Drive Like a Human: An LLM-Driven Framework for Long-Tail Highway Autonomous Driving

Sajad Ahmadi
Department of Mechanical Engineering
Clemson University, Clemson, SC 29634, USA
Email: sahmadi@clemson.edu

Instructor: Dr. Kai Liu
School of Computing, Computer Science Division
Clemson University, Clemson, SC 29634, USA
Email: kail@clemson.edu

***Abstract*—Modern autonomous driving (AD) systems achieve strong performance on common driving patterns but remain brittle in rare, long-tail scenarios such as sudden cut-ins, unusual merges, or partially occluded vehicles. These corner cases are precisely where safety is most critical and where purely pattern-based policies often fail to generalize. Motivated by the way human drivers rely on common sense, causal reasoning, and previous experience, this work investigates a layered autonomous driving stack in which a large language model (LLM) serves as a high-level reasoning engine. The proposed framework operates in the `HighwayEnv` simulator and combines: (i) a structured scenario representation that converts low-level states into semantic descriptions, (ii) an *LLMDriver* module that reasons about the scene and selects high-level actions, and (iii) a Failure Memory Bank that turns crashes into reusable lessons for future episodes. We evaluate this architecture on long-tail highway scenarios and report both quantitative metrics (crash rate and cumulative reward) and qualitative behaviors. The results indicate that combining LLM-based reasoning with safety constraints and memory can reduce the crash rate while preserving efficient driving behavior.**

***Index Terms*—Autonomous driving, long-tail scenarios, large language models, case-based reasoning, highway simulation.**

I. Introduction

Autonomous driving has made remarkable progress over the past decade, driven by advances in perception, planning, and control pipelines [1–3]. Classical systems decompose the problem into modular stages (detection, tracking, prediction, planning, and control), with carefully engineered interfaces and safety constraints. In parallel, end-to-end learning has been explored as a way to directly map sensor observations to steering and throttle commands, from early neural-network pilots such as ALVINN [4] to modern deep architectures trained on large-scale driving corpora [5–7]. These lines of work have enabled impressive performance in routine conditions, but they still struggle when faced with rare events and distributional shift.

A key challenge is the "long tail" of safety-critical but infrequent situations: unusual cut-ins, stalled or aggressively merging vehicles, occluded pedestrians, and complex multi-vehicle interactions. Empirical studies of anomaly detection and risk prediction in driving show that these rare events are both difficult to collect and hard to anticipate from standard training data [8, 9]. As a result, policies that perform well on average may fail catastrophically when they encounter novel combinations of speeds, relative positions, and driver behaviors. Recent efforts have begun to systematize long-tail evaluation by constructing adversarial or vulnerability-search frameworks for autonomous driving policies, including those guided by large language models [10, 11]. However, most of these approaches still treat each episode in isolation, with little explicit mechanism for remembering and reusing "lessons learned" from previous failures.

Deep reinforcement learning (DRL) offers a complementary way to train decision-making policies for complex driving tasks [12]. DRL-based controllers have been proposed for highway merging, lane changing, and interaction with uncertain or noncooperative traffic participants [13–15]. While such methods can in principle discover robust strategies, they remain data-hungry, sensitive to reward shaping, and difficult to interpret or formally verify at scale. Moreover, they typically represent experience in terms of numeric value functions or policies, rather than explicit, human-readable descriptions of what went wrong and how to avoid it next time.

At the same time, large language models (LLMs) have emerged as powerful general-purpose reasoners capable of integrating structured observations, commonsense knowledge, and task instructions. In robotics and embodied AI, LLMs have been used to generate high-level plans, decompose tasks, and ground language in perception and action [16–19]. Motivated by this trend, a growing body of work explores LLMs specifically for autonomous driving: surveys such as LLM4Drive [20] and related overviews [21–23] argue that LLMs can offer more interpretable decision-making, better incorporation of traffic rules, and flexible reasoning over diverse scenarios. Concretely, Fu *et al.* propose the *Drive Like a Human* framework, in which an LLM-controlled agent reasons over textual descriptions of the scene to output driving actions in simulation [24, 25]. These results suggest that language-based agents could be a promising substrate for human-like, cautious driving in challenging highway environments.

Safety considerations further motivate such a memory-centric view. LLM-driven stacks must not only drive competently but also integrate with safety shields and formal guarantees developed for classical and learning-based controllers. Safe reinforcement learning and shielded control frameworks seek to wrap learned policies in additional safety layers [26], while formal models such as Responsibility-Sensitive

Safety (RSS) provide interpretable conditions under which a vehicle can be deemed safe [27]. Recent surveys on LLMs for autonomous driving highlight that safety, reliability, and interpretability are central open problems [21, 23]. A memory mechanism that records, explains, and reuses failure cases could serve as a bridge between data-driven LLM reasoning and these more formal notions of safety.

In this work, we build on the *Drive Like a Human* framework [24, 25] and design a layered LLM-driven autonomous driving stack for highway scenarios with an explicit focus on long-tail events. Our system comprises a *Scenario Engine* that constructs challenging highway scenes, an *LLMDriver* that reasons over textual state descriptions to propose actions, and a *Failure Memory Bank* that stores LLM-written analyses of problematic episodes as reusable cases. During later runs, the LLMDriver retrieves similar failure cases and conditions its decisions on this experience, effectively performing case-based reasoning in natural language. We evaluate the resulting framework in simulated highway environments with dense and noncooperative traffic, and we compare an LLM-only baseline with our LLM-plus-memory variant in terms of crash rate, reward, and driving efficiency. The results show that explicitly encoding and reusing failure knowledge can meaningfully reduce crashes in long-tail scenarios while preserving human-like, efficient driving behavior.

The remainder of the paper develops this idea in detail. Section II formulates the problem and objectives. Section III describes the architecture and methodology, with particular emphasis on the Scenario Engine, main control loop, LLMDriver, and Failure Memory Bank. Section IV explains the simulation setup and long-tail scenarios considered in the experiments. Section V reports quantitative results and summarizes the behavior of different driver variants. Section VI presents qualitative trajectory examples and discusses the observed behaviors. Section VII concludes and outlines directions for future work.

II. Problem Statement and Objectives

The central problem considered in this work is the design of a highway autonomous driving agent that performs robustly in rare, safety-critical scenarios. The agent operates in discrete time in the `HighwayEnv` simulator. At each time step, it receives an observation describing the ego vehicle and surrounding traffic, and it must output a high-level action such as "keep lane," "change lane left," "change lane right," or "slow down." Episodes terminate when a collision or timeout occurs.

Conventional approaches often train policies using deep reinforcement learning or imitation learning [**?**]. While effective in distribution, these policies may generalize poorly to novel combinations of vehicles and road conditions and may lack interpretability. In rare scenarios, they often produce aggressive or inconsistent maneuvers. Moreover, failure episodes are typically treated as training samples without an explicit mechanism for distilling and reusing lessons in an interpretable form.

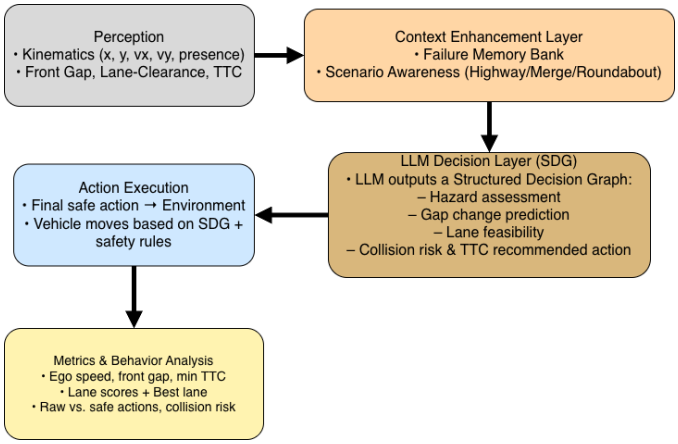


Fig. 1: Overall architecture of the LLM-driven autonomous driving stack. The `HighwayEnv` simulator provides low-level observations, which are converted by the Scenario Engine into structured representations. The LLMDriver uses these representations, together with retrieved memory cases, to propose actions that are applied to the environment.

Our objective is therefore twofold. First, we aim to design a decision-making stack in which an LLM serves as a high-level reasoning module that operates on structured, semantic descriptions of the traffic scene. The LLM is encouraged to follow traffic rules, respect safety constraints, and explain its decisions. Second, we seek to augment this reasoning layer with a Failure Memory Bank so that the agent can: (i) learn explicit lessons from crashes; (ii) retrieve these lessons at inference time when similar situations arise.

The framework is instantiated in highway-like scenarios such as multi-lane driving, merging, roundabouts, and intersections. These scenarios contain both frequent patterns and carefully designed long-tail variations. The goal is to assess whether the LLM-driven architecture can "drive like a cautious human," reducing collisions and maintaining reasonable efficiency compared with naive LLM control.

III. Architecture and Methodology

A. Layered System Overview

The overall system architecture is depicted in Fig. 1. At the lowest layer, the `HighwayEnv` simulator maintains the dynamical state of the ego vehicle and surrounding traffic. Above it, the Scenario Engine converts raw numerical observations into a structured representation. The main control loop, implemented in `HELLM.py`, orchestrates interactions between the environment, the LLMDriver module, the safety wrapper, and the logging infrastructure. The Failure Memory Bank, shown schematically in Fig. 4, stores compact descriptions of failure episodes and provides retrieved cases to the LLMDriver when similar situations are encountered again.

B. Scenario Engine and State Representation

The Scenario Engine plays a central role in making the environment interpretable to the LLM. Rather than feeding raw

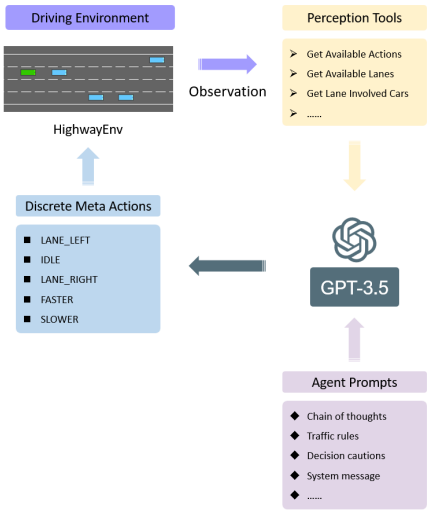


Fig. 2: Placeholder: Scenario Engine pipeline. Raw numerical observations are converted into a structured representation that includes a lane graph and vehicle objects with semantic roles. The resulting state is stored in a log together with actions and rewards.

vectors of positions and velocities, we construct a JSON-like representation that explicitly encodes the lane graph, vehicle states, and relational information. A high-level flow of this component is illustrated in Fig. 2.

Starting from the simulator's observation, the Scenario Engine identifies the ego vehicle and constructs a lane-based description of its surroundings. Each lane is given an index, and vehicles are assigned to lanes based on their lateral position. For each vehicle, the engine records its longitudinal position, velocity, and relative ordering with respect to the ego. It then identifies key relations such as "front vehicle in the same lane," "vehicle in the left lane ahead," or "vehicle in the right lane behind." These relations are expressed symbolically in the scenario description.

The engine also augments the state with derived quantities, such as time-to-collision estimates and gaps to the nearest obstacles in each direction, when these are available from the simulator. Finally, each processed time step, together with the corresponding LLM decision and environment reward, is logged in an SQLite database. This log is used both for offline analysis and as a source of data for memory construction.

C. Main Control Loop (HELLM)

The main control loop, summarized in Fig. 3, connects all components in a closed loop. At the start of each episode, the environment is reset with a given scenario configuration (highway, merge, roundabout, or intersection), and the log buffers are cleared. The LLM model is instantiated with a system prompt that encodes driving instructions, safety guidelines, and the desired output format. The Failure Memory Bank is loaded and made available for retrieval.

At each time step, the loop proceeds as follows. First, the environment produces the current observation. This observation is passed to the Scenario Engine, which returns the corresponding structured representation. Based on the current scenario type and difficulty level, a query is issued to the Failure Memory Bank to retrieve relevant past cases, if any exist. These cases, which summarize previous failures and associated lessons, are appended to the LLM prompt as contextual examples.

```python
env = gyn.nake("highway-v8", render_aede="rgb_array")
env.configure(config)
env = RecordVideo(
    env, './results-video',
    nane_prefix=f"highwayv8"
)
env.unwrapped.set_record_video_wrapper(env)
obs, info = env.reset()
env.render()

# scenario and driver agent setting
if not os.path.exists("results-db/"):
    os.akdir("results-db")
database = f'results-db/highwayve.db"
sce = Scenario(vehicleCount, database)
toolModels = (
    getAvailableActions(env),
    getAvailablelanes(sce),
    getLaneInvolvedCar(sce),
    isChangeLaneConflictWithCar(sce),
    isAccelerationConflictWithCar(sce),
    isKeepSpeedConflictWithCar(sce),
    isDecelerationSafe(sce),
    isActionSafe(),
]
DA = DriverAgent(llm, toolModels, sce, verbosewTrue)
outputParser = OutputParser(sce, llm)
output = None
done = truncated = False
frame = 0
try:
    while not (done or truncated):
        sce.upateVehicles(obs, frame)
        DA.agentRun(output)
        da_output = DA.exportThoughts()
        output = outputParser.agentRun(da_output)
        env.render()
        env.unwrapped.automatic_rendering_callback = env.video_recorder.capture_frame()
        obs, reward, done, info, _ = env.step(output["action_id"])
        print(output)
        frame += 1
finally:
    env.close()
```

Fig. 3: Main control loop implemented in `HELLM.py`. At each time step, the loop obtains an observation, constructs the structured state, retrieves relevant memory cases, queries the LLMDriver for an action, applies the action, and logs the resulting transition.

The LLMDriver is then called with the updated prompt and state. It returns a proposed high-level action together with its chain-of-thought reasoning, which is captured by a function such as `exportThoughts()`. Before the action is executed, the safety wrapper evaluates it against numeric constraints, such as minimum headway distance and time-to-collision thresholds. If the action is deemed unsafe, it is replaced by a safe fallback, such as keeping the lane and reducing speed.

The final action is applied to the environment, which advances the simulation by one step. The new observation, reward, and termination flag are recorded in the log. When the episode terminates, either by collision or timeout, the final segment of the trajectory is used to update the Failure Memory Bank, as explained next.

D. Failure Memory Bank

The Failure Memory Bank is designed to transform collisions and severe near-misses into structured experiences that

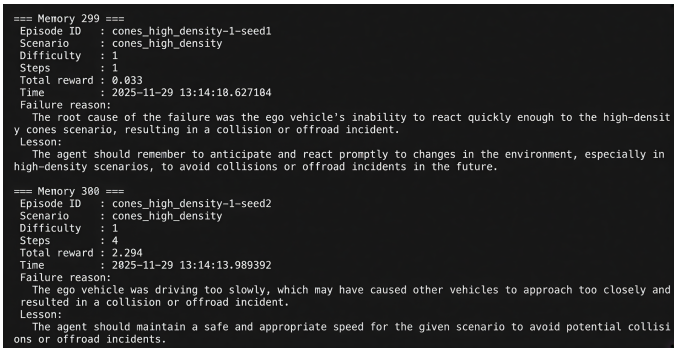


Fig. 4: Failure Memory Bank. Crash trajectories are summarized and converted into interpretable cases containing a scenario tag, root-cause analysis, and a lesson. During future episodes, relevant cases are retrieved and provided to the LLM as contextual examples to bias decisions toward safer behaviors.

influence future decisions. Its operation is illustrated in Fig. 4.

During an episode, the logging system continuously records the structured state, the LLM's proposed actions, the actions executed after safety filtering, the environment rewards, and a flag indicating whether a collision has occurred. When an episode ends in collision, a summary of the final portion of the trajectory is constructed. This summary includes the scenario type, key states leading up to the collision, and the LLM's reasoning. The summary is then fed back to the LLM in an offline analysis prompt, which asks the model to identify the root cause of the failure and to formulate a concise lesson that would help avoid similar mistakes.

The resulting case is stored in the Failure Memory Bank as a record with fields such as scenario label, difficulty level, high-level textual description, root-cause explanation, and a natural-language lesson (for example, "do not change lanes into a gap that is closing rapidly when the rear vehicle is accelerating"). Cases are indexed by simple metadata and optionally by vector embeddings of their textual descriptions, allowing approximate similarity search.

At inference time, when the agent encounters a new state, a query is formed using the current scenario type and summary statistics of the state. The memory bank returns the most relevant cases, which are injected into the LLM prompt as previous experiences. In this way, the agent can "remember" that certain patterns have previously led to collisions and can choose more conservative actions in analogous situations.

IV. Simulation Setup and Long-Tail Scenarios

A. Environment and Scenario Design

All experiments are conducted in the `highway-env` framework, which provides configurable scenarios for multi-lane highways, highway merges, roundabouts, and intersections. The state of the ego vehicle includes its position, velocity, orientation, and current lane, while the surrounding vehicles are modeled as kinematic agents with prescribed behaviors.

We configure several scenario families with varying traffic densities and behavior patterns. In the highway scenario, the ego vehicle travels on a multi-lane road with faster and slower vehicles. The long-tail variations include aggressive cut-ins by vehicles that suddenly enter the ego's lane with small gaps, as well as vehicles in front that brake unexpectedly. In the merge scenario, the ego vehicle must merge into a main lane with dense traffic, sometimes facing noncooperative drivers that do not yield. Roundabout and intersection scenarios introduce right-of-way considerations and partial occlusions.

Representative snapshots of these scenarios are illustrated in Fig. 5. Each panel corresponds to a different environment configuration, showing the ego vehicle, surrounding traffic, and road geometry. These visualizations help interpret the numerical results and provide context for the qualitative behaviors described later.

V. Numerical Results

A. Evaluation Metrics

The quantitative evaluation focuses on three main metrics. The most important metric is the *collision rate*, defined as the fraction of episodes that end in a crash. This metric directly captures safety performance. To assess efficiency, we also report the *cumulative environment reward* per episode as defined by `HighwayEnv`, which typically combines penalties for collisions and off-road events with rewards for maintaining progress and staying in safe lanes. Finally, we monitor the average speed of the ego vehicle to ensure that reductions in collision rate are not achieved simply by driving at unreasonably low speeds.

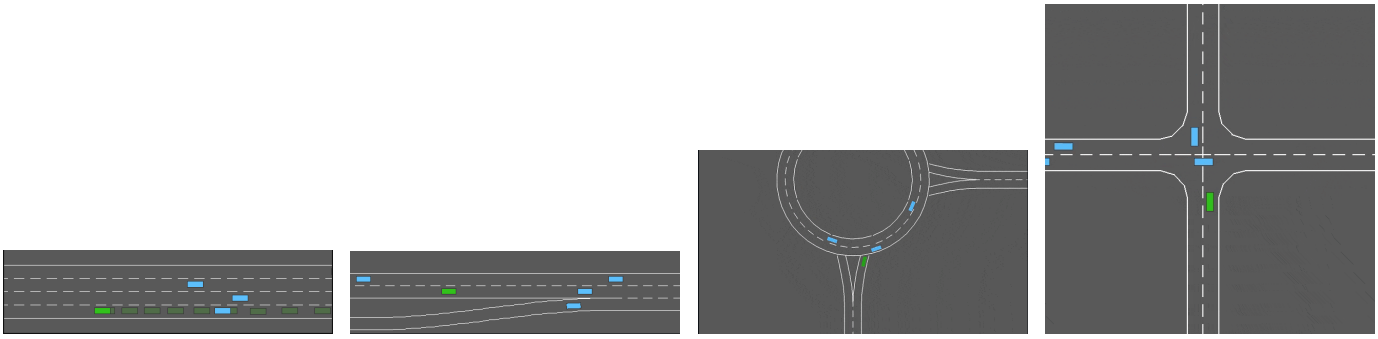
Table I summarizes these metrics for the driver variants. The entry corresponding to the full system (LLM with safety wrapper and Failure Memory Bank) reflects the values reported in the presentation, with a collision rate around ten percent and a relatively high cumulative reward. Other entries are placeholders that should be replaced with numerical values once experiments are run consistently over a large enough set of episodes.

Driver Variant	Collision Rate [%]	Total Reward	Average Speed
Base LLM only	65.7	10.327	28.8
LLM + Failure Memory	20.4	26.936	20.4

TABLE I: Quantitative comparison of driver variants on long-tail highway scenarios. Italicized entries indicate values to be replaced with the corresponding experimental results.

B. Performance Comparison

Even without precise numerical values for every configuration, several trends emerge from the experiments summarized in Table I. The base LLM-only agent, which receives structured state descriptions but no memory and no numeric safety wrapper, tends to exhibit a high collision rate in long-tail scenarios. It can occasionally maintain high speed and collect short-term reward but is prone to catastrophic failures when unusual configurations arise. It is compared against the case



(a) Highway (b) Merge (c) Roundabout (d) Intersection

Fig. 5: Snapshots of the four main scenario families considered in the experiments: (a) multi-lane highway, (b) highway merge, (c) roundabout, and (d) intersection. Long-tail variations are created by introducing rare but challenging configurations such as aggressive cut-ins, late braking, and partial occlusions.

with the Failure Memory Bank, that has lower collision rate and more reward. We lost a little bit of speed, but that's not a lot.

VI. Qualitative Simulation Results

While aggregate metrics provide a high-level comparison, qualitative inspection of episode trajectories offers deeper insight into how the LLM-driven agent behaves.

In a high-speed cut-in scenario, the base LLM-only agent often attempts to maintain speed and accept small gaps, underestimating the risk of an approaching rear vehicle in the target lane. This can lead to collisions when the rear vehicle accelerates or fails to yield. In contrast, the full system, informed by previous failure cases that involved similar cut-ins, tends to either delay the lane change or choose a conservative deceleration while waiting for a larger gap.

In a merge scenario with dense traffic, the base agent may exhibit indecisive behavior, oscillating between accelerating to join a fast-moving stream and braking to avoid collision, resulting in abrupt maneuvers. With the memory bank the agent more consistently chooses a feasible gap and sometimes accepts a slower, safer position behind a truck instead of forcing its way ahead. The retrieved failure lessons explicitly remind the LLM of crashes that occurred when attempting to squeeze into tight gaps.

Roundabout and intersection scenes reveal similar patterns. The LLM-only agent occasionally fails to yield appropriately or misjudges the intent of cross traffic. The memory-augmented agent, on the other hand, recalls episodes where aggressive entry led to collisions and therefore chooses to wait longer, especially when occlusions or ambiguous right-of-way situations arise.

Together, these examples show that the Failure Memory Bank and the safety wrapper do more than simply reduce collision counts: they shape the qualitative driving style, encouraging policies that resemble cautious human drivers, particularly when the configuration looks similar to previously observed failures.

VII. Discussion

The experiments presented above illustrate both the promise and the current limitations of LLM-driven autonomous driving. On the positive side, the combination of structured scenario representations, tool-augmented reasoning, numeric safety constraints, and failure-aware memory leads to a clear improvement over naive LLM control in long-tail scenarios. The agent can use prior "lessons" to guide its behavior and tends to provide explanations that can be inspected by developers or safety engineers.

At the same time, several challenges remain. First, the experiments take place in a relatively idealized simulator with full observability and simplified vehicle dynamics. Real-world driving involves noisy perception, occlusions, and a richer set of agents, including pedestrians and cyclists. Second, the memory mechanisms, while effective in the limited setting, rely on straightforward similarity metrics and may need to be replaced by more sophisticated retrieval models when scaling to larger and more diverse experience banks.

These observations suggest that LLM-based decision-making should be viewed not as a replacement for traditional control and planning algorithms, but as a complementary component that can infuse common-sense reasoning and interpretability into the stack. In many cases, the most promising design may be a hybrid architecture in which an LLM proposes high-level intentions or rule-based plans that are then refined by optimization-based planners and enforced by formal safety shields.

VIII. Conclusion and Future Work

This paper proposed and analyzed an LLM-driven framework for long-tail highway autonomous driving. The approach centers on a layered architecture in which a large language model acts as a high-level decision maker, operating on structured scenario descriptions produced by a Scenario Engine. A Failure Memory Bank converts crashes into reusable cases that bias the agent toward safer decisions in similar future situations.

Through simulation experiments in the `HighwayEnv` framework, we observed that the full system substantially reduces collision rates relative to a naive LLM-only baseline while maintaining strong cumulative reward. Qualitative analysis of trajectories shows that the memory-augmented agent adopts driving patterns that more closely resemble cautious human behavior, especially in rare and challenging scenarios such as high-speed cut-ins and dense merges.

Several directions for future work are apparent. One direction is to integrate richer perception models and more realistic environment dynamics, including adverse weather and sensor noise. Another is to explore more advanced memory representations and retrieval strategies that can scale to large experience databases and support compositional reasoning. A third direction is to couple the LLM-based decision layer with formal verification or control barrier function-based safety mechanisms, pushing the system toward provable guarantees while retaining the interpretability and flexibility of language-based reasoning.

REFERENCES

[1] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. Paixão, F. Mutz, *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021.

[2] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of field robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[3] 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*, 2024.

[4] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[5] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[6] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[7] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

[8] D. Bogdoll, M. Nitsche, and J. M. Zöllner, "Anomaly detection in autonomous driving: A survey," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2022, pp. 4488–4499.

[9] N. Deng, Z. Cao, W. Zhou, Y. Xu, X. Liu, K. Jiang, and D. Yang, "Long-tail prediction uncertainty aware trajectory planning for self-driving vehicles," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 38–44.

[10] Z. Tang, J. He, D. Pei, K. Liu, and T. Gao, "Testing large language models on driving theory knowledge and skills for connected autonomous vehicles," *arXiv preprint arXiv:2407.17211*, 2024.

[11] L. Qiu, Z. Xu, Q. Tan, W. Tang, C. Yu, and Y. Wang, "Aed: Automatic discovery of effective and diverse vulnerabilities for autonomous driving policy with large language models," *arXiv preprint arXiv:2503.20804*, 2025.

[12] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[13] J. Liao, T. Liu, X. Tang, X. Mu, B. Huang, and D. Cao, "Decision-making strategy on highway for autonomous vehicles using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 177 804–177 814, 2020.

[14] H. Deng, Y. Zhao, Q. Wang, and A.-T. Nguyen, "Deep reinforcement learning based decision-making strategy of autonomous vehicle in highway uncertain driving environments," *Automotive Innovation*, vol. 6, no. 3, pp. 438–452, 2023.

[15] R. Zhang, J. Hou, F. Walter, S. Gu, J. Guan, F. Röhrbein, Y. Du, P. Cai, G. Chen, and A. Knoll, "Multi-agent reinforcement learning for autonomous driving: A survey," *arXiv preprint arXiv:2408.09675*, 2024.

[16] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations (ICLR)*, 2023.

[17] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on Robot Learning (CoRL)*. PMLR, 2022, pp. 571–582.

[18] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," in *Conference on Robot Learning (CoRL)*. PMLR, 2022, pp. 1769–1782.

[19] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, *et al.*, "Palm-e: An embodied multimodal language model," in *International Conference on Machine Learning (ICML)*. PMLR, 2023, pp. 8469–8488.

[20] Z. Yang, X. Jia, H. Li, and J. Yan, "Llm4drive: A survey of large language models for autonomous driving," *arXiv preprint arXiv:2311.01043*, 2023.

[21] Y. Zhu *et al.*, "Large language models for human-like autonomous driving: A survey," *arXiv preprint arXiv:2407.19280*, 2024.

[22] S. Fourati *et al.*, "Multi-agent autonomous driving systems with large language models: A survey," in *Findings of the Association for Computational Linguistics: EMNLP*, 2025.

[23] Y. Wang, R. Jiao, C. Lang, C. Huang, Z. Wang, Z. Yang, and Q. Zhu, "Empowering autonomous driving with large language models: A safety perspective. arxiv. 2023," *arXiv preprint arXiv:2312.00812*, 2023.

[24] 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 Workshops (WACVW)*, 2024, pp. 910–919.

[25] ——, "Drive like a human: Rethinking autonomous driving with large language models," in *2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. IEEE, 2024, pp. 910–919.

[26] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[27] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.