Learning Autonomous Driving Tasks via Human Feedbacks with Large Language Models

Yunsheng Ma[1], Xu Cao[2], Wenqian Ye[3], Can Cui[1], Kai Mei[4], Ziran Wang[1],
[1]Purdue University, [2]UIUC, [3]University of Virginia, [4]Rutgers University
Correspondence: ziran@purdue.edu

Abstract

Traditional autonomous driving systems have mainly focused on making driving decisions without human interaction, overlooking human-like decision-making and human preference required in complex traffic scenarios. To bridge this gap, we introduce a novel framework leveraging Large Language Models (LLMs) for learning human-centered driving decisions from diverse simulation scenarios and environments that incorporate human feedback. Our contributions include a GPT-4-based programming planner that integrates seamlessly with the existing CARLA simulator to understand traffic scenes and react to human instructions. Specifically, we build a human-guided learning pipeline that incorporates human driver feedback directly into the learning process and stores optimal driving programming policy using Retrieval Augmented Generation (RAG). Impressively, our programming planner, with only 50 saved code snippets, can match the performance of baseline extensively trained reinforcement learning (RL) models. Our paper highlights the potential of an LLM-powered shared-autonomy system, pushing the frontier of autonomous driving system development to be more interactive and intuitive.

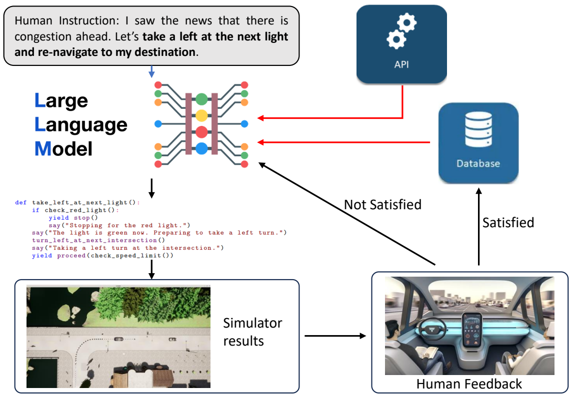

Figure 1: Schematic view of human feedback learning in the CARLA autonomous driving simulation.

1 Introduction

Autonomous driving has the potential to revolutionize transportation and urban mobility and has been a primary focus of research and development for the past two decades (Cui et al., 2024b). Traditional development of autonomous driving has primarily focused on achieving perfect black box safe navigation, and the role of humans in the system is often minimized, due to the doubts about the feasibility and benefits of integrating human and autonomous driving systems collaboratively (Parasuraman and Riley, 1997; Andrew, 2003). However, the human-like decision is still an important factor in designing autonomous driving systems. There will be instances where autonomous vehicles must make a choice between undesirable outcomes, like deciding between harming pedestrians or endangering the vehicle and its occupants to protect the pedestrians (Bonnefon et al., 2016). Creating the human-centered decision-making algorithms that enable these vehicles to navigate such ethical dilemmas presents a significant challenge.

In recent months, the advancements in large language models (LLMs) have shown emerging capabilities in many aspects. One study (Wei et al., 2022a) introduced this concept and defined emergence as abilities that are "not present in smaller models but are present in larger models”. Experiments show that LLMs can tackle complicated language tasks on challenges benchmarks including TruthfulQA (Lin et al., 2022), Massive Multi-task Language Understanding (MMLU) (Hendrycks et al., 2021), and Word in Context (WiC) (Pilehvar and Camacho-Collados, 2019).

This advancement also offers new possibilities for redefining human-vehicle interaction and has inspired researchers to reconsider the human roles

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 4985–4995
November 12-16, 2024

within the autonomous driving ecosystem. For example, the "Drive as you speak" concept (Cui et al., 2024a) proposes a more intuitive, human-like interaction inside autonomous vehicles, enabling the vehicles to understand and respond to natural language commands. Moreover, the incorporation of reasoning and reflective modules in the DiLu framework (Wen et al., 2024) endows autonomous vehicles with decision-making processes that mirror human common-sense reasoning. These efforts show the potential of autonomous vehicle systems not just as independent units, but as interactive platforms that engage with users.

Essentially, an autonomous driving system is an AI-infused system (Amershi et al., 2019) that interfaces with users. Therefore, it is crucial to incorporate a human-centric approach in its design and operation. The Human-Computer Interactions (HCI) community has proposed principles, guidelines, and strategies for AI system interactions. These are becoming increasingly relevant in autonomous vehicle development.

Drawing from the design patterns suggested in the People + AI Guidebook (Google PAIR, 2019), such as "let users supervise automation" and "let users give feedback," as well as the "shared autonomy" concept, which views rich, effective communication as the most essential design element (Fridman, 2018), this paper focuses on two essential features as shown in Fig. 2: (1) to interact with users based on spontaneous user instructions, and (2) to adapt and learn from user feedback in natural language. Specifically, our contributions are summarized as follows:

- An LLM-based programming planner is developed to serve as a plug-in module for classical autonomous driving pipelines to enable them to understand traffic scenes and follow human instructions.

- A human-guided learning pipeline is proposed for the programming planner to allow it to continuously learn from human feedback.

- The experiment results from the CARLA autonomous driving simulation show that the proposed programming planner, using only 50 saved code snippets, matches the performance of reinforcement learning (RL) methods.

2 Related Work

2.1 Human-Centered Autonomous Driving

Human-centered autonomous driving prioritizes the demand for the safety and comfort of street stakeholders in the system design (Xing et al., 2021). Unlike traditional autonomous driving technologies, which primarily focus on the efficiency and performance of the vehicle itself, human-centered autonomous driving emphasizes integrating user feedback into the driving planning and control to create more intuitive, trustworthy, and user-friendly autonomous driving experiences (Fridman, 2018). Previous research has demonstrated that language models can facilitate smoother interactions by answering questions between humans and vehicles (Deruyttere et al., 2019). Nonetheless, these advancements are often constrained by the models' scale and performance, grappling with safety and efficiency challenges that are important for real-world application. Moreover, human-centered autonomous driving recognizes the critical role of interactions with other street stakeholders, not only language-based communication with drivers. To address these complexities, recent studies suggest leveraging large language models such as GPT-4 to understand and integrate key street stakeholders' information and user's verbal feedback (Fu et al., 2024b; Yang et al., 2024b; Cui et al., 2024b).

2.2 Reasoning with Large Language Models

Reasoning is a fundamental aspect of human intelligence. It is defined as a cognitive concept involving the use of evidence, arguments, and logic, playing a central role in intellectual activities such as problem solving, decision making, and critical thinking. Recently, LLMs have shown some emergent behaviors, including the ability to "reason". One approach to encourage LLMs to reason is using chain-of-thought (CoT) prompting (Wei et al., 2022b). CoT decomposes a complicated reasoning task, such as a multi-step math word problem into intermediate steps, and solves each before giving the final answer. Additionally, LLMs have demonstrated zero-shot reasoning abilities by introducing phrases like "Let's think step by step" before each answer (Kojima et al., 2022). Tree-of-Thoughts (Yao et al., 2023) further generalizes CoT by exploring multiple reasoning paths. A closely related previous study is Auto-CoT (Zhang et al., 2023), which leverages zero-shot CoT to generate diverse reasoning chains as demonstrations and samples

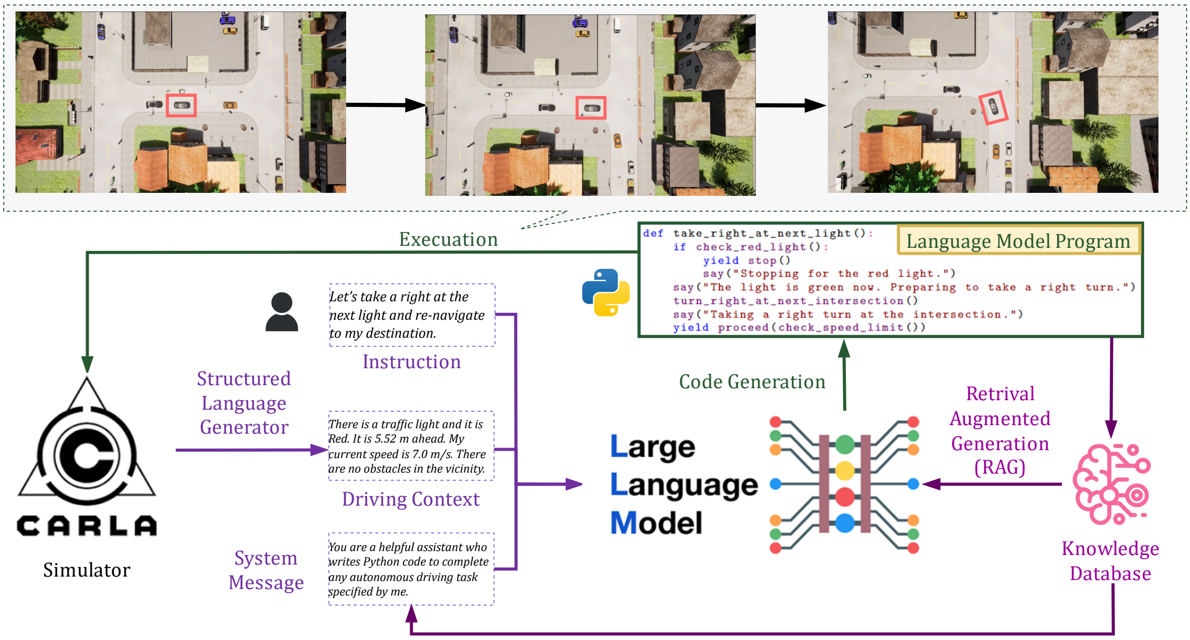

Figure 2: An overview of the proposed programming planner. Initially, the Large Language Model (LLM) is given a prompt composed of human instruction, driving context, and system message. It then conducts chain-of-thought reasoning to generate language model programs, which serve as driving policies. This policy code is executed in the CARLA simulator to complete the driving task as specified in the instructions. Verified code is then added to the knowledge database for future reference, utilizing Retrieval Augmented Generation.

them as examples for in-context learning. Concurrently, there also exist some pioneer autonomous driving works (Sima et al., 2023; Wen et al., 2024) focusing on the reasoning in the driving context.

2.3 Language-Guided Autonomous Driving

LLMs have shown remarkable potential in complicated scenarios such as driving scene understanding and decision-making (Rufus et al., 2021; Marcu et al., 2023; Mao et al., 2023; Malla et al., 2023; Nanwani et al., 2023; Cao et al., 2024). Recent advancements focus on building visual-language models to generate driving policy such as DriveMLM (Wang et al., 2023b) and DriveVLM (Tian et al., 2024). An equally crucial area of research is the development of language-guided closed-loop autonomous driving systems. These systems leverage multimodal sensor data from simulators, as demonstrated by LimSim++ (Fu et al., 2024a) and LMDrive (Shao et al., 2024). Additionally, RAG-Driver (Yuan et al., 2024) introduces a novel retrieval-augmented in-context learning approach, significantly enhancing the zero-shot generalization capabilities of driving LLMs.

Building on LLMs' success in NLP, there are also a great number of research works utilizing LLMs in the code understanding and generation domain (Liu et al., 2023). Furthermore, this success catalyzed the exploration of utilizing LLMs for robotics planning and control (Yang et al., 2024a), as evidenced by studies such as Voyager (Wang et al., 2023a) and Code as Policies (Liang et al., 2023). Recent advancements have seen LLMs generate executable code that interfaces with driving planning and control APIs, including LaMPilot (Ma et al., 2024), LangProp (Ishida et al., 2024). Despite these advancements, there still needs improvement in current works to embed LLMs in enhancing human-centered human-vehicle interactions and teaming within driving models.

3 Methodology

3.1 Overview

This section elaborates on the framework of our proposed programming planner. It integrates the world knowledge and reasoning capabilities of LLMs into the autonomous driving system that powers two essential shared autonomy features, namely following user instructions and integrating human feedback. As illustrated in Fig. 2, the programming planner has the following key designs: (1) Inspired by the 'code as policies' concept (Liang et al., 2023), we let LLMs generate

System Message ($\mathbf{X}_{\mathbf{sys}}^{\mathbf{0}}$)

You are a helpful assistant that writes python code to complete any autonomous driving task specified by me.

Here are the APIs you can use: {apis}

Here are some useful programs written with the provided APIs: {programs}

At each round of conversation, I will give you

Code from last round: ... Command: ... Context: ... Feedback: ...

You should then respond to me with

Explain (if applicable): Are there any steps missing in your plan? Why is the code not satisfactory?

Plan: How to complete the task step by step.

Code: 1) Write a function taking NO argument. 2) Only use the APIs that I provided. 3) Anything defined outside a function will be ignored, define all your variables inside your functions. 4) Name your function in a meaningful way (considering both command and context). 5) You should only write one function, which will be called by me. Do not write multiple functions. 6) When calling 'proceed()' or 'stop()' function, you should always 'yield' the return value of the function. 7) You can reuse the above useful programs. 8) Do not write infinite loops. 9) Do not use any constant from the context, as it may dynamically change. Call the APIs to get the latest values. 10) Always provide feedback to the user using the 'say' function.

You should only respond in the format as described below: {response_format}

Instruction ($\mathbf{X}_{\mathbf{q}}^{\mathbf{0}}$)

Take a left at the next light.

Driving Context ($\mathbf{X}_{\mathbf{c}}^{\mathbf{0}}$)

There is a traffic light and it is Red. It is 5.52 m ahead. My current speed is 7.0 m/s. There are no obstacles in the vicinity.

Chain-of-Thought Reasoning ($\mathbf{X}_{\mathbf{r}}^{\mathbf{0}}$)

1) First, check the status of the traffic light. If it's red, the vehicle should stop before the intersection. 2) Once the traffic light turns green, the vehicle should prepare to take a left turn at the intersection. 3) The vehicle will then turn left at the next intersection. 4) Throughout the process, provide feedback to the passenger about what actions the vehicle is planning to take.

Code as Policies ($\mathbf{X}_{\mathbf{p}}^{\mathbf{0}}$)

```
def take_left_at_next_light():
    if check_red_light():
        yield stop()
        say("Stopping for the red light.")
    say("The light is green now. Preparing to take a left turn.")
    turn_left_at_next_intersection()
    say("Taking a left turn at the intersection.")
    yield proceed(check_speed_limit())
```

Human Feedback ($\mathbf{X}_{\mathbf{f}}^{\mathbf{1}}$)

You should have slowed down a little bit while making turns.

Explanation ($\mathbf{X}_{\mathbf{r}}^{\mathbf{1}}$)

Reducing the speed slightly below the speed limit during a turn can enhance safety, especially in autonomous driving scenarios where precise control is crucial for both passenger comfort and safety.

Code as Policies ($\mathbf{X}_{\mathbf{p}}^{\mathbf{1}}$)

```
# ... [omitted]
# Reduce speed by a small factor (e.g., 90% of the speed limit) to ensure a safer turn
safe_turn_speed = check_speed_limit() * 0.9
# ... [omitted]
yield proceed(safe_turn_speed)
```

Human Feedback ($\mathbf{X}_{\mathbf{f}}^{\mathbf{2}}$)

The turning speed was too fast.

Explanation ($\mathbf{X}_{\mathbf{r}}^{\mathbf{2}}$)

It appears that the safe turning speed calculated as 90% of the speed limit was too fast for a safe left turn. To address this, we can further reduce the target speed for the turn to ensure safety.

Code as Policies ($\mathbf{X}_{\mathbf{p}}^{\mathbf{2}}$)

```
# ... [omitted]
# Reduce the speed to 70% of the speed limit for a safer turn
safe_turn_speed = check_speed_limit() * 0.7
# ... [omitted]
```

Table 1: This example illustrates the reasoning and code generation process of the proposed programming planner. The three blocks indicate that the generations occur in different iterations. Content with `blue` titles (chain-of-thought reasoning, explanation, and code as policies) is generated by LLMs. Content with `red` titles (instruction and human feedback) is provided by humans. The initial prompt includes three components: a system message ($\mathbf{X}_{\mathbf{sys}}$), human instruction ($\mathbf{X}_{\mathbf{q}}$), and the current driving context ($\mathbf{X}_{\mathbf{c}}$). Subsequent prompts also include human feedback ($\mathbf{X}_{\mathbf{f}}$) and the policy code ($\mathbf{X}_{\mathbf{p}}$) from the last round. For brevity, the system message, human instruction, driving context, and code from the last round are not included in subsequent prompts.

Method	Learning	Navigation	DS (↑)	RC (↑)	IP (↑)
Roach Expert (Zhang et al., 2021)	Reinforcement Learning	Waypoint	63.4	100.0	0.63
TCP Expert (Wu et al., 2022)	Reinforcement Learning	Waypoint	68.1	95.8	0.71
Programming Planner Expert (Ours)	0-Shot CoT	Human Instruction	0.00	48.7	0.00
Programming Planner Expert (Ours)	3-Shot CoT	Human Instruction	17.3	65.3	0.31
Programming Planner Expert (Ours)	Human-Feedback CoT	Human Instruction	65.4	95.1	0.68

Table 2: Driving performance of expert drivers on CARLA leaderboard testing routes. The driving score (DS) is calculated by multiplying the route completion percentage (RC) by the infraction penalty (IP). The "Navigation" column specifies the format of the navigation information. Here, "Waypoint" refers to a predetermined sequence of GPS-style coordinates and `enum` route instructions, while "Human Instruction" refers to natural language navigation commands like "turn right at the next light", which is a more challenging setting. ↑: Higher values are better.

language model programs (LMPs). These LMPs serve as the action space, leveraging the inherent potential of programs to symbolize actions that are both temporally extensive and hierarchically structured. For instance, a complex maneuver like overtaking can be programmatically split into a series of actions, including lane changes, acceleration, and possibly another lane change, composed with conditional logic. (2) We propose a human-guided learning pipeline to allow it to continuously learn from user's feedback.

As our experiments are based on the CARLA (Dosovitskiy et al., 2017) (the most widely used and realistic publicly available autonomous driving simulator), with Python as the programming language for code generation, We'll use specific terminologies related to this setup for clarity, but it's important to note that our approach isn't restricted to this setting.

The programming planner with the LLM backbone ℓ takes inputs including system messages ($\mathbf{X_{sys}}$), human instructions ($\mathbf{X_q}$), driving context ($\mathbf{X_c}$). It leverages CoT reasoning ($\mathbf{X_r}$) (Wei et al., 2022b) to write policy code ($\mathbf{X_p}$), which can be formulated as:

$$[\mathbf{X_r}, \mathbf{X_p}] = \ell([\mathbf{X_{sys}}, \mathbf{X_q}, \mathbf{X_c}]) \qquad (1)$$

An example to show the generation process is provided in Table 1. The system message ($\mathbf{X_{sys}}$) is a high-level instruction that guides the behavior of the model ℓ, which sets the overall objective for the generation. It also includes the API documentation (detailed in Section 3.2) and retrieved programs (detailed in Section 3.3). The human instructions ($\mathbf{X_q}$) are spontaneous commands from users. The driving context ($\mathbf{X_c}$), including information on current road conditions and other road users, forms the foundation for the planning process. This information can be derived from sensor data with perception modules or directly communicated by humans.

3.2 Programming Planner

LLMs have shown remarkable abilities in comprehending complex contexts and creating high-level plans by leveraging their inherent common-sense knowledge. However, despite their proficiency in high-level reasoning, LLMs face challenges when it comes to generating precise low-level control signals for autonomous vehicles. As highlighted by Cui et al. in their survey (Cui et al., 2024b), LLMs struggle with accurately interpreting spatial coordinates and creating detailed motion control commands. Moreover, the substantial size and autoregressive nature of these models often introduce significant latency in output generation, posing major obstacles for real-time objectives such as obstacle avoidance. This gap between high-level planning and low-level execution necessitates a novel approach to effectively harness the capabilities of LLMs in autonomous driving systems.

To address these limitations, we propose a plug-and-play programming planner that seamlessly integrates with existing autonomous driving systems, rather than attempting to replace them entirely. This design allows our approach to synergize the emerging capabilities of LLMs with the proven efficiency of classical autonomous driving algorithms. Instead of directly generating real-time, low-level control signals like steering or throttle changes, which LLMs are ill-suited for, our planner operates at a higher level of abstraction. It generates code snippets at lower frequencies, which serve as temporary policies to guide the strategic navigation of the ego vehicle based on user instructions. These policy code snippets are structured as Python generator functions, which gracefully unfold until the completion of each policy.

These generated code snippets involve calls to various functional primitives — specialized driving APIs based on classical planning and control algorithms. The API suite covers a broad spectrum of driving functions, including navigation functions like turning and lane changing, control functions such as waiting or proceeding, and perception functions which involve monitoring the immediate driving environment, including other vehicles, pedestrians, traffic signals, and other functions like checking the vehicle's current speed or providing user feedback. The API documentation, which includes a complete list of these functional primitives and their details, is provided in the Appendix. The same documentation is also included in the system message ($\mathbf{X_{sys}}$).

The API suite provides substantial flexibility in formulating driving policies. For example, Navigation APIs can be used to specify the vehicle's future trajectory, dynamically creating waypoints based on map data and classical planning algorithms. Control APIs, on the other hand, help with precise vehicle control by issuing momentary control signals. Perception APIs, like the `check_red_light` function in Table 1, can be used for identifying and tracking nearby traffic objects. This is vital for actions that react to the immediate driving context, enabling a closed-loop control policy.

3.3 Human-in-the-Loop Learning

An additional challenge arises from the phenomenon known as "LLM hallucination", where LLMs may make up facts as their knowledge is limited to the data they were trained on (Huang et al., 2023). This becomes problematic in the context of generating policy code, where LLMs might make up non-existent API functions. Additionally, the generation process of LLMs restricts that tokens generated early cannot be modified within the cycle. This constraint limits the models' ability to revise initial responses, potentially leading to suboptimal solutions.

To address these challenges, we introduce a human-in-the-loop learning approach, based on Retrieval Augmented Generation (RAG) (Lewis et al., 2020) to ground LLMs to generate responses to the input queries based on custom knowledge databases. The key process is that after a generated policy code ($\mathbf{X_p}$) has been executed, the human passenger will provide feedback ($\mathbf{X_f}$) in natural language, which, will be fed back into the LLM together with $\mathbf{X_p}$. This feedback loop enables continuous learning: positive feedback (i.e., the human is satisfied with the execution) results in the code ($\mathbf{X_p}$) being committed to the database to be retrieved for reuse in the future. Otherwise, the feedback will serve as guidance for another round of improvement. The new generation process can be formulated as:

$$[\mathbf{X_r}, \mathbf{X'_p}] = \ell([\mathbf{X_{sys}}, \mathbf{X_q}, \mathbf{X_c}, \mathbf{X_p}, \mathbf{X_f}]), \quad (2)$$

where $\mathbf{X'_p}$ is the improved code based on $\mathbf{X_p}$ and the human feedback $\mathbf{X_f}$ as a cue.

The database serves two purposes: firstly, as a repository for autonomous driving knowledge, offering insights into corner cases for continuous learning following the knowledge-driven autonomous driving paradigm (Wen et al., 2024); and secondly, as the database is built with personal feedback, it automatically incorporates personalized preferences in the interaction process. This methodology transforms the programming planner from a static, open-loop system into a dynamic, continuous learning framework, which we show in Section 4.4 that improved fundamentally compared to baselines.

4 Experiments and Results

4.1 Task Description

The experiments utilize the official testing routes from the CARLA (under CC-BY License) leaderboard 1.0 (CARLA, 2020) but with a more challenging setting. In each route, the agents start from a specific point. Originally, agents were provided with a series of GPS-style coordinates to drive to a destination (Chen et al., 2019; Prakash et al., 2021; Chitta et al., 2021; Toromanoff et al., 2020; Chen et al., 2021; Chekroun et al., 2023; Chen and Krähenbühl, 2022; Zhang et al., 2021; Wu et al., 2022). However, in our setting, agents are given only natural language navigation instructions based on their current position (e.g., turn left at the next intersection). As our study mainly focuses on reasoning and planning tasks, the driving context information is crafted using a structured language generator (Chen et al., 2024) derived from the CARLA's privileged information, eliminating the influence of perception.

4.2 Evaluation Metrics

The evaluation metrics include (1) Route Completion (RC), which measures the percentage of the route distance that an agent completes; (2) Infraction Penalty (IP), which keeps track of various infractions (e.g., collisions, running a red light) in the simulation. The IP aggregates these infractions committed by an agent as a geometric series. Agents start with an ideal 1.0 base score, which is reduced each time an infraction is committed; (3) Driving Score (DS), the principal metric, is defined as the product of the route completion and the infraction penalty.

4.3 Implementation Details

Our programming planner leverages GPT Model APIs (`gpt-4-turbo-preview` and `gpt-3.5-turbo`)(OpenAI, 2023) as the LLMs. Specifically, we use GPT-4 as the planning LLM (ℓ). For the database implementation, we utilize the open-source vector database Chroma(Chroma, 2023), where the key is the embedding vector for the program description generated by GPT-3.5, and the value is the policy code itself, following the approach in (Wang et al., 2023a).

To facilitate efficient retrieval, we employ `text-embedding-ada-002` APIs for text embedding. In the RAG process, we include the top-3 retrieved programs in the prompt to provide additional context for generating the driving policy. The navigation instructions used in our experiments are provided by (Shao et al., 2024). We generate the driving environment context using a structured language generator, following (Chen et al., 2024).

The execution of an LMP involves using the Python `exec` function, which takes the LMP code as an input string, along with two dictionaries defining the execution scope: (i) `apis`, containing all driving APIs the code may call, and (ii) `local_vars`, initially empty, but later holding a generator variable named `policy` once executed.

4.4 Results

In the experiments, the programming planner—without human-in-the-loop learning and RAG integration—is referred to as either the 0-shot or 3-shot chain-of-thought baselines, in line with standard practices (Wei et al., 2022b). Both baselines use the same API documentation for all test routes.

In the few-shot setting, in-context examples are created by a human programmer proficient in Python. These programmers are given API documentation and allowed to write and test their code on the official training routes. The same examples are used across all test routes. The human-in-the-loop learning process is conducted on the official training routes and concludes once the database contains 50 programs.

We compare our programming planner-based agent with RL experts that have access to privileged information, such as Roach (Zhang et al., 2021) and TCP (Wu et al., 2022). It's important to note that these comparisons aren't necessarily fair. Our setting of following user instructions is inherently more difficult than following waypoints. The comparison aims to reference previous works, and the results can be found in Table 2.

From these results, we make the following observations: Initially, without any few-shot examples, the out-of-the-box LLM struggles with the precise reasoning needed for complex closed-loop driving in CARLA. The 3-shot baseline also falls short, resulting in significantly lower scores according to CARLA's standard metrics. However, our programming planner with 50 code snippets learned from human feedback, performs on par with the Roach Expert RL baseline. Most of the programming planner's failures stem from the trade-off made for the LLM low-level planning frequency. Here's an example of this failure:

```python
# Command: Alright, you can start driving.
# Context Info:
# There is a vehicle at a distance of 6.8 m.
# It is moving at a speed of 0.0 m/s.
# It is at an angle of 1.9 degrees.
# The angle between the headings is 0.0 degrees.
# My current speed is 0.0 m/s.
def start_driving_with_front_vehicle_check():
    (is_front_vehicle,
     distance_to_front_vehicle,
     speed_of_front_vehicle)
     = check_front_vehicle()
    if (is_front_vehicle
        and distance_to_front_vehicle < 5
        and speed_of_front_vehicle == 0.0):
        yield stop()
    else:
        speed_limit = check_speed_limit()
        yield proceed(speed_limit)
```

In this case, the vehicle could collide with a front vehicle if it starts to move. This issue arises because the LLM reasoning is based on the current context information, not accounting for all potential future actions of surrounding agents. However, due to the latency of LLM reasoning, it's not feasible to employ LLM to reason at every single timestamp. We leave this issue to be addressed in future research.

5 Discussion

5.1 The Benefits and Risks of Driving with Human Feedback Learning

Many previous autonomous driving strategies have relied heavily on data-driven approaches, such as reinforcement learning and meta-learning, facing challenges like data selection bias, long-tailed failure cases, lack of interpretability, and difficulties in transferring weights of models. Thus those RL-based methods have poor sample efficiency and it is quite challenging to collect massive amounts of offline data to pretrain the model. To address these issues, our approach integrates human driver feedback with LLMs to steer the creation of driving policy programs within the driving environments. Leveraging the advanced capabilities of CodeLLMs and RAG, our proposed framework matches the performance of reinforcement learning-based methods, even under a more challenging condition where access to waypoints during driving is restricted. This demonstrates the potential of combining human insights with the power of LLMs to enhance the capability of autonomous driving.

There are also some potential risks of human feedback learning. Creating an efficient and stable continual learning mechanism is essential in human feedback learning due to the absence of a clear mathematical or logical method to accurately capture subjective human preferences. However, collecting human feedback from a limited demographic with similar behavior preferences can lead to skewed model performance, especially when applied across diverse user groups or in contexts affected by evaluators' inherent biases. This issue particularly appears in driving scenarios, where varying driving styles, such as aggressive or defensive, significantly influence the feedback, underscoring the need for a broad approach to feedback collection to ensure the model's safety, versatility, and fairness.

5.2 Future Works

In autonomous driving, there exists an often overlooked area — adaptability to the individual driving preferences of users. Traditional autonomous driving systems have largely ignored the personalization factor, focusing instead on creating a centralized and general solution. However, the core objective of the autonomous driving system is not only transportation, but also to serve human needs. This perspective highlights the notion that the autonomous driving system should not only follow basic self-defined navigational commands but also understand and adapt to the unique driving styles of its users. To build better personalized autonomous driving systems, our proposed method provides a potential direction, that is using LLMs to understand the human needs for driving and embed human needs via code generation into existing driving pipelines.

Besides, real-world driving demands interaction with other road users. Interacting with human-driven vehicles and understanding human behavior poses a significant challenge in autonomous driving. Human drivers often rely on non-verbal signals, such as slowing down to yield the right-of-way or using visible gestures to communicate with other road users. These non-verbal cues play a vital role in communication on the road. In the past, there have been numerous accidents involving autonomous driving systems because they behaved unexpectedly. In the future, LLMs with human feedback learning can be utilized to enhance autonomous driving systems, where they can incorporate context-rich information from various sources and also analyze drivers' driving styles to better understand these social cues and make informed decisions. By estimating other drivers' characteristics (such as selfishness or altruism) and types of social interaction (such as cooperation, conflict, competition, coercion, or exchange) based on these social cues, LLMs can improve autonomous driving systems' decision-making capabilities and overall safety.

6 Conclusion

In this paper, we proposed a human-centered autonomous driving learning pipeline with CodeLLMs supported by Retrieval-Augmented Generation (RAG). Our method exhibits strong learning abilities to embed human driving knowledge and preference into the decision-making process of autonomous driving. We conducted interactive experiments in the CARLA autonomous driving simulator by letting experienced human drivers provide feedback. After learning with human feedback, our programming planner achieves much better performance than zero-shot and few-shot chain-of-thought baselines. More importantly, our framework can perform compelling results using only 50 committed code snippets in the Knowledge Database for RAG without any parameter updates. Although our framework has shown promising results in the current setting, there are still some limitations. Future improvements could focus on a better personalization and understanding of human behavior in real-world scenarios.

Limitations

While LLMs can act as a bridge to connect users and vehicles, they have inherent limitations, including significant computational costs and high latency. To enhance time efficiency during inference, we leverage RAG and human feedback learning to collect user input and optimize action code. However, this process also requires considerable time. Ideally, human feedback could be replaced by another LLM acting as a code debugging agent, a promising area for future research.

A pertinent question is whether humans provide optimal solutions during human feedback learning. Our experiments indicate that users may sometimes mislead the autonomous driving system by providing incorrect feedback, potentially endangering other road users. Consequently, our experiments and research conclusions are currently limited to simulation environments. There remains a significant journey ahead to fully integrate LLMs into human-centric autonomous driving systems. The primary objective of our paper is not to propose a framework ready for deployment in real vehicles but to introduce a novel human-centric framework for autonomous driving tasks and emphasize the critical role of human-vehicle interaction in enhancing AI systems for human use.

Ethical Statement

We acknowledge that our work is aligned with the ACL Code of the Ethics and will not raise ethical concerns. We do not use sensitive datasets/models that may cause any potential issues.

References

Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *CHI*.

Alex M. Andrew. 2003. Humans and Automation: System Design and Research Issues. *Kybernetes*, 32(9/10).

Jean-François Bonnefon, Azim Shariff, and Iyad Rahwan. 2016. The social dilemma of autonomous vehicles. *Science*, 352(6293):1573–1576. Publisher: American Association for the Advancement of Science.

Xu Cao, Tong Zhou, Yunsheng Ma, Wenqian Ye, Can Cui, Kun Tang, Zhipeng Cao, Kaizhao Liang, Ziran Wang, James M Rehg, et al. 2024. Maplm: A real-world large-scale vision-language benchmark for map and traffic scene understanding. In *CVPR*, pages 21819–21830.

CARLA. 2020. Autonomous Driving Leaderboard.

Raphael Chekroun, Marin Toromanoff, Sascha Hornauer, and Fabien Moutarde. 2023. GRI: General Reinforced Imitation and Its Application to Vision-Based Autonomous Driving. *Robotics*, 12(5):127.

Dian Chen, Vladlen Koltun, and Philipp Krähenbühl. 2021. Learning To Drive From a World on Rails. In *ICCV*.

Dian Chen and Philipp Krähenbühl. 2022. Learning From All Vehicles. In *CVPR*.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. 2019. Learning by Cheating. In *CoRL*.

Long Chen, Oleg Sinavski, Jan Hünermann, Alice Karnsund, Andrew James Willmott, Danny Birch, Daniel Maund, and Jamie Shotton. 2024. Driving with LLMs: Fusing Object-Level Vector Modality for Explainable Autonomous Driving. In *ICRA*.

Kashyap Chitta, Aditya Prakash, and Andreas Geiger. 2021. NEAT: Neural Attention Fields for End-to-End Autonomous Driving. In *ICCV*.

Chroma. 2023. The AI-native open-source embedding database.

Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, and Ziran Wang. 2024a. Drive as You Speak: Enabling Human-Like Interaction with Large Language Models in Autonomous Vehicles. In *WACVW*.

Can Cui, Yunsheng Ma, Xu Cao, Wenqian Ye, Yang Zhou, Kaizhao Liang, Jintai Chen, Juanwu Lu, Zichong Yang, Kuei-Da Liao, Tianren Gao, Erlong Li, Kun Tang, Zhipeng Cao, Tong Zhou, Ao Liu, Xinrui Yan, Shuqi Mei, Jianguo Cao, Ziran Wang, and Chao Zheng. 2024b. A Survey on Multimodal Large Language Models for Autonomous Driving. In *WACVW*.

Thierry Deruyttere, Simon Vandenhende, Dusan Grujicic, Luc Van Gool, and Marie-Francine Moens. 2019. Talk2Car: Taking Control of Your Self-Driving Car. In *EMNLP-IJCNLP*.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. In *CoRL*.

Lex Fridman. 2018. Human-Centered Autonomous Vehicle Systems: Principles of Effective Shared Autonomy. *arXiv*.

Daocheng Fu, Wenjie Lei, Licheng Wen, Pinlong Cai, Song Mao, Min Dou, Botian Shi, and Yu Qiao. 2024a. LimSim++: A Closed-Loop Platform for Deploying Multimodal LLMs in Autonomous Driving. *arXiv*.

Daocheng Fu, Xin Li, Licheng Wen, Min Dou, Pinlong Cai, Botian Shi, and Yu Qiao. 2024b. Drive Like a Human: Rethinking Autonomous Driving with Large Language Models. In *WACVW*.

Google PAIR. 2019. People + AI Guidebook.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In *ICLR*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *arXiv*.

Shu Ishida, Gianluca Corrado, George Fedoseev, Hudson Yeo, Lloyd Russell, Jamie Shotton, João F. Henriques, and Anthony Hu. 2024. LangProp: A code optimization framework using Language Models applied to driving. *arXiv*.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *NeurIPS*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS*.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as Policies: Language Model Programs for Embodied Control. In *ICRA*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In *ACL*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In *NeurIPS*. arXiv.

Yunsheng Ma, Can Cui, Xu Cao, Wenqian Ye, Peiran Liu, Juanwu Lu, Amr Abdelraouf, Rohit Gupta, Kyungtae Han, Aniket Bera, James M. Rehg, and Ziran Wang. 2024. LaMPilot: An Open Benchmark Dataset for Autonomous Driving with Language Model Programs. In *CVPR*.

Srikanth Malla, Chiho Choi, Isht Dwivedi, Joon Hee Choi, and Jiachen Li. 2023. DRAMA: Joint Risk Localization and Captioning in Driving. In *WACV*.

Jiageng Mao, Junjie Ye, Yuxi Qian, Marco Pavone, and Yue Wang. 2023. A Language Agent for Autonomous Driving. *arXiv*.

Ana-Maria Marcu, Long Chen, Jan Hünermann, Alice Karnsund, Benoit Hanotte, Prajwal Chidananda, Saurabh Nair, Vijay Badrinarayanan, Alex Kendall, Jamie Shotton, and Oleg Sinavski. 2023. LingoQA: Video Question Answering for Autonomous Driving. *arXiv*.

Laksh Nanwani, Anmol Agarwal, Kanishk Jain, Raghav Prabhakar, Aaron Monis, Aditya Mathur, Krishna Murthy Jatavallabhula, AH Abdul Hafez, Vineet Gandhi, and K Madhava Krishna. 2023. Instance-level semantic maps for vision language navigation. In *RO-MAN*, pages 507–512. IEEE.

OpenAI. 2023. GPT-4 Technical Report. *arXiv*.

Raja Parasuraman and Victor Riley. 1997. Humans and Automation: Use, Misuse, Disuse, Abuse. *Human Factors*, 39(2):230–253.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations. In *NAACL*. ArXiv:1808.09121 [cs].

Aditya Prakash, Kashyap Chitta, and Andreas Geiger. 2021. Multi-Modal Fusion Transformer for End-to-End Autonomous Driving. In *CVPR*.

Nivedita Rufus, Kanishk Jain, Unni Krishnan R Nair, Vineet Gandhi, and K Madhava Krishna. 2021. Grounding linguistic commands to navigable regions. In *IROS*, pages 8593–8600. IEEE.

Hao Shao, Yuxuan Hu, Letian Wang, Steven L. Waslander, Yu Liu, and Hongsheng Li. 2024. LMDrive: Closed-Loop End-to-End Driving with Large Language Models. In *CVPR*.

Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Ping Luo, Andreas Geiger, and Hongyang Li. 2023. DriveLM: Driving with Graph Visual Question Answering. *arXiv*.

Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Chenxu Hu, Yang Wang, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. 2024. DriveVLM: The Convergence of Autonomous Driving and Large Vision-Language Models. *arXiv*.

Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. 2020. End-to-End Model-Free Reinforcement Learning for Urban Driving Using Implicit Affordances. In *CVPR*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv*.

Wenhai Wang, Jiangwei Xie, ChuanYang Hu, Haoming Zou, Jianan Fan, Wenwen Tong, Yang Wen, Silei Wu, Hanming Deng, Zhiqi Li, Hao Tian, Lewei Lu, Xizhou Zhu, Xiaogang Wang, Yu Qiao, and Jifeng Dai. 2023b. DriveMLM: Aligning Multi-Modal

Large Language Models with Behavioral Planning States for Autonomous Driving. *arXiv*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *NeurIPS*.

Licheng Wen, Daocheng Fu, Xin Li, Xinyu Cai, Tao Ma, Pinlong Cai, Min Dou, Botian Shi, Liang He, and Yu Qiao. 2024. DiLu: A Knowledge-Driven Approach to Autonomous Driving with Large Language Models. In *ICLR*.

Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. 2022. Trajectory-guided Control Prediction for End-to-end Autonomous Driving: A Simple yet Strong Baseline. In *NeurIPS*.

Yang Xing, Chen Lv, Dongpu Cao, and Peng Hang. 2021. Toward human-vehicle collaboration: Review and perspectives on human-centered collaborative automated driving. *Transportation Research Part C: Emerging Technologies*.

Ke Yang, Jiateng Liu, John Wu, Chaoqi Yang, Yi R. Fung, Sha Li, Zixuan Huang, Xu Cao, Xingyao Wang, Yiquan Wang, Heng Ji, and Chengxiang Zhai. 2024a. If LLM Is the Wizard, Then Code Is the Wand: A Survey on How Code Empowers Large Language Models to Serve as Intelligent Agents. *ICLR Workshop on LLM Agents*.

Yi Yang, Qingwen Zhang, Ci Li, Daniel Simões Marta, Nazre Batool, and John Folkesson. 2024b. Human-Centric Autonomous Systems With LLMs for User Command Reasoning. In *WACVW*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *arXiv preprint*. ArXiv:2305.10601 [cs].

Jianhao Yuan, Shuyang Sun, Daniel Omeiza, Bo Zhao, Paul Newman, Lars Kunze, and Matthew Gadd. 2024. RAG-Driver: Generalisable Driving Explanations with Retrieval-Augmented In-Context Learning in Multi-Modal Large Language Model. *arXiv*.

Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. 2021. End-to-End Urban Driving by Imitating a Reinforcement Learning Coach. In *ICCV*.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2023. Automatic Chain of Thought Prompting in Large Language Models. In *ICLR*.