MASTER THESIS

Toward a Multi Agent Approach for LLM-Based Dynamic Vehicle Control and Communication in Accidental Condition

RAFI MD ASHIFUJJMAN

STUDENT ID:
71330708

SUPERVISOR:
NAOKI FUKUTA

Department of Informatics
Graduate School of Integrated Science and Technology
Shizuoka University

Toward a Multi Agent Approach for LLM-Based Dynamic Vehicle Control and Communication in Accidental Condition

Author:
RAFI MD ASHIFUJJMAN

Supervisor:
Naoki FUKUTA

MASTER OF INFORMATICS

Department of Informatics
Graduate School of Integrated Science and Technology
Computer Science

Abstract

In autonomous vehicles (AVs), making the right decision in unknown traffic situations remains a critical challenge. Traditional AVs, which are based on pre-programmed rules or pre-trained models, often struggle in scenarios where human-like reasoning is required. This research examines how Large Language Models (LLMs) can act as a main decision agent for AVs, responding to text-based real-world traffic situations simulation.

We propose a multi-agent framework consisting of two key components, a Decision Agent powered by LLMs, and a Communication Agent that enables Vehicle-to-Vehicle (V2V) information exchange. The Communication Agent shares structured, real-time data such as obstacle presence, speed, and intended actions, which is then incorporated into the Decision Agent's reasoning process.

To evaluate the proposed system, we conducted a series of text-based simulations that simulate real-world traffic conditions. These scenarios are used to test whether LLMs, when provided with scenario-specific prompts and contextual V2V data, can generate consistent and human-like decisions. Our results indicate that while LLMs show strong reasoning abilities, their precision and reliability of decision improve significantly when augmented with contextual information through V2V communication. This research presents a preliminary approach to bridge the gap between current AV capabilities and the demands of real-world accidental conditions.

Contents

List of Figures

List of Abbreviations

AI	Artificial Intelligence
AV	Autonomous Vehicle
ADS	Autonomous Driving System
LLM	Large Language Model
MAS	Multi-Agent System
NLP	Natural Language Processing
V2V	Vehicle-to-Vehicle

Chapter 1

Introduction

The advancement of autonomous vehicle technology brings us closer to the long-held vision of fully automated transportation, where vehicles operate without human intervention. However, achieving this goal of complete automation remains a significant challenge, particularly in ensuring safe and reliable decision-making in complex and uncertain environments.

1.1 Research Background

Our research investigates the potential of integrating Large Language Models (LLMs) [1] into a multi-agent model [2] to improve decision-making in autonomous vehicles (AVs), particularly in the unknown-unsafe driving domain. Our main concern is to explore the use of an LLM as the primary decision-making agent within an autonomous driving system (ADS) [3], evaluating its ability to reason [4] and handle uncertain traffic situations. We aim to contribute to the development of safer and more reliable Level 5 (complete automation) autonomous vehicles, as defined by the Society of Automotive Engineers (SAE International) [5], [6]. Although traditional AVs can handle predictable 'known-safe' and 'known-unsafe' situations, they struggle with highly unpredictable events, such as a pedestrian unexpectedly crossing the road while the light is red for oncoming traffic representing an unknown unsafe condition. These scenarios, which are difficult to cover in training datasets, represent the primary barrier to achieving complete automation. Human drivers use common sense [7] and past experiences to navigate these unexpected events. However, current AV systems, even those that use advanced reinforcement learning (RL) [8], struggle in these scenarios. This limitation is especially evident where the correct action is not immediately clear [9]. Addressing these unknown-unsafe scenarios is critical for improving an AV's ability to handle real-world driving environments and ensuring the development of truly reliable autonomous systems.

1.2   Research Objectives

Determining how autonomous vehicles (AVs) should make decisions in complex or fully unknown traffic situations remains a major challenge in the development of fully autonomous systems. This problem has been widely explored in the fields of artificial intelligence, robotics, and autonomous system design [10], [11]. Traditional autonomous vehicle systems used these approaches to take driving decisions,

- Rule-based system, which operates on predefined logic and hard-coded behaviors such as 'if-then' conditions. These systems cannot take decisions in unfamiliar or edge-case situations that fall outside of their rule sets.

- Supervised learning or machine learning-based models, which are heavily dependent on training data. They often fail in new contexts since they cannot reason beyond the patterns they have learned.

The ability to make safe and contextually appropriate decisions in real-time becomes especially critical when AVs encounter unknown-unsafe scenarios traffic conditions not represented in training data or handled poorly by traditional algorithms.

And this could be a solution to simulate all the situation of this Unknown Unsafe domain and make a dataset to train a model that will work as a decision agent for the AVs. However, a fundamental challenge is to prepare for truly unknown situations. Any scenario researchers create is technically **known** even if unfamiliar to AVs, making it difficult to test how vehicles handle genuinely unknown-unsafe situations. In addition, recreating complex real-world scenarios in simulation environments is a challenge. And accurately simulating unpredictable human actions or complicated environmental conditions remains nearly impossible. In real life, it is nearly impossible to include all the uncertain situations in training datasets for traditional autonomous driving systems. Human drivers rely on common sense to navigate these situations effectively. To address this limitation, we are going to investigate whether LLMs can provide human-like reasoning [12] that traditional systems lack, such as how human drivers handle unexpected situations. The main objective of this research is to evaluate how effectively LLMs can act as decision agents in unfamiliar traffic conditions, and potentially bridging the gap between programmed responses and adaptive decision making in autonomous vehicle systems.
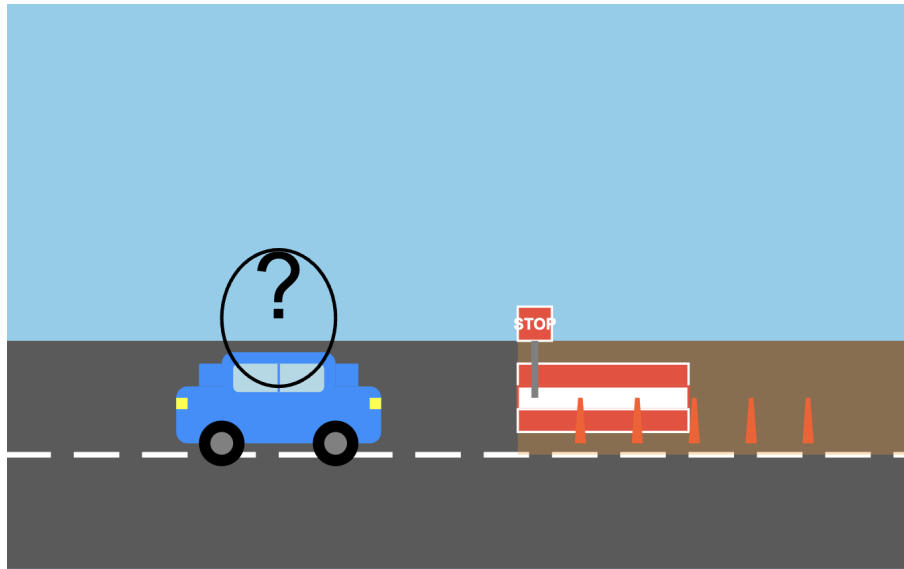
FIGURE 1.1: What a current AV might do in this situation

1.3 Research Motivation

1.3.1 Motivating Example 1 : What a current AV might do in Unknown situation

Imagine a traffic situation where a road is partially blocked by construction cones, with a 'STOP' sign placed near the cones shown in Figure 1.1. A traditional autonomous vehicle might see the 'STOP' sign and interpret it as a complete road closure, coming to a complete stop since that is what its preprogrammed data tell it to do.

However, a human driver would use common sense to recognize that the road is only partially closed, slow down, and safely navigate through the open section. This scenario highlights the need for a reasoning system that can interpret context beyond literal rules. Although the situation described in Figure 1.1 can now be handled with some additional rules, it highlights a key limitation of current AVs. They struggle to reason in unfamiliar or uncertain situations that are not covered by pre-programmed logic or training data.

1.3.2 Motivating Example 2 : Lack of communication between vehicles

Now imagine a more complicated situation where an autonomous truck comes to a sudden stop when it sees the 'STOP' sign. There is another car, driven by a human or another AV, right behind it. This sudden stop could easily cause a crash, as shown in Figure 1.2. An autonomous truck (AV1) suddenly brakes
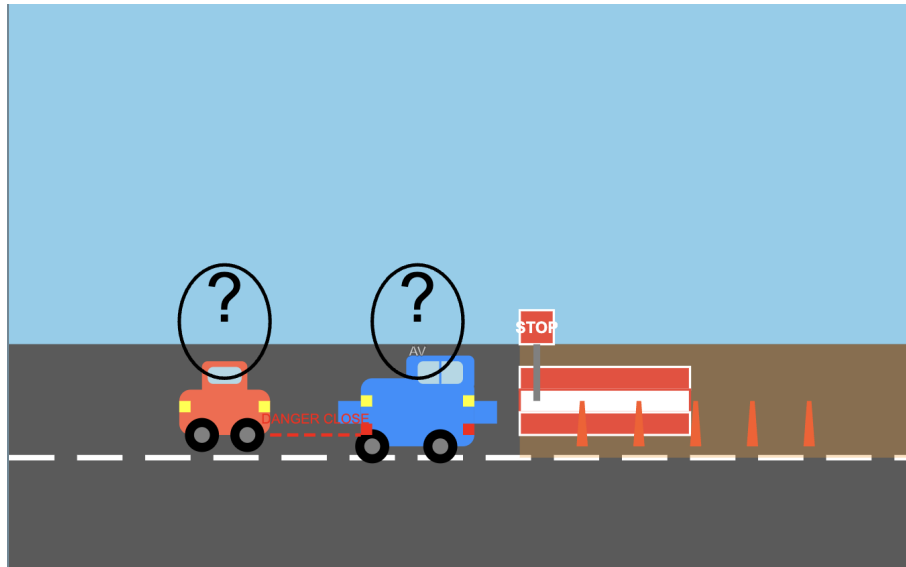
FIGURE 1.2: Lack of communication between vehicles may lead to an accident

upon detecting a partially visible 'STOP' sign, an action based on its own perception. Behind it, another vehicle (AV2), which could be human-driven or autonomous, is following closely. Due to AV1's size, AV2 has no perception of the sign and no prior indication that AV1 is about to stop. Without any communication between vehicles, AV2 is unaware of the hazard ahead, greatly increasing the risk of a rear-end collision. This scenario illustrates a limitation of perception in autonomous systems. The issue is not only whether each vehicle can make the right decision for itself but also whether it can contribute to a collectively safe environment. In the near future autonomous vehicles will begin to share the road with human drivers, and this limitation will become even more common. since human drivers cannot anticipate AV decisions and AVs cannot rely on human predictability, this mix mode can lead to rear-end collision more often. In our research, we will investigate how LLMs utilize this shared V2V information to make driving decisions. Specifically, our goal is to assess whether the inclusion of contextual data, such as obstacle positions, speed, and vehicle intentions, enables the LLM to produce safer and more accurate decisions in complex traffic scenarios.

1.4 Research Questions

One of the primary objectives of this investigation is to answer two main research questions.

- **Research Question 1**: Can an agent make decisions with the help of an

LLM without undergoing a specific learning process for each new situation?

- **Research Question 2**: Is a single centralized decision-making agent sufficient to process all incoming Vehicle-to-Vehicle (V2V) data, or is a multi-agent system required for improved efficiency and scalability?

- **Sub-Research Question**: Does repeated querying of LLMs on the same scenario lead to variations in decision-making outcomes?

1.5   Research Contribution

This research introduces a novel multi-agent architecture that integrates LLMs with Vehicle-to-Vehicle (V2V) communication [13] to enhance autonomous vehicle decision-making in unknown-unsafe driving scenarios. This work demonstrates how their combined use enables emerging reasoning capabilities and allows autonomous vehicles to make more accurate and context-aware decisions in safety-critical scenarios. This research presents a standardized text-based simulation framework that replicates real traffic situations, used to test the capabilities of LLMs in logical reasoning, rule compliance, and dynamic decision-making for autonomous vehicles.

Additionally, a multi-agent communication system is implemented, enabling the vehicle to share structured information such as obstacle locations, speed, and future intentions in a certain radius. This data serves as a contextual input for the LLM-based Decision Agent. Our work demonstrates how contextual information sharing can improve the decision-making process in potentially uncertain scenarios to safe outcomes. Through empirical experimentation, this research provides definitive evidence that structured V2V communication data significantly improves the accuracy of LLM-based decision making. Our research develops approaches for quantifying the consistency and reliability of decisions in LLM-based systems, addressing one of the most critical concerns for the deployment of AI in safety-critical applications. By addressing these research areas, our work contributes to the advancement of both theoretical understanding and practical implementation of AI-enhanced autonomous driving systems, with particular emphasis on navigation through safety-critical scenarios.

1.6   Thesis Outline

This thesis is organized into five different chapters and two appendix, as follows:

Chapter 1 *Introduction*: This chapter describes a brief introduction to the thesis, including the research background, objectives, questions, contributions, and key motivations.

Chapter 2 *Background and Related Work*: This chapter introduces the research topics of autonomous vehicle decision-making, unknown-unsafe scenarios, LLMs, and Multi-Agent Systems, defining the basic concepts and discussing related work in the field.

Chapter 3 *Approach and Example-based Analysis*: This chapter describes the challenges and solving approach of this research and introduces the detailed multi-agent model and example scenarios used for evaluation.

Chapter 4 *Simulation and Results*: This chapter presents the simulation setup and implementation process used to evaluate the performance of LLMs in autonomous vehicle decision-making. It describes the design of text-based scenarios, the multi-agent architecture including the Decision and Communication Agents, and how vehicle-to-vehicle (V2V) data was integrated. The chapter also analyzes the results across various models, highlights improvements by V2V communication, and discusses limitations such as consistency, unpredictability, and model selection challenges.

Chapter 5 *Conclusion*: This chapter describes the summary of the work, discusses the key findings, and outlines directions for future research as a conclusion.

AppendixA: This section provides the simulation code used to create and test the experimental scenarios described in this thesis.

AppendixB: This section will outline the setup, prerequisites, and execution instructions for each simulation code.

Chapter 2

Background and Related Work

This chapter introduces the fundamental concepts and technologies that form the basis for this research. This chapter examines the current state of autonomous vehicle decision-making systems, identifies the critical challenges they face, and explores emerging technologies that can potentially address these limitations.

2.1 Autonomous Vehicle Decision-Making

The existing definitions of autonomous driving specified by the SAE [6] for Levels 1 to 5 can be interpreted as follows. Following are the summarizations of definitions in [6]. At level 1 vehicles, driving assistance systems can sometimes help the driver complete some lateral or longitudinal driving tasks. At level 2 vehicles can automatically provide multidimensional assistance. At level 3, vehicles can perform automatic acceleration and deceleration steering in a specific environment without the driver's intervention. At level 4, if a vehicle is currently in an autonomous driving state under limited conditions, the driver is not required to continuously control the steering wheel. At Level 5, vehicles can run automatically under any conditions and scenarios. At this level, the automatic system of a vehicle completely replaces the human driver and achieves complete automation. A level 5 AV is capable of sensing its environment and operating without human involvement.

We assume that the decision-making process in autonomous vehicles follows a fundamental three-stage pipeline: perception, planning, and control, as shown in the basic model of an agent [14].

- **Perception (See)**: The vehicle uses a variety of sensors and cameras, such as LiDAR (Light Detection and Ranging), and radar systems to construct a comprehensive three-dimensional representation of its environment. This stage involves detecting and classifying objects such as other vehicles, pedestrians, traffic signs, and lane markings.

- **Planning and Decision-Making (Think)**: This represents the core focus of our research. Based on the perceived environmental information, the vehicle's computational system must determine the appropriate course of

action. This involves planning and taking decisions such as lane changes, speed adjustments, and navigation through complex traffic scenarios. Traditional rule-based approaches perform well in structured and predictable traffic environments. They can only execute actions that have been explicitly predefined and encoded. When unexpected or novel situations arise that fall outside the scope of their programmed rules, these systems often do not respond appropriately [15]. Our research focuses on improving the intelligence of this decision-making component through advanced reasoning capabilities.

- **Control (Act)**: Once a decision is made, the system translates the planned actions into control signals that activate the vehicle's physical components, including steering, acceleration, and braking systems. The decision-making module is particularly critical, as it is responsible for planning and performing safely.

2.2   The Challenge of Unknown-Unsafe Scenarios

The driving domain can be divided into three primary categories: known-safe, known-unsafe, and unknown-unsafe. Although traditional AV systems can make proper decisions in the first two categories, which can be addressed with pre-programmed rules, their primary limitation lies in handling the complex and unpredictable nature of unknown-unsafe scenarios [16] shown in Figure 2.1.
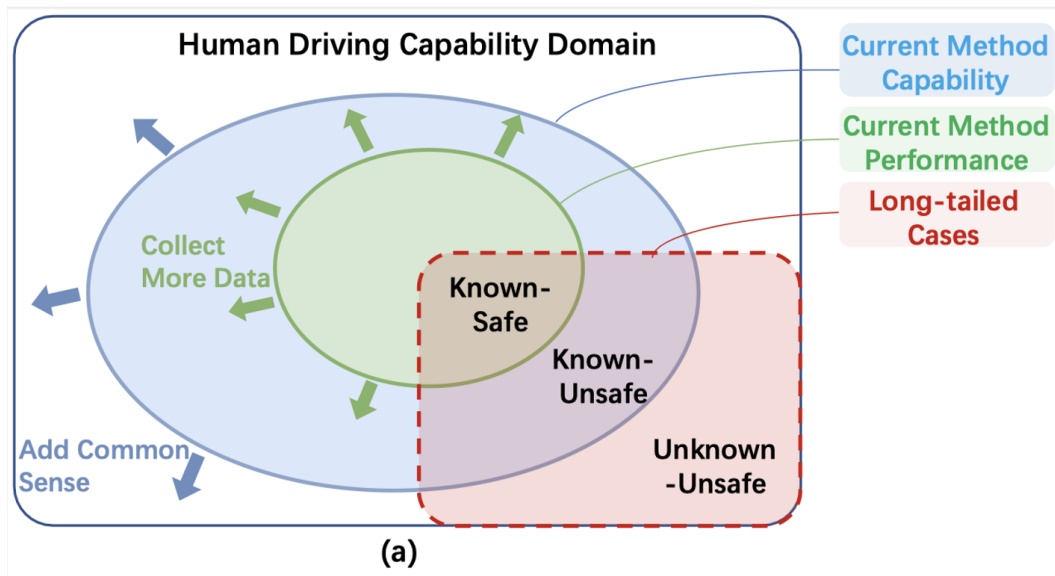
FIGURE 2.1: Human driving domain and existing autonomous driving systems

- Known Safe Scenarios: In [16], these are defined as routine traffic conditions where the environment is predictable and the appropriate response is well-established. For example, driving straight on an empty highway or stopping at a red light. AVs can reliably navigate these situations using pre-defined logic.

- Known Unsafe Scenarios: In [16], these are defined as situations where a clear and present danger is identifiable based on established rules. Examples include an object stationary on the road, a speeding in a school zone, or another vehicle running a red light. AVs can be trained to recognize and react to these predictable hazards.

- Unknown-Unsafe Scenarios: In [16], these are defined as unexpected and unplanned situations that are not explicitly covered by pre-programmed rules and can occur without warning or precedent. Examples include a ball rolling onto the road, a pedestrian unexpectedly crossing a red light, or ambiguous hand signals from another driver. These situations require quick context-aware judgment and are difficult to cover exhaustively in training datasets.

As mentioned in [16] traditional AVs are generally capable of handling known-safe and known-unsafe scenarios effectively using pre-programmed rules, supervised learning, or reinforcement learning models. For example, they can detect and respond to traffic lights, lane markings, and other common characteristics of the driving environment. However, they will struggle with new or unusual situations that are not directly covered by their pre-programmed behaviors. Human drivers, on the other hand, use common sense and past experiences to handle unexpected events, such as recognizing that traffic cones on a moving truck are not a stationary obstacle. Current AV systems, even those that use advanced reinforcement learning (RL), will struggle to make such a nuanced distinction. This limitation poses a serious safety risk in real-world driving and can lead to several critical failure modes, such as AVs can incorrectly identify a safe situation as dangerous, leading to unnecessary and potentially disruptive actions to sudden braking.

For example, an AV might stop for a tree shadow across the road, treating it as a solid obstacle. The AVs can fail to recognize an actual danger since it does not fit to a pre-programmed pattern. This is the most critical failure mode, as it can lead directly to collisions. Addressing this gap is essential for the development of fully autonomous systems that can be trusted to operate safely and reliably in the complexity of real-world environments.

2.3 Large Language Models for Reasoning

An important development in artificial intelligence is represented by LLMs [1]. LLMs are a type of Machine Learning (ML) model trained in vast amounts of

text and code from the Internet, allowing them to comprehend and produce language similar to that of a human. LLMs represent a significant leap in natural language processing and artificial intelligence [1]. Unlike traditional rule-based systems, these machine learning models, driven by deep learning techniques, excel at comprehending and generating extensive human-like texts, as mentioned in [1].

We believe that, having a reasoning capability is a fundamental aspect of human intelligence, essential for problem solving, decision-making, and critical thinking. In recent years, LLMs have demonstrated emerging abilities [17], such as in-context learning [18], role play [19], and analogical reasoning [20]. These abilities allow LLMs to go beyond natural language processing problems to facilitate a wider range of tasks, such as code generation [21], robotic control [22], and autonomous agents [23]. Among these abilities, human-like reasoning has been receiving a significant attention from both academia and industry, since it demonstrates great potential for LLMs to generalize complex real-world problems through abstract and logical reasoning. A notable breakthrough in this area is the 'chain of thought' prompting technique [24], which can obtain step-by-step human-like reasoning processes at the test time without additional training.

This research investigates the potential of LLMs to serve as the main decision makers within AV systems, assessing whether they can reason and react more like humans in complex scenarios.

2.4 Multi-Agent Systems and V2V Communication

A Multi-Agent System (MAS) is a computerized system composed of multiple interacting intelligent agents [25]. In the context of autonomous driving, MAS is particularly relevant as it allows the overall system to function more efficiently by dividing the responsibilities among individual agents [26]. Each agent operates individually with a limited view of the environment, and through coordination and communication, they work together to achieve a common goal [27]. This research [26] presents an autonomous vehicle system based on multi-agents to reduce the complexity of the autonomous system by splitting tasks between different agents, which in turn reduces execution time and allows quicker intervention in complex scenarios.

In the context of autonomous driving, this paradigm is highly relevant. We believe that, **Vehicle-to-Vehicle (V2V) communication** is a key enabling technology for a multi-agent approach to driving. V2V allows vehicles to wirelessly exchange information about their position, speed, direction, and intended actions. We believe that this will create a collective awareness that extends far beyond the perception range of any single vehicle's onboard sensors (e.g., LiDAR, cameras).

By structuring a V2V system within a multi-agent model, we can address some of the fundamental limitations of single-agent AVs. This research proposes an approach to integrate LLM-based decision-making within a multi-agent model, where V2V communication provides the crucial context needed for their robust reasoning in accidental conditions.

2.5   Related Work

2.5.1   Traditional Approaches to Decision-Making in AVs

Research in autonomous vehicle decision making has evolved through several paradigms. Traditional autonomous vehicles are based on data-driven approaches, which are categorized into modular and end-to-end frameworks [28]. Modular-based systems break the entire autonomous driving process into separate components, such as the perception module, the prediction module, and the planning module. Including tasks such as object detection [29] and object occupancy prediction. Early systems relied heavily on rule-based and finite-state machines [30], which are effective for predictable structured environments, but lack the flexibility to handle novel situations [31].

2.5.2   Multi-Agent Systems In AVs

In autonomous driving, incorporation of the concept of multi-agent should be very useful [32]. Instead of providing one system doing everything, different agents can handle different tasks, such as making decisions, detecting obstacles, or sharing vehicle information. Research in this area has often focused on optimizing traffic flow, coordinating intersections, and cooperative path planning [33]. However, these works typically used simpler, more formally defined agent behaviors. This research [34] presents experiments on learning decision-making policies in multi-agent environments for autonomous systems like connected autonomous vehicles. The agents in [34] were able to learn to navigate their environment and avoid collisions even in a partially observable environment with obstacles and other moving agents.

However, we believe that, learning decision-making policies is still challenging due to the non-stationary nature of the environment. This research [35] explores a multi-agent system (MAS) architecture designed to facilitate cooperative control of CAVs. This hierarchical architecture presented in [35] enables vehicles to collaborate effectively in complex traffic environments, sharing information and making collective decisions that enhance safety and efficiency. This study emphasizes the importance of cooperation among autonomous vehicles, particularly in scenarios where rapid decision-making and coordination are critical to preventing accidents and ensuring smooth traffic flow.

2.5.3 LLMs in Autonomous Driving

Human drivers instinctively reason with common sense knowledge to predict hazards in unfamiliar scenarios and to understand the intentions of other road users. However, this essential capability is completely missing from traditional decision-making systems in autonomous driving. Recent advances have introduced LLMs as potential decision makers in AV systems.

In [36], Fu et al. established the concept of 'Drive like a human', demonstrating that LLMs can generate plausible driving behaviors in response to textual descriptions of traffic scenes. Their work primarily focused on a single agent perspective and highlighted the promise of LLMs for high-level reasoning.

In [37], they introduced a method for training systems to better understand driving situations. They connected numerical driving data with GPT-3.5, a powerful LLM, which helped the system answer questions and make better driving decisions.

The 'AccidentGPT' [38], introduced a model that can analyze traffic accidents. It uses different types of data (such as text and images) to recreate accident scenes and generate detailed reports. This helps generate detailed reports on what may have caused the accident. It shows that LLMs can also be useful after an accident, not just while driving.

This research [39], reviews the integration of LLMs into autonomous driving systems, highlighting their potential to improve decision-making, perception, and interaction through advanced reasoning and contextual understanding. The survey categorizes their research into planning, perception, question answering, and addressing the challenges of transparency, scalability of real-world application.

The 'DriveMLM' [40], created a new self-driving system that connects LLMs with vehicle planning tasks. This allowed the car to make decisions on a simulator and control the vehicle more effectively by using a standard planning module.

The 'Drive As You Speak' [41], took things a step further by allowing drivers to talk to the car using natural language. This made it possible for LLMs to understand and follow voice commands, improving the user experience and making driving more intuitive.

In [42], they explores the ability of integrating LLMs into autonomous driving (AD) structures to emulate human-like behavior. LLMs can use their memory to apply past experiences to future decision-making, improving adaptability

and decision-making in AD systems. It can improve reliability and safety by enabling human-like reasoning and adaptability.

The 'DriveLLM' [43], a decision-making framework that integrates LLMs with existing autonomous driving stacks. This integration allows for the use of common sense in decision-making. DriveLLM also features a unique cyber-physical feedback system, allowing it to learn and improve from its mistakes.

In [44], they explored how the combination of LLMs with visual models, called Vision Foundation Models (VFMs), can make autonomous driving systems smarter and more capable. Their work explains how technology evolved from basic sensors in early systems to more advanced deep learning methods that improve the way self-driving cars see, plan, and make decisions.

The intersection of LLMs and multi-agent communication presents a promising new direction for autonomous driving research. Existing work has shown that LLMs can replicate human-like reasoning and MAS frameworks improve coordination. In real-world case studies, the proposed model outperforms traditional decision-making methods in complex scenarios, including difficult edge cases. However, using massive amounts of human-generated data to train these models introduces potential bias issues, as the models can unintentionally reinforce the social biases found in the data [45]. However, this research [46] shows a potential approach of using LLM to evaluate even fairness. However, our research uniquely combines these strengths, and the use of LLMs as decision agents within a cooperative V2V-enabled architecture represents a novel integration.

Chapter 3

A Multi-Agent Approach for LLM-Based Decision-Making

This chapter begins by introducing the overall approach and then provides a detailed explanation of our methodology, including how the experiments were set up, what assumptions were made, and the key factors considered when selecting models for evaluation. This is followed by illustrative example scenarios that demonstrate the system's behavior in representative traffic situations.

3.1 Conceptual Model: A Multi Agent Approach

To contribute to the advancement of full automation in AVs, this research follows a conceptual model based on a multi-agent architecture. The model is designed to emulate human-like reasoning while enabling cooperative behavior among vehicles through structured communication. The model focused on two primary agents: the Decision Agent and the Communication Agent, which work collaboratively to enhance decision-making capabilities in dynamic and uncertain traffic scenarios. The interaction between the Decision Agent and the Communication Agent is the core of this model. The Decision Agent uses both its internal scenario understanding and externally shared V2V data to make informed decisions. This hybrid reasoning system provides the AV with both self-perception and extended perception capabilities with shared information.

- We investigated whether LLMs as a decision agent can replicate human-like decision-making processes in autonomous vehicle contexts.

- We also investigated how the information sharing facilitated by the communication agent impacts and potentially improves the accuracy og the decision agent in the autonomous driving system.

During this investigation with various LLMs as main decision agent, we proceed under two assumptions.

- 1) Certain AI systems can transform real-world situations into text-based explanations, which are later used as input for the LLMs

- 2) the outputs generated by the LLMs can be translated into actual decisions made by the decision agent by interpreting the responses from the LLMs.

We have investigated the effectiveness of various LLMs for use as decision-making agents in autonomous driving systems. Our initial objective was to answer the research question (Research Question 1): Can an agent make decisions with the help of an LLMs without undergoing a specific learning process for each new situation? We have also assumed that vehicle-to-vehicle (V2V) communication has already been achieved through a standardized networking protocol, such as LAN-based direct communication or a low-latency wireless system. Under this assumption, each vehicle can communicate with other vehicles within a defined radius (e.g., 10 meters). When a vehicle enters the information sharing radius of another vehicle, it can exchange data regarding their current status, such as obstacle detection, traffic conditions, direction, and future intended actions. To evaluate the impact of V2V communication on autonomous decision-making, we have investigated how LLMs process and utilize this information. Specifically, we have examined whether the incorporation of V2V exchanged data improves decision accuracy and enables AVs to make more precise and context-aware decisions. However, V2V communication introduces a critical question (Research Question 2): Is a single centralized decision-making agent sufficient to process all incoming V2V data and make autonomous driving decisions, or is a multi-agent system required for improved efficiency and scalability? In this research, we have investigated the answers to these two research questions. Through this multi-agent approach, the research aims to bridge the gap between current autonomous vehicle capabilities and the complex decision-making requirements of fully automated transportation systems.

To test LLMs as decision agent, we have prepared a set of simple, text-based simulation scenarios of different traffic situations using Python. These scenarios are designed to cover situations that require logical reasoning or the application of common sense knowledge, such as recognizing a traffic red light and stopping accordingly. These scenarios will be used as input for various LLMs to assess how effectively they handle both types of challenges. The experiments will be conducted using the LLM selected in Figure 3.3.

3.2 The Decision Agent

The Decision Agent is responsible for making context-aware driving decisions. It utilizes the reasoning capabilities of pre-trained LLMs, such as GPT-4, Claude, and open-source alternatives. These LLMs are capable of processing natural language prompts that describe traffic scenarios and generate corresponding driving actions (e.g. STOP or FORWARD).

Hugging Face[1] is a platform where various open LLMs are available. LM Studio[2] is a platform to test and integrate LLMs available on Hugging Face into a locally available system on ordinary computing devices, even without powerful GPUs. Now, **the question is what factors were considered in the selection of LLMs for this study?**

One key factor in our model selection was quantization, which optimizes model performance while reducing computational requirements. This research [47] indicates that 8-bit quantization enables the majority of LLMs to maintain a performance level comparable to their nonquantized equivalents, regardless of model size (e.g. 7B to 70B parameters). Moreover, LLMs that are quantized to 4 bits can also up hold similar performance to their non-quantized versions across most benchmarks [47]. This approach achieves memory reduction of 50 to 75% while preserving precision in complex tasks such as reasoning, decision making, and domain-specific applications [48].

The models chosen for this investigation were selected using a structured approach based on three key factors, **popularity and performance in text-to-text generation**, **LM Studio recommendations for optimized accuracy**, and **comparative evaluations of the top commercial services**. More specifically, we follow these criteria for the choice of LLMs to be tested.

- 1) Some models were chosen based on the highest number of downloads from Hugging Face, ensuring widespread adoption and benchmark effectiveness.

- 2) Models suggested by LM Studio were included due to their strong performance and compatibility with local inference environments.

- 3) The best commercial services were selected based on their comparative performance with open-source counterparts, prioritizing accuracy, interpret ability, and real-time inference.

The selected models are shown in Figure 3.1. However, not all the Hugging Face models are fully compatible with the LM Studio runtime. As a result, some models were tested directly on the Hugging Face interface to avoid compatibility issues. Furthermore, while models such as OpenAI o1, DeepSeek R1, and Llama 3.1 405B have demonstrated strong benchmark performance, they were not included in this study due to limited quantization support, lack of local deployment feasibility, and limited open source availability.

3.3 The Communication Agent

The **Communication Agent** facilitates structured Vehicle-to-Vehicle (V2V) information exchange, allowing AVs within a defined communication radius to

---

[1]https://huggingface.co/
[2]https://lmstudio.ai/

Recommendation	Model Name	Benchmark Performance
LM Studio Recommended Models	Mistral 7B	74.6% (MMLU), 83.5% (GSM8K)
	OpenHermes 2.5	72.1% (MMLU), 79.2% (BIG-bench)
	DeepSeek 7B	76.3% (MMLU), 85.4% (GSM8K)
Hugging Face Open-Source Models	LLaMA 2 13B	75.8% (MMLU), 80.7% (BIG-bench)
	Mixtral 8x7B	78.2% (MMLU), 88.1% (GSM8K)
Top Commercial Services	Claude 3 Sonnet	82.3% (MMLU), 90.2% (BIG-bench)
	GPT-4 Turbo	85.6% (MMLU), 92.4% (GSM8K)
	Falcon 7B/40B	73.4% (MMLU), 78.9% (BIG-bench)

FIGURE 3.1: Selected Models for the Experimentation

share and receive critical environmental data. This includes information such as obstacle presence, road conditions, current speed, and the intended action of the vehicle. To ensure compatibility with LLMs, the data is formatted in a simple, human-readable structure that can be seamlessly incorporated into text-based prompts which followed the standards in the V2V communication [49]. Table 3.1 defines the standardized message schema used by the Communication Agent.

The Communication Agent ensures that all AVs within the information-sharing radius can access and benefit from collective situational awareness. For example, if a vehicle cannot directly see a stop sign due to traffic or any reason to sight block, it can still receive a warning from another vehicle that has a clear line of sight. This enables more accurate and safer decisions, particularly in complex or uncertain scenarios.

The structured data format allows an effective seamless integration of the exchanged data into the LLM-based Decision Agent, allowing it to reason with both local observations and external context. Figure 3.2 illustrates the information exchange process facilitated by the Communication Agent. This multi-agent communication approach will improve the accuracy of the AVs decision by expanding its environmental perception beyond the local perception limit.

Field	Type	Description
`Vehicle_ID`	String	A unique identifier for the broadcasting vehicle.
`Current_Speed`	Float	The vehicle's current speed in meters per second (m/s).
`Obstacle_Presence`	Boolean	Indicates whether the vehicle detects a critical obstacle in its immediate path.
`Obstacle_Type`	Enum / Null	Type of detected obstacle (e.g., 'stop sign', 'pedestrian', 'road block'). This is `null` if `Obstacle_Presence` is `False`.
`Obstacle_Proximity`	Float	Distance to the detected obstacle in meters.
`Intended_Action`	Enum	Planned action from the broadcasting vehicle Decision Agent (e.g., 'stop', 'forward'

TABLE 3.1: Structured Information Schema for V2V Communication Messages.

3.4 Text-Based Simulation and Scenario Design

In this research, we initially used a series of traffic situations and developed text-based simulation scenarios using Python to evaluate the effectiveness of the LLMs as the decision agent in AV. These scenarios were designed to test the capabilities of the LLMs in these three domains, logical reasoning, rule compliance, and dynamic decision making. as shown in Figure 3.3. We extended our experiment to cover these scenarios, while the capability checking domain of the LLMs was the same.

- **Scenario 1: Rule compliance** This scenario tested whether the LLMs could correctly interpret a green traffic light and determine that the AV should proceed. It would assess the basic understanding of LLMs of common traffic rules. This simulation is shown in figure 3.3(A)

- **Scenario 2: Common sense** In this case, a stop sign appears on a partially blocked road. The goal was to evaluate how the LLMs interprets such ambiguous situations and whether it can determine an appropriate action based on partial cues. This simulation is shown in figure 3.3(B)

- **Scenario 3: Dynamic Decision Making** A more complex test involving a pedestrian suddenly crossing in front of the AV. This scenario assessed whether the LLMs could dynamically adapt its decision to an unexpected

```
1    import json
2    from dataclasses import dataclass, asdict
3    from typing import Dict, List, Any
4    import random
5    import time
```

PROBLEMS   OUTPUT   PORTS   TERMINAL   >_ Pyth

```
--- Vehicle 2 Communication Agent: Sending Own Status ---
Nearby Vehicles: ['001', '003']
Vehicle 2 Status Details:
 Vehicle ID: 002
 Obstacle Presence: True
 Obstacle Type: stop_sign
 Obstacle Proximity: 13.123529423503923 meters
 Traffic Density: heavy
 Intended Action: stop

--- Vehicle 2 Communication Agent: Receiving Status from Vehicle 001 ---
Received Status Details:
 Vehicle ID: 001
 Obstacle Presence: True
 Obstacle Type: pedestrian
 Obstacle Proximity: 14.77517543510873 meters
 Traffic Density: heavy
 Intended Action: turn_right

--- Vehicle 2 Communication Agent: Receiving Status from Vehicle 003 ---
Received Status Details:
 Vehicle ID: 003
 Obstacle Presence: False
 Obstacle Type: traffic_light
 Obstacle Proximity: 2.6441289855223764 meters
 Traffic Density: light
 Intended Action: forward
```

FIGURE 3.2: An illustration of Communication Agent

and potentially hazardous situation. This simulation is shown in Figure 3.3(C)

3.5 Prompt Engineering

The Natural Language Processing capabilities of LLMs and the growing interest in their potential applications have made prompt engineering increasingly important, as the primary method of interaction with LLMs is through prompts [50]. The design of the prompt determines the LLM's behavior, making prompt engineering critical, since the prompt acts as a guide and directly impacts the

```
AV = Autonomous Vechiels,
S = Stop Sign of a partial road,
Other lane is open.


AV          S
- - - - - - - - - -


---------------------
Initial Position:
AV          S
- - - - - - - - - -


---------------------
Position Right Before STOP sign:
        AV S
- - - - - - - - - -


---------------------

Simulation ended.
```

(A) Traffic Light

```
H = Human Car,
AV = Autonomous Vechiels,
TL = traffic light.


Initial Position:
AV           TL:green
- - - - - - - - - - -
H
---------------------
Position Right Before TL:
        AV TL:green
- - - - - - - - - - -
        H
---------------------
Simulation ended.
```

(B) Stop Sign on Partial Road

```
H = Human,
AV = Autonomous Vechiels,


Initial Position:


AV
- - - - - - - - - -


---------------------
Position Right Before H:


        AV    H
- - - - - - - - - -


---------------------


Simulation ended.
```

(C) Human Suddenly Crossing Road

FIGURE 3.3: Illustration of real-life traffic scenarios tested in the LLM-based decision system.

results. To carry out our experiment, we tested different LLM models with text-based traffic simulation scenarios using a simple prompt. Each scenario was presented to various LLM models using this simple prompt asking the models to simply respond with a single word without further explanation,

"What would the AV do? Please, just answer STOP or FORWARD."

We are assuming that the output from the LLMs would be used as a real decision in the autonomous driving system.

Chapter 4

Simulation and Results

4.1 Simulation-based Analysis

This section details the responses generated by a diverse set of LLMs when presented with these foundational text-based scenarios. The scenarios were designed to test fundamental aspects of AV intelligence such as basic rule following, common sense, and dynamic decision making. The models' outputs were analyzed for these characteristics:

- Decision Correctness: We evaluated whether LLMs can make safe and reasonable decisions even in unfamiliar situations when provided with appropriate contextual data, without requiring specific training for each individual scenario.

- Following to Instructions: Whether the model's output conformed to the prompt's explicit formatting constraints (e.g., "Please just answer STOP or FORWARD").

- Consistency: We also investigated the consistency of the LLM's response in multiple runs. Inconsistent response is unacceptable in autonomous vehicle driving systems, where safety depends on stable and reliable decision-making behavior.

4.2 Experiment 1: Basic Rule-Following

For initial testing, we selected a basic traffic scenario involving an AV approaching a green traffic light, as illustrated in Figure 3.3 (B), and presented it to various language models to evaluate their decision-making behavior. The scenario was communicated through the prompt: 'What would the AV on the above do? Please just answer STOP or FORWARD.' The selected models included Claude[1] 3.7 Sonnet, GPT[2]-4 Turbo, Falcon[3] 3 7B and open-source alternatives such as

---

[1]https://claude.ai/chats

[2]https://chat.openai.com/

[3]https://chat.falconllm.tii.ae/

OpenHermes-2.5-Mistral-7B and google/flan-t5-large. The responses of these models, sourced from Hugging Face, LM Studio, and commercial providers.

GPT-4 Turbo and DeepSeek-V3 provided direct and expected responses **FORWARD**, which aligns with the logic of the situation shown in Figure 4.1.

However, Claude deviated from the expected format by including an explanatory response, despite the prompt requesting a one-word answer.

OpenHermes-2.5 and Falcon3 7B also responded with a clear **FORWARD**, demonstrating alignment with the input instructions shown in Figure 4.2. Surprisingly, the google/flan-t5-large output differed from all others by responding **STOP**, contradicting the intended logic of the scenario.

(A) Output from DeepSeek-V3	(B) Output from GPT-4 turbo
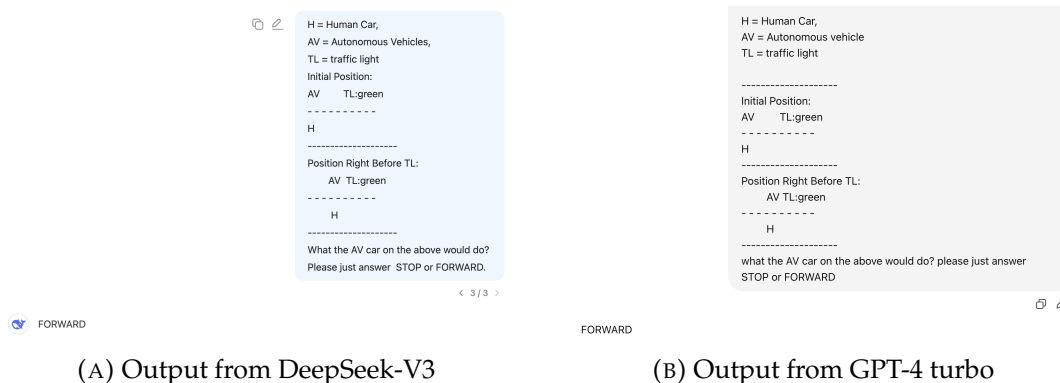
FIGURE 4.1: Responses from different LLMs on Scenario 1: DeepSeek-V3, GPT-4 turbo

(A) Output from Falcon3 7B	(B) Output from OpenHermes-2.5-Mistral-7B

FIGURE 4.2: Responses from different LLMs on Scenario 1: Falcon3 7B, and OpenHermes-2.5-Mistral-7B

The Claude 3 Sonnet model also determined that the correct action was **FORWARD**. However, it failed to follow the prompt's constraint to 'just answer STOP or FORWARD.' Instead, it provided a full sentence explanation of its reasoning shown in Figure 4.3.

```
R'  H = Human Car,
    AV = Autonomous Vehicles,
    TL = traffic light
    Initial Position:
    AV      TL:green
    - - - - - - - - - -
    H
    --------------------
    Position Right Before TL:
        AV  TL:green
    - - - - - - - - - -
       H
    --------------------
    What the AV car on the above would do? Please just answer  STOP or FORWARD.
                                                                          Edit
```

Based on the scenario you've presented:
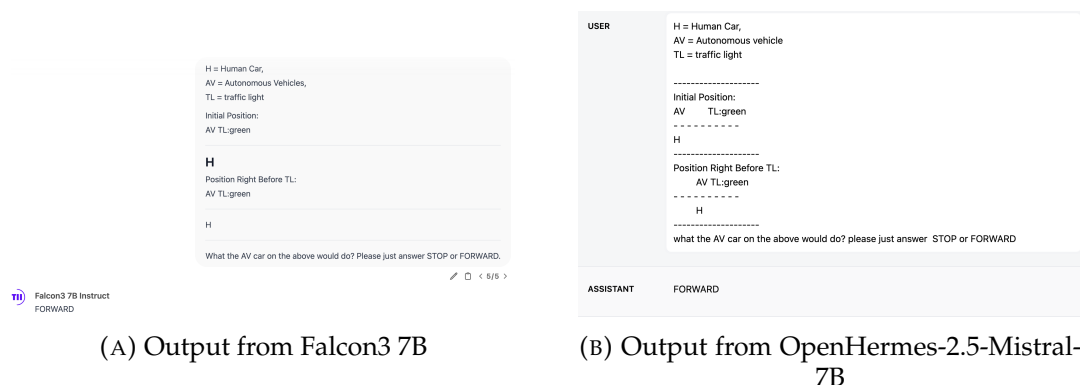
FORWARD

The autonomous vehicle (AV) is approaching a traffic light that is green, and there is a human-driven car (H) in a separate lane that appears to be parallel to the AV's path. Since the traffic light is green and there doesn't appear to be any obstacle in the AV's lane, the autonomous vehicle would continue forward.

FIGURE 4.3: Responses from different LLMs on Scenario 1: Claude 3

The google/flan-t5-large model responded with **STOP** and provided an alternative explanation for its decision shown in Figure 4.4. However, stopping at a green light is incorrect and dangerous in the given situation.

Most of the responses were as expected and also demonstrated that not all LLMs can be used as a decision agent, both the promise and the current limitations of this approach. The results emphasize the importance of careful model selection, as most of the other tested models produced the expected and appropriate response that shows that LLMs have the potential to make driving decisions. However, choosing the right model is a critical factor in ensuring consistent and safe behavior in autonomous vehicle systems.

```
H = Human Car,
AV = Autonomous Vehicles,
TL = traffic light
Initial Position:
AV      TL:green
- - - - - - - - - -
H
- - - - - - - - - - - - - - - - - - -
Position Right Before TL:
    AV  TL:green
- - - - - - - - - -
     H
- - - - - - - - - - - - - - - - - - -
What the AV car on the above would do? Please just answer  STOP or FORWARD.
```

Compute   ⌘+Enter

STOP

FIGURE 4.4: Responses from different LLMs on Scenario 1: google/flan-t5

4.3 Consistency test of Decision Agent : Experiment 1

In safety-critical domains like autonomous driving, consistent decision-making is as vital as correctness. For autonomous vehicles, it is not enough for the decision agent to give the correct answer once, it must give the same correct answer every time when faced with the same situation. A wrong decision, such as stopping when the traffic light is green, can lead to accidents. To address our sub-research question, 'Does repeated querying of LLMs in the same scenario lead to variations in decision-making outcomes?' we conducted an experiment to test the stability responses of LLMs.

We selected the model, OpenHermes-2.5-Mistral-7B and presented the scenario of recognizing a green traffic light. The prompt was executed 20 times and the responses were mapped such as **FORWARD** = 1 (the correct action) and **STOP** = 0 (an incorrect and unsafe action). We then calculated the mean and the variance of the responses to quantify consistency. As shown in Figure 4.5, out of the 20 responses, the model responded **FORWARD** 19 times and **STOP** once. This resulted in a mean of 0.95 and a variance of 0.0475. Although this result indicates that the model is generally reliable in this scenario, even a single incorrect response in real-world traffic can lead to hazardous consequences.
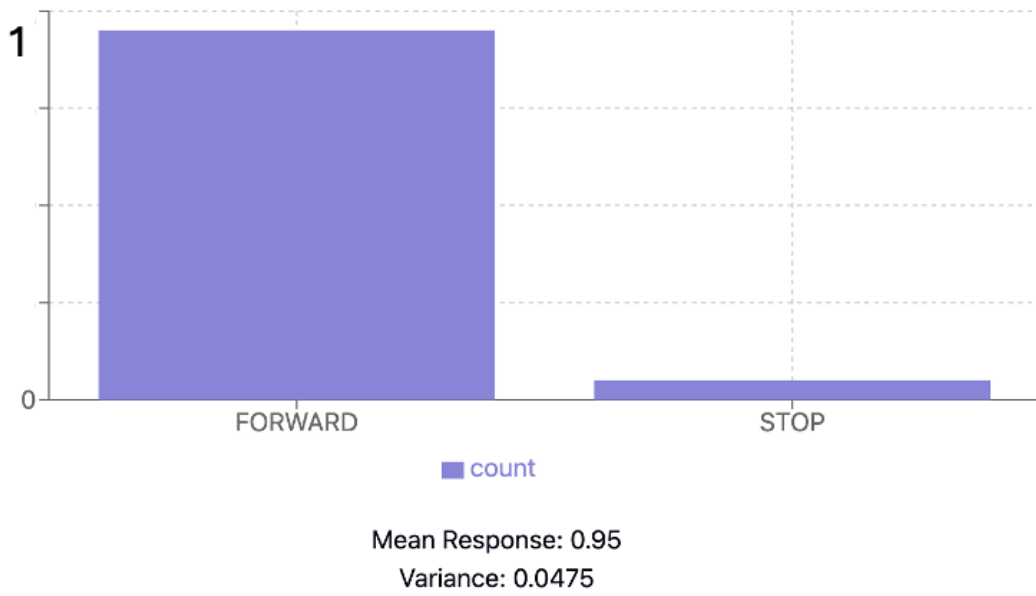
FIGURE 4.5: LLM Model Consistency on the Tested Scenario

To further enhance decision reliability and reduce such variance, this research adopted a multi-agent model. In this setup, a dedicated 'Communication Agent' is responsible for transmitting and receiving structured Vehicle-to-Vehicle (V2V) data. This agent ensured consistent information sharing in a standardized text format, such as obstacle presence, vehicle speed, and intended actions. The Decision Agent would utilize this extended contextual information to make safer and more informed driving decisions.

4.4 Experiment 2: Stop sign view blocked from the following vehicle

To demonstrate how a multi-agent model can improve decision-making reliability, we have prepared a simulation where an AVs view was blocked. In this setup, an autonomous car (AV2) is driving behind an autonomous truck (AV1). There is a stop sign ahead, but AV2 could not see the sign since the truck in front is blocking its view, as shown in Figure 4.6. This kind of situation is common and dangerous in real-world driving, as it can lead to accidents if the car behind does not know why the car in front is stopping. This challenge will become more significant in mixed traffic environments where both human-driven and autonomous vehicles will share the road, each with different driving decision-making mechanisms and behaviors.

In this experiment, we investigated whether the **Decision Agent** can improve its situational awareness by incorporating additional context shared through

```
AV2 = Autonomous Car(Can not see the stop sign),
AV1 = Autonomous Truck,
S = Stop Sign


AV2        AV1 S
- - - - - - - - - - -
_____
Initial Position:
AV2        AV1 S
- - - - - - - - - - -
_____
Position Right Before STOP:
     AV2  AV1 S
- - - - - - - - - - -
_____
Simulation ended.
```

FIGURE 4.6: Extended Traffic Scenario with V2V Communication

a **Communication Agent**. The Communication Agent enables AV2 to receive status updates directly from AV1, extending its perception beyond its own local sensors. The data format included obstacle detection, proximity, and the intended action of AV1,

Information received by AV2 Communication Agent:

- Vehicle ID: 001
- Obstacle Presence: True
- Obstacle Type: Stop sign
- Obstacle Proximity: 15 meters
- Intended Action: stop

4.4.1 Decision Agent's Response: Without V2V Communication Data

First, the scenario was presented to the LLMs. For initial testing, we chose the model GPT-4 Turbo, without any V2V communication data. Based on that perception, the model responded with **FORWARD**, which means the car would continue to go forward, as shown in Figure 4.7.

> **Prompt:** "What would the AV above do? Just answer STOP or FORWARD."
>
> **Model (GPT-4 Turbo) Response:** FORWARD

```
AV2 = Autonomous Car- cant see the stop sign
AV1 = Autonomous Truck,
S = Stop Sign

AV1      S
- - - - - - - - - - - - -


- - - - - - - - - - - - - - - - - - - - - - - - -
Initial Position:
 AV2   AV1    S
- - - - - - - - - - - - -


- - - - - - - - - - - - - - - - - - - - - - - - -
Position Right Before STOP:
   AV2      AV1 S
- - - - - - - - - - - - -


- - - - - - - - - - - - - - - - - - - - - - - - -
Simulation ended.
What the AV2 on the above would do? Please just answer  STOP or
FORWARD.
```
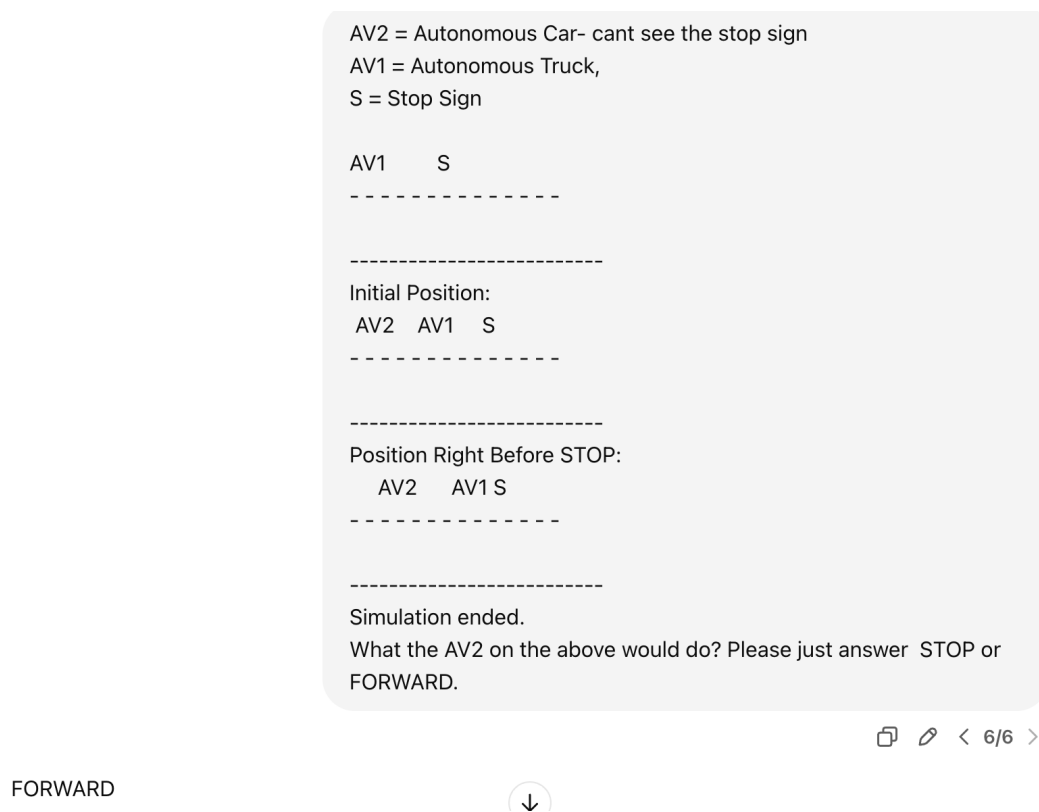
⧉ ✎ ‹ **6/6** ›

FORWARD

FIGURE 4.7: LLM Models Response without Communication agent's information (GPT-4 Turbo)

The same unsafe response, FORWARD, was also responded by Gemini 2.5 Flash, although Claude 3.5 and DeepSeek responded correctly with STOP, as shown in Figure 4.8.
However, both Claude and DeepSeek included additional explanations, despite the fact that the prompt clearly asked for a one-word answer. This inconsistency highlights a key issue: even advanced models can behave unpredictably when

relying solely on their own perception. Without a complete environmental context, such as shared contextual information, they may make unsafe decisions or fail to follow important instructions.
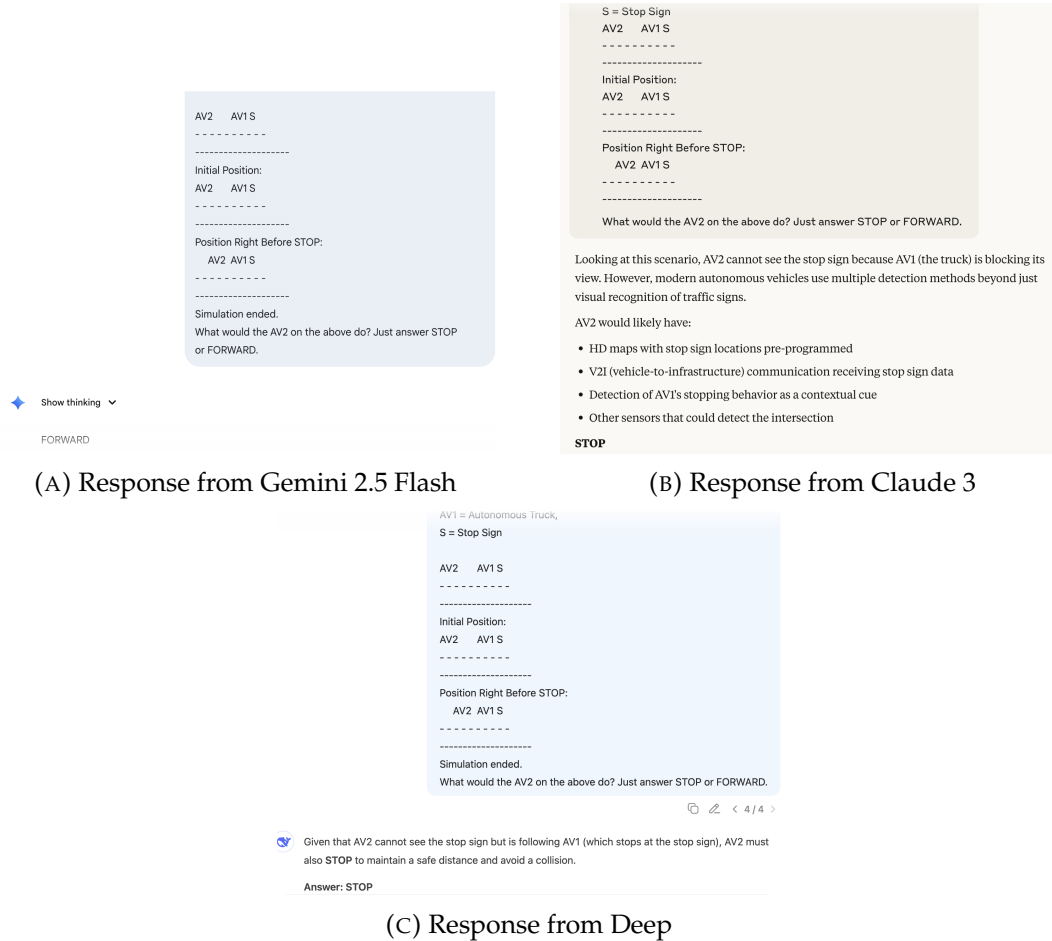
(A) Response from Gemini 2.5 Flash

(B) Response from Claude 3

(C) Response from Deep

FIGURE 4.8: LLM Models Response without V2V Communication support on Experiment-2

4.4.2 Decision Agent's Response: With V2V Communication Data

We then again tested the same situation and examined whether the incorporation of V2V data improves the accuracy of the decisions and enables AVs to make more precise and context-aware decisions. By enabling the Communication Agent, AV1 shares its intended action and information about the obstacle ahead with AV2. These data are structured and passed as additional input presented to the model, as shown in Figure 4.9. For this case, first we used GPT-4 Turbo. With access to this contextual information, the model responds with **STOP**, even without direct visual confirmation of the traffic sign in the

Information from communication agent:
AV2 Communication Agent: Sending Own Status ---
Vehicle 2 Status Details:
  Vehicle ID: 2
  Obstacle Presence: True
  Obstacle Proximity: 14 meters
  Traffic Density: light
  Intended Action: Forward

AV2 Communication Agent: Receiving Status from AV1 ---
Received Status Details:
  Vehicle ID: 001
  Obstacle Presence: True
  Obstacle Proximity: 5 meters
  Traffic Density: light
  Intended Action: stop

What the AV2 on the above would do? Please just answer STOP or
FORWARD.

STOP

FIGURE 4.9: LLM Models Response with V2V Communication
support (GPT-4 Turbo)

case where GPT-4 Turbo was used. Now, with access to this extra context, the
model made a safer decision, even though AV2 could not see the stop sign itself.
This shows how V2V communication improves the accuracy of LLMs decision-
making. Unlike the first run (without communication), where the model chose
to go forward and risked a collision, the shared information helped it choose a
more context-aware and safe action.

We also tested the same scenario with V2V communication using other pop-
ular commercial models, Gemini 2.5 Flash[4], Claude 3.5, and DeepSeek[5]. All of
these models gave the correct response **STOP** after receiving the information
shared by the Communication Agent shown in Figure 4.10. This shows that
when models have access to extra context from V2V data, they can make safer
and more reliable decisions.

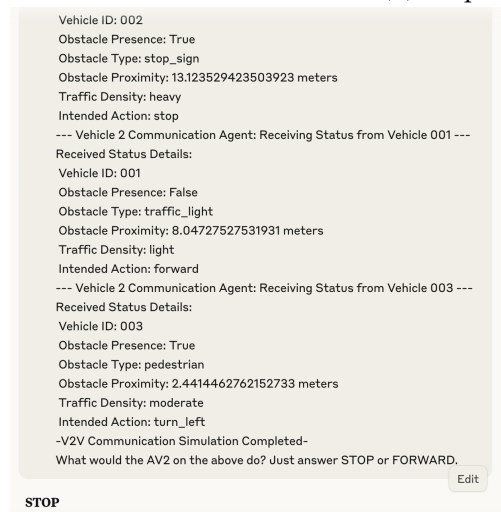[4]https://gemini.google.com/
[5]https://chat.deepseek.com/a/chat
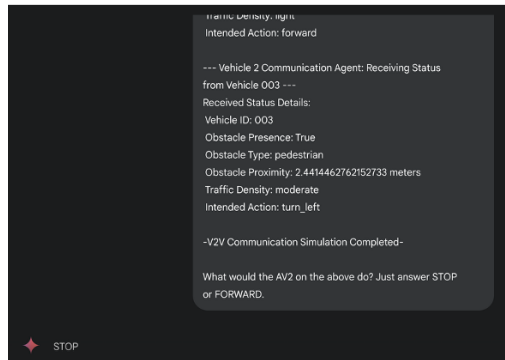
(A) Response from Gemini 2.5 Flash
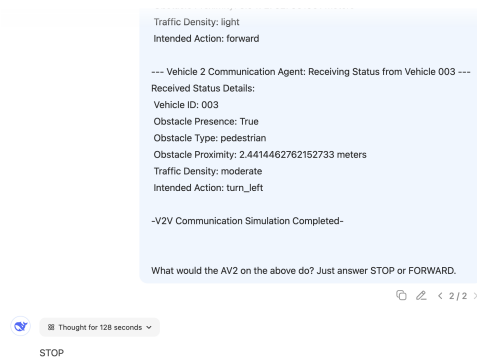
(B) Response from Claude 3

(C) Response from Deep

FIGURE 4.10: LLM Models Response with V2V Communication support on Experiment-2

4.5 Consistency test of Decision Agent: Experiment 2

To confirm that V2V communication also resolves the consistency issue, we selected the model GPT-4 Turbo for further testing. We used the scenario previously described in Figure 4.6, where an autonomous car (AV2) follows a truck (AV1) that sees a stop sign, but AV2 cannot see it due to visual obstruction. We enabled the Communication Agent, which provided AV2 with structured V2V data from AV1, specifically, AV1's perception of the stop sign and its intention to stop. The prompt was executed 20 times and the responses were mapped as STOP = 1 (the correct and safe action) and FORWARD = 0 (an incorrect and unsafe action). We then calculated the mean and the variance of the responses to evaluate consistency. As shown in Figure 4.11, the model responded with STOP in all 20 trials.

This resulted in a mean of 1 and a variance of 0, indicating perfect consistency at least within these 20 trials. These results confirm the potential that incorporating V2V information through our multi-agent model not only improves decision-making accuracy in complex scenarios but also eliminates the variability observed in single-agent systems.
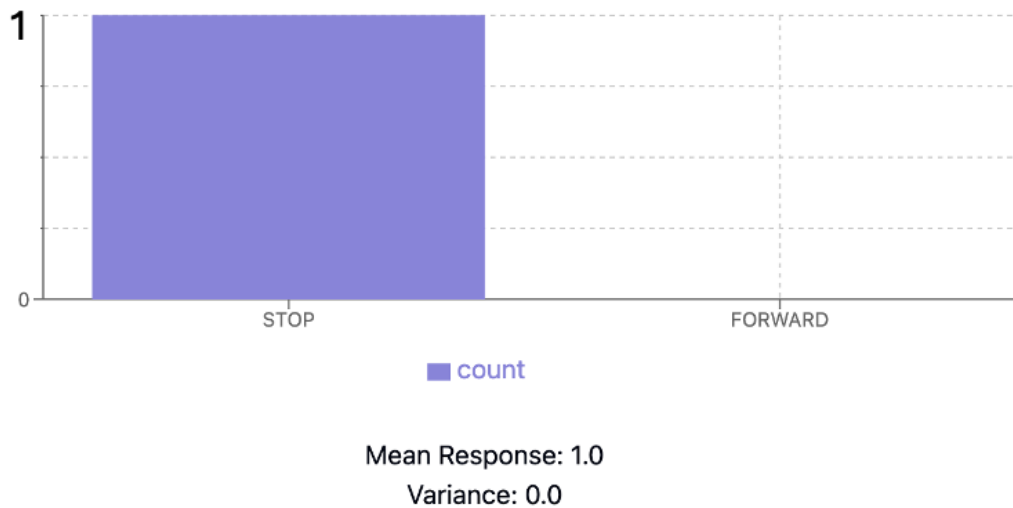

Mean Response: 1.0
Variance: 0.0

FIGURE 4.11: LLM Model Consistency with V2V Communication support(GPT4-Turbo)

4.6 Discussion

The results from these experiments clearly show both the potential and the current limitations of using LLMs as a decision agent in autonomous vehicles.

In Experiment 1, we observed that LLMs demonstrated strong capabilities in understanding and responding to traffic situations using rule-following and common-sense reasoning. However, a major limitation is inconsistent behavior when used without additional context. During consistency evaluation, we tested OpenHermes-2.5-Mistral-7B on the same driving scenario 20 times. The model responded correctly in 19 of 20 trials (mean response = 0.95), while it made an incorrect decision, choosing **STOP** when the correct action was **FORWARD.** As illustrated in Figure 4.5, this variation is a critical safety concern, even a single incorrect decision in real-world driving can result in dangerous consequences that could cause rear-end collisions.

In Experiment 2, we introduced our multi-agent model, which includes a Communication Agent responsible for sharing contextual Vehicle-to-Vehicle (V2V) data. When this enhanced contextual information version of the scenario was repeated for consistency, GPT-4 Turbo responded with correct action (STOP) in 20 of the trials. This resulted in a perfect mean of 1.0 and no variance, which confirmed the potential of the complete consistency of the decision.
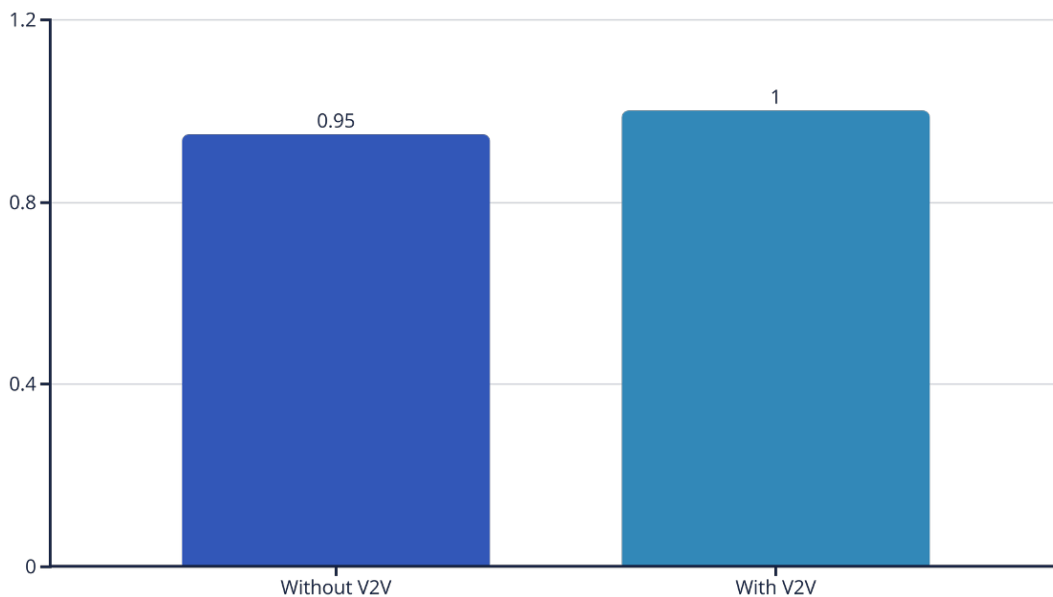

FIGURE 4.12: Comparison of Decision Consistency With and Without V2V Communication

The comparison between the two setups highlights an improvement by the approach. Without V2V communication, the LLM responded correctly most of the time with a variance (0.0475), showing somewhat inconsistent behavior that is unacceptable in safety-critical systems. However, when V2V data was introduced through our multi-agent model, the LLM responded correctly without any variance. As illustrated in Figure 4.12, this demonstrates how shared contextual information enables the LLM to make consistently accurate and safe decisions.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

We presented an approach for leveraging Large Language Models (LLMs) to enhance reasoning capabilities in uncertain traffic situations within a multi-agent model for autonomous vehicles. We have investigated LLMs from Hugging Face, LM Studio, as well as major commercial services such as Claude, GPT-4, and Falcon. Their responses to basic traffic scenarios were analyzed, revealing varying levels of consistency and accuracy. Our results show that:

- For RQ1, LLMs can make decisions without specific learning processes for new situations when provided with the appropriate context. LLMs were able to make intelligent and safe decisions even in unfamiliar or unusual traffic situations. They did not require specific training for each scenario and showed strong general reasoning skills, similar to how a human driver would respond to unexpected conditions.

- For RQ2, the use of multiple agents, such as a Decision Agent and a Communication Agent, improved the system performance compared to the single agent setup. When vehicles shared data such as speed, obstacles, and intentions through V2V communication, decisions became more accurate, consistent, and safer. This design also made the system more scalable and flexible for handling complex real-world traffic situations.

- We also find a challenge of consistency. Repeated querying of LLMs leads to variations in decision-making outcomes, which is critical for safety, more contextual information, such as V2V communication, improves decision accuracy from variation to perfect consistency (GPT-4 Turbo: mean = 1.0).

These results reveal both the promise and the current limitations of this approach, highlighting the need for improved consistency in the model responses. The proposed approach aims at the realization of adaptive autonomous systems capable of human-like reasoning in unpredictable situations. By integrating contextual data derived from V2V communication with LLM reasoning, our goal is to bridge the critical gap in decision-making accuracy for autonomous vehicles.

5.2 Limitations and Future Work

5.2.1 Current Challenges and Limitations

The following are the current challenges and limitations of this study.

- The current study focuses on a limited set of traffic situations. Future work should incorporate a broader range of complex and diverse scenarios to evaluate LLMs based decision agent in more generalize real-world traffic conditions. For the next step, we are expanding simulation to include multi-lane and multi-vehicle scenarios for more realistic testing such as 3 lanes, 3 cars, with varying visibility and road conditions.

- At present, our approach has no fail safe mechanisms or real-time error detection to catch and correct unsafe decisions generated by the LLM. Future research should explore the integration of redundancy systems to ensure the robustness of decision making in critical situations. To improve the safety of our approach as the next step, it is possible to implement a Dynamic Emergency Messaging system. This mechanism will allow a vehicle to broadcast real-time emergency alerts such as a STOP command to nearby vehicles whenever a critical event occurs (e.g., detecting a sudden obstacle or a pedestrian crossing unexpectedly).

- Our study assumes ideal conditions for Vehicle-to-Vehicle (V2V) communication. However, in dense traffic environments, network congestion and latency can affect the information exchange. Furthermore, when the number of vehicles increases, the volume of shared information becomes large, making it challenging to identify and prioritize the most relevant data. Efficient filtering and refinement of this information will be crucial to ensure quick and reliable decision-making. Future work should explore how these communication limitations affect system performance and investigate methods to optimize V2V data handling and reduce latency.

5.2.2 Future Work

A key challenge will involve a human (H) suddenly crossing the road, creating an uncertain situation for the AVs as shown in Figure 5.1. In this case, one of the vehicles, AV3, will not be able to see the human due to limited perception. This will help us study the importance of vehicle-to-vehicle (V2V) communication in preventing accidents.

In this extended simulation scenario, all autonomous vehicles will share detailed information each other, including their position, speed, nearby obstacles, and planned actions. This V2V data, along with visual input from each vehicle, will be passed to the LLMs. We can then analyze how the LLM uses this

```
H = Human,
AV1-AV7 = Autonomous Vechiels,
Initial Position:
AV1

- - - - - - - - - - - -


---------------------
Position Right Before Human:
    AV1     H

- - - - - - - - - - - -

 AV2
Final Position:
   AV6 AV3  AV1 H

- - - - - - - - - - - -

  AV2   AV4   AV5 AV7

---------------------
```

FIGURE 5.1: Expanded simulations of multi-lane roads with sudden human crossing for Next Step

combined information to make safe and reasonable decisions in a complex and partially visible environment.

The simulation will incorporate extensive V2V communication, where a large amount of information about their current status and intentions will be shared, as shown in the Figure 5.2.

```
--- AV2 Communication Agent: EMERGENCY — Human Crossing Detected ---
Broadcasting to Nearby Vehicles: ['AV1', 'AV3', 'AV4', 'AV5', 'AV6', 'AV7']
AV2 Status Details:
 Vehicle ID: AV2
 EMERGENCY STATUS: True
 Human Detected: True
 Obstacle Type: human_pedestrian
 Human Proximity: 8.5 meters
 Current Speed: 25.0 km/h
 Intended Action: emergency_brake
 Lane Position: same_lane

--- AV2 Communication Agent: Receiving Status from AV1 ---
Received Status Details:
 Vehicle ID: AV1
 Emergency Status: True
 Human Detected: True
 Obstacle Presence: True
 Obstacle Type: human_pedestrian
 Obstacle Proximity: 23.0 meters
 Current Speed: 0.0 km/h
 Intended Action: emergency_brake
 Lane Position: same_lane
```

```
--- AV2 Communication Agent: Receiving Status from AV3 ---
Received Status Details:
 Vehicle ID: AV3
 Emergency Status: False
 Human Detected: True
 Obstacle Presence: True
 Obstacle Type: human_pedestrian
 Obstacle Proximity: 12.0 meters
 Current Speed: 15.0 km/h
 Intended Action: slow_down
 Lane Position: adjacent_left

--- AV2 Communication Agent: Receiving Status from AV4 ---
Received Status Details:
 Vehicle ID: AV4
 Emergency Status: False
 Human Detected: True
 Obstacle Presence: True
 Obstacle Type: human_pedestrian
 Obstacle Proximity: 18.7 meters
 Current Speed: 20.0 km/h
 Intended Action: slow_down
 Lane Position: opposite_lane
```

```
--- AV2 Communication Agent: Receiving Status from AV5 ---
Received Status Details:
 Vehicle ID: AV5
 Emergency Status: False
 Human Detected: True
 Obstacle Presence: True
 Obstacle Type: human_pedestrian
 Obstacle Proximity: 16.5 meters
 Current Speed: 12.0 km/h
 Intended Action: slow_down
 Lane Position: adjacent_left

--- AV2 Communication Agent: Receiving Status from AV6 ---
Received Status Details:
 Vehicle ID: AV6
 Emergency Status: False
 Human Detected: True
 Obstacle Presence: True
 Obstacle Type: human_pedestrian
 Obstacle Proximity: 21.2 meters
 Current Speed: 10.0 km/h
 Intended Action: slow_down
 Lane Position: adjacent_left

--- AV2 Communication Agent: Receiving Status from AV7 ---
Received Status Details:
 Vehicle ID: AV7
 Emergency Status: False
 Human Detected: True
 Obstacle Presence: True
 Obstacle Type: human_pedestrian
 Obstacle Proximity: 25.8 meters
 Current Speed: 8.0 km/h
 Intended Action: slow_down
 Lane Position: adjacent_left
```

FIGURE 5.2: Illustration of extended experiments V2V exchanged data among autonomous vehicles.

We can provide this large amount of V2V data, which includes different perceptions and perspectives from each AV to the LLMs. Since these data may contain overlapping or less relevant information, some filtering might be necessary. As an initial step, we investigated whether LLMs can process this diverse and complex input effectively and still make safe and correct driving decisions.

Further investigations of the extensions to handle these complicated cases in an efficient computation cost is one of our future works.

Bibliography

[1] Y. Chang, X. Wang, J. Wang, *et al.*, "A survey on evaluation of large language models," *ACM transactions on intelligent systems and technology*, vol. 15, no. 3, pp. 1–45, 2024.

[2] Y. Wang, Z. Wu, J. Yao, and J. Su, "Tdag: A multi-agent framework based on dynamic task decomposition and agent generation," *Neural Networks*, p. 107 200, 2025.

[3] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[4] J. L. Espejel, E. H. Ettifouri, M. S. Y. Alassan, E. M. Chouham, and W. Dahhane, "Gpt-3.5, gpt-4, or bard? evaluating llms reasoning ability in zero-shot setting and performance boosting through prompts," *Natural Language Processing Journal*, vol. 5, p. 100 032, 2023.

[5] M. Raza, "Autonomous vehicles: Levels, technologies, impacts and concerns," *International Journal of Applied Engineering Research*, vol. 13, no. 16, pp. 12 710–12 714, 2018.

[6] T. Eley IV, J. L. King, K. Lyytinen, and J. V. Nickerson, *SAE J3016 as a Learning Device for the Driving Automation Community: Technical, Socio-technical, and Systemic Learning*. SAE International, 2024.

[7] C. Geertz, "Common sense as a cultural system," *The Antioch Review*, vol. 33, no. 1, pp. 5–26, 1975.

[8] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[9] D. Bogdoll, J. Breitenstein, F. Heidecker, *et al.*, "Description of corner cases in automated driving: Goals and challenges," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 1023–1028. DOI: 10.1109/ICCVW54120.2021.00119.

[10] H. He, J. Gray, A. Cangelosi, Q. Meng, T. M. McGinnity, and J. Mehnen, "The challenges and opportunities of human-centered ai for trustworthy robots and autonomous systems," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 4, pp. 1398–1412, 2022. DOI: 10.1109/TCDS.2021.3132282.

[11] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020. DOI: 10.1109/ACCESS.2020.2983149.

[12] P. N. Johnson-Laird, "Mental models and human reasoning," *Proceedings of the National Academy of Sciences*, vol. 107, no. 43, pp. 18 243–18 250, 2010.

[13] A. Demba and D. P. Möller, "Vehicle-to-vehicle communication technology," in *2018 IEEE international conference on electro/information technology (EIT)*, IEEE, 2018, pp. 0459–0464.

[14] S. Pendleton, H. Andersen, X. Du, *et al.*, "Perception, planning, control, and coordination for autonomous vehicles," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:114862052.

[15] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[16] D. Fu, X. Li, L. Wen, *et al.*, "Drive like a human: Rethinking autonomous driving with large language models," *2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, pp. 910–919, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259924488.

[17] R. Schaeffer, B. Miranda, and S. Koyejo, "Are emergent abilities of large language models a mirage?" *Advances in neural information processing systems*, vol. 36, pp. 55 565–55 581, 2023.

[18] X. Wang, W. Zhu, M. Saxon, M. Steyvers, and W. Y. Wang, "Large language models are latent variable models: Explaining and finding good demonstrations for in-context learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 15 614–15 638, 2023.

[19] M. Shanahan, K. McDonell, and L. Reynolds, "Role play with large language models," *Nature*, vol. 623, no. 7987, pp. 493–498, 2023.

[20] L. H. M. Lam, R. K. Thatikonda, and E. Shareghi, "A closer look at tool-based logical reasoning with llms: The choice of tool matters," in *Australasian Language Technology Association Workshop*, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:270219176.

[21] J. Gehring, K. Zheng, J. Copet, *et al.*, "Rlef: Grounding code llms in execution feedback with reinforcement learning," *arXiv preprint arXiv:2410.02089*, 2024.

[22] A. Brohan, Y. Chebotar, C. Finn, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on robot learning*, PMLR, 2023, pp. 287–318.

[23] X. Deng, Y. Gu, B. Zheng, *et al.*, "Mind2web: Towards a generalist agent for the web," *Advances in Neural Information Processing Systems*, vol. 36, pp. 28 091–28 114, 2023.

[24] J. Wei, X. Wang, D. Schuurmans, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[25] M. Wooldridge, *An introduction to multiagent systems*. John wiley & sons, 2009.

[26] N. Ayache, A. Yahyaouy, and S. M. Abdelouahed, "An autonomous vehicular system based on muli-agents control: Architecture and behavior simulation," in *2017 Intelligent Systems and Computer Vision (ISCV)*, IEEE, 2017, pp. 1–7.

[27] O. F. Rana and K. Stout, "What is scalability in multi-agent systems?" In *Proceedings of the fourth international conference on Autonomous agents*, 2000, pp. 56–63.

[28] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, pp. 10 164–10 183, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259287283.

[29] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, "Detr3d: 3d object detection from multi-view images via 3d-to-2d queries," *ArXiv*, vol. abs/2110.06922, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:238744328.

[30] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on control systems technology*, vol. 15, no. 3, pp. 566–580, 2007.

[31] J. Van Brummelen, M. OâĂŹbrien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transportation research part C: emerging technologies*, vol. 89, pp. 384–406, 2018.

[32] J. Cao, S. Leng, and K. Zhang, "Multi-agent learning empowered collaborative decision for autonomous driving vehicles," *2020 International Conference on UK-China Emerging Technologies (UCET)*, pp. 1–4, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:222096915.

[33] A. Iftekhar and N. Fukuta, "Toward an efficient and resilient electric vehicle charging place allocation using multiagent approach," in *2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, 2024, pp. 663–664. DOI: 10.1109/IIAI-AAI63651.2024.00126.

[34] X. Wang, B. Lv, K. Wang, and R. Zhang, "Asts: Autonomous switching of task-level strategies," ENG, *International Journal of Applied Mathematics and Computer Science*, vol. 33, no. 4, 553âĂŞ568, 2023.

[35] J. Liang, Y. Li, G. Yin, *et al.*, "A mas-based hierarchical architecture for the cooperation control of connected and automated vehicles," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 2, pp. 1559–1573, 2022.

[36] D. Fu, X. Li, L. Wen, *et al.*, "Drive like a human: Rethinking autonomous driving with large language models," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 910–919.

[37] L. Chen, O. Sinavski, J. Hünermann, *et al.*, "Driving with llms: Fusing object-level vector modality for explainable autonomous driving," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 14 093–14 100.

[38] L. Wang, Y. Ren, H. Jiang, *et al.*, "Accidentgpt: A v2x environmental perception multi-modal large model for accident analysis and prevention," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2024, pp. 472–477.

[39] M. Elhenawy, A. Abutahoun, T. I. Alhadidi, *et al.*, "Visual reasoning and multi-agent approach in multimodal large language models (mllms): Solving tsp and mtsp combinatorial challenges," *arXiv preprint arXiv:2407.00092*, 2024.

[40] W. Wang, J. Xie, C. Hu, *et al.*, "Drivemlm: Aligning multi-modal large language models with behavioral planning states for autonomous driving," *arXiv preprint arXiv:2312.09245*, 2023.

[41] C. Cui, Y. Ma, X. Cao, W. Ye, and Z. Wang, "Drive as you speak: Enabling human-like interaction with large language models in autonomous vehicles," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 902–909.

[42] A. K, S. Sudarshan G S, and S. J U, "Llm's for autonomous driving: A new way to teach machines to drive," in *2023 3rd International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, 2023, pp. 1–6. DOI: 10.1109/ICMNWC60182.2023.10435998.

[43] L. Chen, O. Sinavski, J. HÃijnermann, *et al.*, "Driving with llms: Fusing object-level vector modality for explainable autonomous driving," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 14 093–14 100. DOI: 10.1109/ICRA57147.2024.10611018.

[44] C. Cui, Y. Ma, X. Cao, *et al.*, "A survey on multimodal large language models for autonomous driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 958–979.

[45] B. Friedman and H. Nissenbaum, "Bias in computer systems," *ACM Transactions on information systems (TOIS)*, vol. 14, no. 3, pp. 330–347, 1996.

[46] S. S. Duddela and N. Fukuta, "Toward an acceptable dynamic allocation considering fairness on autonomous car parking," in *2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, 2024, pp. 286–291. DOI: 10.1109/IIAI-AAI63651.2024.00061.

[47] R. Jin, J. Du, W. Huang, *et al.*, "A comprehensive evaluation of quantization strategies for large language models," *ArXiv*, vol. abs/2402.16775, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:268032411.

[48] R. K. Kodali, Y. P. Upreti, and L. Boppana, "A quantization approach for the reduced size of large language models," *2024 16th International Conference on Knowledge and Smart Technology (KST)*, pp. 144–148, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:269189980.

[49] D. Jiang and L. Delgrossi, "Ieee 802.11p: Towards an international standard for wireless access in vehicular environments," *VTC Spring 2008 - IEEE Vehicular Technology Conference*, pp. 2036–2040, 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:6891449.

[50] D. C. Schmidt, J. Spencer-Smith, Q. Fu, and J. White, "Towards a catalog of prompt patterns to enhance the discipline of prompt engineering," *ACM SIGAda Ada Letters*, vol. 43, no. 2, pp. 43–51, 2024.

Publications

International Conference

1. M. A. Rafi, and N. Fukuta, "Toward a Multi Agent Approach for LLM-Based Dynamic Vehicle Control and Communication in Accidental Condition", Proc. 18th IIAI International Congress on Advanced Applied Informatics (IIAI AAI2025 / SCAI2025), 2025. (Full Paper, to be published from IEEE)

Toward a Multi Agent Approach for LLM-Based Dynamic Vehicle Control and Communication in Accidental Condition

Rafi Md Ashifujjman
Graduate School of Integrated Science and Technology
Shizuoka University
Hamamatsu, Japan
rafi.md.ashifujjman.23@shizuoka.ac.jp

Naoki Fukuta
College of Informatics, Academic Institute
Shizuoka University
Hamamatsu, Japan
fukuta@inf.shizuoka.ac.jp

Abstract—In autonomous vehicles (AVs), making the right decision in unknown traffic situations remains a challenge. Traditional AVs rely on pre-trained models, which often struggle in such cases where human-like reasoning is required. In this paper, we examine how Large Language Models (LLMs) act as a decision agent responding to scenario-specific text prompts derived from real-world traffic situations. In addition, we propose a Communication Agent that enables vehicle-to-vehicle (V2V) information sharing, such as speed, obstacles, and future intentions, in a structured format to provide contextual support for the decision agent. This paper proposes a preliminary approach to assess whether LLMs within a multi-agent framework, when provided with structured prompts and contextual data, can support consistent and human-like decision making.

Index Terms—autonomous vehicles, large language models (LLMs), multi-agent, vehicle-to-vehicle (V2V) communication, decision making agent, communication agent.

I. INTRODUCTION

This paper explores the potential of integrating Large Language Models (LLMs) [1] into a multi-agent framework [2] to improve decision making in autonomous vehicles (AVs), particularly in unknown and unsafe traffic situations. Our main concern would be to explore the use of LLMs for autonomous driving systems (ADS) as the main decision-making agent within a multi-agent framework to evaluate its reasoning ability [3] to handle uncertain traffic situations. Our goal is to contribute to the development of safer and more reliable Level 5 [4] autonomous vehicles. We aim to contribute to Level 5 (complete automation [5]). There are numerous instances in which traditional autonomous vehicles are capable of making effective decisions in familiar scenarios using pre-trained models. However, they may have trouble facing new or unusual situations where their programmed knowledge is not directly covered in such situations. Human drivers can use common sense and past experiences to handle unexpected events, such as knowing that traffic cones on a moving truck are not dangerous. However, current AV systems, even those that use rule-based approaches or reinforcement learning (RL [6]), will struggle in these scenarios. This limitation is especially evident in what we call unknown-unsafe domain situations

where the correct action is not immediately clear [7]. During this investigation with various LLMs, we proceed under two assumptions: 1) certain AI systems can transform real-world situations into text-based explanations, which are later used as input for the LLMs, and 2) the outputs generated by the LLMs can be translated into actual decisions made by the AV's decision agent by interpreting the responses from the LLMs. We are investigating the effectiveness of various LLMs for use as decision-making agents in autonomous driving systems. Our initial objective is to answer the research question (Research Question 1): Can an agent make decisions with the help of an LLM without undergoing a specific learning process for each new situation? In our research, we will assume that vehicle-to-vehicle (V2V) communication has been achieved through a standardized networking protocol, such as LAN-based direct communication or a low-latency wireless system. Under this assumption, each vehicle can communicate with other vehicles within a defined radius (e.g., 10 meters). When a vehicle enters the information sharing radius of another vehicle, it can exchange data regarding their current status, such as obstacle detection, traffic conditions, direction, and future intended actions. To evaluate the impact of V2V communication on autonomous decision-making, we will investigate how LLMs process and utilize this information. Specifically, we will examine whether the incorporation of V2V exchanged data improves decision accuracy and enables AVs to make more precise and context-aware decisions. However, V2V communication introduces a critical question, (Research Question 2): Is a single centralized decision-making agent sufficient to process all incoming V2V data and make autonomous driving decisions, or is a multi-agent system required for improved efficiency and scalability? In this paper, we investigate the answers to these two research questions.

II. MOTIVATION AND BACKGROUND

Imagine a traffic situation, where a road is partially blocked due to construction, with traffic cones and a 'STOP' sign indicating a partial road block. A typical autonomous vehicle (AV) might see the 'STOP' sign and interpret it as a complete
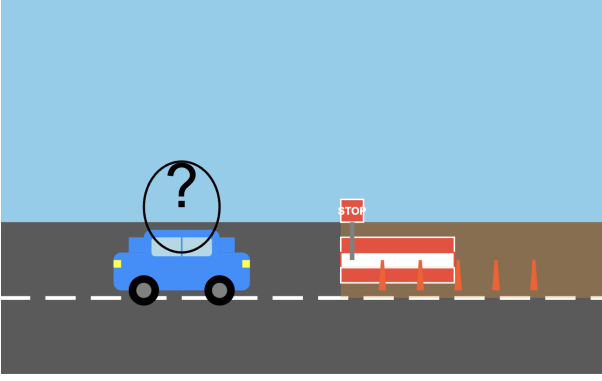
Fig. 1: What a current AV might do in this situation?

road closure, coming to a complete stop because that is what its pre-programmed data tell it to do. However, a human driver in the same situation might use common sense, realize that the road is only partially closed, and safely continue through the open section.
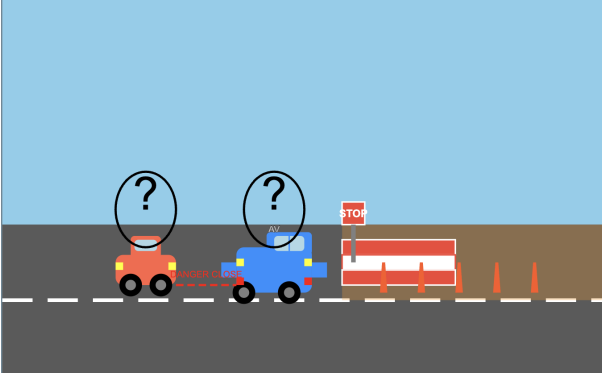

Fig. 2: Lack of communication between vehicles may lead to a potential accident.

Now imagine a more complicated situation where an autonomous truck comes to a sudden stop when it sees the 'STOP' sign. There is another car, driven by a human or another AV, right behind it. This sudden stop could easily cause a crash. Now imagine a human driver encountering a partially blocked road. To navigate safely, the driver would shift into the adjacent open lane, relying on instinct, experience, and situational awareness. Anticipating the possibility of an oncoming vehicle from the opposite direction or from behind, the driver would proactively slow down, assessing the risk, and would take the necessary precautions to avoid a collision. In contrast, a traditional AV, which lacks human-like reasoning and predictive thinking, may not recognize the potential danger. Without contextual understanding, it can continue at its normal speed, assuming the lane is clear, increasing the risk of a head-on collision.

However, if the AV had better reasoning abilities, it might recognize that the road is only partially closed and proceed safely through the open part. And if the AV could communicate with the vehicle behind it, it could alert the other

driver about its intended actions, particularly if the lane were to close suddenly, helping to prevent a potential accident. A fundamental challenge in traditional autonomous vehicles (AVs) is to prepare for all truly unknown situations [8]. Any scenario we create is technically 'known' to us, even if it feels unfamiliar to the AVs, making it difficult to test how AVs handle truly unknown-unsafe situations. In addition, it is difficult to recreate complex real-world scenarios in simulation environments. For example, it is difficult to accurately simulate unpredictable human actions or complicated environmental conditions. In real life, there could be countless unexpected situations that are nearly impossible to include in training data sets for traditional autonomous driving systems (ADS) to avoid accidents. Human drivers use common sense to deal with these situations, so through this paper we will explore whether LLMs can also show human-like reasoning to handle them effectively or not.

To address these challenges, we propose a possible approach for integrating LLMs into a multi-agent framework for autonomous driving control system. LLMs have the potential to provide common sense or reasoning ability that traditional systems lack, helping AVs better understand the context of complex real-world scenarios. Incorporating Vehicle-to-Vehicle communication using multi-agent, which will allow vehicles to exchange information with other vehicles. This capability is also essential for creating realistic and complex driving scenarios.

III. PRELIMINARY

Recommendation	Model Name	Benchmark Performance
LM Studio Recommended Models	Mistral 7B	74.6% (MMLU), 83.5% (GSM8K)
	OpenHermes 2.5	72.1% (MMLU), 79.2% (BIG-bench)
	DeepSeek 7B	76.3% (MMLU), 85.4% (GSM8K)
Hugging Face Open-Source Models	LLaMA 2 13B	75.8% (MMLU), 80.7% (BIG-bench)
	Mixtral 8x7B	78.2% (MMLU), 88.1% (GSM8K)
Top Commercial Services	Claude 3 Sonnet	82.3% (MMLU), 90.2% (BIG-bench)
	GPT-4 Turbo	85.6% (MMLU), 92.4% (GSM8K)
	Falcon 7B/40B	73.4% (MMLU), 78.9% (BIG-bench)

Fig. 3: Selected Models for the Experimentation

To investigate the use of LLMs as a decision agent in AVs, we have utilized open-source and local deployment platforms. Hugging Face[1] offers a wide range of open-source pre-trained models, while LM Studio[2] integrates LLM models from Hugging Face into a local system, even without powerful GPUs. Now, the question is what factors were considered in the selection of LLMs for this study? One key factor in

---

[1] https://huggingface.co/
[2] https://lmstudio.ai/

(a) Stop sign on Partial road   (b) Traffic Light   (c) Human suddenly crossing road

Fig. 4: Illustration of real life traffic scenarios

our model selection was quantization, which optimizes model performance while reducing computational requirements. This research [9] indicates that 8-bit quantization enables the majority of LLMs to maintain a performance level comparable to their nonquantized equivalents, regardless of model size (e.g. 7B to 70B parameters). Moreover, LLMs that are quantized to 4 bits can also uphold similar performance to their non-quantized versions across most benchmarks. This approach achieves memory reduction of 50 to 75% while preserving precision in complex tasks such as reasoning, decision making, and domain-specific applications. The models chosen for this investigation were selected using a structured approach based on three key factors: popularity and performance in text-to-text generation, LM Studio recommendations for optimized accuracy, and comparative evaluations of the top commercial services. Specifically, 1) Some models were chosen based on the highest number of downloads from Hugging Face, ensuring widespread adoption and benchmark effectiveness. 2) Models suggested by LM Studio were included due to their strong performance and compatibility with local inference environments. 3) The best commercial services were selected based on their comparative performance with open-source counterparts, prioritizing accuracy, interpretability, and real-time inference. The selected models are shown in Fig. 3. However, not all Hugging Face models are fully compatible with the LM Studio runtime. As a result, some models were tested directly on the Hugging Face interface to avoid compatibility issues. Furthermore, while models such as OpenAI's o1, DeepSeek R1, and Llama 3.1 405B have demonstrated strong benchmark performance, they were not included in this study due to limited quantization support, lack of local deployment feasibility, and restricted open-source availability.

To test these selected models, we have prepared a set of simple text-based simulation scenarios using Python, as illustrated in Fig. 4. These scenarios are designed to cover situations that require logical reasoning or the application of common rules knowledge, such as recognizing a red traffic light and stopping accordingly. These scenarios will be used as input for various LLMs to assess how effectively they handle both types of challenges. The experiments will be conducted using the LLM selected in Fig.3.

IV. OUR APPROACH

The main phase of this research focuses on using the shared data of the communication agent within our proposed multi-agent framework. The Communication Agent ensures that vehicles share critical information, including obstacle detection, traffic conditions, speed, and intended actions, in a standardized format within a defined radius, the expected format is demonstrated in Fig. 5. This structured data exchange is expected to significantly improve the accuracy of Decision Agent by providing a more comprehensive contextual understanding of the driving environment. For initial testing, we selected a basic traffic scenario involving an autonomous vehicle (AV) approaching a green traffic light, as illustrated in Fig. 4(b), and presented it to various language models to evaluate their decision-making behavior. The scenario was communicated through the prompt: 'What would the AV on the above do? Please just answer STOP or FORWARD.' The selected models included Claude[3] 3.7 Sonnet, GPT[4]-4 Turbo, Falcon[5] 3 7B and open-source alternatives such as OpenHermes-2.5-Mistral-7B and google/flan-t5-large. The responses of these models, sourced from Hugging Face, LM

[3]https://claude.ai/chats
[4]https://chat.openai.com/
[5]https://chat.falconllm.tii.ae/

```
1    import json
2    from dataclasses import dataclass, asdict
3    from typing import Dict, List, Any
4    import random
5    import time
```

PROBLEMS   OUTPUT   PORTS   TERMINAL   ⟩_ Pyth

```
––– Vehicle 2 Communication Agent: Sending Own Status –––
Nearby Vehicles: ['001', '003']
Vehicle 2 Status Details:
 Vehicle ID: 002
 Obstacle Presence: True
 Obstacle Type: stop_sign
 Obstacle Proximity: 13.123529423503923 meters
 Traffic Density: heavy
 Intended Action: stop

––– Vehicle 2 Communication Agent: Receiving Status from Vehicle 001 –––
Received Status Details:
 Vehicle ID: 001
 Obstacle Presence: True
 Obstacle Type: pedestrian
 Obstacle Proximity: 14.77517543510873 meters
 Traffic Density: heavy
 Intended Action: turn_right

––– Vehicle 2 Communication Agent: Receiving Status from Vehicle 003 –––
Received Status Details:
 Vehicle ID: 003
 Obstacle Presence: False
 Obstacle Type: traffic_light
 Obstacle Proximity: 2.6441289855223764 meters
 Traffic Density: light
 Intended Action: forward
```

Fig. 5: An illustration of Communication Agent

Studio, and commercial providers, are shown in Fig. 11. GPT-4 Turbo and Falcon provided direct and expected responses 'FORWARD', which aligns with the logic of the situation. However, Claude deviated from the expected format by including an explanatory response, despite the prompt requesting a one-word answer. OpenHermes-2.5 responded with a clear 'FORWARD', demonstrating alignment with the input instructions. Surprisingly, the google/flan-t5-large output differed from all others by responding 'STOP', contradicting the intended logic of the scenario.

A major consideration was consistency. The subresearch question, 'Does repeated querying of LLMs on the same scenario lead to variations in decision-making outcomes?' will be one of such investigations of consistency. To test this, each scenario was submitted to the models multiple times for regenerating the outputs. This approach allowed us to observe whether the models produced stable and repeatable responses
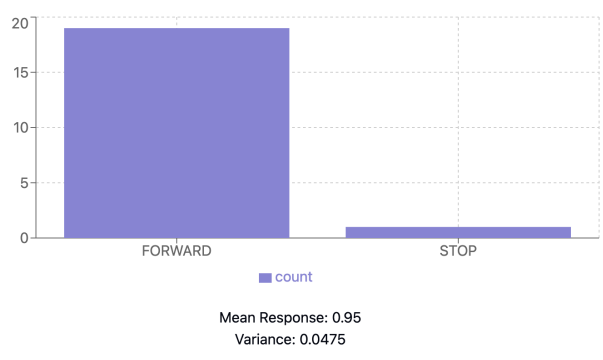
Mean Response: 0.95
Variance: 0.0475

Fig. 6: LLM Model Consistency on the Tested Scenario

or if their decisions varied unpredictably. Consistency is especially critical in autonomous vehicle (AV) applications, where uncertain output can cause serious safety concerns.

For giving an example of this analysis, we ran the prompt at least 20 times on OpenHermes-2.5-Mistral-7B, and the consistency results are presented in Fig. 6. Consistency was calculated by mapping 'FORWARD' = 1 and 'STOP' = 0. 'FORWARD' mean is closer to 1, it indicates that the output is having greater consistency with the expected response.

```
AV2 = Autonomous Car,
AV1 = Autonomous Truck,
S   = Stop Sign


AV1            S
– – – – – – – – – – – – – – –


–––––––––––––––––––––––––––––
Initial Position:
 AV2     AV1      S
– – – – – – – – – – – – – – –


–––––––––––––––––––––––––––––
Position Right Before STOP:
     AV2        AV1 S
– – – – – – – – – – – – – – –


–––––––––––––––––––––––––––––
Simulation ended.
```

Fig. 7: Extended Traffic Scenario with V2V Communication

To investigate the impact of V2V communication on autonomous decision-making, we extended our simulation to a more complex traffic scenario. In this extended scenario, AV2 (an autonomous car) follows AV1 (an autonomous truck), with a stop sign ahead that AV2 cannot see because its line of sight is blocked by AV1, as shown in Fig. 7. To investigate how LLMs process and utilize this information, this scenario was presented to the LLM with the same prompt: 'What would the AV on the above do? Please, just answer STOP or FORWARD'. The model responded with 'FORWARD', as shown in Fig. 8, for the case of using GPT-4 Turbo. This decision could potentially lead to an accident. We then examine whether the incorporation of V2V exchanged data improves the accuracy of the decisions and enables AVs to make more precise and context-aware decisions. By enabling
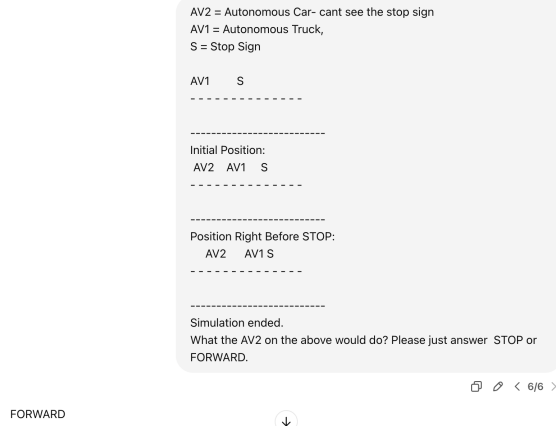
```
AV2 = Autonomous Car- cant see the stop sign
AV1 = Autonomous Truck,
S = Stop Sign

AV1      S
- - - - - - - - - - - - - -


- - - - - - - - - - - - - - - - - - - - - - - - - -
Initial Position:
 AV2  AV1   S
- - - - - - - - - - - - - -


- - - - - - - - - - - - - - - - - - - - - - - - - -
Position Right Before STOP:
   AV2      AV1 S
- - - - - - - - - - - - - -


- - - - - - - - - - - - - - - - - - - - - - - - - -
Simulation ended.
What the AV2 on the above would do? Please just answer  STOP or
FORWARD.
```

FORWARD

Fig. 8: LLM Model response without Communication agent's information (GPT-4 Turbo)


```
Information from communication agent:
AV2 Communication Agent: Sending Own Status ---
Vehicle 2 Status Details:
  Vehicle ID:  2
  Obstacle Presence: True
  Obstacle Proximity: 14 meters
  Traffic Density: light
  Intended Action: Forward

AV2 Communication Agent: Receiving Status from AV1 ---
Received Status Details:
  Vehicle ID: 001
  Obstacle Presence: True
  Obstacle Proximity: 5 meters
  Traffic Density: light
  Intended Action: stop

What the AV2 on the above would do? Please just answer  STOP or
FORWARD.
```

STOP

Fig. 9: LLM Model response with Communication agent's information (GPT-4 Turbo)

the Communication Agent, AV1 shares its intended action and information about the obstacle ahead with AV2. These data are structured and passed as additional input presented to the model, as shown in Fig. 9, for the case of using GPT-4 Turbo. With access to this contextual information, the model responds with 'STOP', even without direct visual confirmation of the traffic sign in the case where GPT-4 Turbo was used.
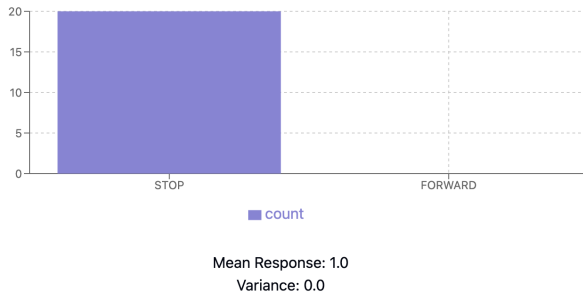

Mean Response: 1.0
Variance: 0.0

Fig. 10: LLM Model Consistency with V2V Communication support (GPT-4 Turbo)

To evaluate the consistency of the model decision, we ran the same prompt 20 times. The consistency was calculated by mapping 'FORWARD' = 1 and 'STOP' = 0. The resulting mean value of 1 for 'STOP' indicates that the output is having greater consistency with the expected response. The consistency of GPT-4 Turbo's responses with V2V support is shown in Fig. 10, which confirms that communication agent inputs help LLMs generate more accurate decisions in critical traffic environments.

V. CONCLUSION

We presented an approach for utilizing Large Language Models (LLMs) to enhance reasoning capabilities in uncertain traffic situations within a multi-agent framework for autonomous vehicles. We have investigated LLMs from Hugging Face, LM Studio, as well as major commercial services such as Claude, GPT-4 Turbo, and Falcon. Their responses to basic traffic scenarios were analyzed, revealing varying levels of consistency and accuracy. These results revealed both the promise and the current limitations of this approach, highlighting the need for improved consistency in the model responses. The proposed approach aims at the realization of adaptive autonomous systems capable of human-like reasoning in unpredictable situations. By integrating contextual data derived from V2V communication with LLM reasoning, we can have further investigations to bridge the critical gap in decision-making accuracy for autonomous vehicles.

REFERENCES

[1] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, *et al.*, "A survey on evaluation of large language models," *ACM transactions on intelligent systems and technology*, vol. 15, no. 3, pp. 1–45, 2024.

[2] A. Iftekhar and N. Fukuta, "Toward an efficient and resilient electric vehicle charging place allocation using multiagent approach," in *2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pp. 663–664, 2024.

[3] S. S. Duddela and N. Fukuta, "Toward an acceptable dynamic allocation considering fairness on autonomous car parking," in *2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pp. 286–291, 2024.

[4] J. Cho and J. Heo, "The more you know, the more you can trust: Drivers' understanding of the advanced driver assistance system," in *HCI in Mobility, Transport, and Automotive Systems. Automated Driving and In-Vehicle Experience Design: Second International Conference, MobiTAS 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part I 22*, pp. 230–248, Springer, 2020.

[5] M. Raza, "Autonomous vehicles: levels, technologies, impacts and concerns," *International Journal of Applied Engineering Research*, vol. 13, no. 16, pp. 12710–12714, 2018.

[6] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[7] D. Fu, X. Li, L. Wen, M. Dou, P. Cai, B. Shi, and Y. Qiao, "Drive like a human: Rethinking autonomous driving with large language models," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 910–919, 2024.

[8] T. Singh, E. van Hassel, A. Sheorey, and M. Alirezaei, "A systematic approach for creation of sotif's unknown unsafe scenarios: An optimization based method," tech. rep., SAE Technical Paper, 2024.

[9] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning*, pp. 38087–38099, PMLR, 2023.

(a) Output from DeepSeek-V3

(b) Output from GPT-4 turbo

(c) Output from Claude 3.7 Sonnet

(d) Output from Falcon3 7B

(e) Output from google/flan-t5-large

(f) Output from OpenHermes-2.5-Mistral-7B

Fig. 11: Responses from different LLMs

Acknowledgements

Alhamdulillah. In the name of Allah, the Most Gracious and the Most Merciful.

Appendix A

Simulation Source Code

A.1 Traffic Light Recognition Simulation: Autonomous Vehicle Approaching Green Traffic Light

```python
import time
class RoadCollisionSimulator:
    def __init__(self, road_length):
        self.road = [['_' for _ in range(road_length)] for _
            in range(3)]

        self.av_position = 0
        self.human_position = 0
        self.traffic_light_position = road_length // 2 + 2

        self.road[0][self.av_position] = 'AV'
        self.road[0][self.traffic_light_position] = 'TL:
            green'
        self.road[2][self.human_position] = 'H'

        self.road[1] = ['-' if i % 2 == 0 else '_' for i in
            range(road_length)]

        self.output_file = open("TrafficLightRecog.txt", "w"
            )

    def write_road_to_file(self, description):
        self.output_file.write(description + "\n")
        for lane in self.road:
            road_line = ''.join(lane)
            self.output_file.write(road_line + "\n")
        self.output_file.write('-' * len(self.road[0]) + "\n
            ")
        self.output_file.flush()

    def move_av(self):
```

```python
27                if self.av_position < self.traffic_light_position -
                      2:
28                    self.road[0][self.av_position] = '␣'
29                    self.av_position += 1
30                    self.road[0][self.av_position] = 'AV'
31
32        def move_human(self):
33            if self.human_position < len(self.road[2]) - 1:
34                self.road[2][self.human_position] = '␣'
35                self.human_position += 1
36                self.road[2][self.human_position] = 'H'
37
38        def simulate(self):
39            self.write_road_to_file("AV␣=␣Autonomous␣Vehicle,\nH
                  ␣=␣Human,\nTL:green␣=␣Traffic␣Light␣(Green)\n")
40            self.write_road_to_file("Initial␣Position:")
41
42            while self.av_position < self.traffic_light_position
                  - 2:
43                self.move_av()
44                self.move_human()
45                time.sleep(0.3)
46
47            self.write_road_to_file("Position␣Right␣Before␣TL:")
48
49            self.move_av()
50            self.move_human()
51
52            self.write_road_to_file("Final␣Position:")
53
54            self.output_file.write("Simulation␣ended.\n")
55            self.output_file.close()
56
57    road_length = 20
58
59    simulator = RoadCollisionSimulator(road_length)
60
61    simulator.simulate()
```

CODE LISTING A.1: Traffic Light Recognition Simulation -
Autonomous Vehicle Approaching Green Traffic Light

A.2 Stop Sign Detection Simulation: Autonomous Vehicle Approaching Stop Sign on Partial Road

```python
import time
class RoadCollisionSimulator:
    def __init__(self, road_length):
        self.road = [['_' for _ in range(road_length)] for _
            in range(2)]

        self.av_position = 0
        self.stop_sign_position = road_length // 2

        self.road[0][self.av_position] = 'AV'
        self.road[0][self.stop_sign_position] = 'S'

        self.road[1] = ['-' if i % 2 == 0 else '_' for i in
            range(road_length)]

        self.output_file = open("StopSign.txt", "w")

    def write_road_to_file(self, description):
        self.output_file.write(description + "\n")
        for lane in self.road:
            road_line = ''.join(lane)
            self.output_file.write(road_line + "\n")
        self.output_file.write('-' * len(self.road[0]) + "\n
            ")
        self.output_file.flush()

    def move_av(self):
        if self.av_position < self.stop_sign_position - 1:
            self.road[0][self.av_position] = '_'
            self.av_position += 1
            self.road[0][self.av_position] = 'AV'

    def simulate(self):
        self.write_road_to_file("AV_=_Autonomous_Vehicles,\
            nS_=_Stop_Sign_of_a_partial_road,\nOther_lane_is_
            open.\n")

        self.write_road_to_file("Initial_Position:")

        while self.av_position < self.stop_sign_position -
            1:
            self.move_av()
            time.sleep(0.3)

        self.write_road_to_file("Position_Right_Before_STOP_
            sign:")
```

```
40
41          self.output_file.write("Simulation ended.\n")
42          self.output_file.close()
43
44   road_length = 15
45
46   simulator = RoadCollisionSimulator(road_length)
47
48   simulator.simulate()
```

CODE LISTING A.2: Stop Sign Detection Simulation - Autonomous
Vehicle Approaching Stop Sign on Partial Road

A.3   Sudden Human Crossing Detection Simulation: Autonomous Vehicle Encountering Unexpected Pedestrian

```
1   import time
2   class RoadCollisionSimulator:
3       def __init__(self, road_length):
4           self.road = [['␣' for _ in range(road_length)] for _
                in range(2)]
5
6           self.av_position = 0
7           self.human_position = road_length // 2 + 1
8
9           self.road[0][self.av_position] = 'AV'
10
11          self.road[1] = ['-' if i % 2 == 0 else '␣' for i in
                range(road_length)]
12
13          self.output_file = open("Suddenhumancrossing.txt", "
                w")
14
15      def write_road_to_file(self, description):
16          self.output_file.write(description + "\n")
17          for lane in self.road:
18              road_line = ''.join(lane)
19              self.output_file.write(road_line + "\n")
20          self.output_file.write('-' * len(self.road[0]) + "\n
                ")
21          self.output_file.flush()
22
23      def move_av(self):
```

```
24          if self.av_position < len(self.road[0]) - 1:
25              self.road[0][self.av_position] = '_'
26              self.av_position += 1
27              if self.road[0][self.av_position] != 'H':
28                  self.road[0][self.av_position] = 'AV'
29
30      def add_human(self):
31          self.road[0][self.human_position] = 'H'
32
33      def simulate(self):
34          self.write_road_to_file("H_=_Human,\nAV_=_Autonomous
                _Vehicles,\n")
35
36          self.write_road_to_file("Initial_Position:")
37
38          while self.av_position < self.human_position - 2:
39              self.move_av()
40              time.sleep(0.3)
41
42          self.add_human()
43
44          self.write_road_to_file("Position_Right_Before_H:")
45
46          self.output_file.write("Simulation_ended.\n")
47          self.output_file.close()
48
49  road_length = 15
50
51  simulator = RoadCollisionSimulator(road_length)
52
53  simulator.simulate()
```

CODE LISTING A.3: Sudden Human Crossing Detection Simulation - Autonomous Vehicle Encountering Unexpected Pedestrian

A.4 Stop sign view blocked from the following vehicle: Autonomous Truck and Car Approaching Stop Sign with Possible Road Collision Simulation

```
1  import time
2  class RoadCollisionSimulator:
```

```python
        def __init__(self, road_length):
            self.road = [['␣' for _ in range(road_length)] for _
                in range(2)]
            self.road[0][road_length // 2] = 'S'

            self.av1_position = road_length // 2 - 2
            self.av2_position = 0

            self.road[0][self.av1_position] = 'AV1'
            self.road[0][self.av2_position] = 'AV2'

            self.collision_occurred = False

            self.road[1] = ['-' if i % 2 == 0 else '␣' for i in
                range(road_length)]

            self.output_file = open("ExtendedScenario.txt", "w")

        def write_road_to_file(self, description):
            self.output_file.write(description + "\n")
            for lane in self.road:
                road_line = ''.join(lane)
                self.output_file.write(road_line + "\n")
            self.output_file.write('-' * len(self.road[0]) + "\n
                ")
            self.output_file.flush()

        def detect_collision(self):
            if self.road[0][self.av1_position] == 'S':
                self.road[0][self.av1_position] = 'X'
                self.collision_occurred = True
                return True
            return False

        def move_av2_car(self):
            if not self.collision_occurred and self.av2_position
                < self.av1_position - 1:
                self.road[0][self.av2_position] = '␣'
                self.av2_position += 1
                if self.av2_position < len(self.road[0]):
                    self.road[0][self.av2_position] = 'AV2'

        def move_av1_truck(self):
            if not self.collision_occurred:
                self.road[0][self.av1_position] = '␣'
                self.av1_position += 1
                if self.av1_position < len(self.road[0]):
```

```python
                    if self.road[0][self.av1_position] == 'S':
                        pass
                    else:
                        self.road[0][self.av1_position] = 'AV1'

    def simulate(self):
        self.write_road_to_file("AV2_=_Autonomous_Car,\nAV1_
            =_Autonomous_Truck,\nS_=_Stop_Sign\n")
        self.write_road_to_file("Initial_Position:")

        stop_sign_position = len(self.road[0]) // 2
        while self.av1_position < stop_sign_position - 1:
            self.move_av2_car()
            self.move_av1_truck()
            time.sleep(0.5)

        self.write_road_to_file("Position_Right_Before_STOP:
            ")

        self.move_av2_car()
        self.move_av1_truck()

        self.detect_collision()

        self.write_road_to_file("Final_Position:")

        self.output_file.write("Simulation_ended.\n")
        self.output_file.close()

road_length = 20

simulator = RoadCollisionSimulator(road_length)

simulator.simulate()
```

CODE LISTING A.4: Multi-Vehicle Collision Simulation - Autonomous Truck and Car Approaching Stop Sign with Possible Road Collision Simulation

A.5 Multi-Agent Coordination Simulation: Three Autonomous Vehicles Coordinating Movement Around Human Obstacle

```python
import time
```

```python
class RoadCollisionSimulator:
    def __init__(self, road_length):
        self.road = [['_' for _ in range(road_length)] for _
            in range(3)]

        self.av1_position = 0
        self.human_position = road_length // 2 + 2
        self.av2_position = 0
        self.av3_position = 0

        self.road[0][self.av1_position] = 'AV1'
        self.road[0][self.human_position] = 'H'
        self.road[2][self.av2_position] = 'AV2'

        self.road[1] = ['-' if i % 2 == 0 else '_' for i in
            range(road_length)]

        self.output_file = open("NextStep.txt", "w")

    def write_road_to_file(self, description):
        self.output_file.write(description + "\n")
        for lane in self.road:
            road_line = ''.join(lane)
            self.output_file.write(road_line + "\n")
        self.output_file.write('-' * len(self.road[0]) + "\n
            ")
        self.output_file.flush()

    def move_av1(self):
        if self.av1_position < len(self.road[0]) - 1:
            self.road[0][self.av1_position] = '_'
            self.av1_position += 1
            if self.road[0][self.av1_position] != 'H':
                self.road[0][self.av1_position] = 'AV1'

    def move_av2(self):
        if self.av2_position < len(self.road[2]) - 1:
            self.road[2][self.av2_position] = '_'
            self.av2_position += 1
            self.road[2][self.av2_position] = 'AV2'

    def add_av3(self):
        self.av3_position = 0
        self.road[0][self.av3_position] = 'AV3'

    def simulate(self):
```

```
45        self.write_road_to_file("H␣=␣Human,\nAV1,AV2,AV3␣=␣
              Autonomous␣Vehicles,\n")
46        self.write_road_to_file("Initial␣Position:")
47
48        while self.av1_position < self.human_position - 2:
49            self.move_av1()
50            self.move_av2()
51            time.sleep(0.3)
52
53        self.write_road_to_file("Position␣Right␣Before␣Human
              :")
54
55        self.move_av1()
56        self.move_av2()
57        self.add_av3()
58
59        self.write_road_to_file("Final␣Position:")
60
61        self.output_file.write("Simulation␣ended.\n")
62        self.output_file.close()
63
64    road_length = 20
65
66    simulator = RoadCollisionSimulator(road_length)
67
68    simulator.simulate()
```

CODE LISTING A.5: Multi-Agent Coordination Simulation - Three Autonomous Vehicles Coordinating Movement Around Human Obstacle

A.6 Vehicle-to-Vehicle (V2V) Communication System: Multi-Agent Information Sharing and Status Broadcasting

```
1    import json
2    from dataclasses import dataclass, asdict
3    from typing import Dict, List, Any
4    import random
5    import time
6    import math
7
8    @dataclass
9    class VehicleStatus:
```

```python
10        vehicle_id: str
11        timestamp: float
12
13        obstacle_presence: bool
14        obstacle_type: str
15        obstacle_proximity: float
16
17        traffic_density: str
18        road_conditions: str
19
20        current_speed: float
21        direction: str
22
23        intended_action: str
24
25        x_coordinate: float
26        y_coordinate: float
27
28    class V2VCommunicationSystem:
29        def __init__(self, communication_radius: float = 50.0):
30            self.communication_radius = communication_radius
31            self.vehicles = {}
32
33        def add_vehicle(self, vehicle_id: str, x: float, y:
             float):
34            self.vehicles[vehicle_id] = {
35                'position': (x, y),
36                'status': None
37            }
38
39        def calculate_distance(self, vehicle1_id: str,
             vehicle2_id: str) -> float:
40            pos1 = self.vehicles[vehicle1_id]['position']
41            pos2 = self.vehicles[vehicle2_id]['position']
42            return math.sqrt((pos1[0] - pos2[0])**2 + (pos1[1] -
                 pos2[1])**2)
43
44        def get_nearby_vehicles(self, vehicle_id: str) -> List[
             str]:
45            nearby_vehicles = []
46            for other_id in self.vehicles.keys():
47                if other_id != vehicle_id:
48                    distance = self.calculate_distance(
                         vehicle_id, other_id)
49                    if distance <= self.communication_radius:
50                        nearby_vehicles.append(other_id)
51            return nearby_vehicles
```

```python
class Vehicle2CommunicationAgent:
    def __init__(self, v2v_system: V2VCommunicationSystem):
        self.vehicle_id = "002"
        self.v2v_system = v2v_system

        self.v2v_system.add_vehicle(self.vehicle_id, 10, 10)

        self.received_statuses = {}
        self.nearby_vehicles = []

        self.output_file = open("v2v_communication_output.
            txt", "w")

    def write_to_file(self, message):
        self.output_file.write(message + "\n")
        self.output_file.flush()

    def send_own_status(self):
        self.write_to_file("\n Vehicle 2 Communication Agent
            : Sending Own Status ")

        self.nearby_vehicles = self.v2v_system.
            get_nearby_vehicles(self.vehicle_id)
        self.write_to_file(f"Nearby Vehicles: {self.
            nearby_vehicles}")

        vehicle_2_status = VehicleStatus(
            vehicle_id="002",
            timestamp=time.time(),

            obstacle_presence=True,
            obstacle_type="stop_sign",
            obstacle_proximity=13.123529423503923,

            traffic_density="heavy",
            road_conditions=random.choice(['normal', 'wet',
                'construction', 'icy']),

            current_speed=random.uniform(0, 60),
            direction='forward',

            intended_action='stop',

            x_coordinate=10,
            y_coordinate=10
        )
```

```
 94
 95            self.v2v_system.vehicles[self.vehicle_id]['status']
                   = asdict(vehicle_2_status)
 96
 97            self.write_to_file("Vehicle_2_Status_Details:")
 98            self.write_to_file(f"_Vehicle_ID:_{vehicle_2_status.
                   vehicle_id}")
 99            self.write_to_file(f"_Obstacle_Presence:_{
                   vehicle_2_status.obstacle_presence}")
100            self.write_to_file(f"_Obstacle_Type:_{
                   vehicle_2_status.obstacle_type}")
101            self.write_to_file(f"_Obstacle_Proximity:_{
                   vehicle_2_status.obstacle_proximity}_meters")
102            self.write_to_file(f"_Traffic_Density:_{
                   vehicle_2_status.traffic_density}")
103            self.write_to_file(f"_Intended_Action:_{
                   vehicle_2_status.intended_action}")
104
105            return vehicle_2_status
106
107        def receive_vehicle_status(self, sender_vehicle_id: str)
                   :
108            self.write_to_file(f"\n_Vehicle_2_Communication_
                   Agent:_Receiving_Status_from_Vehicle_{
                   sender_vehicle_id}_")
109
110            sender_status = self.v2v_system.vehicles[
                   sender_vehicle_id]['status']
111
112            self.write_to_file("Received_Status_Details:")
113            self.write_to_file(f"_Vehicle_ID:_{sender_status['
                   vehicle_id']}")
114            self.write_to_file(f"_Obstacle_Presence:_{
                   sender_status['obstacle_presence']}")
115            self.write_to_file(f"_Obstacle_Type:_{sender_status.
                   get('obstacle_type',_'N/A')}")
116            self.write_to_file(f"_Obstacle_Proximity:_{
                   sender_status['obstacle_proximity']}_meters")
117            self.write_to_file(f"_Traffic_Density:_{
                   sender_status['traffic_density']}")
118            self.write_to_file(f"_Intended_Action:_{
                   sender_status['intended_action']}")
119
120            self.received_statuses[sender_vehicle_id] =
                   sender_status
121
122        def close_file(self):
```

```
123              self.output_file.close()
124
125      def simulate_v2v_communication():
126          v2v_system = V2VCommunicationSystem(communication_radius
                 =50.0)
127
128          v2v_system.add_vehicle("001", 5, 5)
129          v2v_system.add_vehicle("003", 20, 20)
130
131          vehicle_2_comm = Vehicle2CommunicationAgent(v2v_system)
132
133          vehicle_2_status = vehicle_2_comm.send_own_status()
134
135          v2v_system.vehicles["001"]['status'] = {
136              'vehicle_id': "001",
137              'timestamp': time.time(),
138              'obstacle_presence': random.choice([True, False]),
139              'obstacle_type': random.choice(['traffic_light', '
                   pedestrian', 'none']),
140              'obstacle_proximity': random.uniform(0, 15),
141              'traffic_density': random.choice(['light', 'moderate
                   ', 'heavy']),
142              'road_conditions': random.choice(['normal', 'wet', '
                   construction', 'icy']),
143              'current_speed': random.uniform(0, 120),
144              'direction': random.choice(['forward', 'stationary'
                   ]),
145              'intended_action': random.choice(['forward', 'stop',
                   'turn_left', 'turn_right']),
146              'x_coordinate': 5,
147              'y_coordinate': 5
148          }
149
150          v2v_system.vehicles["003"]['status'] = {
151              'vehicle_id': "003",
152              'timestamp': time.time(),
153              'obstacle_presence': random.choice([True, False]),
154              'obstacle_type': random.choice(['traffic_light', '
                   pedestrian', 'none']),
155              'obstacle_proximity': random.uniform(0, 15),
156              'traffic_density': random.choice(['light', 'moderate
                   ', 'heavy']),
157              'road_conditions': random.choice(['normal', 'wet', '
                   construction', 'icy']),
158              'current_speed': random.uniform(0, 120),
159              'direction': random.choice(['forward', 'stationary'
                   ]),
```

```
160        'intended_action': random.choice(['forward', 'stop',
              'turn_left', 'turn_right']),
161        'x_coordinate': 20,
162        'y_coordinate': 20
163    }

164
165    vehicle_2_comm.receive_vehicle_status("001")
166    vehicle_2_comm.receive_vehicle_status("003")

167
168    vehicle_2_comm.write_to_file("\n-V2V_Communication_
           Simulation_Completed-")

169
170    vehicle_2_comm.close_file()

171
172 if __name__ == "__main__":
173    simulate_v2v_communication()
```

CODE LISTING A.6: Vehicle-to-Vehicle (V2V) Communication
System - Multi-Agent Information Sharing and Status Broadcasting

A.7 LLM Decision Consistency Analysis: Statistical Evaluation of Language Model Decision Patterns in Autonomous Driving Scenarios

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  # Example responses from the LLM (1 for FORWARD, 0 for STOP)
4  responses = [1] * 19 # 19 FORWARD + 1 FORWARD (counted as 1)
5  mean_response = np.mean(responses)
6  variance_response = np.var(responses)
7  std_dev = np.std(responses)
8  plt.figure(figsize=(8, 5))
9  plt.bar(['Mean'], [mean_response], color='skyblue', label=f'
       Mean:_{mean_response:.2f}')
10 plt.errorbar(['Mean'], [mean_response], yerr=[std_dev],
11             fmt='none', ecolor='red', capsize=10, label=f'
                 Std_Dev:_{std_dev:.2f}')
12 plt.title('LLM_Decision_Tendency_(FORWARD=1,_STOP=0)', pad
       =20)
13 plt.ylim(0, 1.2)
14 plt.ylabel('Response_Value')
15 plt.legend()
16 plt.grid(axis='y', alpha=0.3)
```

```
17    plt.text(0, mean_response+0.05, f'Variance:_{
         variance_response:.2f}',
18              ha='center', va='bottom', fontsize=10)
19 plt.show()
```

A.8 Next Step: Extended simulations of multi-lane roads with sudden human crossing

```
1    import time
2 class RoadCollisionSimulator:
3     def __init__(self, road_length):
4         self.road = [['_' for _ in range(road_length)] for _ in
             range(3)]
5
6         self.av1_position = 0
7         self.human_position = road_length // 2 + 2
8         self.av2_position = 0
9         self.av3_position = 0
10
11        self.road[0][self.av1_position] = 'AV1'
12        self.road[0][self.human_position] = 'H'
13        self.road[2][self.av2_position] = 'AV2'
14
15        self.road[1] = ['-' if i % 2 == 0 else '_' for i in
             range(road_length)]
16        self.output_file = open("NextStep.txt", "w")
17    def write_road_to_file(self, description):
18        self.output_file.write(description + "\n")
19        for lane in self.road:
20            road_line = ''.join(lane)
21            self.output_file.write(road_line + "\n")
22        self.output_file.write('-' * len(self.road[0]) + "\n")
23        self.output_file.flush()
24    def move_av1(self):
25        if self.av1_position < len(self.road[0]) - 1:
26            self.road[0][self.av1_position] = '_'
27            self.av1_position += 1
28            if self.road[0][self.av1_position] != 'H':
29                self.road[0][self.av1_position] = 'AV1'
30    def move_av2(self):
31        if self.av2_position < len(self.road[2]) - 1:
32            self.road[2][self.av2_position] = '_'
33            self.av2_position += 1
34            self.road[2][self.av2_position] = 'AV2'
```

```
35    def add_av3(self):
36        self.av3_position = 0
37        self.road[0][self.av3_position] = 'AV3'
38    def simulate(self):
39        self.write_road_to_file("H_=_Human,\nAV1,AV2,AV3_=_
              Autonomous_Vehicles,\n")
40        self.write_road_to_file("Initial_Position:")
41        while self.av1_position < self.human_position - 2:
42            self.move_av1()
43            self.move_av2()
44            time.sleep(0.3)
45        self.write_road_to_file("Position_Right_Before_Human:")
46        self.move_av1()
47        self.move_av2()
48        self.add_av3()
49        self.write_road_to_file("Final_Position:")
50        self.output_file.write("Simulation_ended.\n")
51        self.output_file.close()
52 road_length = 20
53 simulator = RoadCollisionSimulator(road_length)
54 simulator.simulate()
```

CODE LISTING A.7: Next Step: Expanded simulations of multi-lane roads with sudden human crossing.

A.9   Next Step: V2V data exchange among autonomous vehicles

```
1  import json
2  from dataclasses import dataclass, asdict
3  from typing import Dict, List, Any
4  import random
5  import time
6  import math
7  @dataclass
8  class VehicleStatus:
9      vehicle_id: str
10     timestamp: float
11
12     obstacle_presence: bool
13     obstacle_type: str
14     obstacle_proximity: float
15
16     traffic_density: str
17     road_conditions: str
```

```python
18      lane_position: str

20      current_speed: float
21      direction: str

23      intended_action: str

25      x_coordinate: float
26      y_coordinate: float

28      emergency_status: bool
29      human_detected: bool
30  class V2VCommunicationSystem:
31      def __init__(self, communication_radius: float = 100.0):
32          self.communication_radius = communication_radius
33          self.vehicles = {}
34      def add_vehicle(self, vehicle_id: str, x: float, y: float):
35          self.vehicles[vehicle_id] = {
36              'position': (x, y),
37              'status': None
38          }
39      def calculate_distance(self, vehicle1_id: str, vehicle2_id:
            str) -> float:
40          pos1 = self.vehicles[vehicle1_id]['position']
41          pos2 = self.vehicles[vehicle2_id]['position']
42          return math.sqrt((pos1[0] - pos2[0])**2 + (pos1[1] -
                pos2[1])**2)
43      def get_nearby_vehicles(self, vehicle_id: str) -> List[str]:
44          nearby_vehicles = []
45          for other_id in self.vehicles.keys():
46              if other_id != vehicle_id:
47                  distance = self.calculate_distance(vehicle_id,
                        other_id)
48                  if distance <= self.communication_radius:
49                      nearby_vehicles.append(other_id)
50          return nearby_vehicles
51  class AV2CommunicationAgent:
52      def __init__(self, v2v_system: V2VCommunicationSystem):
53          self.vehicle_id = "AV2"
54          self.v2v_system = v2v_system

56          self.v2v_system.add_vehicle(self.vehicle_id, 15, 10)

58          self.received_statuses = {}
59          self.nearby_vehicles = []

61          self.output_file = open("V2VforNextstep.txt", "w")
```

```python
def write_to_file(self, message):
    self.output_file.write(message + "\n")
    self.output_file.flush()
def send_own_status(self):
    self.write_to_file("\n_AV2_Communication_Agent:_
        EMERGENCY_-_Human_Crossing_Detected_")

    self.nearby_vehicles = self.v2v_system.
        get_nearby_vehicles(self.vehicle_id)
    self.write_to_file(f"Broadcasting_to_Nearby_Vehicles:_{
        self.nearby_vehicles}")

    av2_status = VehicleStatus(
        vehicle_id="AV2",
        timestamp=time.time(),

        obstacle_presence=True,
        obstacle_type="human_pedestrian",
        obstacle_proximity=8.5,

        traffic_density="moderate",
        road_conditions="normal",
        lane_position="same_lane",

        current_speed=25.0,
        direction="forward",

        intended_action="emergency_brake",

        x_coordinate=15,
        y_coordinate=10,

        emergency_status=True,
        human_detected=True
    )

    self.v2v_system.vehicles[self.vehicle_id]['status'] =
        asdict(av2_status)

    self.write_to_file("AV2_Status_Details:")
    self.write_to_file(f"_Vehicle_ID:_{av2_status.vehicle_id
        }")
    self.write_to_file(f"_EMERGENCY_STATUS:_{av2_status.
        emergency_status}")
    self.write_to_file(f"_Human_Detected:_{av2_status.
        human_detected}")
```

```python
101            self.write_to_file(f" Obstacle Type: {av2_status.
                obstacle_type}")
102            self.write_to_file(f" Human Proximity: {av2_status.
                obstacle_proximity} meters")
103            self.write_to_file(f" Current Speed: {av2_status.
                current_speed} km/h")
104            self.write_to_file(f" Intended Action: {av2_status.
                intended_action}")
105            self.write_to_file(f" Lane Position: {av2_status.
                lane_position}")

107            return av2_status
108        def receive_vehicle_status(self, sender_vehicle_id: str):
109            self.write_to_file(f"\n AV2 Communication Agent: 
                Receiving Status from {sender_vehicle_id} ")

111            sender_status = self.v2v_system.vehicles[
                sender_vehicle_id]['status']

113            self.write_to_file("Received Status Details:")
114            self.write_to_file(f" Vehicle ID: {sender_status['
                vehicle_id']}")
115            self.write_to_file(f" Emergency Status: {sender_status.
                get('emergency_status', False)}")
116            self.write_to_file(f" Human Detected: {sender_status.get
                ('human_detected', False)}")
117            self.write_to_file(f" Obstacle Presence: {sender_status
                ['obstacle_presence']}")
118            self.write_to_file(f" Obstacle Type: {sender_status.get
                ('obstacle_type', 'N/A')}")
119            self.write_to_file(f" Obstacle Proximity: {sender_status
                ['obstacle_proximity']} meters")
120            self.write_to_file(f" Current Speed: {sender_status['
                current_speed']} km/h")
121            self.write_to_file(f" Intended Action: {sender_status['
                intended_action']}")
122            self.write_to_file(f" Lane Position: {sender_status['
                lane_position']}")

124            self.received_statuses[sender_vehicle_id] = 
                sender_status
125        def close_file(self):
126            self.output_file.close()
127    def simulate_human_crossing_scenario():
128        v2v_system = V2VCommunicationSystem(communication_radius
                =100.0)

```

```python
130    v2v_system.add_vehicle("AV1", 10, 10)

132    v2v_system.add_vehicle("AV3", 8, 15)

134    v2v_system.add_vehicle("AV4", 10, 25)

136    v2v_system.add_vehicle("AV5", 8, 20)

138    v2v_system.add_vehicle("AV6", 8, 25)

140    v2v_system.add_vehicle("AV7", 8, 30)

142    av2_comm = AV2CommunicationAgent(v2v_system)

144    av2_status = av2_comm.send_own_status()

146    v2v_system.vehicles["AV1"]['status'] = {
147        'vehicle_id': "AV1",
148        'timestamp': time.time(),
149        'obstacle_presence': True,
150        'obstacle_type': "human_pedestrian",
151        'obstacle_proximity': 23.0,
152        'traffic_density': "moderate",
153        'road_conditions': "normal",
154        'lane_position': "same_lane",
155        'current_speed': 0.0,
156        'direction': "stationary",
157        'intended_action': "emergency_brake",
158        'x_coordinate': 10,
159        'y_coordinate': 10,
160        'emergency_status': True,
161        'human_detected': True
162    }

164    v2v_system.vehicles["AV3"]['status'] = {
165        'vehicle_id': "AV3",
166        'timestamp': time.time(),
167        'obstacle_presence': True,
168        'obstacle_type': "human_pedestrian",
169        'obstacle_proximity': 12.0,
170        'traffic_density': "moderate",
171        'road_conditions': "normal",
172        'lane_position': "adjacent_left",
173        'current_speed': 15.0,
174        'direction': "forward",
175        'intended_action': "slow_down",
176        'x_coordinate': 8,
```

```python
177        'y_coordinate': 15,
178        'emergency_status': False,
179        'human_detected': True
180    }
181
182    v2v_system.vehicles["AV4"]['status'] = {
183        'vehicle_id': "AV4",
184        'timestamp': time.time(),
185        'obstacle_presence': True,
186        'obstacle_type': "human_pedestrian",
187        'obstacle_proximity': 18.7,
188        'traffic_density': "light",
189        'road_conditions': "normal",
190        'lane_position': "opposite_lane",
191        'current_speed': 20.0,
192        'direction': "forward",
193        'intended_action': "slow_down",
194        'x_coordinate': 10,
195        'y_coordinate': 25,
196        'emergency_status': False,
197        'human_detected': True
198    }
199
200    v2v_system.vehicles["AV5"]['status'] = {
201        'vehicle_id': "AV5",
202        'timestamp': time.time(),
203        'obstacle_presence': True,
204        'obstacle_type': "human_pedestrian",
205        'obstacle_proximity': 16.5,
206        'traffic_density': "moderate",
207        'road_conditions': "normal",
208        'lane_position': "adjacent_left",
209        'current_speed': 12.0,
210        'direction': "forward",
211        'intended_action': "slow_down",
212        'x_coordinate': 8,
213        'y_coordinate': 20,
214        'emergency_status': False,
215        'human_detected': True
216    }
217
218    v2v_system.vehicles["AV6"]['status'] = {
219        'vehicle_id': "AV6",
220        'timestamp': time.time(),
221        'obstacle_presence': True,
222        'obstacle_type': "human_pedestrian",
223        'obstacle_proximity': 21.2,
```

```python
224          'traffic_density': "moderate",
225          'road_conditions': "normal",
226          'lane_position': "adjacent_left",
227          'current_speed': 10.0,
228          'direction': "forward",
229          'intended_action': "slow_down",
230          'x_coordinate': 8,
231          'y_coordinate': 25,
232          'emergency_status': False,
233          'human_detected': True
234      }
235
236      v2v_system.vehicles["AV7"]['status'] = {
237          'vehicle_id': "AV7",
238          'timestamp': time.time(),
239          'obstacle_presence': True,
240          'obstacle_type': "human_pedestrian",
241          'obstacle_proximity': 25.8,
242          'traffic_density': "moderate",
243          'road_conditions': "normal",
244          'lane_position': "adjacent_left",
245          'current_speed': 8.0,
246          'direction': "forward",
247          'intended_action': "slow_down",
248          'x_coordinate': 8,
249          'y_coordinate': 30,
250          'emergency_status': False,
251          'human_detected': True
252      }
253
254      av2_comm.receive_vehicle_status("AV1")
255      av2_comm.receive_vehicle_status("AV3")
256      av2_comm.receive_vehicle_status("AV4")
257      av2_comm.receive_vehicle_status("AV5")
258      av2_comm.receive_vehicle_status("AV6")
259      av2_comm.receive_vehicle_status("AV7")
260
261      av2_comm.write_to_file("\n_Human_Crossing_Emergency_V2V_
         Communication_Completed_")
262
263      av2_comm.close_file()
264  if __name__ == "__main__":
265      simulate_human_crossing_scenario()
```

CODE LISTING A.8: Next Step: V2V data exchange among autonomous vehicles.

Appendix B

Quick Start Guide

This section will outline the setup, prerequisites, and execution instructions for each simulation code.

Table of Contents

Setup and Prerequisites

All simulation scripts are written in Python. **Python Installation:** Ensure that you have Python 3 installed on your system. You can download it from python.org.

```bash
# Navigate to the directory where you will store the scripts
cd your_project_directory
```

CODE LISTING B.1: Navigate to the directory (Bash)

Required Libraries: Most of the simulation scenarios used only built-in Python libraries ('time', 'json', 'math', 'random', 'dataclasses'). However, some scenarios required additional libraries for numerical operations, plotting, and advanced data handling.

To install all the necessary external libraries to run the simulation code, the following command can be used.

```
pip install numpy matplotlib
```

CODE LISTING B.2: Installing Required Python Libraries (Bash)

Scenario 1: Traffic Light Recognition Simulation

Filename: 'scenario1.py'

Description: Simulation of an autonomous vehicle approaching a green traffic light, along with a human-driven vehicle. The output file is 'TrafficLightRecog.txt'.

Extra Libraries: No additional libraries are needed.

How to Run:

```
python3 scenario1.py
```

CODE LISTING B.3: Running Scenario 1 (Bash)

Scenario 2: Stop Sign Detection Simulation - Autonomous Vehicle Approaching Stop Sign on Partial Road

Filename: 'scenario2.py'

Description: Simulation of an autonomous vehicle approaching a stop sign on a partial road. The output file is 'StopSign.txt'.

Extra Libraries: No additional libraries are needed.

How to Run:

```
python3 scenario2.py
```

CODE LISTING B.4: Running Scenario 2 (Bash)

Scenario 3: Sudden Human Crossing Detection Simulation - Autonomous Vehicle Encountering Unexpected Pedestrian

Filename: 'scenario3.py'

Description: Simulation of an autonomous vehicle encountering an unexpected pedestrian. The output file is 'Suddenhumancrossing.txt'.

Extra Libraries: No additional libraries are needed.
How to Run:

```
python3 scenario3.py
```

CODE LISTING B.5: Running Scenario 3 (Bash)

Scenario 4: Multi-Vehicle Collision Simulation - Autonomous Truck and Car Approaching Stop Sign with Possible Road Collision Simulation

Filename: 'scenario4.py'

Description: Simulation of an autonomous truck and an autonomous car approaching a stop sign. The car's perception is blocked by the truck, focusing on potential collision scenarios and their resolution. The output file is 'ExtendedScenario.txt'.
Extra Libraries: No additional libraries are needed.
How to Run:

```
python3 scenario4.py
```

CODE LISTING B.6: Running Scenario 4 (Bash)

Scenario 5: Multi-Agent Coordination Simulation - Three Autonomous Vehicles Coordinating Movement Around Human Obstacle

Filename: 'scenario5.py'

Description: Simulation of three autonomous vehicles coordinating their movement around a human obstacle. The output file is 'NextStep.txt'.
Extra Libraries: No additional libraries are needed.
How to Run:

```
python3 scenario5.py
```

CODE LISTING B.7: Running Scenario 5 (Bash)

Scenario 6: Vehicle-to-Vehicle (V2V) Communication System - Multi-Agent Information Sharing and Status Broadcasting

Filename: 'scenario6.py'

Description: *An illustration of the Communication Agent, information sharing, and status broadcast w*

Scenario 7: LLM Decision Consistency Analysis - Statistical Evaluation of Language Model Decision Patterns in Autonomous Driving Scenarios

Filename: 'scenario7.py'

Description: The output provides a basic consistent value of LLMs decision patterns, calculating the mean and variance.
Extra Libraries: 'numpy' 'matplotlib'
How to Run:

```
1 python3 scenario7.py
```

CODE LISTING B.8: Running Scenario 7 (Bash)

Scenario 8: Next Step: Extended simulations of multi-lane roads with sudden human crossing

Filename: 'scenario8.py'

Description: An extended version of Scenario 5, focusing on a more real world traffic situation. The output file is 'NextStep.txt'.
Extra Libraries: No additional libraries are needed.
How to Run:

```
1 python3 scenario8.py
```

CODE LISTING B.9: Running Scenario 8 (Bash)

Scenario 9: Next Step: V2V data exchange among autonomous vehicles.

Filename: 'scenario9.py'

Description: This scenario extends V2V communication and basic road simulation to visualize a more complex real-world traffic situation involving multiple autonomous vehicles dynamically sharing huge information within a communication radius. The output file is 'V2VforNextstep.txt'.
Extra Libraries: No additional libraries are needed.
How to Run:

```
1 python3 scenario9.py
```

CODE LISTING B.10: Running Scenario 9 (Bash)