Journal of Machine Learning and Deep Learning (JMLDL)
Volume 3, Issue 2, Date: 31- December– 2026
ISSN: 2978-4336, DOI: 10.64820/AEPJMLDL.32.18.36.122026

Knowledge-Grounded Reasoning, Trajectory Refinement, and Hardware-Efficient Deployment in Autonomous Driving: A Layered Survey Across Three Generations of AI

Gaurav Pokharkar [1, *], Atharva Khandve [2], Gopi Gaddamadugu [3], Akshay Choudhari [4]
[1] Independent Researcher, Detroit, MI, USA
[2] Independent Researcher, San Jose, CA, USA
[3] Independent Researcher, Detroit, MI, USA
[4] Independent Researcher
Email: [1] gpokhark@asu.edu; [2] khandveatharva@gmail.com; [3] gk.gopikris@gmail.com; [4] aachoud1@asu.edu
ORCID: [1] 0009-0003-9610-898X; [2] 0009-0002-1232-024X; [3] 0009-0002-9775-5479; [4] 0000-0001-5852-1165
*Corresponding Author

Abstract— Autonomous driving has evolved through three overlapping generations of artificial intelligence: rule-based controllers, deep-learning perception and policy networks, and knowledge-grounded systems built on large language models (LLMs), vision-language models (VLMs), and vision-language-action (VLA) models. This review covers 107 records (90 autonomous-driving contributions and 17 adjacent foundational sources), retained after a citation-integrity re-audit of nine database and proceedings searches spanning January 2016 through June 2026. We organize the evidence around three interacting pillars: knowledge-grounded reasoning; trajectory generation and refinement; and hardware-aware execution. A two-axis taxonomy classifies methods by functional role and by an operational runtime class (hard real-time, soft real-time, cognitive, or not demonstrated). Foundation-model methods primarily support semantic interpretation, memory, explanation, and high-level planning; dynamically feasible motion still depends on constrained planning, checking, and fallback mechanisms. Most LLM- and VLM-based studies report neither automotive-platform latency nor worst-case timing, and heterogeneous hardware and evaluation protocols across the reported benchmarks preclude a defensible global ranking. We therefore propose a layered design principle: cognitive outputs should enter the driving stack as checked goals or constraints, while requirement-derived deadlines and runtime assurance remain the responsibility of faster planning, monitoring, and control layers.

Keywords—autonomous driving; large language models; vision-language models; trajectory planning; hardware efficiency; runtime classification; ADAS; closed-loop evaluation

I. INTRODUCTION

Autonomous driving (AD) sits at the intersection of robotics, computer vision, machine learning, and control theory. Over roughly two decades its dominant engineering paradigm has shifted twice and is shifting again. Classical systems decomposed driving into a sequential pipeline of localization, mapping, perception, prediction, planning, and control, with each stage specified and validated independently [1]. Deep representation learning then enabled end-to-end approaches that optimize perception and planning jointly rather than in isolation. In these architectures, planning became the final objective rather than a downstream consumer of perception outputs [2], [3]. Most recently, foundation models have opened a third paradigm, knowledge-grounded driving, and the surveyed literature shows large language models (LLMs), vision-language models (VLMs), and vision-language-action (VLA) models contributing commonsense reasoning, interpretability, memory, and language-grounded interaction to the driving stack [4], [5], [6], [7].

Modern AD still faces a persistent gap. Systems perform well on common driving patterns but fail in the long tail: rare scenarios, unusual road geometry, implicit social negotiation, adversarial inputs, and ambiguous human behavior. Each generation addresses part of the gap while introducing its own constraints. Rule-based controllers are auditable but brittle at the edge of their design envelope. Deep networks are robust to variation but opaque and vulnerable to distribution shift. In practice, agentic LLM/VLM systems reason through novel situations that break the first two generations, at the cost of expense, hallucination, and certification difficulty. Figure 1 summarizes this progression.

Reasoning, refinement, and deployment are the three pressures shaping the field, and none of them can be solved in isolation. Reasoning systems must produce plans that can be refined and verified; planning systems must run under hardware constraints; hardware evaluation must account for

Received: 23 -8 -2026
Revised: 24- 9 -2026
Published: 31-12-2026

© 2026 The Author(s). This article is published under the Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0). This license permits others to share, copy, distribute, remix, adapt, and build upon the material, provided they give appropriate credit to the original authors and source. Any adapted or derivative work must be distributed under the same CC BY-SA 4.0 license.

the growing use of LLMs, VLMs, and generative planners. Three pillars organize the literature around this interaction:

Knowledge-based reasoning adds semantic scene understanding, reflection, tool use, retrieval, and natural-language explanation to AD [8], [9], [10].

Trajectory planning and refinement improves the safety and quality of planned motion through planning-oriented architectures, vectorized representations, generative planners, and multi-stage decoder refinement [3], [11], [12], [13].

Hardware-efficient deployment becomes essential because the computational demands of foundation-model reasoning conflict with the real-time, power-limited constraints of onboard automotive hardware [11], [14], [15].

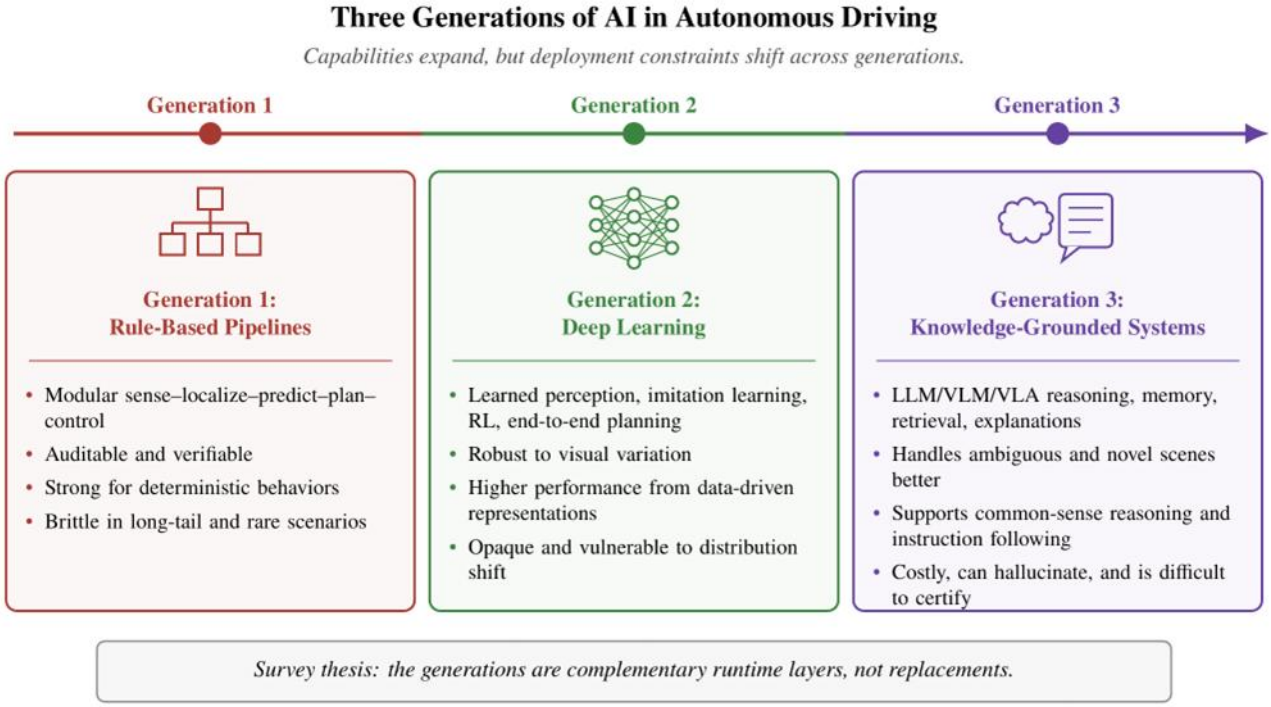


Fig. 1. Three generations of AI in autonomous driving. Each generation expands capability while shifting where the hard constraints lie; the central argument of this survey is that they are complementary runtime layers rather than replacements.

The second contribution is a real-time class axis that cuts across all three pillars and the AD stack. Methods differ in architecture, but the timing evidence they report is what constrains whether a method can plausibly serve in a control path or should remain supervisory. We define hard-real-time, soft-real-time, cognitive, and not-demonstrated classes. The third contribution is a deployment-grounded analysis that makes timing and assurance constraints concrete for three advanced driver assistance system (ADAS) functions: adaptive cruise control, automatic emergency braking, and lane keeping assistance.

The remainder of the paper is structured as follows. Section II describes the scope, methodology, taxonomy, and comparison with existing surveys. Section III covers foundations: backbones, foundation models, perception, datasets, and simulators. Section IV surveys knowledge-based reasoning. Section V surveys trajectory planning and refinement. Section VI addresses hardware efficiency and the real-time class axis. Section VII presents the ADAS deployment analysis. Section VIII covers benchmarks, safety, and verification. Section IX identifies open challenges. Section X presents future research directions. Section XII concludes.

II. Survey Scope and Methodology

A. Scope and Research Questions

This survey covers the full autonomous-driving stack with particular emphasis on the transition toward knowledge-grounded reasoning and on the deployment constraints that this transition creates. It spans both research-oriented full-autonomy systems and commercially deployed ADAS functions (adaptive cruise control, automatic emergency braking, and lane keeping assistance) commonly associated with SAE Level 1–2 capabilities [16]. We include commercial ADAS because these functions convert the reasoning–refinement–deployment argument into concrete, requirement-bounded and safety-governed engineering problems that ground the abstract taxonomy in practice.

Six research questions organize the review and link the three pillars to gaps in the existing literature. **RQ1**: What roles do LLMs, VLMs, and VLA models currently perform in autonomous-driving systems, and what interfaces connect their outputs to the planning stack? **RQ2**: How are semantic intentions produced by reasoning models converted into dynamically feasible and safety-checked trajectories? **RQ3**: What latency, memory, power, and hardware evidence is reported for systems claimed to be deployable on

automotive-grade hardware? **RQ4**: How are methods evaluated across the spectrum from open-loop logged replay through non-reactive simulation, reactive closed-loop simulation, and physical deployment, and what each evaluation level can and cannot establish? **RQ5**: What evidence supports placing a method in the cognitive, soft-real-time, or hard-real-time runtime class, and how often is this placement left implicit or undemonstrated? **RQ6**: What assurance mechanisms—monitors, shields, fallback controllers, uncertainty gates, or minimum-risk maneuvers—constrain or supervise foundation-model outputs before they reach actuators?

B. Screening Methodology

This survey is a structured narrative review. Candidate papers were retrieved through nine searches: IEEE Xplore; CVF Open Access; the NeurIPS, ICLR, CoRL, and AAAI proceedings portals; ACM Digital Library; MDPI Open Access; and arXiv (principally cs.CV, cs.RO, cs.LG, cs.AI, and eess.SY). Database-specific query strings, dates, filters, and retrieval counts are provided in search_log.csv. The search period spans January 2016 through June 2026, with the final retrieval pass completed in June 2026. The year 2016 lower bound captures PilotNet and formative deep-learning work; earlier sources needed to explain enabling architectures or safety concepts are retained as explicitly marked adjacent foundations.

Eligibility was assessed at two stages. In the first stage, title and abstract were screened against the following criteria.

Inclusion criteria:

- Research or review work directly addressing autonomous or automated driving—perception, prediction, decision-making, planning, control, evaluation, or functional safety.
- A clear contribution to at least one of the three survey pillars: knowledge-grounded reasoning, trajectory planning and refinement, or hardware-efficient deployment.
- Sufficient technical detail to classify the method by the two-axis taxonomy defined in Section II-C.
- Full text available in English from the publisher, an open-access repository, or arXiv.

Exclusion criteria:

- General robotics or computer-vision work with no demonstrated or explicitly argued driving relevance.
- Opinion pieces and position papers without a technical method, experiment, or evidence base.
- Duplicate records and superseded preprint versions where a peer-reviewed version was available and substituted.
- Papers for which only an abstract, slide deck, or workshop summary could be retrieved.
- Sources whose authorship, venue, or quantitative results could not be independently verified in a publisher or official repository record.

In the second stage, full texts of retained candidates were read to verify the claimed contribution and to extract the evidence needed for the master evidence matrix in the supplementary material. Reported benchmark numbers were taken directly from the primary source—the specific table or figure identified during full-text review—rather than from secondary citations or survey summaries. Twelve records in the provisional corpus were removed during the citation-integrity re-audit because the primary full text or current publication status could not be independently verified; none contributes a quantitative result in the revised synthesis.

The searches returned 420 records. After removing 110 duplicates, 310 unique records entered title-and-abstract screening, of which 170 were excluded (150 outside the AD scope or below the technical-detail threshold, and 20 not retrievable as full text). Of the 140 full texts assessed, 33 were excluded: 12 superseded by peer-reviewed versions, 6 with insufficient technical detail, 3 unverifiable during the original screen, and 12 removed during the citation-integrity re-audit. The retained corpus comprises 107 records: 90 AD contributions and 17 adjacent foundational sources, as identified in included_studies.csv. Figure 2 summarizes the screening flow.

Quality assessment was applied to the 68 retained empirical primary-method studies for which the rubric was applicable. Eight dimensions—method reproducibility, dataset and split clarity, baseline appropriateness, closed-loop evaluation, hardware transparency, robustness or uncertainty evaluation, code/data availability, and peer-review status—were scored from 0 to 3. The item-level scores and notes are reported in quality_assessment.csv; they are used to qualify the synthesis, not combined as a meta-analytic weight. Standards, datasets, software releases, surveys, and adjacent foundations were retained for context but were not assigned an empirical-method quality score.

C. Taxonomy

The survey organizes methods along two independent axes, each governed by explicit classification rules so that the placement of any given method is reproducible from the stated criteria rather than from the authors' judgment alone.

Functional axis. Every surveyed method is assigned one primary functional role and, where applicable, one or more secondary roles. *Knowledge-grounded reasoning* covers systems that perform language or multimodal semantic reasoning, retrieval from external memory, natural-language explanation, tool use, counterfactual analysis, or instruction interpretation as their central contribution. *Trajectory generation and refinement* covers systems whose primary contribution is motion prediction, trajectory proposal, candidate scoring, iterative correction, constraint checking, or trajectory selection, whether through learned or hybrid methods. *Hardware-aware execution* covers systems whose primary contribution is model compression, accelerator design, predictable scheduling, early-exit inference, target-platform benchmarking, runtime monitoring, or safety-

relevant control execution. Systems whose contributions span multiple roles are classified by the role most central to their stated contribution, with secondary roles noted in the evidence matrix.

Runtime axis. Every method is assigned one of four runtime classes based on the latency and deployment evidence reported in the primary source, not on the authors' general claims. A method is classified as *hard real-time* when a timing deadline is derived from a documented safety or control requirement, worst-case execution time or scheduling analysis is discussed, and the authors argue that deadline misses would have safety consequences. A method is classified as *soft real-time* when timely updates are necessary for acceptable driving performance but the system explicitly tolerates or mitigates bounded latency overruns without immediate safety consequence. A method is classified as *cognitive* when its outputs are episodic, advisory, or asynchronous, intended to be consumed through a faster checked layer rather than sent directly to vehicle actuators. A method is classified as *not demonstrated* when the primary source reports insufficient latency, deployment, or timing evidence to support any of the three preceding classes; this is not a negative judgment but an honest record of what the evidence supports. The boundaries among classes are the authors' proposed operational taxonomy and are not normative requirements of ISO 26262 or any other standard.

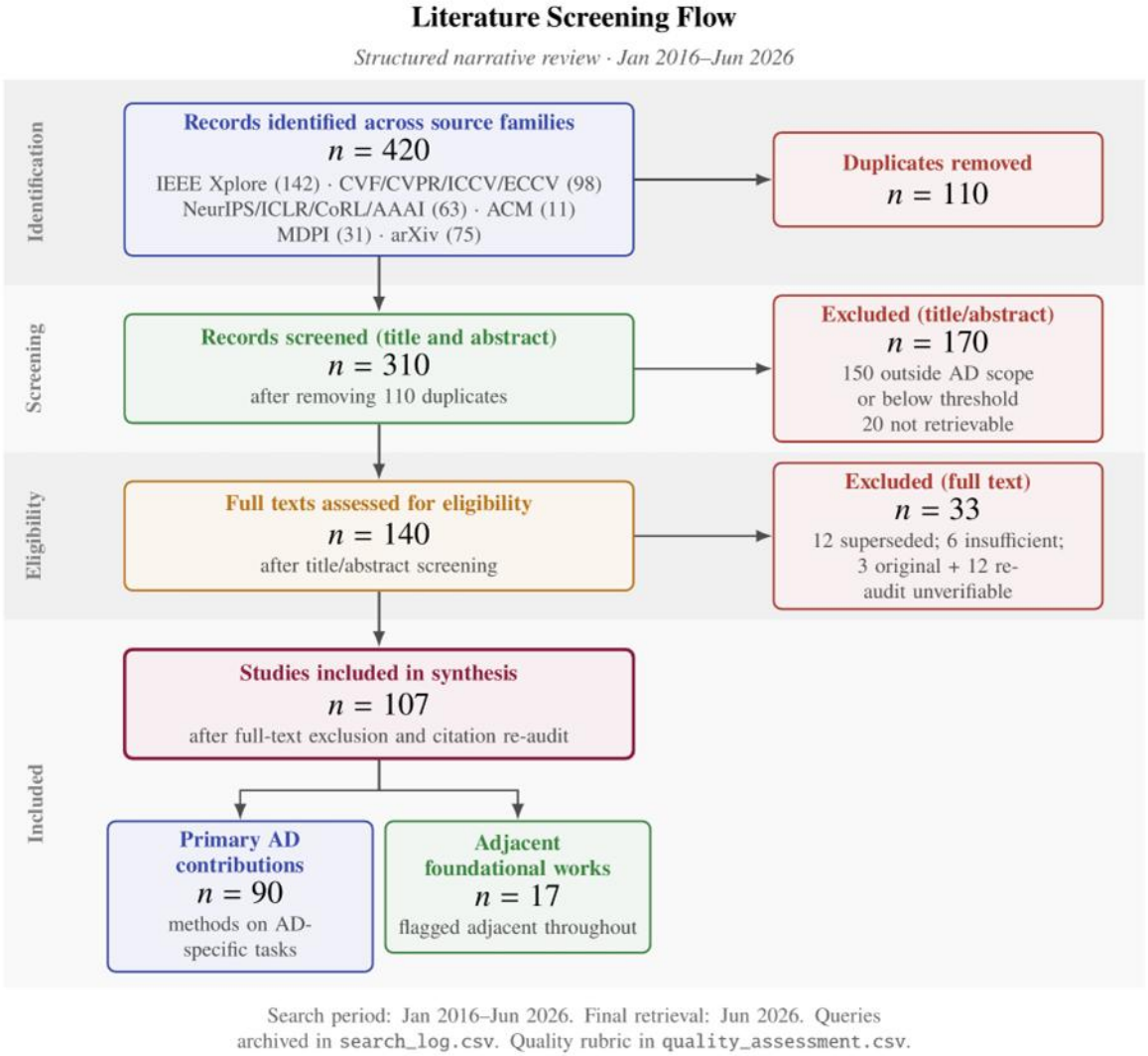


Fig. 2. Literature screening flow for the structured narrative review. Records identified through nine searches were deduplicated, screened on title and abstract, assessed in full text, and citation-audited before inclusion. The retained corpus contains 90 AD contributions and 17 adjacent foundational sources.

Figure 3 visualizes the three pillars together with the real-time class axis, and Table I summarizes the same taxonomy in tabular form: the three pillars, the core question each addresses, representative papers, and the primary limitation each addresses. Figure 4 arranges the surveyed methods by architectural layer, from infrastructure and perception to runtime safety and deployment.

D. Comparison with Existing Surveys

Several recent surveys address overlapping territory but treat the three pillars as separate topics rather than as interacting constraints on a single deployable system. Surveys on broad AD architecture [1] and end-to-end driving [2] establish strong foundations for the modular and learning-based stacks respectively, but predate or do not

centrally address foundation-model reasoning. Surveys on LLMs for AD [4], [5] develop detailed reasoning taxonomies but treat trajectory refinement and hardware deployment as secondary concerns. Surveys on VLMs and VLA models [6], [7] cover multimodal reasoning and action generation but typically do not address the runtime-class placement problem or functional-safety implications. Hardware-efficiency surveys [14] provide rigorous compression and deployment analysis but do not connect these results to foundation-model reasoning or trajectory-refinement architectures. The two-axis taxonomy that maps each method by functional pillar and by runtime class is the distinguishing contribution here: it answers RQ1–RQ6, and the accompanying ADAS deployment analysis grounds the abstract architecture in timing-bounded, safety-governed practice. Table II records the specific coverage gaps this survey addresses relative to the most closely related works.

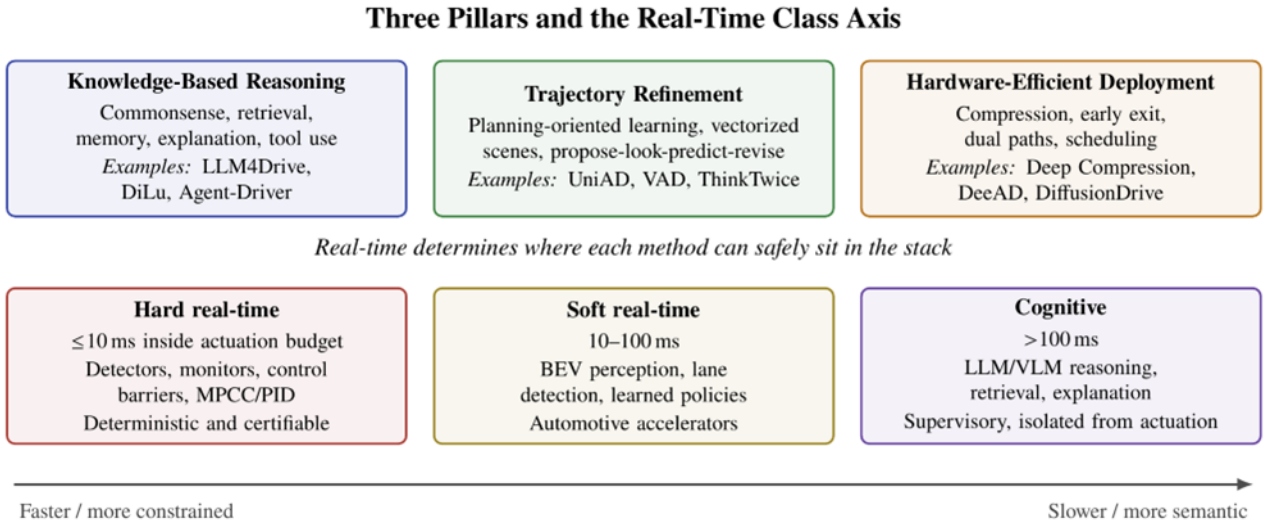


Fig. 3. The three pillars of modern autonomous driving and the orthogonal runtime-evidence axis. Each pillar constrains the next, while reported timing and integration evidence limits the runtime placement that can be supported.

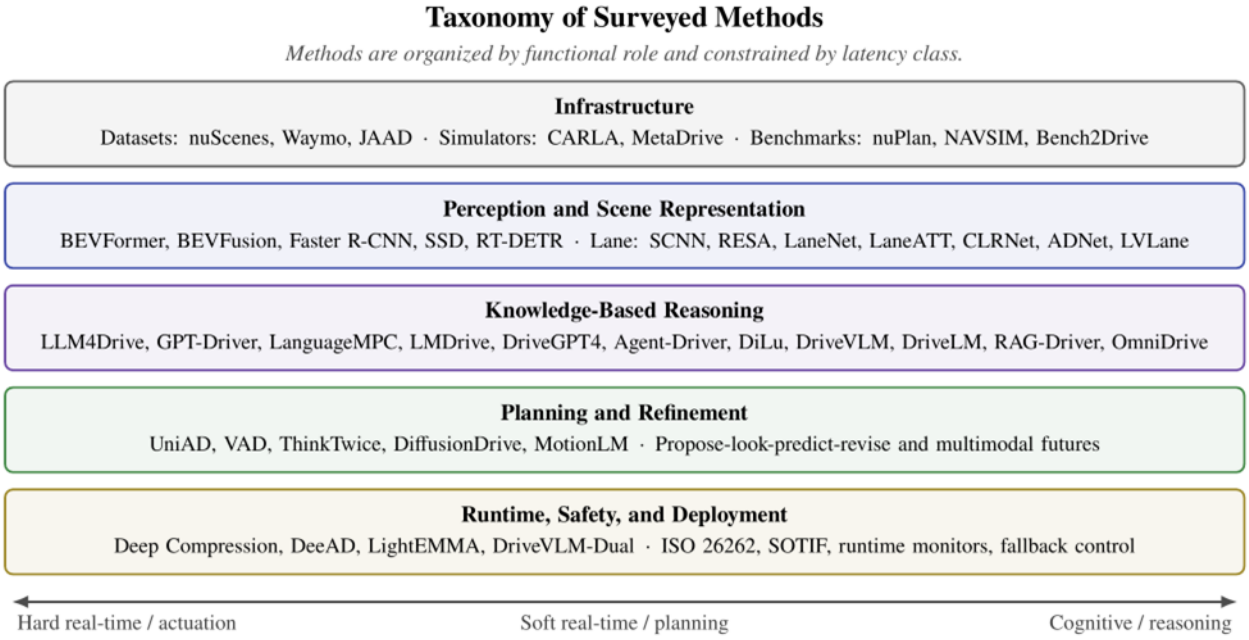


Fig. 4. Taxonomy of surveyed methods across infrastructure, perception, reasoning, planning/refinement, runtime safety, and deployment. The figure emphasizes that modern AD systems combine architectural layers rather than replacing them with a single paradigm.

TABLE I. THREE-PILLAR TAXONOMY OF MODERN AUTONOMOUS DRIVING.

Pillar	Core question	Representative papers	Primary limitation addressed
Knowledge-based reasoning	How can AD reason beyond fixed rules and frequent training scenarios?	LLM4Drive, DiLu, Agent-Driver, DriveVLM, DriveLM, RAG-Driver, OmniDrive	Long-tail understanding, commonsense, explanation, instruction following

Pillar	Core question	Representative papers	Primary limitation addressed
Trajectory planning and refinement	How can candidate trajectories be generated, checked, revised, and evaluated?	UniAD, VAD, ThinkTwice, DiffusionDrive, MotionLM, nuPlan, NAVSIM, Bench2Drive	One-shot planning, weak closed-loop evaluation, multimodal futures
Hardware-efficient deployment	How can complex reasoning and planning fit onboard real-time constraints?	Deep Compression, DeeAD, LightEMMA, VAD, DiffusionDrive, DriveVLM-Dual	Latency, memory, energy, architectural feasibility

TABLE II. COMPARISON WITH EXISTING SURVEYS AND ADJACENT SURVEY STREAMS.

Survey / stream	Main coverage	Strength	Gap addressed by this survey
Yurtsever et al. [1]	Broad AD practices, architectures, datasets, and emerging technologies.	Strong foundation for traditional modular AD stacks.	Does not center LLM/VLM/VLA reasoning, trajectory refinement as a reasoning-control interface, or real-time class placement.
Chen et al. [2]	End-to-end autonomous driving, challenges, and frontiers.	Strong treatment of planning-oriented end-to-end learning.	Less focused on hardware-constrained foundation-model reasoning and ADAS deployment constraints.
LLM4Drive [5]	LLMs for autonomous driving.	Strong LLM-for-AD taxonomy.	Less focused on trajectory refinement, runtime safety layers, and ADAS/ADS testing.
Human-like LLM AD survey [4]	LLMs for human-like autonomous driving.	Highlights common sense, reasoning, and deployment cost.	This paper integrates those concerns with planning refinement and timing-class architecture.
VLM/VLA survey stream [6], [7]	Vision-language and vision-language-action models for AD.	Covers multimodal reasoning and action generation.	This paper emphasizes layered runtime assurance, testing, and hard/soft/cognitive placement.
Edge AI / hardware-efficiency surveys [14], [15]	Compression, approximate computing, edge deployment, accelerators.	Strong hardware and energy lens.	Usually does not connect hardware efficiency to LLM/VLM reasoning, trajectory refinement, and functional-safety placement.

III. FOUNDATIONS FOR MODERN AUTONOMOUS DRIVING

A. Neural and Foundation-Model Building Blocks

Modern AD inherits its modeling capacity from deep backbones and sequence models. Residual learning reformulated network layers to learn residual functions with reference to layer inputs, easing optimization of very deep networks and enabling models up to 152 layers, with an ensemble achieving 3.57% ImageNet error [17]. The Transformer dispensed with recurrence and convolution in favor of attention, providing a scalable mechanism for aggregating information across space, sensors, and time [18]. Both are visible throughout AD: residual networks remain common in camera and LiDAR perception, while attention now appears in bird's-eye-view (BEV) perception, multi-agent prediction, and reasoning and planning decoders.

Large foundation models supply the general-purpose reasoning substrate for knowledge-based AD. GPT-4 is a large-scale multimodal model that accepts image and text inputs and exhibits strong performance across professional and academic benchmarks [19]. LLaMA showed that strong foundation language models can be trained efficiently and released openly, enabling academic reproducibility and on-device adaptation [20]. Two general agent techniques are especially important for driving. ReAct interleaves reasoning traces with actions and observations, letting a model induce and update plans while gathering information from external sources, reducing hallucination relative to pure chain-of-thought [21]. Toolformer showed that language models can teach themselves to call external tools through self-supervised API use [22]. A broader survey of LLM-based autonomous agents formalizes this pattern using memory, planning, action, and interaction with the environment [23]. On the multimodal side, generalized decoding frameworks that jointly handle pixel-, image-, and language-level outputs provide the vision-language grounding that driving VLMs rely upon [24].

B. Perception, Datasets, and Simulators as Infrastructure

Reasoning and refinement still depend on robust perception. BEVFormer learns unified BEV representations from multi-camera images using spatiotemporal transformers, with grid-shaped BEV queries aggregating spatial features across views via spatial cross-attention and temporal self-attention recurrently fusing history; it achieved 56.9% NDS on the nuScenes test set, 9.0 points above the prior best and on par with LiDAR baselines, while improving velocity estimation and low-visibility recall [25]. BEVFusion unifies camera and LiDAR features in a shared BEV space that preserves both geometry and semantics, is task-agnostic across 3D perception tasks, and, after optimizing the view-transformation bottleneck with BEV pooling that cuts latency by more than 40×, sets a new nuScenes state of the art with higher detection mAP/NDS and substantially higher segmentation mIoU at lower cost [26]. Multimodal-fusion surveys organize the broader field around what, when, and how to fuse, noting the absence of a general design guideline [27].

General object and lane detection supply structured inputs for planners. Faster R-CNN introduced a Region Proposal Network sharing convolutional features for near-real-time detection [28]; SSD discretized bounding boxes into multi-scale default boxes for single-pass detection [29]; and RT-DETR demonstrated that a real-time detection transformer can match or beat YOLO models on the speed–accuracy trade-off [30]. Lane detection has its own dense literature reflecting the difficulty of parsing long, thin, weakly-textured structures: SCNN generalized convolution to slice-by-slice message passing across rows and columns [31]; RESA propagated spatial information with a recurrent

feature-shift aggregator [32]; LaneNet targeted real-time detection [33]; CondLaneNet used conditional convolution with a recurrent instance module for complex topologies [34]; LaneATT introduced anchor-based attention for global context [35]; CLRNet combined cross-layer refinement with a Line IoU loss that regresses each lane as a whole [36]; ADNet decomposed anchors for flexible shape prediction [37]; and LVLane extended the task to joint detection and classification in challenging conditions [38].

Data and simulation are equally critical infrastructure. nuScenes was among the first datasets to carry the full sensor suite (cameras, LiDAR, and radar) with extensive annotation [39]. The Waymo Open Dataset emphasizes scale and diversity, with 1,150 twenty-second scenes of synchronized, calibrated LiDAR and camera data, exhaustively annotated with consistent 2D and 3D boxes [40]. JAAD contributes pedestrian joint-attention and crossing-behavior annotations [41]. CARLA provides an open-source urban simulator with configurable sensors and scenarios, used to compare modular, imitation-based, and RL-based driving [42], and MetaDrive composes diverse scenarios to study generalization of RL policies to unseen environments [43].

IV. Knowledge-Based Reasoning for Autonomous Driving

A. From Rule-Based Pipelines to Knowledge-Grounded Driving

Classical pipelines are interpretable but brittle because their rules must explicitly encode edge cases; learning-based systems improve data-driven behavior but still fail under distribution shift. LLMs and VLMs open a third path: knowledge-grounded driving that supplies commonsense reasoning, language interaction, memory retrieval, and reflective planning to the AD stack. DiLu introduces a knowledge-driven approach that gives an LLM access to a knowledge base of driving rules and an experience pool of prior decisions, enabling a reasoning module to generate and reflect on candidate actions using chain-of-thought prompting, with a memory module that refines knowledge over episodes [8]. DriveLM extends vision-language grounding through graph-structured visual question answering that models perception, prediction, and planning as a connected reasoning graph over key objects, showing that reasoning about object interdependencies, rather than treating questions independently, improves planning outcomes [44]. OmniDrive provides a holistic vision-language dataset with counterfactual reasoning questions, enabling VLMs to answer not just what happened but what would have happened under alternative actions [45].

B. LLMs as Planners and Decision Makers

A first design pattern uses the LLM directly in planning. GPT-Driver reformulates motion planning as a language-modeling problem, representing planner inputs and trajectory outputs as language tokens and achieving competitive results relative to rule-based and learned baselines on nuScenes [46]. LanguageMPC treats an LLM as the high-level decision maker that sets behavioral parameters (target speed, lane preference, aggressiveness) for a model predictive control (MPC) layer that executes safe trajectories; the LLM determines what to do while MPC enforces what is feasible [47].

A second pattern is language-guided closed-loop driving. LMDrive integrates multimodal sensor data with natural-language instructions for closed-loop driving in CARLA, releasing roughly 64K instruction-following data clips to support fine-tuning [48]. DriveGPT4 provides an interpretable end-to-end driving system that interprets multiple-image inputs, answers driving-related questions, and predicts vehicle control signals jointly, connecting visual grounding to both verbal explanation and executable action [49].

C. LLM Agents: Memory, Tool Use, Retrieval, and Reflection

A third pattern treats the LLM as the cognitive core of an agent. Agent-Driver equips an LLM with a tool library accessible via function calls, a cognitive memory of commonsense and experiential knowledge, a reasoning engine for chain-of-thought decision generation, and a reflection module that critiques and revises plans against driving principles; on nuScenes planning metrics it outperforms prior methods while providing natural-language rationales [9]. RAG-Driver applies retrieval-augmented in-context learning, retrieving demonstrations from a driving experience pool and prompting a multimodal LLM to generate generalizable explanations of control signals and speed predictions; the retrieval component improves generalization to new environments beyond the training distribution [10].

LLMs can also guide rather than replace learning. LLM-guided deep reinforcement learning uses an LLM expert inside the RL loop, with an expert-policy-constrained algorithm and an LLM-intervened interactive environment, enabling the RL agent to benefit from semantic guidance while retaining the adaptability of learned policies [50].

D. Vision-Language Reasoning and Counterfactual Understanding

VLM-based systems connect visual scene understanding to semantic reasoning and planning. DriveVLM builds a VLM-based stack with scene description, scene analysis, and hierarchical planning, and its dual-system variant explicitly separates a slow VLM reasoning path from a fast traditional planning pipeline, acknowledging that VLM latency precludes direct actuation at driving speeds [51]. Table III summarizes the roles that language and vision-language models play in the surveyed systems.

TABLE III. Roles of language and vision-language models in autonomous driving.

LLM/VLM role	Representative systems	Suitable time scale	Deployment caution
Semantic planner	GPT-Driver, LanguageMPC	Cognitive / low-frequency supervisory	Must not directly replace hard real-time control without safety isolation

LLM/VLM role	Representative systems	Suitable time scale	Deployment caution
Closed-loop policy conditioner	LMDrive, DriveGPT4	Soft real-time if compressed; otherwise cognitive	Needs closed-loop validation and fallback behavior
Memory & reflection agent	Agent-Driver, DiLu	Cognitive / episodic updates	Memory retrieval can surface stale or irrelevant examples
Retrieval / explanation module	RAG-Driver, DriveLM	Cognitive / supervisory	Explanations must be faithful to actual planner state
Scenario generator	ChatScene	Offline validation	Generated scenarios need simulator-valid code and coverage metrics
Training-time teacher	LLM-guided DRL	Offline training	Teacher bias can be distilled into smaller policies

LLMs are also useful for safety-critical scenario generation. ChatScene uses an LLM agent to convert natural-language descriptions of dangerous scenarios into domain-specific simulator code for CARLA, enabling diverse and semantically meaningful safety-critical test cases to be generated at scale [52].

V. Trajectory Planning and Refinement

A. From Direct Steering to Command-Conditioned Learned Control

Early end-to-end driving systems demonstrated that neural networks could map raw sensor observations directly to low-level control. PilotNet, for example, learned steering commands from front-facing camera images using a convolutional network trained with supervised imitation, demonstrating feasibility for highway lane keeping [53]. Conditional Imitation Learning (CIL) addressed part of this limitation by conditioning the driving policy on high-level navigational commands such as "turn left," "turn right," or "go straight" [54]. Reinforcement learning and imitation-learning surveys further show that learned policies can enter the autonomous-driving stack at multiple levels, including behavior selection, trajectory generation, and low-level control [55], [56]. Hybrid approaches that combine RL adaptation with classical controllers have also been shown to extend stability to unseen urban scenarios [57].

Accordingly, the relevance of early imitation and reinforcement learning is not only historical. These methods establish the design pattern that later knowledge-grounded systems inherit: separate high-level intent (command, goal, language instruction) from low-level execution (trajectory generation, control), and ensure that the high-level component is commanded and interpretable while the low-level component is robust and fast.

B. Planning-Oriented End-to-End Driving

Planning-oriented AD reframes perception and prediction as upstream tasks optimized for safe planning. UniAD jointly models tracking, mapping, motion forecasting, occupancy prediction, and planning in a unified transformer architecture, using query-based attention mechanisms and showing that joint optimization of constituent tasks leads to better planning than pipelining independent models [3]. VAD adopts a complementary strategy, representing the driving scene as a collection of vectorized instances (agents and map elements) rather than dense grids, using instance-level queries with explicit interaction modeling to achieve a substantial reduction in collision rate while operating substantially faster than prior art [11]. MotionLM reformulates multi-agent motion forecasting as a language-modeling problem, jointly predicting the future trajectories of all agents using discrete motion tokens and autoregressive decoding, and achieving top performance on the Waymo Open Motion Dataset interactive challenge [58].

C. Multi-Stage Trajectory Refinement as the Interface Between Reasoning and Control

Trajectory refinement is the mechanism that makes high-level intent usable in safety-critical driving. A planner that emits a single trajectory in one pass may fail when local evidence is ambiguous, when the first estimate conflicts with predicted agent behavior, or when the gap between high-level semantic instruction and geometric feasibility is large. Multi-stage refinement addresses this by iteratively revising a candidate trajectory using richer representations, additional context, or explicit feedback from a verifier.

ThinkTwice is a representative example of this principle [12]. Rather than relying only on a stronger encoder, it increases decoder capacity through cascaded refinement modules. Its Look-Twice module and subsequent Zoom-In module both use a coarse trajectory estimate as an attention anchor to re-query BEV features in a focused neighborhood, correcting errors that a single-pass decoder misses. The key insight is that the first trajectory estimate defines a relevant subspace for further feature extraction, and querying that subspace yields a more accurate second estimate.

This idea is central to knowledge-grounded autonomous driving. An LLM or VLM may produce an instruction such as "merge behind the truck," "yield to the pedestrian," or "take the right lane after the construction zone." These instructions must be converted into a trajectory that is geometrically consistent, dynamically feasible, and safe under the predicted behavior of nearby agents. A refinement layer serves exactly this interface role: it takes a coarse candidate anchored to the semantic instruction and iteratively improves it against local evidence.

For ADAS and automated driving system (ADS) testing, trajectory refinement also provides a natural test target. A useful test case should not only ask whether the system chooses the correct high-level behavior; it should ask whether the resulting trajectory, after refinement, meets safety and comfort criteria. Refinement failures often reveal subtle planning errors that single-pass evaluations miss, making staged refinement both a design tool and a testing lens.

D. Generative Planning and Motion Forecasting

Driving futures are inherently multimodal. DiffusionDrive applies a truncated diffusion policy with prior multi-mode anchors, learning to denoise from an anchored Gaussian distribution to a multi-mode trajectory distribution in substantially fewer denoising steps than standard diffusion, achieving 88.1 PDMS at 45 frames per second (FPS) on NAVSIM [13]. Trajectory prediction surveys and pedestrian-trajectory-prediction reviews confirm that generative and attention-based approaches now dominate the prediction stack, with social force models [59] giving way to deep social models, transformers, and context-aware networks [60], [61]. Tables IV and V compare representative methods, first by architecture and reasoning role, then by evaluation, strengths, limitations, and testing relevance.

TABLE IV. REPRESENTATIVE METHOD COMPARISON: ARCHITECTURE AND REASONING (PART I OF II).

Method	Year	Core idea	Input modality	LLM/VLM usage	Planning role
PilotNet	2016	Direct image-to-steering imitation	Camera	None	Direct control
CIL	2018	Command-conditioned imitation	Camera + route command	None	Conditional control
LanguageMPC	2023	LLM sets decision parameters for MPC	Scene state + language	High-level decision maker	Supervisory planning
Agent-Driver	2024	LLM agent with tools, memory, reasoning	Structured scene inputs	Cognitive agent	Decision/planning support
RAG-Driver	2024	Retrieval-augmented driving explanations	Multimodal inputs + retrieval	Explanation/reasoning	Supervisory
ThinkTwice	2023	Cascaded trajectory decoder refinement	BEV features	None directly	Refinement layer
VAD	2023	Vectorized scene representation	Agents + map vectors	None	Efficient planning
DiffusionDrive	2025	Truncated diffusion for multimodal planning	Camera/BEV features	None directly	Generative planning
DriveVLM-Dual	2024	Slow VLM + fast pipeline	Vision + language	VLM reasoner	Supervisory / dual system
DeeAD / LightEMMA	2025	Efficient VLA/early-exit multimodal models	Multimodal	VLA efficiency	Reasoning/action efficiency

TABLE V. REPRESENTATIVE METHOD COMPARISON: EVALUATION, STRENGTHS, LIMITATIONS, AND TESTING RELEVANCE (PART II OF II).

Method	Eval. / metric	Strength	Limitation	ADAS/ADS testing relevance
PilotNet	Steering prediction	Simple end-to-end baseline	Not commandable or interpretable	Historical baseline for end-to-end control
CIL	CARLA-style driving tasks	Separates command from control	Weak under distribution shift	Tests command-following behavior
LanguageMPC	Driving-task performance	Keeps control feasible through MPC	LLM latency and safety limits	Useful for ACC-style behavior tuning
Agent-Driver	nuScenes planning metrics	Interpretability and few-shot behavior	Cognitive latency and certification risk	Tests rare-scene reasoning
RAG-Driver	Explanation generalization	Adapts via retrieved examples	Retrieval may be stale or irrelevant	Useful for explainable test reports
ThinkTwice	Planning metrics	Revises coarse plans using local evidence	Adds decoder complexity	Planner stress tests
VAD	Collision rate, speed	Efficient explicit constraints	Depends on vector quality	Real-time planning evaluation
DiffusionDrive	PDMS, FPS	Diverse plans with reduced denoising	Generative reliability needs validation	Multimodal-future testing
DriveVLM-Dual	Deployment-oriented evaluation	Acknowledges latency split	VLM cannot directly actuate	Runtime architecture example
DeeAD / LightEMMA	Accuracy + compute cost	Treats efficiency as metric	Needs verification evidence	Hardware-aware testing

VI. HARDWARE EFFICIENCY AND THE REAL-TIME CLASS AXIS

The third pillar is deployment feasibility. LLM reasoning, VLM scene analysis, BEV perception, diffusion planning, and closed-loop simulation all increase computational demand, while onboard compute is bounded by vehicle power budgets, thermal constraints, cost targets, and qualification requirements. Hardware efficiency is therefore not an optional add-on but a prerequisite for deployment, and the real-time class axis formalizes exactly where each method can safely operate.

A. Real-Time Class Definitions

We propose operational classes for organizing methods by runtime role. **Hard real-time** placement requires a

deadline derived from a system or safety requirement, together with evidence about worst-case execution or scheduling; mean latency or frames per second alone is insufficient. A missed deadline in this class can violate the associated safety or control requirement. **Soft real-time** placement requires a documented update requirement and a stated mechanism by which bounded overruns can be tolerated or mitigated without immediate safety consequence. **Cognitive** methods produce episodic, advisory, or asynchronous outputs that are consumed by a faster checked layer rather than sent directly to actuators. Methods lacking the necessary timing and integration evidence are labeled **not demonstrated**. Euro NCAP protocols define automatic emergency braking test scenarios and performance outcomes, but do not establish a universal internal perception-to-actuation deadline for every implementation [62]. The classes are therefore an analytical taxonomy, not normative requirements or universal millisecond bands.

B. *Efficiency as a Design Principle and Runtime Architecture Constraint*

Hardware efficiency is not a post-processing step applied after model design. It is a design constraint that determines whether a method can be placed in the hard real-time, soft real-time, or cognitive class and therefore whether it can serve a given function in the stack. A model that achieves state-of-the-art accuracy but cannot meet timing requirements is not deployable in that position, regardless of its benchmark score.

Four efficiency strategies are especially relevant. First, model compression reduces the storage and memory bandwidth required by large networks. Deep Compression combines pruning, trained quantization, and Huffman coding to compress AlexNet by 35 × and VGG-16 by 49 × without accuracy loss [14]. Second, early-exit architectures allow a model to produce an output at an intermediate layer when confidence is sufficient, reducing average latency without retraining. DeeAD exploits this principle for VLA models in driving [15]. Third, vectorized representations replace dense spatial maps with compact agent and map element vectors, lowering memory and compute per inference step [11]. Fourth, hardware-aware model design incorporates latency, memory, and energy targets from the start of model development rather than applying compression post hoc [63]. LightEMMA similarly demonstrates that lightweight end-to-end multimodal models can approach the accuracy of larger models with substantially lower compute cost [64].

Figure 5 presents efficiency as a proposed layered runtime-assurance architecture rather than as a simple stack. The cognitive layer contains LLM/VLM/VLA reasoning, retrieval, explanation, and scenario generation. The soft-real-time layer contains perception, prediction, and trajectory planning and refinement, and consumes cognitive-layer outputs as goals or constraints. The hard-real-time layer contains safety monitoring, emergency response, and low-level control under requirement-derived deadlines. The architecture is a design principle synthesized by the authors; it does not by itself establish safety or certifiability.

C. *Hardware Evidence: Reported Platforms, Latency, and Runtime Placement*

Table VI consolidates hardware and latency evidence drawn directly from primary sources across the surveyed literature. A central finding is that most LLM- and VLM-based methods report no measured latency on automotive-grade hardware and therefore remain *not demonstrated* on the runtime axis. Classical controllers and detectors provide useful implementation measurements, but a real-time label, mean FPS, or an onboard demonstration is not by itself evidence of a requirement-derived deadline or worst-case schedulability. Relative efficiency gains, such as the 29% latency reduction reported for DeeAD [15] and the 40 × BEV-pooling speedup reported for BEVFusion [26], are likewise insufficient for deployment decisions without an absolute baseline on target hardware. This pattern—strong benchmark accuracy with absent or incomplete timing conditions—is the most consistent methodological shortfall in the surveyed literature.

VII. ADAS Deployment Analysis: ACC, AEB, and LKA

Adaptive cruise control (ACC), automatic emergency braking (AEB), and lane keeping assistance (LKA) are commercially deployed Level 1–2 ADAS functions. They are narrower than full autonomy, but they make the reasoning–refinement–deployment argument concrete because each implementation must derive timing requirements, sensor budgets, and safety obligations from its item definition and operating context. They also represent the AI stack that must be extended, not replaced, to reach higher automation levels.

TABLE VI. Hardware and latency evidence from primary sources. NR = not reported in the primary source. Measured throughput or a real-time claim does not establish hard-real-time suitability without a requirement-derived deadline and worst-case timing evidence.

Method	Year	Task	Platform	Opt. / precision	Latency / FPS	RT class
Reported implementation timing; hard-real-time evidence incomplete						
ACC + Fuzzy [65]	2006	ACC stop-and-go	Real vehicle ECU	Fuzzy logic	Real-time loop	Not demonstrated
CACC Platoon [66]	2011	Cooperative ACC	3-vehicle ECU	Classical + V2V	Real-time V2V	Not demonstrated
SSD-300 [29]	2016	2D detection	Titan X GPU	Standard	59 FPS	Not demonstrated
Faster R-CNN [28]	2017	2D detection	K40 GPU	Standard	17 FPS	Not demonstrated
RT-DETR-R18 [30]	2024	2D detection	NVIDIA T4	Standard	114 FPS	Not demonstrated

Method	Year	Task	Platform	Opt. / precision	Latency / FPS	RT class
DAVOS [63]	2026	AV OS scheduling	Automot. ECU (target)	OS-level scheduling	Arch. proposal only; NR	Not demonstrated
Throughput or relative-latency evidence; integration deadlines unreported						
LaneATT [35]	2021	Lane detection	Unspecif. GPU	Anchor attention	535 FPS	Not demonstrated
YOLO-AV Edge [67]	2022	Pedestrian det.	NVIDIA Xavier AGX	Lightweight arch.	45 FPS	Not demonstrated
Diffusion-Drive [13]	2025	E2E planning	Unspecif. GPU	10× fewer diffusion steps	45 FPS	Not demonstrated
DeeAD [15]	2025	VLA driving	Unspecif. GPU	Early-exit (28% layer sparsity)	−29% vs. full VLA	Not demonstrated
Design-time efficiency — parameter reduction, no inference timing reported						
Deep Compression [14]	2016	Model compression	ASIC/GPU (target)	Pruning + quantization + Huffman coding	35× AlexNet; 49× VGG-16 (no acc. loss)	Design-time
Cognitive — inference latency on vehicle hardware NR in primary source						
UniAD [3]	2023	E2E planning	8× A100 (train)	None	NR on vehicle	Not demonstrated
Language MPC [47]	2023	Supervisory planning	GPT-4 API (cloud)	None	NR on vehicle	Cognitive
DriveVLM-Dual [51]	2024	Planning	Unspecified	Dual-system design	Acknowledged; NR	Cognitive

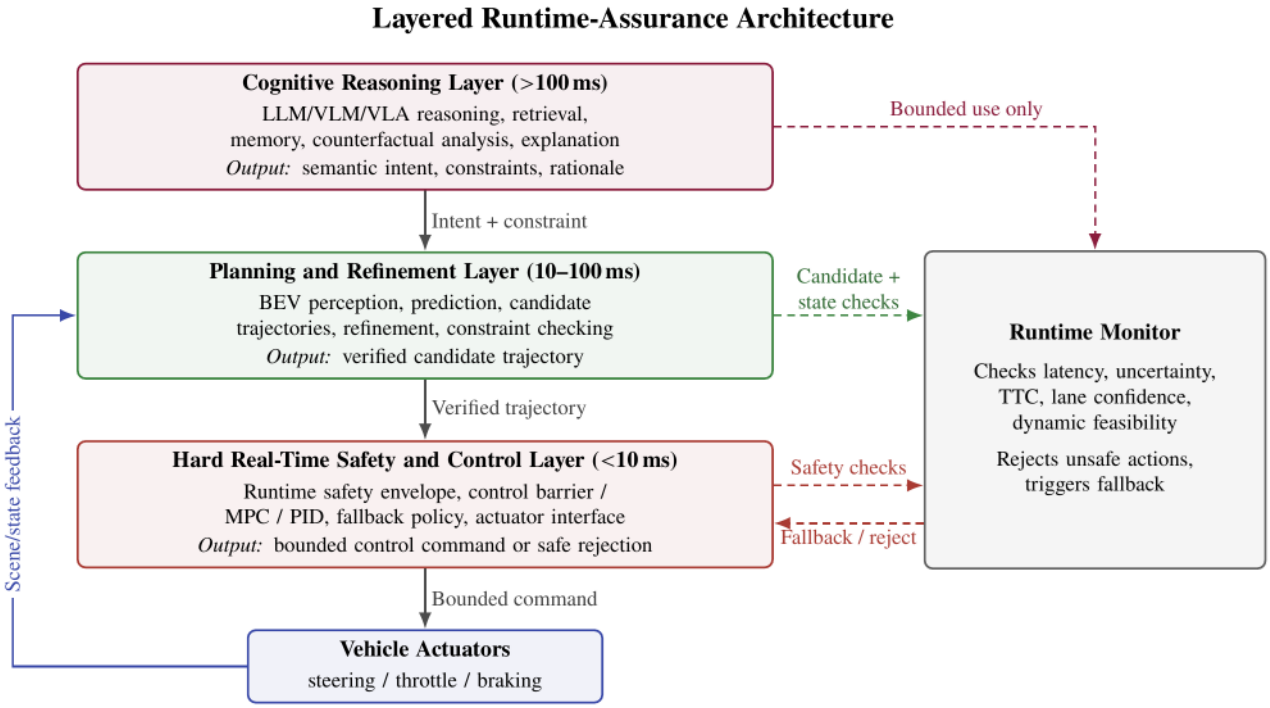


Fig. 5. Proposed layered runtime-assurance architecture for knowledge-grounded autonomous driving. Cognitive reasoning produces semantic intent and explanations but is isolated from direct actuation. A time-sensitive refinement layer converts intent into checked trajectory candidates, while a requirement-bounded safety-and-control layer retains actuation authority.

A. Adaptive Cruise Control

ACC is primarily a longitudinal control problem. The vehicle must estimate the relative distance and velocity of a lead vehicle and choose acceleration or braking commands that preserve comfort and safe headway. Classical ACC uses radar-based ranging with PID or model predictive control [66]. Cooperative ACC extends this to vehicle-to-vehicle communication, requiring reliable data exchange and tolerance of data loss [68]. Reinforcement learning has been applied to ACC to achieve adaptive behavior in stop-and-go traffic without explicit rule encoding, with safety constraints imposed through reward shaping and constraint-based RL [69].

A concrete scenario illustrates the layered design. Suppose the ego vehicle is traveling at highway speed and another vehicle cuts in from the adjacent lane. A purely reactive controller may brake sharply. A time-sensitive prediction module can anticipate the cut-in trajectory and issue a smoother deceleration command. A cognitive LLM layer could recognize the highway merge pattern, recall from memory that aggressive merges at this location are common, and suggest to the checked planner that a larger

headway should be maintained proactively. The LLM does not command the brake directly; it updates a parameter consumed by the faster planning cycle.

For ADAS testing, ACC should be evaluated with scenarios that vary cut-in speed, lead-vehicle deceleration, sensor noise, road curvature, and weather. The key question is not merely whether the vehicle avoids collision but whether headway recovery is smooth, jerk is bounded, and the system degrades gracefully when sensor data is lost or degraded.

B. Automatic Emergency Braking

AEB illustrates why foundation-model reasoning should not be placed directly in an actuation path without requirement-specific timing and assurance evidence. An AEB implementation must detect collision risk and command braking within deadlines derived from its item definition, operating scenario, sensing pipeline, and actuator dynamics. No universal 150 ms allocation can be inferred from the function name alone. Because the surveyed LLM/VLM studies generally omit worst-case automotive-platform latency, their suitability for this role is not demonstrated.

Consider an urban scenario in which a child emerges from behind a parked van. A VLM may be useful offline or in a supervisory mode: it can explain that the occlusion was hazardous, generate similar test scenarios for the simulator, or help a safety analyst understand why the system failed. The braking decision, however, must come from a perception, threat-assessment, and control pipeline shown to meet the implementation's requirement-derived deadlines. Detectors such as Faster R-CNN and SSD illustrate candidate perception architectures, but their workstation-GPU FPS results do not establish suitability for an AEB actuation path without target-platform profiling, worst-case analysis, and system-level assurance [28], [29]. The VLM layer sits above and beside this pipeline, not inside it.

For ADAS/ADS testing, AEB scenarios should include occluded pedestrians, sudden lead-vehicle braking, false-positive objects, reflective surfaces, low-light scenes, and adversarial sensor artifacts. The time-to-collision at which the system activates, the false-positive rate, and the braking deceleration profile are all measurable pass/fail criteria.

C. Lane Keeping Assistance

LKA is a lateral-control problem that depends on lane perception, vehicle pose, road curvature, and steering correction. Its implementation is time-sensitive, but its runtime class must be derived from item-specific update, overrun, and fallback requirements. Deep lane detectors improve robustness over classical edge-based methods, especially under shadows, rain, and faded markings, but they add inference latency that must be budgeted alongside control-loop frequency requirements [36], [38]. Time to line crossing has been analyzed as a theoretical metric for departure prediction and avoidance timing [65].

A concrete example is a construction zone in which faded white lane markings conflict with orange temporary lane guidance. A lane detector may produce uncertain or unstable lane estimates. A VLM could be used in a cognitive supervisory role to recognize the construction-zone context, identify which markings are authoritative, and issue a high-level constraint to a faster lane-processing component: "prefer orange cones over white lines." The faster component would still require its own timing and fallback evidence; the VLM-supplied prior does not establish that evidence.

For testing, LKA should be evaluated using faded lanes, lane splits, lane merges, shadows, rain, construction cones, and map-lane conflicts. These scenarios directly test the interaction between perception confidence and control stability, and they can be generated systematically using LLM-based scenario generation tools such as ChatScene [52].

D. Deployment Lessons from the ADAS Deployment Analysis

ACC, AEB, and LKA illustrate that the same AI method can be appropriate in one runtime role and unsupported in another. LLM/VLM reasoning can help set ACC behavior parameters, generate AEB test scenarios, explain LKA decisions for a safety analyst, and update planner priors for construction zones, but it cannot replace a perception-and-control path whose timing and safety evidence has been derived for the specific AEB item. Table VII summarizes the proposed runtime role and design implication for each function. Table VIII maps each function to a representative test scenario.

TABLE VII. ADAS FUNCTIONS AS A REAL-TIME DEPLOYMENT ANALYSIS.

Function	Sensor emphasis	Proposed runtime role	Best AI role	Design recommendation
ACC	Long-range radar plus camera/LiDAR context	Requirement-specific control; optional cognitive supervision	RL for adaptation; LLM for behavior parameters; MPC/PID for execution	Use LLM/RL to set intent or parameters; retain constrained control for actuation
AEB	Camera/LiDAR/radar for hazard detection and TTC	Requirement-bounded safety path	Qualified detection and prediction; LLM/VLM only supervisory/offline	Keep generative reasoning out of the direct braking path
LKA	Forward camera lane perception; optional map/BEV context	Requirement-specific control; optional cognitive supervision	Lane-specific deep networks; VLM for complex topology	Learned perception with confidence monitoring and classical lateral fallback

TABLE VIII. ADAS FUNCTION-TO-TEST-SCENARIO MAPPING.

Function	Representative scenario	Primary failure mode	Timing class	Suggested test metric

Function	Representative scenario	Primary failure mode	Timing class	Suggested test metric
ACC	Highway cut-in with variable gap and relative speed	Late or overly aggressive braking; unstable headway	Requirement-specific control + optional cognitive	Minimum TTC, jerk, headway recovery, comfort
AEB	Occluded pedestrian emergence from behind parked vehicle	Missed hazard, delayed braking, false positive	Requirement-bounded safety path	Reaction time, collision rate, false-positive rate
LKA	Construction zone with faded original lanes and temporary cone guidance	Incorrect lane selection or overconfident steering	Requirement-specific control + optional cognitive	Lane-confidence trace, lateral error, fallback activation

VIII. Benchmarks, Safety, and Verification

A. Closed-Loop Evaluation and Benchmarking

A central methodological insight of recent work is that open-loop evaluation, which replays logged data and computes metrics such as L2 trajectory error against recorded futures, does not reliably reflect closed-loop performance, because it cannot reveal whether a planner recovers safely after its own actions shift the future state. nuPlan was among the first closed-loop, ML-based planning benchmarks, pairing a large dataset with a lightweight closed-loop simulator and planning-specific metrics, explicitly to overcome the inadequacy of open-loop L2-based evaluation for long-term planning [70]. NAVSIM occupies a middle ground, using large real datasets with a non-reactive simulator and gathering simulation-based metrics such as progress and time-to-collision by unrolling BEV abstractions over a short horizon, so the evaluated policy and environment do not influence each other; this yields metrics correlated with closed-loop performance without the full cost of reactive simulation, and provides the PDMS metric used by recent generative planners [13], [71]. Bench2Drive provides a benchmark for evaluating the multiple abilities of end-to-end systems in a fully closed-loop manner, noting that fixed-route CARLA tests suffer high variance and that per-method training data makes fair comparison infeasible, and supplying official training data and a multi-ability protocol [72]. The collective message of nuPlan, NAVSIM, and Bench2Drive is that the evaluation protocol is not a neutral instrument but an active determinant of which research directions appear fruitful.

B. Functional Safety and the Latency Budget

Deployed ADAS functions are developed within functional-safety and intended-functionality processes. ISO 26262 assigns ASILs through item-specific hazard analysis and risk assessment (HARA); the ASIL depends on the item definition, operating context, severity, exposure, and controllability, and cannot be assigned from a function label alone [73]. ISO 21448 addresses hazards arising from functional insufficiencies and reasonably foreseeable misuse, including performance limitations that may appear under distribution shift [74]. Neither standard mandates the layered architecture proposed here or makes a learned component "certifiable" merely because it meets an average-latency target. Our narrower inference is that a cognitive reasoner proposed for an actuation path needs requirement-derived timing evidence, monitored interfaces, and an independently justified fallback or safety mechanism.

C. Adversarial Robustness and Verification

Safety must include adversarial and verification perspectives. Deep networks are vulnerable to small, structured perturbations, first characterized as an intriguing and counterintuitive property of neural networks [75]. For driving the stakes are physical: the first security study of LiDAR-based perception in the AV setting showed that, by formulating spoofing as an optimization problem, an attacker can craft spoofed returns that fool the detector into perceiving obstacles in front of the victim vehicle [76]. Because a reasoning model is only as reliable as the perception and scenario context it receives, robustness must be a first-class design objective. One practical path is architectural: monitor the outputs of reasoning modules, constrain them through planners and controllers, and use runtime checks to reject unsafe actions, with LLM-driven scenario generation such as ChatScene used to stress-test those monitors [52].

D. Representative Quantitative Comparisons

Perception and planning results are evaluated on different benchmarks, sensors, and protocols that are not mutually comparable. Tables IX and X therefore separate perception results (2D and 3D detection, lane detection) from planning and forecasting results. Open-loop displacement error on nuScenes, closed-loop driving score on CARLA, non-reactive simulation score on NAVSIM, and offline forecasting error on Waymo measure different quantities; combining them in a single ranking is methodologically unsound and is deliberately avoided here. Table XI restates the operational rules behind the runtime classes used in this survey. Figure 6 shows how reasoning, planning, closed-loop simulation, and failure mining can be linked into a validation loop for ADAS/ADS testing.

TABLE IX. Perception benchmark results from primary sources. NR = not reported in primary source. FPS values are on the listed platform and are not transferable to automotive-grade embedded hardware without re-profiling.

Method	Year	Task / Dataset	Metric / value	Comparison context	Hardware (FPS)
Faster R-CNN [28]	2017	2D det., VOC 2007	mAP = 76.4	R-CNN: 66.0; Fast R-CNN: 66.9	K40 GPU (17 FPS)
SSD-300 [29]	2016	2D det., VOC 2007	mAP = 74.3	YOLO: 63.4 mAP at 45 FPS	Titan X (59 FPS)
RT-DETR-R18 [30]	2024	2D det., COCO val	mAP50:95 = 53.1	YOLOv8-L: 52.9 mAP at same FPS	NVIDIA T4 (114 FPS)

Method	Year	Task / Dataset	Metric / value	Comparison context	Hardware (FPS)
BEVFormer [25]	2022	Cam. 3D det., nuScenes test	NDS = 56.9 (+9.0 pts)	DETR3D: 49.4 NDS; on par with LiDAR	Wkstn. GPU (NR)
BEVFusion [26]	2023	Cam.+LiDAR 3D det.+seg., nuScenes test	NDS = 72.9; mAP = 70.2	cam-only: 60.2 NDS; 40× BEV pooling speedup	Wkstn. GPU (NR)
CLRNet [36]	2022	Lane det., CULane	F1 = 80.47	LaneATT: 77.02; CondLaneNet: 79.48	NR (NR)
LaneATT [35]	2021	Lane det., CULane	F1 = 77.02	SCNN: 71.60	NR GPU (535 FPS)
YOLO-AV Edge [67]	2022	Pedestrian det., AV dataset	mAP50 = 72.3	YOLOv5s: 70.1 at 50 FPS; YOLOv5x: 75.1 at 18 FPS	Xavier AGX (45 FPS)

TABLE X. PLANNING AND FORECASTING BENCHMARK RESULTS FROM PRIMARY SOURCES. OPEN-LOOP L2 ERROR, CLOSED-LOOP DRIVING SCORE, NON-REACTIVE SIMULATION SCORE, AND OFFLINE TRAJECTORY ERROR ARE DISTINCT METRICS FROM DISTINCT EVALUATION PROTOCOLS AND SHOULD NOT BE RANKED AGAINST EACH OTHER.

Method	Year	Task / Benchmark	Metric / value	Comparison context	Eval. mode
UniAD [3]	2023	Planning, nuScenes val	L2 at 3 s = 1.03 m; collision = 0.31%	ST-P3: L2 = 1.33 m	Open-loop
ThinkTwice [12]	2023	Planning, nuScenes val	L2 at 3 s = 0.94 m; collision = 0.25%	UniAD: L2 = 1.03 m	Open-loop
VAD-Base [11]	2023	Planning, nuScenes val	L2 at 3 s = 0.60 m; collision = 0.20%	UniAD: L2 = 1.03 m	Open-loop
TransFuser [77]	2023	E2E driving, CARLA Town05	Route compl. = 76.0%; IS = 0.51	CILRS: RC = 45.3%; LAV: RC = 68.5%	Closed-loop
DiffusionDrive [13]	2025	Planning, NAVSIM val	PDMS = 88.1; FPS = 45	SparseDrive: PDMS = 85.0; Transfuser: 75.5	Non-reactive sim
MotionLM [58]	2023	Multi-agent forecasting, Waymo OMD test	minADE = 1.22 m; minFDE = 2.14 m	1st place Waymo interactive challenge	Offline
Social LSTM [78]	2016	Pedestrian forecasting, ETH+UCY	ADE = 0.73 m; FDE = 1.48 m	Linear: ADE = 1.33 m; S-F: ADE = 1.12 m	Offline

TABLE XI. OPERATIONAL RUNTIME-CLASS RULES. THE CLASSES ARE EVIDENCE CATEGORIES, NOT UNIVERSAL LATENCY BANDS.

Runtime class	Required evidence	Illustrative method role	Permitted inference
Hard real-time	Requirement-derived deadline plus worst-case execution and scheduling evidence	Safety monitor or low-level controller	Suitability only for the analyzed item and platform
Soft real-time	Documented update requirement plus explicit tolerance or mitigation of bounded overruns	Perception, prediction, or planning component	Performance-relevant placement under the reported conditions
Cognitive	Episodic, advisory, or asynchronous output consumed through a faster checked layer	LLM/VLM reasoner or agentic memory [9], [47], [51]	Supervisory use without direct actuation authority
Not demonstrated	Insufficient timing, integration, or overrun evidence for another class	Most benchmark-only learned methods	No deployment-class inference from mean FPS or architecture alone

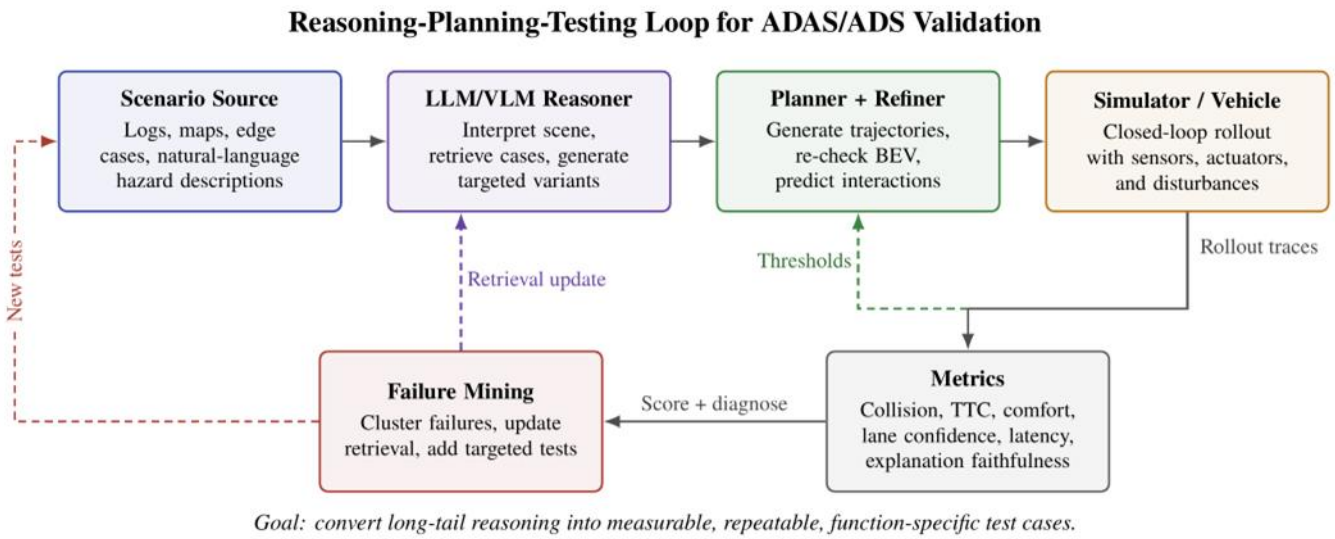


Fig. 6. Reasoning-planning-testing loop for ADAS/ADS validation. LLM/VLM modules generate or interpret scenarios, planners produce candidate trajectories, simulators evaluate closed-loop behavior, and failures are fed back into retrieval, refinement, and test-case generation.

IX. Open Challenges

A. Verifiable Knowledge-Grounded Reasoning

LLM/VLM systems can explain scenes, retrieve memories, and reason about rare events, but their outputs carry no guarantee of spatial grounding, temporal consistency, or dynamic feasibility. A driving system cannot treat a natural-language rationale as evidence of safety. The challenge is designing interfaces that convert semantic reasoning into machine-checkable constraints: admissible lanes, yield relations, target gaps, speed bounds, and uncertainty estimates.

B. Faithful Explanations and Planner-State Alignment

Many language-based driving systems generate explanations, but an explanation is useful only if it reflects the actual planner state. A model may correctly say "the vehicle slowed for a pedestrian" even if the underlying controller slowed because of a different object or a conservative fallback. Future evaluation must distinguish plausible explanations from faithful explanations tied to perception, prediction, and control traces.

C. Closed-Loop Evaluation Under Long-Tail Conditions

Open-loop displacement error misses how the system changes the future through its own actions. Closed-loop simulation is more informative, but it is expensive, sensitive to simulator fidelity, and difficult to standardize across methods. Long-tail testing is hardest of all because rare events must be both realistic and sufficiently diverse.

D. Real-Time Deployment of Foundation-Model Components

Foundation models improve semantic reasoning but conflict with onboard latency, memory, power, and qualification constraints. The unresolved question is not whether LLM/VLM models can reason about driving scenes; it is what runtime placement their reported evidence can support. The field needs clearer rules for when a cognitive model may supervise, when it may condition a planner, and when it must be excluded from direct actuation.

E. Safety Assurance Under Distribution Shift and Adversarial Inputs

Autonomous-driving systems must handle distribution shift, rare road layouts, sensor degradation, and adversarial perturbations. LLM/VLM modules add another source of risk because they may hallucinate, retrieve stale examples, or over-prioritize linguistic plausibility over physical evidence. Robustness must therefore be evaluated across perception, reasoning, planning, and control rather than as a separate perception-only problem.

X. Future Research Directions

A. Runtime-Assured LLM/VLM Driving Architectures

Runtime-assured architectures would convert LLM/VLM outputs into constrained planning objectives rather than direct commands: a reasoning layer would propose intent, a refinement layer would generate and verify candidate trajectories, and a control layer would enforce safety envelopes. This structure would make knowledge-grounded driving compatible with functional-safety principles.

B. Reasoning-Aware Closed-Loop Benchmarks

Future benchmarks should evaluate reasoning quality, explanation faithfulness, recovery after planner error, and response to counterfactual scene changes, alongside driving score and collision rate. For example, a benchmark should test whether a model changes its plan when a retrieved memory becomes irrelevant, when a pedestrian intention changes, or when a construction-zone lane marking contradicts the map.

C. ADAS-Grounded Test-Case Generation

LLM-based scenario generation should be connected more directly to ADAS/ADS testing. Instead of generating generic "dangerous scenes," future systems should generate parameterized test cases tied to specific functions: cut-in and

stop-and-go scenarios for ACC, occluded pedestrian emergence for AEB, and lane ambiguity or construction-zone cases for LKA. Each generated case should include expected hazards, measurable pass/fail criteria, and coverage labels.

D. Hardware-Aware Foundation Models for Driving

Future driving models should be designed with latency and deployment constraints from the beginning. Promising strategies include smaller specialized models, quantization, pruning, early exits, vectorized scene representations, truncated diffusion, dual-path architectures, and predictable scheduling in vehicle operating systems [63]. Model quality should be reported jointly with latency, memory footprint, power estimate, and target hardware class.

E. Unified Reasoning-Planning-Safety Interfaces

The field needs standardized interfaces between semantic reasoning, trajectory planning, and runtime safety monitoring. Such an interface should specify what a reasoning model may output, what evidence supports that output, how uncertainty is represented, how the planner converts intent into motion, and how monitors reject unsafe actions. This would make it easier to compare modular, end-to-end, and foundation-model-based driving systems under a common safety framework. Table XII summarizes the open challenges of Section IX and the corresponding research directions.

TABLE XII. OPEN CHALLENGES AND FUTURE RESEARCH DIRECTIONS.

Open challenge	Why it matters	Future direction
LLM/VLM hallucination	Unsafe semantic reasoning can mislead planners.	Runtime-constrained reasoning with machine-checkable outputs.
Explanation faithfulness	Plausible explanations may not reflect actual planner state.	Tie explanations to logged perception, prediction, and control traces.
Open-loop / closed-loop gap	Logged trajectory error does not show recovery behavior.	Closed-loop benchmarks with reactive agents and counterfactual scene changes.
Real-time deployment	Cognitive models are too slow for direct actuation.	Slow-fast architectures, early exits, compression, and hardware-aware model design.
Long-tail scenario coverage	Rare events dominate safety risk.	LLM-guided scenario generation with coverage metrics and ADAS-specific pass/fail criteria.
Hardware certification	Efficient models still need predictable execution.	Report latency, memory, energy, target hardware, and safety-layer placement with every method.

XI. REVIEW LIMITATIONS

This review is structured and evidence-traceable but is not a statistical meta-analysis. The search is limited to English-language, full-text records available through the listed sources; terminology changes rapidly, especially for VLA systems, so relevant work may have escaped the queries. Search-engine ranking and arXiv versioning also limit exact retrospective reproducibility. The original search did not preserve per-record database provenance or the identities of all duplicate and early-stage excluded records. The supplied screening_log.csv therefore documents the 107 retained records and the 12 records removed during re-audit, but it cannot reconstruct the other 301 record-level decisions; a future update must rerun and export the searches to close this reproducibility gap. The evidence matrix records what each source reports, but missing latency, power, or closed-loop results cannot be interpreted as evidence of poor performance. Conversely, a reported mean latency or simulator score does not establish worst-case timing, vehicle-level safety, or deployment readiness. The taxonomy is an author-proposed analytical device, and formal inter-rater agreement has not yet been established. Finally, heterogeneous datasets, splits, simulator configurations, hardware, precision, and reporting conventions prevent pooled effect estimates or a defensible global method ranking.

XII. CONCLUSION

The review questions yield six evidence-calibrated answers. For RQ1, LLMs, VLMs, and VLA models are used chiefly for semantic scene interpretation, instruction following, memory, explanation, tool use, and high-level planning; direct control is reported less often and is supported by weaker deployment evidence. For RQ2, semantic intent reaches motion through command conditioning, candidate generation and scoring, learned or optimization-based refinement, constraint checking, and, in stronger architectures, a monitored planner or fallback controller. For RQ3, automotive-platform latency, memory, power, precision, and worst-case timing are sparsely reported, particularly for foundation-model components.

For RQ4, open-loop logged-data evaluation remains dominant and cannot establish interactive safety; reactive closed-loop simulation, hardware-in-the-loop testing, and physical deployment provide progressively stronger but still scope-limited evidence. For RQ5, most foundation-model methods belong in the not-demonstrated class when the classification is based strictly on reported timing and integration evidence; runtime placement should not be inferred from model size or mean FPS. For RQ6, the relevant assurance mechanisms include constraint checkers, monitors, uncertainty gates, fallback controllers, safety shields, and minimum-risk behavior, but their integration with foundation-model reasoners remains inconsistent and under-evaluated.

The main architectural lesson is that layering matters more than scale: the reviewed evidence supports using cognitive models for semantic interpretation and high-level intent, with faster checked planning and safety-control mechanisms retaining authority over motion. This is a

proposed design principle, not evidence that one universal three-layer architecture is already validated or certifiable. ADAS functions such as ACC, AEB, and LKA make the division of responsibility concrete. Progress toward deployable knowledge-grounded driving therefore depends on explicit interfaces, requirement-derived timing, closed-loop testing, and runtime assurance, not merely on larger models.

Author Contributions

Conceptualization, G.P. and A.K.; methodology, G.P. and A.K.; investigation, G.P. and A.K.; data curation, G.P. and A.K.; writing—original draft preparation, G.P. and A.K.; writing—review and editing, G.P., A.K., G.G. and A.C.; visualization, G.P. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

No new experimental vehicle data were created in this study. Six supplementary data files accompany the review: (1) included_studies.csv, the bibliographic identity, publication status, foundation flag, and primary pillar for the 107 retained records; (2) search_log.csv, the query strings, dates, filters, and retrieval counts for the nine searches; (3) screening_log.csv, the available record-level decisions for the retained corpus and citation re-audit; (4) evidence_matrix.csv, 18 extracted attributes per retained record, including runtime class, hardware evidence, and closed-loop evaluation status; (5) quantitative_extraction.csv, benchmark and hardware values with source locations; and (6) quality_assessment.csv, item-level quality scores and notes for the 68 retained empirical primary-method studies. These files are provided as supplementary material with the manuscript.

Acknowledgments

The authors thank the research community whose open-access publications made this survey possible.

Conflicts of Interest

Author Gaurav, Gopi are employed in the ADAS domain. This work was completed independently, without support, resources, or input from their current employer. The views, interpretations, and conclusions expressed in this manuscript are solely those of the authors and do not reflect the views or positions of any employer.

Abbreviations

The following abbreviations are used in this manuscript:

Abbreviation	Definition
AD	Autonomous Driving
ADAS	Advanced Driver Assistance Systems
ADS	Automated Driving System
ACC	Adaptive Cruise Control
AEB	Automatic Emergency Braking
LKA	Lane Keeping Assistance
LLM	Large Language Model
VLM	Vision-Language Model
VLA	Vision-Language-Action
BEV	Bird's-Eye View
MPC	Model Predictive Control
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
CNN	Convolutional Neural Network
SAE	Society of Automotive Engineers
ISO	International Organization for Standardization
SOTIF	Safety Of The Intended Functionality
ASIL	Automotive Safety Integrity Level
TTC	Time-To-Collision
FPS	Frames Per Second
NDS	nuScenes Detection Score
PDMS	Planning-Directed Multi-Score
RAG	Retrieval-Augmented Generation
VQA	Visual Question Answering

Abbreviation	Definition
PID	Proportional-Integral-Derivative
mAP	mean Average Precision
mIoU	mean Intersection over Union

REFERENCES

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020, doi: 10.1109/ACCESS.2020.2983149.

[2] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 10164–10183, 2024, doi: 10.1109/TPAMI.2024.3435937.

[3] Y. Hu *et al.*, "Planning-oriented autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 17853–17862. doi: 10.1109/CVPR52729.2023.01712.

[4] Y. Li, K. Katsumata, E. Javanmardi, and M. Tsukada, "Large language models for human-like autonomous driving: A survey," *arXiv preprint arXiv:2407.19280*, 2024, doi: 10.48550/arXiv.2407.19280.

[5] 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, doi: 10.48550/arXiv.2311.01043.

[6] X. Zhou *et al.*, "Vision language models in autonomous driving: A survey and outlook," *IEEE Trans. Intell. Veh.*, in press, doi: 10.1109/TIV.2024.3402136.

[7] S. Jiang *et al.*, "A survey on vision-language-action models for autonomous driving," *arXiv preprint arXiv:2506.24044*, 2025, doi: 10.48550/arXiv.2506.24044.

[8] L. Wen *et al.*, "DiLu: A knowledge-driven approach to autonomous driving with large language models," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2024.

[9] J. Mao, J. Ye, Y. Qian, M. Pavone, and Y. Wang, "A language agent for autonomous driving," in *Proc. Conf. Lang. Model. (COLM)*, 2024.

[10] J. Yuan *et al.*, "RAG-Driver: Generalisable driving explanations with retrieval-augmented in-context learning of large vision-language model," *arXiv preprint arXiv:2402.10828*, 2024, doi: 10.48550/arXiv.2402.10828.

[11] B. Jiang *et al.*, "VAD: Vectorized scene representation for efficient autonomous driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2023, pp. 8340–8350. doi: 10.1109/ICCV51070.2023.00765.

[12] X. Jia, P. Wu, L. Chen, J. Liu, H. Li, and Y. Qiao, "Think twice before driving: Towards scalable decoders for end-to-end autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 21983–21994. doi: 10.1109/CVPR52729.2023.02106.

[13] B. Liao, S. Chen, Z. Jiang, H. Zhou, H. Li, and Q. Zhang, "DiffusionDrive: Truncated diffusion model for end-to-end autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2025.

[14] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2016.

[15] H. Hu, L. Huang, N. Guan, and C. J. Xue, "DeeAD: Dynamic early exit of vision-language action for efficient autonomous driving," *arXiv preprint arXiv:2511.20720*, 2025, doi: 10.48550/arXiv.2511.20720.

[16] SAE International, "SAE J3016:2021 Taxonomy and Definitions for Terms Related to Driving Automation Systems," SAE International, Warrendale, PA, USA, 2021.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[18] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Annual Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 5998–6008.

[19] OpenAI, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023, doi: 10.48550/arXiv.2303.08774.

[20] H. Touvron *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023, doi: 10.48550/arXiv.2302.13971.

[21] S. Yao *et al.*, "ReAct: Synergizing reasoning and acting in language models," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2023.

[22] T. Schick *et al.*, "Toolformer: Language models can teach themselves to use tools," in *Proc. Annual Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2023.

[23] L. Wang *et al.*, "A survey on large language model based autonomous agents," *Front. Comput. Sci.*, vol. 18, no. 6, p. 186345, 2024, doi: 10.1007/s11704-024-40231-1.

[24] X. Zou *et al.*, "Generalized decoding for pixel, image, and language," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023, pp. 15116–15127. doi: 10.1109/CVPR52729.2023.01451.

[25] Z. Li *et al.*, "BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *Proc. Eur. Conf. Comput. Vis. (ECCV), LNCS vol. 13669*, Springer, 2022, pp. 1–18. doi: 10.1007/978-3-031-20077-9_1.

[26] Z. Liu *et al.*, "BEVFusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2023, pp. 2774–2781. doi: 10.1109/ICRA48891.2023.10160968.

[27] D. Feng *et al.*, "Deep multi-modal object detection and semantic segmentation for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1341–1360, 2021, doi: 10.1109/TITS.2019.2962527.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[29] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV), LNCS vol. 9905*, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2.

[30] W. Lv *et al.*, "DETRs beat YOLOs on real-time object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024, pp. 16965–16974. doi: 10.1109/CVPR52733.2024.01605.

[31] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial CNN for traffic scene understanding," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 7276–7283.

[32] T. Zheng *et al.*, "RESA: Recurrent feature-shift aggregator for lane detection," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 3547–3554.

[33] Z. Wang, W. Ren, and Q. Qiu, "LaneNet: Real-time lane detection networks for autonomous driving," *Neural Netw.*, vol. 116, pp. 222–232, 2019, doi: 10.1016/j.neunet.2019.04.003.

[34] L. Liu, X. Chen, S. Zhu, and P. Tan, "CondLaneNet: A top-to-down lane detection framework based on conditional convolution," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2021, pp. 3773–3782. doi: 10.1109/ICCV48922.2021.00375.

[35] L. Tabelini, R. Berriel, T. M. Paixão, C. Badue, A. F. De Souza, and T. Oliveira-Santos, "Keep your eyes on the lane: Real-time attention-guided lane detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 294–302. doi: 10.1109/CVPR46437.2021.00036.

[36] T. Zheng *et al.*, "CLRNet: Cross layer refinement network for lane detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2022, pp. 898–907. doi: 10.1109/CVPR52688.2022.00097.

[37] L. Xiao, X. Li, S. Yang, and W. Yang, "ADNet: Lane shape prediction via anchor decomposition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2023, pp. 6404–6413.

[38] Z. Rahman and B. T. Morris, "LVLane: Deep learning for lane detection and classification in challenging conditions," *arXiv preprint arXiv:2307.06853*, 2023, doi: 10.48550/arXiv.2307.06853.

[39] H. Caesar *et al.*, "NuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*

(CVPR), 2020, pp. 11621–11631. doi: 10.1109/CVPR42600.2020.01164.

[40] P. Sun *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 2446–2454. doi: 10.1109/CVPR42600.2020.00252.

[41] A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "Are they going to cross? A benchmark dataset and baseline for pedestrian crosswalk behavior," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, 2017, pp. 206–213. doi: 10.1109/ICCVW.2017.33.

[42] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. Robot Learn. (CoRL)*, 2017.

[43] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, and B. Zhou, "MetaDrive: Composing diverse driving scenarios for generalizable reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3461–3475, 2023, doi: 10.1109/TPAMI.2022.3190471.

[44] C. Sima *et al.*, "DriveLM: Driving with graph visual question answering," in *Proc. Eur. Conf. Comput. Vis. (ECCV), LNCS vol. 15110*, Springer, 2024. doi: 10.1007/978-3-031-72943-0_15.

[45] S. Wang, X. Jia, L. Chen, C. Zhang, H. Li, and J. Yan, "OmniDrive: A holistic LLM-agent framework for autonomous driving with 3D perception, reasoning and planning," *arXiv preprint arXiv:2405.01533*, 2024, doi: 10.48550/arXiv.2405.01533.

[46] J. Mao, Y. Qian, J. Ye, H. Zhao, and Y. Wang, "GPT-Driver: Learning to drive with GPT," *arXiv preprint arXiv:2310.01415*, 2023, doi: 10.48550/arXiv.2310.01415.

[47] H. Sha *et al.*, "LanguageMPC: Large language models as decision makers for autonomous driving," *arXiv preprint arXiv:2310.03026*, 2023, doi: 10.48550/arXiv.2310.03026.

[48] H. Shao, Y. Hu, L. Wang, S. L. Waslander, Y. Liu, and H. Li, "LMDrive: Closed-loop end-to-end driving with large language models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024, pp. 15120–15130. doi: 10.1109/CVPR52733.2024.01432.

[49] Z. Xu *et al.*, "DriveGPT4: Interpretable end-to-end autonomous driving via large language model," *IEEE Robot. Autom. Lett.*, vol. 9, no. 10, pp. 8186–8193, 2024, doi: 10.1109/LRA.2024.3440097.

[50] Q. Wu, X. Zhang, P. Zhao, and L. Sun, "Large language model guided deep reinforcement learning for autonomous driving," *arXiv preprint arXiv:2412.18511*, 2024, doi: 10.48550/arXiv.2412.18511.

[51] X. Tian *et al.*, "DriveVLM: The convergence of autonomous driving and large vision-language models," in *Proc. Conf. Robot Learn. (CoRL)*, 2024.

[52] J. Zhang, C. Xu, and B. Li, "ChatScene: Knowledge-enabled safety-critical scenario generation for autonomous vehicles," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024, pp. 15459–15469. doi: 10.1109/CVPR52733.2024.01465.

[53] M. Bojarski *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016, doi: 10.48550/arXiv.1604.07316.

[54] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 4834–4841. doi: 10.1109/ICRA.2018.8460487.

[55] B. R. Kiran *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926, 2022, doi: 10.1109/TITS.2021.3054625.

[56] Z. Zhu and H. Zhao, "A survey of deep RL and IL for autonomous driving policy learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14043–14065, 2022, doi: 10.1109/TITS.2021.3134702.

[57] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, F. Arango, P. Revenga, and L. M. Bergasa, "Enhancing autonomous driving in urban scenarios: A hybrid approach with reinforcement learning and classical control," *Sensors*, vol. 25, no. 1, p. 117, 2024, doi: 10.3390/s25010117.

[58] A. Seff *et al.*, "MotionLM: Multi-agent motion forecasting as language modeling," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2023, pp. 8579–8590. doi: 10.1109/ICCV51070.2023.00787.

[59] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, no. 5, pp. 4282–4286, 1995, doi: 10.1103/PhysRevE.51.4282.

[60] X. Gu, C. Li, L. Gao, and X. Niu, "A review of pedestrian trajectory prediction methods based on deep learning technology," *Sensors*, vol. 25, no. 23, p. 7360, 2025, doi: 10.3390/s25237360.

[61] K. Saleh, "Pedestrian trajectory prediction for real-time autonomous systems via context-augmented transformer networks," *Sensors*, vol. 22, no. 19, p. 7495, 2022, doi: 10.3390/s22197495.

[62] Euro NCAP, "AEB Pedestrian Test Protocol Version 4.0," European New Car Assessment Programme, Brussels, Belgium, 2022.

[63] Y. Wang, Y. He, B. Tian, L. Xian, and W. Shi, "DAVOS: An autonomous vehicle operating system in the vehicle computing era," *arXiv preprint arXiv:2601.05072*, 2026, doi: 10.48550/arXiv.2601.05072.

[64] Z. Qiao, H. Li, Z. Cao, and H. X. Liu, "LightEMMA: Lightweight end-to-end multimodal model for autonomous driving," *arXiv preprint arXiv:2505.00284*, 2025, doi: 10.48550/arXiv.2505.00284.

[65] J. E. Naranjo, C. Gonzalez, R. Garcia, and T. de Pedro, "ACC+Stop&Go maneuvers with throttle and brake fuzzy control," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 2, pp. 213–225, 2006, doi: 10.1109/TITS.2006.874707.

[66] J. Ploeg, B. T. M. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer, "Design and experimental evaluation of cooperative adaptive cruise control," in *Proc. 14th IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, 2011, pp. 260–265. doi: 10.1109/ITSC.2011.6082981.

[67] R. Garg, A. Choudhary, and N. K. Verma, "Lightweight and faster YOLO for real-time and embedded systems edge deployment of autonomous vehicle pedestrian detection," in *Proc. 2022 IEEE Int. Conf. Image Process. (ICIP)*, 2022, pp. 3716–3720. doi: 10.1109/ICIP46576.2022.9897768.

[68] H. Rezaee, K. Zhang, T. Parisini, and M. M. Polycarpou, "Cooperative adaptive cruise control in the presence of communication and radar stochastic data loss," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 6, pp. 4964–4976, 2024, doi: 10.1109/TITS.2023.3335310.

[69] Y. Wang, Q. Zhang, and R. Yang, "Safety-first reinforcement learning for adaptive cruise control," *Sensors*, vol. 24, no. 8, p. 2657, 2024, doi: 10.3390/s24082657.

[70] H. Caesar *et al.*, "nuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles," *arXiv preprint arXiv:2106.11810*, 2021, doi: 10.48550/arXiv.2106.11810.

[71] D. Dauner, M. Hallgarten, A. Geiger, and K. Chitta, "NAVSIM: Data-driven non-reactive autonomous vehicle simulation and benchmarking," in *Proc. Annual Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2024.

[72] X. Jia, P. Wu, L. Chen, J. Yan, H. Li, and Y. Qiao, "Bench2Drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving," *arXiv preprint arXiv:2406.03877*, 2024, doi: 10.48550/arXiv.2406.03877.

[73] International Organization for Standardization, "ISO 26262:2018 Road Vehicles—Functional Safety," ISO, Geneva, Switzerland, 2018.

[74] International Organization for Standardization, "ISO 21448:2022 Road Vehicles—Safety of the Intended Functionality," ISO, Geneva, Switzerland, 2022.

[75] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2014.

[76] Y. Cao *et al.*, "Adversarial sensor attack on LiDAR-based perception in autonomous driving," in *Proc. 2019 ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2019, pp. 2267–2281. doi: 10.1145/3319535.3339815.

[77] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "TransFuser: Imitation with transformer-based sensor fusion for autonomous driving," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 12878–12895, 2023, doi: 10.1109/TPAMI.2022.3200245.

[78] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 961–971. doi: 10.1109/CVPR.2016.110.