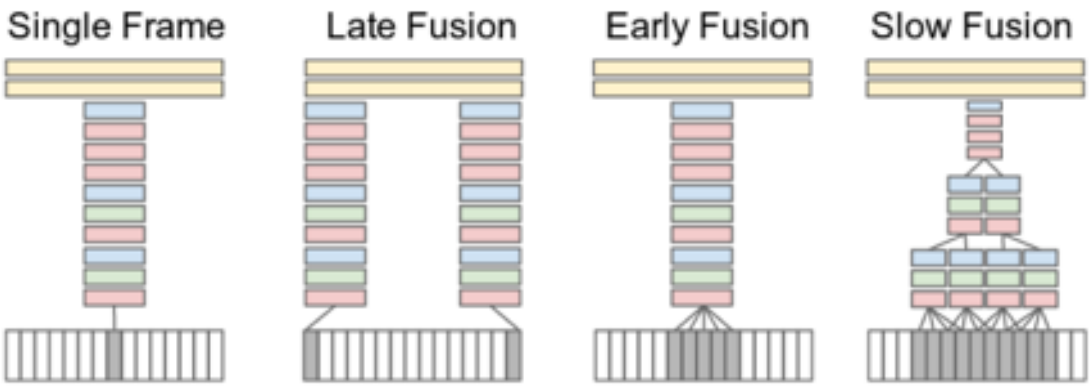


FIGURE 7.1: Spatiotemporal fusion strategies [239].

These approaches laid the foundation for 3D CNNs, ConvLSTMs, and more advanced spatiotemporal architectures that better captured long-range temporal dependencies. Insights from this evolution directly influenced modern transformer- and state-space-based video models used in autonomous driving.

The introduction of self-attention and transformer architectures fundamentally changed visual perception. Building on the Vision Transformer (ViT), Bertasius et al. proposed TimeSformer, the first fully attention-based model for video understanding [241]. TimeSformer tokenises each frame into patches, augments them with temporal and positional embeddings, and processes them using a standard transformer encoder to jointly learn spatial and temporal dependencies. Evaluated on benchmarks such as Kinetics-400/600 [242, 243], Something-Something V2 [244], and Diving48 [245], TimeSformer surpassed prior convolutional approaches. However, its joint space–time attention introduced substantial computational and memory overhead, particularly for longer frame sequences.

To alleviate these issues, Arnab et al. developed ViViT, which factorises attention into separate spatial and temporal components via divided multi-head attention [246]. This reduced computational cost and mitigated out-of-memory errors while preserving state-of-the-art performance.

For this thesis, the VideoMAE architecture is selected due to its superior generalisation and data efficiency. VideoMAE employs masked autoencoding to learn robust spatiotemporal representations without requiring extensive pretraining data. A detailed explanation of VideoMAE is presented in the subsequent section.

While transformers excel at modelling global dependencies, their memory and time complexity scales quadratically with sequence length, limiting deployment in resource-constrained systems. State space models (SSMs), originally rooted in control theory, provide a more computationally efficient alternative by modelling hidden states through linear recurrence relations. In 2023, Islam and Bertasius introduced ViS4mer, a hybrid architecture combining transformer-based spatial encoders with an S4 state space decoder to capture temporal dependencies efficiently [247]. ViS4mer achieved competitive accuracy on long-range video datasets such as Long Video Understanding [248], Breakfast [249], and COIN [250], using fewer pretraining samples than transformer-only models. This demonstrated the potential of SSMs to balance accuracy and scalability, an important property for autonomous driving, where frame rate, vehicle dynamics, and temporal consistency affect predictions.

Following advances in S4, the Mamba architecture introduced a selective state space mechanism capable of learning input-dependent transition matrices while retaining linear-time complexity [146, 251]. In 2024, researchers adapted Vision Mamba (ViM) for video input by providing short frame sequences to the model and demonstrated early results on film-related datasets [252, 253]. The publicly available implementation from the Chinese team forms the basis of one of the models evaluated in this chapter.

Beyond video-only models, several studies have integrated auxiliary modalities such as vehicle state, control commands, or past outputs to improve prediction accuracy. Xu et al. proposed a motion prediction framework incorporating both image sequences and previous motion outputs, achieving 72.4% accuracy on a custom crowd-sourced dataset [254]. Similarly, Codevilla et al. introduced two conditional imitation learning models that processes camera images, vehicle state measurements, and high-level driving commands to predict control actions in CARLA and small-scale real-world environments [152]. Both models showed good levels of success achieving rates of 88% and 78% on driving from a random location to a destination point. These works demonstrate that incorporating temporal or feedback signals can significantly enhance driving performance. However, most experiments remain constrained to simulation or controlled environments, limiting real-world applicability.

The progression from handcrafted features to CNNs, transformers, and state space models has significantly improved the ability of systems to understand and predict motion in video data. However, most prior work is either computationally intensive or limited to idealised environments. This thesis seeks to bridge this gap by evaluating transformer and SSM-based models on realistic driving data, demonstrating that both models can effectively integrate video and vehicle state information to improve end-to-end driving performance.

7.3 Models and Initial Training

Two strategies were explored to incorporate temporal information into the AI model. The first approach concatenates multiple frames into a single composite image, which is then processed as a unified input. In this configuration, spatial and temporal information is integrated simultaneously, allowing the model to construct a contextual understanding that captures both dimensions jointly.

The second approach processes each frame individually, first extracting spatial feature maps from each image. These feature maps are subsequently passed through a sequential model that explicitly models temporal dependencies. By separating spatial and temporal processing, this method allows the model to learn spatial features independently before integrating temporal information to make predictions.

In addition to temporal modelling, a feedback mechanism was implemented to improve control predictions. This mechanism incorporates the vehicle's speed and steering outputs from the previous time step to inform the next control action, analogous to how a human

driver considers the prior accelerator and steering positions when responding to new visual stimuli.

This section presents the four temporal models: two that jointly model spatial and temporal relations and two that model them sequentially, along with a single feedback-enhanced model.

7.3.1 VideoMamba

Li et al. extended the Vision Mamba (ViM) architecture to handle video sequence data for video comprehension tasks [252]. Similar to prior sequence-based vision models, the input video frames are first divided into non-overlapping patches. Each patch is embedded with a classification token, a spatial patch embedding, and a temporal patch embedding to jointly capture spatial and temporal dependencies across frames. The authors investigated several bidirectional scanning mechanisms, as illustrated in Figure 7.2, and found that the *spatial-first* approach yielded the strongest performance.

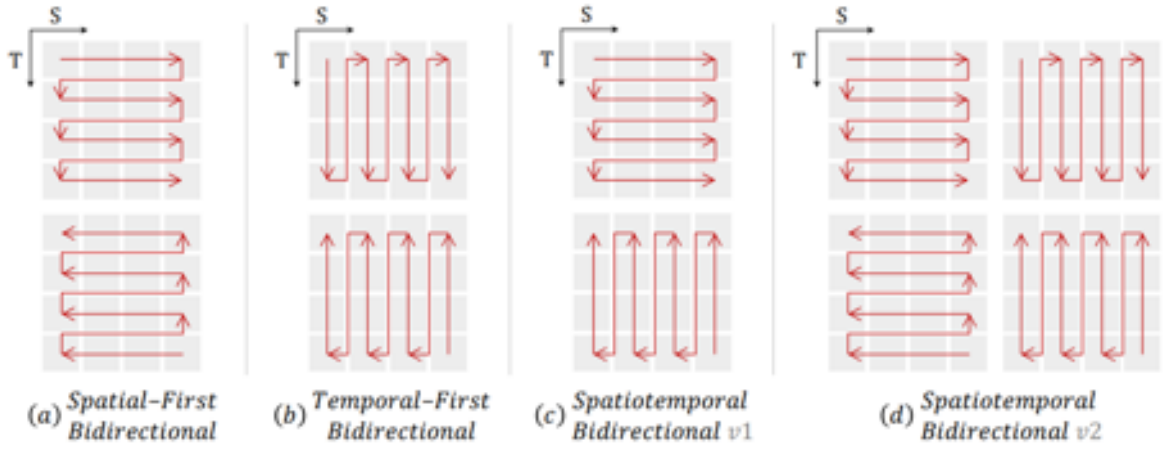
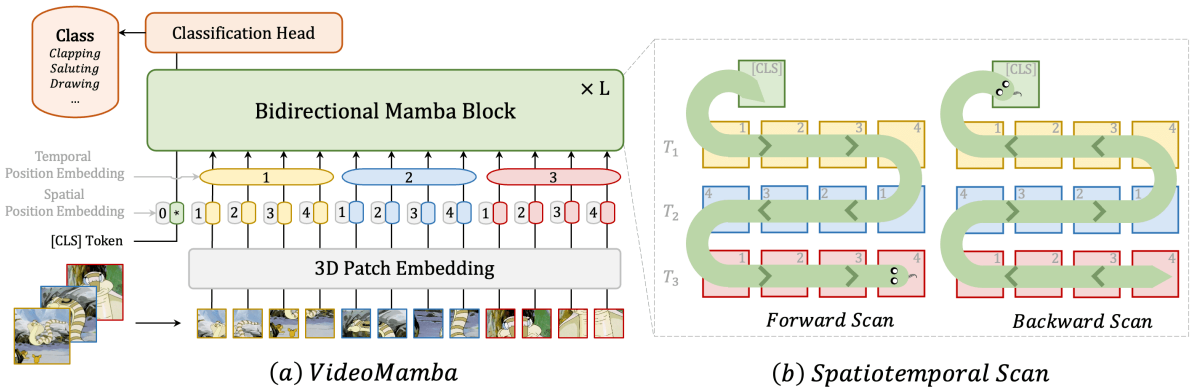


FIGURE 7.2: VideoMamba scanning mechanisms [252].

Their study further demonstrated that the original ViM model could be optimised by removing the complex middle classification token and rotary position embeddings, thereby reducing model complexity. Additionally, self-distillation was introduced to enhance generalisation and help with overfitting issues, and pretraining strategies such as masking were employed to further improve performance. In this work, however, pretraining and distillation were deliberately avoided to ensure all models were evaluated under comparable conditions. The final architecture developed by Li et al. is shown in Figure 7.3.



(a) *VideoMamba* (b) *Spatiotemporal Scan*

FIGURE 7.3: Final VideoMamba framework [252].

VideoMamba represents a spatiotemporal model capable of learning rich temporal dynamics from sequences of visual inputs, enabling stronger contextual understanding compared to single-frame models. The implementation used in this work was adapted from the official repository for single-modality video data [252]. Owing to its linear time complexity and robust spatiotemporal representations, it is expected that VideoMamba will demonstrate improved generalisation and more stable vehicle control performance compared to frame-based architectures with a small increase to model size.

7.3.2 VideoMAE

The second model is based on VideoMAE, a transformer-based video understanding framework [255]. It extends ImageMAE [256], which itself builds upon the Vision Transformer (ViT) architecture.

In the ImageMAE framework, an image is divided into a series of non-overlapping patches, a high proportion of which are randomly masked. The visible patches are then passed through an autoencoder structure that learns to extract key semantic features before reconstructing the original image. This masked reconstruction process encourages the encoder to develop generalisable feature representations from incomplete visual information. Once pretrained, the encoder can be reused for downstream tasks such as image classification by attaching a task-specific head. An example of the masked autoencoder structure is shown in Figure 7.4.

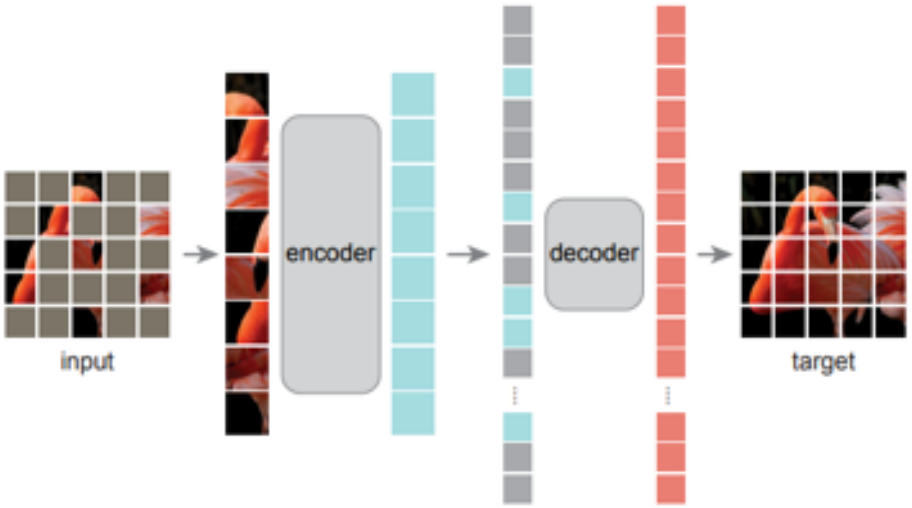


FIGURE 7.4: Masked autoencoder structure [256].

VideoMAE extends this concept to operate over temporal sequences of images. Unlike static images, videos contain both spatial and temporal information, so the masking process must also account for dependencies across time. As explained in [255], directly applying random masks to each frame would result in information leakage between consecutive frames, as unmasked regions in one frame could reveal masked regions in another. To address this, Tong *et al.* introduced a tube masking strategy, illustrated in Figure 7.5. In this approach, the same spatial mask is applied consistently across the temporal axis, preventing information leakage while preserving temporal coherence.

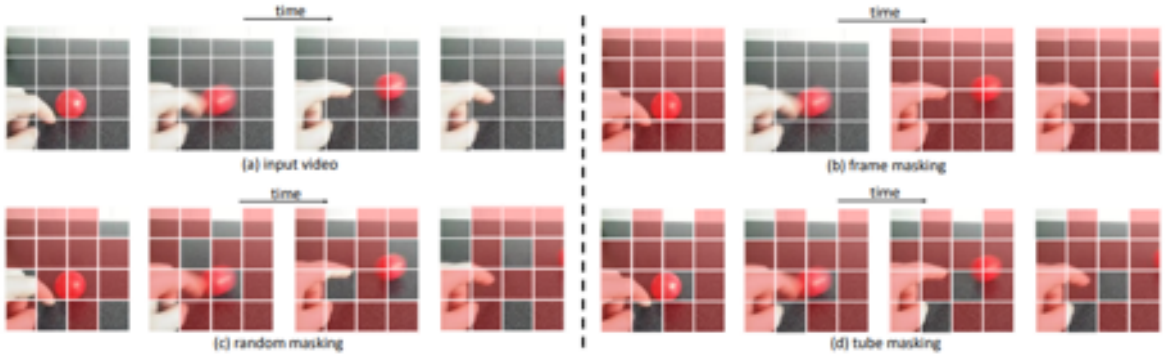


FIGURE 7.5: Sample of possible masking strategies [255].

Additionally, the model incorporates a tubelet embedding mechanism, where the video input is partitioned into spatiotemporal patches of size $T \times H \times W$, with T representing a subset of the frames, and H and W representing spatial dimensions. This reduces the overall input dimensionality and leverages spatiotemporal redundancies, particularly effective when processing high-frame-rate video with slow-moving obstacles.

An improved version, VideoMAE V2, was proposed by Wang *et al.* [257] to enhance training efficiency and reconstruction quality for larger models. The updated approach introduces a dual masking strategy, where an additional masking stage is applied before decoding. This encourages the model to reconstruct from more compressed latent representations, thereby improving generalisation. The dual masking process is shown in Figure 7.6.

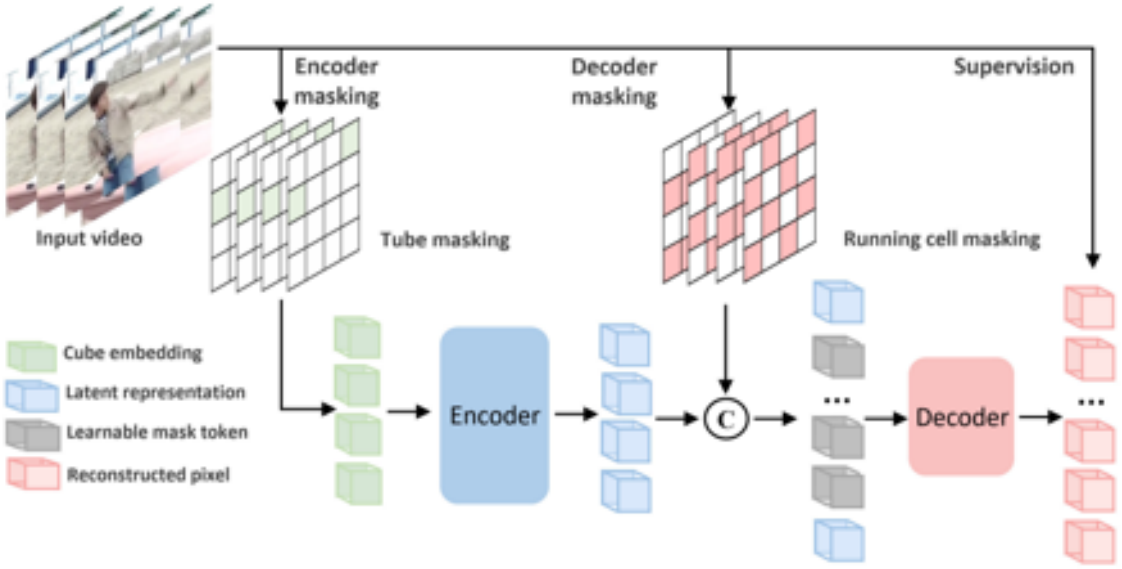


FIGURE 7.6: Dual masking strategy used in the VideoMAE V2 model [257].

In this work, all models are trained from scratch to ensure a consistent comparison framework; therefore, the VideoMAE model was trained without large-scale pretraining. It is important to note that VideoMAE typically relies on extensive pretraining to capture robust spatiotemporal representations, and removing this step is expected to limit its performance in this context. However, doing so allows for a fair evaluation across all architectures under the same data and compute constraints. Furthermore, due to the high computational cost of VideoMAE training, pretraining was omitted in favour of a direct end-to-end learning approach.

The implementation used in this work is based on the publicly available github repository for videoMAEv2 [257]. While configured as a spatiotemporal Vision Transformer with tubelet embeddings rather than a full masked autoencoder, this setup can be extended in future work to incorporate pretraining and reconstruction for improved temporal reasoning.

7.3.3 Sequence Models

The previous models combine multiple images into a single sequence before passing the sequence into the model, which jointly learns both spatial and temporal relationships at the

same time. However, this approach can bias the model toward learning temporal dependencies at the expense of spatial context for SSMs, depending on the size and capacity of the state representation, or computational and memory efficiency for transformer models.

An alternative approach is proposed here: to reuse the trained image models from Chapter 5 to analyse each input frame independently. Each image is processed by the pre-trained model to generate its own spatial feature map, resulting in eight separate feature representations for an eight-frame sequence. These feature maps are then combined in a early fusion approach and passed into a sequential model that explicitly captures temporal dependencies between frames, using the key spatial features extracted earlier. The overall architecture for this approach is shown in Figure 7.7.

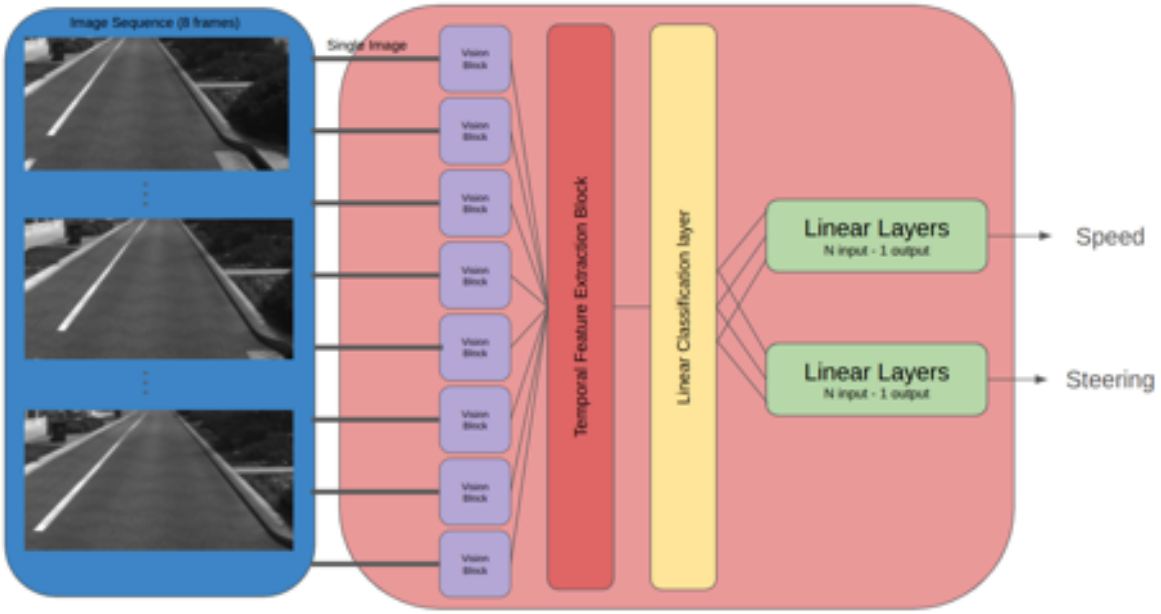


FIGURE 7.7: Sequence-based model used to evaluate spatial and temporal relationships separately.

In this design, the previously trained image models are loaded with their weights frozen, under the assumption that they already extract robust spatial features. The resulting feature maps from each frame are concatenated and projected to a lower-dimensional representation to reduce computational complexity. This compacted sequence is then processed through a final VMamba block, which models the temporal relationships between consecutive frames using the cross-sectional scan technique described in Chapter 5.

The extracted temporal features are subsequently passed through a dual linear regression behaviour model, from Chapter 5, to produce steering and speed outputs. Initially, inference was slow due to the sequential processing of each spatial frame. To address this, the

implementation was modified to process all frames in parallel as a batch. This parallelisation allowed spatial feature extraction to occur simultaneously across frames, significantly improving inference speed while maintaining model accuracy.

7.3.4 VMamba Feedback

This model extends the standard VMamba configuration described in Chapter 5, designed to test whether incorporating control-related metadata from the previous frame can enhance driving performance. Rather than introducing temporal sequence modelling, this approach provides additional contextual input by feeding the prior frame's steering angle, vehicle speed, and related control parameters into the model. The intuition follows that, similar to how human drivers make adjustments based on their current steering and throttle positions rather than purely on visual input, the model may benefit from awareness of its most recent control state.

Using VMamba as the image processing backbone, these previous-frame parameters are passed through several multilayer perceptron (MLP) layers to expand their dimensionality, ensuring balanced feature scaling relative to the vision features. The outputs of the vision encoder and the MLP layers are then concatenated and passed into the dual-branch regression head, as previously described in Chapter 5. This allows the model to retain awareness of recent control states without the computational overhead of full temporal modelling.

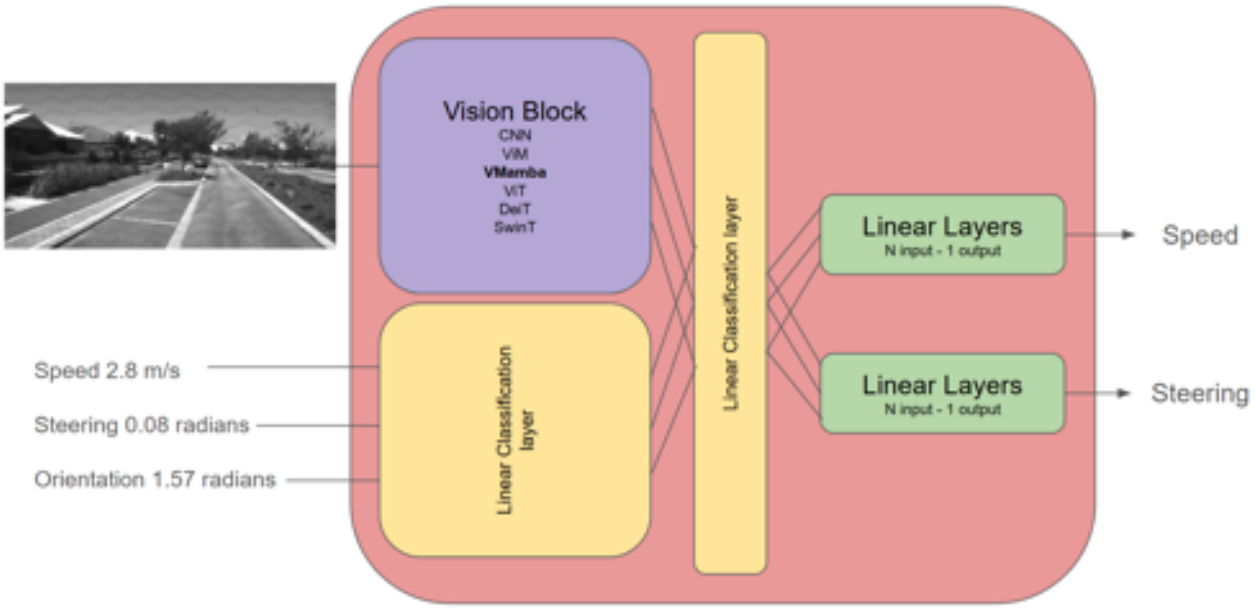


FIGURE 7.8: Feedback model architecture incorporating previous-frame speed, steering and orientation parameters and VMamba vision model.

7.3.5 Testing

The VideoMAE and VideoMamba models were tested within the EyeSim simulator to verify their ability to perform autonomous driving tasks. It was considered unnecessary to test the Feedback and Sequence models in EyeSim, as their corresponding base architectures had already demonstrated functionality both in simulation and on-road environments. The key parameters used for the EyeSim experiments are shown in Table 7.1. Any parameters not listed were retained from the original model configurations.

Parameter	**VideoMAE**	**VideoMamba**
Image size	(64, 192)	(66, 200)
Patch size	(8, 16)	4
Embedding dimension	756	24
Features extracted	200	100
Dropout rate	0.1	0.1
Frame size	8	8
Depth	8	2
Channels	3	3
d_{state}	–	20
BiMamba	–	True
MLP layers	–	2
MLP ratio	4.0	–
Attention heads	8	–
Tubelet size	2	–

TABLE 7.1: Model parameters used for EyeSim testing.

As shown in Table 7.1, the VideoMAE model required a significantly larger number of parameters to learn this relatively simple driving task. This can be attributed to the lack of pretraining and the limited size of the data set, which hinders its ability to form robust spatiotemporal representations. Despite these limitations, the model was still able to complete several laps of the EyeSim track, though with reduced stability and control consistency.

In contrast, the VideoMamba model demonstrated more stable performance, successfully completing multiple autonomous laps with fewer parameters. Its compact architecture and state-space formulation allowed it to generalise effectively from limited training data. An

example of the autonomous operation of the VideoMamba model in EyeSim is shown in Figure 7.9.

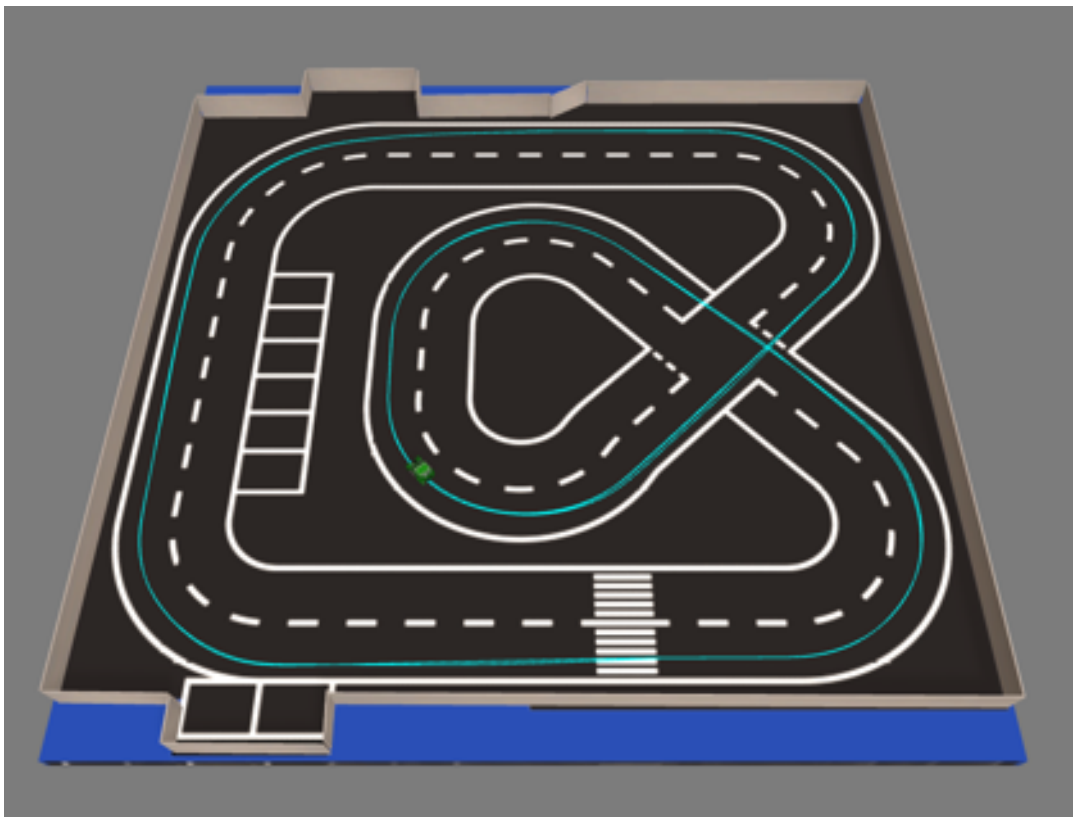
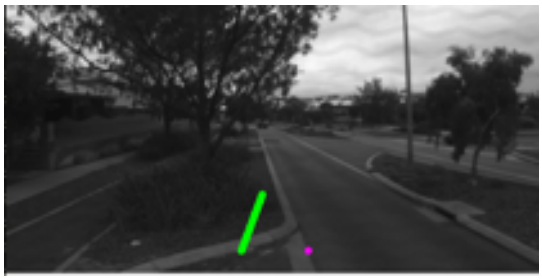


FIGURE 7.9: VideoMamba model driving autonomously in the EyeSim simulator.

Testing of the Feedback model showed that it always replicated the previous input to the model no matter what the image input was. This result is shown in figure 7.10. This is because while training the model learns that it can achieve the best results if it just replicates the existing output. At this point the model was put to the side for future work but it is possible that a residual model may be better where the system looks to use the image as an update to the existing output commands rather than predict new values based on them.



(A) Feedback model with feedback set to 0.

(B) Feedback model with previous command.

FIGURE 7.10: Feedback model constantly outputting the previous models output, Ground truth (green) vs Model Output (purple).

7.4 Results

Following completion of simulation-based testing, all models were retrained using real-world driving data collected from the Eglinton route. The training configuration was standardised across all architectures, employing grey scale images with resolution (180×400), a single input channel, and a sequence of eight temporal frames. The model-specific architectural parameters used during training are summarised in Table 7.2.

Parameter	VideoMAE	VideoMamba	ViM Sequence	VMamba Sequence
Patch size	(18, 20)	(18, 40)	(9, 20)	(9, 20)
Embedding dimensions	512	512	512	[256, 512]
Features extracted	1000	1024	1024	1024
Dropout rate	0.1	0.1	0.1	0.2
Depth	6	8	8	[3, 4]
d_{state}	-	16	16	32
MLP ratio	3.0	-	-	-
Attention heads	8	-	-	-
Tubelet size	2	-	-	-

TABLE 7.2: Model parameters for real-world Eglinton driving experiments.

Table 7.3 presents the model sizes, inference times, and parameter counts when deployed on the shuttle bus platform. Although the sequence-based models exhibit inference times marginally above the target threshold of 0.05 s, the increase is minimal and not expected to degrade real-time performance. However, this improved temporal reasoning is accompanied by a substantial increase in model complexity, particularly for the VMamba Sequence model, which may present challenges in resource-constrained embedded environments.

Model	File size (MB)	Inference (s)	Parameters (10^6)
VideoMamba	202.7	0.031	16.87
VideoMAE	212.3	0.044	17.96
ViM Sequence	460.3	0.058	79.62
VMamba Sequence	1100	0.069	235.28

TABLE 7.3: Model size, inference time, and parameter count for real world deployment.

The training and validation losses for each driving behaviour are reported in Table 7.4. The two sequential architectures that jointly incorporate spatial and temporal information

demonstrated worse loss values than the models that evaluate the spatial and temporal components separately. This effect is particularly evident for VideoMAE, which exhibited higher losses, likely due to the absence of large-scale pretraining and the relatively small dataset. In contrast, ViM Sequence and VMamba Sequence achieved comparable performance to image-based models.

	Training Time (min)			Training Loss			Validation Loss		
Model	**Lane**	**Pull-in**	**Reverse**	**Lane**	**Pull-in**	**Reverse**	**Lane**	**Pull-in**	**Reverse**
VideoMamba	**148.98**	**57.78**	**36.6**	0.0031	0.0022	0.0014	0.0032	0.0022	0.0015
VideoMAE	209.64	90.00	48.18	0.0168	0.0104	0.0065	0.0168	0.0104	0.0064
ViM Sequence	229.2	96.90	53.64	0.0021	0.0014	0.0009	0.0022	**0.0005**	0.0009
VMamba Sequence	191.64	88.20	49.63	**0.0012**	**0.0006**	**0.0006**	**0.0012**	0.0006	**0.0006**

TABLE 7.4: Training duration and loss values for each driving behaviour.

As in Chapter 5, the speed and steering accuracy for each model behaviour are presented in Table 7.5, separated into training and validation results. Training accuracy is defined as predictions within 5% of the ground-truth values, while validation accuracy is measured within 10% to provide an indication of output variance and generalisation performance.

Examination of the results reveals substantial variance between training and validation performance for most models, with the exception of VMamba. This effect is particularly pronounced for VideoMAE, which may be attributed to the absence of masked autoencoder pretraining, resulting in behaviour more closely resembling a video-based Vision Transformer (ViT) rather than a true masked autoencoder architecture.

Compared with the single-image models presented in Chapter 5, performance degradation is observed for VideoMamba relative to its base architecture, VMamba. Similarly, VideoMAE demonstrates reduced performance compared to ViT for lane following speed and steering predictions, although improvements are observed for the pullin and reversing tasks. In contrast, both the ViM and VMamba sequence-based models exhibit improved accuracy across all evaluated behaviours.

These differences are attributed to the way that the respective architectures process temporal and spatial information. VideoMamba and VideoMAE model spatial and temporal features jointly, whereas the ViM and VMamba sequence models first extract spatial features from individual frames before modelling temporal relationships. By identifying salient spatial features prior to temporal aggregation, the latter approach appears to be better suited to

capture scenario dynamics, as the simultaneous processing of spatial and temporal information may hinder the model's ability to retain selectively important features over time.

	Speed Accuracy (%)						Steering Accuracy (%)					
	Training			Validation			Training			Validation		
Model	**Lane**	**Pull-in**	**Reverse**	**Lane**	**Pull-in**	**Reverse**	**Lane**	**Pull-in**	**Reverse**	**Lane**	**Pull-in**	**Reverse**
VideoMamba	87.19	93.52	**99.95**	95.42	98.62	**99.98**	85.34	86.21	99.49	95.74	95.48	99.87
VideoMAE	65.20	82.28	80.83	81.87	94.07	96.37	73.99	75.22	95.52	89.68	88.70	99.56
ViM Sequence	90.65	97.71	99.17	97.35	99.73	99.88	88.24	92.75	99.87	97.25	98.74	99.97
VMamba Sequence	**95.21**	**99.12**	99.29	**98.62**	**99.91**	99.92	**93.94**	**97.48**	**99.96**	**98.68**	**99.58**	**99.99**

TABLE 7.5: Training and validation accuracy for speed and steering across lane following, pull-in, and reverse tasks.

As in the previous chapters, model performance was also evaluated using previously unseen data through analysis of histograms and statistical summaries. The histograms are presented in Figures 7.11–7.16, with the corresponding statistical results shown in Tables 7.6–7.11.

The results indicate that the models experienced greater difficulty accurately predicting speed outputs during the lane-following and reversing tasks. In contrast, the pull-in task demonstrates stronger speed control but exhibits a wider spread in steering predictions. Consistent with observations from earlier analyses, architectures that jointly process temporal and spatial information appear to exhibit reduced performance, whereas models that separate spatial feature extraction from temporal modelling achieve improved accuracy. This trend is reflected in the mean, median and standard deviation metrics compared to their single-image counterparts.

Overall, the best-performing sequence-based models achieve performance comparable to, or only marginally better than, the single-image models. Based on these results, the additional computational and implementation overhead associated with video-based approaches may not be justified, as comparable performance improvements could potentially be achieved through alternative system enhancements.

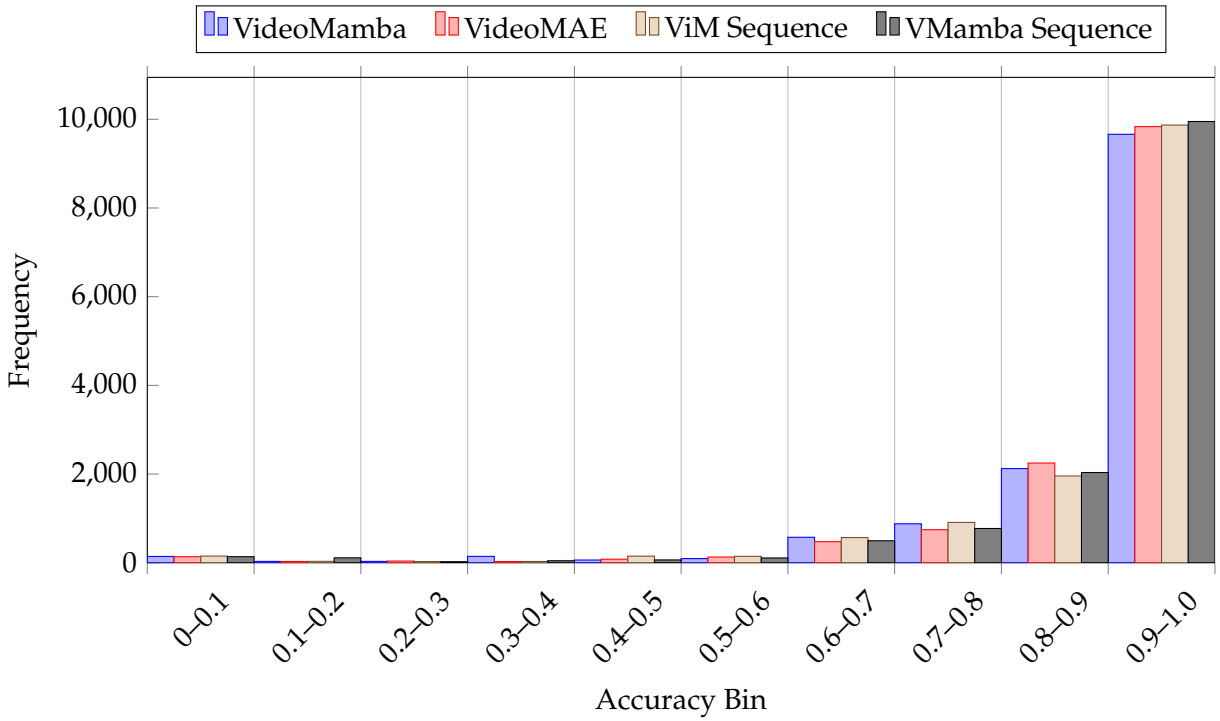


FIGURE 7.11: Lane following bin frequencies for five models - speed.

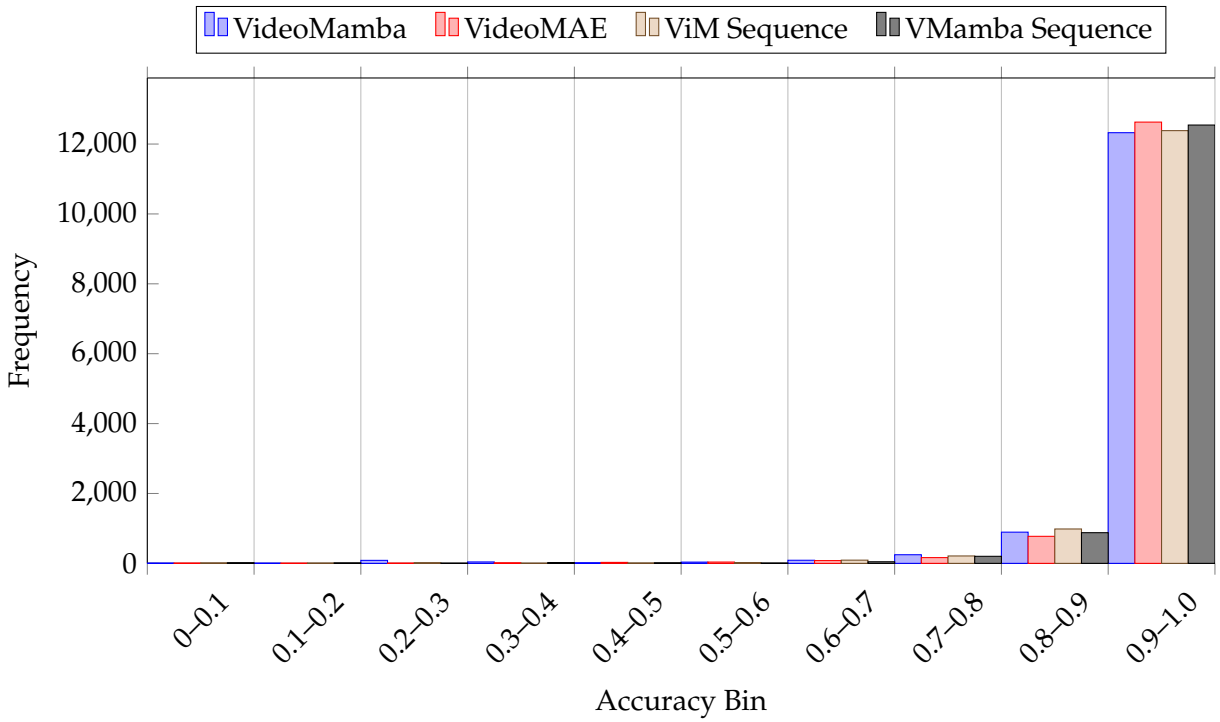


FIGURE 7.12: Lane following bin frequencies for five models - steering.

Speed Error (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
Video Mamba	30.11	51.95	68.34	77.24	84.08	87.81	90.43	93.71	94.30	5.15
VideoMAE	29.17	55.13	72.67	81.53	87.14	90.30	92.04	84.74	95.71	3.70
ViM Seq.	34.38	54.51	68.93	77.92	83.93	87.58	90.09	93.39	94.43	4.71
VMamba Seq.	34.52	55.16	69.90	79.56	85.09	88.90	90.98	93.82	94.46	4.78
Steering Error (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
Video Mamba	58.06	82.76	90.00	93.38	95.26	96.48	97.15	97.57	97.83	1.96
VideoMAE	60.81	86.69	93.42	95.83	97.02	97.88	98.52	98.89	99.17	0.58
ViM Seq.	57.32	83.28	91.93	95.57	97.27	98.19	99.18	99.40	99.52	0.39
VMamba Seq.	60.86	85.47	93.03	96.30	98.00	98.69	99.05	99.25	99.35	0.64

TABLE 7.6: CDF Lane following Error Comparison for Speed and Steering

Speed Error					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
Video Mamba	14.48	0.00	99.96	9.47	18.09
VideoMAE	**13.20**	0.00	**99.83**	8.88	**16.55**
ViM Seq.	13.89	0.00	99.96	**8.70**	17.81
VMamba Seq.	13.83	0.00	99.99	**8.70**	18.73
Steering Error					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
Video Mamba	6.99	0.00	100.00	3.59	11.29
VideoMAE	5.60	0.00	100.00	3.35	7.87
ViM Seq.	6.00	0.00	100.00	3.67	**7.69**
VMamba Seq.	**5.51**	0.00	100.00	**3.07**	8.09

TABLE 7.7: Summary of Lane Following Model Error for Speed and Steering

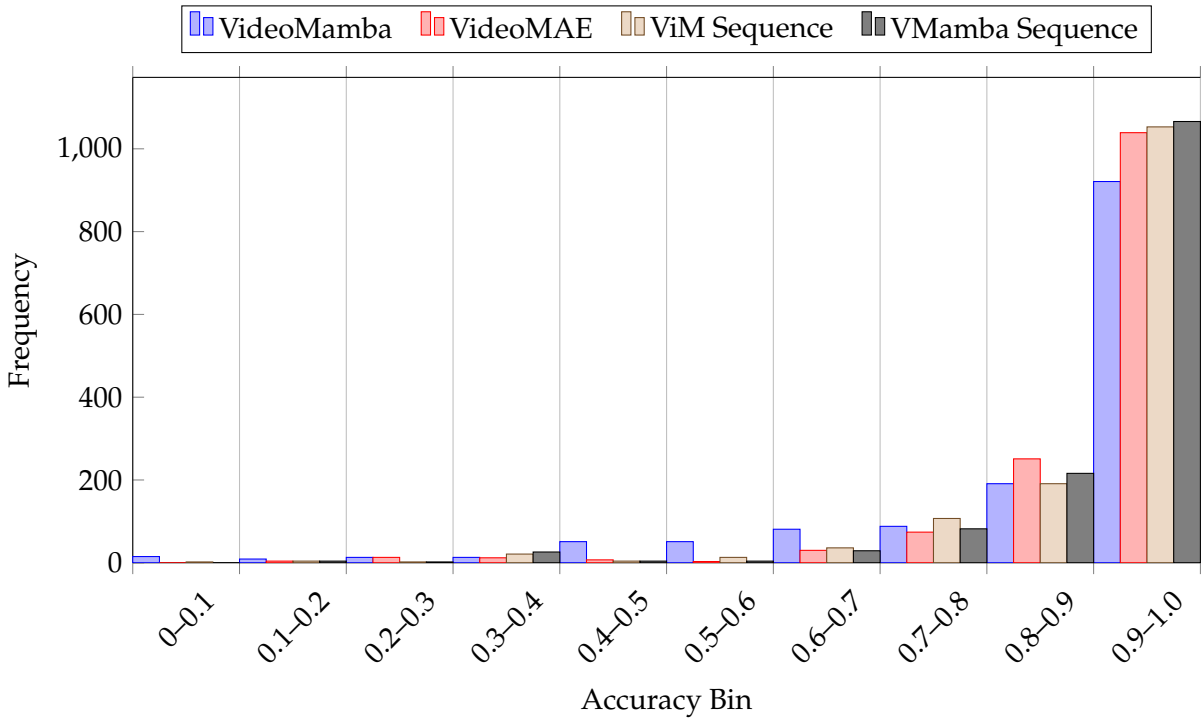


FIGURE 7.13: Pullin bin frequencies for five models - speed.

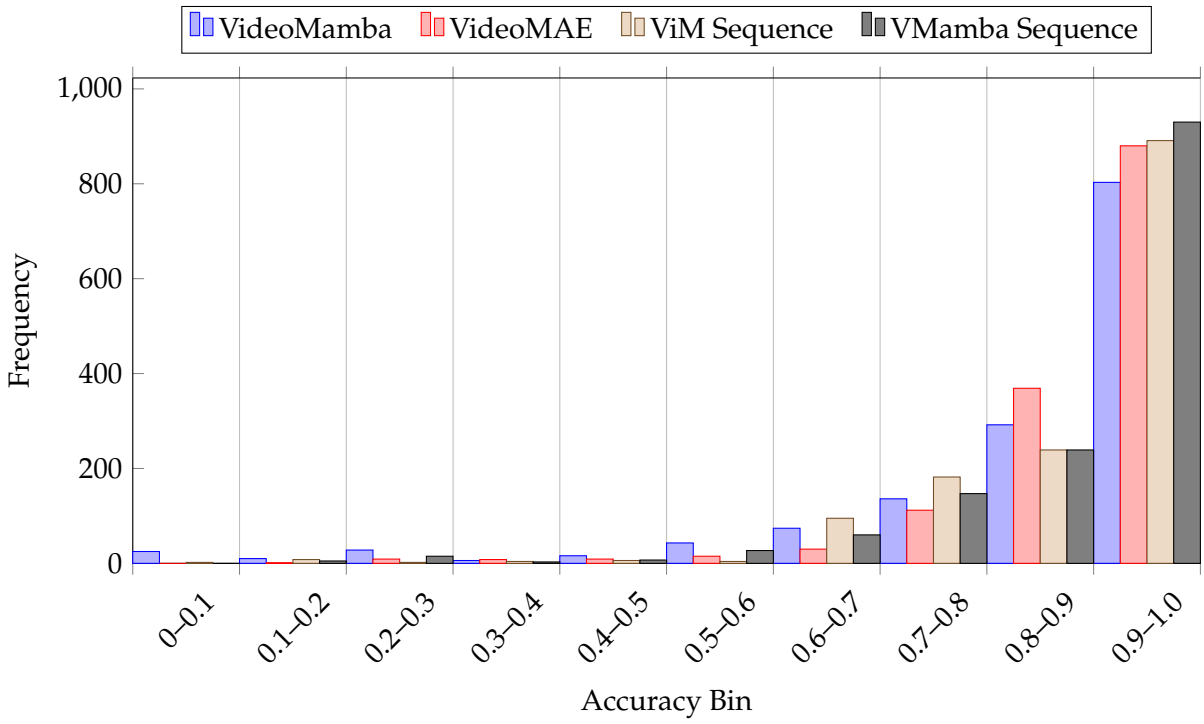


FIGURE 7.14: Pullin bin frequencies for five models - steering.

Speed Accuracy (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
Video Mamba	49.90	64.27	73.48	77.60	81.30	83.74	86.32	89.39	91.56	7.05
VideoMAE	49.41	72.51	84.58	90.02	93.09	95.18	96.37	97.28	97.49	2.51
ViM Seq.	52.27	73.48	82.62	86.81	91.49	94.28	95.74	96.79	97.49	2.30
VMamba Seq.	56.87	74.39	83.18	89.46	92.95	95.18	96.30	97.21	97.49	2.51
Steering Accuracy (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
Video Mamba	36.22	56.04	67.90	76.41	81.09	85.90	87.65	91.07	93.16	5.93
VideoMAE	36.22	61.41	76.55	87.16	92.67	94.98	96.16	97.07	97.77	1.88
ViM Seq.	39.64	62.18	72.23	78.86	85.48	91.56	95.46	98.19	98.46	1.54
VMamba Seq.	50.66	64.90	74.04	81.58	87.72	91.84	94.63	96.02	96.79	2.09

TABLE 7.8: CDF Pullin Accuracy Comparison for Speed and Steering

Speed Accuracy					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
Video Mamba	13.73	0.00	100.00	5.08	19.25
VideoMAE	8.93	0.00	89.60	5.13	**12.24**
ViM Seq.	9.07	0.001	94.34	4.63	12.78
VMamba Seq.	**8.18**	0.00	**89.55**	**3.45**	12.34
Steering Accuracy					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
Video Mamba	15.42	0.00	100.00	8.31	19.79
VideoMAE	**10.68**	0.00	**80.68**	7.26	**11.30**
ViM Seq.	11.59	0.00	90.11	6.84	12.63
VMamba Seq.	10.73	0.01	87.03	**4.92**	13.83

TABLE 7.9: Summary of Pullin Model Accuracy for Speed and Steering

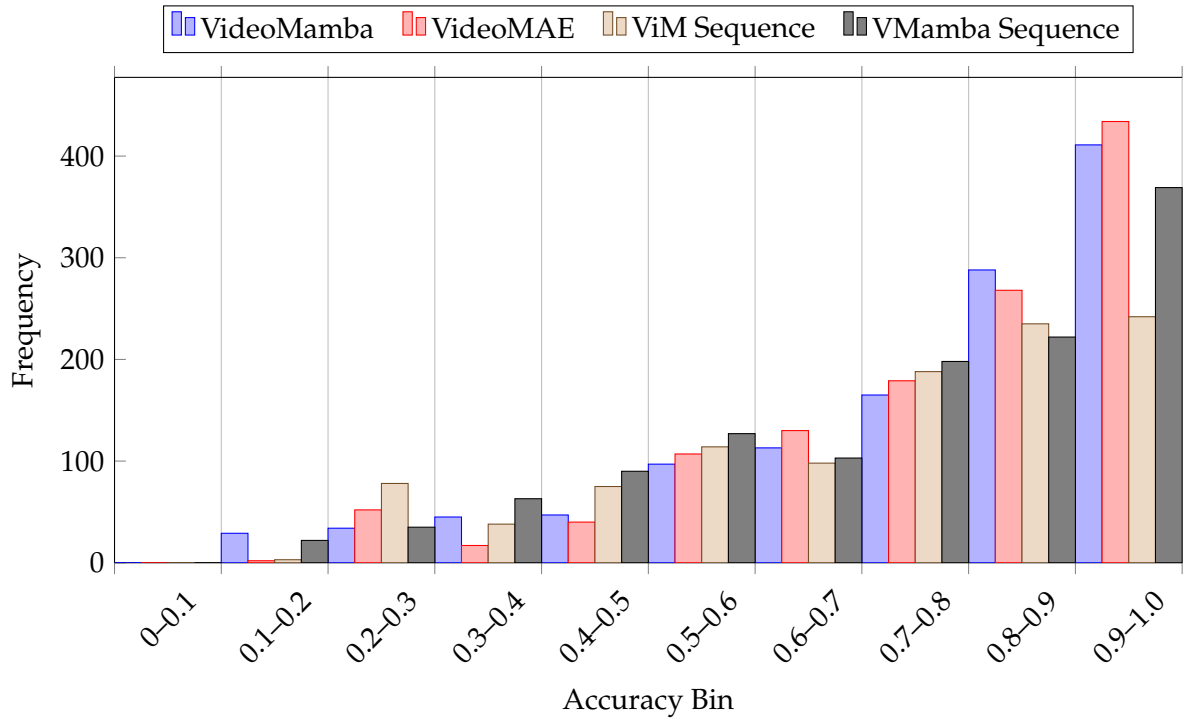


FIGURE 7.15: Reverse bin frequencies for five models - speed.

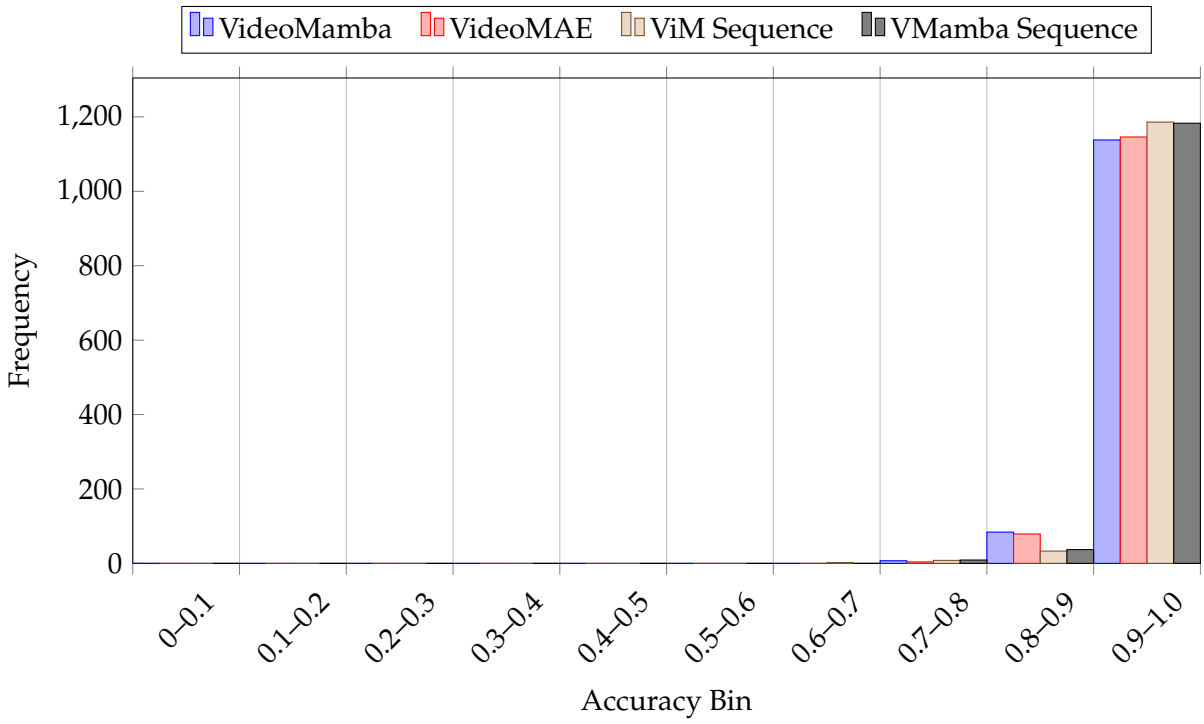


FIGURE 7.16: Reverse bin frequencies for five models - steering.

Speed Accuracy (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
Video Mamba	14.97	33.44	47.44	56.88	64.93	70.30	77.30	79.50	83.16	12.61
VideoMAE	22.70	35.31	46.95	57.12	65.74	71.68	75.67	82.26	88.61	9.03
ViM Seq.	19.69	32.55	42.23	51.67	62.25	66.97	69.32	74.94	78.68	15.79
VMamba Seq.	17.41	30.02	40.11	48.09	58.26	64.20	67.45	72.58	77.14	17.09
Steering Accuracy (CDF)										
Model	**<5%**	**<10%**	**<15%**	**<20%**	**<25%**	**<30%**	**<35%**	**<40%**	**<45%**	**>50%**
Video Mamba	78.19	92.60	97.64	99.43	99.67	100.00	100.00	100.00	100.00	0.00
VideoMAE	76.24	93.25	97.40	99.67	99.92	100.00	100.00	100.00	100.00	0.00
ViM Seq.	86.33	96.50	98.94	99.19	99.51	99.84	100.00	100.00	100.00	0.00
VMamba Seq.	86.82	96.26	98.21	99.27	99.51	100.00	100.00	100.00	100.00	0.00

TABLE 7.10: CDF Reverse Accuracy Comparison for Speed and Steering

Speed Accuracy					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
Video Mamba	23.71	0.01	84.76	**15.93**	21.08
VideoMAE	**21.90**	0.00	81.02	16.51	**19.76**
ViM Sequence	25.15	0.00	**80.70**	18.98	22.15
VMamba Sequence	26.43	0.11	83.63	21.17	21.94
Steering Accuracy					
Model	**Mean**	**Min**	**Max**	**Median**	**Std Dev**
Video Mamba	3.79	0.00	27.38	2.63	3.84
VideoMAE	3.65	0.00	**26.38**	2.55	3.76
ViM Sequence	2.96	0.00	31.86	2.13	**3.42**
VMamba Sequence	**2.73**	0.00	28.46	**2.01**	3.49

TABLE 7.11: Summary of Reverse Model Accuracy for Speed and Steering

7.4.1 Saliency Maps and Ground Truth Comparison

The saliency map and ground truth comparison methods described in Chapter 5 are repeated to verify the models before on road testing. The results demonstrated that the models are able to focus on generalised features and align well with the ground-truth data.

7.4.2 Driving Performance — Overview

The driving performance of each model is summarised in the following three tables. Table 7.16 presents the raw on-road results for the best performing runs of each model.

To ensure a fair comparison, Table 7.13 removes periods when the vehicle's speed was approximately zero. These instances typically correspond to stationary moments, such as waiting in traffic or parked, and disproportionately reduce the calculated autonomy time depending on external conditions.

Model	videoMAE	VideoMamba	Sequence ViM	Sequence VMamba	VMamba
Drive time (min)	28.72	27.42 ± 1.29	29.24	27.25 ± 1.15	24.03 ± 2.23
Time autonomy (%)	65.04	73.89 ± 2.21	87.50	89.01 ± 4.33	84.76 ± 5.40
Disengagements	8	7.33 ± 3.79	15	9.67 ± 2.08	1.00 ± 0.00
Interventions	29	19 ± 4.36	18	14.33 ± 3.21	14.33 ± 4.16

TABLE 7.12: Raw driving performance results for each model.

Model	videoMAE	VideoMamba	Frozen ViM	Frozen VMamba	VMamba
Drive time (min)	22.26	21.61 ± 0.66	23.55	23.36 ± 0.62	19.40 ± 1.76
Time autonomy (%)	78.78	86.07 ± 3.28	94.17	92.60 ± 2.49	90.77 ± 3.62
Disengagements	8	7.33 ± 3.79	15	9.67 ± 2.08	1.00 ± 0.00
Interventions	29	19 ± 4.36	18	14.33 ± 3.21	14.33 ± 4.16

TABLE 7.13: Driving performance excluding low-speed (stationary) segments.

Finally, Table 7.14 presents adjusted results, removing disengagements and interventions that occurred in untrained or out-of-distribution scenarios. This adjustment provides a more equitable comparison by normalising for differences in model exposure to unfamiliar conditions.

Model	videoMAE	VideoMamba	Frozen ViM	Frozen VMamba	VMamba
Drive time (min)	21.72	20.76 ± 1.02	23.29	22.83 ± 0.64	$\mathbf{18.42 \pm 1.46}$
Time autonomy (%)	80.60	89.05 ± 4.46	95.07	94.79 ± 4.33	$\mathbf{95.77 \pm 2.73}$
Disengagements	7	5.67 ± 4.16	13	7.00 ± 2.00	$\mathbf{0.67 \pm 0.58}$
Interventions	22	8.67 ± 3.51	7	6.33 ± 0.57	$\mathbf{5.33 \pm 1.53}$

TABLE 7.14: Final performance metrics for models tested on suburban roads. Results are adjusted for duplicate counts (disengagements and interventions) and excluding untrained scenarios (mean $\pm$ standard deviation).

When compared to the single-frame image model, VMamba, none of the temporal models matched its overall driving performance. Among the sequential variants, those that first processed the spatial features before applying the temporal component outperformed those that attempted joint spatiotemporal evaluation.

While the Sequential ViM model exhibited slightly higher autonomy than the Sequential VMamba, it suffered from a high number of disengagements and was therefore only tested once. All sequential models tended to exhibit delayed control responses during operation, initially suspected to stem from long inference times. However, as shown in Table 7.3, even the slowest temporal model operated near 14.5 Hz, comfortably above the input frame rate, indicating that the latency likely arose from internal model state propagation rather than hardware or processing delays.

Interventions were categorised into four main types, along with two minor categories:

1. Poor lane-following behaviour requiring manual correction.
2. Encounters with untrained driving scenarios (excluded from adjusted results).
3. Acceptable navigation but with degraded passenger comfort (e.g., weaving or excessive proximity to parked vehicles).
4. Poor behaviour during parking, reversing, or lane re-entry, requiring manual takeover.

Only Category 2 interventions were excluded from the adjusted results, along with the associated time segments.

Minor intervention categories:

- **Emergency stops (E-stops):** Triggered by low-level hardware safety systems. The manual intervention is excluded as duplicate data, but the associated time is retained.

- **Invalid interventions:** Rare occurrences (e.g., during setup or pre-start phases) excluded from both counts and timing.

Table 7.15 summarises the breakdown of manual intervention categories across all recorded drives. While the ViM sequence model initially appears to perform well with only seven interventions, across categories 1 and 4, this result is misleading as it represents a single run. Repeated trials would likely expose much poorer consistency. Similarly, the videoMAE model, also tested once, exhibited the highest proportion of Category 1 interventions, highlighting its weak lane-following performance.

	Category 1		Category 2		Category 3		Category 4		E-stops		Invalid	
Model	Count	%	Count	%	Count	%	Count	%	Count	%	Count	%
VideoMAE	19	67.86	1	3.57	0	0	3	10.71	3	10.71	2	7.14
VideoMamba	14	25.45	14	25.45	1	1.82	11	20	9	16.36	6	10.91
ViM Sequence	6	35.29	4	23.53	0	0	1	5.88	5	29.41	1	5.88
VMamba Sequence	8	20	16	40	0	0	10	25	6	15	0	0
VMamba	11	29.73	15	40.54	1	2.70	4	10.81	4	10.81	2	5.41

TABLE 7.15: Breakdown of manual interventions by category across all drives.

The results for each model are further illustrated in the following sections using map-based visualisations. These show average performance across all runs, with interventions plotted individually to highlight consistent areas of difficulty.

Because the VideoMAE and Sequence ViM models exhibited particularly poor stability and lane-following, only one drive was performed for each to prevent vehicle risk.

Although VideoMamba also underperformed relative to the Sequence ViM model, multiple runs were recorded to more comprehensively assess the behaviour of a spatiotemporal architecture.

7.4.2.1 videoMAE

As shown in Figure 7.17b, the videoMAE model struggles to follow the optimal path, particularly on narrow roads. Figure 7.17f highlights the instability of the path, with the model exhibiting noticeable lateral movements. This instability contributes to reduced driving speed, as shown in Figures 7.17c and 7.17d. Due to this performance, the model was evaluated in a single run, as reflected by the intervention counts in Figure 7.17g. Compared to the single-frame models from Chapter 5, the videoMAE shows reduced trajectory stability and higher intervention frequency.

(A) Driving mode distribution throughout the route.

(B) Deviation from the optimal path (m).

(C) Vehicle speed (m/s).

(D) Speed relative to the optimal trajectory.

(E) Steering relative to the optimal trajectory.

FIGURE 7.17: On-road performance of the videoMAE model, showing driving mode distribution, path deviation, speed profile, speed relative to optimal and steering relative to optimal.

(F) Trajectory smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 7.17: Additional videoMAE performance metrics, including trajectory smoothness, and intervention counts.

7.4.2.2 videoMamba

The *videoMamba* model exhibits a high number of disengagements, particularly along narrower sections of the route, as shown in Figure 7.18g. As illustrated in Figure 7.18f, the model's steering output is highly unstable, resulting in noticeable vehicle swaying and frequent manual takeovers. This instability also affects overall speed regulation, with fluctuations in throttle control further reducing passenger comfort, as shown in Figure 7.18c.

Overall, the poor steering consistency and irregular speed control indicate that the *videoMamba* model struggles to maintain smooth and reliable driving behaviour in constrained environments. The combination of frequent disengagements and erratic control responses suggests limitations in the model's temporal processing capability, reducing its suitability for real-time driving applications.

(A) Driving mode distribution throughout the route.

(B) Deviation from the optimal path (m).

(C) Average vehicle speed (m/s).

(D) Average speed relative to the optimal trajectory.

(E) Average steering relative to the optimal trajectory.

FIGURE 7.18: On-road performance of the videoMamba model, showing driving mode distribution, path deviation, average speed, average speed relative to optimal and average steering relative to optimal.

(F) Trajectory smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 7.18: Additional videoMamba performance metrics, including smoothness, and intervention counts.

7.4.2.3 Sequence ViM

The *Sequence ViM* model further highlights the limitations of temporal architectures, exhibiting a higher number of disengagements compared to the original ViM model, as shown in Figure 7.19g. The smoothness map in Figure 7.19f indicates unstable steering behaviour, although not as severe as that observed with the *videoMamba* model. Path deviation results in Figure 7.19b reveal notable divergence from the optimal trajectory, while Figure 7.19c demonstrates moderate fluctuations in speed that further reduce stability and driving comfort.

Overall, the *Sequential ViM* model struggles to maintain consistent control across the route, with both steering instability and speed irregularities contributing to frequent disengagements. These results suggest that temporal sequence integration, as implemented in this configuration, does not enhance performance for real-time driving tasks and may instead amplify noise in the model's temporal predictions.

(A) Driving mode distribution throughout the route.

(B) Deviation from the optimal path (m).

(C) Vehicle speed (m/s).

(D) Speed relative to the optimal trajectory.

(E) Steering relative to the optimal trajectory.

FIGURE 7.19: On-road performance of the ViMSequence model, showing driving mode distribution, path deviation, speed, speed relative to optimal and steering relative to optimal.

(F) Trajectory smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 7.19: Additional ViMSequence performance metrics, including trajectory smoothness and intervention counts.

7.4.2.4 Sequence VMamba

The Sequence VMamba model continues to demonstrate the reduced performance characteristics observed in temporal models. As shown in Figure 7.20g, the number of disengagements is significantly higher than in the original single-image VMamba model. These occur most frequently along the narrow road section and at the large bend in the wider section of the route.

The steering smoothness profile in Figure 7.20f highlights inconsistent control behaviour, which corresponds closely with the observed disengagements and manual interventions. Several additional interventions occurred near the roundabouts, which, based on ROSbag playback, were caused by delayed model responses rather than perception errors.

Due to these inconsistencies, the average vehicle speed shown in Figure 7.20c is noticeably lower than that of the single-image models, particularly in the wide road section. Despite these issues, the deviation from the optimal path (Figure 7.20b) remains relatively stable overall, with only minor deviations observed.

(A) Driving mode distribution throughout the route.

(B) Deviation from the optimal path (m).

(C) Average vehicle speed (m/s).

(D) Average speed relative to the optimal trajectory.

(E) Average steering relative to the optimal trajectory.

FIGURE 7.20: On-road performance of the VMambaSequence model, showing driving mode distribution, path deviation, average speed, average speed relative to optimal and average steering relative to optimal.

(F) Trajectory smoothness.

(G) Interventions (blue = manual takeover, red = disengagement).

FIGURE 7.20: Additional VMambaSequence performance metrics, including smoothness and intervention counts.

7.4.3 Driving performance - best drive per model

The results from each model are show in table 7.16, note that an extra model is tested using GNSS+RTK as a comparison. This was not done in the simulation system as a GNSS+RTK model provides little value in a simulated environment for comparison.

Model	videoMAE	VideoMamba	Sequence ViM	Sequence VMamba	VMamba
Drive time (min)	21.72	19.92	23.29	22.68	**18.01**
Time autonomy (%)	80.60	88.29	95.07	94.87	**98.23**
Disengagements	7	1	13	5	**0**
Interventions	22	9	7	5	**4**

TABLE 7.16: Best driving performance for each model.

The following sections present the driving metrics mapped for each model's best run. Unlike the earlier analysis, all driving modes are included, and the metrics for speed, steering smoothness, and path deviation are reported across all autonomous and manual modes. Ideally, the steering smoothness values remain low except during expected manoeuvres such as turning at intersections, moving through round abouts, or parking. Similarly, vehicle speed should vary gradually to enhance passenger comfort, while deviations from the optimal path should remain minimal outside of parking manoeuvres.

7.4.3.1 VideoMAE

The VideoMAE model was evaluated in a single on-road run, and its limitations are evident across several performance metrics. As shown in Figures 7.21b and 7.21c, the model is run at low speeds even on wider road segments, and there are noticeable fluctuations in steering that result in an uneven driving trajectory. The deviation from the optimal path, illustrated in Figure 7.21d, further highlights the model's instability, where repeated divergence and recovery cycles contribute to a high number of manual interventions and disengagements (Figure 7.21e).

A closer examination of the ROSbag playback revealed that several disengagements occurred near or in the round abouts, primarily caused by delayed steering corrections when re-entering the main road. Overall, the VideoMAE model demonstrates limited adaptability to dynamic urban environments, with both steering instability and under-speed behaviour negatively impacting driving smoothness and autonomy.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

(C) Steering smoothness expressed as angular velocity (rad/s).

FIGURE 7.21: Performance of the VideoMAE model during its best driving run showing driving mode, speed and steering smoothness,.

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 7.21: Performance of the VideoMAE model during its best driving run, showing path deviation and intervention distribution.

7.4.3.2 VideoMamba

Examining the disengagement data in Figure 7.22e, it is clear that the model struggled during narrow sections of the road, where multiple disengagements and manual interventions occurred. The intervention near the exit of the parking lot can be attributed to environmental changes, as the house in front of the parking lot had changed significantly since the original data was collected. This, along with the intervention near the right-hand turn round about, which has also changed compared to the original dataset, indicates that the model lacked the generalisation capability expected from incorporating temporal information.

Despite these issues, Figure 7.22c shows a relatively smooth steering profile for most of the journey, suggesting that the interventions were more likely caused by delayed reactions to environmental cues rather than unstable control, for example, the vehicle not turning sufficiently on narrow roads with a bend. Furthermore, Figure 7.22d shows that the model was generally able to maintain proximity to the optimal path, indicating that while temporal fusion improved stability, it did not sufficiently enhance responsiveness or situational adaptability.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

(C) Steering smoothness expressed as angular velocity (rad/s).

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 7.22: Performance of the VideoMamba model during its best driving run, including driving mode, speed, steering smoothness, path deviation and interventions.

7.4.3.3 Sequence ViM

Figure 7.23e shows a high frequency of disengagements throughout the entire route, which reflects the reason this model was only tested once. Despite this, Figure 7.23c indicates relatively stable steering performance, and while Figure 7.23b shows minor speed fluctuations, these alone do not account for the poor overall performance.

As seen in Figure 7.23d, the model frequently deviates from the optimal path, indicating that the model is driving at an angle relative to the road. This behaviour suggests that the model's reaction to the driving environment is delayed, likely due to the accumulation and interpretation of temporal information over multiple frames. As a result, the system tends to disengage before it can complete the corrective steering actions required to realign with the lane.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

(C) Steering smoothness expressed as angular velocity (rad/s).

FIGURE 7.23: Performance of the ViMSequence model during its best driving run, including driving mode, speed and steering smoothness.

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 7.23: Performance of the ViMSequence model during its best driving run, including path deviation and interventions.

7.4.3.4 Sequence VMamba

The Sequence VMamba model demonstrates relatively smooth driving performance, as seen in Figure 7.24c. However, a number of interventions and disengagements are still observed in Figure 7.24e. While Figure 7.24d shows some displacement from the optimal path, it does not fully account for all of the interventions and disengagements. This suggests that delayed responses in the model's output predictions contribute to these events.

Several interventions occur at consistent locations, including the parking lot exit and the right-hand roundabout, indicating that the model has not fully generalised to these areas. The vehicle speed profile (Figure 7.24b) shows generally smooth transitions, but in higher-speed sections the model's limitations reduce the duration over which higher speeds can be safely maintained.

(A) Driving mode throughout the best-performing route.

(B) Vehicle speed profile (m/s).

(C) Steering smoothness expressed as angular velocity (rad/s).

(D) Distance from optimal path during the route (m).

(E) Interventions (blue = manual takeover, red = disengagement).

FIGURE 7.24: Performance of the VMambaSequence model during its best driving run, including speed, steering smoothness, path deviation, and interventions.

7.4.4 Parking, reversing and exiting parallel bays

Using the same parking criteria from Chapter 5 the models were further evaluated for their driving performance which is summarised in table 7.17.

Model	VideoMAE	VideoMamba	Sequence ViM	Sequence VMamba	VMamba
Parking Score	1.25 ± 1.5	0.77 ± 0.83	2.5 ± 1.41	$\mathbf{2.88 \pm 1.11}$	2.58 ± 1.24
Parking left (%)	75	61.54	75	76.47	66.67
Parking right (%)	25	38.46	25	23.53	33.33
Parking known bays (%)	75	53.85	50	52.94	75
Parking unknown bays (%)	25	46.15	50	47.06	25
Reverse Score	$\mathbf{2.00 \pm 0.00}$	1.23 ± 0.93	1.25 ± 1.04	1.53 ± 0.87	1.42 ± 0.67
Reverse left (%)	66.67	76.92	75	82.35	66.67
Reverse right (%)	33.33	23.08	25	17.65	33.33
Reverse known bays (%)	33.33	46.15	25	35.29	50
Reverse unknown bays (%)	66.67	53.85	75	64.71	50
Pullout Score	1.33 ± 1.53	3.38 ± 0.87	3.25 ± 0.71	3.18 ± 1.07	$\mathbf{4.00 \pm 0.00}$
Pullout left (%)	66.67	76.92	75	82.35	63.64
Pullout right (%)	33.33	23.08	25	17.65	36.36
Pullout known bays (%)	33.33	69.23	50	47.06	54.55
Pullout unknown bays (%)	66.67	30.77	50	52.94	45.45

TABLE 7.17: Parking, Reverse and pullout driving results.

The results in Table 7.17 indicate clear improvements in parking and reversing behaviour for the temporal models compared to the non-temporal baselines. In particular, the Sequence VMamba model achieved the highest overall parking score (2.88/4.00), demonstrating an enhanced ability to align and settle within both known and unseen bays. Although a slight output lag was observed, the model maintained stable steering corrections and effectively centred itself within the bay. This suggests that temporal context enabled the model to better anticipate motion trajectories and positional adjustments during low-speed manoeuvres.

The reversing results further highlight the benefits of temporal modelling, with VideoMAE achieving perfect reversals (2.00/2.00) across the limited test cases. While this outcome is likely influenced by the small sample size, temporal models such as Sequence VMamba also exhibited consistent improvements (1.53/2.00) in reversing accuracy relative to static baselines.

In contrast, the pull-out task revealed some degradation in performance for several temporal models. This likely arises from delayed steering adjustments when transitioning from

confined parking spaces into the narrow exit lanes, exposing a potential limitation of temporal aggregation mechanisms. Further optimisation to mitigate this aggregation lag could better showcase the advantages of temporal modelling. Future work could explore methods such as adaptive temporal weighting to prioritise more recent frame information, alternative temporal architectures, or shorter sequence windows to improve responsiveness during transitional manoeuvres.

7.4.5 Lane following - Road features

Using the same metrics from Chapter 5, the models performance on various road features is tested. Table 7.18 summarises the performance of each model at these critical landmarks.

Model	VideoMAE	VideoMamba	Sequence ViM	Sequence VMamba	VMamba
Roundabout 1 (R1)	0.00	**3.00 ± 0.00**	1.00	1.67 ± 0.58	**3.00 ± 0.00**
Snaking road	0.00	1.33 ± 0.58	1.00	0.67 ± 0.58	**1.67 ± 0.58**
Roundabout 2 (R2)	0.00	2.67 ± 0.58	3.00	**3.00 ± 0.00**	2.67 ± 0.58
Intersection 1 (I1)	2.00	1.67 ± 0.58	**2.00**	1.00 ± 0.00	1.33 ± 0.58
Dual lane unmarked 1	1.00	0.33 ± 0.58	**2.00**	1.00 ± 0.00	1.67 ± 0.58
Intersection 2 (I2)	1.00	**1.67 ± 0.58**	1.00	1.00 ± 1.00	1.33 ± 0.58
Carpark	0.00	**3.00 ± 0.00**	2.00	2.67 ± 1.15	1.67 ± 0.58
Intersection 3 (I3)	1.00	0.67 ± 0.58	1.00	**1.67 ± 0.58**	**1.67 ± 0.58**
Dual lane unmarked 2	0.00	1.00 ± 0.00	**2.00**	**2.00 ± 0.00**	1.67 ± 0.58
Intersection 4 (I4)	1.00	1.67 ± 0.58	0.00	1.00 ± 1.00	**2.00 ± 0.00**
Round about right	1.00	1.67 ± 0.58	**3.00**	2.33 ± 1.15	2.33 ± 0.58
Dual lane marked	0.00	1.33 ± 1.15	0.00	**2.00 ± 0.00**	**2.00 ± 0.00**
Roundabout 3 (R3)	1.00	2.67 ± 0.58	**3.00**	1.33 ± 1.53	2.00 ± 0.00
Roundabout 4 (R4)	1.00	2.67 ± 0.58	2.00	**3.00 ± 0.00**	2.00 ± 0.00
Total score (/36)	**9.00**	**25.33 ± 0.49**	**23.00**	**24.33 ± 0.54**	**28.67 ± 0.41**

TABLE 7.18: Landmark driving results.

The results presented in Table 7.18 indicate that the original single-frame VMamba model remains the strongest overall performer, achieving the highest total score of 28.67/36 across all landmark scenarios. However, several temporal variants demonstrate notable improvements in specific driving contexts. In particular, the VideoMamba model performed well in the parking lot section (3.00/5.00), one of the most challenging environments due to its narrow lanes and irregular markings. This improvement highlights the temporal model's

enhanced ability to maintain stability and spatial awareness in confined areas with reduced visual structure.

Despite this, earlier results (see Chapter 5) show that certain non-temporal models achieved comparable or superior performance in the same carpark scenario, suggesting that temporal aggregation alone does not fully resolve low-speed navigation challenges. However, the two best-performing temporal architectures, VideoMamba (25.33/36.00) and Sequence VMamba (24.33/36.00), demonstrated consistent control across most road features, including complex roundabouts and intersections.

These findings suggest that, while the integration of temporal information introduces lag in decision making, it also contributes to smoother motion continuity and better path maintenance. Overcoming this temporal delay could further unlock the advantages of temporal vision models for autonomous driving.

7.5 Discussion

The results presented in this chapter were compared against the baseline single-frame models to assess whether incorporating temporal information and metadata feedback provides meaningful improvements in driving performance. Specifically, the analysis evaluates if the performance gains justify the increased computational cost and model complexity associated with these extended architectures.

This discussion is organised into two main parts: the first examines the effectiveness of the temporal models in capturing sequential dependencies, while the second focuses on the feedback-based approach that integrates additional vehicle state information into the prediction process.

7.5.1 Temporal Models

The results presented in Tables 7.4 and 7.14 indicate that the VideoMAE model performed significantly worse than the single-image transformer models, which were able to successfully complete multiple autonomous drives. This under performance is attributable to the absence of large-scale pretraining, which is particularly critical for Masked Autoencoder (MAE)-based architectures. Overall, the integration of temporal information did not improve model performance; instead, it resulted in a notable degradation.

During real-time operation, all temporal models exhibited a perceived delay in steering and speed control. However, Table 7.3 confirms that this issue is not caused by inference latency. Rather, it is likely due to the aggregation of information across multiple frames before producing an output. This aggregation introduces a delay in decision-making, which is undesirable for real-time control tasks such as autonomous driving. More research is required to quantify and mitigate this perceptual lag.

The sequence-based models implemented in this study use bidirectional VMamba blocks to process temporal information. Although bidirectional processing is effective for off-line video classification tasks, it is less suitable for causal real-time regression tasks such as speed and steering prediction. In contrast, a unidirectional (forward-only) state space mechanism, similar to that used in natural language processing, may be more appropriate, as it naturally assigns greater weight to recent observations. VideoMamba suffers from a similar limitation, as its forward–backward scan mechanism introduces latency that may be beneficial for context aggregation but detrimental to real-time control. Future work should therefore explore the use of purely forward-scanning Mamba blocks for temporal modelling.

Analysis of the training and data set evaluation, presented in Tables 7.4 –7.11 and Figures 7.11 – 7.16, suggested that the final performance of the model would be at least comparable to that of the single-image models. However, as demonstrated in Table 7.14, this expectation was not realised. In several cases, models were evaluated only once during on-road testing due to safety concerns arising from unstable behaviour.

These findings further reinforce the limitation of assessing models solely using offline datasets. Although data set-based metrics indicated competitive performance, additional challenges emerged during real-world deployment, particularly those associated with temporal latency observed across all models presented in this chapter. This discrepancy highlights that evaluation on curated datasets alone is insufficient to fully characterise operational performance, as real-world conditions introduce system-level factors that are not captured during standard training and testing procedures.

Additionally, reducing the temporal window from eight frames to three may help alleviate perceptual lag while still preserving short-term temporal coherence. A shorter frame sequence would also reduce the extent of temporal smoothing, enabling more immediate control actions.

The breakdown of manual intervention types in Table 7.15 demonstrates that temporal integration did not produce improvements in lane follow-up or parking performance. This is consistent with the map-based metrics in Sections 7.4.2 and 7.4.3, which show unstable

steering behaviour, frequent swerving, and an increased number of interventions. Table 7.17 indicates that the VMamba sequence model achieved marginal improvements in parking tasks compared to the original VMamba. However, this may be a consequence of the perceptual delay, causing the model to progress further into the bay before turning, which incidentally produced a better alignment. Notably, VideoMAE achieved perfect performance in reverse parking tasks, outperforming all other models. However, when switching back to forward driving, VMamba remained the overall strongest performer. Table 7.18 supports this, showing that although alternative models excelled in specific scenarios, VMamba consistently achieved the best overall results.

In summary, the temporal models tested in this study did not improve performance over single-frame models and introduced a perceptual delay that significantly impairs real-world usability. Future work should prioritise resolving this latency through causal temporal modelling, reduced input frame windows, or more sophisticated temporal fusion strategies. In addition, adaptive fusion mechanisms should be explored, in which the temporal context varies with vehicle speed, for example, shorter temporal windows for high-speed driving and extended temporal context for low-speed manoeuvres.

7.5.2 Feedback Model

The model developed using previous-frame metadata demonstrated concerning behaviour initially during training and then during ground-truth validation and was therefore not deployed on the real vehicle. The network exhibited a strong tendency to ignore incoming visual information, instead predicting outputs similar to those of the previous frame. This occurred because the use of mean absolute error (MAE) as the training objective, combined with the small temporal differences between consecutive control metadata, resulted in minimal loss. Consequently, gradient updates were small, and the model had little incentive to explore visual features during training.

A potential improvement would be to incorporate several past frames of metadata and apply masking or dropout techniques to encourage better generalisation. However, as seen in the temporal models discussed earlier, this approach risks introducing artificial lag due to accumulated historical context.

A more effective strategy may involve re-structuring the model to include a residual connection rather than the current direct concatenation of metadata and visual features. In such a configuration, the network would learn to predict corrective offsets to the previous control commands based on visual input, rather than relying primarily on the metadata embedding.

This design would promote stronger attention to visual cues during back propagation and improve robustness in dynamic scenes.

Finally, training for this model was limited by the availability of metadata. The collected metadata only contained the previous steering and speed *commands*, which are not necessarily equivalent to the vehicle's actual physical states due to latency and control feedback effects. Future implementations would benefit from including true measured steering and velocity feedback, allowing the model to learn a more accurate relationship between control inputs, vehicle state, and visual perception.

7.6 Conclusion

This chapter aimed to evaluate the impact of incorporating temporal information and metadata feedback into vision-based driving models. The results demonstrate that in their current form, both approaches did not improve overall driving performance and, in all cases, introduced additional challenges. Temporal models suffered from a perceptual lag that reduced their effectiveness in real-time driving, while the feedback model was overly influenced by prior control commands, limiting its ability to respond accurately to new observations.

Despite these limitations, this work has highlighted several critical insights. Firstly, there exists a significant gap between performance in simulation or dataset-based evaluation and performance in real-world environments. Secondly, models that rely on additional temporal or feedback information must be carefully structured to avoid latency or feedback-induced bias. Finally, the increased computational cost and inference time associated with these models currently constrain their applicability in resource-limited systems such as the autonomous shuttle bus.

7.6.1 Novel Contributions

This chapter makes several key contributions to the understanding and evaluation of temporal and feedback models for autonomous driving. Firstly, it provides a comprehensive assessment of temporal vision models in real-world driving, highlighting the perceptual lag that can emerge despite promising performance in simulation environments. Secondly, it critically evaluates metadata feedback models, demonstrating that simple reliance on previous commands can degrade driving performance rather than enhance it. Third, the work identifies and characterises the simulation-to-reality gap, emphasising the necessity of extensive real-world testing to validate model effectiveness. Finally, the chapter quantifies the

impact of increased model complexity and temporal aggregation on resource-limited vehicles, providing insights into the practical constraints associated with deploying these models in operational autonomous systems.

7.6.2 Future Work

Building on the findings of this chapter, several avenues for future research are identified. First, the metadata feedback model could be improved by incorporating additional vehicle state information, such as steering and speed sensor feedback, GPS orientation, and IMU data. Such features have the potential to provide richer context for model predictions, enabling more precise control in real-world driving. Careful consideration will be required to prevent over fitting to specific routes or environments, which could reduce the model's ability to generalise to previously unseen scenarios.

Second, the current implementation of sequence models freezes the model weightings, which limits the model's capacity to adapt to new temporal and vehicle data. Future work should explore adaptive or incremental training strategies that allow previously frozen layers to continue learning during training with new feedback inputs. This could improve responsiveness and mitigate performance degradation arising from static weight assignments.

Third, the temporal models investigated in this work introduce perceptual lag due to bidirectional aggregation of sequential data. Replacing this with a forward-only temporal modelling approach, such as a Mamba block, could reduce latency while preserving the benefits of temporal smoothing. Additionally, optimising the temporal window by using fewer consecutive frames, such as the last three, may provide sufficient temporal context while keeping predictions closer to real-time observations.

Fourth, further research is needed on the design of resource-aware architectures that balance temporal or feedback enhancements with the computational and memory constraints of embedded autonomous systems. This includes exploring model compression, pruning, or lightweight transformer variants to maintain real-time inference capabilities without sacrificing performance.

Finally, extensive real-world testing remains crucial. The results of this chapter underscore the gap between simulation or data set performance and deployment in live environments. Systematic evaluation across diverse scenarios, lighting conditions, and road types will be necessary to fully validate the potential of temporal and feedback models for autonomous driving.

Collectively, these future directions aim to address the limitations identified in this work, guiding the development of more robust, responsive, and efficient autonomous driving models capable of bridging the gap between controlled experiments and real-world deployment.

Chapter 8

Exploration of Driver Assist Systems to Improve Autonomous Driving in Suburban Driving

8.1 Introduction

The neural networks developed in chapters four and five have shown their ability to drive on the suburban streets of Eglinton. Although they were able to perform well for most of the drive, there were situations of suboptimal behaviour that resulted in interventions, disengagements, and longer drive times. Specifically, the models did not show consistent behaviour across drives, which resulted in cases where the vehicle tended to drift toward the curb, which impacted driving performance. Because we cannot guarantee that a trained model will never perform poorly in all situations, additional safeguards are required to ensure the networks perform correctly that are independent of the models and human teachers.

This chapter introduces two deterministic management tools to prevent unsafe manoeuvres and poor driving performance. In addition, a variation of reinforcement learning is introduced as a potential method for future model improvements. These additional measures will improve the robustness of the autonomous system resulting in improved passenger safety. The vehicle will no longer rely solely on stochastic models, which will increase public trust and acceptance of the technology. In addition, the addition of safety measures means that more generalised models could be used to improve wider driving scenarios, which is a key metric in SAE level 4 autonomy.

Chapter 5 has already briefly mentioned the slow/stop node, which uses safety LiDAR information to alter the driving output, and this chapter will expand on that description. The

second management tool works on a simple vision-based grid system to detect usable roads and shift the driving perspective of the models to improve road driving.

The introduction of these tools should achieve the following:

1. **Maintain SAE performance** – checks should not introduce sources of data that require external resources.
2. **Reduce the frequency of interventions / disengagements** – minimise human- and / or hardware-level interventions are required.
3. **Enhance passenger comfort** – evaluated by speed changes and percent change in steering angle over time.
4. **Minimise impact on trip duration** – maintain driving performance as much as possible to encourage future users.

The chapter is broken down into the following five sections:

1. A review of current safety systems on vehicles.
2. Description of system updates.
3. Results from vehicle driving in real-world scenarios.
4. Discussion of findings.
5. Conclusion and suggestions for future work.

8.2 Related Works

The rapid development of autonomous vehicles (AVs) has been accompanied by substantial public concern about the safety and reliability of current AI-driven systems. Ahmed et al. conducted a survey examining interactions between AVs, pedestrians, and cyclists, noting that a large proportion of road fatalities occur in collisions involving vulnerable road users [258]. Their findings highlight that current perception systems, particularly pedestrian and cyclist detection and intent estimation algorithms, struggle to deliver robust and reliable performance. Public perception reflects similar concerns. A Deloitte consumer report revealed that in 2019, approximately 50% of respondents across several major AV-developing nations believed that autonomous vehicles would not be safe [259]. Furthermore, a report by CNBC described local frustrations with Alphabet's Waymo vehicles, with residents criticising their inability to navigate common road scenarios, such as T-intersections, and some resorting to

risky or illegal driving behaviours to avoid them [260]. These findings indicate that achieving public trust in AVs requires systems that balance safety, reliability, and practicality without impeding normal traffic flow.

End-to-end learning approaches have been widely explored as a method to improve driving performance by learning steering and speed commands directly from sensory inputs. Tampuu et al. provide an overview of end-to-end driving architectures and training methodologies, highlighting a common problem with imitation learning called distribution shift problem [261]. In imitation learning, models are trained on expert demonstrations; however, during deployment, minor deviations from the training distribution can lead to situations the model has not been trained for and it needs to be able to continue without disruption. In Chapter 5, a strategy was applied to alleviate this problem using shifted image enhancements to increase the robustness of the model, although the results showed that recovery from larger deviations remained unreliable. Cai et al. describe this phenomenon as the distribution mismatch problem in the context of imitation reinforcement learning [262]. As vehicle actions directly influence subsequent observations, the assumption of independent and identically distributed (i.i.d.) data no longer holds. By combining imitation learning with reinforcement learning, models can explore corrective behaviours and improve trajectory recovery. Likewise, Kendall et al. introduced a reinforcement learning-based driving model that learns directly from randomly initialised parameters, using travelled distance without intervention as the only reward signal [263]. Although this model successfully completed simple driving tasks using only camera, speed, and steering as an input, its real-world evaluation was limited to an isolated road with minimal external interactions. These studies collectively emphasise that AV systems must extend beyond passive imitation, incorporating exploratory learning and corrective safety mechanisms to re-align the vehicle with familiar operating states.

Object-level understanding through perception tasks, such as semantic segmentation, is essential for many autonomous driving systems. Elhassan et al. evaluated real-time semantic segmentation models based on CNNs, Transformers, and hybrid architectures, highlighting the persistent trade-off between accuracy and computational efficiency [264]. While state-of-the-art models deliver high accuracy, their computational demands limit deployment on resource-constrained platforms. Holder et al. conducted real-time evaluations of segmentation algorithms across diverse embedded hardware platforms, including the NVIDIA Xavier, and found that most models failed to achieve true real-time performance, with the best real-time operating model achieving 75.5% mIoU [265]. Similarly, Cakir et al. demonstrated that

FasterSeg could run in real time on an NVIDIA Jetson Xavier, achieving an average mIoU of 65.44% on input images of 320 × 256 pixels on a small-scale Carolo test track [266]. Although these results are promising, they remain insufficient for complex real-world urban environments that require multiple concurrent models and safety systems. Consequently, semantic segmentation may impose excessive computational burden on the autonomous shuttle platform presented in this work. Instead, this thesis explores a lightweight alternative that demonstrates competitive performance while maintaining real-time feasibility on constrained hardware.

8.3 System Updates

To improve safety and reliability, additional safeguards were implemented to function as a driver assist system, similar to those found in modern vehicles for human drivers. These safety layers are designed to detect and correct undesirable actions resulting from AI decision errors. The following sections outline two key system updates that serve as real-time safety checks, ensuring that the vehicle maintains stable and appropriate behaviour under variable conditions.

8.3.1 Slow-stop node

Guaranteed reliable driving performance using end-to-end learning methods is not yet possible, as these systems make predictions solely based on the data they have been trained on. This presents a significant challenge when aiming to design a safe and dependable autonomous system, particularly on the narrow roads in Eglinton.

As outlined in Chapter 2, the autonomous shuttle buses are equipped with four safety LiDAR sensors that create a protective curtain around the vehicle. This safety layer acts as the final line of defence and should only activate in cases of software malfunction or erratic vehicle behaviour. When triggered, the safety system initiates a low-level emergency stop (E-stop), disabling all driving commands until the system is manually rearmed and the PLC safety checks confirm that operation can resume. Although this process typically takes between 10 and 20 seconds, an acceptable delay in isolation, frequent activations negatively impact public perception of the vehicle's reliability. However, this subsystem is essential to maintain safety and therefore cannot be removed.

The challenge, then, is to maintain a robust low-level safety mechanism while minimising unnecessary triggers. Each type of E-stop event must be addressed with software-based

strategies where possible, ensuring the vehicle can maintain continuous motion without resorting to driving at minimal speed for the entire route.

Through extensive testing and analysis of trial drives, five main categories of E-stop events were identified:

1. System failure due to invalid mechanical constraints (e.g., exceeding safe steering limits at a given speed).
2. E-stop triggered by obstacles directly in front of the vehicle.
3. E-stop triggered by obstacles in close proximity to the vehicle's sides.
4. E-stop triggered when the vehicle turns toward nearby obstacles.
5. E-stop triggered by suddenly appearing or transient obstacles.

Category 1: Mechanical constraint failures These failures occurred when the vehicle is operated near the limits of its mechanical range through CANbus commands. Despite pre-set limits, overflow effects occasionally caused random behaviour, such as unintended activation of the accessibility ramp after a stop. To prevent this, an additional safety factor was applied beyond the documented mechanical constraints.

Category 2: Frontal obstacle detection This category includes potential collisions with objects or road users ahead, particularly at intersections or roundabouts. Another case occurs when entering a winding road section, as shown in Figure 8.1.

FIGURE 8.1: Example of a snaking road section requiring dynamic speed adjustment.

To mitigate these risks, a new *slow stop node* was developed. This node dynamically adjusts the vehicle's speed based on the available free space ahead. The initial implementation used a single forward detection block; however, this approach significantly restricted the vehicle's ability to navigate bends smoothly. The system was therefore extended to a multi block configuration that considers both longitudinal and lateral regions, adapting not only to forward distance but also to the current steering angle.

The layout of this system is shown in Figure 8.2. The green region represents the *frontal slow zone*, which scales with vehicle speed and triggers a speed reduction whenever obstacles (shown in grey) enter this area. The smaller red region indicates the *stop zone*, in which the vehicle speed is reduced to near zero while a nearby obstacle remains within the critical threshold. The purple regions on either side vary with both speed and steering angle and help minimise the impact of the slow zone during cornering by allowing smoother navigation through curved sections.

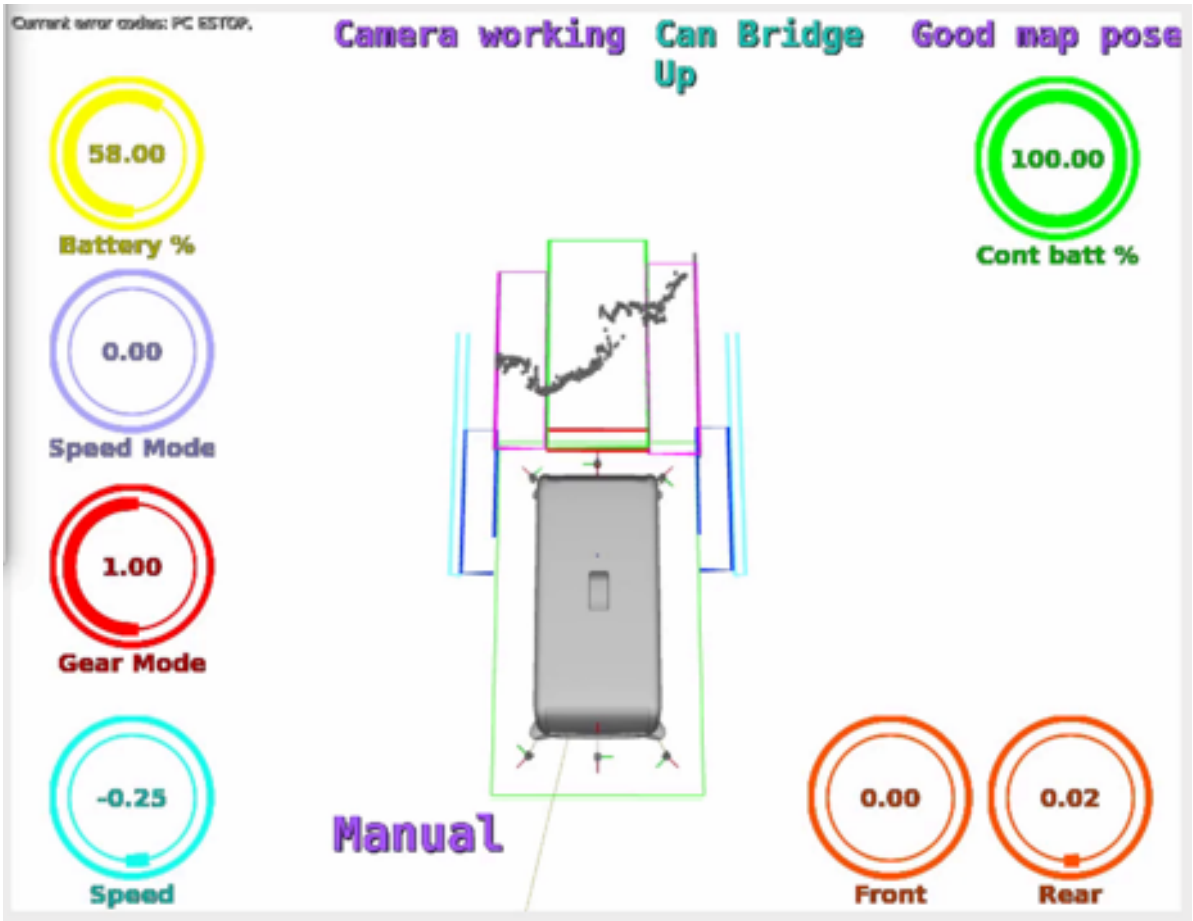


FIGURE 8.2: Layout of the slow-stop node detection system displayed in RVIZ.

Category 3: Side obstacle proximity This issue is particularly critical on the narrow one-way streets of Eglinton, as shown in Figure 8.3. In these areas, poorly trained models occasionally exhibit a tendency to drive too close to the curb, triggering unnecessary emergency stops (E-stops) even when no real collision risk is present.

To address this, the slow-stop node was extended to include side-obstacle detection along both edges of the vehicle's projected driving path. This adaptation introduces a small corrective steering bias, rightward when obstacles appear on the left, and leftward when obstacles appear on the right. These adjustments are tuned to be subtle enough to preserve passenger comfort while effectively preventing safety triggers.

As illustrated in Figure 8.2, the light blue regions inhibit steering actions toward nearby obstacles and dynamically expand in both width and length with speed and steering angle. The darker blue regions apply gentle steering corrections away from detected obstacles, likewise adapting with vehicle speed to maintain safe lateral clearance without abrupt manoeuvres.

FIGURE 8.3: Example of a narrow one-way road in Eglinton, highlighting conditions that challenge side obstacle detection.

Category 4: Turning toward side obstacles Even with side clearance checks, the vehicle may occasionally steer toward obstacles such as trees or parked cars due to insufficiently generalised training. While better training data can mitigate this, additional logic was implemented in the slow zone node to monitor lateral regions further ahead. When obstacles

were detected, turning in that direction was restricted. For narrow or curved roads (as in Figure 8.3), both directions remained permissible, but the maximum allowed speed was capped to ensure safe passage. This is covered by the regions discussed in Category 3.

Category 5: Sudden or transient obstacles This category is the most challenging. It includes objects that appear abruptly, such as pedestrians or debris, but also transient detections caused by environmental factors. For instance, tall grass or small plants may momentarily fall below the LiDAR detection height (typically 30 cm) and then reappear, triggering an unnecessary E-stop. Similarly, small road bumps (see Figure 8.4) can cause the sensors to misinterpret terrain changes as obstacles. Solutions include:

- Slowing the vehicle in known problematic areas (learned from prior data);
- Adjusting LiDAR trigger thresholds to ignore small, short-lived objects;
- Increasing trigger duration to require multiple confirming scans; or
- Introducing complementary vision-based detection to cross-check LiDAR inputs.

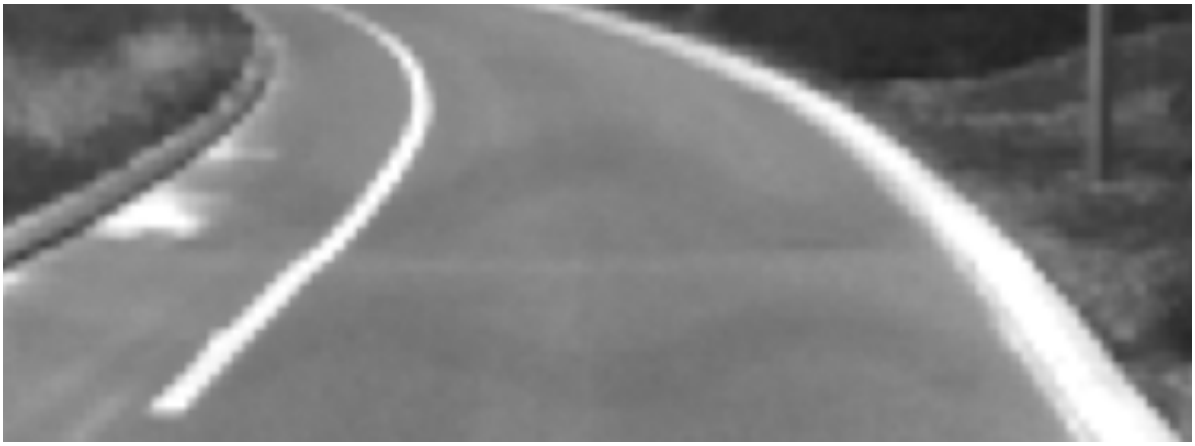


FIGURE 8.4: Example location where a small road bump caused a false E-stop trigger.

To further improve reliability, a second safety mechanism, the *low-level detection system*, was introduced. This system uses camera data to provide an additional perception layer, diversifying the safety sensor suite and improving resilience against LiDAR-only failure modes.

8.3.2 Low-Level Detection System

An additional improvement to the perception stack was developed to address hazards outside the visibility range of the safety LiDARs. The safety LiDARs are mounted approximately

30 cm above ground level, enabling them to detect most obstacles that could pose a risk to the vehicle. However, certain low-lying hazards, such as road debris, construction materials, or curb obstructions, may remain undetected. This issue is particularly relevant in the suburb of Eglinton, where ongoing construction often introduces low level hazards. Furthermore, vehicles with high ground clearance, such as utes with elevated suspensions, may not be fully captured by the LiDAR safety envelope.

To mitigate these limitations, a new camera-based low-level detection system was introduced to supplement the LiDAR coverage. The proposed system uses computer vision techniques to classify small image segments as either *road* or *non-road*, rather than performing explicit object detection or semantic segmentation. This design choice avoids reliance on predefined object classes, allowing the model to generalise to previously unseen obstacles while maintaining focus on the immediate road surface. Each input image is cropped into patches of size 110×400, representing the immediate driving area in front of the vehicle.

A compact Vision Mamba (ViM) model was trained to classify grey scale image segments of size 7×10 pixels. This pushes the architectural limits of the model, as the original ViM framework was validated on RGB images of 224×224 resolution [194], and its smaller variant on the CIFAR-100 dataset at 32×32 resolution [267]. The small patch size enables fine-grained analysis, allowing the vehicle to maintain accurate alignment even when the drivable area is restricted.

The training data was manually labelled by selecting image patches that clearly contained only the road surface, excluding curbs, vegetation, and other characteristics (Figure 8.5). The segmented patches were stored in separate folders for *road* and *non-road* categories, simplifying data management during model training (Figure 8.6). The model predicts the class of each segment using arg max of its output logits. The main model parameters are summarised in Table 8.1.

FIGURE 8.5: Manual road labelling process. Image patches containing only road surface are selected.

Parameter	mini-ViM
Patch Size	(1, 1)
Classifications	2
Embedding Dimension	128
Depth	4
State Space Size	16
Dropout	0.1

TABLE 8.1: Model parameters for the miniature ViM used for low level detection on the nUWAy shuttle bus.

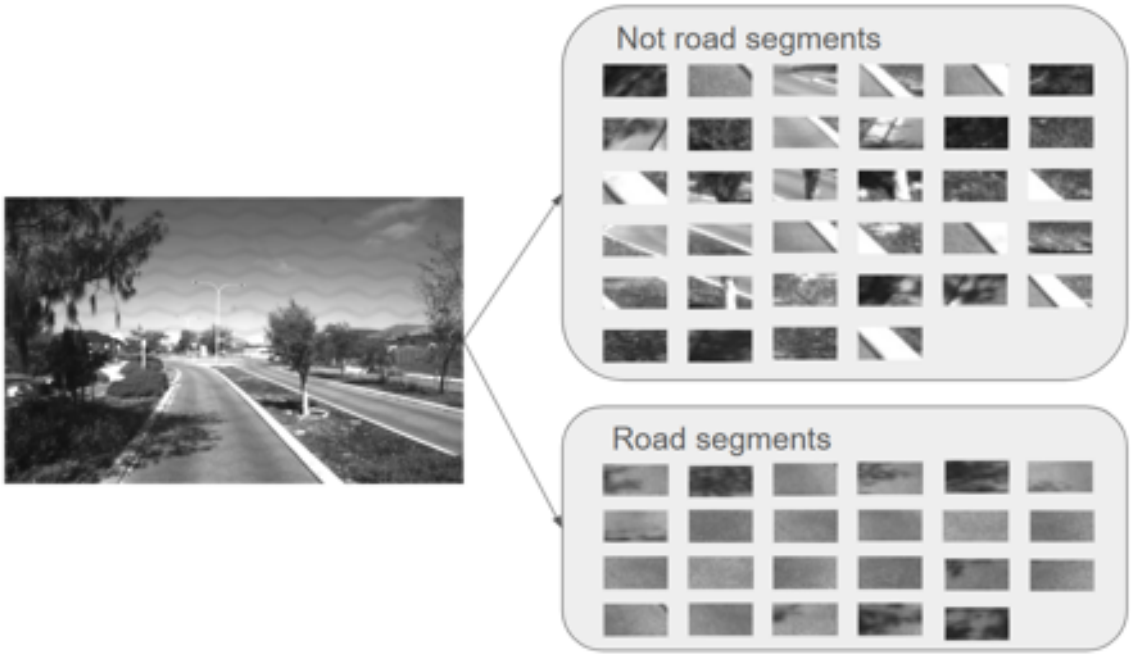


FIGURE 8.6: Road image segmentation and classification using the ViM model. Each segment is classified as road or non-road.

After training, the model achieved a classification loss of *0.0183%* on the training set and *0.0179* on the validation set. The model obtained a training accuracy of *89%* and a validation accuracy of *89.23%*. With a batch size of *15*, each epoch was completed in approximately *1.03 minutes*. The complete prediction to point offset occurs in approximately *0.458 seconds*.

Once predictions are obtained, the system locates the centre of the largest continuous "road" region in each image row. A line of best fit is calculated through these points and its intersection with the midline of the image height is used to estimate the centre of the road. The deviation between this estimated centre and the midpoint of the image (the *centre line*) indicates the lateral displacement of the vehicle. If this deviation exceeds 3%, experimentally

determined to balance responsiveness and stability, the input image is shifted by two pixels in the corresponding direction, up to a maximum of 40 pixels. This feedback mechanism effectively nudges the vehicle's control response to maintain proper alignment with the centre of the road. The script runs at 2 Hz, ensuring smooth corrections without abrupt steering changes.

System Optimisation Subsequent testing identified opportunities for both image processing and run-time optimisation.

Image processing improvements:

- Replaced the standard line of best fit with a RANSAC -based fit to reject outliers and improve the estimation of the road centre.
- Multi-sampling introduced: each patch is classified seven times, and the modal prediction is selected to reduce stochastic noise.
- Implemented spatial smoothing: if a patch is classified as non-road but surrounded by road patches on all four sides, it is reclassified as road to correct for local misclassifications.

Runtime optimisations:

- Implemented turning lockouts when the vehicle is too close to an edge. The system checks the first six lower rows for continuous non-road segments near the frame edges; if more than 11 consecutive non-road tiles are found, turning in that direction is disabled.
- Dynamic adjustment of the target center line position when wide unmarked roads are detected, ensuring that the vehicle avoids driving in the middle of open lanes.
- Only activate the system if more than 200 road tiles are detected, it is assumed that less than 200 road tiles means the vehicle is no longer able to detect the road.
- Shifting only occurs if the angle of the line of best fit is less than 13°, as any more typically indicates a turn where adjusting the steering during the manoeuvre may impact its driving ability.

These improvements collectively enhanced both the reliability and safety of low level object detection and aim to produce a smoother driving experience.

8.4 Results

This section presents the outcomes of the implemented safety systems and evaluates their impact on the overall driving performance of the autonomous shuttle. The results highlight how each safety enhancement contributed to increased operational reliability, smoother vehicle control, and reduced risk during navigation.

8.4.1 Slow-Stop Node

Figure 8.7 illustrates the spatial distribution of disengagements during an average drive prior to the implementation of the slow-stop node. These results, obtained using the original PilotNet model before the second stage of data cleaning, demonstrate a high frequency of disengagements occurring throughout the route. This highlights the model's limited ability to maintain autonomous control without an additional safety mechanism.

FIGURE 8.7: Number and distribution of disengagements prior to the introduction of the slow/stop node.

8.4.1.1 Driving Performance

The introduction of the slow-stop node resulted in measurable improvements in autonomous stability and drivability. Table 8.2 provides a comparison of driving performance for a subset of models with and without the safety system enabled. The complete set of metrics for the remaining models operating with the safety system is presented later in this chapter.

Model	ViM		PilotNet		EfficientNet		PointMamba	
	Safety	No Safety	Safety	No Safety	Safety	No Safety	Safety	No Safety
Drive time (min)	18.76 ± 0.61	19.40	20.26 ± 0.53	21.12	20.89 ± 1.70	20.17	19.53	17.91
Time autonomy (%)	95.67 ± 3.27	88.42	90.07 ± 2.76	83.40	93.93 ± 2.15	79.72	74.59	79.72
Disengagements	2.67 ± 2.52	5	6.00 ± 1.00	8	3.33 ± 2.52	8	4	9
Interventions	4.00 ± 2.65	10	11.33 ± 5.13	19	6.67 ± 3.06	9	32	24
Slow active (min)	1.40 ± 0.16	-	1.26 ± 0.16	-	1.52 ± 0.27	-	1.58	-
Slow active (%)	7.28 ± 1.07	-	6.08 ± 0.87	-	7.10 ± 0.80	-	7.16	-
Stop active (min)	0.63 ± 0.12	-	0.92 ± 0.16	-	0.74 ± 0.22	-	0.92	-
Stop active (%)	3.23 ± 0.49	-	4.45 ± 0.70	-	3.46 ± 0.88	-	4.16	-

TABLE 8.2: Autonomous driving results with and without the slow/stop node.

The reduction in emergency stop (E-stop) events allowed the vehicle to maintain motion without requiring system resets. Although the slow-stop node temporarily reduces vehicle speed, the overall drive time decreased for all models except PointMamba. This indicates that smoother driving more than compensates for speed reductions. In contrast, PointMamba exhibited reduced performance due to its limited ability to maintain central lane positioning, as discussed in Chapter 6.

Excluding PointMamba, all models demonstrated an increase in autonomous driving time of approximately 7– 14%, along with substantial reductions in disengagements and interventions. For example, ViM experienced nearly a 50% reduction in interventions. Across all models, the safety node was active in slow mode for approximately 6–7% of the total drive duration and in stop mode for 3–4%.

Table 8.3 provides a breakdown of the types of manual intervention. Both ViM and PilotNet recorded reductions in Category 1 interventions (lane centring corrections) and Category 4 interventions (parking-related disruptions), indicating a stabilising effect from the safety mechanism.

	Category 1		Category 2		Category 3		Category 4		E-stops		Invalid	
Item	Count	%	Count	%	Count	%	Count	%	Count	%	Count	%
ViM (average)	3	25.00	4.67	38.89	0	0.00	1.33	11.11	6	16.67	1	8.33
ViM (without)	5	19.23	11	42.31	0	0	5	19.23	5	19.23	0	0
PilotNet (average)	8.67	49.06	4	22.64	0.33	1.89	2.33	13.21	2	11.32	0.33	1.89
PilotNet (without)	13	44.83	6	20.69	0	0	6	20.69	3	10.34	1	3.45
EfficientNet (average)	3.67	35.48	3.33	32.26	0	0.00	2.67	25.81	0.33	3.23	0.33	3.23
EfficientNet (without)	3	20	5	33.33	1	6.67	5	33.33	0	0	1	6.67
PointMamba	32	64.00	13	26.00	0	0.00	-	-	4	8.00	1	2.00
PointMamba (without)	24	55.81	6	13.95	0	0	-	-	7	16.28	6	13.95

TABLE 8.3: Breakdown of manual interventions.

The remaining models were only evaluated with the safety system active. Their results are summarised in Table 8.4, showing that slow-stop activations accounted for approximately 9–10% of total drive time for most models, while videoMAE and Sequence ViM both had a higher percentage of activation. Although this does not significantly hinder operational performance, it is noticeable to passengers and provides a benchmark for future system improvements.

Model	Swin	VMamba	DeiT	Joint LiDAR–Vision	videoMAE	VideoMamba	Sequence ViM	Sequence VMamba
Slow active (min)	1.55 ± 0.22	1.37 ± 0.10	1.16 ± 0.24	1.27 ± 0.12	1.40	1.27 ± 0.15	1.92	1.55 ± 0.21
Slow active (%)	7.43 ± 1.12	7.07 ± 0.15	6.11 ± 0.78	5.95 ± 0.64	6.29	5.85 ± 0.57	8.16	6.60 ± 0.88
Stop active (min)	0.70 ± 0.24	0.48 ± 0.02	0.54 ± 0.13	0.71 ± 0.06	1.07	0.70 ± 0.25	0.93	0.84 ± 0.08
Stop active (%)	3.33 ± 1.15	2.46 ± 0.22	2.83 ± 0.46	3.32 ± 0.21	4.79	3.23 ± 1.07	3.96	3.61 ± 0.45

TABLE 8.4: Slow and stop activity results across all models.

Spatial analysis of slow-stop activations shown in the following map metrics further identifies recurring locations of performance degradation. These regions provide valuable insight for future improvements through data augmentation, targeted retraining, and model-specific optimisation.

8.4.1.2 Impact of the Slow-Stop Node

This section evaluates the influence of the slow stop node on the best-performing run of each model. In the following figures, green indicates normal operation, yellow denotes periods where the slow mechanism reduces vehicle speed, and red represents brief full-stop activations to avoid disengagements.

ViM As shown in Figure 8.8, the ViM model relies significantly on the slow-stop system, particularly at the bend that leads back to the third roundabout. This behaviour is expected, as the vehicle attempts to maintain high speeds and its suboptimal lane positioning increases the likelihood of slow stop interventions. Additional activations are observed around the second roundabout, where the model struggles with consistent lateral control. In the model trained with pull-out data, some instances of steering toward the curb are also evident, further emphasising the necessity of the slow-stop node.

FIGURE 8.8: Slow-stop activations for the ViM model.

PilotNet Figure 8.9 shows that the PilotNet model is frequently affected by the slow stop system across multiple road segments. This is consistent with its overall performance, where the vehicle often approaches bends too quickly or drifts toward the curb, triggering safety interventions.

FIGURE 8.9: Slow-stop activations for the PilotNet model.

EfficientNet As illustrated in Figure 8.10, the EfficientNet model demonstrates higher reliance on the slow-stop mechanism on bends and narrow roads. Conversely, it maintains more stable control on wider road sections, resulting in fewer interventions.

FIGURE 8.10: Slow-stop activations for the EfficientNet model.

PointMamba Despite its relatively poor driving performance, the PointMamba model shows fewer slow-stop activations than expected (Figure 8.11). However, portions of the drive required manual intervention, limiting the number of activations recorded. The model continues to struggle on mild bends and narrow straight segments, where it depends heavily on the slow-stop node to maintain safe operation.

FIGURE 8.11: Slow-stop activations for the PointMamba model.

Swin The Swin model exhibits a substantial number of slow and stop activations throughout the drive (Figure 8.12), particularly on high-speed sections with bends. This suggests that the model struggles to maintain a central position on the road, which increases the dependency on the safety mechanism.

FIGURE 8.12: Slow-stop activations for the Swin model.

VMamba Figure 8.13 indicates that VMamba requires fewer interventions at the bend leading to the third roundabout, demonstrating improved lane positioning compared to earlier models. However, narrow road segments still pose challenges, as indicated by spikes in slow and stop activations.

FIGURE 8.13: Slow-stop activations for the VMamba model.

DeiT As shown in Figure 8.14, the DeiT model demonstrates relatively stable road positioning and a lower frequency of slow-stop interventions compared to other vision-based models. However, the distribution of activations throughout the drive suggests that while the model is consistent, it still intermittently relies on the safety system.

FIGURE 8.14: Slow-stop activations for the DeiT model.

ViT The ViT model exhibits persistent slow-stop activations throughout the drive (Figure 8.15), indicating poor lane maintenance and potential vehicle misalignment. This continuous reliance highlights the model's limited capacity for stable autonomous control.

FIGURE 8.15: Slow-stop activations for the ViT model.

Joint LiDAR–Vision In Figure 8.16, the Joint LiDAR–Vision model displays increased slow-stop reliance in narrow road segments and during the return to the Stocklands office. As the LiDAR component did not sufficiently improve perception in this scenario, it is likely that the vision subsystem, affected by lighting conditions, contributed to the higher number of safety interventions on the return.

FIGURE 8.16: Slow-stop activations for the Joint LiDAR–Vision model.

VideoMAE As shown in Figure 8.17, the VideoMAE model exhibits frequent slow-stop activations throughout the entire drive. This behaviour aligns with the model's characteristic swaying motion, where the safety system intervenes repeatedly to prevent disengagements. The consistent activation pattern highlights the model's instability in maintaining a centred trajectory during operation.

FIGURE 8.17: Slow-stop activations for the VideoMAE model.

VideoMamba Figure 8.18 shows that the VideoMamba model experiences fewer slow-stop activations than several other models, indicating a reduced dependence on the safety mechanism. Although this suggests improved driving stability, Chapter 7 noted that the model exhibited a high number of disengagements and manual interventions. This implies that, in many cases, the rapid and unstable responses of the model prevented the slow-stop node from acting in time to correct unsafe behaviour.

FIGURE 8.18: Slow-stop activations for the VideoMamba model.

Sequence ViM As illustrated in Figure 8.19, the Sequence ViM model exhibits a high number of slow-stop activations, particularly along narrow road sections and at bends. This indicates a strong dependence on the safety system to compensate for unstable lateral control and oscillatory steering behaviour. The frequent interventions highlight the model's difficulty in maintaining a consistent lane position, reinforcing observations from Chapter 7 regarding its sensitivity to dynamic inputs.

FIGURE 8.19: Slow-stop activations for the ViM Sequence model.

Sequence VMamba The Sequence VMamba model, shown in Figure 8.20, records fewer slow-stop activations compared to several other temporal models, despite exhibiting noticeable swaying behaviour. This is mainly attributed to the high number of disengagements and manual interventions observed during testing, where the safety system was unable to respond quickly enough to mitigate unsafe conditions. Consequently, while activations appear to be reduced, they reflect model instability rather than improved control performance.

FIGURE 8.20: Slow-stop activations for the VMamba Sequence model.

8.4.1.3 Without Slow-Stop Node

This section presents the map-based performance metrics of models evaluated without the slow-stop safety node. These results enable a direct assessment of each model's inherent stability, lane-keeping capability, and speed regulation in the absence of external corrective interventions.

PilotNet Figures 8.21c and 8.21b demonstrate that the PilotNet model without the slow-stop mechanism performs comparably to the average results reported in Chapter 5. However, Figure 8.21d shows increased lateral deviation from the centre line, which leads to a rise in both interventions and disengagements, as illustrated in Figure 8.21e. These findings suggest that while PilotNet can maintain basic control autonomously, it is more prone to safety-critical deviations in the absence of corrective mechanisms.

(A) PilotNet driving mode without slow-stop node.

(B) PilotNet speed without slow-stop node.

FIGURE 8.21: PilotNet model tested without the slow-stop safety system, including driving mode and speed.

(C) PilotNet steering smoothness without slow-stop node.

(D) PilotNet distance from centreline without slow-stop node.

(E) PilotNet interventions without slow-stop node.

FIGURE 8.21: PilotNet model tested without the slow-stop safety system, including steering smoothness, path deviation and interventions.

EfficientNet The EfficientNet model exhibits lower average speeds on wider road segments compared to its performance in Chapter 5 (Figure 8.22b). Furthermore, Figure 8.22d shows several peaks in lateral distance error, some of which align with intervention and disengagement events (Figure 8.22e). The remaining disengagements likely correspond to instances where the model maintained lane position but approached the road edge at an unsafe steering angle, indicating limitations in angular control rather than positional drift.

(A) EfficientNet driving mode without slow-stop node.

(B) EfficientNet speed without slow-stop node.

(C) EfficientNet steering smoothness without slow-stop node.

(D) EfficientNet lateral distance without slow-stop node.

(E) EfficientNet interventions without slow-stop node.

FIGURE 8.22: EfficientNet model tested without the slow-stop safety system, including driving mode, speed, steering smoothness, path deviation and interventions.

ViM The ViM model without the slow-stop node exhibits similar speed and steering smoothness to its Chapter 5 counterpart (Figures 8.23b and 8.23c). However, Figure 8.23d shows substantially greater lateral deviation from the centre line. Although these deviations do not always coincide with interventions and disengagements (Figure 8.23e), they suggest that the model consistently adopts a suboptimal approach angle to the road. Although this behaviour is tolerable on wider roads, it demonstrates that the slow-stop safety system contributes meaningfully to maintaining consistent lane centring.

(A) ViM driving mode without slow-stop node.

(B) ViM speed without slow-stop node.

(C) ViM steering smoothness without slow-stop node.

(D) ViM lateral distance without slow-stop node.

FIGURE 8.23: ViM model tested without the slow-stop safety system, including driving mode, speed, steering smoothness and path deviation.

(E) ViM interventions without slow-stop node.

FIGURE 8.23: ViM model tested without the slow-stop safety system, including interventions.

8.4.1.4 Parking, Reversing, and Exiting Parallel Bays

Using the evaluation metrics established in Chapter 5, the models were evaluated in structured parking scenarios, including parking within bays, reversing, and exiting parallel bays. These evaluations were conducted with and without the integration of the slow-stop safety node. The results are summarised in Table 8.5.

The results indicate that both ViM and PilotNet benefit from the inclusion of the safety system, particularly in tasks involving reversing and exiting parking bays. This suggests that the safety node provides meaningful corrective interventions that enhance the performance of models that exhibit moderate instability in low-speed manoeuvres. In contrast, EfficientNet shows minimal improvement, except in reversing, which reinforces the observation that the safety system is not universally beneficial and may provide limited value when the underlying model already performs reliably.

These findings highlight that, while safety mechanisms can support low-level manoeuvres, their influence is model-dependent. The goal should remain the development of inherently robust models rather than relying extensively on external systems to mitigate failure cases.

Model	ViM	ViM (without)	PilotNet	PilotNet (without)	EfficientNet	EfficientNet (without)
Pullin Score	3.00 ± 0.68	2.00 ± 1.55	2.42 ± 1.31	1.25 ± 0.5	2.00 ± 1.63	2.00 ± 1.91
Pullin Left (%)	85.71	66.67	83.33	75	87.5	85.71
Pullin Right (%)	14.29	33.33	16.67	25	12.5	14.29
Pullin Known Bays (%)	57.14	66.67	66.67	75	75	85.71
Pullin Unknown Bays (%)	42.86	33.33	33.33	25	25	14.29
Reverse Score	1.85 ± 0.38	1.88 ± 0.35	1.33 ± 0.89	1.00 ± 1.00	1.93 ± 0.27	1.00 ± 1.00
Reverse Left (%)	84.62	75	76.92	66.67	93.33	85.71
Reverse Right (%)	15.38	25	23.08	33.33	6.67	14.29
Reverse Known Bays (%)	46.15	50	46.15	33.33	64.29	57.14
Reverse Unknown Bays (%)	53.85	50	53.85	66.67	35.71	42.86
Pullout Score	3.93 ± 0.27	3.33 ± 1.03	3.50 ± 1.08	2.00 ± 1.73	3.78 ± 0.43	3.86 ± 0.38
Pullout Left (%)	85.71	66.67	81.82	66.67	93.33	85.71
Pullout Right (%)	14.29	33.33	18.18	33.33	6.67	14.29
Pullout Known Bays (%)	50.00	33.33	45.45	66.67	85.71	85.71
Pullout Unknown Bays (%)	50.00	66.67	54.55	33.33	14.29	14.29

TABLE 8.5: Parking, reverse and pullout driving results.

8.4.1.5 Lane Following — Road Features

Table 8.6 presents the influence of the slow-stop safety node on autonomous driving performance in key road characteristics. As outlined in Chapter 5, several performance metrics are affected by the intervention of the safety system; therefore, this table isolates and highlights these effects.

The results indicate that the integration of the slow-stop node can have both positive and negative impacts depending on the underlying stability of the model. For example, EfficientNet, which already demonstrated good performance, experienced a reduction in overall score when the safety system was enabled, suggesting that system intervention may occasionally impede well-performing models. In contrast, PilotNet, a comparatively less stable model, showed a marginal improvement, demonstrating that weaker models may benefit from corrective safety mechanisms.

ViM did not show notable changes in overall performance, indicating that the safety system neither improved nor hindered its ability to traverse the various features of the road. These findings suggest that, while the slow-stop node can enhance robustness for less reliable models, its utility is limited when applied to models with inherently stable lane-following behaviour.

Overall, the results underscore the importance of prioritising improvements in model consistency and predictive stability rather than relying heavily on external safety interventions to mitigate failure modes.

Model	PilotNet (without)	PilotNet	EfficientNet (without)	EfficientNet	ViM (without)	ViM
Roundabout 1	3	2.00 ± 1.00	3	3.00 ± 0.00	3	2.67 ± 0.58
Snaking Road	2	1.33 ± 0.58	0	1.00 ± 1.00	2	1.67 ± 0.58
Roundabout 2	3	3.00 ± 0.00	3	2.67 ± 0.58	3	3.00 ± 0.00
Intersection 1	2	1.33 ± 0.58	2	2.00 ± 0.00	2	2.00 ± 0.00
Dual lane unmarked 1	1	1.67 ± 0.58	2	2.00 ± 0.00	2	1.33 ± 1.15
Intersection 2	0	0.67 ± 0.58	1	0.33 ± 0.58	2	1.33 ± 0.58
Carpark	2	2.67 ± 1.53	2	2.67 ± 0.58	2	3.67 ± 1.15
Intersection 3	1	0.33 ± 0.58	2	1.33 ± 0.58	2	1.33 ± 1.15
Dual lane unmarked 2	1	1.33 ± 0.58	2	2.00 ± 0.00	1	1.67 ± 0.58
Intersection 4	0	1.67 ± 0.58	2	2.00 ± 0.00	1	2.00 ± 0.00
Roundabout right	1	1.33 ± 0.58	3	1.00 ± 1.00	2	2.00 ± 0.00
Dual lane marked	2	2.00 ± 0.00	2	1.00 ± 1.00	2	2.00 ± 0.00
Roundabout 3	1	2.67 ± 0.58	3	2.67 ± 0.58	3	2.67 ± 0.58
Roundabout 4	3	2.67 ± 0.58	3	2.67 ± 0.58	2	2.33 ± 1.15
Total Score (/36)	22	$\mathbf{24.67 \pm 0.59}$	30	$\mathbf{26.33 \pm 0.46}$	29	$\mathbf{29.67 \pm 0.54}$

TABLE 8.6: Landmark driving results.

8.4.2 Low-Level Detection

The following figures illustrate the model's low-level prediction outputs on previously unseen data. Figure 8.24a demonstrates the model's ability to accurately identify and track the road centre line along a straight segment. Minor misclassifications occur on driveway and grass regions; however, due to subsequent filtering and correction steps in the detection pipeline, these have negligible impact on overall performance. The two blue patches represent tiles reclassified based on neighbouring classifications. The yellow tile corresponds to the detected centre of the current row; although a single error is present, the RANSAC regression method successfully mitigates its effect. The light blue line denotes the true image centre line, the red line shows the regression fit, and the yellow marker indicates the estimated road centre. In this case, the predicted centre aligns closely with the true road position in the original image.

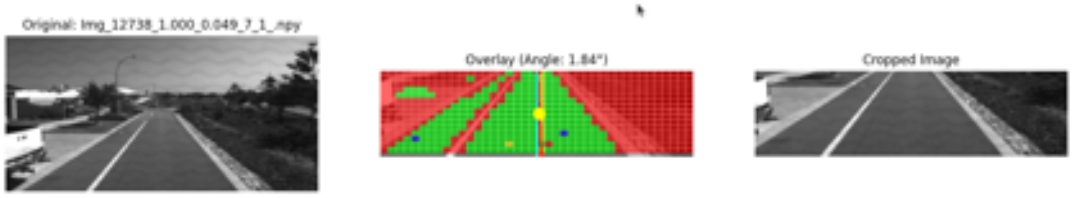
Figure 8.24b presents model performance on a curved road section. The predicted centre line adheres well to the road geometry, with a detected steering angle of 12.22 degrees, which remains within the allowable limits for image shifting. The resulting lateral shift to the left supports smoother vehicle alignment through the bend and reduces the likelihood of edge-related disengagements.

In Figure 8.24c, the system correctly identifies an anomalous object on the surface of the road, represented by a thin white line. This object could potentially compromise vehicle

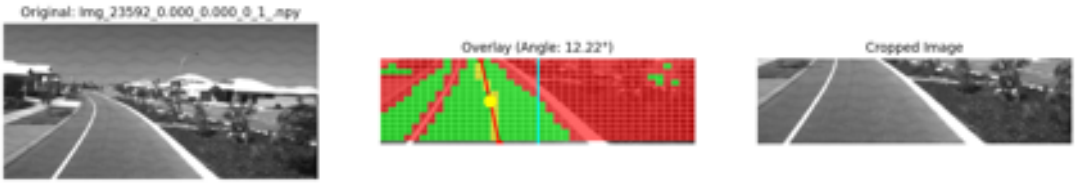
safety if undetected. In particular, the safety LiDAR system fails to recognise this anomaly, thereby demonstrating the complementary redundancy provided by the vision-based detection component. The current implementation does not include an avoidance or response mechanism, and potential solutions are discussed in the discussion section.

Figure 8.24d highlights a failure case in which the model struggles to reliably detect a drivable road section under strong shadowing conditions. Although sufficient information remains to estimate the centre of the road, the confidence and consistency of the model are notably reduced. Although shadow-based augmentation was included during training, the dataset likely remains insufficiently diverse. Furthermore, the shadowed road surface appears visually similar to darker grass regions, leading to misclassification. Incorporating RGB imagery or could enhance local texture discrimination and robustness in these scenarios.

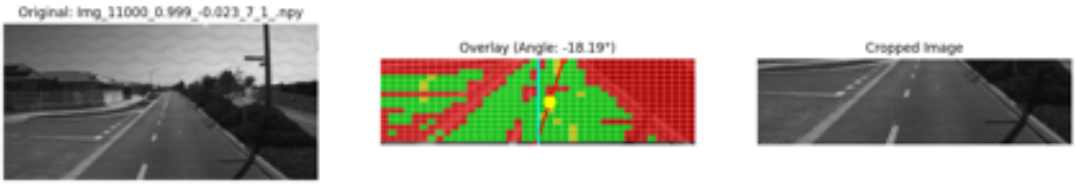
Figure 8.24e illustrates a complete failure to detect the surface of the road, primarily due to excessive illumination that causes a loss of contrast in grey scale images. This exemplifies a core limitation of grey scale-based perception under varying lighting conditions. In this instance, the system correctly disengages due to the excessive steering angle and insufficient road patch detection.



(A) Low-level detection on a straight road.

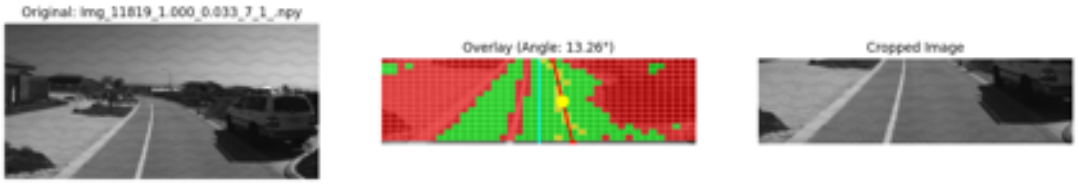


(B) Low-level detection on a curved road.

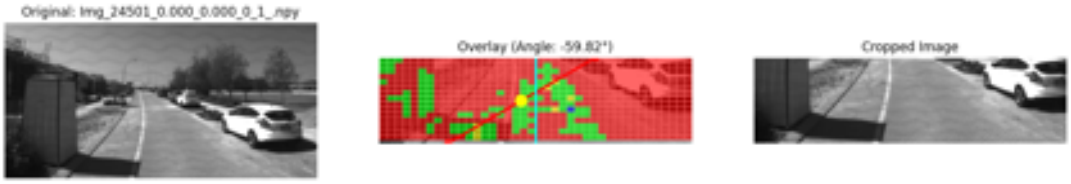


(C) Detection of an anomaly on the road surface.

FIGURE 8.24: Examples of low-level prediction at key segments of the driving route.



(D) Reduced detection accuracy under shadowed conditions.



(E) Failure to detect road under high illumination.

FIGURE 8.24: Examples of low-level prediction at key segments of the driving route.

8.4.2.1 Driving Performance

During on-road testing of the low-level safety system, ambient lighting conditions were significantly brighter than those represented in the training dataset. As a result, the visual road detection module was activated for only 2–3% of the total driving time. At face value, this low activation rate might suggest reliable autonomous performance; however, a comparison between the time spent calculating steering angles and the total duration of the drive indicates that the model was not functioning as intended.

Subsequent analysis of the recorded drive data, including its reprocessing through the prediction network, confirmed that extreme lighting conditions adversely affected model performance. This further demonstrates the model's vulnerability to illumination variability and highlights a dependency on visual consistency for reliable operation.

Although additional experiments could have been conducted under more favourable lighting conditions, this outcome represents a critical failure in the robustness of the current implementation. Consequently, further refinement of the perception model and safety mechanisms is necessary. Proposed improvements and adaptation strategies to mitigate these limitations are discussed in the following section.

8.5 Discussion

The results demonstrate how safety systems can be implemented to improve both reliability and operational safety of the autonomous vehicle. This discussion expands upon those results and explores strategies for further improvement. It is organised into three sections: the first two sections focus on the performance of the slow stop node and the low-level detection system, while the third section discusses exploratory work aimed at enhancing driving performance through reinforcement learning.

8.5.1 Slow-stop Node

From Table 8.2, it is evident that the disabling of the slow-stop node generally leads to reduced driving performance, particularly in terms of autonomy, disengagements, and manual interventions. The only exception is the PointMamba model, which shows marginal improvements in autonomy and a reduction in interventions, albeit with an increase in disengagements. This behaviour reflects the variable nature of PointMamba: when initialised from a favourable position, the model can perform reliably, provided that its predictions remain stable. However, even in such instances, the slow-stop node is only able to compensate for limited instabilities in lower performing models.

Table 8.3 further supports this observation. Excluding PointMamba, manual interventions in categories 1 and 4 increase when the slow-stop node is inactive. EfficientNet is the sole exception, showing a minor reduction in interventions without the node. This suggests that for most models, the slow-stop system plays a meaningful role in maintaining safety and continuity of operation, particularly during more critical intervention scenarios.

Tables 8.2 and 8.4 also report the proportion of driving time during which each model relies on the "slow" and "stop" components of the node. In most models, the slow component is engaged for approximately 7% of the drive. Notably, the Joint LiDAR–Vision model relies on it the least, indicating greater inherent performance. In contrast, the engagement of the stop component is more disruptive as it completely halts vehicle movement. Only the VMamba and DeiT models activate the stop function for less than 3% of the total drive.

Analysis of the map-based performance metrics shows that the slow-stop node is triggered more frequently during turning manoeuvres and on narrow road sections. This behaviour could be mitigated by redesigning the detection region using sloped or tapered boundaries rather than rectangular zones. However, such modifications would incur an additional computational cost. Alternatively, reducing the detection region size could lessen

its influence, although this would only be appropriate once model reliability improves to a sufficient degree.

Table 8.5 demonstrates that the slow-stop node also improves performance during structured parking and low-speed manoeuvres. However, this benefit is not universal: EfficientNet shows limited improvement, reinforcing that the effectiveness of the safety system depends on the model.

Similarly, Table 8.6 indicates that while the slow-stop node can reduce performance at key landmarks due to its direct interference with model outputs, it simultaneously compensates for poor predictions and contributes to overall drive stability. These outcomes emphasise that future development should prioritise improving model robustness rather than increasing dependence on safety mechanisms. A more reliable model would allow for a less intrusive and more adaptive safety system.

In total, the DeiT model engages the slow-stop node for 8.94% of the total drive, compared to 9.53% for VMamba. This positions DeiT as the most self-reliant model overall, consistent with its strong autonomy performance in Chapter 5, albeit with the trade-off of increased architectural complexity. With further data augmentation and generalisation-focused training, reliance on the slow-stop system is expected to decrease across all models. Nonetheless, current results confirm that the slow-stop node remains a critical component for ensuring stability and safety in the present system implementations.

8.5.2 Low-Level Detection

The low-level detection system showed promise, with initial results indicating reliable differentiation between the road surface and surrounding obstacles on very small images. However, as discussed in Section 6.4.2.1, on-road performance was limited, with only a small number of activations once all safety restrictions were applied. Offline analysis of the ROSbag data revealed that the patch-based detection approach was highly sensitive to changes in lighting. This limitation arises due to the limited contextual information available within each patch, which amplifies the effect of minor visual variations. Despite expanding the dataset and retraining the model, only marginal improvements were observed in regions affected by shadows, bright driveways, or high-contrast surfaces. Misclassifications in these scenarios remained prevalent as shown in figure 8.24e. Although further data collection on the order of one million additional samples may enhance robustness, improvements are not guaranteed due to the inherent limitations of the method.

A potential refinement would be to incorporate neighbouring image tiles as additional contextual information, allowing the model to infer road presence based on surrounding spatial cues. This approach aligns with human visual reasoning, where contextual awareness supports accurate interpretation of ambiguous visual inputs.

Although the limited on-road dataset suggests that the system functions as intended under controlled conditions, additional testing is required in various lighting environments. Furthermore, optimisation of the image shift speed and distance calibration parameters is necessary to ensure stable integration with the real-time control system.

8.5.2.1 Future Adaptations

Once the system has been enhanced to reliably detect road segments under variable lighting and environmental conditions, the perception–control pipeline can be extended to support intelligent navigation through unsafe or partially obstructed road environments. The current detection framework is capable of distinguishing between drivable (road) and non drivable (unsafe) areas. Unsafe regions may include surface disturbances, unidentified obstacles, potholes, lane markings, or parked vehicles. Although lane markings are treated as non drivable in this work, future implementations could introduce a third classification *semi-drivable*, to accommodate cases such as oncoming traffic avoidance or temporary lane encroachment when no viable alternative path exists.

To operate this framework, the vehicle's behaviour can be defined based on the spatial distribution of unsafe regions relative to the vehicle's trajectory. The following decision structure is proposed:

- **No unsafe regions**: Continue normal autonomous driving.
- **Unsafe regions only at road edges**: Maintain a central lane position.
- **Large unsafe region on one edge**: Drive cautiously closer to the opposite side of the road.
- **Unsafe region in the centre of the road**: Traverse the region only if its width is smaller than the vehicle's wheelbase; otherwise, stop.
- **Offset unsafe region spanning from the centre**: Estimate the region's centre line and proceed if the width remains below the wheelbase threshold; otherwise, stop.
- **Two unsafe regions on opposite sides**: Proceed only if the remaining central gap exceeds the vehicle width; otherwise, stop.

- **No safe regions**: Perform a controlled stop.

This behavioural logic provides a deterministic foundation for low-level path selection in irregular environments. Future iterations may incorporate probabilistic reasoning, reinforcement learning, or semantic segmentation to improve decision robustness and generalisation to unseen scenarios.

8.5.2.2 Semantic Segmentation as an Alternative

Semantic segmentation provides a potential alternative to patch-based detection by offering pixel-level classification of road surfaces, obstacles, and boundaries. However, this approach was not adopted in the current work due to high computational cost and slower inference speeds, which make it unsuitable for real-time, resource-constrained platforms. Preliminary experiments by a collaborating student using SAM2 (Segment Anything Model v2) demonstrated high-quality segmentation performance [268, 269], but required model reduction to run on constrained systems and was only operational for a small subset of images. If future developments enable lightweight, real-time segmentation with minimal computational overhead, integration into the perception pipeline may be viable.

However, the proposed low-level detection system remains advantageous due to its speed, simplicity, and suitability for training with limited resources. With further improvements in generalisation and robustness to lighting variability - as discussed previously - the method has the potential for deployment in real-world, resource-constrained autonomous driving systems.

8.5.3 Exploratory Reinforcement Learning

As a stretch goal to improve the autonomous driving system, this work explored the potential of using reinforcement learning (RL) to fine-tune driving behaviour. Although RL has been successfully applied in simulated driving environments [270, 271, 272], its direct application to real-world autonomous vehicles is limited due to safety restrictions and the need to obey traffic rules.

A more feasible approach is to first train a model using imitation learning and then apply RL as a fine-tuning mechanism. This allows the vehicle to maintain safe baseline behaviour while exploring improvements, particularly in areas where the original training data may contain suboptimal actions. In this work, a prototype evaluation model was developed to predict the performance of the vehicle given its current state, including position, speed, steering,

and sensory inputs. The predicted performance could then be used to adjust ground truth actions to encourage safer and more optimal driving paths.

Although early implementations of this pipeline have been developed, the system has not yet been evaluated and there is no performance data available.

Further work could include training an RL model in simulated environments followed by transfer learning to real vehicles, although previous studies suggest that this remains a challenging problem.

Overall, this exploration highlights a promising direction for future research: combining imitation learning with reinforcement learning to improve real-world autonomous driving performance in a controlled and safe manner.

8.6 Conclusion

This chapter investigated the integration of deterministic safety mechanisms and performance optimisation strategies to enhance the robustness of the autonomous driving pipeline developed in Chapters 4 and 5. The aim was to mitigate inconsistent model behaviour, reduce reliance on human interventions, and improve system resilience in real-world suburban environments.

Among the systems evaluated, the slow-stop safety node proved the most effective. It consistently reduced severe deviations from the lane centre and prevented unsafe manoeuvres without introducing substantial delays and actually demonstrated improved drive times in trip duration. This confirms the value of independent fail-safe systems in complementing stochastic deep learning models. However, its reliance on regenerative braking within the current EasyMile platform limited the responsiveness of emergency stops, highlighting the need for future vehicles to incorporate active braking via CANbus for improved decision latency and safety at higher speeds.

The low-level vision-based road detection module demonstrated an ability to classify road regions accurately under controlled conditions, but its live deployment yielded minimal activations due to strong susceptibility to lighting variability and insufficient contextual awareness in patch-based analysis. As such, it has not yet contributed to improved autonomy, but it remains a promising foundation for future road segmentation and drivable-space management.

8.6.1 Novel Contributions

This work presents one of the first integrations of deterministic safety logic, LiDAR-based slow-stop control, and state space model (VMamba/ViM) perception within a real-world autonomous driving platform. To the authors' knowledge, this also represents the smallest successful deployment of a Vision Mamba-based model for image based classification for use in vehicle driving application. The chapter contributes:

- A practical framework for coupling deep learning perception with rule-based safety overrides.
- Empirical evidence that simple safety systems can substantially improve intervention rates without compromising usability.
- A critical assessment of low-level road detection under real-world lighting variability.
- Early discussions of reinforcement learning for post-training optimisation of end-to-end driving models.

8.6.2 Future Work

Moving forward, several developments are required to improve reliability and support higher-speed autonomous operation:

- **Braking hardware**: Future vehicle platforms must support CANbus-enabled active braking to allow rapid execution of safety overrides without disengagements.
- **Low-level detection improvements**: Extend the binary road/non-road classifier to include semi-drivable regions such as loose gravel, shallow water, or debris-prone surfaces, which may not damage the vehicle but could induce slip at higher speeds.
- **Context-aware vision**: Incorporate spatial context (e.g., neighbouring tiles or semantic segmentation) to reduce lighting sensitivity and improve generalisation.
- **Reinforcement learning refinement**: Enhance the performance of the vehicle driving by using PPO reinforcement learning to encourage model exploration which will allow to model to drive better than the original human drivers.

In summary, while the autonomous system has not yet met full SAE Level 4 autonomy requirements, this chapter demonstrates meaningful progress toward safer, more reliable,

and generalisable autonomous driving through a combination of deterministic safeguards and learning-based enhancements.

Chapter 9

Conclusion

This thesis investigated the application of both mathematical model and artificial intelligence (AI) architectures, with a particular focus on recent State Space Models (SSMs), to enhance the performance and safety of autonomous vehicles (AVs). Chapter 4 demonstrated the influence of human interactions on mathematical model based systems in semi-structured environments, while Chapters 5 to 8 highlighted the potential of SSMs for reactive and proactive decision making, contributing to improved reliability and trust in autonomous driving systems. This final chapter summarises the comparative performance of the models, key findings, limitations, and directions for future research.

9.1 Final Comparison of AI Systems on Suburban Roads

Table 9.1 presents a comprehensive comparison of the AI models evaluated in Chapters 5 to 7. The Joint LiDAR–Vision model achieved the highest performance in autonomy and landmark navigation. However, it is substantially larger than most other models, which may limit its deployment on resource-constrained platforms. Furthermore, this model was evaluated solely on lane-following tasks, excluding parking, which may have affected the reported autonomy metrics.

In terms of reliability, VMamba recorded the lowest average number of disengagements, despite a slightly higher number of interventions. This characteristic improves rider comfort, as disengagements typically require abrupt heavy braking. Overall, the results establish a solid baseline for further improvement. Future enhancements, including expanded data augmentation, refined training procedures, and integration of advanced safety systems, are expected to yield significant gains.

Moreover, broader testing across varied driving scenarios and more complex tasks, such as parking manoeuvres, will be necessary to assess generalisation and real-world robustness.

Model	Autonomy (%)	Disengagements / Interventions	Total Parking Score (/10)	Landmark Score (/36)
PilotNet	90.07 ± 2.76	6.00/11.33	7.25 ± 1.09	24.67 ± 0.59
EfficientNet	93.93 ± 2.15	3.33/6.67	7.71 ± 0.78	26.33 ± 0.46
ViT	92.40 ± 2.00	2.00/6.67	8.16 ± 0.86	25.00 ± 0.44
Swin	94.51 ± 1.72	3.33/5.67	7.76 ± 0.69	27.67 ± 0.54
DeiT	95.87 ± 3.21	2.00/4.33	8.44 ± 0.82	28.33 ± 0.44
ViM	95.67 ± 3.27	2.67/3.33	$\mathbf{8.78 \pm 0.44}$	29.67 ± 0.54
VMamba	95.77 ± 2.73	**0.67/5.33**	8.16 ± 0.64	28.67 ± 0.41
PointMamba	84.49	4/32	–	17.00
Joint LiDAR–Vision	$\mathbf{97.31 \pm 2.23}$	3.00/2.67	–	$\mathbf{31.33 \pm 0.35}$
VideoMAE	80.60	7.00/22.00	4.58 ± 1.01	9.00
VideoMamba	89.05 ± 4.46	5.67/8.67	5.38 ± 0.88	25.33 ± 0.49
Sequence ViM	95.07	13.00/7.00	7.00 ± 0.96	23.00
Sequence VMamba	94.79 ± 4.33	7.00/6.33	7.59 ± 1.02	24.33 ± 0.54

TABLE 9.1: Comparison of all AI driving models across key performance metrics. Total Parking Score combines the scores from Pull-in, Reverse, and Pull-out (/10). Landmark Score is the total performance across major driving events (/36).

9.2 Overall Findings

The primary goal of this research was to evaluate how autonomous vehicles perceive and interact with their environment using mathematical and sequential learning models, and to assess the viability of SAE Level 4 autonomy. Lightweight tools were also developed to support control, perception, and safety in both suburban and campus environments.

Initial experiments showed that mathematical model approaches are feasible but encounter substantial challenges. Localisation using only 2D LiDAR is difficult in semi-structured environments due to limited unique features in a 2D plane, and conventional controllers often fail to understand human decision-making. Simplified control strategies combined with lightweight localisation yielded a more robust and stable system. A recovery module was introduced to ensure consistent and safe startup behaviour.

Subsequently, a mixture-of-experts framework was implemented for suburban driving, integrating classical and modern neural network architectures. The recently introduced Mamba SSM demonstrated strong performance at low computational cost and generalised reasonably well across different driving behaviours, even without extensive data augmentation. However, it was sensitive to variations in lighting, highlighting the gap between offline training and real-world deployment.

Further work focused on SSM-based models across different sensor modalities. LiDAR-only SSMs performance was unreliable as they were sensitive to perturbations and unfamiliar scenarios but showed future potential with better training. Generating synthetic 3D environmental models from LiDAR data was identified as a strategy to improve robustness. Combining LiDAR with vision data significantly improved performance, particularly around key landmarks. These results justify the added complexity and cost of joint LiDAR–vision systems.

Temporal feedback strategies were evaluated for further generalised performance. Metadata-based feedback showed limited real-world benefit due to over reliance on past commands, while temporal visual models inspired by video classification architectures exhibited latency issues in real-world conditions. Although practical gains were limited, these experiments highlighted gaps between simulation, dataset-based training, and deployment.

Final comparisons between data set-based evaluation and on-road performance demonstrated that substantial discrepancies can arise when models are deployed in real-world environments. Dataset evaluations often fail to capture system-level effects and data drift encountered during real-world operation, which can ultimately contribute to performance degradation or system failure.

The findings of this work emphasise the critical importance of real-world testing for validating autonomous driving algorithms. Such evaluation enables the identification of practical influences and operational constraints that are not adequately represented in simulation or offline data set testing, providing a more reliable assessment of true system performance.

Finally, layered safety mechanisms were incorporated to ensure safe operation in the real world. A slow/stop node effectively overrode unsafe neural network decisions without severely disrupting driving flow, and a vision-based module provided additional hazard detection. Despite low-resolution grey scale input, SSM-based models achieved fast, accurate inference, although performance was still sensitive to lighting changes. Reinforcement learning was suggested as a future avenue to surpass human driving baselines by optimising reward structures for proactive behaviour.

Collectively, these studies demonstrate that end-to-end models can be deployed safely under normal driving conditions and identify the technical steps necessary for progression toward SAE Level 4 autonomy.

9.3 Limitations

Despite careful design, real-world testing imposed several constraints. Some evaluation metrics, though based on a defined scoring framework, remained partially subjective, introducing the potential for unconscious bias. Efforts were made to mitigate this by incorporating multiple complementary quantitative metrics wherever possible.

Additional test drives at the Amberton Beach site under diverse lighting, weather, and traffic conditions would have improved the robustness of the evaluation. However, time constraints, vehicle maintenance requirements, and the defined research scope limited each model to three test runs. This was sufficient for comparative analysis, but insufficient to claim fully autonomous capability.

Testing was restricted to a single primary route. Although segmentation of this route could theoretically be used to assess generalisation, the unique characteristics of each road segment limited its effectiveness. Future research should involve testing across wider geographic areas to better assess generalisation and suitability for Level 4 autonomy.

The majority of the route consisted of single-lane roads, minimising interaction with oncoming traffic. Although some dual-lane segments were included, the overall environment remained relatively controlled. More complex scenarios involving multiple vehicles, intersections, and multi-lane traffic should be evaluated before deployment at scale.

Finally, occasional spikes in accuracy were observed in the data recorded from the GPS + RTK positioning system. These may have contributed to distance discrepancies reported in Chapters 4–8. Improved spatial covariance estimation and sensor uncertainty modelling are recommended for future trajectory analysis.

9.4 Future Research

Autonomous driving remains an intricate and multifaceted challenge, and this work opens several promising directions for further investigation. The author believes that a modularised system, where individual models are responsible for specialised driving behaviours, offers greater potential than a single monolithic architecture. Such a system would require a high-level decision-making module capable of selecting the appropriate behavioural model at any given moment. This also introduces questions about how users can interact intuitively and safely with the system to guide it toward their desired destination.

Safety remains a critical priority in AI-based autonomous systems. While AI models can exhibit stochastic behaviour, safety mechanisms must be incorporated without severely compromising performance. Future models should be developed hand-in-hand with advanced simulation environments to allow rapid prototyping and safe experimentation. This naturally highlights the importance of *transfer learning*, which enables models trained in simulated or alternate environments to adapt effectively to real-world conditions while retaining learned behaviours. Chapters 4, 5 and 7 explored imitation learning and Chapter 6 presented an adjustment to integrate reinforcement learning; however, true reinforcement learning in real vehicles is currently impractical due to safety and mechanical risks. Robust simulation environments paired with strong transfer learning techniques will therefore be essential for future progress.

Beyond model development, broader systemic and societal considerations must also be addressed. With increasing interest and adoption of autonomous vehicles, research has already begun in logistics, digital twins, and fleet optimisation. Once the technology matures, coordinated multi-vehicle systems will be essential to efficiently manage ride requests, route scheduling, and resource allocation across entire regions.

Several other avenues for future research include:

- **Generalisation to new environments:** Current models are tailored to a specific geographic area. Future work should aim to generalise the system to operate reliably in varied weather, road layouts, and traffic conditions.

- **Enhanced human–vehicle interaction:** Expanding interaction modalities through voice commands, speech understanding, and natural language interfaces (e.g., large language models) could reduce reliance on manual controllers. However, such systems must be designed to mitigate misinterpretations, malicious prompts, and current issues with latency.

- **Security and robustness:** As AI systems become more prevalent, new methods of adversarial attacks are being found. Recent work (e.g., [273]) demonstrates how AI vision models can be manipulated, underscoring the need to harden systems against cyber physical threats.

- **Predictive modelling and future perception:** A promising next step is to integrate a future prediction module. The concept involves forecasting future observations and comparing them with actual sensor input to detect anomalies, similar to how human peripheral vision detects unexpected motion. Vision-based Mamba models such as

VMRNN [274] already show strong results in image sequence prediction. Preliminary adaptation of such models for autonomous driving has been explored in simulation, but current predictions only capture faint scene outlines. Improving these predictive capabilities will be vital for proactive planning, such as early lane adjustments or speed adaptations.

9.5 Final Remarks

The works presented in this thesis form an initial step toward achieving SAE Level 4 autonomy. Although the performance of the proposed models is still short of the required standards for fully autonomous driving, the results demonstrate promising potential and highlight clear avenues for future improvement. Each model contributes valuable insights, whether through architectural design, sensor fusion, or training methodologies, and collectively they provide a foundation upon which more robust and capable systems can be developed. Ultimately, while this research does not deliver a complete solution, it lays the groundwork and direction for subsequent advancements in autonomous vehicle perception and control.

Bibliography

[1] R. Harrington, C. Senatore, J. Scanlon, and R. Yee, "The role of infrastructure in an automated vehicle future," *Bridge*, vol. 40, pp. 48–55, 06 2018.

[2] "Car accidents survey amp; statistics 2021: Car research amp; statistics - budget direct™," Oct 2019. [Online]. Available: https://www.budgetdirect.com.au/car-insurance/research/car-accident-statistics.html#10-common-causes-of-car-accidents

[3] "The most common causes of car accidents in australia | qbe au," 2022. [Online]. Available: https://www.qbe.com/au/news/the-most-common-causes-of-car-accidents

[4] EasyMile, "On-demand and fully driverless shuttle serves a norwegian technology park," 2021. [Online]. Available: https://easymile.com/news/demand-and-fully-driverless-shuttle-serves-norwegian-technology-park

[5] T. E. A. R. Lab, "Iseauto – self-driving shuttle," 2025, accessed: 2024-08-23. [Online]. Available: https://iot.ttu.ee/projects/iseauto/

[6] "Easymile ez10," 2022. [Online]. Available: https://en.wikipedia.org/wiki/EasyMile_EZ10

[7] H. Abdullah, "Baidu will launch its autonomous buses in japan by early 2019," 2018. [Online]. Available: https://medium.com/the-21st-century/baidu-will-launch-its-autonomous-buses-in-japan-next-year-725fde19708a

[8] Navya, "Self-driving shuttle for passenger transportation," NA. [Online]. Available: https://navya.tech/en/solutions/moving-people/self-driving-shuttle-for-passenger-transportation/

[9] S. Bus, "Olli debuts in italy: Turin deploys the 3d-printed driverless shuttle," 2020. [Online]. Available: https://www.sustainable-bus.com/its/olli-debuts-in-italy-turin-deploys-the-3d-printed-driverless-shuttle/

[10] 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.

[11] T. Drage, K. L. Lim, J. E. H. Koh, D. Gregory, C. Brogle, and T. Bräunl, "Integrated modular safety system design for intelligent autonomous vehicles," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 258–265.

[12] CiA, "History of can technology," 2022. [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-history/

[13] Z. Lai, K. Quirke-Brown, X. Kong, L. Le, and T. Bräunl, "Shuttle bus performing common maneuvers in roundabout-style suburban residential area using grayscale camera-based end-to-end driving," 2025, manuscript submitted for publication in *IEEE Transactions on Intelligent Vehicles*.

[14] E. P. Company, "What is an optical encoder?" https://www.encoder.com/article-what-is-an-optical-encoder, 2025, accessed: 2025-09-30.

[15] T. Braunl, *Embedded robotics : from mobile robots to autonomous vehicles with Raspberry Pi and Arduino*, fourth edition. ed. Singapore: Springer, 2022.

[16] T. Bräunl, "Mobile robots: Slide deck 08 — localization," Lecture slides, School of Electrical, Electronic and Computer Engineering, University of Western Australia, 2025, accessed via course material, slide deck 08 on localization.

[17] E. D. Kaplan and C. Hegarty, *Understanding GPS/GNSS: Principles and applications*. Artech house, 2017.

[18] S. N. NJANE, "Development of a low cost ntrip-based rtk-gnss base station for precise positioning," *Engineering in Agriculture, Environment and Food*, vol. 17, no. 2, pp. 74–81, 2024.

[19] "Introduction," 2022. [Online]. Available: https://docs.emlid.com/reachrs2/

[20] S. Systems, "Ellipse series," 2022. [Online]. Available: https://www.sbg-systems.com/products/ellipse-series/

[21] J. Gutierrez, R. Gilabert, E. Dill, G. Hernandez, D. Kaeli, and P. Closas, "Multipath mitigation via clustering for position estimation refinement in urban environments," in *Proceedings of the ION 2024 Pacific PNT Meeting*, 2024, pp. 556–568.

[22] C. Iclodean, N. Cordos, and B. O. Varga, "Autonomous shuttle bus for public transportation: A review," *Energies*, vol. 13, no. 11, 2020. [Online]. Available: https://www.mdpi.com/1996-1073/13/11/2917

[23] F. Girrbach, J. D. Hol, G. Bellusci, and M. Diehl, "Optimization-based sensor fusion of gnss and imu using a moving horizon approach," *Sensors*, vol. 17, no. 5, p. 1159, 2017.

[24] Velodyne, "What is lidar?" 2022. [Online]. Available: https://velodynelidar.com/what-is-lidar/

[25] 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.

[26] H. Hu, Z. Liu, S. Chitlangia, A. Agnihotri, and D. Zhao, "Investigating the impact of multi-lidar placement on object detection for autonomous driving," 2022. [Online]. Available: https://arxiv.org/abs/2105.00373

[27] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, 2020.

[28] D. Wang, Z. Liu, S. Shao, X. Wu, W. Chen, and Z. Li, "Monocular depth estimation: A survey," in *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*, 2023, pp. 1–7.

[29] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[30] "Flir grasshopper firewire camera model 590-1," https://www.eureca.de/590-1-Flir_Grashopper_Firewire_02170235.html, 2025, accessed: 2025-08-27.

[31] T. Bräunl, *Embedded Robotics: From Mobile Robots to Autonomous Vehicles with Raspberry Pi and Arduino*, 4th ed. Springer Singapore, 2022, includes application examples using Raspberry Pi and Arduino.

[32] I. TIER IV and contributors, "Awsim: Open sourced digital twin simulator for autoware," GitHub repository, 2025, accessed: 2025-06-09, latest release v2.0.0 :contentReference[oaicite:0]index=0. [Online]. Available: https://github.com/tier4/AWSIM

[33] Carla, "Carla simulator," 2018. [Online]. Available: https://carla.readthedocs.io/projects/ros-bridge/en/latest/

[34] O. Robotics, "Recording and playing back data — ros 2 tutorials: Beginner cli tools," Online tutorial, 2025, accessed: 2025-02-18. [Online]. Available: https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Recording-And-Playing-Back-Data/Recording-And-Playing-Back-Data.html

[35] Fennec, "Spotlight: How uwa's rev lab powers autonomous vehicle technology with foxglove," Blog post on Foxglove.dev, Jan 2025, accessed: 2025-10-18. [Online]. Available: https://foxglove.dev/blog/spotlight-how-uwas-rev-lab-powers-autonomous-vehicle-technology-with-foxglove

[36] S. Macenski, F. Martin, R. White, and J. G. Clavero, "The marathon 2: A navigation system." Institute of Electrical and Electronics Engineers Inc., 10 2020, pp. 2718–2725.

[37] G. Granosik, K. Andrzejczak, M. Kujawinski, R. Bonecki, L. Chlebowicz, B. Krysztofiak, K. Mirecki, and M. Gawryszewski, "Using robot operating system for autonomous control of robots in eurobot, erc and robotour competitions," *Acta Polytechnica CTU Proceedings*, vol. 6, pp. 11–17, 2016.

[38] P. Marín, A. Hussein, D. Martín Gómez, and A. de la Escalera, "Global and local path planning study in a ros-based research platform for autonomous vehicles," *Journal of Advanced Transportation*, vol. 2018, pp. 1–10, 02 2018.

[39] D. Lu, E. Marder-Eppstein, and D. Hershberger, "Costmap 2d," 2018. [Online]. Available: http://wiki.ros.org/costmap_2d

[40] A. Filotheou, E. Tsardoulias, A. Dimitriou, A. Symeonidis, and L. Petrou, "Quantitative and qualitative evaluation of ros-enabled local and global planners in 2d static environments," *Journal of Intelligent & Robotic Systems*, vol. 98, no. 3-4, pp. 567–601, oct 2019.

[41] Open Source Navigation2 Documentation Team, "Getting started," https://docs.nav2.org/getting_started/index.html#getting-started, 2025, accessed: 2025-06-05.

[42] D. Lu and M. Ferguson, "costmap 2d," 2018. [Online]. Available: http://wiki.ros.org/costmap_2d

[43] O. robotics, "Roscon 2019 macau: On use of slam toolbox," 2019. [Online]. Available: https://vimeo.com/378682207

[44] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[45] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[46] S. B. Nashed, "A brief survey of loop closure detection: A case for rethinking evaluation of intelligent systems," *ML Retrospectives: NeurIPS 2020 Workshop*, 2020, camera-ready version. [Online]. Available: https://ml-retrospectives.github.io/neurips2020/camera_ready/21.pdf

[47] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: his life, work, and legacy*, 2022, pp. 287–290.

[48] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[49] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Understanding nodes," 2022. [Online]. Available: https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html

[50] ——, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[51] ——, "Understanding actions," 2025. [Online]. Available: https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Actions/Understanding-ROS2-Actions.html

[52] S. Macenski, R. White, J. Wallace, and A. Merzlyakov, "Nav2," 2020. [Online]. Available: https://navigation.ros.org/index.html

[53] X. Li, Q. Xu, X. Li, H. Xin, Y. Yuan, Z. Shen, and Y. Zhou, "Improving ppp-rtk-based vehicle navigation in urban environments via multilayer perceptron-based nlos signal detection," *GPS Solutions*, vol. 28, no. 29, 2023.

[54] G. Dissanayake, S. B. Williams, and H. F. Durrant-Whyte, "Map management for efficient simultaneous localization and mapping (slam)," *Autonomous Robots*, vol. 12, no. 3, pp. 267–286, 2002.

[55] T. Raack, "Autonomous vehicle technology – sensor fusion and extended kalman filters for object tracking," Online article, Sep 2017, accessed: 2025-05-20. [Online]. Available: https://taylor.raack.info/2017/09/autonomous-vehicle-technology-sensor-fusion-and-extended-kalman-filters-for-object-tracking/

[56] T. Bräunl, *Mobile Robot Programming: Adventures in Python and C*, 2nd ed. Springer International Publishing, 2023, 191 pages.

[57] M. Acharya, "Intuition behind the particle filter," Blog post on MathWorks Autonomous Systems Blog, Jan 2023, accessed: 2025-05-03. [Online]. Available: https://blogs.mathworks.com/autonomous-systems/2023/01/05/intuition-behind-the-particle-filter/

[58] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.

[59] S. Macenski and S. Singh, "Regulated pure pursuit controller," 2021. [Online]. Available: https://navigation.ros.org/configuration/packages/configuring-regulated-pp.html

[60] S. Macenski, "Smac planner," 2021. [Online]. Available: https://navigation.ros.org/configuration/packages/configuring-smac-planner.html

[61] D. Lu and A. Maslow, "Dwb local planner," 2020. [Online]. Available: https://github.com/locusrobotics/robot_navigation/tree/master/dwb_local_planner

[62] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[63] Z. Hong, S. Chun-Long, Z. Zi-Jun, A. Wei, Z. De-Qiang, and W. Jing-Jing, "A modified dynamic window approach to obstacle avoidance combined with fuzzy logic," in *2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, 2015, pp. 523–526.

[64] D. Lu, M. Ferguson, and A. Hoy, "Dwa local planner," 2020. [Online]. Available: http://wiki.ros.org/dwa_local_planner

[65] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 802–807.

[66] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–6.

[67] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model," in *2013 European Conference on Mobile Robots*, 2013, pp. 138–143.

[68] C. Rösmann, F. Hoffmann, and T. Bertram, "Planning of multiple robot trajectories in distinctive topologies," in *2015 European Conference on Mobile Robots (ECMR)*, 2015, pp. 1–6.

[69] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 500–505.

[70] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Transactions on robotics and automation*, vol. 16, no. 5, pp. 615–620, 2000.

[71] J. Song, C. Hao, and J. Su, "Path planning for unmanned surface vehicle based on predictive artificial potential field," *International Journal of Advanced Robotic Systems*, vol. 17, no. 2, p. 1729881420918461, 2020.

[72] Y. Hargas, A. Mokrane, A. Hentout, O. Hachour, and B. Bouzouia, "Mobile manipulator path planning based on artificial potential field: Application on robuter/ulm," *2015 4th International Conference on Electrical Engineering (ICEE)*, pp. 1–6, 2015.

[73] R. Bhadeshiya, R. Biyani, and N. Parikh, "Real time path planning," 2018. [Online]. Available: https://github.com/rishabh1b/RealTimePathPlanning

[74] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[75] W. Chen, J. Sun, W. Li, and D. Zhao, "A real-time multi-constraints obstacle avoidance method based on lidar," *CoRR*, vol. abs/2006.00142, 2020. [Online]. Available: https://arxiv.org/abs/2006.00142

[76] N. S. Karuppusamy and B.-Y. Kang, "Minimizing airtime by optimizing tool path in computer numerical control machine tools with application of a* and genetic algorithms," *Advances in Mechanical Engineering*, vol. 9, no. 12, 2017.

[77] M. A. Hossain and I. Ferdous, "Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique," *Robotics and Autonomous Systems*, vol. 64, pp. 137–141, 2015.

[78] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[79] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, "An improved ant colony algorithm for robot path planning," *Soft computing*, vol. 21, no. 19, pp. 5829–5839, 2017.

[80] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[81] S. Li, M. Q. H. Meng, W. Chen, Y. Li, Z. You, Y. Zhou, L. Sun, H. Liang, K. Jiang, and Q. Guo, "Sp-nn: A novel neural network approach for path planning," in *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2007, pp. 1355–1360.

[82] Y. Luo, X. Zhou, and X. Peng, "A path planning method for a four-wheeled robot based on an intelligent algorithm," in *2018 Chinese Automation Congress (CAC)*, 2018, pp. 2123–2128.

[83] H. Liu, C. Zhao, W. Huang, and W. Shi, "An end-to-end siamese convolutional neural network for loop closure detection in visual slam system," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 3121–3125.

[84] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A survey of end-to-end driving: Architectures and training methods," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 4, pp. 1364–1384, 2022.

[85] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[86] B. Widrow, "An adaptive'adaline'neuron using chemical'memistors', 1553-1552," 1960.

[87] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[88] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015.

[89] K. Gurney, *An introduction to neural networks*. CRC press, 2018.

[90] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 2009.

[91] A. Mehrish, N. Majumder, R. Bharadwaj, R. Mihalcea, and S. Poria, "A review of deep learning techniques for speech processing," *Information Fusion*, vol. 99, p. 101869, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253523001859

[92] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," 2014. [Online]. Available: https://arxiv.org/abs/1412.1842

[93] S. Motie and B. Raahemi, "Financial fraud detection using graph neural networks: A systematic review," *Expert Systems with Applications*, vol. 240, p. 122156, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423026581

[94] L. Cao, "A practical synthesis of detecting ai-generated textual, visual, and audio content," 2025. [Online]. Available: https://arxiv.org/abs/2504.02898

[95] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[96] B. J. Wythoff, "Backpropagation neural networks: a tutorial," *Chemometrics and Intelligent Laboratory Systems*, vol. 18, no. 2, pp. 115–155, 1993.

[97] S. Bag, "Activation functions — all you need to know!" February 2021, [Online; posted 13-February-2021]. [Online]. Available: https://medium.com/analytics-vidhya/activation-functions-all-you-need-to-know-355a850d025e

[98] N. Corporation, Jul 2023. [Online]. Available: https://docs.nvidia.com/deeplearning/performance/dl-performance-fully-connected/index.html

[99] A. D'Agostino, "Introduction to neural networks — weights, biases and activation," December 2021, [Online; posted 28-December-2021]. [Online]. Available: https://medium.com/@theDrewDag/introduction-to-neural-networks-weights-biases-and-activation-270ebf2545aa

[100] N. Devi and B. Borah, "Cascaded pooling for convolutional neural networks," in *2018 Fourteenth International Conference on Information Processing (ICINPRO)*, 2018, pp. 1–5.

[101] P. Raghav, "Understanding of convolutional neural network (cnn) — deep learning," March 2018, [Online; posted 04-March-2018]. [Online]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[102] T. Beuzen, "Deep learning with pytroch," 2021, [Online; posted 2021]. [Online]. Available: https://www.tomasbeuzen.com/deep-learning-with-pytorch/chapters/chapter5_cnns-pt1.html

[103] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[104] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[105] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," 2013. [Online]. Available: https://arxiv.org/abs/1301.3572

[106] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[107] M. S. Ali, "Flattening cnn layers for neural network and basic concepts," June 2024, [Online; posted 24-June-2024]. [Online]. Available: https://medium.com/@muhammadshoaibali/flattening-cnn-layers-for-neural-network-694a232eda6a

[108] F. June, "Batchnorm and layernorm," November 2023, [Online; posted 9-November-2023]. [Online]. Available: https://medium.com/@florian_algo/batchnorm-and-layernorm-2637f46a998b

[109] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015. [Online]. Available: https://arxiv.org/abs/1502.03167

[110] A. Yadav, "Batch normalization vs layer normalization," September 2024, [Online; posted 19-September-2024]. [Online]. Available: https://medium.com/biased-algorithms/batch-normalization-vs-layer-normalization-c44472883bf2

[111] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[112] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural Computation*, vol. 4, no. 4, pp. 473–493, 1992.

[113] H. Y. Xiong, Y. Barash, and B. J. Frey, "Bayesian prediction of tissue-regulated splicing using rna sequence and cellular context," *Bioinformatics*, vol. 27, no. 18, pp. 2554–2562, 07 2011. [Online]. Available: https://doi.org/10.1093/bioinformatics/btr444

[114] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 880–887.

[115] A. Jumba, "Deep learning: Image classification with convolutional neural networks (cnns)," October 2022, [Online; posted 15-October-2022]. [Online]. Available: https://medium.com/@arieljumba/deep-learning-task-image-classification-with-convolutional-neural-networks-cnns-ddd061b6e84b

[116] A. Yadav, "Residual networks explained," Medium article, Sept 2024, accessed: 2025-06-17. [Online]. Available: https://medium.com/biased-algorithms/residual-networks-explained-ee12438a009c

[117] N. Adaloglou, "Intuitive explanation of skip connections in deep learning," Blog post on AI Summer, Mar 2020, accessed: 2025-10-20. [Online]. Available: https://theaisummer.com/skip-connections/

[118] N. Acharya, "Choosing between mean squared error (mse) and mean absolute error (mae) in regression: A deep dive," August 2023, [Online; posted 31-August-2023]. [Online]. Available: https://medium.com/@nirajan.acharya777/choosing-between-mean-squared-error-mse-and-mean-absolute-error-mae-in-regression-a-deep-div

[119] C. P. Hughes, "A brief overview of cross entropy loss," Medium article, Sep 2024, accessed: 2025-10-06. [Online]. Available: https://medium.com/@chris.p.hughes10/a-brief-overview-of-cross-entropy-loss-523aa56b75d5

[120] Musstafa, "Optimizers in deep learning," Medium article, Mar 2021, accessed: 2025-10-06. [Online]. Available: https://musstafa0804.medium.com/optimizers-in-deep-learning-7bf81fed78a0

[121] B. Bodner, "10 pytorch optimizers you must know," Medium article, Jul 2024, accessed: 2025-10-06. [Online]. Available: https://medium.com/@benjybo7/10-pytorch-optimizers-you-must-know-c99cf3390899

[122] 3Blue1Brown, "What is backpropagation really doing? | deep learning, chapter 3," Nov. 2017. [Online]. Available: https://www.youtube.com/watch?v=Ilg3gGewQ5U

[123] C. Vandelaer, "Reinforcement learning: An introduction (part 1/4)," Medium article, Aug 2022, accessed: 2025-10-06. [Online]. Available: https://medium.com/@cedric.vandelaer/reinforcement-learning-an-introduction-part-1-4-866695deb4d1

[124] A. Kuriakose, "A guide to hyperparameter tuning: Enhancing machine learning models," Medium article, Oct 2023, accessed: 2024-08-20. [Online]. Available: https://medium.com/@abelkuriakose/a-guide-to-hyperparameter-tuning-enhancing-machine-learning-models-69dc9e0f02ea

[125] S. Rao, "The comprehensive guide to fine-tuning llm," Medium article, 2025, accessed: 2025-10-21. [Online]. Available: https://medium.com/data-science-collective/comprehensive-guide-to-fine-tuning-llm-4a8fd4d0e0af

[126] D. Singh, "Learning rate schedulers: Best practices, pitfalls, and advanced techniques," Medium article, Dec 2024, accessed: 2025-10-21. [Online]. Available:

https://medium.com/ai-enthusiast/
learning-rate-schedulers-best-practices-pitfalls-and-advanced-techniques-fe741125bdc4

[127] M. Laxman, "Overcoming overfitting and gradient issues in deep learning: An end-to-end guide," Medium article, Jun 2024, accessed: 2025-01-19. [Online]. Available: https://medium.com/@madasulaxman028/overcoming-overfitting-and-gradient-issues-in-deep-learning-an-end-to-end-guide-d7dca1cb6254

[128] Z. L. M. S. A. e. a. Zhang, Aston; Lipton, "Recurrent neural netowkrs," 2023. [Online]. Available: https://d2l.ai/chapter_recurrent-neural-networks/index.html

[129] A. Saxena, "Deep learning: Image classification with convolutional neural networks (cnns)," January 2023, [Online; posted 18-January-2023]. [Online]. Available: https://medium.com/analytics-vidhya/introduction-to-long-short-term-memory-lstm-a8052cd0d4cd

[130] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014. [Online]. Available: https://arxiv.org/abs/1412.3555

[131] L. AI, "Autoencoders," Medium article, Mar 2023, accessed: 2025-10-06. [Online]. Available: https://medium.com/@divakar1591/autoencoders-6fab1a9a9f9c

[132] D. Birla, "Autoencoders," Medium article, Mar 2019, accessed: 2025-10-06. [Online]. Available: https://medium.com/@birla.deepak26/autoencoders-76bb49ae6a8f

[133] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[134] OpenAI, "Gpt-4 technical report," arXiv preprint arXiv:2303.08774, 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[135] P. P. Ray, "Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 121–154, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266734522300024X

[136] DeepSeek-AI, "Deepseek llm: Scaling open-source language models with longtermism," arXiv preprint arXiv:2401.02954, 2024. [Online]. Available: https://arxiv.org/abs/2401.02954

[137] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014. [Online]. Available: https://arxiv.org/abs/1409.3215

[138] M. Saeed, "A gentle introduction to positional encoding in transformer models, part 1," Machine Learning Mastery, Jan 2023, accessed: 2025-07-21. [Online]. Available: https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

[139] K. Moses, "Encoder-decoder seq2seq models, clearly explained!!" March 2021, [Online; posted 12-March-2021]. [Online]. Available: https://medium.com/analytics-vidhya/encoder-decoder-seq2seq-models-clearly-explained-c34186fbf49b

[140] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[141] L. BOURDOIS, "Introduction to state space models (ssm)," Hugging Face Blog, Jul 2024, accessed: 2025-02-20. [Online]. Available: https://huggingface.co/blog/lbourdois/get-on-the-ssm-train

[142] Dec 2024. [Online]. Available: https://en.wikipedia.org/wiki/State-space_representation

[143] E. Martin and C. Cundy, "Parallelizing linear recurrent neural nets over sequence length," 2018. [Online]. Available: https://arxiv.org/abs/1709.04057

[144] J. T. H. Smith, A. Warrington, and S. W. Linderman, "Simplified state space layers for sequence modeling," 2023. [Online]. Available: https://arxiv.org/abs/2208.04933

[145] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Re, "Hippo: Recurrent memory with optimal polynomial projections," 2020. [Online]. Available: https://arxiv.org/abs/2008.07669

[146] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[147] M. Grootendorst, "A visual guide to mamba and state space models," Exploring Language Models, Feb 2024, accessed: 2025-10-20. [Online]. Available: https://newsletter.maartengrootendorst.com/p/a-visual-guide-to-mamba-and-state

[148] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds., vol. 18. MIT Press, 2005. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2005/file/fdf1bc5669e8ff5ba45d02fded729feb-Paper.pdf

[149] M. Bojarski, C. Chen, J. Daw, A. Değirmenci, J. Deri, B. Firner, B. Flepp, S. Gogri, J. Hong, L. Jackel *et al.*, "The nvidia pilotnet experiments," *arXiv preprint arXiv:2010.08776*, 2020.

[150] A. Singh, "End-to-end autonomous driving using deep learning: A systematic review," 2023. [Online]. Available: https://arxiv.org/abs/2311.18636

[151] 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," 2018. [Online]. Available: https://arxiv.org/abs/1807.00412

[152] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," 2018. [Online]. Available: https://arxiv.org/abs/1710.02410

[153] S. Paniego, E. Shinohara, and J. Cañas, "Autonomous driving in traffic with end-to-end vision-based deep learning," *Neurocomputing*, vol. 594, p. 127874, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231224006453

[154] Y. A. Ozaibi, M. Dulva Hina, and A. Ramdane-Cherif, "End-to-end autonomous driving in carla: A survey," *IEEE Access*, vol. 12, pp. 146 866–146 900, 2024.

[155] Y. Lecun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning." 01 2005.

[156] I. Sonata, Y. Heryadi, A. Wibowo, and W. Budiharto, "End-to-end steering angle prediction for autonomous car using vision transformer," *CommIT (Communication and Information Technology) Journal*, vol. 17, pp. 221–234, 09 2023.

[157] S. Chen, "driving-datasets: A collection of labeled car driving datasets," GitHub repository, 2018, accessed: 2025-09-02. [Online]. Available: https://github.com/SullyChen/driving-datasets

[158] A. M. Vyas, "End-to-end learned visual odometry based on vision transformer," Master's thesis, University of Turku, Department of Computing, 2024, master's thesis, Autonomous System Lab. [Online]. Available: https://www.utupub.fi/bitstream/handle/10024/178848/Aman_Manishbhai_Vyas_Thesis.pdf

[159] A. Farzipour, O. N. Manzari, and S. B. Shokouhi, "Traffic sign recognition using local vision transformer," in *2023 13th International Conference on Computer and Knowledge Engineering (ICCKE)*, 2023, pp. 191–196.

[160] R. Liu, Z. Yuan, T. Liu, and Z. Xiong, "End-to-end lane shape prediction with transformers," 2020. [Online]. Available: https://arxiv.org/abs/2011.04233

[161] R. Cheng, X. An, and Y. Xu, "Vit-traj: A spatial–temporal coupling vehicle trajectory prediction model based on vision transformer," *Systems*, vol. 13, no. 3, 2025. [Online]. Available: https://www.mdpi.com/2079-8954/13/3/147

[162] C. Yuan, Z. Zhang, J. Sun, S. Sun, Z. Huang, C. D. W. Lee, D. Li, Y. Han, A. Wong, K. P. Tee, and M. H. A. Jr, "Drama: An efficient end-to-end motion planner for autonomous driving with mamba," 2024. [Online]. Available: https://arxiv.org/abs/2408.03601

[163] D. Dauner, M. Hallgarten, T. Li, X. Weng, Z. Huang, Z. Yang, H. Li, I. Gilitschenski, B. Ivanovic, M. Pavone, A. Geiger, and K. Chitta, "Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking," 2024. [Online]. Available: https://arxiv.org/abs/2406.15349

[164] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," 2019. [Online]. Available: https://arxiv.org/abs/1911.02620

[165] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, D. Ramanan, P. Carr, and J. Hays, "Argoverse 2: Next generation datasets for self-driving perception and forecasting," 2023. [Online]. Available: https://arxiv.org/abs/2301.00493

[166] Y. Huang, Y. Cheng, and K. Wang, "Trajectory mamba: Efficient attention-mamba forecasting model based on selective ssm," 2025. [Online]. Available: https://arxiv.org/abs/2503.10898

[167] X. Jia, Z. Yang, Q. Li, Z. Zhang, and J. Yan, "Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving," 2024. [Online]. Available: https://arxiv.org/abs/2406.03877

[168] B. Jaeger, K. Chitta, and A. Geiger, "Hidden biases of end-to-end driving models," 2023. [Online]. Available: https://arxiv.org/abs/2306.07957

[169] C. Kirkham, N. Shirouzu, and R. Levy, "How tesla and waymo's radically different robotaxi approaches will shape the industry," *Reuters*, Aug 2025, accessed: 2025-09-03. [Online]. Available: https://www.reuters.com/business/autos-transportation/how-tesla-waymos-radically-different-robotaxi-approaches-will-shape-industry-2025-08-28/?utm_source=chatgpt.com

[170] S. Macenski and I. Jambrecic, "Slam toolbox: Slam for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, 2021. [Online]. Available: https://doi.org/10.21105/joss.02783

[171] S. Macenski, "On use of the slam toolbox: A fresh (er) look at mapping and localization for the dynamic world," *ROSCon*, 2019.

[172] R. Community, "How to get the position and orientation of the robot in ros?" ROS Answers — Robotics Stack Exchange, 2018, accessed: 2023-07-20. [Online]. Available: https://answers.ros.org/question/317435/

[173] DDS Foundation, "What is DDS?" May 2022. [Online]. Available: https://www.dds-foundation.org/what-is-dds-3/

[174] Wind River Systems, "VxWorks - The Leading RTOS for the Intelligent Edge," May 2022. [Online]. Available: https://www.windriver.com/products/vxworks

[175] ROS Tutorial, "Real-time programming in ROS 2," May 2022. [Online]. Available: https://docs.ros.org/en/foxy/Tutorials/Real-Time-Programming.html

[176] T. A. Foundation, "Autoware documentation – autonomous driving stack," Online documentation, 2025, accessed: 2025-10-20. [Online]. Available: https://autowarefoundation.github.io/autoware-documentation/main/

[177] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 384–406, 2018.

[178] V. V. Dixit, S. Chand, and D. J. Nair, "Autonomous vehicles: Disengagements, accidents and reaction times," *PLOS ONE*, vol. 11, no. 12, p. e0168054, 2016.

[179] S. S. Banerjee, S. Jha, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2018, pp. 586–597.

[180] M.-B. G. AG, "Mercedes-benz world's first automotive company to certify sae level 3 system for u.s. market," Press release (BusinessWire), Jan 2023, accessed: 2025-10-20. [Online]. Available: https://www.businesswire.com/news/home/20230126005679/en/Mercedes-Benz-worlds-first-automotive-company-to-certify-SAE-Level-3-system-for-U.S.-market

[181] L. Haddad, "Increasing reliability of an autonomous vehicle stack in robot operating system 2," Master's thesis, The University of Western Australia, Perth, Western Australia, Dec 2022, master of Professional Engineering (Software), supervised by Prof. Thomas Bräunl. [Online]. Available: https://roblab.org/theses/2022-REV-Reliability-Haddad.pdf

[182] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016.

[183] R.-N. Contributors, "nav2_navfn_planner: A global planner plugin for ros 2 navigation2," GitHub repository, 2025, accessed: 2025-08-08. [Online]. Available: https://github.com/ros-navigation/navigation2/tree/main/nav2_navfn_planner

[184] O. Robotics and the Navigation2 contributors, "Smac 2d planner — configuration guide, nav2 1.0.0," Online documentation, 2025, accessed: 2025-07-20. [Online]. Available: https://docs.nav2.org/configuration/packages/smac/configuring-smac-2d.html

[185] A. Muraleedharan, H. Okuda, and T. Suzuki, "Real-time implementation of randomized model predictive control for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 1, pp. 11–20, 2021.

[186] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1433–1440.

[187] K. Honda, H. Okuda, T. Suzuki, and A. Ito, "Connection of nonlinear model predictive controllers for smooth task switching in autonomous driving," *Asian Journal of Control*, vol. 25, no. 3, pp. 1805–1822, 2023.

[188] H. Xue, E. L. Zhu, and F. Borrelli, "Learning model predictive control with error dynamics regression for autonomous racing," 2023.

[189] P. Cai, S. Wang, H. Wang, and M. Liu, "Carl-lead: Lidar-based end-to-end autonomous driving with contrastive deep reinforcement learning," 2021. [Online]. Available: https://arxiv.org/abs/2109.08473

[190] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[191] 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," 2016. [Online]. Available: https://arxiv.org/abs/1604.07316

[192] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[193] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *ICLR*, 2021.

[194] L. Zhu, B. Liao, Q. Zhang, X. Wang, W. Liu, and X. Wang, "Vision mamba: Efficient visual representation learning with bidirectional state space model," in *Forty-first International Conference on Machine Learning*.

[195] Y. Liu, Y. Tian, Y. Zhao, H. Yu, L. Xie, Y. Wang, Q. Ye, and Y. Liu, "Vmamba: Visual state space model," *arXiv preprint arXiv:2401.10166*, 2024.

[196] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers amp; distillation through attention," in *International Conference on Machine Learning*, vol. 139, July 2021, pp. 10 347–10 357.

[197] W. Jun, M. Son, and S. Lee, "Performance analysis and enhancement of multi-task learning for autonomous driving," in *2024 IEEE International Conference on Consumer Electronics (ICCE)*, 2024, pp. 1–4.

[198] Y.-C. Wang, J.-L. Lin, Y.-L. Chen, C.-S. Huang, and M.-L. Lai, "Multi-task deep learning model for autonomous driving: Object detection, semantic segmentation, and depth estimation," in *2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*, 2023, pp. 683–684.

[199] H. Chen and T. You, "A lightweight multi-task learning model for scene perception in autonomous driving," in *2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, 2024, pp. 241–246.

[200] K. Team, "Keras applications — models api documentation," https://keras.io/api/applications/, 2025, accessed: 2025-10-06.

[201] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020. [Online]. Available: https://arxiv.org/abs/1905.11946

[202] ——, "Efficientnetv2: Smaller models and faster training," 2021. [Online]. Available: https://arxiv.org/abs/2104.00298

[203] RustamyF, "vision-transformer," https://github.com/RustamyF/vision-transformer/tree/master, 2025, accessed: 2025-09-30.

[204] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[205] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *International Journal of Computer Vision*, vol. 127, no. 3, pp. 302–321, 2019.

[206] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo, "Swin transformer v2: Scaling up capacity and resolution," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[207] hustvl, "Vim: Vision mamba – efficient visual representation learning with bidirectional state space model," https://github.com/hustvl/Vim, 2025, accessed: 2025-09-30.

[208] MzeroMiko, "Vmamba: Visual state space models," https://github.com/MzeroMiko/VMamba, 2025, accessed: 2025-09-30.

[209] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

[210] J. a. N. F. Alves, L. M. S. Russo, and A. Francisco, "Cache-oblivious hilbert curve-based blocking scheme for matrix transposition," *ACM Trans. Math. Softw.*, vol. 48, no. 4, Dec. 2022. [Online]. Available: https://doi.org/10.1145/3555353

[211] B. Spicer, "Customizing vmamba with hilbert curves," Medium article, May 2025, accessed: 2025-04-06. [Online]. Available: https://medium.com/@besp2426/customizing-vmamba-with-hilbert-curves-aa50909c913b

[212] D. Liang, X. Zhou, W. Xu, X. Zhu, Z. Zou, X. Ye, X. Tan, and X. Bai, "Pointmamba: A simple state space model for point cloud analysis," in *Advances in Neural Information Processing Systems*, 2024.

[213] Y. LeCun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, 2004, pp. II–104 Vol.2.

[214] S. Carvalho, J. Humphries, N. Dunne, and S. Leahy, "Impact of light flickering on object detection accuracy using convolutional neural networks," in *2021 Telecoms Conference (ConfTELE)*, 2021, pp. 1–6.

[215] A. Silwal, T. Parhar, F. Yandun, H. Baweja, and G. Kantor, "A robust illumination-invariant camera system for agricultural applications," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3292–3298.

[216] M. Z. Alam, Z. Kaleem, and S. Kelouwani, "How to deal with glare for improved perception of autonomous vehicles," *arXiv preprint arXiv:2404.10992*, 2024, [preprint] IEEE Transactions on Intelligent Vehicles.

[217] S. McCrae and A. Zakhor, "3d object detection for autonomous driving using temporal lidar data," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 2661–2665.

[218] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," 2020. [Online]. Available: https://arxiv.org/abs/1903.11027

[219] Y. Sun, W. Zuo, H. Huang, P. Cai, and M. Liu, "Pointmoseg: Sparse tensor-based end-to-end moving-obstacle segmentation in 3-d lidar point clouds for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 510–517, 2021.

[220] Z. Liu, A. Amini, S. Zhu, S. Karaman, S. Han, and D. L. Rus, "Efficient and robust lidar-based end-to-end navigation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13 247–13 254.

[221] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[222] J. T. Rolfe, "Discrete variational autoencoders," 2017. [Online]. Available: https://arxiv.org/abs/1609.02200

[223] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," 2015. [Online]. Available: https://arxiv.org/abs/1406.5670

[224] M. A. Uy, Q.-H. Pham, B.-S. Hua, D. T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *International Conference on Computer Vision (ICCV)*, 2019.

[225] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, "Point-bert: Pre-training 3d point cloud transformers with masked point modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[226] Y. Pang, W. Wang, F. E. H. Tay, W. Liu, Y. Tian, and L. Yuan, "Masked autoencoders for point cloud self-supervised learning," 2022. [Online]. Available: https://arxiv.org/abs/2203.06604

[227] Z. Wang, Z. Chen, Y. Wu, Z. Zhao, L. Zhou, and D. Xu, "Pointramba: A hybrid transformer-mamba framework for point cloud analysis," 2024. [Online]. Available: https://arxiv.org/abs/2405.15463

[228] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3d shape collections," *SIGGRAPH Asia*, 2016.

[229] G. Zhao, X. Xiao, J. Yuan, and G. W. Ng, "Fusion of 3d-lidar and camera data for scene parsing," *Journal of Visual Communication and Image Representation*, vol. 25, pp. 165–183, 1 2014.

[230] Y. Zha, L. Guo, Y. Chen, Y. Wu, J. Wang, and R. Li, "End-to-end object detection system using multi-source data fusion for autonomous driving," in *2024 7th International Conference on Robotics, Control and Automation Engineering (RCAE)*, 2024, pp. 396–400.

[231] 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*, 2012, pp. 3354–3361.

[232] N. T. Hoai Thu and D. Seog Han, "An end-to-end motion planner using sensor fusion for autonomous driving," in *2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2023, pp. 678–683.

[233] W. LLC. (2025) Waymo driver. [Online]. Available: https://waymo.com/waymo-driver/?ncr

[234] D. Hubert, "Ueber die stetige abbildung einer linie auf ein flächenstück," *Mathematische Annalen*, vol. 38, pp. 459–460, 1891. [Online]. Available: http://eudml.org/doc/157555

[235] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[236] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[237] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local svm approach," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004)*, 2004.

[238] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," 2012. [Online]. Available: https://arxiv.org/abs/1212.0402

[239] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[240] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," 2012. [Online]. Available: https://arxiv.org/abs/1212.0402

[241] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" 2021. [Online]. Available: https://arxiv.org/abs/2102.05095

[242] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," 2017. [Online]. Available: https://arxiv.org/abs/1705.06950

[243] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier, and A. Zisserman, "A short note about kinetics-600," 2018. [Online]. Available: https://arxiv.org/abs/1808.01340

[244] R. Goyal, S. E. Kahou, V. Michalski, J. Materzyńska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax, and R. Memisevic, "The "something something" video database for learning and evaluating visual common sense," 2017. [Online]. Available: https://arxiv.org/abs/1706.04261

[245] Y. Li, Y. Li, and N. Vasconcelos, "Resound: Towards action recognition without representation bias," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[246] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," 2021. [Online]. Available: https://arxiv.org/abs/2103.15691

[247] M. M. Islam and G. Bertasius, "Long movie clip classification with state-space video models," 2023. [Online]. Available: https://arxiv.org/abs/2204.01692

[248] C.-Y. Wu and P. Krähenbühl, "Towards long-form video understanding," 2021. [Online]. Available: https://arxiv.org/abs/2106.11310

[249] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities," in *Proceedings of the 2014 IEEE*

Conference on Computer Vision and Pattern Recognition, ser. CVPR '14. USA: IEEE Computer Society, 2014, p. 780–787. [Online]. Available: https://doi.org/10.1109/CVPR.2014.105

[250] Y. Tang, D. Ding, Y. Rao, Y. Zheng, D. Zhang, L. Zhao, J. Lu, and J. Zhou, "Coin: A large-scale dataset for comprehensive instructional video analysis," 2019. [Online]. Available: https://arxiv.org/abs/1903.02874

[251] T. Dao and A. Gu, "Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality," in *International Conference on Machine Learning (ICML)*, 2024.

[252] K. Li, X. Li, Y. Wang, Y. He, Y. Wang, L. Wang, and Y. Qiao, "Videomamba: State space model for efficient video understanding," 2024.

[253] J. Park, H.-S. Kim, K. Ko, M. Kim, and C. Kim, "Videomamba: Spatio-temporal selective state space model," in *European Conference on Computer Vision*. Springer, 2024, pp. 1–18.

[254] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," 2017. [Online]. Available: https://arxiv.org/abs/1612.01079

[255] Z. Tong, Y. Song, J. Wang, and L. Wang, "Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training," 2022. [Online]. Available: https://arxiv.org/abs/2203.12602

[256] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," 2021. [Online]. Available: https://arxiv.org/abs/2111.06377

[257] L. Wang, B. Huang, Z. Zhao, Z. Tong, Y. He, Y. Wang, Y. Wang, and Y. Qiao, "Videomae v2: Scaling video masked autoencoders with dual masking," 2023. [Online]. Available: https://arxiv.org/abs/2303.16727

[258] S. Ahmed, M. N. Huda, S. Rajbhandari, C. Saha, M. Elshaw, and S. Kanarachos, "Pedestrian and cyclist detection and intent estimation for autonomous vehicles: A survey," 6 2019.

[259] Deloitte, "Deloitte global automotive consumer study—advanced vehicle technologies and multimodal transportation, global focus countries," Tech. Rep., 2019. [Online].

Available: https://www2.deloitte.com/content/dam/Deloitte/us/Documents/manufacturing/us-global-automotive-consumer-study-2019.pdf

[260] J. D'Onfro, "'i hate them': Locals reportedly are frustrated with alphabet's self-driving cars," *CNBC*, 2018-08-28, visited on the 2025-01-29. [Online]. Available: https://www.cnbc.com/2018/08/28/locals-reportedly-frustrated-with-alphabets-waymo-self-driving-cars.html

[261] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A survey of end-to-end driving: Architectures and training methods," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 4, p. 1364–1384, Apr. 2022. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2020.3043505

[262] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, "Vision-based autonomous car racing using deep imitative reinforcement learning," 2021. [Online]. Available: https://arxiv.org/abs/2107.08325

[263] 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)*, 2019, pp. 8248–8254.

[264] M. A. Elhassan, C. Zhou, A. Khan, A. Benabid, A. B. Adam, A. Mehmood, and N. Wambugu, "Real-time semantic segmentation for autonomous driving: A review of cnns, transformers, and beyond," *Journal of King Saud University - Computer and Information Sciences*, vol. 36, no. 10, p. 102226, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S131915782400315X

[265] C. J. Holder and M. Shafique, "On efficient real-time semantic segmentation: A survey," 2022. [Online]. Available: https://arxiv.org/abs/2206.08605

[266] S. Cakir, M. Gauß, K. Häppeler, Y. Ounajjar, F. Heinle, and R. Marchthaler, "Semantic segmentation for autonomous driving: Model evaluation, dataset generation, perspective comparison, and real-time capability," 2022. [Online]. Available: https://arxiv.org/abs/2207.12939

[267] H. Shen, Z. Wan, X. Wang, and M. Zhang, "Famba-v: Fast vision mamba with cross-layer token fusion," 2024. [Online]. Available: https://arxiv.org/abs/2409.09808

[268] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick,

P. Dollár, and C. Feichtenhofer, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024. [Online]. Available: https://arxiv.org/abs/2408.00714

[269] H. G. Kim, "Real-time object detection model for autonomous vehicles," Master's thesis, The University of Western Australia, May 2025, master's Thesis, submitted on 19 May 2025. [Online]. Available: https://roblab.org/theses/

[270] J. Hossain, "Autonomous driving with deep reinforcement learning in carla simulation," 2023. [Online]. Available: https://arxiv.org/abs/2306.11217

[271] E. Delavari, F. K. Khanzada, and J. Kwon, "A comprehensive review of reinforcement learning for autonomous driving in the carla simulator," 2025. [Online]. Available: https://arxiv.org/abs/2509.08221

[272] Yosh, "Ai learns to drive from scratch in trackmania," YouTube video, Mar 2022, accessed: 2024-10-13. [Online]. Available: https://www.youtube.com/watch?v=SX08NT55YhA

[273] T. Wu and M. Shipman, "New attack can make ai 'see' whatever you want," News release, North Carolina State University, Jul 2025, accessed: 2025-10-21. [Online]. Available: https://research.ncsu.edu/new-attack-can-make-ai-see-whatever-you-want/

[274] Y. Tang, P. Dong, Z. Tang, X. Chu, and J. Liang, "Vmrnn: Integrating vision mamba and lstm for efficient and accurate spatiotemporal forecasting," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2024, pp. 5663–5673.