WHAT TO TEST NEXT: INTERPRETABLE COVERAGE GAP DISCOVERY IN DRIVING VLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Driving vision-language models (VLMs) must accurately understand scenes across diverse conditions defined by Operational Design Domains (ODDs), yet verification remains sparse: many slices are missing, making empirical failure rates unreliable. We propose SLICESCORER, a deterministic scoring rule for missing-slice recommendation that combines (i) an exposure-based coverage prior to prioritize rare, under-tested regions, and (ii) a neighbor-failure prior that propagates risk from similar tested conditions. SLICESCORER is deliberately simple - interpretable, auditable, and conservative - properties essential for safety-critical validation. For stress testing beyond the declared ODD, we embed SLICESCORER within SLICENAV, an LLM-orchestrated verification pipeline where the model interprets developer queries to select relevant operators (triage, scoring, acquisition, evaluation) and vocabulary extensions, composing verification workflows while keeping all scoring deterministic and auditable. Experiments on three driving VLMs (WiseAD, DriveMM, Cosmos-Reason2-2B) show that SLICENAV balances failure-risk capture and recommendation diversity competitively with prior slice-discovery methods, with the coverage–risk tradeoff improving most steeply at larger testing budgets. Ablations confirm both scoring components contribute, and qualitative analysis demonstrates end-to-end workflows from developer query to targeted evaluation.

1 INTRODUCTION

Driving vision-language models (VLMs) must work reliably across diverse conditions - foggy nights, construction zones, dense traffic, unusual road layouts, *etc.* - yet testing every scenario is infeasible. Standard benchmarks cover common cases well, but rare conditions where failures matter most remain undertested. When validation budgets are limited, developers need principled guidance: *what should we test next?* This question is not addressed by existing methods (Chung et al., 2018; Sagadeeva & Boehm, 2021).

Performance gap-discovery methods like SliceFinder (Chung et al., 2018) and SliceLine (Sagadeeva & Boehm, 2021) identify failure patterns in *already-evaluated* data - useful for diagnosis, but silent on where to look next. Operational Design Domains (ODD) taxonomies and coverage metrics (ISO 34503:2023; International Organization for Standardization, 2022; de Gelder et al., 2024) describe *where* a system should operate, but do not rank which untested condition to prioritize. Further, simply building an large language model (LLM) based agent would not provide a deterministic, auditable rule for selecting the next scenario to evaluate - precisely what safety-critical verification requires (see Figure 1). Given these important caveats, we pose an important problem in this paper:

> *Given a validation set with partial coverage, how do we prioritize **missing** scenarios to test next - both within and beyond the declared operating domain - in a way that is traceable and auditable?*

To this end, we introduce SLICENAV, an agentic verification framework that recommends which untested driving scenarios to evaluate and orchestrates operator calls based on user query. The core insight is simple: rank missing scenarios by combining **rarity** - how undertested a condition is - with **neighbor-failure similarity** - how close it is to conditions where the model already fails. This scoring is implemented in our proposed SLICESCORER. An LLM then uses this scenario scoring

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

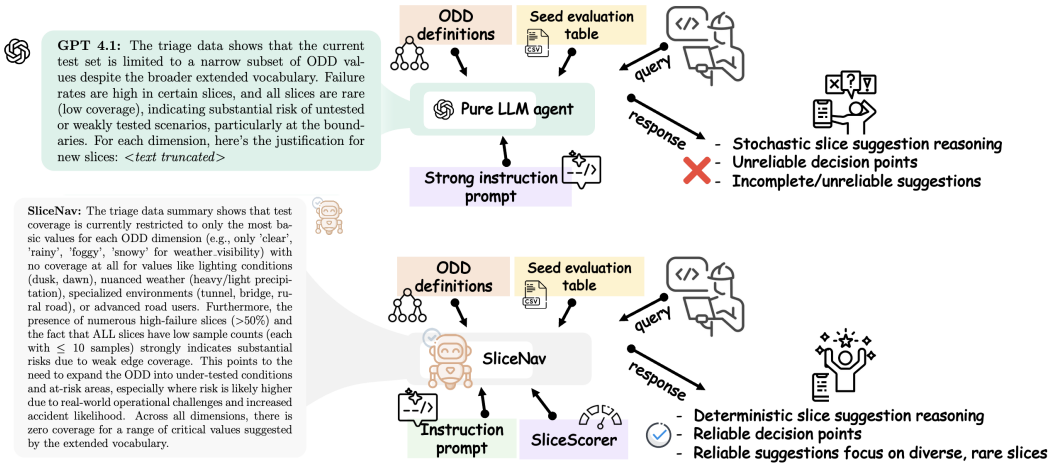


Figure 1: **From stochastic prompting to structured slice reasoning.** *Top:* A pure LLM agent, even when provided with ODD definitions, seed evaluation tables, and strong instruction prompts, produces stochastic slice suggestions with unreliable decision boundaries and incomplete coverage. *Bottom:* SLICENAV augments the LLM with structured instructions and a dedicated SLICESCORER, enabling deterministic slice suggestion reasoning, reliable decision points, and focused discovery of diverse, rare, and high-risk slices. This structured integration leads to more systematic, interpretable, and dependable ODD coverage expansion.

mechanism for both inside-ODD and outside-ODD related developer queries to build an overall fully deterministic and auditable framework.

We evaluate our framework on three driving VLMs: WiseAD (Zhang et al., 2024), DriveMM (Huang et al., 2024), and Cosmos-Reason2-2B (Cosmos) (NVIDIA Corporation, 2026), and measure two complementary objectives: *risk capture* (do top-ranked scenarios reveal actual failures?) and *diversity* (do recommendations span the condition space rather than clustering in one region?). Experiments on three driving VLMs show that SLICESCORER achieves the best joint diversity–risk score in majority settings, sweeping all three VLMs at $K{=}5000$. Outside the declared ODD, SLICESCORER surfaces failures earliest in most settings and leads the joint score at $K{=}500$ across all VLMs. Qualitative runs further demonstrate end-to-end workflows - from developer query through ODD extension, data acquisition, and VLM evaluation - with all scoring remaining deterministic. We summarize our contributions as follows:

1. We formulate **missing-scenario recommendation** for driving VLM verification: given partial test coverage, rank untested conditions by predicted failure risk.
2. We propose SLICESCORER, a deterministic scoring rule combining rarity and neighbor-failure priors - interpretable, auditable, and simple by design.
3. We present SLICENAV, which embeds this scorer in an LLM-orchestrated workflow for flexible, query-driven verification while preserving deterministic ranking.

2 RELATED WORKS

Failure subpopulations discovery methods. Slice discovery aims to diagnose model failures by identifying human-interpretable subpopulations ("slices") where performance deviates significantly from the global average, typically expressed as conjunctions of attribute predicates. Early work such as SliceFinder (Chung et al., 2018) introduced heuristic lattice exploration and statistical controls to surface problematic tabular subgroups, while SliceLine (Sagadeeva & Boehm, 2021) improved scalability through aggressive pruning and a linear-algebraic formulation for efficient top-K conjunctive slice enumeration. Subsequent systems extend this paradigm toward large-scale and interactive settings: AutoSlicer (Liu et al., 2022) emphasizes distributed computation and hypothesis testing for production-scale auditing, and Divisi (Sivaraman et al., 2025) adopts sampling and

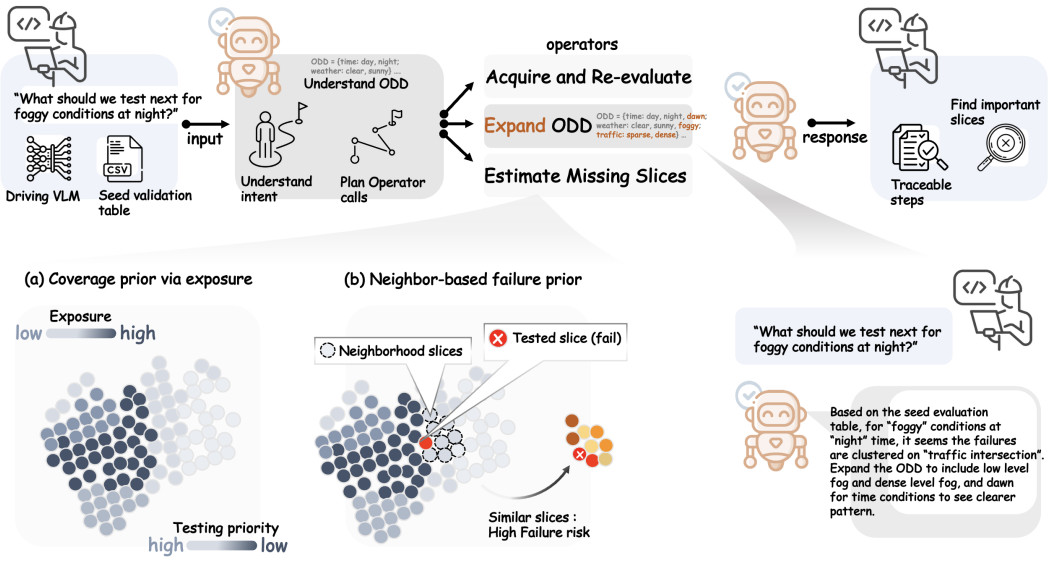


Figure 2: **Framework overview.** The agent takes a natural-language verification goal plus evaluation context and ODD specification, then orchestrates failure triage, in-ODD gap discovery, and controlled out-of-ODD stress proposal via a query-conditioned ODD expansion. It can also close the loop by acquiring data, and re-evaluating the VLM.

constrained (e.g., beam-style) search for fast approximate discovery. Closely related are pattern-mining approaches such as DivExplorer (Pastor et al., 2021), which frame slice discovery as mining frequent itemsets with high metric divergence, integrating support thresholds and redundancy control. In contrast, representation-driven methods for unstructured domains (Eyuboglu et al., 2022a) discover error-prone regions in embedding space, often sacrificing the direct interpretability of explicit predicate-defined slices. Our work follows the SliceLine-style paradigm (Sagadeeva & Boehm, 2021), focusing on scalable, interpretable rule-based slice discovery over large candidate spaces, but addresses the complementary problem of ranking *untested* slices for future evaluation rather than diagnosing failures in already-observed data. Separately, pool-based active learning (Settles, 2009) and Bayesian GP surrogates (Riis et al., 2022; Shahriari et al., 2016) assume a closed loop in which an oracle is queried after each selection and the model is re-fitted. Our task is single-pass: one ranking is produced from a fixed seed table with no iterative feedback, so acquisition-function methods are not directly comparable. We nonetheless evaluate a GP posterior mean as a reference baseline (Section 4); its stochasticity and lack of interpretable slice descriptors confirm the need for a lightweight deterministic scorer.

Importance of evaluation reliability. Recent multimodal evaluation pipelines increasingly use LLMs for scalable scoring of open-ended outputs and structured reporting, but such evaluators must be audited for bias and reliability (Zheng et al., 2023; Dubois et al., 2024; Yu et al., 2023). Separately, interpretability research emphasizes that explanations should be evaluated for faithfulness and practical utility rather than plausibility alone (Doshi-Velez & Kim, 2017; Jacovi & Goldberg, 2020). Motivated by these concerns, we implement our method as an agentic controller that orchestrates structured verification operators.

3 Proposed Method

3.1 Terminology

We establish key terms used throughout. An **Operational Design Domain (ODD)** specifies conditions under which an autonomous system operates, instantiated as condition dimensions (weather, time, road type, etc.) with discrete value sets. These specific dimensions, value sets, and granularity are defined by the developer to match their deployment requirements; SliceScorer is agnostic to

the particular taxonomy. A developer may define any ODD taxonomy suited to their deployment, provided it yields a finite, navigable space that can be systematically audited; we provide one concrete instantiation grounded in automotive safety standards in Section 4. A **slice** is a complete value assignment across all ODD dimensions, for example, *(foggy, night, highway, wet, dense traffic, intersection, hazard assessment)*. The **seed validation table** records human-verified VLM performance per slice, requiring a minimum sample threshold. An **observed slice** has at least one sample in the seed table; a **missing slice** (equivalently, a **coverage gap**) has none, leaving VLM behavior unknown. **Exposure** estimates how likely a slice is to appear based on marginal frequencies; low-exposure slices are rare and under-tested. **Next-test recommendation** selects which missing slices to evaluate under a limited budget; SLICENAV ranks candidates so higher-priority slices are more likely to reveal failures or close critical gaps.

3.2 PROBLEM DEFINITION

ODD taxonomy and slice space. Let K denote the number of ODD condition dimensions (*e.g.*, weather visibility, sky conditions, time of day). Each dimension $k \in \{1, \ldots, K\}$ has a finite in-ODD value set V_k. A slice is formally a complete assignment:

$$s = (s_1, \ldots, s_K), \quad s_k \in V_k, \tag{1}$$

and the in-ODD slice universe is the Cartesian product $\mathcal{S}_{\text{ODD}} = \prod_{k=1}^{K} V_k$. For each slice s, the seed table records the sample count $n(s)$ and aggregated performance metrics. We use LLM-judge scores uniformly across question types (free-form QA, multiple-choice, captioning), averaged across all samples for slice s. The set of observed slices (those with evaluation support) is:

$$\mathcal{S}_{\text{obs}} = \{s \in \mathcal{S}_{\text{ODD}} : n(s) > 0\}. \tag{2}$$

The missing slices are $\mathcal{S}_{\text{miss}} = \mathcal{S}_{\text{ODD}} \setminus \mathcal{S}_{\text{obs}}$. Our primary goal is to rank $s \in \mathcal{S}_{\text{miss}}$ so that high-risk coverage gaps are prioritized for testing. A developer may also provide a natural-language query q describing a focus (*e.g.*, "foggy night hazards" or "construction zone failures").

3.3 OVERVIEW OF THE WORKFLOW

SLICENAV is an agentic VLM ODD verification framework that reframes slice analysis as a *test selection strategy*: given a seed validation table, it recommends which untested condition slices should be evaluated next and keeps each recommendation traceable to observed evidence. It combines *(i)* a deterministic slice-prioritization model (SLICESCORER), which scores untested slices using exposure rarity and observed failure evidence, with *(ii)* an agentic orchestration layer, where an LLM interprets developer queries, selects verification operators, and composes multi-step workflows. This treats ODD verification as a sequential decision problem over a combinatorial condition space, addressing a gap left by existing slice analysis tools (Chung et al., 2018; Sagadeeva & Boehm, 2021; Eyuboglu et al., 2022b), which diagnose failures within already evaluated datasets but do not guide where evaluation effort should go next. Figure 2 summarizes the end-to-end workflow. Given the inputs above, SLICENAV executes (all or a subset of) the following operators depending q:

1. **Triage:** Summarize where the VLM fails among observed slices.
2. **In-ODD scoring:** Enumerate missing in-ODD slices and rank them using SLICESCORER.
3. **Out-of-ODD scoring:** Extend the ODD vocabulary and rank newly reachable missing slices.
4. **Acquisition:** Retrieve matching samples for recommended slices.
5. **Evaluation:** Run the VLM on acquired samples and update the validation table.

The LLM translates developer intent into operator sequences and produces human-readable responses. This separation ensures that critical risk estimation remains deterministic (via SLICESCORER) while enabling flexible interaction with verification workflows. A run of SLICENAV produces a natural-language report summarizing failures, coverage gaps, and recommended next tests.

3.4 DETERMINISTIC SLICE SCORING WITH SLICESCORER

We now define SLICESCORER, the function used to rank missing slices $s \in \mathcal{S}_{\text{miss}}$. The scorer is fully determined by the seed validation table, making recommendations reproducible and auditable.

Design rationale. Unlike diagnostic methods like SliceFinder (Chung et al., 2018) and SliceLine (Sagadeeva & Boehm, 2021) that search for *observed* subpopulations with high error, our goal is *next-test selection*: prioritizing *missing* slices. This requires: (i) a notion of exposure rarity to identify under-tested regions, and (ii) a way to extrapolate risk from observed evidence. We present SLICESCORER ϕ, a multiplicative combination of rarity and neighbor-failure priors:

$$\phi(s) = w_{\text{rare}}(s) \cdot w_{\text{nb}}(s), \qquad s \in \mathcal{S}_{\text{miss}}. \tag{3}$$

Sorting missing slices by $\phi(s)$ yields a prioritized next-test list that prefers (i) under-tested regions and (ii) candidates that are similar to observed high-error slices. This decomposition yields an auditable rationale for each recommendation, essential when triggering costly or safety-critical testing.

Rarity prior via exposure. Not every missing slice is equally worth testing. We estimate each slice's exposure and prioritize low-exposure gaps. Let $N = \sum_{s \in \mathcal{S}_{\text{obs}}} n(s)$ be the total evaluated support. For each dimension k and each value v that appears in the seed table, the marginal support is $n_k(v) = \sum_{s \in \mathcal{S}_{\text{obs}}: s_k = v} n(s)$. We apply Laplace smoothing to handle values with zero samples gracefully; without it, a single zero marginal would collapse the entire exposure to zero. The smoothed marginal probability (with smoothing parameter $\alpha > 0$) is:

$$P_k(v) = \frac{n_k(v) + \alpha}{N + \alpha |V_k|}, \tag{4}$$

for values $v \in V_k$. We approximate joint exposure by a product of per-dimension probabilities, assuming dimension independence for tractability since computing true joint probabilities is infeasible with sparse data:

$$E(s) = \left(\prod_{k=1}^{K} P_k(s_k) \right)^{\tau}, \tag{5}$$

where $\tau \in (0, 1]$ reduces dynamic range so exposures remain comparable as K increases. Finally, we convert exposure to a rarity-based priority by subtracting from 1 (so rare slices score high) and raising to a power γ that controls how aggressively we prioritize rare over common slices:

$$w_{\text{rare}}(s) = (1 - E(s))^{\gamma}. \tag{6}$$

Note that our goal is to provide a deterministic and auditable prioritization rule that remains usable when seed coverage is sparse. The independence approximation is therefore used as a conservative coverage prior, and not as a claim about the true data-generating distribution.

Neighbor-based failure prior. Rarity alone does not indicate *which* missing slices are risky. We propagate failure evidence from *observed* slices that are semantically close to the candidate. We use embedding similarity rather than exact dimension matches because it captures that related concepts (e.g., "heavy_fog" and "fog") should share risk signals even if textually different. Each slice embedded with a text encoder; let $\text{sim}(s, s') \in [0, 1]$ denote non-negative clipped cosine similarity between slice embeddings. The neighbor set $\mathcal{N}(s) \subseteq \mathcal{S}_{\text{obs}}$ contains the top-m observed slices by similarity, restricted to those with $\text{sim}(s, s') \geq \theta$; if no neighbor exceeds this threshold, $\mathcal{N}(s) = \emptyset$. For an observed slice $s' \in \mathcal{S}_{\text{obs}}$, let $\epsilon(s') = 1 - \bar{y}(s') \in [0, 1]$ denote the error rate, where $\bar{y}(s')$ is the mean LLM-judge score for slice s'. We define the neighbor-based error estimate:

$$\hat{\epsilon}(s) = \min\left(\max_{s' \in \mathcal{N}(s)} \text{sim}(s, s') \cdot \epsilon(s'),\ \kappa \right), \tag{7}$$

where we take the max because we want the strongest risk signal. We multiply by similarity so closer neighbors provide more reliable evidence; and we cap with $\kappa \in (0, 1]$ to prevent a single extreme neighbor from dominating. When $\mathcal{N}(s) = \emptyset$, we set $\hat{\epsilon}(s) = 0$. The neighbor weight is:

$$w_{\text{nb}}(s) = 1 + \hat{\epsilon}(s). \tag{8}$$

The additive 1 ensures slices without neighbor evidence still receive a baseline score; the multiplicative form in $\phi(s)$ then boosts slices near observed failures proportionally, bounded by $(1 + \kappa)$.

3.5 SLICENAV: AGENTIC ORCHESTRATION

While SLICESCORER provides deterministic slice ranking, SLICENAV's agentic layer enables flexible, intent-driven verification workflows. The LLM makes two key decisions: *operator selection* and *outside-ODD extension*.

Operator selection. Given a developer's natural-language query, the LLM interprets the intent and selects which operators to invoke. For example, the query "Where does the model fail on foggy nights?" triggers triage followed by in-ODD scoring with a focus on foggy-night slices. The query "Stress test beyond current ODD for construction scenarios" triggers vocabulary extension and out-of-ODD scoring. This dispatch mechanism allows developers to interact with SLICENAV conversationally while the underlying scoring remains deterministic.

Extension outside-ODD. Beyond identifying coverage gaps *within* the declared ODD, we support stress-test recommendations *outside* the base domain. We maintain an extended vocabulary of plausible out-of-ODD values (*e.g.*, *fog_dense*, *tunnel*, *glare*). Given query q, the LLM selects relevant extensions, yielding expanded value sets $V_k^+(q) \supseteq V_k$ per dimension. The expanded slice space is:

$$\mathcal{S}^+(q) = \prod_{k=1}^{K} V_k^+(q). \tag{9}$$

Missing slices in this expanded space are $\mathcal{S}^+_{\text{miss}}(q) = \mathcal{S}^+(q) \setminus \mathcal{S}_{\text{obs}}$. To score these out-of-ODD candidates, we adapt SLICESCORER to handle novel vocabulary. Since out-of-vocabulary values $v \notin V_k$ have no marginal counts, we estimate their probability by matching to the nearest observed value: $P_k(v) = P_k(v^*) \cdot \text{sim}(v, v^*) \cdot \beta$, where $v^* \in V_k$ is the nearest observed value and $\text{sim}(v, v^*)$ denotes cosine similarity between their text embeddings, and $\beta \in (0, 1]$ discounts confidence in the extrapolated estimate. The same $\phi(s)$ formula (Equation (3)–Equation (8)) then applies to rank out-of-ODD candidates.

Workflow composition. Given a query, SLICENAV chains operators into a verification workflow. SLICESCORER keeps scoring deterministic, while the LLM handles intent, sequencing, and readable output - preserving reproducibility and auditability with natural-language interaction.

4 EXPERIMENTS

Baselines and VLMs. Within-ODD, our primary baseline is SliceFinder (Chung et al., 2018); we additionally report SliceLine (Sagadeeva & Boehm, 2021) as a reference baseline, since its pruning-based enumeration was designed for diagnostic slice discovery in already-observed data rather than ranking missing slices. Both methods are adapted as described in Appendix F. For outside-ODD testing, baselines include embedding-based k-Nearest neighbor (kNN) with $k = 1$ retrieval, which retrieves slices nearest in embedding space without failure-aware scoring, and random selection. We use driving VLMs WiseAD (Zhang et al., 2024), DriveMM (Huang et al., 2024), and Cosmos-Reason2-2B (NVIDIA Corporation, 2026). We additionally report GP-Mean—the posterior mean of a Gaussian process fitted to slice-level failure estimates—as a reference baseline ($O(|\mathcal{S}_{\text{obs}}|^3)$ kernel fitting, stochastic, non-interpretable). Ground-truth data construction, including dataset aggregation, annotation, and expert verification, is detailed in Appendix J.

Evaluation metrics. We evaluate recommendations against an oracle slice table constructed from a large, independently collected dataset disjoint from the seed set; the full oracle construction procedure and LLM judge details are in Appendices K and G. We assess two complementary objectives: failure discovery and diversity. We report both metrics together because the goal is to identify high-risk missing slices while also covering diverse operating conditions. Regret@K measures how much oracle failure risk is missed relative to the best possible top-K recommendation list:

$$\text{Regret@}K = \frac{1}{K}\left(\sum_{i=1}^{K} \epsilon_i^\star - \sum_{i=1}^{K} \epsilon_i\right), \tag{10}$$

where $\epsilon_1^\star \geq \cdots \geq \epsilon_K^\star$ are the oracle failure risks of the top-K slices under the oracle ranking, and ϵ_i is the oracle failure risk of the i-th slice returned by the evaluated method. Lower Regret indicates faster failure discovery. We also report ILD@K (intra-list diversity), defined as the mean pairwise label disagreement across dimensions in the top-K recommendation list (Ziegler et al., 2005); higher ILD indicates broader coverage of the condition space. To capture both objectives in a single number, we report PO@K = ILD@K $\times$ (1 $-$ Regret@K), a joint score where higher values indicate diverse failure discovery.

Table 1: **In-ODD analysis.** R@$K\downarrow$ = missed failure risk; ILD@$K\uparrow$ = recommendation diversity (shown in grey). PO↑ = ILD×(1−Regret), a joint score capturing diverse failure discovery (higher is better; bold = best per budget among non-reference methods). SLICESCORER achieves the highest PO in 5 of 9 settings. SliceLine (below separator) is a reference baseline designed for diagnostic slice discovery in already-observed data, not missing-slice ranking; GP-Mean‡ is a reference baseline requiring $O(|\mathcal{S}_{\text{obs}}|^3)$ kernel fitting, stochastic, and non-interpretable.

		$K = 500$			$K = 1000$			$K = 5000$		
	Method	R@$K\downarrow$	ILD@$K\uparrow$	PO@$K\uparrow$	R@$K\downarrow$	ILD@$K\uparrow$	PO@$K\uparrow$	R@$K\downarrow$	ILD@$K\uparrow$	PO@$K\uparrow$
WiseAD	SLICESCORER	0.5618	0.5788	0.2536	0.3851	0.5980	0.3677	0.0258	0.6203	**0.6043**
	SliceFinder	0.5103	0.5321	**0.2606**	0.3460	0.5724	**0.3743**	0.0764	0.5724	0.5287
	GP-Mean‡	0.5160	0.5890	0.2851	0.3460	0.6050	0.3957	—	—	—
	SliceLine	0.5151	0.5270	0.2555	0.3357	0.5676	0.3771	0.0302	0.6200	0.6013
DriveMM	SLICESCORER	0.3493	0.5535	**0.3602**	0.2105	0.5741	0.4533	0.0130	0.6204	**0.6123**
	SliceFinder	0.3193	0.5220	0.3553	0.1809	0.5693	**0.4663**	0.0369	0.5693	0.5483
	GP-Mean‡	0.3010	0.5740	0.4012	0.1800	0.5940	0.4871	—	—	—
	SliceLine	0.2882	0.5208	0.3707	0.1536	0.5829	0.4934	0.0136	0.6204	0.6120
Cosmos	SLICESCORER	0.4847	0.5852	**0.3016**	0.3114	0.5885	0.4052	0.0154	0.6204	**0.6108**
	SliceFinder	0.4419	0.5279	0.2946	0.2453	0.5718	**0.4315**	0.0530	0.5718	0.5415
	GP-Mean‡	0.4340	0.5760	0.3260	0.2560	0.6000	0.4464	—	—	—
	SliceLine	0.3918	0.5341	0.3248	0.2523	0.5804	0.4340	0.0186	0.6201	0.6086

ODD taxonomy. The taxonomy used here is one concrete instantiation; discretization is a deliberate engineering choice that makes the ODD space navigable, measurable, and auditable, and continuous attributes (e.g., visibility distance) can be incorporated as ranges without changing the scoring logic. Our example taxonomy is grounded in ISO 34503, ISO 21448 (SOTIF), and SAE J3016 (see Appendix E). It uses eight structured dimensions—`weather_visibility`, `sky_conditions`, `time`, `environment`, `road_condition`, `road_type`, `traffic_type`, and `question_category`—covering environmental conditions, road context, traffic state, and QA task type. To enable stress testing beyond the declared domain, we define an extended ODD vocabulary that refines and expands each dimension with finer-grained values (e.g., decomposing weather into intensities such as `rain_heavy` and `fog_dense`, adding scene types like `tunnel` and `parking_lot`, and introducing a `road_users_present` dimension for interacting agents). The full extended vocabulary and its construction are detailed in Appendix E.1.

Experimental protocol. Exact protocols for Analysis 1–4 (inputs, outputs, baseline adaptations) are in Appendix L. SLICESCORER hyperparameters: smoothing α=1.0 (Equation (4)), dynamic range τ=0.6 (Equation (5)), rarity emphasis γ=2.0 (Equation (6)), neighbor cap κ=0.8 (Equation (7)), neighbor count m=5, similarity threshold θ=0.3, and decay β=0.8 for out-of-ODD extension. Slice tuples are embedded with `all-MiniLM-L6-v2` (Reimers & Gurevych, 2019) using cosine similarity. No dataset-specific sweeps were performed. Computationally, SLICESCORER scores each candidate in $O(m)$ time (m=5), making full ranking linear in $|\mathcal{S}_{\text{miss}}|$; GP methods require $O(|\mathcal{S}_{\text{obs}}|^3)$ kernel fitting, prohibitive in the 91M+ outside-ODD space.

4.1 ANALYSIS 1: IN-ODD MISSING-SLICE RANKING

Table 1 reports in-ODD results across three VLMs and three budgets. Among non-reference methods, SLICESCORER achieves the highest PO@K in 5 of 9 settings, including a clean sweep of all three VLMs at K=5000. SliceFinder wins PO at K=1000 across all three VLMs, but its marginal-lift model saturates at larger budgets where SLICESCORER continues to improve. From K=500 to K=5000, SLICESCORER's Regret drops by 95–97% while ILD rises monotonically—meaning recommendations become both more failure-accurate and more diverse as budget grows, the regime where real test campaigns operate. GP-Mean, included as a reference, is absent at K=5000 entirely because its cubic kernel fitting cannot scale to the full candidate set; its stochastic and non-interpretable

Table 2: **Outside-ODD analysis.** Same conventions as Table 1. †Random achieves high ILD by construction (uniform sampling) and is shown as a diversity ceiling; its PO is not bolded.

		$K = 500$			$K = 1000$			$K = 5000$		
	Method	R@$K\downarrow$	ILD@$K\uparrow$	PO@$K\uparrow$	R@$K\downarrow$	ILD@$K\uparrow$	PO@$K\uparrow$	R@$K\downarrow$	ILD@$K\uparrow$	PO@$K\uparrow$
WiseAD	SLICESCORER	0.9940	0.4453	**0.0027**	0.9930	0.4806	**0.0034**	0.3532	0.5560	0.3596
	Embedding kNN	1.0000	0.4789	0.0000	0.9980	0.5037	0.0010	0.3544	0.5776	**0.3729**
	GP-Mean‡	0.9910	0.4550	0.0041	0.9920	0.4830	0.0039	—	—	—
	Random†	1.0000	0.8003	0.0000	0.9990	0.8002	0.0008	0.3560	0.8000	0.5152
DriveMM	SLICESCORER	0.9979	0.3979	**0.0008**	0.7616	0.4421	0.1054	0.1827	0.5405	0.4418
	Embedding kNN	1.0000	0.4291	0.0000	0.7626	0.4631	**0.1099**	0.1825	0.5554	**0.4540**
	GP-Mean‡	0.9990	0.4090	0.0004	0.7660	0.4470	0.1046	—	—	—
	Random†	1.0000	0.8003	0.0000	0.7626	0.8002	0.1900	0.1830	0.8000	0.6536
Cosmos	SLICESCORER	0.9743	0.5032	**0.0129**	0.6608	0.5348	0.1814	0.1507	0.6132	0.5208
	Embedding kNN	0.9744	0.4812	0.0123	0.6599	0.5428	**0.1846**	0.1508	0.6457	**0.5483**
	GP-Mean‡	0.9790	0.5240	0.0110	0.6660	0.5590	0.1867	—	—	—
	Random†	0.9744	0.8003	0.0205	0.6611	0.8002	0.2712	0.1511	0.8000	0.6791

nature further limits practical use. Note that even $K{=}500$ is sparse relative to the 3,840 in-ODD slices, underscoring the need for a scorer that improves gracefully with budget.

Ablations. The coverage-only variant, which optimizes rarity without neighbor-based failure propagation, underperforms on the joint objective across all datasets. For example, at $K{=}500$, Regret worsens from 0.5583 to 0.6265 on WiseAD and from 0.4855 to 0.5155 on Cosmos relative to the neighbor-only variant. Diversity also drops substantially: ILD@1000 decreases from 0.5996 to 0.5638 on WiseAD and from 0.5912 to 0.5638 on Cosmos. These results indicate that neighbor-based failure propagation is important for early failure discovery, while the full scorer provides the most stable risk–coverage tradeoff.

4.2 Analysis 2: Outside-ODD missing slice ranking

Table 2 reports outside-ODD results, where the primary objective is discovering failures beyond the declared operating domain. SLICESCORER achieves the lowest Regret@K in 7 of 9 settings, meaning it surfaces high-risk slices earlier than all competitors across nearly every VLM–budget pair; it also sweeps PO@K for all three VLMs at $K{=}500$. kNN achieves higher PO at larger budgets, but this advantage stems from embedding spread rather than failure-aware ranking: it retrieves slices nearest in embedding space without any failure signal, so its diversity is incidental. Random selection serves only as a diversity ceiling—uniform sampling is maximally diverse by construction but provides no failure prioritization. GP-Mean is again absent at $K{=}5000$ (marked N/A) and cannot operate over the 91M+ outside-ODD candidate space due to its $O(|\mathcal{S}_{\text{obs}}|^3)$ fitting cost. SLICESCORER maintains reasonable diversity (ILD 0.4–0.6) while dominating on risk capture, making it the only method that balances both objectives at scale.

4.3 Analysis 3: Agentic workflow examples with Cosmos (qualitative)

Figure 3 reports three completed runs from the qualitative sweep, selected to cover broad outside-ODD exploration, focused outside-ODD intent, and a full acquisition/evaluation loop. The examples show that SLICENAV is not only producing natural-language rationales: it selects different operator sequences from the user query, extends the ODD vocabulary when needed, and then passes the resulting candidate set to deterministic SLICESCORER ranking. In the broad case, SLICENAV uses the LLM controller (GPT 5.2) and expands multiple dimensions (*e.g.*, transitional lighting and road-surface states); in the focused case, it uses the query to add a targeted environment value (`tunnel`); in the acquisition case, it skips vocabulary extension and chains missing-slice discovery VLM evaluation. Appendix O.1 provides additional completed examples covering generic acquisition, foggy-highway retrieval/evaluation, snow/ice coverage, dawn/dusk illumination, and construction-zone expansion.

Ex.	Developer query and tools	Vocabulary selected	Top recommendations and evidence
A	**Query:** "What driving scenarios outside our current ODD should we prioritize for testing?" **Tools:** find missing in-ODD slices → extend ODD vocabulary → rank missing outside-ODD slices	`+dawn`, `+dusk`, `+puddles`, `+muddy`, `+icy`, `+light_rain`, `+heavy_rain`	**Top-2 slices:** (`clear`, `clear`, `dusk`, `city downtown`, `dry`, `intersection`, `sparse`); (`clear`, `clear`, `dusk`, `city downtown`, `wet`, `intersection`, `sparse`).
	Rationale: expands lighting and road-surface values adjacent to current high-failure rainy/intersection patterns.		
B	**Query:** "What tunnel and underpass scenarios need testing?" **Tools:** explicit vocabulary match → rank missing outside-ODD slices → summarize existing failures	`+tunnel`; no `underpass` slice surfaced in the top outside-ODD recommendations	**Top-2 slices:** (`foggy`, `clear`, `day`, `tunnel`, `dry`, `intersection`, `dense`); (`foggy`, `clear`, `day`, `tunnel`, `dry`, `non-intersection`, `dense`).
	Rationale: targets a user-specified deployment gap and varies road geometry after selecting the new environment value.		
C	**Query:** "Find untested night driving scenarios, generate synthetic test data, and run VLM evaluation." **Tools:** rank missing in-ODD slices → retrieve images → run VLM evaluation → analyze by conditions	No vocabulary extension; the controller switches from slice discovery to acquisition and evaluation	**Top-2 slices:** (`rainy`, `cloudy`, `night`, `city suburbs`, `wet`, `intersection`, `sparse`, `hazard assessment`); (`rainy`, `cloudy`, `night`, `city suburbs`, `snow`, `intersection`, `sparse`, `hazard assessment`).
	Evidence: generated 10 synthetic samples; on the 8 night slices, MCQ accuracy was 0.0 across categories, with the worst slice also scoring 0.0 VQA and 0.0 judge.		

Figure 3: **Agentic workflow examples for qualitative analysis.** Each row is one completed run from the 15-query qualitative sweep. Broad queries trigger vocabulary extension across multiple ODD dimensions, focused queries select targeted outside-ODD values, and acquisition queries chain slice discovery to data generation/retrieval and model evaluation (see rationales in Appendix O).

Appendix O.2 presents t-SNE visualizations and a comparative qualitative table. An end-to-end slice-to-executable-test pipeline is sketched in Appendix P.

4.4 Analysis 4: Reliability and stochasticity of pure LLM framework

To check the reliability and stochasticity of a pure LLM agent, we implemented a strong LLM agent baseline instructed to output missing-slice suggestions, and compared it with our tool-grounded pipeline across 15 queries (listed in the Appendix M) for the VLM Cosmos. We set GPT-5.2 as both frameworks' LLM. The baseline's prompts have been given in Appendix N. We repeated each query 10 times and recorded both slice-suggestion variance. For slice variance, we measure returned list-size variability (coefficient of variation over 'number of slices requested') and pairwise set overlap (Jaccard (Jaccard, 1901)) across runs using full slice tuples. The pure LLM baseline is highly unstable: average pairwise Jaccard is only 0.027. This shows both count drift and near-disjoint slice sets across runs. In comparison, SliceNav is far more stable: average pairwise Jaccard is 0.957.

5 Conclusions

We present SliceNav, an agentic framework for systematic ODD verification of driving VLMs that turns sparse validation coverage into traceable recommendations for what to test next. The core component, SliceScorer, deterministically ranks missing full-slice scenarios by combining two complementary signals: exposure rarity, which prioritizes under-tested regions of the ODD, and failure evidence from nearby observed slices, which directs testing toward conditions likely to reveal model weaknesses. This design makes the resulting coverage-risk trade-offs interpretable and auditable under limited testing budgets. Around this scorer, SliceNav uses an agentic layer to map developer queries to composable verification operators, including failure triage, in-ODD gap discovery, out-of-ODD stress proposal, acquisition, and evaluation. Across quantitative studies and qualitative workflows, SliceNav supports diverse, risk-aware scenario selection for broad exploration, targeted testing, and closed-loop acquisition/evaluation.

References

Mina Alibeigi, William Ljungbergh, Adam Tonderski, Georg Hess, Adam Lilja, Carl Lindström, Daria Motorniuk, Junsheng Fu, Jenny Widahl, and Christoffer Petersson. Zenseact open dataset:

A large-scale and diverse multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20178–20188, 2023.

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

Yeounoh Chung, Tim Kraska, Steven Euijong Whang, and Neoklis Polyzotis. Slice finder: Automated data slicing for model validation. *arXiv preprint arXiv:1807.06068*, 2018.

SAE On-Road Automated Vehicle Standards Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *SAE Standard J*, 3016:1, 2014.

Charles Corbière, Simon Roburin, Syrielle Montariol, Antoine Bosselut, and Alexandre Alahi. Drivingvqa: Analyzing visual chain-of-thought reasoning of vision language models in real-world scenarios with driving theory tests. *arXiv preprint arXiv:2501.04671*, 2025.

Erwin de Gelder, Maren Buermann, and Olaf Op Den Camp. Coverage metrics for a scenario database for the scenario-based assessment of automated driving systems. In *2024 IEEE International Automated Vehicle Validation Conference (IAVVC)*, pp. 1–8. IEEE, 2024.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 11198–11201, 2024.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. Domino: Discovering systematic errors with cross-modal embeddings. *arXiv preprint arXiv:2203.14960*, 2022a.

Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. Domino: Discovering systematic errors with cross-modal embeddings. In *ICLR*, 2022b.

Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76 (5):378–382, 1971.

Anurag Ghosh, Shen Zheng, Robert Tamburo, Khiem Vuong, Juan Alvarez-Padilla, Hailiang Zhu, Michael Cardei, Nicholas Dunn, Christoph Mertz, and Srinivasa G Narasimhan. Roadwork: A dataset and benchmark for learning to recognize, observe, analyze and drive through work zones. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6132–6142, 2025.

Zhijian Huang, Chengjian Fen, Feng Yan, Baihui Xiao, Zequn Jie, Yujie Zhong, Xiaodan Liang, and Lin Ma. Drivemm: All-in-one large multimodal model for autonomous driving. *arXiv preprint arXiv:2412.07689*, 2024.

International Organization for Standardization. Road vehicles — safety of the intended functionality. Standard ISO 21448:2022, International Organization for Standardization, Geneva, CH, jun 2022. URL `https://www.iso.org/standard/77490.html`.

ISO 34503:2023. Road vehicles — Test scenarios for automated driving systems — Specification for operational design domain. Standard ISO 34503:2023, International Organization for Standardization, Geneva, CH, Aug 2023. URL `https://www.iso.org/standard/78952.html`. Accessed: 05 March 2026.

Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? In *ACL*, 2020.

Mourad A Kenk and Mahmoud Hassaballah. Dawn: vehicle detection in adverse weather nature dataset. *arXiv preprint arXiv:2008.05402*, 2020.

Ritchie Lee, Mykel J. Kochenderfer, Ole J. Mengshoel, Guillaume P. Brat, and Michael P. Owen. Adaptive stress testing: Finding likely failure events with reinforcement learning. *Journal of Artificial Intelligence Research*, 69:1165–1201, 2020.

Zifan Liu, Evan Rosen, et al. Autoslicer: Scalable automated data slicing for ml model analysis. 2022.

Srikanth Malla, Chiho Choi, Isht Dwivedi, Joon Hee Choi, and Jiachen Li. Drama: Joint risk localization and captioning in driving. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 1043–1052, 2023.

Aboli Marathe, Deva Ramanan, Rahee Walambe, and Ketan Kotecha. Wedge: A multi-weather autonomous driving dataset built from generative vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3318–3327, 2023.

Ana-Maria Marcu, Long Chen, Jan Hünermann, Alice Karnsund, Benoit Hanotte, Prajwal Chidananda, Saurabh Nair, Vijay Badrinarayanan, Alex Kendall, Jamie Shotton, et al. Lingoqa: Visual question answering for autonomous driving. In *European Conference on Computer Vision*, pp. 252–269. Springer, 2024.

NVIDIA. NVIDIA Alpamayo. `https://developer.nvidia.com/drive/alpamayo`, 2025. Accessed: 2026-05-06.

NVIDIA Corporation. Cosmos-Reason2 — NVIDIA Documentation. `https://docs.nvidia.com/cosmos/latest/reason2/index.html`, 2026. Accessed: 2026-03-05.

OpenAI. Introducing GPT-5.2, 12 2025. URL `https://openai.com/index/introducing-gpt-5-2/`. Accessed: 2026-03-05.

Eliana Pastor, Luca De Alfaro, and Elena Baralis. Looking for trouble: Analyzing classifier behavior via pattern divergence. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 1400–1412, 2021.

Liang Peng, Jun Li, Wenbo Shao, and Hong Wang. Pesotif: A challenging visual dataset for perception sotif problems in long-tail traffic scenarios. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1–8. IEEE, 2023.

Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian adverse driving conditions dataset. *The International Journal of Robotics Research*, 40(4-5):681–690, 2021.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Christoffer Riis, Francisco Antunes, Frederik Hüttel, Carlos Lima Azevedo, and Francisco Pereira. Bayesian active learning with fully Bayesian Gaussian processes. In *Advances in Neural Information Processing Systems*, volume 35, pp. 12141–12153, 2022.

Svetlana Sagadeeva and Matthias Boehm. Sliceline: Fast, linear-algebra-based slice finding for ml model debugging. In *SIGMOD*, 2021.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2016.

Marcel Sheeny, Emanuele De Pellegrin, Saptarshi Mukherjee, Alireza Ahrabian, Sen Wang, and Andrew Wallace. Radiate: A radar dataset for automotive perception in bad weather. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–7. IEEE, 2021.

Sural Shounak, Naren, and Rajkumar Ragunathan. ContextVLM: Zero-shot and few-shot context understanding for autonomous driving using vision language models. *arXiv [cs.CV]*, August 2024.

Venkatesh Sivaraman, Zexuan Li, and Adam Perer. Divisi: Interactive search and visualization for scalable exploratory subgroup analysis. pp. 1–17, 2025.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025. URL https://arxiv.org/abs/2508.02324.

X.AI. Realworldqa. https://x.ai/blog/realworldqa, November 2024. Blog post.

Li Xu, He Huang, and Jun Liu. Sutd-trafficqa: A question answering benchmark and an efficient network for video reasoning over traffic events. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9878–9888, 2021.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2636–2645, 2020.

Weihao Yu, Zhengyuan Yang, Linjie Li, et al. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*, 2023.

Oliver Zendel, Katrin Honauer, Markus Murschitz, Daniel Steininger, and Gustavo Fernandez Dominguez. Wilddash-creating hazard-aware benchmarks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 402–416, 2018.

Songyan Zhang, Wenhui Huang, Zihui Gao, Hao Chen, and Chen Lv. Wisead: Knowledge augmented end-to-end autonomous driving with vision-language model. *arXiv preprint arXiv:2412.09951*, 2024.

Lianmin Zheng et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.

Xingcheng Zhou, Konstantinos Larintzakis, Hao Guo, Walter Zimmer, Mingyu Liu, Hu Cao, Jiajie Zhang, Venkatnarayanan Lakshminarasimhan, Leah Strand, and Alois C Knoll. TUMTraffic-VideoQA: A benchmark for unified spatio-temporal video understanding in traffic scenes. *arXiv [cs.CV]*, February 2025.

Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web (WWW)*, pp. 22–32, 2005.

SUPPLEMENTARY MATERIAL

CONTENTS

A AI Use Statement **14**

B Ethics Statement **14**

C Reproducibility Statement **14**

D Limitations and Broader Impacts **14**

E Grounding Slice Dimensions in ODD Standards **14**

E.1 Extended ODD Vocabulary 15

F Baseline Adaptation Details **16**

F.1 Task Mismatch 16

F.2 SliceLine Adaptation 16

F.3 SliceFinder-Inspired Marginal-Lift Model 16

F.4 Hyperparameters 17

F.5 Limitations of Adaptations 17

G LLM-as-a-Judge via VLMEvalKit **17**

H Dataset collection for evaluations in this paper **18**

I Datastore for retrieval with SLICENAV **19**

J Ground-Truth Data Construction **20**

K Oracle Ranking Methodology **20**

L Experimental protocols for Analysis 1–4 **20**

M Analysis 4 query set (15 prompts) **21**

N LLM choose-slices baseline prompts **21**

O Full rationales for qualitative workflow examples **23**

O.1 Additional qualitative examples 24

O.2 Qualitative visualizations 24

P Slice-to-executable-test validation pipeline **25**

A AI USE STATEMENT

Generative AI was used for editing (grammar, spelling, word choice). As described in Section 4, Qwen3-30B was used for ODD attribute annotation of collected data, GPT-5.2 for QA-pair categorization and as the LLM controller in SLICENAV, and GPT-4.1 as the LLM evaluation judge. It was not used for idea generation, experiment design, or code writing. All AI-generated annotations were verified through a 4-expert audit (Section 4), and all AI-assisted work was reviewed by the authors.

B ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics and aims to improve autonomous driving safety by identifying untested VLM failure modes before deployment. All experiments use publicly available datasets from 15 previously published sources; no proprietary data was collected. Evaluations are entirely computational—no human subjects were involved and no real vehicles were deployed. Knowledge of failure patterns is reported as aggregated metrics rather than exploitable attack vectors. We have no conflicts of interest.

C REPRODUCIBILITY STATEMENT

All hyperparameters are listed in Section 4; baseline adaptation details and hyperparameters in Appendix F; experimental protocols for Analysis 1–4 in Appendix L; LLM judge prompts in Appendix G; and dataset collection in Appendix H.

D LIMITATIONS AND BROADER IMPACTS

Limitations. While SLICENAV systematically identifies critical testing gaps, it depends on the granularity and comprehensiveness of the underlying ODD taxonomy and the seed evaluation table. If the taxonomy lacks certain environmental factors or if the seed table is excessively sparse, the SLICESCORER might struggle to extrapolate neighbor failures reliably. Furthermore, while the LLM-driven candidate generation is constrained to logical ODD extensions, its output quality depends on the underlying LLM's reasoning capabilities. Adaptive stress testing methods (Lee et al., 2020) address a complementary problem: given a simulator, searching for concrete failure trajectories within a specific scenario. SliceScorer can first identify high-priority ODD regions, after which AST may search for failure trajectories within those regions.

Broader Impacts. By providing a structured, traceable framework to uncover "unknown unknown" failures in driving VLMs, our work supports systematic evaluation of scene understanding components increasingly used in autonomous driving stacks. Improved VLM evaluation methodologies help identify conditions where perception may degrade, informing downstream safety validation efforts. However, developers must not treat SLICESCORER as a definitive guarantee of safety; it is a supplementary recommendation tool intended to prioritize testing efforts, not a complete proof of ODD coverage.

E GROUNDING SLICE DIMENSIONS IN ODD STANDARDS

Our slice space dimensions are derived from established ODD specification standards. Table 3 provides a detailed mapping of each slice dimension to specific clauses in ISO 34503:2023 (ISO 34503:2023), ISO 21448:2022 (SOTIF) (International Organization for Standardization, 2022), and SAE J3016 (Committee et al., 2014).

Standards Overview. ISO 34503:2023 defines a hierarchical ODD taxonomy with three top-level categories (Clause 8.2): *scenery elements* (roads, junctions), *environmental conditions* (weather, illumination), and *dynamic elements* (traffic agents). ISO 21448 (SOTIF) identifies triggering conditions that can activate functional insufficiencies in perception systems, with Annex B providing a 6-layer scenario factor model. SAE J3016 defines ODD as operating conditions including "environmental,

geographical, and time-of-day restrictions, and/or the requisite presence or absence of certain traffic or roadway characteristics" (Section 3.21).

Table 3: **Mapping of slice dimensions to ODD standards.** Clause references are aligned with ISO 34503:2023 taxonomy structure and ISO 21448 Annex B layers. This table provides an alignment between our slice dimensions and representative ODD/scenario-factor concepts in ISO 34503, ISO 21448, and SAE J3016; the mapping is intended to justify coverage of major ODD-relevant factors rather than to reproduce each standard's taxonomy verbatim.

Slice Dimension	ISO 34503:2023	ISO 21448 (SOTIF)	SAE J3016 §3.21
`weather visibility`	Clause 10 (Environmental conditions: weather, visibility)	SOTIF scenario factors: (Environmental conditions)	"environmental"
`sky conditions`	Clause 10 (Environmental conditions: illumination)	SOTIF scenario factors: (Environmental conditions)	"environmental"
`time`	Clause 10 (Environmental conditions: illumination/day-night)	SOTIF scenario factors: (Environmental conditions)	"time-of-day restrictions"
`environment`	Clause 9 (Scenery elements: operational area, road context)	SOTIF scenario factors: (Road-level environment)	"geographical"
`road condition`	Clause 9 (Scenery elements: surface condition)	SOTIF scenario factors: (Road-level environment)	"roadway characteristics"
`road type`	Clause 9 (Scenery elements: road type, junctions)	SOTIF scenario factors: (Road-level environment)	"roadway characteristics"
`traffic type`	Clause 11 (Dynamic elements: traffic participants)	SOTIF scenario factors: (Dynamic elements)	"traffic characteristics"

Coverage and Scope. Our seven ODD-derived dimensions cover all three top-level ISO 34503 categories: environmental conditions (`weather_visibility`, `sky_conditions`, `time`), scenery elements (`environment`, `road_condition`, `road_type`), and dynamic elements (`traffic_type`). The `question_category` dimension is VLM-specific and captures the type of reasoning task (e.g., scene observation, hazard assessment, future prediction); it is not derived from ODD standards but is essential for evaluating VLM capabilities across different question modalities.

Extensibility. ISO 34503 Clause 8.1 explicitly allows stakeholders to extend the taxonomy with additional attributes when the base taxonomy does not adequately represent a specific operating environment. Our framework supports this extensibility: developers can add new dimensions or refine existing value sets as needed for their specific ODD verification requirements.

E.1 Extended ODD Vocabulary

To enable controlled stress testing beyond the declared operating domain, we define an extended ODD vocabulary that refines and expands each base dimension as follows.

Weather and visibility. Weather is decomposed into finer-grained intensities (e.g., `rain_light`, `rain_heavy`, `fog_dense`, `snow_heavy`) and additional phenomena (e.g., `dust`, `smoke`, `haze`).

Sky and illumination. Sky and illumination are extended with conditions such as `partly_cloudy`, `overcast`, `sun_glare`, `dawn`, and `dusk`.

Environment and road context. Environmental and road context dimensions are broadened to include specific scene types (e.g., `industrial_area`, `parking_lot`, `tunnel`, `bridge`, `rural_road`) and more detailed road structures (e.g., `roundabout`, `merge_ramp`, `intersection_signalized`).

Traffic and road users. Traffic is refined into finer density regimes (e.g., `moderate`, `stop_and_go`, `traffic_jam`), and the previously empty `road_users_present` dimension is introduced to explicitly model interacting agents (e.g., `pedestrians`, `cyclists`, `mixed_vulnerable_users`).

Design notes. The extended ODD provides a structured superset of the base domain, enabling systematic exploration of out-of-domain conditions. Dimensions that are not active in the base ODD, such as `road_users_present`, are assigned a default base value `unspecified`; outside-ODD expansion adds additional values to the same dimension rather than introducing a new tuple axis. Discretization is a deliberate engineering choice that makes the ODD space navigable, measurable, and auditable, rather than an assumption that real-world driving conditions are inherently discrete. Continuous attributes such as visibility distance or rainfall intensity can be incorporated as meaningful ranges without changing SliceScorer's scoring logic.

Figure 4: SliceLine Adaptation for Missing-Slice Ranking

```
Require: Triage table T with columns {d_1, ..., d_K} and error e_i = 1 − y_i
Require: Candidate full slices S
Ensure: Ranked list of slices
1: Fit SliceLine on T with error e; extract P = {(C_j, σ_j)}_{j=1}^m // C_j: constraints, σ_j: score
2: for each slice s ∈ S do
3: M_s ← {(C_j, σ_j) ∈ P : s satisfies C_j}
4: if M_s ≠ ∅ then
5: (C*, σ*) ← arg max_{(C,σ)∈M_s} σ
6: assign s: specificity = |C*|, score = σ*, rank = j*
7: return slices sorted by (specificity desc, rank asc, score desc, slice id asc)
```

F Baseline Adaptation Details

SliceLine (Sagadeeva & Boehm, 2021) and SliceFinder (Chung et al., 2018) were originally designed for a different task: discovering high-error *observed* subpopulations (partial predicates with non-zero support). Our task is ranking *missing* full slices (zero-support combinations) by predicted failure risk. This section documents how we adapt these methods for fair comparison.

F.1 Task Mismatch

Original task: Given a dataset with observed examples, find partial predicates (e.g., `weather_visibility=foggy AND time=night`) that have high error rates. These predicates describe subpopulations *present* in the data.

Our task: Given a sparse validation set, rank *missing* full slices (complete assignments of all ODD dimensions with zero observed support) by predicted failure risk, to guide acquisition of new test cases.

Neither SliceLine nor SliceFinder directly addresses missing-slice ranking. We therefore adapt each method as described below.

F.2 SliceLine Adaptation

SliceLine uses a lattice-based search to find partial predicates (subsets of dimension-value constraints) with statistically significant error rates. We adapt it for missing-slice ranking via predicate expansion:

1. **Fit SliceLine** on the observed triage table, treating $\epsilon(s)$ as the error signal.
2. **Extract predicates** from SliceLine's top-k discovered slices, each with an associated error score.
3. **Expand to full slices:** For each candidate full slice s, find all predicates whose constraints are satisfied by s. Assign s the score of its best-matching (highest-scoring) predicate.
4. **Rank by specificity then discovery order:** Sort matched slices first by predicate specificity (number of constrained dimensions, descending), then by predicate discovery rank (ascending), then by predicate score (descending), with slice identifier as the final deterministic tie-breaker.

This expansion allows SliceLine's partial-predicate signal to propagate to full slices, including those with zero support.

F.3 SliceFinder-Inspired Marginal-Lift Model

The original SliceFinder uses decision-tree-based partitioning to discover interpretable high-error subgroups. Since decision trees require observed support, we adapt the core intuition—that slice error can be decomposed into interpretable factors—using an *additive marginal-lift model*.

For each dimension d and value v, we compute the marginal lift:

$$\text{lift}(d, v) = \mathbb{E}[\text{badness} \mid d = v] - \mathbb{E}[\text{badness}]$$

where badness $= 1 -$ mean_llm_judge_score and expectations are weighted by sample count.

Figure 5: Marginal-Lift Model (SliceFinder-Inspired)

Require: Triage table T with dimensions $\{d_1, \ldots, d_K\}$, weights w_i, scores y_i
Require: Untested full slices $\mathcal{U}$
Ensure: Ranked list of untested slices
$\bar{b} \leftarrow \sum_i w_i(1-y_i) / \sum_i w_i$ // global badness
for each dimension d_k and value $v \in \text{vocab}(d_k)$ **do**
  $\bar{b}_{d_k=v} \leftarrow \sum_{i:x_i[d_k]=v} w_i(1-y_i) / \sum_{i:x_i[d_k]=v} w_i$
  $\text{lift}(d_k, v) \leftarrow \bar{b}_{d_k=v} - \bar{b}$
for each untested slice $s = (v_1, \ldots, v_K) \in \mathcal{U}$ **do**
  $\widehat{b}(s) \leftarrow \text{clip}\left(\bar{b} + \sum_{k=1}^{K} \text{lift}(d_k, v_k),\ 0,\ 1\right)$
return $\mathcal{U}$ sorted by $\widehat{b}(s)$ descending

For an untested (zero-support) full slice $s = (v_1, \ldots, v_K)$, we predict:

$$\widehat{\text{badness}}(s) = \text{clip}\left(\mathbb{E}[\text{badness}] + \sum_{k=1}^{K} \text{lift}(d_k, v_k),\ 0,\ 1\right)$$

Slices are ranked by predicted badness (descending).

F.4 Hyperparameters

Table 4 lists the hyperparameters used for each baseline.

Table 4: Baseline adaptation hyperparameters.

Method	Parameter	Value	Description
SliceLine	k	1000–5000	Number of predicates extracted
SliceLine	max_l	8	Maximum predicate length (all dimensions)
SliceLine	min_sup	1	Minimum support for predicates
SliceLine	α	0.95	Effect size threshold
Marginal-Lift	rule_length	8	Full slices only (all dimensions)
Marginal-Lift	top_k	1000–5000	Number of ranked slices retained

F.5 Limitations of Adaptations

We acknowledge that these adaptations are approximations:

- **SliceLine:** Partial predicates may over-generalize; a high-error predicate like `weather_visibility=foggy` propagates its score to all foggy slices regardless of other dimensions. The specificity-first ranking is a heuristic without theoretical justification.
- **Marginal-Lift:** The additive decomposition ignores dimension interactions. For example, "foggy + night" may be substantially worse than the sum of marginal effects for foggy and night individually.

Despite these limitations, we believe these adaptations represent reasonable baselines for a task the original methods were not designed for. The fact that SliceScorer outperforms both adaptations suggests its design—combining coverage priors with neighbor-based failure propagation—is better suited to missing-slice ranking.

G LLM-as-a-Judge via VLMEvalKit

Description. We use VLMEvalKit (Duan et al., 2024) to score VLM predictions with an LLM judge (GPT-4.1 in our case) to decide whether each predicted answer is correct or incorrect. No image or video is sent to the judge; only text (question, options if MCQ, predicted answer, ground-truth answer) is passed.

Components. We build the judge using VLMEvalKit's standard judge-construction helper and its packaged default system prompt, reproduced in Listing 1. For the user prompt, we follow Listing 2 for multiple-choice questions and Listing 3 for open-ended visual QA. For MCQ, the prompt includes the question text, the option list (formatted as "A. ...", "B. ..."), the model's predicted answer, and the ground-truth answer. For open-ended VQA, it includes the question, the model's free-text response, and the reference answer text (including alias keys when provided by the benchmark).

Listing 1: LLM Judge System Prompt

```
{"task": "Evaluate whether the answer to a question is correct.",
 "requirements": [
  "Compare an answer to a question with the ground truth answer. Determine whether it is
       correct.",
  "You must ignore any analysis of the problem if present. You must focus only on the final
       answer.",
  "You must answer in the following json format: {"verdict": "(1 for correct, 0 for incorrect)
       "}"
 ]}
```

Listing 2: LLM Judge User Prompt (MCQ setting)

```
You are an AI assistant evaluating a multiple-choice question answer.

Question: <question>

Options:
A. <option_0>
B. <option_1>
...

Predicted Answer: <predicted_answer>
Ground Truth Answer: <ground_truth_answer>

Please judge whether the predicted answer is correct. Respond with only "CORRECT" or "
     INCORRECT".
```

Listing 3: LLM Judge User Prompt (VQA or open-ended setting)

```
You are an AI assistant evaluating a visual question answering response.

Question: <question>

Predicted Answer: <predicted_text>
Ground Truth Answer: <ground_truth_text>

Please judge whether the predicted answer is correct. Consider semantic equivalence - answers
     that convey the same meaning should be considered correct even if worded differently.
     Respond with only "CORRECT" or "INCORRECT".
```

H DATASET COLLECTION FOR EVALUATIONS IN THIS PAPER

We collected 15 datasets spanning diverse conditions and attributes to populate the oracle ranking table with broad slice coverage under the base and extended ODD definitions. Table 5 summarizes the statistics of the seed evaluation table. The seed set was constructed with two key design choices: (1) each slice contains at least 10 examples to ensure statistical reliability, and (2) all QA annotations were human-generated to maximize label quality and support trustworthy downstream decisions. Table 6 reports the statistics of the oracle ranking dataset pool used to evaluate all VLMs and construct the ground-truth failure triage. Importantly, there is no overlap in examples between the seed evaluation table and the ground-truth slice-based evaluation set.

Human verification of predicted labels. Four experts independently judged whether each model-proposed scene attribute and QA field was correct (*Yes*/*No*/*Unsure*) on 100 stratified examples (same pool for all raters). Pooled inter-rater agreement over all judgments yields Fleiss' κ=0.2395 and mean observed pairwise agreement $\bar{P}$=0.8263. Our dataset collection consists of both images and videos and covers multiple QA task types, including multiple-choice (MCQ) and open-ended VQA. To annotate pixel-level attributes aligned with the ODD dimensions, we use the Qwen3-30B model. For constructing QA pairs corresponding to the ODD `question_category` axis, we proceed in two

stages. First, we classify ground-truth QA pairs from datasets that already contain human-generated annotations. These labeled examples are then used as few-shot demonstrations to prompt GPT-5.2, which generates QA pairs for datasets that lack original QA annotations.

Slice signature construction. Each test case (one visual input and its associated QA) is assigned a *slice signature*: a single tuple over the same eight discrete ODD fields used in the main text (seven scene/context dimensions plus `question_category`). The seven scene-related fields are produced by the Qwen3-30B vision–language annotator under *forced choice*: for each field, the model must select exactly one label from that dimension's finite in-ODD vocabulary (there are no continuous raw measurements in the signature; discretization is implicit in this closed-set prediction). The `question_category` label comes from dataset metadata when the QA is native to the benchmark; for QA pairs produced by the GPT-5.2 generation stage above, the category is assigned in that pipeline and mapped to the same closed `question_category` set. For *video* clips, the VLM is given the full temporal input and returns one global label per scene field (a single seven-tuple for the clip, not a separate tuple per frame). Test cases that share the same eight-tuple are grouped into the same slice for aggregation and scoring.

Table 5: **Seed Evaluation set.** Dataset, number of examples, QA type (MCQ/VQA), and number of unique slices.

Dataset	# Examples	QA type	# Unique slices
Context-VLM (Shounak et al., 2024)	941	MCQ	156
DRAMA (Malla et al., 2023)	875	MCQ	131
EPFL (Corbière et al., 2025)	70	MCQ	37
LingoQA (Marcu et al., 2024)	19	VQA	13
RealWorldVQA (X.AI, 2024)	22	VQA	17
SUTD (Xu et al., 2021)	451	MCQ	132
TUMTraffic-VQA (Zhou et al., 2025)	572	MCQ	118

Table 6: **Oracle Evaluation Dataset.** Sample counts for datasets used in oracle.

Dataset	Count
DRAMA (Malla et al., 2023)	104,481
Context-VLM (Shounak et al., 2024)	34,267
TUMTraffic-VQA (Zhou et al., 2025)	24,309
SUTD (Xu et al., 2021)	5,624
LingoQA (Marcu et al., 2024)	962
EPFL (Corbière et al., 2025)	929
RealWorldVQA (X.AI, 2024)	522
Canadian Adverse Driving Conditions dataset (Pitropov et al., 2021)	6,899
DAWN (Kenk & Hassaballah, 2020)	1,027
PESOTIF (Peng et al., 2023)	1,036
RADIATE (Sheeny et al., 2021)	32,997
ZOD (Alibeigi et al., 2023)	99,998
WEDGE (Marathe et al., 2023)	3,360
WildDash2 (Zendel et al., 2018)	4,256
Roadworks (Ghosh et al., 2025)	8,549

I DATASTORE FOR RETRIEVAL WITH SLICENAV

SLICENAV uses a retrieval datastore to ground slice recommendations in existing driving footage before additional simulation or evaluation is requested. The datastore aggregates large-scale autonomous-driving video sources with complementary coverage: nuPlan (Caesar et al., 2021) contributes geographically diverse closed-loop driving logs , NVIDIA Alpamayo/Physical AI open datasets (NVIDIA, 2025) contribute large-scale long-tail driving data for autonomous-vehicle development,

and BDD100K (Yu et al., 2020) contributes diverse road-scene videos spanning weather, time, and scene conditions. Table 7 summarizes the video inventory used for retrieval. If datastore retrieval returns no matching examples for a recommended slice, we generate backup synthetic examples for that slice using Qwen-Image (Wu et al., 2025).

Table 7: **Retrieval datastore used by SLICENAV.** Video counts are approximate and describe the source inventory available for retrieval.

Dataset	Videos
nuPlan (Caesar et al., 2021)	∼14,000
Alpamayo (NVIDIA, 2025)	1,000,000
BDD100K (Yu et al., 2020)	56,413

J GROUND-TRUTH DATA CONSTRUCTION

The full ODD slice space forms a large Cartesian product (3,840 in-ODD; 91M+ outside-ODD), making exhaustive coverage infeasible. We aggregated images and videos from 15 datasets focused on diverse weather and driving conditions (see Appendix H for full dataset listings and statistics). Scene attributes were annotated using Qwen3-30B (Yang et al., 2025); QA categories were assigned using GPT-5.2 (OpenAI, 2025) with few-shot demonstrations from existing labeled pairs. Four domain experts independently verified 100 stratified examples across nine fields (seven ODD attributes, QA-category fit, and image relevance), producing 3,600 field-level decisions with Fleiss' κ=0.2395 (Fleiss, 1971) and mean pairwise agreement of 0.8263; this audit evaluates pipeline reliability, not per-example manual verification (see Appendix H for the full verification protocol). For acquisition-based workflows in Analysis 4.3, SLICENAV queries a retrieval datastore described in Appendix I.

K ORACLE RANKING METHODOLOGY

To evaluate slice recommendation quality, we construct an oracle slice table from a large, independently collected dataset (see Appendix H, Table 6) that is disjoint from the seed validation set (see Appendix H, Table 5). For each slice s, we evaluate the VLM on all corresponding oracle examples and compute per-example correctness using an LLM-based judge. Let $y_i \in [0, 1]$ denote the judge score for oracle example i in slice s. We aggregate these scores into the slice-level oracle estimate

$$\bar{y}_{\text{oracle}}(s) = \frac{1}{N_{\text{oracle}}(s)} \sum_{i=1}^{N_{\text{oracle}}(s)} y_i, \tag{11}$$

where $N_{\text{oracle}}(s)$ is the number of oracle examples in slice s. Matching the notation in Section 3.4, we define the oracle slice failure risk as $\epsilon_{\text{oracle}}(s) = 1 - \bar{y}_{\text{oracle}}(s)$. The oracle ranking is obtained by sorting slices in descending order of $\epsilon_{\text{oracle}}(s)$, which serves as a high-confidence approximation of slice-level failure risk. To build the judge and prompt, we reused VLMEvalKit (Duan et al., 2024) functions for consistency; prompts and interface details are provided in Appendix G.

L EXPERIMENTAL PROTOCOLS FOR ANALYSIS 1–4

Analysis 1 (in-ODD missing-slice ranking): We rank missing in-ODD slices from the seed triage table (no user query) using SLICESCORER and adapted baselines (Sagadeeva & Boehm, 2021; Chung et al., 2018), and evaluate with paired Regret@K and ILD@K against the oracle relevance table.

Analysis 2 (outside-ODD ranking): We rank slices from a shared expanded outside-ODD candidate pool with SLICESCORER, adapted baselines, embedding-kNN, and random, and evaluate again with paired Regret@K and ILD@K.

Analysis 3 (qualitative workflow examples): We run 15 developer queries through the full agent and report representative completed runs showing query interpretation, optional ODD expansion, deterministic slice ranking, and acquisition/evaluation chaining.

Analysis 4 (reliability and stochasticity): We repeat the 15-query set across 10 runs for both baseline LLM agent and SLICENAV, and quantify run-to-run variation in returned slices (set-overlap Jaccard) (Appendix M).

M ANALYSIS 4 QUERY SET (15 PROMPTS)

For the Analysis 4 reliability/stochasticity study, we use the following 15 developer prompts and repeat each prompt across 10 runs.

1. What driving scenarios outside our current ODD should we prioritize for testing?
2. Identify high-risk conditions we haven't validated yet.
3. What blind spots exist in our current test coverage?
4. Suggest untested edge cases that could cause failures in deployment.
5. We're expanding to construction zones - what untested scenarios should we prioritize?
6. What snow and ice scenarios are we missing?
7. What nighttime edge cases should we test?
8. We're deploying on rural highways - what scenarios haven't we covered?
9. What tunnel and underpass scenarios need testing?
10. We're concerned about heavy rain - what wet road scenarios should we prioritize?
11. What rush hour congestion scenarios are undertested?
12. What dawn and dusk visibility scenarios are we missing?
13. Find high-risk untested scenarios and acquire test samples for evaluation.
14. Identify missing foggy highway slices, retrieve test images, and evaluate the model.
15. Find untested night driving scenarios, generate synthetic test data, and run VLM evaluation.

N LLM CHOOSE-SLICES BASELINE PROMPTS

This section reproduces the prompts used by the LLM baseline in Analysis 4. Note that the baseline conditions on a *compact text summary* of the seed triage CSV - marginals, support statistics, and aggregate judge scores - not the verbatim full per-row table in the prompt. At our seed-inventory scales, embedding the complete table would exceed practical context-length budgets for general-purpose chat models, so verbatim full-table conditioning is infeasible for this listing baseline. Variance and request-vs.-fulfillment analyses for LLM listing should be read under this summary-conditioned regime. Repeated listing calls use the API *default* decoding settings (we do not set temperature or sampling knobs to deterministic values), so run-to-run variability reflects ordinary chat-model behavior rather than a controlled greedy decode.

Listing 4: Choose-slices baseline system prompt

```
You are an expert analyzing VLM (Vision-Language Model) failure patterns in autonomous driving
     scenarios.

Your task is to choose which outside-ODD slices you recommend for testing.

"Outside-ODD" means: at least one attribute value in the slice is in the extended ODD but NOT
    in the base ODD (i.e. not yet covered by triage).

You will be given:
- A summary of triage data (tested slices and performance)
- Base ODD and extended ODD definitions

Respond with ONLY a CSV of your chosen outside-ODD slices. Do not generate any code. Use the
    exact dimension names as column headers. One row per slice.
```

Listing 5: Choose-slices baseline instructions prompt

```
## Triage Summary (observed slices inside ODD)
```

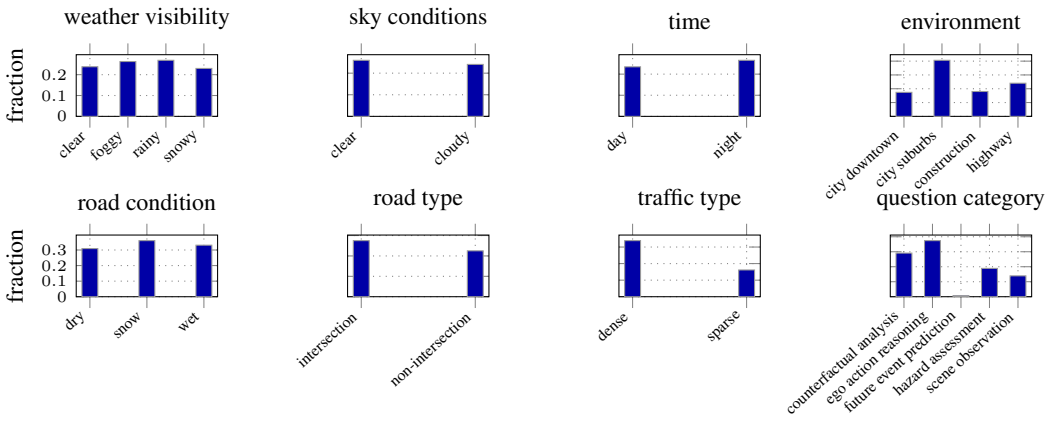


Figure 6: **WiseAD SliceScorer recommendation profile.** Distribution of top-ranked missing-slice recommendations across ODD attributes.

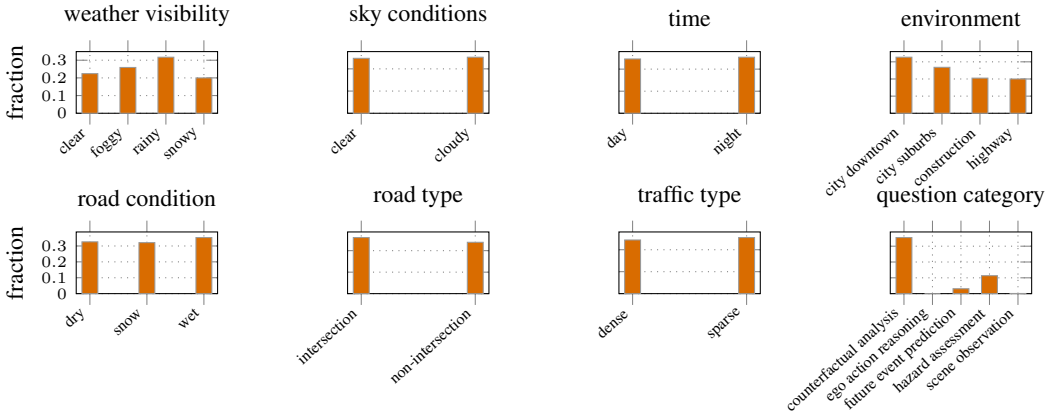


Figure 7: **DriveMM SliceScorer recommendation profile.** Distribution of top-ranked missing-slice recommendations across ODD attributes.

```
{triage_summary}

## Slice Dimensions (use these exact names as CSV columns)

{slice_cols}

## Base ODD Values (inside ODD - already tested)

{base_odd}

## Extended ODD Values (superset - choose outside-ODD slices from these)

{extended_odd}

## Task

Choose the outside-ODD slices you recommend for testing. Each chosen slice must have at least
     one attribute value that is in the extended ODD but NOT in the base ODD.

Output your chosen slices as CSV only:
- First line: header with exactly these columns (comma-separated): {csv_header}
- Following lines: one row per slice, same column order, comma-separated values
- Strictly Output {top_k} slices.

Do not include any explanation or code. Only the CSV (you may wrap it in a ```csv code block).
```



Figure 8: **Cosmos SliceScorer recommendation profile.** Distribution of top-ranked missing-slice recommendations across ODD attributes.

Figures 6–8 show the distribution of top-ranked missing-slice recommendations across ODD attributes for each VLM.

O FULL RATIONALES FOR QUALITATIVE WORKFLOW EXAMPLES

Broad outside-ODD query (Example A). For the broad prompt, SLICENAV first inspected the current triage table and found that failures concentrate around rainy conditions, intersections, hazard assessment, ego action reasoning, and counterfactual reasoning. Because the user asked for scenarios outside the current ODD, the controller expanded the ontology before ranking missing slices. The selected additions expose distribution shifts adjacent to the observed failure modes: `dawn` and `dusk` test transitional lighting, `puddles`, `muddy`, and `icy` test road-surface states beyond `dry`/`wet`/`snow`, and `light_rain`/`heavy_rain` refine the coarse `rainy` bucket. The top recommendations therefore prioritize downtown intersections at dusk with dry or wet surfaces, followed by nighttime puddle/muddy variants, rather than listing arbitrary exotic conditions.

Focused outside-ODD query (Example B). For the tunnel/underpass prompt, the query itself supplied a target deployment environment. The controller therefore skipped broad LLM vocabulary search and inserted the explicit new value `tunnel` into the environment dimension. The outside-ODD missing-slice ranker then returned top slices all using `environment=tunnel`, concentrated on foggy daytime dense-traffic cases while varying road condition and road type. A separate failure-summary pass found no tunnel or underpass groups in the existing triage failures, supporting the interpretation that these are coverage gaps rather than already-tested weak spots. Underpass did not appear in the top recommendations, so the paper reports the evidenced tunnel result instead of inventing unsupported underpass slices.

Full acquisition loop (Example C). For the night-driving acquisition prompt, the controller stayed inside the base ODD because the requested value `night` already exists in the taxonomy. It ranked missing night slices, then chained the recommendation to synthetic data generation and VLM evaluation. The highest-priority gaps were rainy nighttime city-suburb intersections for hazard assessment, varying road surface and traffic density, with additional night gaps for counterfactual highway reasoning and dense downtown ego-action reasoning. The run generated 10 synthetic samples, one per candidate slice, and evaluated Cosmos-Reason2-2B. Restricting the summary to the 8 night slices, all categories had MCQ accuracy 0.0; the worst slice was rainy/cloudy/night/city-suburbs/wet/intersection/sparse/hazard-assessment, with MCQ accuracy 0.0, VQA accuracy 0.0, and mean judge score 0.0.

O.1 Additional qualitative examples

Generic high-risk acquisition (Example D). **Query:** "Find high-risk untested scenarios and acquire test samples for evaluation." **Tools:** within-ODD missing-slice ranking → datastore retrieval → simulation fallback for uncovered slices → dataset inspection. **Vocabulary:** no extension; this run stayed within the base ODD. **Top-2 slices:** (rainy, cloudy, night, city suburbs, wet, intersection, sparse, hazard assessment) and (rainy, cloudy, night, city suburbs, snow, intersection, sparse, hazard assessment). **Rationale:** the run shows that acquisition is a workflow decision, not only a ranking output: most slices were acquired from the datastore, while a missing retrieval was filled by simulation. The response also flags slice-fidelity risks from returned data fields, motivating post-filtering before treating retrieved samples as valid slice coverage.

Foggy-highway retrieval and evaluation (Example E). **Query:** "Identify missing foggy highway slices, retrieve test images, and evaluate the model." **Tools:** foggy-highway gap enumeration → within-ODD slice ranking → datastore retrieval → VLM evaluation → condition-level analysis. **Vocabulary:** no extension; the query constrains existing ODD values. **Top-2 intended slices:** (foggy, cloudy, day, highway, dry, intersection, sparse, ego action reasoning) and (foggy, cloudy, day, highway, dry, intersection, sparse, hazard assessment). **Rationale:** the combinatorial audit finds 225 missing foggy-highway combinations out of 240 possible combinations. The retrieval and evaluation step is especially useful as evidence because the returned samples collapse to only two realized non-intersection dry daytime foggy-highway slices with missing question category, showing why acquisition workflows need slice-fidelity checks in addition to ranking.

Snow and ice coverage (Example F). **Query:** "What snow and ice scenarios are we missing?" **Tools:** within-ODD missing-slice ranking → refined top-K snow query → existing-failure summary. **Vocabulary:** no automatic extension; the artifact notes that `ice` is not an explicit road-condition value, so `road_condition=snow` is used as the closest snow/ice proxy. **Top-2 slices:** (rainy, cloudy, night, city suburbs, snow, intersection, sparse, hazard assessment) and (clear, cloudy, night, city downtown, snow, intersection, dense, ego action reasoning). **Rationale:** the run surfaces mixed-precipitation and snowy-surface gaps at intersections, then separates hazard assessment gaps from a secondary ego-action reasoning gap in dense urban night traffic.

Dawn/dusk illumination extension (Example G). **Query:** "What dawn and dusk visibility scenarios are we missing?" **Tools:** outside-ODD vocabulary expansion → outside-ODD missing-slice ranking → targeted inspection of dawn/dusk outputs. **Vocabulary:** `+dawn` and `+dusk` are added to the `time` dimension. **Top-2 slices:** (clear, clear, dusk, city downtown, dry, intersection, sparse) and (clear, clear, dusk, city downtown, wet, intersection, sparse). **Rationale:** unlike the broad outside-ODD query, this focused prompt produces a narrow twilight cluster: clear downtown sparse-traffic scenes at dawn or dusk, varying road surface and road geometry. The returned table omits question category, so the rationale explicitly treats this as a scene-condition coverage gap.

Construction-zone expansion (Example H). **Query:** "We're expanding to construction zones – what untested scenarios should we prioritize?" **Tools:** within-ODD missing-slice ranking → existing-failure summary → construction-specific gap analysis. **Vocabulary:** no extension; `environment=construction` already exists in the base ODD. **Top-2 slices:** (rainy, cloudy, night, construction, snow, intersection, dense, hazard assessment) and (rainy, clear, night, construction, wet, intersection, dense, hazard assessment). **Rationale:** the construction-specific audit finds a large gap: 26 evaluated construction slices versus 934 untested construction combinations. The prioritized slices combine adverse weather, night driving, dense traffic, intersections, and hazard assessment, matching the safety-critical conditions a deployment expansion should test first.

O.2 Qualitative visualizations

Figures 9 and 10 show t-SNE visualizations of the slice-space spread for DriveMM and Cosmos respectively. Figure 11 presents a comparative qualitative table of example slices selected by each method.

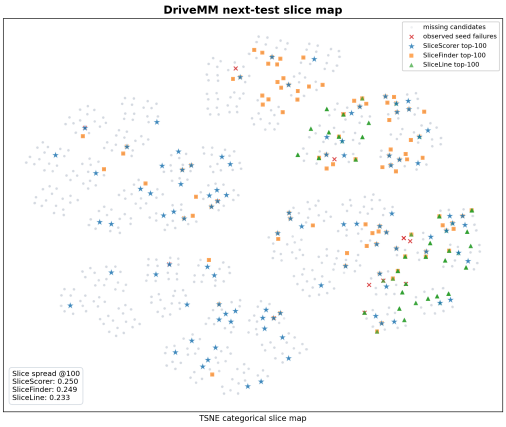


Figure 9: t-SNE slice-space visualization for DriveMM, comparing the spread of top-K recommendations from SLICESCORER, SliceFinder, and SliceLine. Each point represents a recommended slice; color encodes the method. SLICESCORER recommendations cover a broader region of the slice space, reflecting higher intra-list diversity.

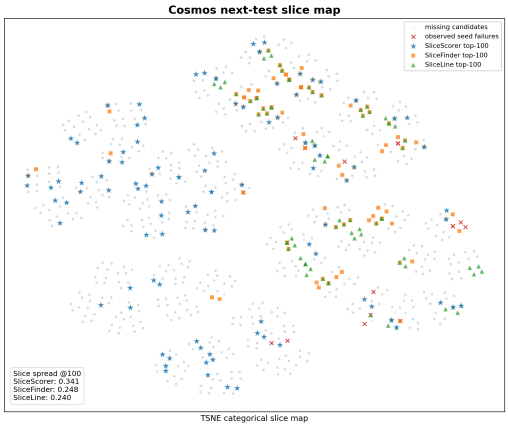


Figure 10: t-SNE slice-space visualization for Cosmos-Reason2-2B, analogous to the DriveMM visualization. SLICESCORER again produces more spatially dispersed recommendations compared to baselines.

Case	VLM	Method	Slice	Ranks O / SF / SL	Evidence
SliceScorer-prioritized high-risk success	DriveMM	SS	`clear, clear, night, downtown; snow, intersection, sparse, counterfactual`	23 / 239 / 2071	score=1.740; rarity=0.640; NN risk=0.800, d=2
Common high-risk success	WiseAD	SS	`foggy, clear, night, highway; dry, non-int., dense, ego action`	56 / 3 / 80	score=1.589; rarity=0.659; NN risk=0.500, d=1
SliceScorer false positive	Cosmos	SS	`rainy, cloudy, night, suburbs; wet, intersection, dense, hazard`	6 / – / 2736	score=1.691; rarity=0.677; NN risk=0.200, d=1
Baseline-prioritized high-risk success	WiseAD	SF	`clear, clear, night, suburbs; dry, intersection, dense, ego action`	1028 / 68 / 49	additive risk=0.477; rarity=0.625
SliceLine-prioritized high-risk success	Cosmos	SL	`clear, clear, day, suburbs; wet, non-int., dense, ego action`	343 / 273 / 40	N-5 predicate; score=0.552; 63 tuples

Note. O=SLICESCORER, SF=SliceFinder, SL=SliceLine; NN=nearest observed slice; d=categorical Hamming distance.

Figure 11: Comparative qualitative table showing example slices selected by each method (SLICESCORER, SliceFinder, SliceLine) with columns for slice attributes, predicted risk, oracle failure rate, and evidence sources. SLICESCORER prioritizes diverse, high-risk slices while baselines tend to cluster around similar conditions.

P SLICE-TO-EXECUTABLE-TEST VALIDATION PIPELINE

Figure 12 shows how a ranked slice recommendation is converted into an executable evaluation bundle. SLICESCORER first ranks missing target slices from the seed inventory and triage statistics (estimated from seed evaluation set). For each target slice, the acquisition stage retrieves candidate media and QA pairs from the datastore in Appendix I, with simulation used as a fallback when retrieval is insufficient. Retrieved and simulated candidates are then normalized into a common evaluation schema and annotated with slice-fidelity indicators that compare their metadata against the

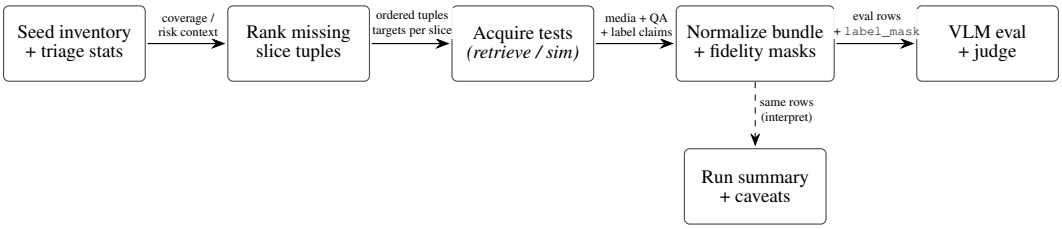


Figure 12: **Slice-to-executable-test flow.** Arrows show what each stage *hands off* to the next; the dashed branch is narrative interpretation of the normalized bundle (qualitative Example D: generic high-risk acquisition; Appendix O).

requested slice attributes. Candidates with missing, ambiguous, or mismatched metadata are retained for auditability but flagged separately, so they are not silently counted as valid evidence for the target slice. The final stage runs the VLM and judge on the normalized bundle and reports both evaluation results and acquisition caveats, including metadata gaps, road-type mismatches, or question-category mismatches.

Exemplar artifacts per stage. An important purpose of our SLICENAV pipeline is to make the transition from slice recommendation to executable testing explicit and auditable. In Example D, the seed inventory and triage statistics define the current coverage and risk context. The ranked-slice stage produces `suggest_top_missing_slices_within_ODD.csv`, whose leading target is a rainy, cloudy, night, city-suburbs, wet-road, intersection, sparse-traffic, hazard-assessment slice. The acquisition stage writes candidate media and QA items to `dataset_retrieval.jsonl`; one retrieved QA item asks, "What actions are currently taking place in the scene?" The normalization stage formats these candidates for VLM evaluation and attaches fidelity annotations indicating whether each candidate matches the requested slice attributes. The VLM-evaluation stage runs the configured model and judge on the normalized bundle. Finally, `agent_response.md` summarizes the recommended slices, evaluation outputs, and caveats about acquisition fidelity.