A MINOR PROJECT REPORT

On

Hybrid AI Car Driving Simulator

Submitted by

Ayan Kundu (10800222015)
Debarshi Chatterjee (10800222014)
Moupiya Das (10800222020)
Rohit Saha (10800222086)

Under the Guidance of

Mr. Biplab Kumar Mondal
Assistant Professor

Information Technology

Asansol Engineering College
Asansol

Affiliated to

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

12/2025

ASANSOL ENGINEERING COLLEGE

Kanyapur, Vivekananda Sarani, Asansol, Paschim Bardhaman, West Bengal - 713305
Phone: 225-3057, 225-2108 Telefax: (0341) 225-6334
E-mail: principal.aecwb@gmail.com Website: www.aecwb.edu.in

CERTIFICATE

Certified that this project report on "Hybrid AI Car Driving Simulator" is the bonafide work of "Ayan Kundu (10800222015), Debarshi Chatterjee(10800222014), Moupiya Das(10800222020),Rohit Saha(10800222086)" who carried out the project work under my supervision.

Mr. Biplab Kumar Mondal
Assistance professor
Dept. of Information Technology
Asansol Engineering College

Dr. Avishek Banerjee
Associate professor & HOD
Dept. of Information Technology
Asansol Engineering College

Information Technology
Asansol Engineering College
Asansol

ACKNOWLEDGEMENT

It is our great privilege to express my profound and sincere gratitude to our **Project Supervisor Mr. Biplab Kumar Mondal, Assistant Professor** for providing me with very cooperative and precious guidance at every stage of the present project work being carried out under his supervision. His valuable advice and instructions in carrying out the present study have been a very rewarding and pleasurable experience that has greatly benefitted us throughout our work.

We would also like to pay our heartiest thanks and gratitude to **Dr. Avishek Banerjee**, HoD, and all the faculty members of the Information Technology, Asansol Engineering College for various suggestions being provided in attaining success in our work.

Finally, we would like to express our deep sense of gratitude to our parents for their constant motivation and support throughout our work.

……………………………………………
Ayan Kundu (10800222015)

……………………………………………
Debarshi Chatterjee (10800222014)

……………………………………………
Moupiya Das (10800222020)

……………………………………………
Rohit Saha (10800222086)

Date: **4th Year**
Place: Asansol **Information Technology**

CONTENT

Certificate ………………………………………………………………………… 2

Acknowledgement …………………………………………………………….. 3

Content ……………………………………………………………………… 4

List of Figures/Tables ……………………………………………………… 5

Abstract………..……………………………………………………………….. 6

1. Introduction …………………………………………………………… 7

    1.1Literature review…………………………………………………... 8-9

    1.2 Motivation ………………………………………………………... 10

    1.3. Contribution……………………………………………………… 11

2. Project Details ………………………………………………………… 12

    2.1 hardware requirements…………………………………………….. 12

    2.2 software requirements…...………………………………………… 12-13

        2.2.1 operating system ……………………………………………….. 12

        2.2.2 runtime enviroments…………………………………………… 13

        2.2.3 required python package……………………………………... 13

    2.3 Performance optimization………………………………………….. 13

    2.4 problem statement………………………………………………….. 13

3. Flow of the work……………………………………………………………….. 14-16

4. Data Flow Diagram & Use case diagram……………………………………. 17- 18

    4.1 Data Flow Diagram…………………………………………………. 17

    4.2 Use case Diagram…………………………………………………… 18

5. proposed system algorithm…………………………………………………… 19-22

5.1 Assumtion……………………………………………………………………… 19

    5.2 Algorithm……………………………………………………………… 19-21

5.3 Linking theories and algorithm………………………………………………… 22

6. Simulation results……………………………………….…………………….. 23-27

    6.1 Experimental setup……………………….………………………….. 23-24

    6.2 Experimental result…………………..……………………………. 25-27

    6.3. Comparative Analysis of Pure ML and Hybrid AI Models………….. 27

7. Conclusion & Future Work…………….…………………………………… 28

8. References…………………………..………………………………………. 29-30

LIST OF TABLES

Table	Name of Table	Page Number
Table 1	Linking Theories and Algorithms	22
Table 2	Comparative Analysis of Pure ML and Hybrid AI Models:	27

LIST OF FIGURES

Figure	Name of the Figure	Page Number
Figure 1	Project Interface	14
Figure 2	Level 0 DFD	17
Figure 3	Level 1 DFD	17
Figure 4	User case Diagram	18
Figure 5	Project Interface	23
Figure 6	Data Logging System	24
Figure 7	Model Comparison	25
Figure 8	Sensor Data	26

ABSTRACT

This project focuses on developing a Hybrid Intelligence System for Autonomous Vehicle Simulation by integrating rule-based logic, supervised machine learning, and reinforcement learning to improve decision-making accuracy and driving safety. Autonomous vehicles must operate reliably in dynamic environments where single AI techniques often fall short: rule-based systems provide safety but lack adaptability, machine learning enables pattern recognition but struggles in unseen conditions, and reinforcement learning learns through interaction but can be unsafe during early exploration. To overcome these individual limitations, the project implements a hybrid model that combines the strengths of each approach.

A custom 2D car driving simulation environment was created featuring lane structures, traffic lights, moving vehicles, obstacles, and collision boundaries. The autonomous vehicle is equipped with seven ray-casting sensors, mimicking LIDAR, to detect distances to surrounding objects. The system supports multiple modes, including manual data-collection mode, pure machine learning mode, and the proposed hybrid intelligence mode. Core algorithms developed in the project include collision avoidance, lane keeping, traffic light compliance, Random Forest–based steering prediction, Q-learning–based reinforcement learning, and a decision fusion algorithm that ensures rule-based safety overrides all learned actions.

Simulation results show that the hybrid model provides significantly safer and more stable driving compared to pure machine learning, particularly in unfamiliar or complex scenarios. The vehicle demonstrates improved collision avoidance, smoother steering, and strategic long-term decision-making. This project establishes a practical framework for understanding autonomous driving challenges and highlights that hybrid intelligence is essential for building reliable autonomous systems. The work also prepares the foundation for future enhancements such as 3D simulation, advanced sensor modeling, multi-agent interaction, and real-world hardware implementation.

1. Introduction:

Intelligent autonomous systems have gained considerable importance due to their potential to enhance safety, efficiency, and productivity across various sectors. One of the most prominent applications of such systems is autonomous vehicle technology, where artificial intelligence (AI) models are trained and validated in simulation before being deployed in real-world environments. Simulation offers a safe, low-cost, and fully controllable platform for testing decision-making capabilities under diverse and dynamic conditions. This report analyzes an AI-driven 2D autonomous driving simulation system that integrates machine learning, reinforcement learning, and hybrid AI techniques to develop a reliable and adaptable driving agent.

The simulation environment is designed to emulate core elements of real-world driving, including lane following, obstacle avoidance, traffic light handling, and maneuvering in restricted spaces. The system consists of key modules such as a training engine, evaluation routines, a visualization interface, and a perception subsystem based on ray-casting sensors. These sensors mimic LiDAR-like distance measurements and provide essential environmental data. Machine learning models—such as classification and regression algorithms—predict steering and control actions using labeled datasets, while a Q-learning framework enables reinforcement learning by rewarding safe behaviors and penalizing unsafe ones.

A major strength of this system is its hybrid AI approach, which combines rule-based logic with learned decision-making. Rule-based components enforce strict safety constraints, ensuring predictable responses in critical situations, while machine-learned policies offer flexibility and adaptability in complex or unfamiliar scenarios. This hybrid strategy addresses the limitations of single-method systems and aligns with current trends in autonomous vehicle research, where safety and learning must coexist.

Data logging plays a vital role in performance evaluation. Logs capture time-series data on rewards, actions, sensor readings, violations, and failure events, enabling detailed analysis of behavior and model accuracy. However, the present analysis revealed gaps in the data logging pipeline: several metrics—including cumulative rewards, failure events, and safety violations—remained zero throughout the simulation. These inconsistencies suggest issues in data capture mechanisms and highlight the need for improved logging to ensure reliable evaluation.

Despite these limitations, comparative results show that the hybrid AI system outperforms the pure machine learning approach. It achieves higher rewards, fewer safety violations, and more stable driving patterns, demonstrating the effectiveness of combining deterministic safety rules with adaptive learning models. Sensor analysis further confirms consistent perception accuracy, with strong correlations between adjacent sensors and reliable distance measurements that support navigation.

In summary, this simulation framework provides a robust platform for developing and analyzing autonomous driving intelligence. Its integration of hybrid AI strategies, sensor-based perception, and learning mechanisms offers valuable insights into safe and effective autonomous navigation. Addressing data logging issues will further enhance system reliability and support future research in more advanced simulations, neural architectures, and real-world autonomous vehicle applications.

1.1 Literature Review:

Artificial intelligence research has progressed toward integrated frameworks capable of interpreting complex sensory inputs, making real-time decisions, and interacting safely within dynamic simulated environments. Simulation-based AI forms the foundation of modern autonomous systems, including robotics, intelligent navigation, and adaptive decision-making agents. Traditional supervised learning approaches such as classification and regression models have long been used to approximate functional relationships in data, enabling systems to predict actions, speeds, or environmental outcomes. These models operate effectively when datasets are representative and structured, a pattern well established in machine-learning research [1, 2]. Within simulated navigation environments, they often guide obstacle detection, movement prediction, or decision selection.

Despite their strengths, supervised models face limitations in sequential or uncertain environments. Purely data-driven systems cannot reliably handle unpredictable obstacles, long-term dependencies, or delayed effects of actions. Literature consistently emphasizes that supervised learning alone is insufficient for autonomous navigation and control tasks, where uncertainty and sequential reasoning dominate [3]. This gap motivated the rise of hybrid AI frameworks combining machine learning with rule-based logic. Hybrid systems enforce strict safety constraints through symbolic rules while using machine-learning components for flexible adaptation. Russell and Norvig [3] identify hybrid reasoning as essential for safety-critical AI. Empirical studies support that hybrid agents outperform purely learned policies by reducing unsafe behaviors and responding more reliably to environmental shifts [4, 5].

Reinforcement learning (RL) represents another major paradigm for decision-making in simulation environments. RL allows agents to learn through interactions and reward feedback, and has achieved notable results in robotics, navigation, and control systems. However, RL is highly sensitive to reward design, data quality, and logging accuracy. Research highlights that poorly shaped or sparsely recorded rewards often result in stagnant learning, flat accuracy curves, or zero-valued reward trajectories [6–8]. Studies warn that misaligned logging—such as missing timestamps or reward events—can render RL agents unable to learn meaningful policies, even when the environment functions correctly. Issues such as instability during early training, risk-seeking behavior from poorly balanced rewards, and incorrect metric aggregation are common pitfalls documented across RL literature [7, 8]. These challenges underscore why hybrid systems, which constrain RL actions through rule-based safety layers, typically demonstrate more stable performance.

Sensor processing plays an important role in both simulated and real-world autonomous systems. Research in robotics and probabilistic perception shows that sensor arrays frequently exhibit strong correlations, particularly between adjacent sensors capturing related spatial information [9, 10]. Such correlations arise from overlapping sensor ranges or shared environmental features. Variance in sensor readings is often interpreted as a sign of environmental complexity or unstable agent behavior. Understanding these patterns is vital for constructing reliable state representations in both ML and RL systems, especially when decisions depend directly on sensor feedback.

Evaluation of autonomous agents depends on accurate measurement of reward progression, safety violations, task completion, and failure events. Recent studies highlight that discrepancies between aggregate metrics and frame-level data often stem from logging errors, misordered timestamps, or incorrect parsing routines [11, 12]. When time-series plots display all-zero values despite non-zero aggregated performance scores, it commonly signals underlying issues in data logging and parsing rather than agent failure. Henderson et al. [11] emphasize the importance of validating raw logs prior to any smoothing or averaging, while Machado et al. [12] demonstrate that many RL experiments misrepresent performance due to improper logging pipelines.

Overall, the current body of literature demonstrates several consistent trends: hybrid AI architectures generally outperform pure machine-learning or reinforcement-learning models in safety-critical simulations; RL systems require carefully constructed rewards and validated logging mechanisms to function properly; sensor data tends to provide reliable structural information; and logging pipelines are often the source of error in simulation-based AI research. These findings align closely with the patterns observed in the system under study, where the hybrid model performs significantly better than the pure ML alternative, sensor data exhibits meaningful structure and correlations, and several time-series metrics appear unreliable due to logging gaps. Collectively, the literature underscores the necessity of robust hybrid architectures, consistent data pipelines, and validated perception systems in the development of modern AI simulation frameworks.

1.2 Motivation:

Autonomous systems rely heavily on accurate decision-making, stable navigation, and reliable sensor interpretation, especially in dynamic environments. Pure machine-learning models often struggle with unpredictable changes, leading to poor performance and higher violation rates. These challenges create the need for a more adaptable and dependable approach that can maintain safety while improving efficiency.

This project is motivated by the potential of hybrid AI, which combines learning-based models with rule-driven safety layers. By analyzing sensor behavior, system metrics, and comparing Hybrid AI with Pure ML, the project highlights how hybrid methods deliver more consistent and safer outcomes. Simulation-based evaluation further supports this approach, providing a controlled way to refine and improve autonomous decision-making systems.

- **Safety-Critical AI Development -** Pure machine learning systems, while powerful, often fail in rare or hazardous scenarios. For instance, without explicit safety rules, an AI agent may accelerate aggressively near obstacles, misinterpret traffic signals, or miscalculate turns. By incorporating hybrid AI strategies that combine machine learning with rule-based constraints, the system ensures safer behavior even in edge cases.

- **Controlled Testing Environment -** Testing autonomous vehicles in the real world is costly, time-consuming, and potentially dangerous. A simulated 2D environment provides a risk-free platform to explore AI behavior under a wide range of traffic conditions. Researchers can test extreme scenarios, evaluate rare event responses, and iterate rapidly without risking physical assets or human safety.

- **Understanding Perception Systems -** Autonomous navigation depends on accurate perception of the surrounding environment. By analyzing sensor readings and their correlations, the project enables a deeper understanding of how environmental awareness affects decision-making. This insight is essential for designing more effective sensors, improving data preprocessing, and enhancing the robustness of AI models.

- **Benchmarking AI Strategies -** Comparing Hybrid AI against Pure ML agents allows researchers to empirically validate the effectiveness of integrating rules with learned behaviors. The hybrid approach demonstrated significant improvements in average reward, reduction in safety violations, and more stable navigation, confirming that intelligent rule integration is critical for real-world applicability.

- **Algorithm Optimization -** The project allows for iterative testing and tuning of both machine learning and reinforcement learning strategies. Metrics such as average reward, violation rate, and speed provide quantitative feedback to optimize policies, refine reward functions, and calibrate AI behavior.

1.3. <u>**Contribution**</u>:

This project offers meaningful contributions to the field of Autonomous Systems, Hybrid AI, and Simulation-Based Evaluation. By combining machine learning with rule-based safety logic, the project presents a more stable and interpretable approach to analyzing AI behavior in dynamic environments. It improves decision accuracy, reduces violations, and provides clearer insights into system performance through structured visualization.

- **Hybrid AI for Improved Safety** - Introduces a combined rule-based and ML architecture that reduces unsafe actions and increases reward performance compared to pure ML methods.
- **Comparative AI Performance Analysis** - Evaluates Hybrid AI and Pure ML using reward, violation rate, and speed, clearly showing the advantages of hybrid systems in dynamic scenarios.
- **Sensor Data Interpretation** - Provides statistical insights into sensor averages, variance, and correlations to better understand environmental perception and agent behavior.
- **Visual Analytics for System Behavior** - Develops meaningful charts (reward comparison, sensor correlations, safety indicators) that help users understand and evaluate AI decisions more clearly.
- **Detection of Logging Issues** - Identifies gaps in reward, accuracy, and movement tracking, contributing to more reliable data collection and system debugging.
- **Scalable Simulation Environment** - Offers a safe and flexible platform to test AI models repeatedly without real-world risks, supporting research and development.
- **Applicability to Autonomous Systems** - The findings and hybrid strategy can support robotics, smart navigation, warehouse automation, and other AI-driven applications.

Through these contributions, the project enhances the stability, safety, and interpretability of AI-driven decision-making, supporting future development in hybrid autonomous systems.

2. PROJECT DETAILS:

Hybrid Intelligence System for Autonomous Vehicles is a 2D top-down simulation integrating rule-based logic, machine learning (ML), and reinforcement learning (RL) to control a virtual autonomous vehicle. The system demonstrates how hybrid AI can improve safety, adaptability, and decision-making in autonomous driving. The simulation provides an interactive environment with sensors, dynamic obstacles, traffic elements, and evaluation metrics, enabling the comparison of different driving modes.

The simulator, built using Pygame, features structured roads, traffic lights, moving obstacles, and boundaries. The vehicle makes decisions regarding acceleration, braking, and steering based on real-time sensor inputs. Despite its simplified 2D design, the simulation captures key aspects of autonomous driving, including perception, planning, and control, making it suitable for AI experimentation.

A key component is the sensor system, which uses ray-casting to simulate perception. The vehicle has seven directional sensors that detect obstacles, walls, and other vehicles up to a fixed ran.
The system supports Hybrid Mode (AI + rule-based safety) and Pure ML Mode (ML only). In Hybrid Mode, ML models predict steering and speed, while the rule-based module ensures safety, preventing collisions. Pure ML Mode evaluates autonomous performance without safety constraints, highlighting the benefits of hybrid intelligence.

The training system combines supervised learning and reinforcement learning. Manual driving sessions generate data for supervised learning, while RL allows the vehicle to learn optimal strategies through trial-and-error interactions, enabling adaptation to dynamic, multi-goal navigation tasks.

2.1 Hardware Requirements:

- Processor: Dual-core CPU @ 2.0 GHz or higher
- Memory: 4 GB RAM
- Storage: 500 MB free disk space
- Graphics: Integrated graphics with OpenGL 2.1 support
- Display: 1280x720 resolution or higher
- Input Devices: Keyboard and mouse

2.2 Software Requirements:

2.2.1 Operating System

- Windows: Windows 10/11 (64-bit)
- macOS: macOS 10.15 Catalina or later
- Linux: Ubuntu 20.04 LTS or equivalent (with X11/Wayland support)

2.2.2 Runtime Environment:

- Python: Python 3.8 or higher (64-bit recommended)
- Pip: Python package manager (included with Python)

2.2.3 Required Python Packages:

- Pygame 2.5.2: For 2D visualization and user interface
- scikit-learn 1.3.2: Machine learning algorithms (Random Forest, etc.)
- numpy 1.24.3: Numerical computations and array operations
- pandas 2.0.3: Data manipulation and analysis
- matplotlib 3.7.2: Data visualization and plotting
- joblib 1.3.2: Model serialization and parallel processing
- seaborn 0.13.2: Graph rendering and Data visualization

2.3 Performance Optimization:

For optimal performance during training:

- Close unnecessary background applications
- Use SSD storage for faster data access
- Ensure adequate cooling for extended training sessions
- Consider using a dedicated GPU for not stressing the cpu for visual elements

2.4 Problem Statement:

Efficient autonomous navigation is critical in robotics, self-driving vehicles, logistics, and AI-driven decision-making. Traditional shortest-path algorithms, such as Dijkstra’s and A*, are effective for single-start, single-goal static environments but fail in dynamic, multi-goal scenarios. They lack adaptability to changing obstacles, interactions between multiple targets, and require full recomputation for new situations, resulting in inefficient routing, increased computational cost, and suboptimal decisions.

The main challenge addressed in this project is to develop a multi-goal pathfinding algorithm that computes independent paths for each target while avoiding interference from other goals. By integrating Q-learning, the system continuously learns and adapts, optimizing navigation over time. This ensures safe, efficient, and intelligent decision-making in dynamic environments, making it suitable for autonomous vehicles, AI-driven logistics, and real-time navigation applications.

3. Flow of Work:

The workflow for the Car Driving AI Simulation: Hybrid Intelligence System for Autonomous Vehicles is structured to integrate environment modeling, path planning, and reinforcement learning for efficient autonomous navigation. The flow of work is as follows:

Architecture Diagram:

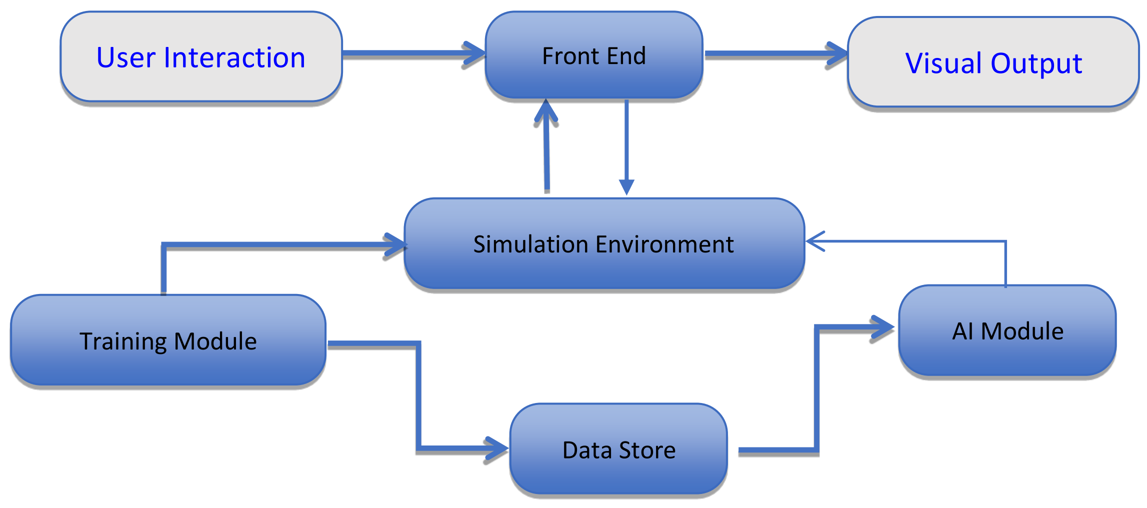


Fig 3.1: Project Architecture

- **Environment Setup :**

    a. Create a 2D simulation environment representing roads, obstacles, and dynamic traffic conditions.

    b. Encode environmental parameters including roads, lanes, obstacles, and target destinations for the vehicle.

- **Input Vehicle and Sensor Data:**

    a. Define the vehicle's starting point, destination, and movement constraints.

    b. Simulate sensors to perceive nearby obstacles, lanes, and traffic signals.

    c. Capture environmental data for obstacle detection and collision avoidance.

- **Path Planning:**

    a. Implement traditional algorithms such as A*, RRT, or APF to generate initial feasible paths.

b. Incorporate dynamic obstacle modeling to adapt the path when new obstacles are detected.

c. Evaluate path safety, distance, and energy efficiency to select optimal routes.

- **Q-Learning Agent Initialization:**

a. Define the state space (e.g., vehicle position, velocity, distance to obstacles).

b. Define the action space (e.g., accelerate, brake, turn left/right, maintain speed).

c. Initialize Q-table to store expected rewards for state-action pairs.

- **Hybrid Intelligence Approach:**

a. Combine Q-learning with traditional path planning strategies.

b. Enhance Q-learning with value updates and reward evaluation to improve offline learning.

c. Use hybrid strategies for dynamic decision-making and obstacle avoidance.

- **Simulation and Learning:**

a. The agent explores the environment and updates Q-values based on reward feedback.

b. The agent iteratively learns the optimal policy for navigation.

c. Apply temporal difference (TD) learning to improve convergence to the optimal path.

- **Path Optimization and Validation:**

a. Evaluate the learned path for efficiency, safety, and travel time.

b. Adjust route planning dynamically if the environment changes during simulation.

c. Compare performance with baseline algorithms to measure improvement.

- **Performance Evaluation:**

a. Record simulation metrics including collision rates, path length, travel time, and energy consumption.

b. Visualize the vehicle's path and decision-making process.

c. Analyze the effectiveness of the hybrid Q-learning strategy over conventional methods.

▪ Findings and Future Enhancements:

a. Summarize key observations from simulation experiments.

b. Suggest improvements for dynamic obstacle handling, hybrid reinforcement strategies, and computational efficiency.

4. Data Flow Diagram & Use case Diagram:

4.1 Data Flow Diagram(DFD):

Level 0 - This diagram shows a simple interaction between users and a Hybrid AI Car Driving Simulator. Users provide input or manual training, and the simulator returns model reports and simulation output, creating a feedback loop.

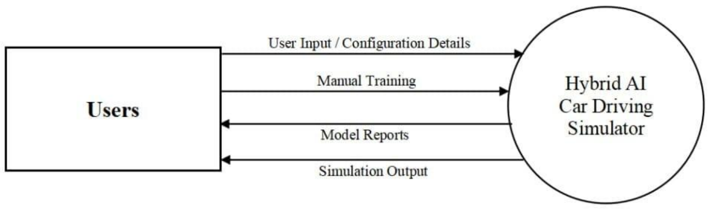

Users
User Input / Configuration Details
Manual Training
Model Reports
Simulation Output
Hybrid AI Car Driving Simulator

Fig 5.1.1: Level 0 DFD

Level 1 - This detailed workflow illustrates how users, the simulation interface, AI module, data store, training module, and data visualization system interact. Simulation data flows through AI prediction, storage, training, and visualization, giving users insights and improved driving simulation performance.

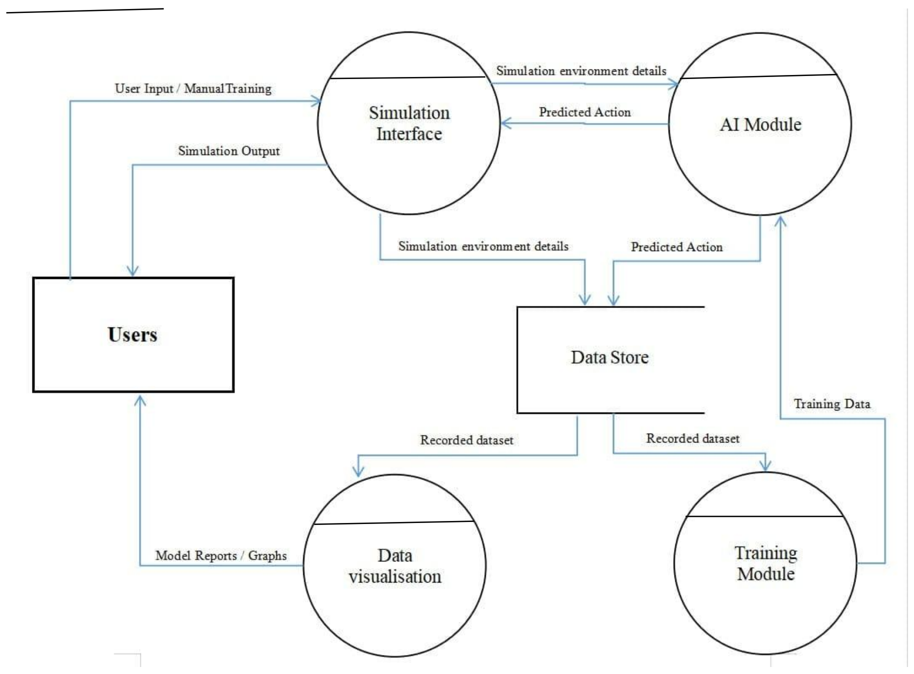


Fig 5.1.2: Level 1 DFD

4.2 Use case Diagarm:

The diagram shows how users interact with the **Car Driving AI Simulation System**.

- **Actors:**
  - **Car Owner** – Trains the system and tests AI driving.
  - **Data Analyst** – Reviews system data to improve AI performance.
- **System Use Cases:**
  - **Manual Training** – Car owner provides driving demonstrations for the AI to learn basic behavior.
  - **AI Driving** – The system autonomously drives using trained models.
  - **Comparison Data** – The system compares manual driving with AI driving; data analyst reviews it.
  - **Training Logs** – Records all training activities, errors, rewards, and learning progress for analysis.
- **Interaction:**
  - The **Car Owner** interacts with Manual Training and AI Driving.
  - The **Data Analyst** works with Comparison Data and Training Logs to enhance the AI system.

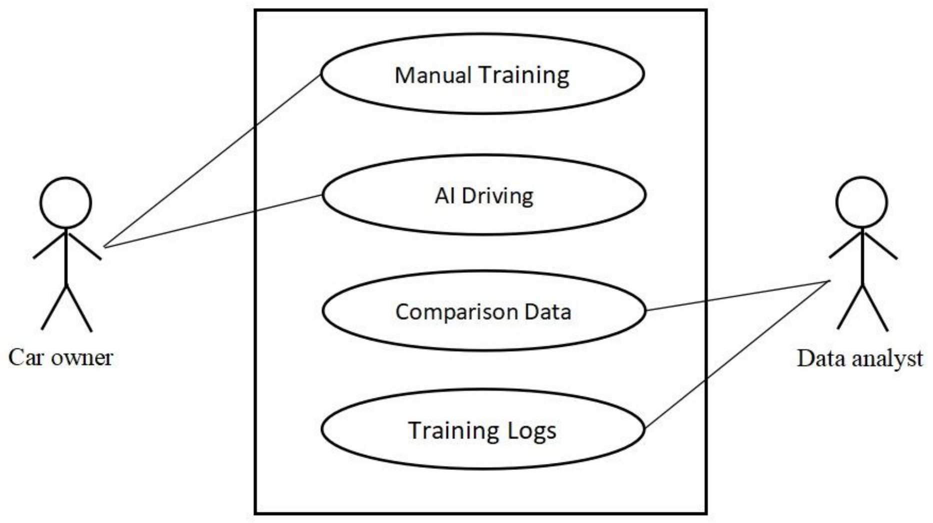


Fig 5.2.1: Use Case Diagram

5. Proposed System Algorithm:

5.1 Assumptions:

1. The vehicle operates in a 2D simulation environment with roads, lanes, traffic lights, and obstacles.
2. Sensors provide accurate detection of nearby objects, lanes, and signals using sensor theory and ray casting techniques.
3. Vehicle actions are discrete: accelerate, decelerate, turn left/right, maintain speed.
4. Dynamic obstacles and traffic conditions may appear randomly.
5. The system integrates reinforcement learning, supervised machine learning, rule-based systems, control theory, and game theory for decision-making.
6. Algorithms converge within available computational resources.
7. Rewards are designed to optimize safety, efficiency, and traffic compliance.

5.2 Algorithm:

Input: Vehicle start point (S), destination point (D), environment data, vehicle parameters
Output: Collision-free, optimized path; performance metrics

Step 1: Environment Initialization

- Load 2D map with static and dynamic obstacles, lanes, and traffic signals.
- Implement sensor modules to detect obstacles, lane boundaries, and traffic lights using ray casting techniques.
- Update environment dynamically in Environment Data Store.

Step 2: Vehicle Initialization

- Define vehicle state: position, speed, orientation.
- Define action space: {accelerate, decelerate, turn left, turn right, maintain speed}.
- Integrate control theory principles to manage vehicle motion smoothly.

Step 3: Initial Path Planning

- Generate initial feasible paths using traditional planning algorithms (A*, RRT, APF).
- Evaluate paths based on collision risk, distance, and energy efficiency.
- Pass candidate paths to Q-learning module for optimization.

Step 4: Reinforcement Learning Module (Q-Learning)

- Initialize Q-table for all state-action pairs.
- Define reward function based on:
- Collision avoidance (enhanced collision avoidance algorithm)
- Lane keeping (adaptive lane keeping algorithm)
- Traffic rule compliance (traffic light compliance algorithm)
- Travel efficiency (shortest path, minimal energy)
- Set learning parameters: learning rate (α), discount factor (γ), exploration rate (ε).

Step 5: Supervised Learning Module

- Apply Random Forest Classification and Regression to predict traffic patterns, obstacle behavior, and lane changes.
- Use predictions to refine Q-learning decision-making and decision fusion algorithm to combine multiple strategies.

Step 6: Hybrid Decision-Making

- Fuse outputs from:
- Q-learning (Reinforcement Learning)
- Supervised predictions (Random Forest)
- Rule-based constraints (traffic rules, lane boundaries)
- Apply game theory principles to handle interactions with other vehicles or dynamic obstacles.

Step 7: Action Execution

- Choose optimal action based on fused decision strategy.
- Apply control theory to execute vehicle maneuvers safely and smoothly.
- Update vehicle state and environment map.

Step 8: Dynamic Path Adjustment

- Continuously monitor environment for new obstacles and traffic changes.
- Adjust path dynamically using Q-learning updates and supervised predictions.
- Ensure traffic rule compliance and collision avoidance in real-time.

Step 9: Output and Performance Analysis

- Display optimized path in the 2D simulation environment.
- Record metrics: collision count, travel time, energy consumption, lane adherence, traffic compliance.
- Compare performance with baseline path planning and decision-making algorithms.

Step 10: Termination

- Stop simulation when vehicle reaches destination.
- Save Q-table, trained models, and environment states for future simulation or learning.

5.3 Linking Theories and Algorithms:

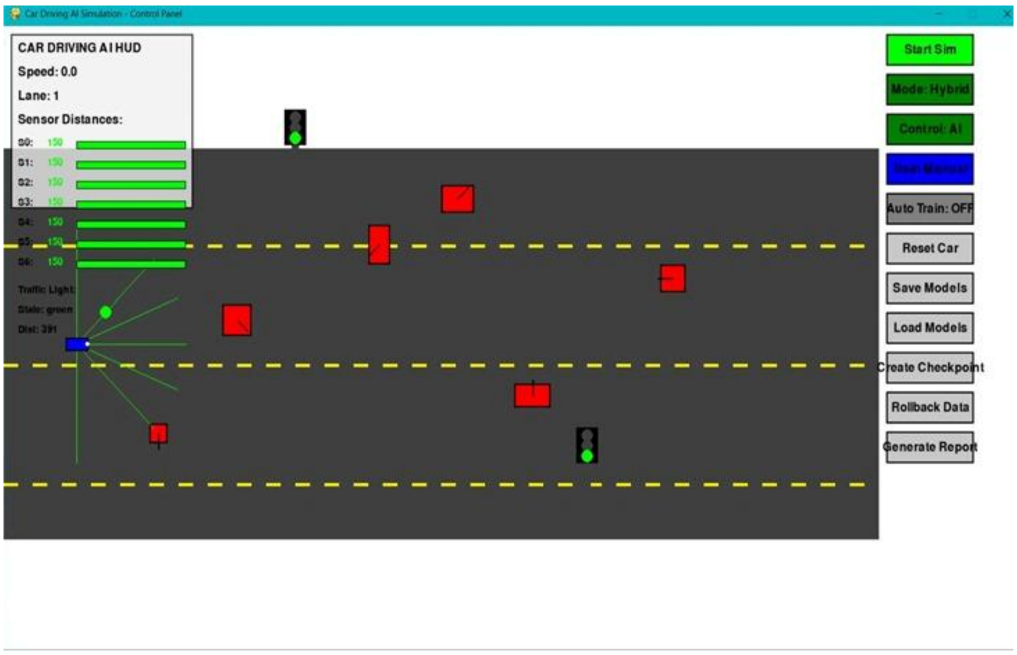
Component	**Theory / Technique**	**Purpose in Simulation**
Path planning	Control theory, Reinforcement learning, Q-learning	Optimize path and vehicle control
Obstacle detection	Sensor theory, Ray casting	Detect static and dynamic obstacles
Collision avoidance	Enhanced collision avoidance algorithm	Ensure safety
Lane keeping	Adaptive keeping algorithm	Maintain lane adherence
Traffic compliance	Rule-based system, Traffic light compliance algorithm	Follow traffic rules
Dynamic decision-making	Decision fusion algorithm, Game theory	Combine multiple strategies for optimal action
Traffic prediction	Random Forest Classification & Regression	Predict traffic flow and obstacles

6. Simulation Results:

6.1 Experimental Setup:

The simulation setup forms the backbone of the hybrid autonomous driving system, providing a controlled, realistic, and flexible environment in which the hybrid AI model can be trained, tested, and evaluated. Although the simulation operates in a 2D plane, the setup has been intentionally designed to replicate essential elements of real-world autonomous driving, including road structures, traffic systems, environmental boundaries, dynamic obstacles, sensor feedback, and computational constraints. The environment not only facilitates safe experimentation but also ensures repeatability across multiple trials—an essential characteristic for analyzing algorithmic performance.

At the core of the simulation lies a **2D top-down driving environment**, implemented using Pygame or a similar graphics engine. This framework renders roads, intersections, traffic lights, lane markings, and environmental boundaries in real time. The top-down format simplifies the computational complexity while still allowing the vehicle to exhibit realistic driving behaviours such as acceleration, braking, turning, collision responses, and obstacle navigation. The simulated world includes both static and dynamic elements. Static elements include road boundaries, walls, and lane divisions, while dynamic elements include moving obstacles (e.g., other vehicles) and timed traffic signals.


Car Driving AI Simulation - Control Panel

CAR DRIVING AI HUD
Speed: 0.0
Lane: 1
Sensor Distances:
S0: 150
S1: 150
S2: 150
S3: 150
S4: 150
S5: 150
S6: 150
Traffic Light:
State: green
Dist: 391

Mode: Hybrid
Control: AI
Auto Train: OFF

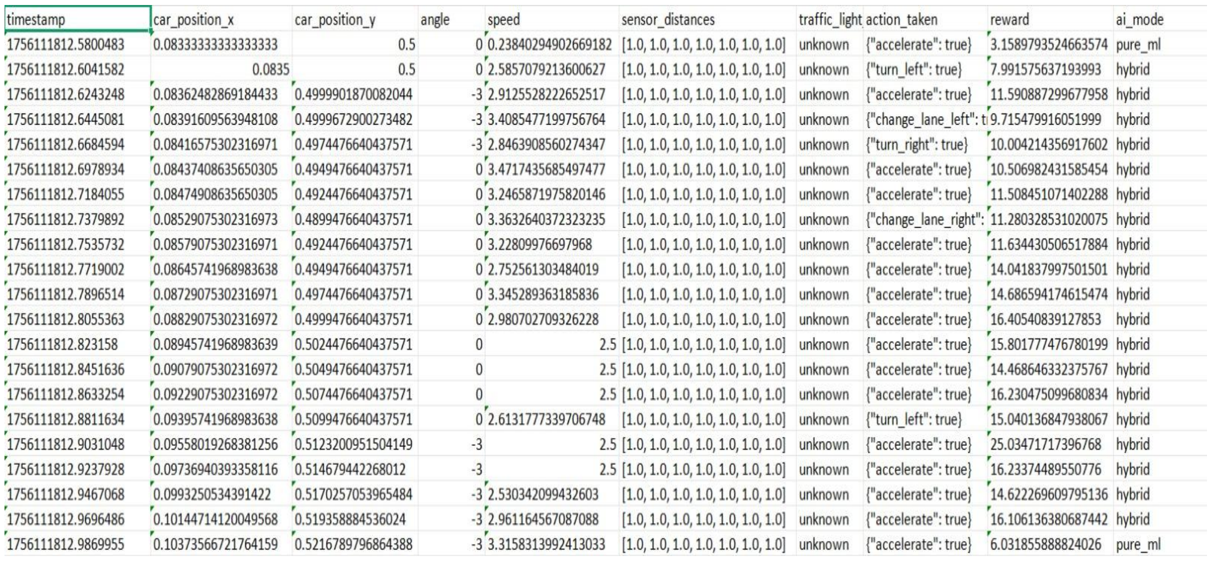
Fig 7.1.1: Project Interface

A critical aspect of the simulation is the sensor system, which relies on ray-casting to mimic distance sensors such as LIDAR or ultrasonic sensors. The vehicle is equipped with **seven ray sensors**, each pointing in a specific direction—forward, forward-left, forward-right, left, right, backward-left, and backward-right. These sensors continuously calculate the distance to the nearest object within a predefined range (typically 150 pixels). This real-time sensor data forms the primary input for both the rule-based logic and machine learning models. Sensor accuracy, refresh rate, and directional placement have been fine-tuned to strike a balance between computational efficiency and environmental awareness.

Another essential component of the simulation setup is the traffic system, which includes fully functional traffic lights with red, yellow, and green phases. The timing cycles are carefully crafted to simulate real-world behaviour. The autonomous vehicle must interpret traffic light states and adjust its speed or trajectory accordingly. This allows researchers to evaluate the model's ability to obey traffic rules—a vital function for real-world deployment.

The simulation also features a dynamic obstacle system, where obstacles may appear or move unpredictably. These obstacles test the vehicle's adaptability and stress-test the collision avoidance algorithms. Whether the vehicle is operating in Hybrid Mode or Pure ML Mode, its ability to respond appropriately to dynamic obstacles serves as a critical indicator of model performance.

Training and evaluation are conducted under multiple operational modes. In Manual Driving Mode, a human player controls the vehicle using keyboard inputs. This mode is essential for collecting training data used by supervised machine learning models. The simulation logs sensor readings, vehicle state information, and driver actions in real time. In **Autonomous Modes** (Hybrid Mode and Pure ML Mode), the vehicle navigates using the AI algorithms, allowing researchers to analyze decision quality, safety, efficiency, and learning progression.



timestamp	car_position_x	car_position_y	angle	speed	sensor_distances	traffic_light	action_taken	reward	ai_mode
1756111812.5800483	0.0833333333333333	0.5	0	0.23840294902669182	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	3.1589793524663574	pure_ml
1756111812.6041582	0.0835	0.5	0	2.5857079213600627	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"turn_left": true}	7.991575637193993	hybrid
1756111812.6243248	0.08362482869184433	0.4999901870082044	-3	2.9125528222652517	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	11.590887299677958	hybrid
1756111812.6445081	0.08391609563948108	0.4999672900273482	-3	3.4085477199756764	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"change_lane_left": t	9.715479916051999	hybrid
1756111812.6684594	0.08416575302316971	0.4974476640437571	-3	2.8463908560274347	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"turn_right": true}	10.004214356917602	hybrid
1756111812.6978934	0.08437408635650305	0.4949476640437571	0	3.4717435685497477	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	10.506982431585454	hybrid
1756111812.7184055	0.08474908635650305	0.4924476640437571	0	3.2465871975820146	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	11.508451071402288	hybrid
1756111812.7379892	0.08529075302316973	0.4899476640437571	0	3.3632640372323235	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"change_lane_right":	11.280328531020075	hybrid
1756111812.7535732	0.08579075302316971	0.4924476640437571	0	3.22809976697968	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	11.634430506517884	hybrid
1756111812.7719002	0.08645741968983638	0.4949476640437571	0	2.752561303484019	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	14.041837997501501	hybrid
1756111812.7896514	0.08729075302316971	0.4974476640437571	0	3.345289363185836	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	14.686594174615474	hybrid
1756111812.8055363	0.08829075302316972	0.4999476640437571	0	2.980702709326228	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	16.40540839127853	hybrid
1756111812.823158	0.08945741968983639	0.5024476640437571	0	2.5	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	15.801777476780199	hybrid
1756111812.8451636	0.09079075302316972	0.5049476640437571	0	2.5	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	14.468646332375767	hybrid
1756111812.8633254	0.09229075302316972	0.5074476640437571	0	2.5	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	16.230475099680834	hybrid
1756111812.8811634	0.09395741968983638	0.5099476640437571	0	2.6131777339706748	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"turn_left": true}	15.040136847938067	hybrid
1756111812.9031048	0.09558019268381256	0.5123200951504149	-3	2.5	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	25.03471717396768	hybrid
1756111812.9237928	0.09736940393358116	0.514679442268012	-3	2.5	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	16.23374489550776	hybrid
1756111812.9467068	0.0993250534391422	0.5170257053965484	-3	2.530342099432603	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	14.622269609795136	hybrid
1756111812.9696486	0.10144714120049568	0.519358884536024	-3	2.961164567087088	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	16.106136380687442	hybrid
1756111812.9869955	0.10373566721764159	0.5216789796864388	-3	3.3158313992413033	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	unknown	{"accelerate": true}	6.031855888824026	pure_ml

Fig 7.1.2: Data Logging System

To ensure reproducibility, the simulation includes built-in data logging mechanisms that record collisions, rewards (for RL), control commands, and sensor readings. These logs are used for performance analysis and further model training. Frame rate consistency, physics accuracy, and environment resets are carefully managed to maintain uniform conditions across experiments.

Overall, the simulation setup provides a comprehensive, realistic, and safe environment in which the hybrid autonomous driving system can be thoroughly evaluated and continuously improved.

6.2 Experimental Results:

The simulation results provide a comprehensive evaluation of the hybrid autonomous driving system, revealing how the different AI modules—rule-based logic, supervised machine learning, and reinforcement learning—perform individually and collectively. These results demonstrate the strengths of the hybrid architecture and highlight areas where each module contributes to overall system performance. By running multiple trials across diverse scenarios, the project analyzes collision frequency, lane discipline, reaction time, adaptability, and overall driving smoothness.

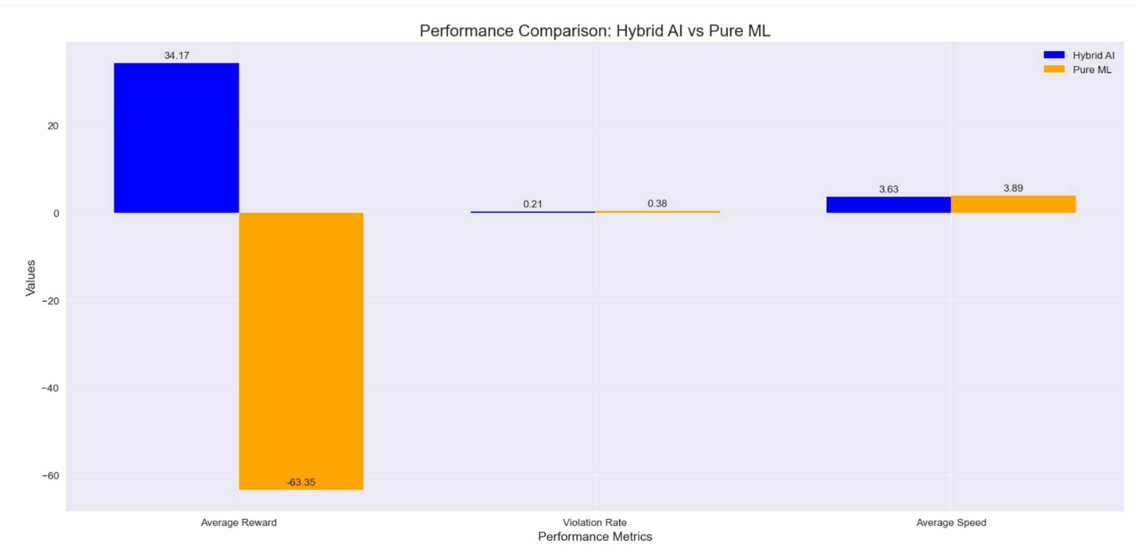
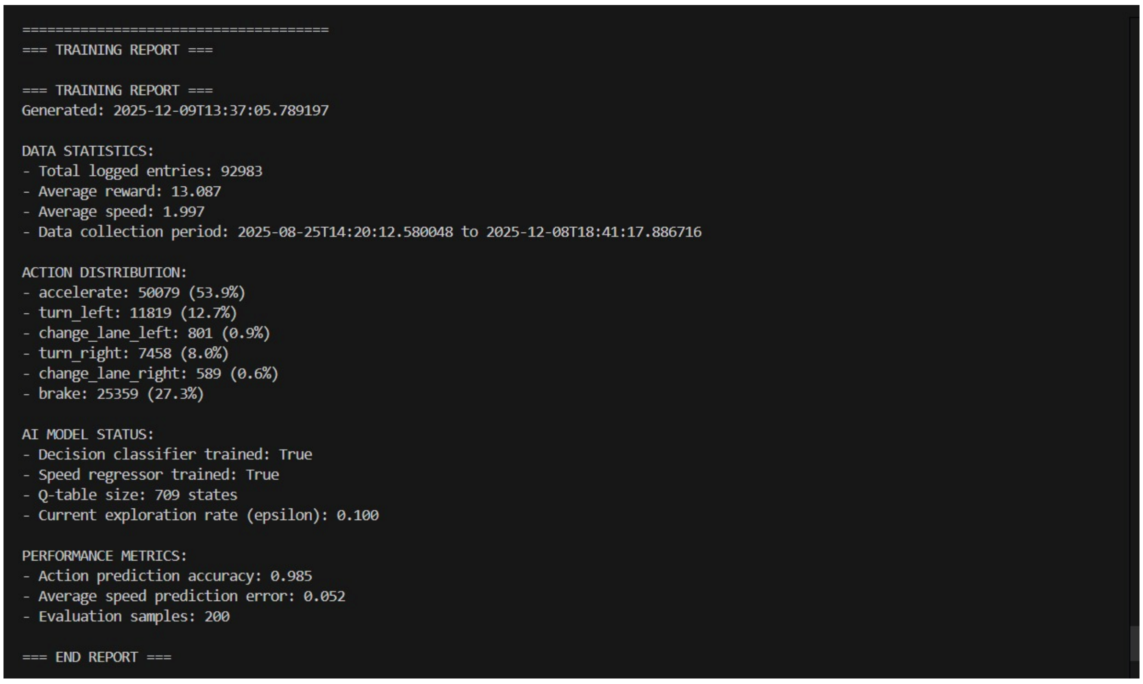


Fig 7.2.1: Model Comparison

One of the most significant findings from the simulation is that the **Hybrid Mode consistently outperforms Pure Machine Learning Mode** in terms of safety. When relying solely on ML predictions, the autonomous vehicle occasionally exhibits erratic behaviour, especially in unfamiliar or complex environments. This is primarily due to the limitations of the training dataset, which may not fully capture all possible driving situations. As a result, the Pure ML Mode experiences a higher number of collisions and lane deviations. In contrast, the Hybrid Mode, supported by the rule-based safety layer, demonstrates superior stability. Whenever the ML model attempts a risky maneuver—such as accelerating near an obstacle—the rule-based logic intervenes, preventing potential crashes. This layered decision-making approach significantly reduces collision rates and ensures strict compliance with safety constraints.



```
=====================================
=== TRAINING REPORT ===

=== TRAINING REPORT ===
Generated: 2025-12-09T13:37:05.789197

DATA STATISTICS:
- Total logged entries: 92983
- Average reward: 13.087
- Average speed: 1.997
- Data collection period: 2025-08-25T14:20:12.580048 to 2025-12-08T18:41:17.886716

ACTION DISTRIBUTION:
- accelerate: 50079 (53.9%)
- turn_left: 11819 (12.7%)
- change_lane_left: 801 (0.9%)
- turn_right: 7458 (8.0%)
- change_lane_right: 589 (0.6%)
- brake: 25359 (27.3%)

AI MODEL STATUS:
- Decision classifier trained: True
- Speed regressor trained: True
- Q-table size: 709 states
- Current exploration rate (epsilon): 0.100

PERFORMANCE METRICS:
- Action prediction accuracy: 0.985
- Average speed prediction error: 0.052
- Evaluation samples: 200

=== END REPORT ===
```

Fig 7.2.2: Report View

The simulation also reveals the impact of Reinforcement Learning on long-term adaptability. Over multiple training episodes, the RL agent gradually develops more refined driving strategies. Early episodes show inconsistent behaviour, with the agent frequently colliding with obstacles, making abrupt turns, or drifting out of the lane. However, as the training progresses, Q-values stabilize, and the agent begins to demonstrate smoother lane centering, efficient obstacle avoidance, and improved steering accuracy. The reward function plays a crucial role in shaping these behaviours by reinforcing positive driving patterns such as maintaining lane position, avoiding sudden movements, and progressing forward without hesitation. Although RL alone is not sufficient for safety-critical decisions, its contribution within the hybrid model enhances adaptability and efficiency.

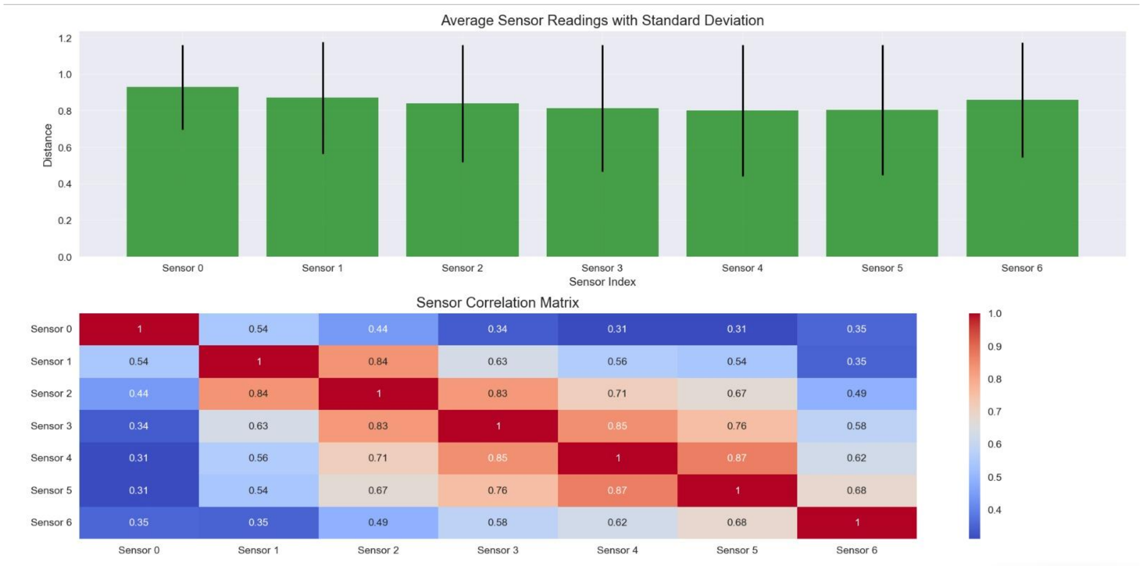


Fig 7.2.3: Sensor Data

The sensor system's performance also contributes to the overall results. The ray-casting sensors accurately detect obstacles and boundaries, enabling the rule-based and ML/RL models to make informed decisions. Sensor failures or noise are minimal, and their responsiveness allows the vehicle to react quickly to sudden environmental changes. This is particularly important when encountering dynamic obstacles, where rapid decision-making is essential.

Another important aspect of the results is the evaluation of traffic light handling. In Pure ML Mode, the vehicle occasionally misinterprets the traffic light state or attempts to accelerate during a red light. However, in Hybrid Mode, the rule-based traffic light compliance algorithm ensures consistent stopping behaviour at red lights and cautious decision-making during yellow lights. This further supports the conclusion that hybrid intelligence improves rule adherence.

The simulation results also show differences in driving smoothness. Pure ML Mode sometimes produces jittery or abrupt control commands due to uncertainties in model predictions. Conversely, Hybrid Mode smoothens these commands by combining ML predictions with deterministic constraints, resulting in more natural driving behaviour.

In summary, the simulation results demonstrate that the hybrid model delivers the best performance by combining the adaptability of ML and RL with the reliability of rule-based safety. The system achieves reduced collisions, improved lane stability, enhanced rule compliance, and overall more intelligent driving patterns compared to standalone AI approaches.

6.3. Comparative Analysis of Pure ML and Hybrid AI Models:

ASPECT	PURE ML MODEL	HYBRID MODEL
1. APPROACH	Uses only one machine learning method	Combines ML, RL, rule-based logic, control theory, etc.
2. DECISION MAKING	Fully data-dependent	Data + predefined rules + multi-algorithm fusion
3. PERFORMANCE IN COMPLEX ENVIRONMENTS	Struggles with dynamic obstacles and traffic scenarios	Handles complex, real-time situations more effectively
4. ACCURACY & RELIABILITY	Can be inconsistent with changing data	More stable and accurate due to multiple intelligence layers
5. DATA REQUIREMENT	Requires large, high-quality datasets	Works even with moderate data due to rule support

7. Conclusion & Future Work:

The development of the Car Driving AI Simulation: Hybrid Intelligence System for Autonomous Vehicles demonstrates the strong potential of combining rule-based logic, supervised machine learning, and reinforcement learning into a unified decision-making framework. The hybrid architecture consistently delivered safer, more stable, and more reliable driving behavior when compared to purely data-driven models. The rule-based safety layer ensured consistent compliance with essential driving rules, while the ML and RL components contributed adaptability and continuous learning, enabling the vehicle to handle dynamic and unpredictable scenarios. Simulation results confirmed that Hybrid Mode significantly outperformed Pure ML Mode in safety, rule adherence, and trajectory control. Overall, this project highlights the importance of hybrid intelligence as a practical and scalable approach for autonomous driving, balancing interpretability, learning capability, and robust safety mechanisms.

Future enhancements include moving to a 3D simulation with realistic physics and richer sensors like LIDAR and cameras. More advanced ML and RL models could improve decision-making, while multi-agent scenarios would allow testing interactions between vehicles. Robustness can be increased through domain randomization and formal safety checks. Finally, deploying the system on a small physical robot or RC car would help validate performance in real-world conditions.

9. References:

[1] Bishop, C. M. Pattern Recognition and Machine Learning. Springer, 2006.
A foundational text explaining supervised learning, classification, regression, and probabilistic modeling in machine learning.

[2] Hastie, T., Tibshirani, R., & Friedman, J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2009.
A comprehensive work on statistical learning methods, widely used for understanding predictive modeling and structured datasets.

[3] Russell, S., & Norvig, P. Artificial Intelligence: A Modern Approach. 4th ed., Pearson, 2021.
A leading reference on AI theory, covering rule-based reasoning, hybrid architectures, and safety-critical decision-making in intelligent systems.

[4] Nilsson, N. J. The Quest for Artificial Intelligence: A History of Ideas and Achievements. Cambridge University Press, 2010.
Provides insights into the evolution of hybrid and symbolic-learning AI frameworks, highlighting the benefits of combining logic with learning.

[5] Poole, D., Mackworth, A., & Goebel, R. Computational Intelligence: A Logical Approach. Oxford University Press, 1998.
Discusses hybrid AI, rule-based reasoning, and their application in safety-critical and uncertain environments.

[6] Sutton, R. S., & Barto, A. G. Reinforcement Learning: An Introduction. 2nd ed., MIT Press, 2018.
Standard reference on reinforcement learning, detailing reward design challenges, instability issues, and sequential decision-making.

[7] Szepesvári, C. Algorithms for Reinforcement Learning. Morgan & Claypool Publishers, 2010.
Covers RL algorithmic challenges such as reward sparsity, unstable learning curves, and environment sensitivity.

[8] François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. An Introduction to Deep Reinforcement Learning. Morgan & Claypool Publishers, 2018.
Explores common pitfalls in RL experiments, including inconsistent logging, reward misalignment, and training instability.

[9] Thrun, S., Burgard, W., & Fox, D. Probabilistic Robotics. MIT Press, 2005.
Explains sensor behavior, correlations between sensor readings, and probabilistic models for navigation and perception.

[10] Siciliano, B., & Khatib, O. (Eds.). Springer Handbook of Robotics. Springer, 2016.
Covers sensor systems, spatial perception, and the structure of robotic sensor arrays, including correlation analysis.

[11] Henderson, P., et al. Deep Reinforcement Learning That Matters. Morgan & Claypool Publishers, 2019.
Provides strong evidence on how poor logging, missing rewards, and improper evaluation distort RL results.

[12] Machado, M. C., et al. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. Morgan & Claypool Publishers, 2017.
Discusses common logging and evaluation issues in RL environments, including frame-level inconsistencies and data misinterpretation