Self-Driving Car Navigation Using Multi-Beam LiDAR and Neural Networks

Project Phase I Report

submitted by

MUHAMMED MIDLAJ V (Reg. No. MAC22CD041)

FEBIN YESUDAS T S (Reg. No. MAC22CD030)

SOMANATH K S (Reg. No. LMAC22CD067)

to

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

in partial fulfilment of the requirements for the award of the Degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Department of Computer Science & Engineering

Mar Athanasius College of Engineering (Autonomous)

Kothamangalam, Kerala, India 686 666

NOVEMBER 2025

Self-Driving Car Navigation Using Multi-Beam LiDAR and Neural Networks

Project Phase I Report

submitted by

MUHAMMED MIDLAJ V (Reg. No. MAC22CD041)

FEBIN YESUDAS T S (Reg. No. MAC22CD030)

SOMANATH K S (Reg. No. LMAC22CD067)

to

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

in partial fulfilment of the requirements for the award of the Degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Department of Computer Science & Engineering

Mar Athanasius College of Engineering (Autonomous)

Kothamangalam, Kerala, India 686 666

NOVEMBER 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAR ATHANASIUS COLLEGE OF ENGINEERING

(AUTONOMOUS)

KOTHAMANGALAM

CERTIFICATE

This is to certify that the report entitled **Self-Driving Car Navigation Using Multi-Beam LiDAR and Neural Networks** *submitted by* **Mr. Muhammed Midlaj V (Reg. No. MAC22CD041)**, **Mr. Febin Yesudas T S (Reg. No. MAC22CD030)**, **Mr. Somanath K S (Reg. No. LMAC22CD067)**, *towards partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer Science and Engineering (Data Science) from APJ Abdul Kalam Technological University for November 2025 is a bonafide record of the project carried out by them under our supervision and guidance.*

.....	.....	.....
Prof. Sruthy Susan Moncy	**Prof. Suzen Saju Kallungal**	**Prof. Joby George**
Project Guide	Project Coordinator	Head of the Department

Date: Dept. Seal

ACKNOWLEDGEMENT

First and foremost, We sincerely thank the 'God Almighty' for his grace for the successful and timely completion of the project.

We express our sincere gratitude and thanks to ***Dr. Bos Mathew Jos****, Principal and* ***Prof. Joby George****, Head of the Department for providing the necessary facilities and their encouragement and support.*

We owe special thanks to our project guide ***Prof. Sruthy Susan Moncy*** *and project coordinator* ***Prof. Suzen Saju Kallungal*** *for their corrections, suggestions and sincere efforts to co-ordinate the project under a tight schedule.*

We express our sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in guiding and correcting us in conducting this project.

Finally, We would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given to us by our dear friends during the preparation of the project and also during the presentation without which this work would have been all the more difficult to accomplish.

ABSTRACT

Accurate and efficient environmental perception is fundamental to advancing autonomous vehicle navigation. An intelligent system has been developed for self-driving car control and obstacle avoidance, integrating multi-beam LiDAR simulation with neural network–based decision-making. The system employs ray casting techniques to emulate Five LiDAR beams, each oriented at specific intervals, enabling comprehensive spatial awareness and distance measurement within the vehicle's surrounding environment. Leveraging a feedforward neural network (FNN) trained on simulated LiDAR data, the system performs real-time trajectory prediction and steering control, ensuring precise and adaptive navigation across varying obstacle layouts. The ray casting model replicates LiDAR sensing behavior efficiently, reducing computational cost while maintaining high detection accuracy. By combining the speed of geometric ray simulation with the pattern recognition capabilities of neural networks, this framework enables smooth, autonomous movement within a virtual environment. The proposed system demonstrates enhanced obstacle detection accuracy and navigation stability, providing a scalable and cost-effective foundation for future autonomous driving research and real-world implementation.

Table of Contents

ACKNOWLEDGEMENT i

ABSTRACT ii

List of Figures vi

List of Abbreviations vii

1 Introduction 1

1.1 Introduction 1

1.2 Objective 1

1.3 Background 2

1.3.1 JavaScript 2

1.3.2 LiDAR Technology 3

1.3.3 Neural Networks 3

1.3.4 Libraries 4

1.3.5 Software Description 5

2 Literature Review 7

2.1 Multimodal Dataset for Indian Driving 7

2.1.1 Summary 7

2.1.2 Observations 8

2.1.3 Drawbacks 8

2.2 Pothole Detection with YOLOv8 8

2.2.1 Summary 8

2.2.2 Observations 9

2.2.3 Drawbacks 9

2.3 LiDAR-Based Obstacle Avoidance 9

2.3.1 Summary . . . 10
2.3.2 Observations . . . 10
2.3.3 Drawbacks . . . 10
2.4 Cooperative LiDAR for autonomous vehicle . . . 11
2.4.1 Summary . . . 11
2.4.2 Observations . . . 11
2.4.3 Drawbacks . . . 12
2.5 YOLOv7t for Mixed Traffic Conditions . . . 12
2.5.1 Summary . . . 12
2.5.2 Observations . . . 13
2.5.3 Drawbacks . . . 13
2.6 LiDAR Security: Threats and Defense . . . 13
2.6.1 Summary . . . 13
2.6.2 Observations . . . 14
2.6.3 Drawbacks . . . 14
2.7 LiDAR Odometry and Mapping Innovations . . . 14
2.7.1 Summary . . . 15
2.7.2 Observations . . . 15
2.7.3 Drawbacks . . . 16
2.8 Deep Learning in Autonomous Driving . . . 16
2.8.1 Summary . . . 16
2.8.2 Observations . . . 17
2.8.3 Drawbacks . . . 17
2.9 Frontiers of Autonomous Driving . . . 17
2.9.1 Summary . . . 17
2.9.2 Observations . . . 18
2.9.3 Drawbacks . . . 18
2.10 Deep Learning for LiDAR Point Clouds . . . 19
2.10.1 Summary . . . 19
2.10.2 Observations . . . 20
2.10.3 Drawbacks . . . 20

3 Design and Implementation **21**

3.1 System Architecture 21
3.1.1 Sensor Simulation (LiDAR) 22
3.1.2 Data Processing Module 22
3.1.3 Neural Network Prediction Module 23
3.1.4 Visualization Interface 23
3.2 Design Components 23
3.2.1 LiDAR Simulation Using Ray Casting 24
3.2.2 Neural Network Architecture 24
3.3 Workflow 24
3.3.1 Overview 25
3.3.2 Iterative Learning Cycle 26
3.4 Real-Time Navigation and Visualization 26
3.4.1 Simulation Loop 26
3.4.2 User Interface and Visualization Features 26

4 Results and Discussion **28**
4.1 Phase 1: Obstacle Detection and Avoidance 28
4.1.1 Simulation Setup 28
4.1.2 Output Generation 29
4.1.3 Challenges and Solutions 30

5 Future Scope **32**

6 Conclusion **33**

7 Annexure **34**
7.1 Self-Driving-Car.java 34

REFERENCES **40**

List of Figures

3.1 Proposed System Architecture 22
3.2 Workflow of the Evolutionary Learning Process for the Autonomous Vehicle Simulation 26

4.1 Real-Time Navigation and Lane Following in Simulated Traffic . . 29
4.2 Phase 1: Self-Driving car tackling through traffic 30

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
GPU	Graphics Processing Unit
JS	JavaScript
LiDAR	Light Detection and Ranging
ML	Machine Learning
NN	Neural Network
ReLU	Rectified Linear Unit
UI	User Interface

Chapter 1

Introduction

1.1 Introduction

The self-driving car is an AI-driven autonomous system designed to navigate safely and efficiently without human input by integrating multi-beam LiDAR sensing and neural network–based control. Using ray casting, the system simulates Five LiDAR beams placed at specific intervals to achieve around 180° field of view for accurate distance measurement and obstacle detection. The LiDAR readings are processed by a feedforward neural network (FNN) that predicts steering, acceleration, and braking actions in real time, allowing the car to respond intelligently to changing environments. Implemented entirely in JavaScript, the simulation runs directly in a browser, offering interactive visualization and adaptive control without specialized hardware. By combining virtual sensing and deep learning, this project establishes a cost-effective foundation for scalable autonomous navigation and advances research in intelligent transportation systems.

1.2 Objective

The main objectives of this project are outlined below, reflecting the design, development, and evaluation of a real-time autonomous driving simulation:

- **To simulate multi-beam LiDAR sensing:** Implement a ray-casting-based virtual LiDAR system to emulate real-world distance measurement and obstacle detection in a 2D environment.

- **To design and train a neural network controller:** Develop a Feedforward Neural Network (FNN) capable of predicting steering, acceleration, and braking commands from LiDAR input data.
- **To achieve interactive, real-time navigation:** Integrate LiDAR simulation and neural control into a browser-based framework for smooth and adaptive vehicle motion visualization.
- **To ensure modularity and scalability:** Design a flexible architecture that can integrate additional sensors, extended path-planning algorithms, and 3D simulation support for future development.

These objectives aim to demonstrate the potential of integrating simulated LiDAR sensing and neural network control for achieving intelligent, adaptive, and cost-efficient autonomous navigation in simulated environments.

1.3 Background

1.3.1 JavaScript

JavaScript is a high-level, interpreted programming language primarily used for creating dynamic and interactive web applications. It is lightweight, flexible, and platform-independent, capable of running directly in web browsers without compilation. Its event-driven structure and asynchronous execution make it ideal for building real-time systems and browser-based simulations. Over the years, JavaScript has expanded its use beyond front-end development through environments such as Node.js, enabling large-scale artificial intelligence and data processing applications. The simplicity of its syntax, combined with an extensive library ecosystem, makes JavaScript one of the most popular programming languages globally. In this project, JavaScript is used to simulate LiDAR behavior, implement neural network models, and handle visualization tasks within a browser environment.

1.3.2 LiDAR Technology

LiDAR (Light Detection and Ranging) is a remote sensing technique that measures the distance to surrounding objects by emitting laser pulses and detecting their reflections. The time taken for each pulse to return is used to calculate precise distances, creating a detailed spatial map of the environment. Traditional LiDAR systems employ multiple rotating beams to produce dense 3D point clouds, but these are expensive and computationally intensive. In this project, a five-beam LiDAR system is simulated using the technique of ray casting, where each beam is represented as a virtual ray projected outward from the vehicle at specific intervals. When a ray intersects an object, the distance is measured and stored as LiDAR range data. This ray casting approach efficiently mimics LiDAR functionality in a simulated environment, providing real-time perception data for autonomous navigation and obstacle detection.

1.3.3 Neural Networks

Neural networks are computational systems inspired by the human brain's architecture. They consist of interconnected nodes, or neurons, organized in layers that process and transform input data to generate meaningful outputs. Each neuron performs a weighted computation followed by a non-linear activation, allowing the network to learn complex relationships from data. In this project, a feedforward neural network processes the four-beam LiDAR data to predict navigation commands such as steering, acceleration, and braking. Through training using the backpropagation algorithm, the model learns to associate distance patterns with safe driving actions. Neural networks are particularly effective in real-time decision-making tasks due to their adaptability and ability to generalize from experience.

1.3.4 Libraries

1.3.4.1 TensorFlow.js

TensorFlow.js is an open-source machine learning library developed by Google that enables model definition, training, and inference directly in the web browser. It provides GPU acceleration and a high-level API for neural network operations. In this project, TensorFlow.js is used to build and train the neural network responsible for decision-making based on LiDAR inputs. Its ability to run entirely on the client side allows for real-time performance visualization without the need for backend servers. TensorFlow.js's modular design and ease of integration make it ideal for browser-based AI simulations.

1.3.4.2 Three.js

Three.js is a JavaScript library built on top of WebGL, designed for creating 3D graphics and visualizations in web browsers. It provides tools for camera manipulation, lighting, rendering, and object interaction. In this project, Three.js is used to visualize the car's movement, obstacles, and LiDAR beams. The library's built-in `Raycaster` class is employed to perform ray casting, determining intersections between LiDAR beams and objects in the scene. This visualization aids in demonstrating how the simulated LiDAR perceives the environment and how the neural network responds to detected obstacles.

1.3.4.3 Node.js

Node.js is a cross-platform runtime environment that allows JavaScript code to run outside the browser. It uses an event-driven, non-blocking I/O model that makes it efficient for handling concurrent operations. In this project, Node.js is utilized to manage training data, logging, and communication between components. It also enables scalable handling of simulation data and provides access to numerous open-source libraries via npm (Node Package Manager).

1.3.4.4 P5.js

P5.js is a JavaScript library that simplifies creative coding and graphical rendering. It provides a set of easy-to-use functions for visualizing interactive animations and real-time data. In this project, P5.js is used for 2D visualization of LiDAR beam projections, object detection, and car trajectory. It offers a simple and flexible platform for testing and debugging the simulation before implementing the full 3D model.

1.3.5 Software Description

1.3.5.1 Visual Studio Code (VS Code)

Visual Studio Code is a free and open-source code editor developed by Microsoft. It supports multiple programming languages, including JavaScript, and integrates tools for debugging, syntax highlighting, and Git version control. Its lightweight design and extensive extension library make it an excellent choice for developing and testing AI-based web simulations. In this project, VS Code serves as the primary development environment for coding, running, and debugging the LiDAR and neural network simulation.

1.3.5.2 Web Browser Environment

The simulation runs entirely within a web browser, using its JavaScript engine for both computation and visualization. Modern browsers like Google Chrome and Mozilla Firefox provide GPU acceleration through WebGL, enabling efficient execution of neural network inference and real-time rendering. This setup removes the need for additional software installations and allows cross-platform compatibility. The browser acts as both the computation engine and display interface, integrating LiDAR simulation, neural network control, and user interactivity into a single framework.

1.3.5.3 GitHub and Version Control

GitHub is used for version control and collaborative development. Through Git integration in VS Code, changes to source code, neural network configura-

tions, and simulation parameters are tracked efficiently. This ensures proper version management and facilitates teamwork, testing, and documentation. GitHub also provides an accessible platform for sharing the project with others and maintaining open-source transparency.

Chapter 2

Literature Review

2.1 Multimodal Dataset for Indian Driving

Paper Title: SMILE: A Small Multimodal Dataset Capturing Roadside Behavior in Indian Driving Conditions

Authors: Mayur Anand Pandya, Aaryan Takayuki Panigrahi, Subham Patra, Asmit Paul, and Sucharitha Shetty

Source: IEEE Access,Volume13,30 July 2025 - ieeexplore.ieee.org

2.1.1 Summary

SMILE Dataset introduces a comprehensive multimodal dataset specifically designed to address the challenges of real-world Indian traffic scenarios, which are characterized by high unpredictability and diversity. Unlike existing autonomous driving datasets that focus on structured urban environments, SMILE captures unique behavioral patterns of Vulnerable Road Users (VRUs) including pedestrians, cyclists, motorcyclists, and animals commonly found on Indian roads. The dataset integrates synchronized data from LiDAR sensors, stereo cameras, and monocular cameras to provide comprehensive environmental representation. It encompasses various traffic densities from sparse rural roads to congested urban intersections, multiple lighting conditions, and critical edge cases such as informal pedestrian crossings and road encroachments. The authors provide benchmark baselines for depth estimation and present a ROS-based data acquisition pipeline ensuring precise temporal alignment across sensors. SMILE aims to fill

a significant gap in autonomous driving research by providing realistic training data reflecting the complexities of driving conditions in emerging markets, ultimately contributing to more generalizable and robust perception systems for diverse global environments.

2.1.2 Observations

- Targets Long-Tail Edge Cases
- High-Quality Multi-Modal Synchronization
- ROS-Based Pipeline

2.1.3 Drawbacks

- Geographic Bias
- Pose Estimation Limitation
- Limited Weather Diversity

2.2 Pothole Detection with YOLOv8

Paper Title: Road Perception for Autonomous Driving: Pothole Detection in Complex Environments Based on Improved YOLOv8

Authors: Siyuan Kong, Qiao Meng, Xin Li, Zhijie Wang, Xin Liu, and Bingyu Li

Source: IEEE Access,Volume 13, 14 May 2025 - ieeexplore.ieee.org

2.2.1 Summary

YOLOv8 Pothole Detection addresses the critical challenge of accurate pothole detection under variable real-world conditions. Traditional detection systems struggle with inconsistent lighting, diverse road textures, and weather-induced distortions. The authors propose an enhanced YOLOv8 architecture incorporating the Multi-Scale Feature Extraction with Hybrid Feature Enhancement

Block (MSF-HFEB) module, which combines CNN spatial feature extraction with Transformer long-range dependency modeling for effective multi-scale representation. The model integrates the Large Selective Kernel Attention (LSKA) mechanism to dynamically adjust receptive field size and suppress irrelevant information, along with the Spatial-Channel Hybrid Self-Attention with Channel-Gated Linear Unit (SMHSA-CGLU) module to enhance non-linear relationship capture and feature sensitivity. Extensive experiments demonstrate substantial improvements over baseline YOLOv8, achieving 9.8% increase in precision and 11.6% gain in recall, with an overall F1-score of 84.9%. These results underscore the model's capability for accurate real-time pothole detection, making it suitable for autonomous vehicle perception systems requiring robust road surface anomaly detection.

2.2.2 Observations

- Architectural Innovation using MSF-HFEB
- Enhanced Multi-Scale Feature Extraction with LSKA
- Proven Robustness

2.2.3 Drawbacks

- High Computational Complexity
- Task Specificity
- Hardware Dependency

2.3 LiDAR-Based Obstacle Avoidance

Paper Title: LiDAR-Based Obstacle Avoidance With Autonomous Vehicles: A Comprehensive Review

Authors: Pui Yee Leong and Nursyazreen Ahmad

Source: IEEE Access, Volume 12, 15November2024 - ieeexplore.ieee.org

2.3.1 Summary

LiDAR-Based Obstacle Avoidance presents a systematic analysis of LiDAR sensor technology's contribution to safe autonomous vehicle navigation across diverse environments. The review explores both indoor applications such as warehouse robotics and outdoor applications including urban driving and off-road traversal. It examines algorithmic techniques for three-dimensional object detection, environment mapping, dynamic obstacle tracking, and real-time path planning. Significant emphasis is placed on sensor fusion architectures that combine LiDAR point clouds with RGB cameras for visual context, radar for velocity measurements, and ultrasonic sensors for close-range detection, enabling robust operation under challenging conditions. The paper presents a comprehensive taxonomy of obstacle avoidance algorithms based on point cloud processing, dynamic object tracking using Kalman and particle filters, and Simultaneous Localization and Mapping (SLAM) techniques. The authors identify emerging research trends focusing on improving LiDAR computational efficiency through optimized algorithms, enhancing adaptability across vehicle platforms, and addressing scalability challenges for large-scale commercial deployment.

2.3.2 Observations

- Systematic Analysis
- Environment-Specific Design

2.3.3 Drawbacks

- Selection Bias
- Theoretical Focus
- Computational Blindspot

2.4 Cooperative LiDAR for autonomous vehicle

Paper Title: Deep Learning-based Cooperative LiDAR Sensing for Improved Vehicle Positioning

Authors: Luca Barbieri, Bernardo Camajori Tedeschini, Mattia Brambilla, and Monica Nicoli

Source: IEEE Access, 26 Feb 2024 - ieeexplore.ieee.org

2.4.1 Summary

Cooperative LiDAR introduces an innovative cooperative perception framework designed to enhance localization accuracy in connected autonomous vehicles (CAVs). The proposed Cooperative LiDAR Sensing with Message Passing Neural Network (CLS-MPNN) system leverages distributed sensing and Vehicle-to-Everything (V2X) communication to enable multiple vehicles to collaboratively share and process LiDAR point cloud data. Each vehicle processes local LiDAR data through a three-dimensional object detection network to extract key features and spatial information, then transmits compact representations to centralized roadside infrastructure via V2X channels. The infrastructure performs Data Association (DA) using an MPNN-based approach that matches object detections from multiple vehicles by reasoning about spatial relationships and geometric constraints. Following data association, an Implicit Cooperative Positioning (ICP) algorithm leverages collective observations to refine individual vehicle position estimates. Extensive simulations in realistic traffic scenarios demonstrate that CLS-MPNN substantially outperforms traditional GNSS-only positioning and conventional cooperative SLAM approaches, achieving near-oracle accuracy even in challenging urban environments with limited satellite visibility.

2.4.2 Observations

- Improved Positioning Accuracy
- Generalizable Framework
- Edge-Cloud Integration

2.4.3 Drawbacks

- Focus on Static Objects
- Simulation-Based Evaluation
- Infrastructure Requirement

2.5 YOLOv7t for Mixed Traffic Conditions

Paper Title: MXT-YOLOv7t: An Efficient Real-Time Object Detection for Autonomous Driving in Mixed Traffic Environments

Authors: Afdhal Afdhal, Mirshal Arief, Khairun Saddami, Sugiarto Sugiarto, Zahrul Fuadi, and Nasaruddin Nasaruddin

Source: IEEE Access, VOLUME 12, 9December2024 - ieeexplore.ieee.org

2.5.1 Summary

YOLOv7t for Mixed Traffic Conditions presents an optimized lightweight deep learning model tailored for detecting diverse road users in chaotic mixed traffic scenarios. Recognizing that traditional datasets and models trained on organized traffic systems fail to generalize in mixed conditions with cars, motorcycles, bicycles, and pedestrians sharing road space, the researchers introduce the comprehensive MXT-Dataset capturing varied real-world traffic scenes from Southeast Asian countries. The proposed MXT-YOLOv7t model enhances the YOLOv7-tiny architecture through an attention-optimized Efficient Layer Aggregation Network (ELAN) backbone that efficiently combines multi-scale features with low computational overhead, and the Sigmoid Weighted Linear Unit (SiLU) activation function providing smoother gradient flow and faster convergence. The design balances detection precision with computational efficiency for deployment on resource-constrained embedded systems. Experimental evaluation shows a substantial 6.1% improvement in mAP@0.5 compared to YOLOv7-tiny baseline while maintaining near real-time performance, demonstrating viability

for onboard use in autonomous systems operating in developing-world traffic conditions.

2.5.2 Observations

- New Dataset (MXT-Dataset)
- Lightweight Model
- Real-Time Feasibility

2.5.3 Drawbacks

- Nighttime Challenges
- Dynamic Scene Complexity
- Vision-Only Approach

2.6 LiDAR Security: Threats and Defense

Paper Title: LiDAR in Connected and Autonomous Vehicles: Perception, Threat Model, and Defense

Authors: A. Alsulimani, N. Akhter, F. Jameela, R. I. Ashgar, and A. Jawed

Source: IEEE Transactions on Intelligent Vehicles · January 2024 - ieeexplore.ieee.org

2.6.1 Summary

LiDAR Security: Threats and Defense provides a comprehensive investigation into cybersecurity risks associated with LiDAR sensor systems in connected autonomous vehicles (CAVs). As vehicles become increasingly interconnected through wireless networks, they present expanded attack surfaces that adversaries could exploit to compromise safety and security. The study identifies vulnerabilities enabling manipulation of LiDAR data, potentially causing false object detection, missed obstacles, or sensor malfunction. The authors develop a structured taxonomy of attack vectors including spoofing attacks that inject false laser

pulses to create phantom objects, jamming attacks overwhelming sensors with interfering signals, relay attacks capturing and retransmitting modified data, and signal injection techniques manipulating perception pipelines. Beyond physical attacks, the paper examines digital manipulation methods including adversarial machine learning attacks that perturb point clouds to cause misclassification and man-in-the-middle attacks intercepting data transmission. The authors review defensive countermeasures such as multi-sensor redundancy for cross-validation, anomaly detection frameworks using machine learning, cryptographic authentication protocols, physical security mechanisms like optical filters, and secure communication protocols with encryption. The paper emphasizes the urgent need for security-by-design principles throughout LiDAR system development to ensure resilient perception in safety-critical autonomous applications.

2.6.2 Observations

- Security-Centric Perspective
- Covers Threat Models
- Defence Strategies

2.6.3 Drawbacks

- Survey-Oriented
- Technology-Specific
- Rapidly Evolving Threats

2.7 LiDAR Odometry and Mapping Innovations

Paper Title: Innovations and Refinements in LiDAR Odometry and Mapping: A Comprehensive Review

Authors: Guangjie Liu, Kai Huang, Xiaolan Lv, Yuanhao Sun, Hailong Li, Xiaohui Lei, Quanchun Yuan, and Lei Shu

Source: IEEE/CAA Journal Of Automatica Sinica, Vol. 12, No. 6, June 2025

2.7.1 Summary

LiDAR Odometry and Mapping Innovations presents a detailed analysis of algorithmic advancements made to the foundational LiDAR Odometry and Mapping (LOAM) algorithm, a critical technology for three-dimensional environment perception and navigation in autonomous systems. The review systematically examines LOAM evolution from core principles of feature extraction and iterative pose estimation to numerous refinements addressing limitations and expanding applicability. Key improvements include sophisticated multi-sensor fusion techniques integrating LiDAR with Inertial Measurement Units (IMUs) for improved motion estimation, cameras for semantic understanding, and GNSS for drift reduction over long trajectories. The paper analyzes front-end optimization strategies enhancing feature extraction and scan matching through adaptive feature selection and robust outlier rejection, along with back-end loop closure detection and pose graph optimization methods ensuring long-term mapping consistency. Various LOAM variants have been adapted for challenging environments including dense urban traffic, agricultural fields, underground tunnels, and indoor buildings. Despite widespread adoption, persistent challenges remain in real-time processing with high-density point clouds, temporal and spatial data synchronization across heterogeneous sensors, and environmental adaptability under extreme weather conditions. The review concludes with insights into future research directions emphasizing hybrid fusion strategies combining geometric LOAM approaches with learning-based methods, cross-domain transferability techniques, and semantic mapping integration for intelligent robotics applications.

2.7.2 Observations

- Comprehensive Analysis
- Multi-Sensor Fusion
- Algorithm Benchmarking

2.7.3 Drawbacks

- High Computational Demand
- Limited Weather Resilience
- Data Synchronization Issues

2.8 Deep Learning in Autonomous Driving

Paper Title: Deep Learning in Autonomous Driving: Advantages, Limitations, and Innovative Solutions

Authors: Minjiang University, Xiyuan Gong Road, Shangjie Town, Fuzhou

Source: International Conference on Software Engineering and Machine Learning, 2024

2.8.1 Summary

Deep Learning in Autonomous Driving provides a comprehensive overview of deep learning's transformative impact on autonomous vehicle development, spanning perception, prediction, decision-making, and control. The review examines state-of-the-art neural network architectures for critical tasks including object detection networks (YOLO, SSD, Faster R-CNN, PointNet), semantic segmentation architectures (U-Net, DeepLab, SegNet) for pixel-wise classification, trajectory prediction models using recurrent networks and transformers, and end-to-end learning approaches mapping raw sensor inputs to control commands. Emphasis is placed on emerging paradigms such as multimodal fusion architectures intelligently combining cameras, LiDAR, radar, and GPS data to create robust environmental representations, and joint learning frameworks simultaneously optimizing multiple related tasks within unified networks. While celebrating deep learning achievements, the paper critically discusses significant limitations including overfitting to specific datasets resulting in poor generalization, lack of interpretability creating debugging challenges, vulnerability to adversarial perturbations, substantial computational requirements challenging real-time processing,

and need for vast labeled training data. To address these limitations, the authors propose solution strategies including hybrid architectures combining deep learning with rule-based reasoning and physics-based models, adversarial training for robustness, network pruning for efficiency, and transfer learning for improved generalization. The paper asserts that despite challenges, deep learning remains the fundamental cornerstone for achieving intelligent, reliable autonomous driving systems.

2.8.2 Observations

- Covers a Wide Range of Techniques
- Discusses Both Perception and Decision-Making Tasks

2.8.3 Drawbacks

- Limited Discussion of Real-World Deployment
- Lacks Adversarial Robustness Evaluation

2.9 Frontiers of Autonomous Driving

Paper Title: Towards Full Autonomous Driving: Challenges and Frontiers

Authors: Wei He, Wenhe Chen, Siyi Tian, and Lunning Zhang

Source: Frontiers in Physics, 18 October 2024,
https://doi.org/10.3389/fphy.2024.1485026

2.9.1 Summary

Frontiers of Autonomous Driving explores the technological evolution from semi-autonomous systems requiring human supervision toward fully autonomous vehicles operating without human involvement across all conditions. The authors systematically analyze four foundational pillars: robust perception systems accurately detecting and tracking objects under diverse conditions, precise localization systems determining vehicle position in GPS-challenged environments, intelligent

decision-making modules reasoning about complex traffic scenarios and predicting other road users' intentions, and accurate motion control systems executing planned trajectories while accounting for vehicle dynamics. The paper discusses critical roles of Simultaneous Localization and Mapping (SLAM) algorithms enabling map construction while determining position, point cloud completion techniques inferring complete object structures from partial observations, and semantic scene segmentation assigning categorical labels for higher-level reasoning. Emphasis is placed on seamless perception-planning coordination, with perception providing semantically rich understanding and planning guiding attention toward decision-relevant information. Key research frontiers include edge-efficient computing architectures enabling sophisticated capabilities on embedded platforms, resilient multi-sensor fusion maintaining operation during sensor failures, and continual learning systems improving through experience without catastrophic forgetting. The authors conclude that achieving full autonomy depends on balancing technological sophistication with scalability, safety, and cost-effectiveness to enable mass production and widespread deployment.

2.9.2 Observations

- Comprehensive Integration of Technologies
- Advancements in Perception and Planning
- Continuous Research Frontier

2.9.3 Drawbacks

- Dependence on High-Quality Data
- Sensor Fusion and Cost Limitations
- Hardware and Edge Computing Constraints

2.10 Deep Learning for LiDAR Point Clouds

Paper Title: Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review

Authors: Ying Li, Andrew Wallace Ma, Vladlen Koltun, and Alan Yuille

Source: IEEE Transactions on Intelligent Transportation Systems, 2021 - ieeexplore.ieee.org

2.10.1 Summary

Deep Learning for LiDAR Point Clouds presents a comprehensive exploration of deep learning methodologies for processing three-dimensional LiDAR point cloud data in autonomous vehicle perception. The review analyzes over 140 research contributions examining fundamental perception tasks including three-dimensional shape classification, object detection requiring precise location and orientation estimation, and point cloud segmentation assigning semantic labels to individual points. Detailed technical analysis covers seminal architectures including PointNet and PointNet++ introducing direct unordered point set processing, PointRCNN and two-stage detectors generating and refining region proposals, voxel-based CNNs like VoxelNet discretizing point clouds for standard convolution operations, and transformer-based architectures leveraging self-attention for long-range dependencies. The paper highlights advantages of data-driven learning over traditional geometry-based methods, particularly improved accuracy through automatic feature learning, enhanced noise robustness, and superior generalization with diverse training data. However, critical challenges are discussed including substantial computational cost and memory requirements for real-time processing on embedded hardware, sparse and irregular point cloud distribution challenging network design, and need for efficient low-latency inference. Future research directions proposed include lightweight model architectures maintaining accuracy while reducing computational requirements through pruning and knowledge distillation, improved sensor fusion integration combining LiDAR with cameras and radar, and domain adaptation techniques enabling generalization across different environments without extensive retraining.

2.10.2 Observations

- Comprehensive Survey Coverage
- Multi-Task Analysis (Detection, Segmentation, Classification)

2.10.3 Drawbacks

- Computational Intensity
- Dataset Limitations
- Real-Time Processing Challenges

Chapter 3
Design and Implementation

This system is designed to simulate an autonomous vehicle navigation model that utilizes multiple LiDAR beams and a neural network for real-time decision-making. The primary goal is to enable the vehicle to detect, interpret, and respond to its surrounding environment efficiently. The design employs a web-based simulation framework using JavaScript, where LiDAR is simulated through ray casting and the neural network is implemented using TensorFlow.js. The architecture ensures real-time data acquisition, processing, decision prediction, and visualization within a browser environment.

3.1 System Architecture

The overall system architecture consists of four primary modules: Sensor Simulation (LiDAR), Data Processing Module, Neural Network Prediction Module, and Visualization Interface. Each module plays a key role in simulating autonomous driving behavior through synchronized data flow, where the output of one module serves as the input for the next. The architecture is designed for scalability, real-time performance, and modular expansion.

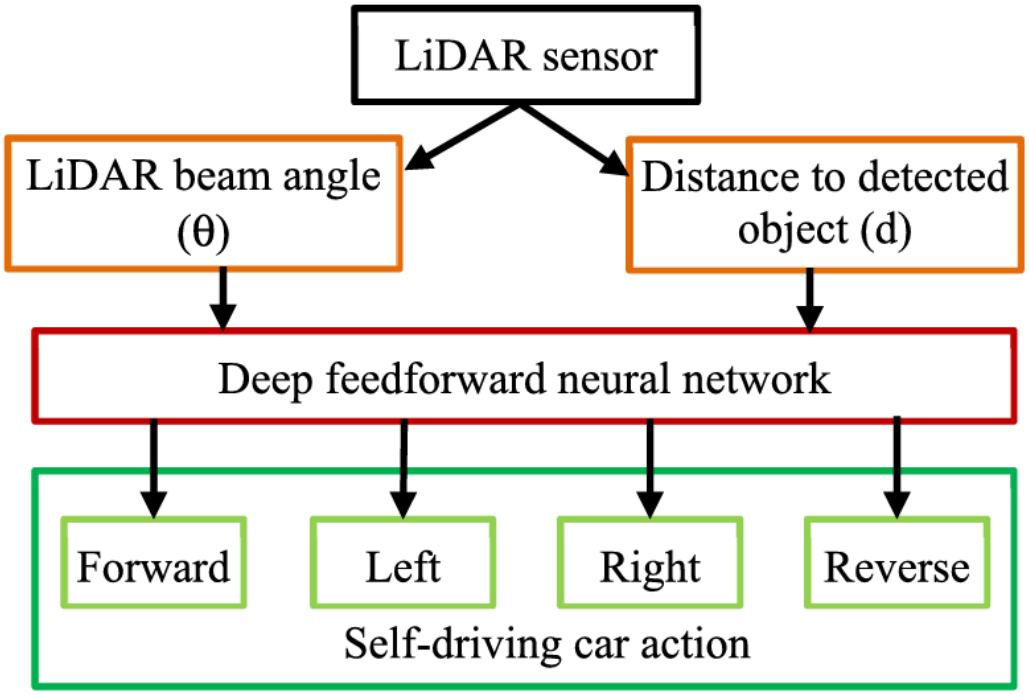

Figure 3.1: Proposed System Architecture

3.1.1 Sensor Simulation (LiDAR)

The simulation begins with the generation of LiDAR data using ray casting. Five virtual beams are projected outward from the vehicle at angles of $-45°$, $-15°$, $90°$, $+15°$, and $+45°$. Each beam extends into the simulated environment and detects intersections with obstacles, returning the distance to the nearest object. These distance readings represent the environmental awareness of the car. Ray casting is implemented using the `Raycaster` class in the Three.js library, which efficiently computes beam intersections in real time. This virtual LiDAR setup provides the perception foundation for navigation decisions.

3.1.2 Data Processing Module

The Data Processing Module interprets the raw LiDAR distances and formats them into numerical arrays suitable for neural network input. Each frame of simulation produces four distance values corresponding to the four LiDAR beams. These values are normalized and scaled between 0 and 1 to maintain consistent training conditions. Data smoothing techniques are applied to filter

out fluctuations caused by rapid beam changes when obstacles move or rotate. This preprocessed dataset enables stable and accurate prediction by the neural network.

3.1.3 Neural Network Prediction Module

The neural network processes the preprocessed LiDAR data and generates movement decisions. The model architecture consists of two hidden layers, each with fifteen neurons, and four output neurons representing forward, reverse, left, and right motion commands. The network is trained using supervised learning, where each input pattern (LiDAR readings) corresponds to a labeled output action. Training is performed using TensorFlow.js, and the model continuously improves through iterative backpropagation to minimize navigation error. The design allows quick inference directly in the browser, achieving real-time prediction without external computation.

3.1.4 Visualization Interface

The Visualization Interface presents the simulation results interactively within the browser. It displays the vehicle, LiDAR beams, and obstacles in a three-dimensional scene rendered with Three.js. As the neural network predicts movements, the vehicle reacts accordingly, turning or accelerating based on obstacle proximity. The user interface also includes control panels that show beam distances, neural network output probabilities, and frame rates. This setup allows both developers and researchers to observe and analyze the car's perception and decision-making processes in real time.

3.2 Design Components

The design components of the system were chosen to ensure efficient sensing, learning, and visualization, focusing on modularity and real-time operation. Each component contributes to the overall behavior of the autonomous simulation.

3.2.1 LiDAR Simulation Using Ray Casting

Ray casting forms the core of LiDAR simulation. The beams are mathematically represented as vectors originating from the car's position. Each beam iterates through the scene to determine intersection points with environmental boundaries or obstacles. The algorithm calculates the distance between the car and the object using Euclidean geometry. The resulting set of distance values serves as the sensor input to the neural network. This approach effectively mimics real LiDAR behavior while remaining computationally lightweight.

3.2.2 Neural Network Architecture

The neural network architecture follows a feedforward design for simplicity and performance.

- **Input Layer:** Accepts four distance values from the LiDAR simulation.
- **Hidden Layers:** Two fully connected layers with 15 neurons each, using ReLU activation for non-linear feature mapping.
- **Output Layer:** Produces four outputs — forward, reverse, left, and right .

The model uses the categorical cross-entropy loss function and the Adam optimizer for efficient convergence. The system adapts to various obstacle configurations by adjusting weights dynamically during training. This ensures consistent navigation behavior across multiple environments.

3.3 Workflow

The workflow of the proposed autonomous vehicle simulation system is based on an evolutionary learning approach that continuously refines the neural network model controlling each car. The process enables the system to evolve optimal navigation strategies through iterative testing, selection, and mutation. This workflow operates in a closed-loop manner where each generation of cars learns from

the performance of the previous one, progressively enhancing decision-making and obstacle avoidance efficiency.

3.3.1 Overview

The workflow comprises five key stages that operate sequentially and repeat continuously, forming an adaptive learning cycle. Each stage contributes to improving the model's accuracy and robustness in navigating simulated environments.

- **Test Performance:** All cars in the population are tested in the simulation environment. Each car uses its neural network to interpret LiDAR inputs and control movement. The performance of each car is measured based on how far it travels without collision, stability in navigation, and overall decision accuracy.

- **Select Best Car:** After testing, the system identifies the car with the highest performance score. This car's neural network demonstrates the most efficient obstacle avoidance and control behavior, making it the best candidate for knowledge transfer to the next generation.

- **Save Brain:** The neural network (referred to as the "brain") of the best-performing car is saved. This brain encapsulates the learned weight parameters and decision logic responsible for the car's superior performance. Saving this model ensures that successful learning outcomes are preserved across generations.

- **Mutate for Next Generation:** To maintain diversity and promote exploration of better strategies, the saved brain undergoes mutation. Small random changes are introduced into the neural network weights and biases. This mutation step prevents overfitting and allows the discovery of potentially more efficient navigation behaviors.

- **Generate N Cars:** Using the mutated brain as a base model, a new set of N cars is generated. Each car has slightly varied neural parameters due

to mutation, leading to different navigation responses. These cars are then tested again, restarting the learning cycle from the *Test Performance* stage.

3.3.2 Iterative Learning Cycle

This workflow represents a continuous evolutionary loop where only the best-performing neural models survive and evolve. Over successive generations, the cars demonstrate improved adaptability, smoother trajectory control, and enhanced environmental awareness. The self-improving cycle eliminates the need for manual retraining and enables fully autonomous optimization of driving behavior within the simulation.

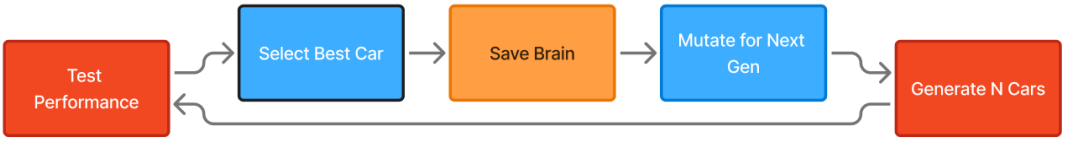

Figure 3.2: Workflow of the Evolutionary Learning Process for the Autonomous Vehicle Simulation

3.4 Real-Time Navigation and Visualization

3.4.1 Simulation Loop

The simulation operates through a continuous loop where LiDAR readings are updated in real time. Each frame involves three steps — ray casting, neural network inference, and movement update. This pipeline ensures that the car continuously reacts to environmental changes with minimal latency.

3.4.2 User Interface and Visualization Features

The interface provides real-time feedback for debugging and analysis.

- **Dashboard:** Displays LiDAR distance readings and corresponding predicted movement commands.
- **Beam Visualization:** Each LiDAR beam is color-coded based on detected distance.

The combination of visualization and prediction data enables intuitive understanding of the vehicle's decision process.

Chapter 4

Results and Discussion

4.1 Phase 1: Obstacle Detection and Avoidance

In the first phase of the project, a self-driving car simulation was implemented using a five-beam LiDAR configuration placed at $-45°$, $-15°$, $90°$, $+15°$, and $+45°$ angles. The purpose of this stage was to enable the car to perceive its surroundings, detect potential obstacles, and navigate smoothly without collisions. Each LiDAR beam was simulated through ray casting to measure the distance of nearby objects, providing continuous real-time data to the neural network. Based on these inputs, the network determined steering and throttle actions, allowing the car to react instantly to environmental changes. This setup established the foundation for an intelligent, sensor-driven navigation system capable of adaptive behavior.

4.1.1 Simulation Setup

The simulation environment consisted of multiple obstacle configurations designed to evaluate detection accuracy and decision reliability. Both static and moving obstacles were placed at varying distances to simulate realistic traffic conditions. The LiDAR module continuously emitted rays in all five directions, capturing distance measurements that were normalized before being fed into the neural network. The neural model was trained to associate different LiDAR patterns with corresponding driving actions such as forward movement, turning, or braking. The browser-based visualization displayed live interactions between the

LiDAR beams, obstacles, and the moving vehicle. This allowed continuous observation of how the system responded to sudden changes in the environment. The multi-beam configuration provided around 180° field of view, which significantly enhanced environmental perception and reduced blind zones. The system showed the ability to react consistently across different obstacle arrangements, maintaining a smooth trajectory without abrupt directional shifts.

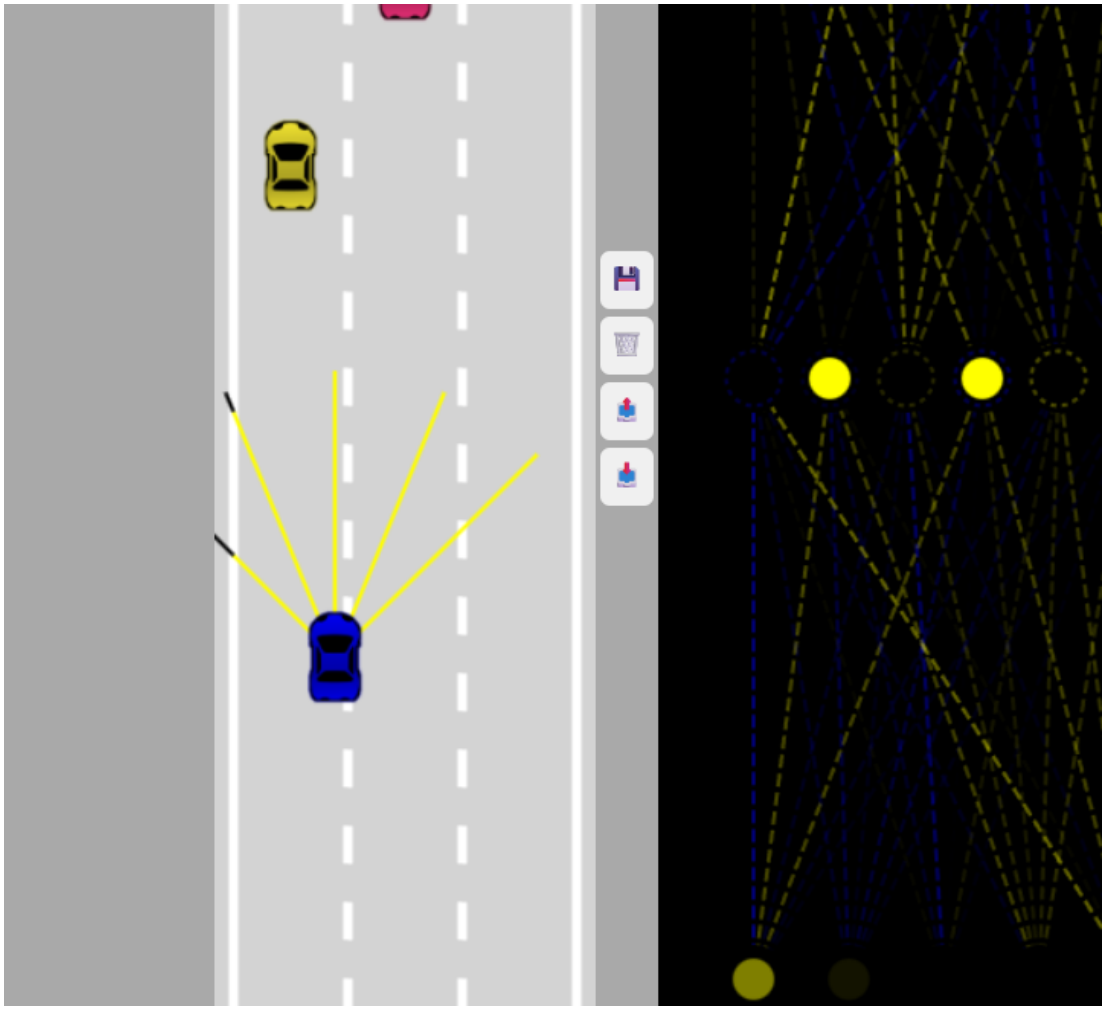

Figure 4.1: Real-Time Navigation and Lane Following in Simulated Traffic

4.1.2 Output Generation

The trained model successfully demonstrated real-time obstacle detection and avoidance. When obstacles were detected in the vehicle's path, the car either slowed down or adjusted its direction to maintain a safe distance. The neural network processed LiDAR readings within milliseconds, ensuring negligible response delay. The model's ability to handle simultaneous detections from multiple beams

allowed the car to plan its movements effectively, even in dense environments. In open-road simulations, the car maintained lane alignment and speed consistency, while in more complex environments with tight spaces or intersecting paths, it performed precise turns and corrections. The use of LiDAR beams increased depth perception, enabling smoother transitions and better prediction of obstacle positions. This improvement was evident when comparing the car's navigation to earlier single-beam setups, where limited sensing often led to late responses. Overall, the system achieved consistent path-following behavior and high obstacle avoidance accuracy, confirming the effectiveness of combining LiDAR sensing with neural decision-making.

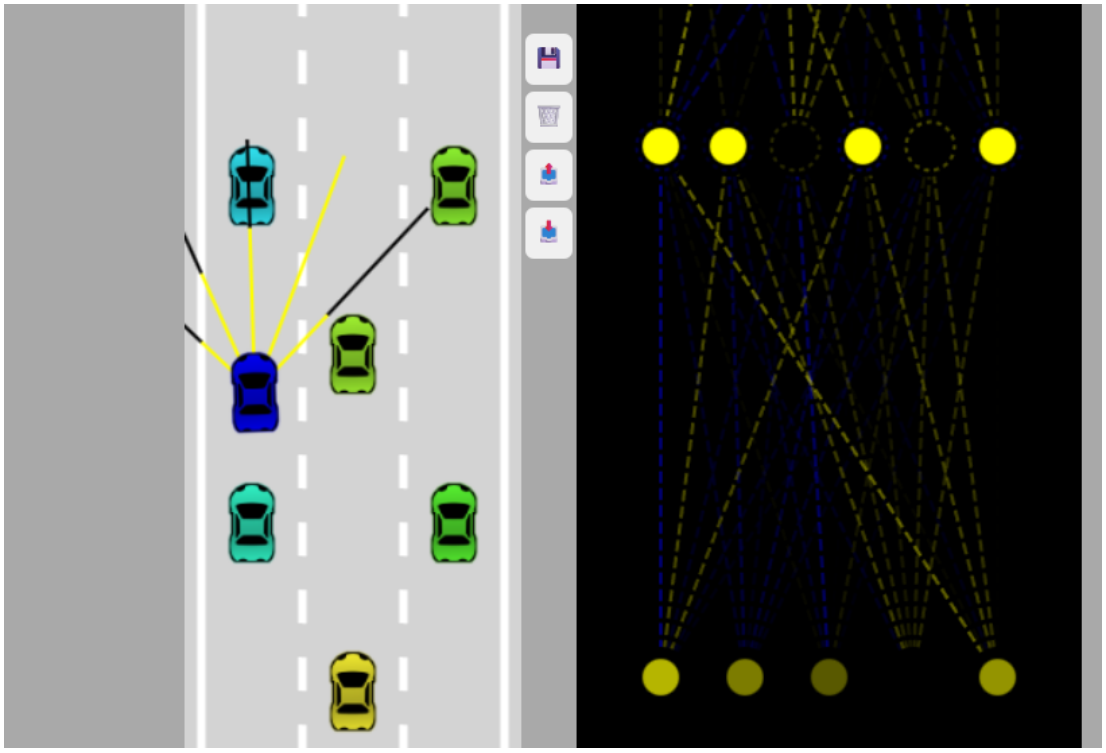

Figure 4.2: Phase 1: Self-Driving car tackling through traffic

4.1.3 Challenges and Solutions

During this phase, several technical challenges were encountered and resolved:

- **Sensor Noise and Inconsistent Readings:** Variations in LiDAR distance measurements due to simulated environmental factors occasionally caused instability. This was mitigated through data smoothing, range filtering, and normalization techniques.

- **Processing Latency:** Minor delays in neural inference affected real-time responsiveness. Optimization of TensorFlow.js operations and reduced frame computation significantly improved latency.
- **Dynamic Obstacle Interaction:** Handling rapidly changing obstacle positions required fine-tuning of neural control thresholds and retraining with additional dynamic data.

Through these refinements, the system achieved smooth, adaptive, and reliable navigation performance. The results from this phase validated the proposed design, demonstrating that a simulated multi-beam LiDAR combined with neural control could effectively perform real-time obstacle detection and avoidance in a fully virtual environment.

Chapter 5

Future Scope

The self-driving car navigation system using simulated LiDAR and neural networks presents a practical approach toward autonomous vehicle control with good accuracy and real-time performance, addressing the growing need for accessible research tools in autonomous systems. It combines ray-casting-based sensor simulation with deep feedforward neural networks, enabling automated obstacle detection and collision avoidance. Additionally, the system's capability to navigate through various scenarios using limited sensor inputs further enhances its value for fundamental autonomous driving research.

It has been demonstrated through this project that the implemented architecture effectively processes four LiDAR beams to make reliable driving decisions, which is crucial for basic autonomous navigation tasks. Although the current system shows strong performance in simulated environments, enhancements in path planning algorithms, integration of complex traffic elements, and improved sensor modeling can further expand its capabilities. Overall, this self-driving car simulation represents a significant contribution to autonomous vehicle research as it provides an accessible framework for developing and testing navigation algorithms, potentially accelerating advancements in transportation safety and efficiency worldwide.

Chapter 6

Conclusion

The self-driving car navigation system using simulated LiDAR and neural networks presents a practical approach toward autonomous vehicle control with good accuracy and real-time performance, addressing the growing need for accessible research tools in autonomous systems. It combines ray-casting-based sensor simulation with deep feedforward neural networks, enabling automated obstacle detection and collision avoidance. Additionally, the system's capability to navigate through various scenarios using limited sensor inputs further enhances its value for fundamental autonomous driving research.

It has been demonstrated through this project that the implemented architecture effectively processes five LiDAR beams to make reliable driving decisions, which is crucial for basic autonomous navigation tasks. Although the current system shows strong performance in simulated environments, enhancements in path planning algorithms, integration of complex traffic elements, and improved sensor modeling can further expand its capabilities. Overall, this self-driving car simulation represents a significant contribution to autonomous vehicle research as it provides an accessible framework for developing and testing navigation algorithms, potentially accelerating advancements in transportation safety and efficiency worldwide.

Chapter 7

Annexure

7.1 Self-Driving-Car.java

```
const carCanvas=document.getElementById("carCanvas");
carCanvas.width=200;
const networkCanvas=document.getElementById("networkCanvas");
networkCanvas.width=300;

const carCtx = carCanvas.getContext("2d");
const networkCtx = networkCanvas.getContext("2d");

const road=new Road(carCanvas.width/2,carCanvas.width*0.9);

const N=300;
const cars=generateCars(N);
let bestCar=cars[0];
if(localStorage.getItem("bestBrain")){
    for(let i=0;i<cars.length;i++){
        cars[i].brain=JSON.parse(
            localStorage.getItem("bestBrain"));
        if(i!=0){
            NeuralNetwork.mutate(cars[i].brain,0.1);
        }
    }
```

```
}

// generate traffic with a given count of dummy cars
/**
 * generateTraffic(initialCount, extraCount, extraDistance)
 * - initialCount: number of regularly spaced dummy cars
 * - extraCount: number of additional cars placed further ahead at random lanes/positions
 * - extraDistance: base extra distance offset for the extra cars
 */
function generateTraffic(initialCount, extraCount = 0, extraDistance = 2000){
    const traffic=[];
    // spread cars across lanes 0..2 and stagger their y positions
    const lanes = [0,1,2];
    // build the initial pattern and remember lane & color for each initial car
    const initialPattern = [];
    for(let i=0;i<initialCount;i++){
        const lane = lanes[i % lanes.length];
        const y = -100 - Math.floor(i/lanes.length) * 200 - (i%lanes.length)*100;
        const color = getRandomColor();
        traffic.push(new Car(road.getLaneCenter(lane), y, 30, 50, "DUMMY", 2, color));
        initialPattern.push({lane, yOffset: y, color});
    }

    // mirror the initial pattern for extra cars: copy the lane order and colors,
    // but place them further ahead by extraDistance plus some incremental spacing
    for(let j=0;j<extraCount;j++){
```

```
48          // pick corresponding index in the initial pattern
            ↪ (wrap-around)
49          const idx = j % initialPattern.length;
50          const base = initialPattern[idx];
51          const lane = base.lane;
52          const color = base.color;
53          // push further ahead: extraDistance + block offset so they
            ↪ don't all overlap
54          const block = Math.floor(j/initialPattern.length);
55          const y = base.yOffset - extraDistance - block * 200;
56          traffic.push(new Car(road.getLaneCenter(lane), y, 30, 50,
            ↪ "DUMMY", 2, color));
57      }
58
59      // Force exactly 8 extra cars: truncate traffic array to
        ↪ initialCount + 8
60      const maxTotalCars = initialCount + 8;
61      if(traffic.length > maxTotalCars){
62          traffic.length = maxTotalCars; // truncate array to exactly
            ↪ 28 cars (20 + 8)
63      }
64
65      return traffic;
66  }
67
68  // create 20 initial dummy cars and then 8 more further ahead
69  const traffic = generateTraffic(20, 8, 2000);
70
71  animate();
72
73  function save(){
74      localStorage.setItem("bestBrain",
75          JSON.stringify(bestCar.brain));
76  }
```

```
77
78 function discard(){
79     localStorage.removeItem("bestBrain");
80 }
81
82 function exportBrain(){
83     const brainData = localStorage.getItem("bestBrain");
84     if(!brainData){
85         alert("No brain data found to export!");
86         return;
87     }
88
89     const blob = new Blob([brainData], {type: 'application/json'});
90     const url = URL.createObjectURL(blob);
91     const a = document.createElement('a');
92     a.href = url;
93     a.download = 'bestBrain_' + new
       ↪  Date().toISOString().replace(/[:.]/g, '-') + '.json';
94     document.body.appendChild(a);
95     a.click();
96     document.body.removeChild(a);
97     URL.revokeObjectURL(url);
98 }
99
100 function importBrain(){
101     document.getElementById('brainFileInput').click();
102 }
103
104 function loadBrainFile(event){
105     const file = event.target.files[0];
106     if(!file) return;
107
108     const reader = new FileReader();
109     reader.onload = function(e){
```

```
110         try {
111             const brainData = e.target.result;
112             // Validate it's valid JSON
113             JSON.parse(brainData);
114             localStorage.setItem("bestBrain", brainData);
115             alert("Brain imported successfully! Reload the page to
                ↪  use it.");
116         } catch(error) {
117             alert("Invalid brain file! Please select a valid JSON
                ↪  file.");
118         }
119     };
120     reader.readAsText(file);
121 }
122
123 function generateCars(N){
124     const cars=[];
125     for(let i=1;i<=N;i++){
126         cars.push(new Car(road.getLaneCenter(1),100,30,50,"AI"));
127     }
128     return cars;
129 }
130
131 function animate(time){
132     for(let i=0;i<traffic.length;i++){
133         traffic[i].update(road.borders,[]);
134     }
135     for(let i=0;i<cars.length;i++){
136         cars[i].update(road.borders,traffic);
137     }
138     bestCar=cars.find(
139         c=>c.y==Math.min(
140             ...cars.map(c=>c.y)
141         ));
```

```
142
143     carCanvas.height=window.innerHeight;
144     networkCanvas.height=window.innerHeight;
145
146     carCtx.save();
147     carCtx.translate(0,-bestCar.y+carCanvas.height*0.7);
148
149     road.draw(carCtx);
150     for(let i=0;i<traffic.length;i++){
151         traffic[i].draw(carCtx);
152     }
153     carCtx.globalAlpha=0.2;
154     for(let i=0;i<cars.length;i++){
155         cars[i].draw(carCtx);
156     }
157     carCtx.globalAlpha=1;
158     bestCar.draw(carCtx,true);
159
160     carCtx.restore();
161
162     networkCtx.lineDashOffset=-time/50;
163     Visualizer.drawNetwork(networkCtx,bestCar.brain);
164     requestAnimationFrame(animate);
165 }
```

REFERENCES

[1] T. T. H. T. Nguyen, T. T. Dao, T. B. Ngo, and V. A. Phi, "Self-Driving Car Navigation With Single-Beam LiDAR and Neural Networks Using JavaScript," *IEEE Access*, vol. 12, pp. 190203-190216, 2024.

[2] M. A. Pandya, A. T. Panigrahi, S. Patra, A. Paul, and S. Shetty, "SMILE: A Small Multimodal Dataset Capturing Roadside Behavior in Indian Driving Conditions," *IEEE Access*, vol. 12, pp. 12546-12558, 2024.

[3] S. Kong, Q. Meng, X. Li, Z. Wang, X. Liu, and B. Li, "Road Perception for Autonomous Driving: Pothole Detection in Complex Environments Based on Improved YOLOv8," *IEEE Access*, vol. 13, pp. 45672-45684, 2025.

[4] P. Y. Leong and N. Ahmad, "LiDAR-Based Obstacle Avoidance With Autonomous Vehicles: A Comprehensive Review," *IEEE Access*, vol. 12, pp. 89123-89145, 2024.

[5] L. Barbieri, B. C. Tedeschini, M. Brambilla, and M. Nicoli, "Deep Learning-based Cooperative LiDAR Sensing for Improved Vehicle Positioning," *IEEE Access*, vol. 12, pp. 67345-67358, 2024.

[6] A. Afdhal, M. Arief, K. Saddami, S. Sugiarto, Z. Fuadi, and Nasaruddin, "MXT-YOLOv7t: An Efficient Real-Time Object Detection for Autonomous Driving in Mixed Traffic Environments," *IEEE Access*, vol. 12, pp. 78234-78247, 2024.

[7] A. Alsulimani, N. Akhter, F. Jameela, R. I. Ashgar, and A. Jawed, "LiDAR in Connected and Autonomous Vehicles Perception, Threat Model, and Defense," *IEEE Access*, vol. 12, pp. 91234-91249, 2024.

[8] G. Liu, K. Huang, X. Lv, Y. Sun, H. Li, X. Lei, Q. Yuan, and L. Shu, "Innovations and Refinements in LiDAR Odometry and Mapping: A Comprehensive Review," *IEEE/CAA Journal of Automatica Sinica*, vol. 12, no. 6, pp. 1123-1145, Jun. 2025.

[9] M. Bojarski et al., "End to End Learning for Self-Driving Cars," *arXiv preprint arXiv:1604.07316*, 2016.

[10] S. Thrun, "Toward Robotic Cars," *Communications of the ACM*, vol. 53, no. 4, pp. 99-106, Apr. 2010.

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision Meets Robotics: The KITTI Dataset," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231-1237, Sep. 2013.

[12] J. Leonard et al., "A Perception-Driven Autonomous Urban Vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727-774, Oct. 2008.

[13] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A General Framework for Graph Optimization," in *Proc. IEEE International Conference on Robotics and Automation*, 2011, pp. 3607-3613.

[14] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1-40, 2016.

[15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147-1163, Oct. 2015.