A Comparative Analysis for Autonomous Vehicle Navigation: Integrating Behavioral Cloning, Path Planning, and NEAT

1st Vinayak Kumar Singh
School of Computer Science and Engineering
Vellore Institute of Technology
Chennai, India
vinayakkumar.singh2023@vitstudent.ac.in

2nd Lekshmi Kalinathan
School of Computer Science and Engineering
Vellore Institute of Technology Chennai, India
Chennai, India
lekshmi.k@vit.ac.in

***Abstract*—This paper presents a comprehensive study of autonomous vehicle navigation, integrating three distinct approaches: Behavioral Cloning, Path Planning, and NeuroEvolution of Augmenting Topologies (NEAT). The research focuses on developing and comparing these methodologies within simulated environments to address the complex challenges of self-driving cars. Behavioral Cloning is implemented to train neural networks on human driving data, while Path Planning algorithms are explored for optimal route generation and obstacle avoidance. The study introduces a novel application of the NEAT algorithm in a custom-built car racing simulator, evolving neural networks for adaptive vehicle control. By leveraging Pygame for visualization and physics simulation, the project creates a testbed for evaluating autonomous driving strategies. The comparative analysis of these approaches provides insights into their relative strengths and limitations, contributing to the advancement of robust and adaptable autonomous driving systems. This research aims to bridge the gap between traditional algorithms and evolutionary computation in the context of self-driving cars, potentially informing future developments in autonomous vehicle technology.**

***Keywords*—Autonomous Vehicles, Behavioral Cloning, Path Planning,NEAT, Self-Driving Cars, Neural Networks, Simulation, Computer Vision, Artificial Intelligence**

I. Introduction

Autonomous vehicle navigation has emerged as a critical field of research and development in recent years, driven by the potential to revolutionize transportation systems and enhance road safety. The quest for efficient and reliable navigation algorithms has led to the exploration of various approaches, ranging from traditional path-planning methods to more advanced artificial intelligence techniques. As the complexity of urban environments and the demands for real-time decision-making increase, the need for robust and adaptable navigation systems becomes paramount.

This study aims to compare the performance of three distinct approaches in the context of autonomous vehicle navigation: Behavioral Cloning, Path Planning, and the NeuroEvolution of Augmenting Topologies (NEAT) algorithm. By integrating these methodologies, we seek to identify the strengths and limitations of each approach and explore potential synergies between them.

Traditional algorithms for autonomous vehicle navigation have long been the backbone of path planning and obstacle avoidance systems. These methods have proven effective in various scenarios, offering reliable solutions for navigating static environments. However, these traditional approaches often struggle with dynamic environments, real-time adaptation, and handling uncertainties inherent in real-world driving scenarios.

In contrast, NEAT represents a more recent and innovative approach to solving complex control problems, including autonomous vehicle navigation.

Developed by Kenneth Stanley and Risto Miikkulainen in 2002, NEAT combines the power of evolutionary algorithms with artificial neural networks, allowing for the simultaneous evolution of network topology and connection weights.

Autonomous vehicle navigation is a complex system that integrates various components and algorithms to perceive the environment, make decisions, and control the vehicle's movements safely and efficiently. Key components include perception systems utilizing sensors like cameras, LiDAR, radar, and ultrasonic sensors; data processing using AI and machine learning algorithms; localization combining GPS, high-definition maps, and inertial measurement units; path planning for determining optimal routes; control systems managing steering, acceleration, and braking; and communication systems enabling information exchange with other vehicles and infrastructure.

Despite significant progress, several challenges remain in autonomous vehicle navigation. These include sensor fusion and perception, environmental adaptability, edge case handling, high-definition mapping, cybersecurity, regulatory and legal frameworks, public trust and acceptance, infrastructure readiness, ethical decision-making, and cost and scalability.

This project focuses on leveraging three distinct approaches—Behavioral Cloning, Path Planning, and the NeuroEvolution of Augmenting Topologies (NEAT) algorithm—to enhance autonomous vehicle navigation. The primary goal is to explore and compare these methodologies within simulated environments, with a particular focus on developing a car racing simulator using the NEAT algorithm.

The main objectives of this study are:

- To implement Behavioral Cloning for autonomous driving using neural networks trained on human driving data.
- To develop and evaluate Path Planning algorithms for optimal route generation and obstacle avoidance.
- To create a car racing simulation environment using Pygame, incorporating the NEAT algorithm for evolving neural networks to control autonomous cars.
- To compare the performance of Behavioral Cloning, Path Planning, and NEAT approaches in various driving scenarios.
- To analyze the adaptability and efficiency of each approach in handling complex and dynamic environments.
- To explore potential integration strategies to combine the strengths of these different approaches.

The scope of this project encompasses several key technical aspects and components, including:

- The development of a 2D racing environment using Pygame for visualization and physics simulation.
- Implementation of Behavioral Cloning using neural networks trained on human driving data.
- Exploration of Path Planning algorithms for autonomous navigation.
- Implementation of the NEAT algorithm for evolving neural networks in the context of car racing.
- Design and implementation of sensor systems, car control mechanisms, and fitness evaluation criteria.
- Comparative analysis of the three approaches in terms of performance, adaptability, and efficiency.

II. Literature Survey

Recent advancements in autonomous driving technologies have made significant progress across several domains, including simulation environments, sensor fusion, object detection, and end-to-end planning. This section reviews relevant works in these areas.

A. Simulation Environments

Simulation plays a critical role in the development and validation of autonomous driving systems. The Vehicle-in-Virtual-Environment (VVE) method introduced in [1] provides a safer and more cost-effective alternative to traditional road testing by offering controlled yet realistic testing scenarios. Building upon this, Waymax [7] introduces a hardware-accelerated, differentiable simulator leveraging real-world driving data to address the sim-

to-real performance gap, enabling faster and more accurate multi-agent interaction simulations.

A comprehensive analysis of open-source simulators [10] highlights critical factors for simulation fidelity and computational efficiency. These studies emphasize the need for improved realism and standardization in simulation platforms for enhanced algorithm development. Additionally, work using the Udacity Self-Driving Vehicle Simulator [11] showcases the integration of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) for training autonomous driving models, demonstrating the importance of model adaptability in diverse driving conditions.

B. Object Detection Systems

Object detection is a cornerstone of autonomous vehicle perception. An enhanced version of YOLOv5 [2] addresses small vehicle and pedestrian detection using structural re-parameterization and neural architecture search, achieving significant improvements in detection accuracy and speed. Similarly, YOLOv8 [3] demonstrates robust performance in mixed traffic environments, with the YOLOv8x variant excelling in handling complex scenarios.

Further advancements are presented in lightweight architectures for small object detection [4], incorporating kernel pruning techniques to enhance traffic sign and light detection. The MCS-YOLO algorithm [13], integrating coordinate attention mechanisms and Swin Transformer architecture, achieves superior performance, particularly in road environment recognition.

C. Sensor Fusion

The integration of multiple sensor modalities has emerged as a critical research area. Multi-modal 3D object detection techniques [5] focus on fusing camera and LiDAR data for improved perception accuracy. Similarly, radar-camera fusion [15] offers enhanced object detection and semantic segmentation, particularly under challenging weather or low-light conditions.

A detailed review of 3D object detection technologies [12] examines LiDAR, camera, and multi-modal approaches, emphasizing the importance of robust and efficient sensor integration. Deep learning-based 3D object detection from images [14] highlights ongoing challenges in extracting accurate 3D information from 2D images, presenting opportunities for further advancements in this domain.

D. End-to-End Planning Approaches

End-to-end learning has gained traction in autonomous driving research. The Pix2Planning framework [6] integrates planning as a language generation task using vision-language models, achieving state-of-the-art performance on CARLA benchmarks. This method addresses accumulated errors and information loss in traditional systems.

Lane detection under challenging conditions has also seen progress. Research integrating the UET-STDC framework with Zero-DCE++ [8] enhances lane segmentation performance in low-light conditions while maintaining real-time processing, addressing critical safety concerns.

E. Evolutionary Computation in Autonomous Driving

Evolutionary computation has demonstrated significant potential in autonomous vehicle control. Togelius et al. [7] showcased the ability of evolutionary algorithms to handle complex racing tasks in simulations. Similarly, Cardamone et al. [9] applied neuroevolution for optimizing racing line trajectories, achieving improvements over traditional approaches.

The NEAT algorithm represents a promising evolutionary approach, simultaneously evolving neural network topology and weights. By combining elements of evolutionary computation, neural networks, and simulation environments, this study seeks to develop a self-learning car racing simulation that can adapt and optimize vehicle control strategies.

III. Behavioral Cloning for Autonomous Driving

Behavioral cloning is a supervised learning technique that trains a model to replicate human driving behavior. By using sensor data (e.g., camera images) as input and corresponding driving actions

(e.g., steering angles, throttle values) as output, the model learns to map perception to control actions. This section details the data collection, preprocessing, model design, and evaluation processes.

A. Data Collection and Preprocessing

The data for behavioral cloning is collected using the Udacity Self-Driving Car Simulator, which provides a virtual environment for safe and controlled experimentation. Two operational modes are utilized:

1) Training Mode: In training mode, a human driver manually controls the car, generating the following data:

- Center, left, and right camera images.
- Steering angles, throttle, and speed values.

This data is saved in a CSV file, where each row corresponds to an image and its associated control values.

2) Autonomous Mode: In autonomous mode, the trained model controls the vehicle. The simulator feeds camera images to the model, which predicts control actions in real-time.

3) Data Augmentation: To improve generalization and robustness, several data augmentation techniques are applied dynamically during training:

- **Horizontal Flipping:** Images are flipped horizontally, and the corresponding steering angles are inverted.
- **Brightness Adjustment:** Image brightness is altered to simulate varying lighting conditions.
- **Translation:** Images are shifted horizontally and vertically, simulating different lane positions.
- **Shadow Addition:** Artificial shadows are introduced to mimic real-world scenarios.
- **Multi-Camera Inputs:** Images from the left and right cameras are used with adjusted steering angles to simulate recovery from off-center positions.

These techniques enhance data diversity and help the model adapt to varying driving conditions.

B. Neural Network Architecture

The model architecture follows NVIDIA's end-to-end learning framework, designed specifically for self-driving applications. The architecture is summarized as follows:

- **Input Layer:** Accepts $66 \times 200 \times 3$ RGB images.
- **Normalization Layer:** Normalizes pixel values to $[-1, 1]$.
- **Convolutional Layers:**
  - Layer 1: 24 filters, 5×5 kernel, stride 2, ReLU activation.
  - Layer 2: 36 filters, 5×5 kernel, stride 2, ReLU activation.
  - Layer 3: 48 filters, 5×5 kernel, stride 2, ReLU activation.
  - Layer 4: 64 filters, 3×3 kernel, stride 1, ReLU activation.
  - Layer 5: 64 filters, 3×3 kernel, stride 1, ReLU activation.
- **Fully Connected Layers:**
  - Layer 1: 100 neurons, ReLU activation.
  - Layer 2: 50 neurons, ReLU activation.
  - Layer 3: 10 neurons, ReLU activation.
- **Output Layer:** A single neuron with `tanh` activation, producing steering angle predictions in $[-1, 1]$.

C. Training Process

1) Loss Function and Optimizer: The model uses Mean Squared Error (MSE) as the loss function to minimize the difference between predicted and actual steering angles. The Adam optimizer is employed for its adaptive learning capabilities.

2) Hyperparameter Tuning: Several hyperparameters are tuned to optimize the model's performance:

- Batch size: Set to 32 for efficient gradient updates.
- Learning rate: Experimented with values ranging from 0.0001 to 0.001.
- Epochs: Training is capped at 50 epochs with early stopping based on validation loss.

3) Model Checkpointing: The best-performing model is saved using a checkpointing mechanism, ensuring that training progress is not lost.

D. Evaluation and Results

The trained model is evaluated using the following metrics:

- **Mean Squared Error (MSE):** Measures the accuracy of steering angle predictions.
- **Autonomous Driving Performance:** Assesses the car's ability to complete laps without leaving the track.
- **Visual Inspection:** Compares predicted and actual steering angles on sample images.

E. Simulation Testing

In autonomous mode, the trained model is tested in the simulator, focusing on:

- Stability on straight roads.
- Smoothness of turns.
- Recovery from off-center positions.
- Performance under varying lighting conditions.

The evaluation demonstrates the model's effectiveness in replicating human driving behavior, while also highlighting areas for improvement, such as handling edge cases or novel scenarios.

IV. Path Planning for Autonomous Driving

Path planning is a critical component of autonomous vehicle navigation, responsible for determining a collision-free and efficient trajectory for the vehicle. It involves making real-time decisions to navigate dynamic environments while adhering to traffic rules. This section elaborates on the sensor fusion, environment perception, behavior planning, and trajectory generation involved in the path planning process.

A. Sensor Fusion and Environment Perception

Effective path planning begins with accurate perception of the environment. This is achieved by fusing data from multiple sensors such as cameras, LiDAR, and radar. Sensor fusion enhances the reliability and completeness of the vehicle's understanding of its surroundings.

1) Sensor Fusion Techniques: Sensor data is processed to detect obstacles, other vehicles, and lane markings. A fusion algorithm combines the strengths of different sensors:

- Cameras provide high-resolution visual data for object detection and classification.
- LiDAR generates detailed 3D maps of the surroundings, enabling precise distance measurements.
- Radar detects objects' velocity, aiding in tracking moving obstacles.

2) Environment Representation: The perception system outputs a comprehensive map of the environment, marking detected objects, free spaces, and dynamic elements. This representation serves as the basis for planning safe and efficient paths.

B. Behavior Planning and Decision Making

Behavior planning determines the high-level actions the vehicle should take, such as lane changes, overtaking, or adjusting speed. These decisions are based on real-time inputs from the perception system.

1) Decision-Making Strategies: The decision-making process evaluates multiple potential actions using a cost function. Factors considered include:

- **Safety:** Avoiding collisions with other vehicles and obstacles.
- **Efficiency:** Maintaining optimal speed and minimizing route length.
- **Comfort:** Reducing sudden changes in acceleration or direction.

2) Cost Function Design: A cost function assigns a numerical value to each action, reflecting its desirability. For example:

$$\text{Cost} = w_1 \cdot C_{\text{collision}} + w_2 \cdot C_{\text{efficiency}} + w_3 \cdot C_{\text{comfort}}, \tag{1}$$

where w_1, w_2, w_3 are weights balancing the importance of different factors.

C. Trajectory Generation

Once a high-level decision is made, a detailed trajectory is generated for the vehicle to follow. This trajectory must be smooth, collision-free, and executable within the vehicle's kinematic constraints.

1) Spline-Based Trajectory Planning: A spline-based approach is used to create smooth paths:

- Anchor points are identified based on the current position, previous path, and future waypoints.
- A spline is fitted to these points, ensuring continuity and smoothness.
- The trajectory is sampled at discrete intervals, generating waypoints for the vehicle to follow.

2) Coordinate Transformations: All calculations are performed in the vehicle's local coordinate system to simplify computations. Points are later transformed back to global coordinates for execution.

D. Evaluation and Results

The path planning system is evaluated using the following metrics:

- **Collision Avoidance:** Ensuring the vehicle maintains safe distances from obstacles and other road users.
- **Trajectory Smoothness:** Quantifying the continuity of the generated paths.
- **Efficiency:** Measuring travel time and adherence to speed limits.

E. Simulation Testing

The path planning algorithm is tested in a simulated highway environment with dynamic traffic. Key performance indicators include:

- Successful execution of lane changes and overtaking maneuvers.
- Responsiveness to sudden obstacles or changes in traffic conditions.
- Maintenance of desired speeds without excessive acceleration or braking.

The evaluation highlights the algorithm's ability to handle complex scenarios while ensuring safety and efficiency. Further improvements, such as enhancing the cost function and trajectory smoothing techniques, are suggested for future work.

V. NeuroEvolution of Augmenting Topologies (NEAT) in Autonomous Driving

The NeuroEvolution of Augmenting Topologies (NEAT) algorithm is a powerful evolutionary approach that evolves both the topology and weights of neural networks. Developed by Stanley and Miikkulainen, NEAT is particularly well-suited for solving control problems in dynamic and complex environments. This section discusses the implementation of NEAT for autonomous driving within a car racing simulator, detailing the genome representation, evolutionary process, simulation environment, and fitness evaluation.

A. Overview of NEAT Algorithm

NEAT is an evolutionary algorithm that optimizes neural networks using genetic operations. Its key features include:

- **Topology Optimization:** NEAT begins with minimal networks and gradually adds complexity by evolving the network structure.
- **Speciation:** Individuals are grouped into species to protect innovative solutions and promote diversity in the population.
- **Crossover and Mutation:** The algorithm combines and mutates genomes to generate offspring, enabling exploration of new solutions.

NEAT's ability to evolve network topology makes it highly adaptable for controlling autonomous vehicles in unpredictable environments.

B. Genome Representation and Network Architecture

Each NEAT individual is represented by a genome encoding the structure and weights of a neural network. The genome consists of:

- **Node Genes:** Represent neurons in the network, including input, hidden, and output nodes.
- **Connection Genes:** Represent synaptic connections between nodes, including attributes like weights, enable/disable status, and innovation numbers.

The network architecture for this project includes:

- **Input Layer:** Five nodes representing radar sensor data.
- **Output Layer:** Four nodes for control actions: steering left, steering right, accelerating, and braking.

- **Hidden Layers:** Dynamically evolved through mutation and crossover.

C. Evolutionary Process

The NEAT evolutionary process involves the following steps:

- **Initialization:** A population of minimal networks is created.
- **Fitness Evaluation:** Each individual is evaluated based on its performance in the car racing simulator.
- **Selection:** The fittest individuals are selected for reproduction.
- **Crossover:** Genetic material from two parents is combined to produce offspring.
- **Mutation:** Random changes are introduced to the genome, including adding nodes or connections and altering weights.
- **Speciation:** Individuals are grouped into species based on genetic similarity, ensuring diverse exploration of the solution space.

1) Configuring NEAT Parameters: Key parameters for the NEAT algorithm include:

- Population size: 30 individuals.
- Mutation probabilities for adding nodes and connections.
- Speciation thresholds to balance diversity and competition.

D. Car Racing Simulation Environment

The simulation environment, implemented using Pygame, provides a controlled platform to evaluate evolved neural networks. Key features include:

- **Track Design:** The track is represented as a 2D image with clearly defined boundaries.
- **Car Model:** Cars are equipped with a physics model to simulate real-world dynamics, including acceleration, braking, and turning.
- **Radar Sensors:** Each car has five virtual sensors providing distance data to obstacles, which serve as inputs to the neural network.
- **Collision Detection:** A mechanism to detect crashes with track boundaries, ending the car's run.

E. Fitness Evaluation

The fitness function is designed to reward efficient and collision-free driving. The fitness score is calculated based on:

- Distance traveled along the track.
- Time survived without collision.
- Speed maintenance, encouraging efficient navigation.

The fitness function is defined as:

$$\text{Fitness} = w_1 \cdot D + w_2 \cdot T + w_3 \cdot S, \qquad (2)$$

where D is the distance traveled, T is the time survived, S is the average speed, and w_1, w_2, w_3 are weighting factors.

F. Evaluation and Results

The performance of the NEAT algorithm is evaluated over multiple generations, focusing on:

- **Convergence:** The number of generations required to evolve high-performing networks.
- **Adaptability:** The ability of networks to handle varying track layouts and conditions.
- **Network Complexity:** The trade-off between network size and performance.

1) Simulation Results: The NEAT algorithm successfully evolves neural networks capable of navigating the track efficiently. Key observations include:

- Faster convergence when initial populations are diverse.
- Improved adaptability with larger populations and longer evolution periods.
- Optimal performance achieved with moderate network complexity, balancing computational efficiency and control precision.

The results demonstrate the potential of NEAT as a robust method for developing control systems for autonomous vehicles, particularly in dynamic and complex scenarios.

VI. Comparative Analysis of Methods

This section compares the performance of the three implemented approaches—Behavioral Cloning, Path Planning, and NeuroEvolution of Augmenting Topologies (NEAT)—based on several

criteria including accuracy, efficiency, adaptability, and computational requirements.

A. Performance Metrics

The evaluation metrics used for comparison are:

- **Accuracy:** The ability of each approach to follow the track without deviations.
- **Efficiency:** Measured by the time taken to complete the track and the average speed maintained.
- **Adaptability:** The capability to handle dynamic changes, such as obstacles or varying track layouts.
- **Computational Cost:** The resources required during training and inference.

B. Results Comparison

Metric	Behavioral Cloning	Path Planning	NEAT
Accuracy	Medium	High	High
Efficiency	High	Medium	Medium
Adaptability	Low	Medium	High
Computational Cost	Low	Medium	High

TABLE I
PERFORMANCE COMPARISON OF METHODS

C. Key Observations

- **Behavioral Cloning:** Performs well in known environments but struggles with unseen scenarios due to overfitting to training data.
- **Path Planning:** Provides reliable and interpretable decisions but is limited by the computational cost of real-time trajectory generation.
- **NEAT:** Excels in adaptability and dynamic environments but requires significant computational resources during evolution.

VII. DISCUSSION

A. Strengths and Limitations of Each Approach

Behavioral Cloning:

- **Strengths:** Simple implementation and fast inference make it suitable for controlled environments.
- **Limitations:** Poor generalization and inability to handle edge cases reduce its practicality in dynamic environments.

Path Planning:

- **Strengths:** Strong in providing interpretable and efficient solutions for static and semi-dynamic environments.
- **Limitations:** Struggles with unpredictable scenarios due to reliance on predefined heuristics and high computational requirements.

NEAT:

- **Strengths:** Adaptive and capable of evolving optimal control strategies for complex scenarios.
- **Limitations:** Computational cost and longer training periods can hinder its real-world applicability.

B. Potential for Hybrid Approaches

A hybrid approach combining the strengths of all three methods could yield a more robust solution. For instance:

- Use Path Planning for high-level decision-making.
- Leverage Behavioral Cloning for low-level, real-time control.
- Employ NEAT to optimize and adapt control strategies for dynamic scenarios.

C. Insights and Future Research Directions

The findings suggest the following areas for further exploration:

- Improving generalization in Behavioral Cloning using diverse training datasets.
- Reducing computational overhead in NEAT by optimizing evolution processes.
- Integrating reinforcement learning with Path Planning to handle dynamic and complex environments.

VIII. CONCLUSION AND FUTURE WORK

A. Summary of Findings

This paper presented a comparative study of Behavioral Cloning, Path Planning, and NEAT for autonomous driving in simulated environments. Key findings include:

- Behavioral Cloning is effective for structured scenarios but lacks adaptability.

- Path Planning provides efficient and interpretable trajectories but requires significant computational resources.
- NEAT demonstrates high adaptability and robustness but incurs high training costs.

B. Contributions

The contributions of this work are:

- Implementation and evaluation of three distinct approaches to autonomous driving.
- Insights into the trade-offs between accuracy, efficiency, and computational cost.
- Recommendations for hybrid approaches to leverage the strengths of each method.

C. Future Work

Future research could focus on:

- Extending the evaluation to real-world driving scenarios.
- Developing hybrid frameworks combining these approaches.
- Exploring the integration of reinforcement learning and evolutionary algorithms.

REFERENCES

[1] Cao, X., Chen, H., Gelbal, S.Y., Aksun-Guvenc, B., and Guvenc, L., 2023. Vehicle-in-Virtual-Environment (VVE) method for autonomous driving system development, evaluation, and demonstration. *Sensors*, 23(11), p.5088.

[2] Jia, X., Tong, Y., Qiao, H., Li, M., Tong, J., and Liang, B., 2023. Fast and accurate object detector for autonomous driving based on improved YOLOv5. *Scientific Reports*, 13(1), p.9711.

[3] Afdhal, A., Saddami, K., Sugiarto, S., Fuadi, Z., and Nasaruddin, N., 2023, August. Real-time object detection performance of YOLOv8 models for self-driving cars in a mixed traffic environment. In *2023 2nd International Conference on Computer System, Information Technology, and Electrical Engineering (COSITE)* (pp. 260-265). IEEE.

[4] Afdhal, A., Saddami, K., Sugiarto, S., Fuadi, Z., and Nasaruddin, N., 2023, August. An improved lightweight small object detection framework applied to real-time autonomous driving. In *2023 2nd International Conference on Computer System, Information Technology, and Electrical Engineering (COSITE)* (pp. 260-265). IEEE.

[5] Wang, Y., Mao, Q., Zhu, H., Deng, J., Zhang, Y., Ji, J., Li, H., and Zhang, Y., 2023. Multi-modal 3D object detection in autonomous driving: a survey. *International Journal of Computer Vision*, 131(8), pp.2122-2152.

[6] Mu, X., Qin, T., Zhang, S., Xu, C., and Yang, M., 2024, June. Pix2Planning: End-to-End Planning by Vision-language Model for Autonomous Driving on Carla Simulator. In *2024 IEEE Intelligent Vehicles Symposium (IV)* (pp. 2383-2390). IEEE.

[7] Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., Harb, J., Pan, X., Wang, Y., Chen, X., and Co-Reyes, J., 2024. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems*, 36.

[8] Liu, Y., Wang, Y., and Li, Q., 2024. Lane detection based on real-time semantic segmentation for end-to-end autonomous driving under low-light conditions. *Digital Signal Processing*, 155, p.104752.

[9] Kindi, H.K., Selviandro, N., and Wulandari, G., 2023. Autonomous vehicle simulation with multi-human driving behavior using deep learning. *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, 8(3), pp.755-763.

[10] Li, Y., Yuan, W., Zhang, S., Yan, W., Shen, Q., Wang, C., and Yang, M., 2024. Choose your simulator wisely: A review on open-source simulators for autonomous driving. *IEEE Transactions on Intelligent Vehicles*.

[11] Parveen, S., and Haq, A.A.U., 2023. Autonomous Vehicles Using Udacity's Self Driving Vehicles Simulator. *Journal of Survey in Fisheries Sciences*, 10(3S), pp.1242-1248.

[12] Mao, J., Shi, S., Wang, X., and Li, H., 2023. 3D object detection for autonomous driving: A comprehensive survey. *International Journal of Computer Vision*, 131(8), pp.1909-1963.

[13] Cao, Y., Li, C., Peng, Y., and Ru, H., 2023. MCS-YOLO: A multiscale object detection method for autonomous driving road environment recognition. *IEEE Access*, 11, pp.22342-22354.

[14] Alaba, S.Y., and Ball, J.E., 2023. Deep learning-based image 3D object detection for autonomous driving. *IEEE Sensors Journal*, 23(4), pp.3378-3394.

[15] Yao, S., Guan, R., Huang, X., Li, Z., Sha, X., Yue, Y., Lim, E.G., Seo, H., Man, K.L., Zhu, X., and Yue, Y., 2023. Radar-camera fusion for object detection and semantic segmentation in autonomous driving: A comprehensive review. *IEEE Transactions on Intelligent Vehicles*.

[16] Hemalatha, T., and Sivakumar, T.K., 2023, October. Advanced self-driving car using CNN: Udacity Simulator. In *International Conference on Artificial Intelligence and Knowledge Processing* (pp. 381-396). Cham: Springer Nature Switzerland.