Enhancing Robot Safety via MLLM-Based Semantic Interpretation of Failure Data

Aryaman Gupta[1]*, Yusuf Umut Ciftci[1,2]*, Somil Bansal[1]
[1]Stanford University, [2]University of Southern California
*Equal Contribution
Correspondence: aryamann@stanford.edu
https://mllm-failure-clustering.github.io/

***Abstract*—As robotic systems become increasingly integrated into real-world environments—ranging from autonomous vehicles to household assistants—they inevitably encounter diverse and unstructured scenarios that lead to failures. While such failures pose safety and reliability challenges, they also provide rich perceptual data for improving future performance. However, manually analyzing large-scale failure datasets is impractical. In this work, we present a method for automatically organizing large-scale robotic failure data into semantically meaningful clusters, enabling scalable learning from failure without human supervision. Our approach leverages the reasoning capabilities of Multimodal Large Language Models (MLLMs), trained on internet-scale data, to infer high-level failure causes from raw perceptual trajectories and discover interpretable structure within uncurated failure logs. These semantic clusters reveal latent patterns and hypothesized causes of failure, enabling scalable learning from experience. We demonstrate that the discovered failure modes can guide targeted data collection for policy refinement, accelerating iterative improvement in agent policies and overall safety. Additionally, we show that these semantic clusters can be employed for online failure detection, offering a lightweight yet powerful safeguard for real-time adaptation. We demonstrate that this framework enhances robot learning and robustness by transforming real-world failures into actionable and interpretable signals for adaptation.**

I. Introduction

Autonomous systems—ranging from self-driving vehicles to household robots—are increasingly deployed in open, dynamic environments. In such unstructured settings, even state-of-the-art robotic systems are prone to failures due to unexpected interactions, unmodeled dynamics, and long-tail edge cases that deviate from training distributions. Traditional validation pipelines, often grounded in simulation or controlled testing, struggle to capture the full complexity of real-world deployment, leaving many failure modes undetected until operation.

A promising direction for improving robustness is to systematically learn from failures that occur during deployment. Robots naturally collect large volumes of perceptual data, including traces of both successful and failed interactions. These failure trajectories can provide valuable insights about the underlying conditions that led to safety violations, brittleness, or policy errors. However, manually curating and analyzing large-scale failure data is time-consuming, brittle, and fundamentally unscalable.

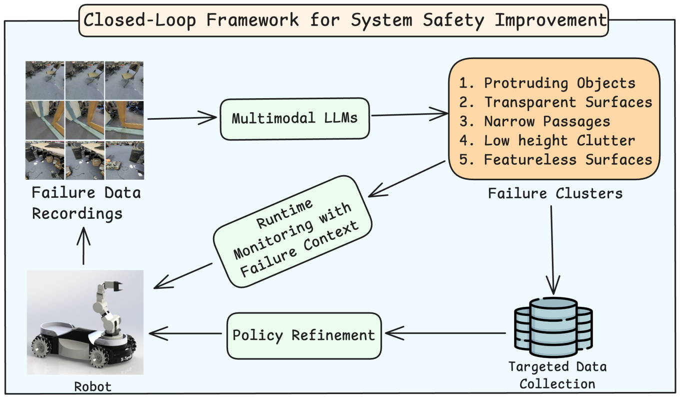


Fig. 1. A closed-loop framework for interpretable failure analysis of autonomous systems. Our method automatically discovers semantically meaningful failure modes from perception recordings without supervision, distilling them into human-understandable clusters annotated with natural language. These clusters support targeted data collection, policy fine-tuning, and real-time semantic failure detection—enabling scalable and continuous safety improvement.

In this work, we propose a framework for automatically organizing failure trajectories into semantically meaningful clusters, enabling scalable, unsupervised learning from failures. Our method leverages the reasoning capabilities of Multimodal Large Language Models (MLLMs)—pretrained on internet-scale vision-language data—to infer high-level failure causes directly from raw observations. By applying prompted reasoning over sequences of perceptual inputs, we identify latent structure in uncurated failure logs, grouping them into interpretable categories described in natural language. Importantly, our framework operates in a completely unsupervised manner, obviating the need for costly human annotation while still isolating nontrivial keywords and cues associated with specific error causes.

The resulting semantic clusters offer multiple downstream benefits. First, they guide targeted data collection, enabling developers to focus training on critical or underrepresented failure modes. Second, these clusters can be integrated directly into online monitoring systems, providing a layer of semantic failure detection that acts as an early-warning mechanism for potential system safety violations during runtime. The semantic failure modes of the system can have many more downstream applications and use cases, e.g., targeted stress testing.

Our framework not only supports large-scale, structured analysis of failure data but also emphasizes interpretability – crucial for deployment in safety-critical domains. By revealing a structured understanding of failure causes in terms that are amenable to human interpretation, we offer actionable insights that can help stakeholders better assess the weaknesses of deployed systems and iterate more effectively.

To summarize, our key contributions are: (a) we propose a novel framework that uses MLLMs to cluster robotic failure data into semantically meaningful groups; (b) our method infers failure causes directly from raw perceptual sequences, eliminating the need for manual annotation or supervision; (c) we demonstrate the effectiveness of our approach on large-scale failure datasets, as well as showcase its potential for supporting downstream applications, including targeted data collection and online failure detection; (d) our framework emphasizes interpretability, generating natural language summaries and keywords for each failure mode to support human-in-the-loop diagnosis.

II. Related Works

a) Semantic Clustering of Images with LLMs.: Recent research has demonstrated the effectiveness of Vision Language Models (VLMs) and Large Language Models (LLMs) for semantic grouping of images. For example, [24] clusters images using human-specified language criteria, while [26] removes the need for manual input by discovering clustering criteria directly from unstructured image collections. Other approaches like [16] highlight key semantic differences between image sets, and [29] and [37] focus on identifying important subpopulations or selecting semantically diverse subsets for training. Our work builds on this line of research by leveraging VLMs for semantic clustering; however, we focus specifically on failure trajectories in autonomous systems rather than static images, introducing a richer temporal and causal structure to the clustering task.

b) Text Clustering and Topic Modeling.: Recent topic modeling methods can be broadly grouped into three categories. First, extensions of classical topic modeling techniques [40, 44, 25, 8] like Latent Dirichlet Allocation (LDA) integrate word embeddings to enhance semantic representation [35, 38, 32, 27]. Second, fully embedding-based approaches [7, 11, 41, 6] that can leverage contextualized representations from pre-trained language models. Third, methods that separate the generation of the clusters from the generation of the topic representation, allowing for a flexible topic model [39, 2, 21]. BERTopic [21], extends this approach by incorporating a class-based variant of Term Frequency-Inverse Document Frequency (TF-IDF). Our work differs in that we use multimodal reasoning to discover semantic failure clusters directly from visual data and language-based explanations.

c) Failure Mining in Autonomous Systems.: Falsification has emerged as a prominent methodology for uncovering failures in autonomous systems. A variety of approaches [23, 18, 12, 19, 14, 45] have been proposed, wherein systems are tested in simulated environments with varied conditions designed to provoke failures. Such controlled testing allows researchers to identify the specific environmental parameters responsible for system breakdowns. However, while these methods effectively reveal failure scenarios tied to controlled variables, they fall short in capturing semantic failure modes of the system that require nuanced interpretation, requiring a human to manually go through the failure scenarios to identify failure modes of the system. In contrast, our approach leverages readily collected failure data to automatically discover semantic failure modes, bypassing the need for manual human inspection.

d) Language-based Failure Reasoning in Robotics.: Recently, several studies have begun to integrate LLMs into robotic systems to generate language-based explanations of failures. For instance, [15] demonstrates the use of LLMs to generate explanations for failures in manipulation tasks, thereby improving system diagnostic capabilities. Similarly, [28] presents an LLM-based framework that not only produces informative failure explanations but also guides a language-based planner in correcting those failures for manipulation tasks. Our work extends these concepts to the domain of failure mode analysis, where the goal is to automatically cluster and interpret semantically rich descriptions from raw failure data, ultimately supporting downstream applications like enhanced monitoring and targeted data collection.

III. Problem Formulation: Finding Failure Clusters from Perception Recordings

We consider the problem of discovering semantic failure clusters from perception data collected during robotic failures. These clusters provide interpretable structure over uncurated failure logs and can be used for downstream tasks such as (1) targeted data collection to improve safety, and (2) online failure detection for real-time reliability.

Formally, we are given a dataset of N sequences, each consisting of K perceptual observations leading up to a failure:

$$D = \{o_{1:K}^n\}_{n=1}^N$$

where $o_{1:K}^n = (o_1^n, o_2^n, \ldots, o_K^n)$ denotes the observation sequence for the n-th failure case. The final observation o_K^n corresponds to the failure event. For example, in autonomous driving, o_K^n might be an image showing a rear-end collision.

Our goal is to construct a system H that maps this dataset to a set of L semantic clusters:

$$H : D \mapsto \{C_l = (s_l, D_l)\}_{l=1}^L ,$$

where each cluster C_l is characterized by a natural language summary s_l and a subset of sequences $D_l \subset D$ that share a common failure mode. These clusters reflect high-level *failure themes* derived from raw observations. For instance, a cluster might be: C_l = `Rear-End Collisions: Insufficient Following Distance`, in which case each sequence in D_l corresponds to a failure where the autonomous car did not maintain a safe following distance from the vehicle in front. This formulation enables unsupervised discovery of interpretable failure modes directly from perceptual logs, providing a scalable mechanism for structuring real-world failure data and guiding robust, adaptive learning.

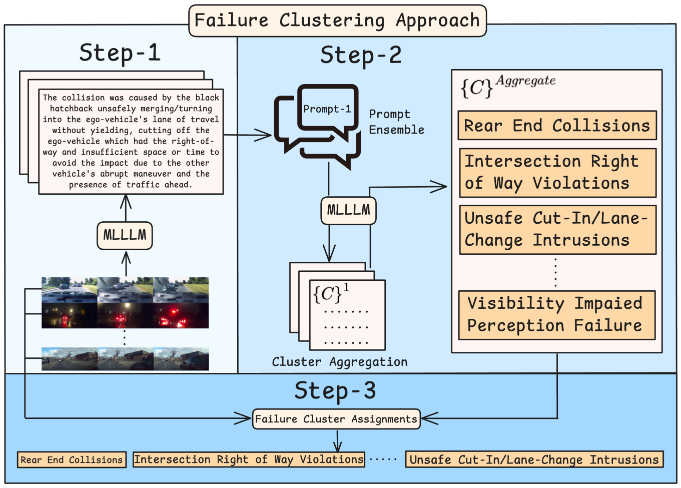

Failure Clustering Approach
Step-1
The collision was caused by the black hatchback unsafely merging/turning into the ego-vehicle's lane of travel without yielding, cutting off the ego-vehicle which had the right-of-way and insufficient space or time to avoid the impact due to the other vehicle's abrupt maneuver and the presence of traffic ahead.
MLLLM
Step-2
Prompt-1
Prompt Ensemble
MLLLM
$\{C\}^1$
Cluster Aggregation
$\{C\}^{Aggregate}$
Rear End Collisions
Intersection Right of Way Violations
Unsafe Cut-In/Lane-Change Intrusions
Visibility Impaied Perception Failure
Step-3
Failure Cluster Assignments
Rear End Collisions
Intersection Right of Way Violations
Unsafe Cut-In/Lane-Change Intrusions

Fig. 2. Overview of our method. (1) *Failure reason inference*: A multimodal large language model (MLLM) analyzes sequences of perceptual observations (e.g., RGB images) to infer high-level failure reasons. (2) *Failure cluster discovery*: An ensemble of prompts is used to group failure reasons into interpretable semantic clusters via a reasoning model, with descriptive keywords and explanations. (3) *Assignment*: Each trajectory is mapped to one of the discovered failure clusters based on its inferred failure reason.

IV. Method

A. Failure Cluster Discovery

Inferring the cause of failure in a robot trajectory is a complex task that requires understanding the robot's environment, the agent's actions, interactions with other agents, and their consequences. Doing so at scale across diverse failure episodes calls for automated systems capable of extracting and reasoning over semantic patterns in raw failure data. Our approach proceeds in three stages: (1) inferring failure reasons from perception sequences using an MLLM, (2) discovering semantic failure clusters via prompted reasoning over inferred causes, and (3) assigning each trajectory to one of the discovered clusters. The resulting clusters support both targeted retraining and online failure detection (Sec. IV-B).

a) Observation Downsampling: To represent the failure event compactly, we downsample the tail of each sequence. Specifically, from each failure trajectory $o^n_{1:K}$, we select the final T frames, sampled from $o_{K-T:K}$ at a reduced frame rate. This balances the need to understand the temporal context of the failure with the limitations of the MLLM's context window.

b) ***Step 1: Inferring Failure Reasons with MLLMs****:* Each downsampled sequence is fed to an MLLM along with a structured prompt. The prompt first asks the model to describe the scene and agent behavior, then to infer a plausible cause of failure. We adopt a Chain-of-Thought (CoT) prompting strategy [43] to improve grounding and interpretability of the inferred reasons.

c) ***Step 2: Discovering Semantic Failure Clusters****:* To uncover structure across failures, we provide all N inferred failure reasons to a reasoning model that is tasked with clustering them into L semantic failure modes. Each cluster C_l is annotated with: a natural language name s_l for the cluster, a short description of the failure type, keywords capturing representative situations in the cluster, and an estimated frequency of occurrence in the dataset. These annotations serve both interpretability and downstream usage in safety-critical settings.

d) Automated Prompt Ensemble and Cluster Aggregation: Since LLM outputs are sensitive to prompt phrasing [46], we generate a prompt ensemble to improve robustness inspired by [34]. Given an initial prompt, an LLM proposes three other prompts using good practices for prompting. We use these ensemble of prompts to infer multiple failure clustering results. The resulting clusterings are merged by an aggregation model that consolidates overlapping clusters and unifies labels and descriptions into a single set of failure clusters C. This approach generates a more comprehensive set of failure clusters by capturing diverse interpretations of the failure reasons and compiling them into one set of clusters.

e) ***Step 3: Assigning Trajectories to Failure Clusters****:* Finally, each inferred failure reason is assigned to one of the L discovered clusters by prompting a reasoning model with the cluster names s_l, keywords, and descriptions generated in the previous step, and asking it to map each failure reason to one of the clusters. Sequences that do not match any cluster are flagged as outliers, which may indicate rare or complex failure modes.

B. Safety Enhancement

The discovered failure clusters support two key mechanisms for enhancing safety in autonomous systems.

a) Online Failure Detection.: Clusters, along with their associated keywords and explanations, form the foundation for semantic failure monitoring in our MLLM-based detection system. Rather than simple pattern matching, the system leverages this structured taxonomy of failure types and contexts to semantically interpret perceptual inputs. When the system recognizes potential failure conditions that align with known failure categories, it can raise warnings or trigger appropriate safeguard policies. This semantic understanding enables more nuanced and accurate failure detection as events unfold. The structured knowledge provided by clusters improves recognition of failures, enhancing the overall safety and reliability of the autonomous system.

b) Targeted Data Collection and Policy Refinement.: Semantic clusters enable *targeted* data augmentation by identifying classes of failures that require additional supervision. This allows practitioners to collect expert demonstrations or counterfactual data in these high-risk regimes and retrain the policy accordingly. Prior work has shown that specification-guided or failure-aware data collection improves robustness in previously unsafe scenarios [10, 22, 13, 36].

V. Experiments

To evaluate our framework for discovering semantic failure clusters, we present two case studies spanning distinct robotic domains. The first involves real-world dashcam footage of traffic incidents, and the second focuses on autonomous indoor navigation in office environments. In both cases, we aim to extract interpretable clusters of failure modes from raw perceptual sequences and demonstrate their utility for downstream tasks such as safety monitoring and targeted data collection.

We use Gemini 2.5 Pro [20] as the MLLM for failure reason inference, and OpenAI o4-mini [33] as the reasoning model for semantic cluster discovery, trajectory assignment, and failure monitoring. Prompt structures and implementation details are provided in the Appendix.

A. Case Study 1: Real-World Car Crash Analysis from Dashcam Videos

We apply our framework to the Nexar driving dataset [30], which includes 1,500 dashcam videos (each approximately 40 seconds, 1280×720 resolution at 30 fps), half of which involve collisions or near-misses involving the ego vehicle. While these recordings are from human-driven vehicles, they serve as a proxy for autonomous driving failures in the absence of large-scale public AV failure datasets. However, our framework is directly applicable to autonomous vehicle logs as well.

1) Failure Cluster Discovery: The box lists the discovered semantic failure clusters $\{C\}$ from Nexar dataset.

(35%) Rear-End Collisions: Insufficient Following Distance
(25%) Intersection Right-of-Way Violations
(18%) Unsafe Cut-In / Lane-Change Intrusions
(8%) Lane Departure & Lateral-Clearance Errors
(7%) Visibility-Impaired Perception Failures
(4%) Pedestrian & Cyclist Detection Failures
(1%) Static-Obstacle & Sudden Intrusion Collisions
(1%) Infrastructure & Clearance Errors
(1%) Other Rare / Long-Tail Cases

Each cluster C_l includes a natural language name s_l, estimated frequency, representative keywords, and a short description. See full output list with keywords and description in Appendix B3 and examples in Fig. 10.

Our system successfully discovers a diverse set of interpretable failure clusters from the driving dataset, as shown in the box above. Qualitative inspection (Fig. 3) shows that these clusters correspond to meaningful and recurring traffic incident types, such as rear-end collisions, unsafe lane changes, or intersection misjudgments. Notably, the discovered clusters closely align with the U.S. DOT Volpe Center's pre-crash typology [31], capturing most major failure types observed in real-world driving. This highlights our method's ability to recover semantically grounded failure categories directly from unstructured video data in a way that aligns with expert-defined taxonomies.

a) Baselines and Results: To evaluate the quality of the failure clusters discovered by our method, we compare against BERTopic [21], a state-of-the-art topic modeling method that combines transformer embeddings with unsupervised clustering and keyword extraction. We apply BERTopic to the same set of failure reasons produced in Step 1 of our pipeline to ensure a direct and fair comparison. We also evaluate a stronger variant, BERTopic-LLM, where a language model summarizes each discovered topic using representative keywords and samples.

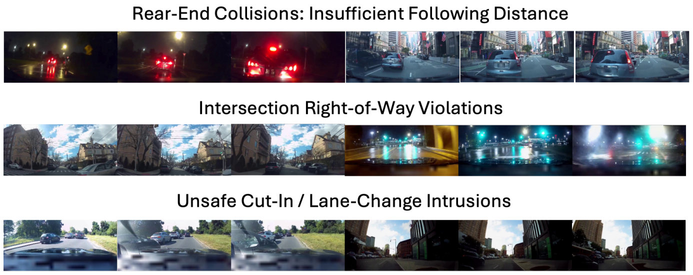

Rear-End Collisions: Insufficient Following Distance
Intersection Right-of-Way Violations
Unsafe Cut-In / Lane-Change Intrusions

Fig. 3. Representative scenes from the top three discovered driving failure clusters.

(83) lead, vehicle, to, failed, in, rearend
(62) suv, the, ego, into, lane, occurred
(46) safe, distance, maintain, following, rearend, andor
(40) to, lead, rearend, in, react, resulting
(39) parked, stationary, its, car, to, failed
(39) another, lane, ego, the, vehicle, collision
(38) ego, the, its, to, failed, vehicle
(32) light, the, green, occurred, because, vehicle
(25) safe, maintain, following, distance, failure, lead
(23) silver, the, lane, ego, occurred, of
(19) pedestrian, yield, who, crosswalk, crossing, car
(19) sun, glare, severe, drivers, the, impaired
(18) light, red, traffic, stop, intersection, with
(17) failure, conditions, nighttime, and, lead, during
(16) white, the, lane, ego, sedan, vehicles
(14) truck, pickup, the, occurred, into, collision

The box shows clusters by BERTopic represented by keywords and item count in parentheses.

(192) Traffic accident causes
(108) Vehicle collision failures
(79) Rear-end collision mechanics
(48) Driving safety violations
(39) Autonomous vehicle failures
(19) Traffic signal violations
(19) Impact of Sun Glare
(13) Autonomous vehicle failures

The box shows clusters by BERTopic-LLM and item count in parentheses.

Standard BERTopic struggles to generate semantically coherent and interpretable clusters. Key failure modes — e.g., `Rear-End Collisions`— are diluted across vague or syntactically noisy topics (e.g., `lead_vehicle_to_failed`, `white_the_lane_ego`). BERTopic-LLM partially improves interpretability, recovering some valid modes such as `Rear-end collision mechanics`, but still includes vague or tautological categories such as `Driving safety violations`, which do not add diagnostic value or clearly

TABLE I
Failure Detection Metrics (%age) for dashcam driving datasets. The left and right half compares the metrics for the IID and OOD tests, respectively. The last column represents the (average) lead detection time of the failure in milliseconds. The best method is highlighted in **bold**.

Method	In-Distribution Trajectories					Out-of-Distribution Trajectories					Time
	TPR	TNR	FPR	FNR	F1	TPR	TNR	FPR	FNR	F1	
Ours	**71.4**	72.5	27.5	**28.6**	**71.4**	**83.0**	70.0	30.0	**17.0**	**77.9**	**610**
NoContext	42.8	85.3	14.7	57.2	54.1	64.0	80.0	20.0	36.0	69.6	473.3
VLM-AD	7.1	91.1	8.9	92.9	12.3	35.0	**94.0**	**6.0**	65.0	49.7	166.6
Leaderboard	52.0	**93.1**	**6.9**	48.0	65.3	18.0	75.0	25.0	82.0	25.2	506.6

differentiate failure modes. In contrast, our method consistently yields sharper, failure-relevant clusters.

2) Failure Monitoring Leveraging Discovered Clusters: We evaluate runtime failure detection by prompting a multimodal reasoning model with recent image history and the learned failure clusters and reasoning about any possible near-future collision (see Appendix A2 for more details).

a) Baselines: We compare our method against *VLM-Based Anomaly Detection (VLM-AD)* methods [17]. These methods provide scene descriptions to an LLM and ask it to detect any possible anomalies in the current observation. We also compare against *Leaderboard Method* – the top-performing failure video classifier on the Kaggle Nexar challenge, based on fine-tuned VideoMAEv2-giant [42].

b) Results: We compare all methods on a set of 200 held-out driving trajectories. As shown in Table I, our method consistently achieves the highest **F1 score** and outperforms baselines in both True Positive Rate (TPR) and False Negative Rate (FNR), showcasing its ability to detect the actual failures both robustly and reliably. These results also indicate that the system failures are not always the same as anomalies or out-of-distribution inputs; in-distribution scenarios, like those specified in our failure clusters, can also lead to system failures that are hard to capture with VLM-AD. We also test all methods on an unseen dashcam dataset of 200 trajectories (see Table I (Middle)). The proposed method demonstrates strong generalization. This suggests our method captures structured semantic patterns beyond dataset-specific cues, unlike the leaderboard classifier, which lacks generalizability.

Finally, we compare the lead failure detection time of different methods (Table I (Right)). Our method detects failures earlier than others, indicating a stronger ability to anticipate failures by correlating the scene observation with the failure clusters.

c) Ablations: We also compare the performance of the proposed failure detection method on removing the failure cluster information from the prompt (NoContext). All other implementation details, such as processing a history of past observations and CoT reasoning, remain the same as our method. This results in significant performance degradation, highlighting the utility of failure clusters information in better detection across environments.

B. Case Study 2: Indoor Robot Navigation

We apply our framework to a vision-based ground robot, navigating unknown indoor office environments [4]. The robot uses a CNN-based policy with a model-based low-level controller. It receives RGB images, ego-velocities, and a goal position, and outputs acceleration commands for the robot. We record robot rollouts in the Stanford office environment [3] and extract front-view image sequences. Colliding trajectories comprise our failure dataset D, which we use for clustering.

1) Failure Cluster Discovery: Our method discovers a set of interpretable failure clusters from the collision trajectories. Notably, clusters such as `walls` and `chairs` were previously identified by manual inspection and reachability analysis in [10]. This validates our method's ability to automatically recover known failure types and uncover new semantic patterns.

(40–45%) Thin–Protruding Objects
(20–25%) Uniform/Featureless Surfaces
(15–20%) Narrow–Gap/Clearance Misjudgment
(9–12%) Low–Height Clutter & Small Floor Obstacles
(8–10%) Box–Like Equipment & Carts
(8–10%) Structural Edges
(5–7%) Bins & Waste Receptacles
(3–5%) Transparent & Reflective Surfaces
(<2%) Overhead & Ceiling Fixtures

The box shows the discovered semantic failure clusters $\{C\}$ for the indoor navigation robot. See the detailed output list in Appendix B3 and examples in Fig. 11.

2) Targeted Data Collection and Policy Fine-Tuning for Enhanced Safety: Following Sec. IV-B, we use the discovered clusters to guide expert data collection in targeted regions of the environment. The robot policy is fine-tuned on this augmented dataset. The failure rate in sampled trajectories drops from 46% to 18%, demonstrating enhanced safety in previously failure-prone situations. This forms a *closed-loop* pipeline of failure discovery, targeted intervention, and policy refinement for continuously enhancing the safety of an autonomous system.

3) Failure Monitoring: We use the same runtime monitoring approach as in Case Study 1. The monitor reasons over the scene, past trajectory, and known failure clusters to preemptively identify any potential failure. As evident in Fig. 4, due to the confined and cluttered nature of the indoor environment, the failure monitor can easily misinterpret static background

TABLE II

Failure Detection Metrics (%age) for Vision-based Indoor Navigation. The left and right half compares the metrics for the IID and OOD tests, respectively. The last column represents the (average) lead detection time of the failure in seconds. The best method is highlighted in **bold**.

	In-Distribution Trajectories					**Out-of-Distribution Trajectories**					
Method	TPR	TNR	FPR	FNR	F1	TPR	TNR	FPR	FNR	F1	Time
Ours	65.0	99.0	1.0	35.0	77.2	67.6	86.2	13.8	32.4	**50.0**	1.21
NoContext	51.7	99.6	0.4	48.3	67.37	45.9	**89.0**	**10.1**	54.1	40.5	0.76
VLM-AD	**83.3**	50.0	50.0	**16.7**	40.0	62.2	60.1	39.9	37.8	27.2	**1.38**
ENet-BC	65.0	**100.0**	**0.0**	35.0	**78.8**	**100.0**	6.3	93.7	**0.0**	22.4	1.01

elements with prior failure contexts, leading to a high false positive rate. To mitigate this, we introduce a simple temporal consistency rule: a failure is flagged only if it persists for three consecutive frames. This helps reduce conservativeness while preserving responsiveness. See Appendix B2 for more details.

a) Baselines: We compare against VLM-AD, where we adapt the prompt from [17] by giving examples relevant to an indoor office environment in the prompt. We additionally compare against *ENet-BC*, a vision-based runtime failure monitor based on EfficientNet-B0, trained on labeled collision data from the same environment [10]. Note that, unlike our method and VLM-AD, ENet-BC requires environment-specific training and does not generalize to unseen layouts.

b) Results: On an in-distribution (IID) test set of 326 trajectories, our method outperforms all LLM-based baselines in F1 score (Table II, Left). ENet-BC achieves similar performance to the proposed method, as expected given its environment-specific training.

To test generalization, we evaluate on an out-of-distribution (OOD) set of 300 trajectories from a different building. The performance of all methods degrades as expected, but the proposed method maintains the highest F1 score, while ENet-BC fails entirely to generalize. This again demonstrates the generalization capabilities of our method.

Our method also detects failures earlier on average, highlighting its ability to reason about impending collisions before impact. Full metrics are in Table II. Although VLM-AD has a slightly higher average time, it is because it fails entirely and detects everything as anomalous, evident from its high FPR.

c) Engaging the Safeguard Policy: We integrate our failure monitor with a reactive safeguard controller that activates upon a failure detection. Fig. 4 shows an example where the nominal policy leads to a collision, while the monitor detects the failure and invokes the safeguard, enabling successful recovery.

VI. Conclusion and Limitations

We present a closed-loop framework for interpretable failure analysis in autonomous systems. Our method automatically discovers semantically meaningful failure modes from perception recordings without supervision, distilling them into human-understandable clusters annotated with natural language. These clusters support targeted data collection, policy fine-tuning, and semantic failure detection—enabling scalable and continuous safety improvement. By leveraging the reasoning capabilities of MLLMs, our approach provides a foundation for understanding, organizing, and responding to failures in complex, unstructured environments.

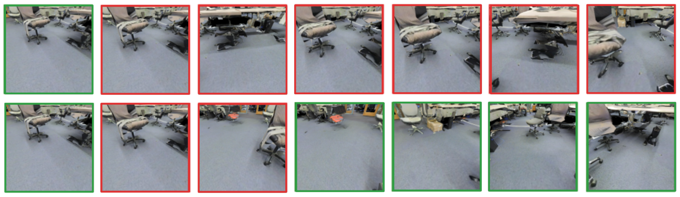


Fig. 4. First-person view images along the robot trajectory while colliding under the nominal policy (top) and safely traversing under the safeguarded policy (bottom). Red and green borders denote failure and safe predictions by the runtime monitor.

While our framework enables interpretable and actionable failure analysis, it has several limitations. First, there is no canonical “correct” way to cluster failure trajectories. Different clustering strategies may highlight different operational insights. Future work could integrate formal techniques like STPA (Systems-Theoretic Process Analysis) or FRAM (Functional Resonance Analysis Method) to complement unsupervised clustering. Second, MLLMs can generate plausible-sounding but incorrect failure explanations, or miss the true cause entirely. Future work could incorporate simulation-based counterfactual testing or causal world models to verify and refine inferred explanations. Finally, while we demonstrate results on datasets of $\sim$700 trajectories, our pipeline can scale to larger datasets and temporally extended events with appropriate batching and model throughput optimization.

VII. Acknowledgements

We gratefully acknowledge research support from DARPA Assured Neuro Symbolic Learning and Reasoning (ANSR) program and by the NSF CAREER program (2240163).

References

[1] URL https://github.com/wzyfromhust/nexar-solution.

[2] Dimo Angelov. Top2vec: Distributed representations of topics. *arXiv preprint arXiv:2008.09470*, 2020.

[3] Iro Armeni, Ozan Sener, Amir Zamir, Helen Jiang, Ioannis K. Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. *2016 IEEE Conference on Computer Vision and Pattern*

Recognition (CVPR), pages 1534–1543, 2016. URL https://api.semanticscholar.org/CorpusID:9649070.

[4] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. Combining optimal control and learning for visual navigation in novel environments. In *Conference on Robot Learning*, pages 420–429. PMLR, 2020.

[5] Wentao Bao, Qi Yu, and Yu Kong. Uncertainty-based traffic accident anticipation with spatio-temporal relational learning. In *ACM Multimedia Conference*, May 2020.

[6] Federico Bianchi, Silvia Terragni, and Dirk Hovy. Pre-training is a hot topic: Contextualized document embeddings improve topic coherence. *arXiv preprint arXiv:2004.03974*, 2020.

[7] Federico Bianchi, Silvia Terragni, Dirk Hovy, Debora Nozza, and Elisabetta Fersini. Cross-lingual contextualized topic models with zero-shot learning. *arXiv preprint arXiv:2004.07737*, 2020.

[8] Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. A novel neural topic model and its supervised extension. In *Proceedings of the AAAI Conference on artificial intelligence*, volume 29, 2015.

[9] Kaustav Chakraborty and Somil Bansal. Discovering closed-loop failures of vision-based controllers via reachability analysis. *IEEE Robotics and Automation Letters*, 8(5):2692–2699, 2023.

[10] Kaustav Chakraborty, Aryaman Gupta, and Somil Bansal. Enhancing safety and robustness of vision-based controllers via reachability analysis. *arXiv preprint arXiv:2410.21736*, 2024.

[11] Adji B Dieng, Francisco JR Ruiz, and David M Blei. Topic modeling in embedding spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453, 2020.

[12] Tommaso Dreossi, Shromona Ghosh, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Systematic testing of convolutional neural networks for autonomous driving. *ArXiv*, abs/1708.03309, 2017. URL https://api.semanticscholar.org/CorpusID:10948572.

[13] Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Kurt Keutzer, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Counterexample-guided data augmentation. *arXiv preprint arXiv:1805.06962*, 2018.

[14] Tommaso Dreossi, Daniel J. Fremont, Shromona Ghosh, Edward J. Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, and Sanjit A. Seshia. Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *International Conference on Computer Aided Verification*, 2019. URL https://api.semanticscholar.org/CorpusID:196614064.

[15] Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *ArXiv*, abs/2410.00371, 2024.

[16] Lisa Dunlap, Yuhui Zhang, Xiaohan Wang, Ruiqi Zhong, Trevor Darrell, Jacob Steinhardt, Joseph Gonzalez, and Serena Yeung-Levy. Describing differences in image sets with natural language. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24199–24208, 2023. URL https://api.semanticscholar.org/CorpusID:265658938.

[17] Amine Elhafsi, Rohan Sinha, Christopher Agia, Edward Schmerling, Issa A. D. Nesnas, and Marco Pavone. Semantic anomaly detection with large language models. *Autonomous Robots*, 47:1035 – 1055, 2023.

[18] Daniel J. Fremont, Edward Kim, Yash Vardhan Pant, Sanjit A. Seshia, Atul Acharya, Xantha Bruso, Paul Wells, Steve Lemke, Qiang Lu, and Shalin Mehta. Formal scenario-based testing of autonomous vehicles: From simulation to the real world. *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2020. URL https://api.semanticscholar.org/CorpusID:212736906.

[19] Shromona Ghosh, Hadi Ravanbakhsh, and Sanjit A. Seshia. Counterexample-guided synthesis of perception models and control. *2021 American Control Conference (ACC)*, pages 3447–3454, 2019. URL https://api.semanticscholar.org/CorpusID:207847751.

[20] Google. Gemini 2.5 language model. https://deepmind.google/technologies/gemini/, 2025. Accessed: May 15, 2025.

[21] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*, 2022.

[22] Aryaman Gupta, Kaustav Chakraborty, and Somil Bansal. Detecting and mitigating system-level anomalies of vision-based controllers. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9953–9959. IEEE, 2024.

[23] Francis Indaheng, Edward Kim, Kesav Viswanadha, Jay Shenoy, Jinkyu Kim, Daniel J. Fremont, and Sanjit A. Seshia. A scenario-based platform for testing autonomous vehicle behavior prediction models in simulation. *ArXiv*, abs/2110.14870, 2021. URL https://api.semanticscholar.org/CorpusID:240070851.

[24] Sehyun Kwon, Jaeseung Park, Minkyu Kim, Jaewoong Cho, Ernest K. Ryu, and Kangwook Lee. Image clustering conditioned on text criteria. *ArXiv*, abs/2310.18297, 2023. URL https://api.semanticscholar.org/CorpusID:264555257.

[25] Hugo Larochelle and Stanislas Lauly. A neural autoregressive topic model. *Advances in Neural Information Processing Systems*, 25, 2012.

[26] Mingxuan Liu, Zhun Zhong, Jun Li, Gianni Franchi, Subhankar Roy, and Elisa Ricci. Organizing unstructured image collections using natural language. *ArXiv*, abs/2410.05217, 2024. URL https://api.semanticscholar.org/CorpusID:273186686.

[27] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Topical word embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29,

2015.

[28] Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. *ArXiv*, abs/2306.15724, 2023.

[29] Yulin Luo, Ruichuan An, Bocheng Zou, Yiming Tang, Jiaming Liu, and Shanghang Zhang. Llm as dataset analyst: Subpopulation structure discovery with large language model. In *European Conference on Computer Vision*, 2024. URL https://api.semanticscholar.org/CorpusID:269605028.

[30] Daniel C. Moura, Shizhan Zhu, and Orly Zvitia. Nexar dashcam collision prediction dataset and challenge, 2025. URL https://arxiv.org/abs/2503.03848.

[31] Wassim G. Najm, John D Smith, and Mikio Yanagisawa. Pre-crash scenario typology for crash avoidance research. 2007. URL https://api.semanticscholar.org/CorpusID:107127275.

[32] Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313, 2015.

[33] Openai. URL https://openai.com/index/introducing-o3-and-o4-mini.

[34] Silviu Pitis, Michael R Zhang, Andrew Wang, and Jimmy Ba. Boosted prompt ensembles for large language models. *arXiv preprint arXiv:2304.05970*, 2023.

[35] Jipeng Qiang, Ping Chen, Tong Wang, and Xindong Wu. Topic modeling over short texts by incorporating word embeddings. In *Advances in Knowledge Discovery and Data Mining: 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part II 21*, pages 363–374. Springer, 2017.

[36] Ameesh Shah, Jonathan DeCastro, John Gideon, Beyazit Yalcinkaya, Guy Rosman, and Sanjit A Seshia. Specification-guided data aggregation for semantically aware imitation learning. *arXiv preprint arXiv:2303.17010*, 2023.

[37] Maying Shen, Nadine Chang, Sifei Liu, and José M. Álvarez. Sse: Multimodal semantic data selection and enrichment for industrial-scale data assimilation. *ArXiv*, abs/2409.13860, 2024. URL https://api.semanticscholar.org/CorpusID:272827561.

[38] Min Shi, Jianxun Liu, Dong Zhou, Mingdong Tang, and Buqing Cao. We-lda: a word embeddings augmented lda model for web services clustering. In *2017 ieee international conference on web services (icws)*, pages 9–16. IEEE, 2017.

[39] Suzanna Sia, Ayush Dalmia, and Sabrina J Mielke. Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too! *arXiv preprint arXiv:2004.14914*, 2020.

[40] Silvia Terragni, Elisabetta Fersini, Bruno Giovanni Galuzzi, Pietro Tropeano, and Antonio Candelieri. Octis: Comparing and optimizing topic models is simple! In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 263–270, 2021.

[41] Laure Thompson and David Mimno. Topic modeling with contextualized word representation clusters. *arXiv preprint arXiv:2010.12626*, 2020.

[42] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. Videomae v2: Scaling video masked autoencoders with dual masking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14549–14560, 2023.

[43] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[44] He Zhao, Dinh Phung, Viet Huynh, Yuan Jin, Lan Du, and Wray Buntine. Topic modelling meets deep neural networks: A survey. *arXiv preprint arXiv:2103.00498*, 2021.

[45] Tong Zhao, Ekim Yurtsever, Joel A. Paulson, and Giorgio Rizzoni. Formal certification methods for automated vehicle safety assessment. *IEEE Transactions on Intelligent Vehicles*, 8:232–249, 2022. URL https://api.semanticscholar.org/CorpusID:247222607.

[46] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR, 2021.

Appendix

The appendix offers additional details, such as prompts, about our implementation. Sections A and B provide the details for dashcam driving datasets and indoor ground navigation, respectively.

A. Case Study 1: Real-World Car Crash Analysis from Dashcam Videos

Section A1 contains all the details of discovering the failure clusters, and Section A2 lists all the details about runtime monitoring for the driving case study. The grey boxes contain prompts.

1) Failure Cluster Discovery:

a) Step 1: Inferring Failure Reasons with MLLMs.:

```
Describe the trajectory of a car from
the sequence of images it observed along
its path, knowing that it undergoes
a collision. After that, provide the
visual semantic reason behind its
failure in brief. Pay attention to the
surrounding objects, other vehicles, and
environmental conditions.

You must provide your answer in the
following format --

trajectory: trajectory_description

failure_reason: semantic_failure_reason

where trajectory_description is the
description of its trajectory and
semantic_failure_reason is the semantic
reason behind failure.
```

b) Step 2: Discovering Semantic Failure Clusters.: We generate an ensemble of prompts from an LLM (OpenAI o4-mini) to infer multiple failure clustering results, which are then aggregated into the final list of clusters. Here are the three prompts generated by the LLM:

Prompt 1:

```
These are semantic failure reasons
for different trajectories of a car.
Your job is to analyze all of them and
come up with clusters of different
semantic failure reasons. Generate
cluster centers based on the types of
visual semantic failures present so that
these reasons can be assigned to those
clusters.

Return the cluster names and the list of
characteristics, keywords which belong
to each cluster. Make sure to include
long tail/rare clusters. Report the
occurrence frequency of each cluster.
```

```
You are a domain expert in automotive
collision analysis. Given a list
of semantic failure reasons for car
trajectories that resulted in crashes,
perform the following steps:

1. Identify and define distinct clusters
of semantic failure types, covering
both common incidents and long{tail/rare
scenarios.

2. For each cluster, provide:

• cluster_name: a concise, descriptive
label

• keywords: a list of characteristic
terms or phrases

• frequency: the count or percentage of
occurrences in the input

• failure modes: a list of specific
failure modes or examples

3. Assign each failure reason to its
corresponding cluster.

4. Output the final result as a
JSON array of objects with keys
‘cluster_name’, ‘keywords’, and
‘frequency’.
```

Prompt 2:

```
You are an expert in automotive semantic
failure classification. Given a list of
trajectory failure reasons that resulted
in car crashes, perform the following:

1. Identify distinct clusters of
semantic failure types, including both
frequent and long{tail/rare cases.

2. For each cluster, define:

• cluster_name: concise label

• keywords: list of representative terms

• count: number of occurrences

• failure modes: specific examples

3. Assign each failure reason to one of
the clusters.

4. Output a single JSON object with two
keys:

• clusters: an array of cluster objects
({‘cluster_name’,‘keywords’,

‘count’})
```

```
• assignments: a mapping from each input
reason to its cluster_name
```

Prompt 3:

```
You are an AI{driven taxonomy engineer
for car collision analysis. Given
semantic failure descriptions of
trajectories that ended in crashes:

• Group descriptions into semantically
coherent clusters (include rare
edge{cases).

• For each cluster, provide:

• Name (short label)

• Key characteristics (list of keywords)

• Example descriptions (up to 3
representative samples)

• Frequency (% of total)

Present your results as a Markdown table
with columns: Cluster Name | Keywords |
Examples | Frequency
```

Each of these prompts is appended with a list of all the failure reasons from the previous step. Finally, we query the LLM again to consolidate these individual lists and generate the final clusters.

c) Step 3: Assigning Trajectories to Failure Clusters.:

```
You are classifying car trajectory
descriptions into predefined clusters
based on failure types.

Assign the trajectory to one or more of
the following clusters:

{list_of_clusters}

Analyze the trajectory description and
identify which type(s) of failures
occurred. Consider the keywords and
notes for each cluster to help with
your classification. Use the "Other"
category only when the trajectory
doesn't reasonably fit into any of the
existing clusters.
```

We compare the distribution of discovered clusters to their actual assignments across trajectories (Fig. 5). Most frequencies align closely. The largest discrepancies appear for the visibility and sudden intrusion clusters. Visibility failures show higher reported than assigned frequency in single-label settings, but match under multi-label assignment—suggesting co-occurrence with other failures. Similarly, *sudden intrusions* is under-assigned due to ambiguity with intersection or cut-in cases.

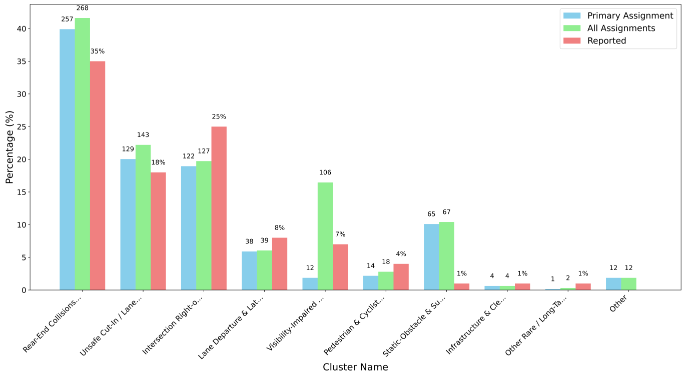


Fig. 5. Reported cluster occurrence frequency during cluster discovery vs the frequency of the assigned clusters with single cluster assignment and multiple cluster assignment.

2) Failure Monitoring Leveraging Discovered Clusters: As described in Sec. IV-B, we directly use the obtained failure clusters to safeguard the system through runtime monitoring. The reasoning model takes a history of past image observations as input and tries to reason about any possible near-future collision while leveraging the knowledge about the most common failures of that system from given clusters in a Chain-of-Thought (CoT) manner. Adding the cluster information in the runtime monitor's prompt helps in zero-shot failure detection for timely executing a safe action, and generalizing across different agents and environments without requiring any specialized training. Here is the prompt:

```
You are a runtime failure monitor for
the ego car navigating outdoors.

Your task is to analyze a sequence of
forward-facing camera images ending
at the car's current position, and
determine whether the car is:

a) SAFE | confidently following a
collision-free trajectory, or

b) UNSAFE | likely to collide with
another vehicle or an object


Operational Context:

- The car must keep on moving without
any physical collisions.

- You are provided with a time-ordered
sequence of image frames at 3 frames
per second, ending at the car's current
position.

- Any collision is considered a failure.

Reasoning Procedure:
```

```
1. Trajectory Inference:
From the image sequence, estimate the
car’s likely near-future motion: Is it
continuing straight, braking, turning,
or maneuvering sharply?
2. Scene Understanding:
Examine the latest image for road
layout, obstacles, other vehicles,
pedestrians, and environmental
conditions.
Pay extra attention to:
- Objects within or near the ego
vehicle’s predicted path
- Occluded or partially visible road
users
- Adverse visibility (glare, fog, night)
affecting perception of relevant objects
3. Collision Risk Assessment:
Determine if a collision is likely.
- Prioritize risks that intersect the
ego path
- Ignore objects that are clearly
outside the trajectory or do not pose
an interaction risk

4. Semantic Failure Matching:
- You are given a list of Known Semantic
Failure Reasons of this car (list below)
that can lead to collisions.
If the current scenario aligns with any
of the known failure reasons, mark it
unsafe and return the exact name of
known failure reason.
- If the failure reason does not match a
listed type but a collision is likely,
briefly describe the new semantic
reason.
- Return SAFE only if you are confident
that the car will safely pass through
the scene without any possible
collision.
- Err on the side of caution, i.e.,
failure, when you are not sure.

Known Semantic Failure Reasons:
{list of clusters}
Output Instructions:
Return only one of the following:
- Name of a known semantic failure
reason (exactly as written above)
- A brief description of a new failure
type
- The word SAFE

Rules:
- Do not provide explanations,
justifications, or degrees of certainty.
- Output must be a single, definitive
label: one listed reason, a new concise
reason, or SAFE.
```

The prompt first outlines the main task for the LLM to act as a runtime failure monitor and detect if the system is currently SAFE or UNSAFE based on the observation history. Further, it provides more information about the system, such as what is meant by a failure here and the frequency of observations. Then, we ask it to sequentially reason about the possible future motion of the system, the surrounding environment, and any possible risk of failure given the most common failure modes of that system through clusters. `list of clusters` contains all the generated clusters along with their keywords and descriptions.

We report the failure detection metrics on both In-Distribution (IID) Trajectories and Out-of-Distribution Trajectories (OOD) in Table I. For IID testing, we took a held-out set of 200 trajectories from the same Nexar dataset, which we used for clustering analysis. For OOD testing, we took another open-source dashcam dataset [5] containing ego car crash videos.

Fig. 6 shows a few examples for True Positives and False Negatives predictions from our method. Rows 1,2 are successfully detected as unsafe. Rows 3,4 are failures, but are detected as safe because the failure event is not evident in those images. Note that the last image in the sequence is taken 1 sec before the collision, and since car driving is a high-speed system, the actual failure sometimes starts even after that, which leads to false safe predictions.

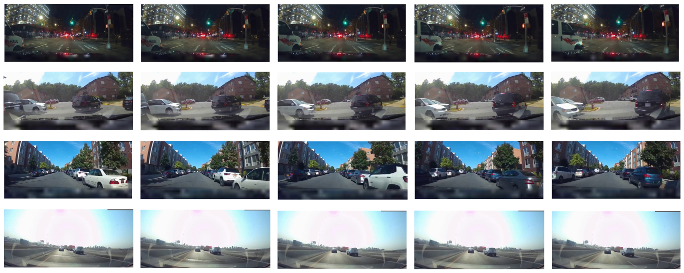


Fig. 6. **(Row-1,2)** True Positives, i.e. failures detected correctly by our method. **(Row-3,4)** False Negatives, i.e., actual failures detected as safe by our method.

a) Baselines.: VLM-AD is an LLM-based few-shot runtime anomaly detection method, where the current image observation is first processed through a scene descriptor, such as an object detector, which further feeds into an LLM. The LLM is asked to find any possible anomalous scenario in the current scene, by providing it with a description of some Normal and Anomalous examples in the prompt. For the driving case study, we directly take the prompt from [17] and use the same reasoning LLM as our method to prevent any model-specific performance difference and perform a fair comparison of the methods. It should also be noted that this method requires manually writing few-shot examples in the prompt to provide contextual information about the system, which in our method is automated with the generated clusters.

For the *Leaderboard-Method* baseline, since the Nexar dataset is also hosted as a Crash Prediction Challenge on Kaggle, we take the best publicly available method [1] as per leaderboard scores to compare with our method. It takes a pre-trained VideoMAEv2-giant model as the backbone and trains a Linear binary classification MLP layer to predict crash. VideoMAEv2-giant [42] is trained on a large amount of video datasets, such as action recognition datasets, sports videos, instructional videos, etc. We tested their trained model on our test dataset for both the IID and OOD cases. Being a classical vision-based classification approach, this method requires environment-specific training and fails to generalize.

B. Case Study 2: Indoor Robot Navigation

Section B1 contains all the details of discovering the failure clusters, and Section B2 lists all the details about runtime monitoring and policy refinement for the indoor robot navigation case study. The grey boxes contain prompts.

1) Failure Cluster Discovery:

a) Step 1: Inferring Failure Reasons with MLLMs.: We randomly sample initial states in the environment and trajectory rollouts with a fixed goal location. For each rollout, we record the front-view image observations and a binary label denoting Failure or Success. Further, we filter out the failure trajectories for our purpose. The following clustering analysis is performed on 228 failure trajectories.

```
Provide a description of the trajectory
of a robot from the sequence of images
it observed along its path, knowing
that it collides in the last image.
After that, provide the visual semantic
reasons behind its failure in brief. Pay
attention to the surrounding objects.

You must provide your answer in the
following format --

trajectory: trajectory_description

failure_reason: semantic_failure_reason

where trajectory_description is the
description of its trajectory and
semantic_failure_reason is the semantic
reason behind failure.
```

b) Step 2: Discovering Semantic Failure Clusters.: Here is the ensemble of prompts given by the LLM. Each of these prompts is appended with a list of all the failure reasons from the previous step.

Prompt 1:

```
These are semantic failure reasons of a
robot navigating indoors based on images
that fails due to collision. Generate
cluster centers based on the types of
visual semantic failures present so that
these reasons can be assigned to those
clusters. Return the cluster names and
the list of characteristics, keywords
which belong to each cluster. Make sure
to include long tail/rare clusters.
Report the occurrence frequency of each
cluster.

You are an expert in robotic vision
failure analysis. Below is a list of
semantic failure reasons for an indoor
robot navigation system that leads to
collisions.

Your tasks:

1. Identify distinct cluster centers
representing each type of visual
semantic failure.

2. Assign each failure reason to the
appropriate cluster.

3. Include long{tail/rare clusters as
separate entries.

4. For each cluster, report:

• cluster_name

• defining keywords or traits

• occurrence_frequency

• example descriptions (up to 3)

5. Present the output as a JSON array
of objects with fields ‘cluster_name’,
‘keywords’, and ‘frequency’.
```

Prompt 2:

```
Act as a taxonomy engineer analyzing
semantic failure reasons of an indoor
vision{based robot that collides. Given
the following descriptions, perform
these steps:

• Group reasons into clusters based on
shared semantic features.
```

```
• Capture both common patterns and
rare/long{tail failure types.

• For each cluster, provide:

{ name (a concise label)

{ terms (list of characteristic
keywords)

{ count (number of examples in that
cluster)

{ failure modes

• Output the final result as valid JSON:
an array of objects with keys ‘name’,
‘terms’, and ‘count’.
```

Prompt 3:

```
You are a domain expert in robotic
vision failure analysis. Given a list
of semantic failure reasons for an
indoor navigation robot that lead to
collisions, perform the following steps:

1. Identify and define distinct clusters
of semantic failure types, including
both common and long{tail/rare cases.

2. For each cluster, provide:

• cluster_name: a concise, descriptive
label

• keywords: list of characteristic terms
or phrases

• frequency: count or percentage of
occurrences in the input

• failure modes: list of specific
failure modes or examples

3. Assign each failure description to
its appropriate cluster.

4. Output the result as a JSON array
of objects with fields ‘cluster_name’,
‘keywords’, and ‘frequency’.
```

c) Step 3: Assigning Trajectories to Failure Clusters.:

```
You are classifying robot trajectory
descriptions into predefined clusters
based on failure types.

Assign the trajectory to one or more of
the following clusters:

    {list_of_clusters}

Analyze the trajectory description and
identify which type(s) of failures
occurred. Consider the keywords and
notes for each cluster to help with
your classification. Use the "Other"
category only when the trajectory
doesn’t reasonably fit into any of the
existing clusters.
```

We compare the distribution of discovered clusters to their actual assignments across trajectories (Fig. 5). Most frequencies align closely. The largest discrepancies appear for the `Narrow-Gap/Clearance Misjudgment` and `Low-Height Clutter  Small Floor Obstacles` clusters.

They show higher reported than assigned frequency in single-label settings, but match under multi-label assignment– suggesting co-occurrence with other failures. Similarly, `Bins  Waste Receptacles` is under-assigned due to ambiguity with `Low{Height Clutter  Small Floor Obstacles` or Box–Like Equipment Carts.

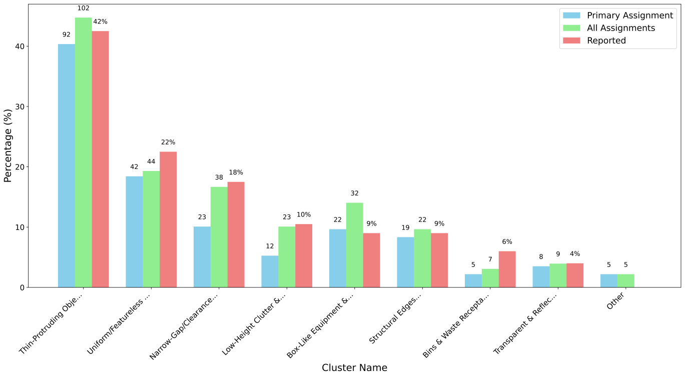


Fig. 7. Reported cluster occurrence frequency during cluster discovery vs the frequency of the assigned clusters with single cluster assignment and multiple cluster assignment.

2) Failure Monitoring Leveraging Discovered Clusters: We perform runtime failure detection in the same way as described in A2 for the indoor navigation robot. The prompt is tailored to an indoor robot with its list of clusters. Here is the detailed prompt:

```
You are a runtime failure monitor for a
vision-based autonomous robot navigating
in an indoor environment.
Your task is to analyze a sequence of
recent image observations, ending at the
```

```
robot’s current position, and determine
whether the robot is:
- SAFE | confidently continuing in free
space on a collision-free path, or
- UNSAFE | no free space ahead and
at significant and credible risk of
collision, based on observable evidence.


Operational Context:
- The robot must reach a predefined goal
without any physical collisions.

- You are provided with a time-ordered
sequence of image frames, each after 0.5
seconds, ending at the robot’s current
location.
- The robot is moving with a maximum
speed of 0.6 m/s.


Evaluation Procedure:
1. Predict Short-Term Trajectory
- Based on the image sequence, estimate
the robot’s likely immediate direction
of movement (e.g., straight, turning,
drifting).
- Incorporate temporal cues for better
motion understanding.
2. Identify Relevant Obstacles
- Inspect the final image for physical
objects that may intersect the predicted
path.
- Focus only on nearby, collision-range
elements that could plausibly interfere
with the robot’s trajectory.
3. Determine Collision Risk
- Mark the situation as unsafe if there
is a visual alignment between the
projected path and an obstacle, else
mark it SAFE.
4. Classify the Risk
- You are given the most common failure
modes of this robot in the list below.
- If the risk matches one of the Known
Semantic Failure Reasons listed below,
return name of that exact label.
- If a new type of visible risk is
present, briefly describe it in concise
terms.
- If no substantial risk is visible
along the projected path, mark it as
SAFE.


Known Semantic Failure Reasons:
{list_of_clusters}


Output Instructions:
Return only one of the following:
- Name of a known semantic failure
reason (exactly as written above)
- A brief description of a new failure
type
- The word SAFE


Rules:
- Do not provide explanations,
justifications, or degrees of certainty.
- Output must be a single, definitive
label: one listed reason, a new concise
reason, or SAFE.
```

We report the failure detection metrics on both IID and OOD test sets II. The IID test set was taken from the same environment, but the OOD test set was taken from a different building with very different semantics, but an indoor office environment in the dataset.

Fig. 8 shows a few examples for True Positives and False Negatives predictions from our method. Rows 1,2 are successfully detected as unsafe. Rows 3,4 are failures, but are detected as safe because the monitor is getting confused seeing some free space ahead, and the temporal consistency rule gives false safe outputs.

a) Baselines.: For the *VLM-AD* baseline, we adapt the prompt from [17] for an indoor robot and test it in the same way as described in A2. Here is the prompt:

```
I am the fault monitor for a vision
based autonomous robot. My job is to
analyze the robot’s observations and
identify anything that could cause the
robot to take actions that are unsafe,
unpredictable or lead to collision. For
each object that the robot observes,
I will reason about whether the object
constitutes a normal observation or
an anomaly. Normal observations do
not detrimentally affect the robot’s
performance, whereas anomalies might.
```

Finally, I will classify whether the overall scene is normal or abnormal.

For example, "The robot is navigating indoors and observes:

- free space ahead in its path
- a chair or desk right in front of it in its path

Free space ahead:

1. Is this common to see while navigating? Yes, robot can often have free space in the room for navigation.
2. Can this influence the robot’s behavior? No, it is good to have free space.
3. Can the robot navigate safely here? Yes, robot should be able to navigate safely there.
4. Can this cause the robot to make unpredictable or unsafe maneuvers? No, it is safe path.

Classification: Normal.

Chairs or desks right in front of the robot in its path:

1. Is this common to see while navigating? Yes, it is common to see chairs or desks in the room. However, being very close to them can cause collision.
2. Can this influence the robot’s behavior? Yes, the robot might collide with them if it is too close.
3. Can the robot navigate safely in its presence? No, if they are in the path, but if they are in the surrounding and doesn’t block the path, then yes.
4. Can this cause the robot to make unpredictable or unsafe maneuvers? Yes, this could cause the robot into colliding with them if it is too close.

Classification: Anomaly.

Get the robot’s current observation and reason about the environmental context. Finally, try to reason if there is anything anomalous or not.

If yes, return the word UNSAFE, else return the word SAFE. Only return the word UNSAFE or SAFE and nothing else.

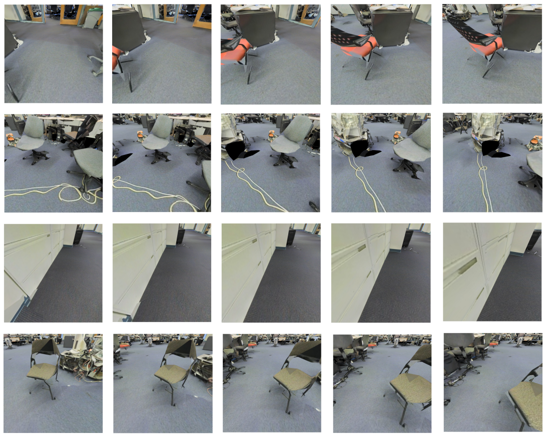


Fig. 8. **(Row-1,2)** True Positives, i.e. failures detected correctly by our method. **(Row-3,4)** False Negatives, i.e., actual failures detected as safe by our method.

For the *ENet-BC* baseline, we follow [9] and first compute the Backward Reach-Avoid Tube (BRAT) for our environment. Further, we collect a dataset of 3000 images with 1500 safe and 1500 unsafe images labeled using the BRAT and follow [22] to train the EfficientNet-B0 model for binary classification. Similar to the *Leaderboard* method in the driving example, it is based on environment-specific training, and there’s no opportunity for generalization across environments, where LLM-based methods shine.

3) Targeted Data Collection and Policy Fine-Tuning for Enhanced Safety: For targeted data collection around the mined failure modes of the robot, we identify regions in the environment corresponding to each cluster. In each failure region, for some starting states and a goal state, we use a Model Predictive Control (MPC) scheme to find a sequence of dynamically feasible waypoints [4]. The collected dataset around all the failure modes comprised approximately 40K training samples, which were then augmented with the original training data. Lastly, we fine-tuned the trained checkpoint for only 20 iterations on the augmented dataset and compared the performance on the previous policy and the retrained policy by rolling out trajectories for 50 initial states in the same environment.

Fig. 9 shows a few examples from the targeted dataset.

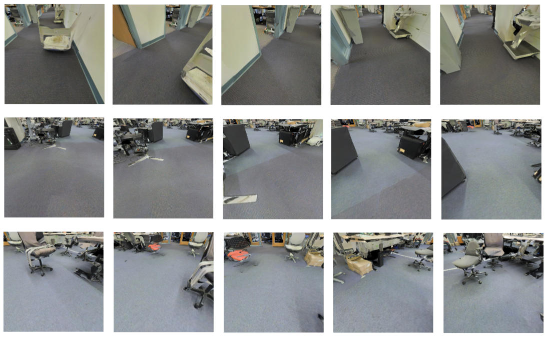


Fig. 9. Expert data collected around identified clusters.

Generated clusters with keywords and their descriptions for driving dashcam datasets.

1. Name: Rear-End Collisions: Insufficient Following Distance
Keywords: rear-end, following distance, tailgating, braking, delayed reaction, deceleration ahead
Description: The vehicle ahead brakes or stops and the ego car fails to leave adequate gap or react in time, leading to a rear-end impact|often exacerbated by poor visibility or road conditions.

2. Name: Unsafe Cut-In / Lane-Change Intrusions
Keywords: lane change, cut-in, merge, encroachment, insufficient clearance, drift, sideswipe
Description: Another vehicle abruptly merges or drifts into the ego car’s lane without providing room, cutting off the ego car and causing a collision.

3. Name: Intersection Right-of-Way Violations
Keywords: left turn, yield, right-of-way, red light, signal violation, cross-traffic, stop sign
Description: A vehicle fails to yield when turning (especially unprotected left turns) or runs a red/yellow light or stop sign, colliding with lawful through traffic.

4. Name: Lane Departure Lateral-Clearance Errors
Keywords: lane departure, drift, misjudged gap, narrow lane, lateral clearance, sideswipe
Description: The ego car drifts out of its lane or misjudges space beside parked/static vehicles, grazing or sideswiping them.

5. Name: Visibility-Impaired Perception Failures
Keywords: glare, night, rain, fog, wet road, low visibility, windshield occlusion
Description: Sun glare, darkness, heavy rain or other occlusions hide a hazard|vehicle, pedestrian or signal|delaying detection and proper reaction.

6. Name: Pedestrian Cyclist Detection Failures
Keywords: pedestrian, cyclist, bicycle, crosswalk, jaywalking
Description: The car fails to detect or yield to a vulnerable road user entering or crossing the roadway, often in a crosswalk or turning path.

7. Name: Static-Obstacle Sudden Intrusion Collisions
Keywords: parked vehicle, door opening, driveway pull-out, backing/reversing, sudden obstacle
Description: A parked car door opens or a vehicle emerges from a driveway/parking stall unexpectedly into the ego car’s path.

8. Name: Infrastructure Clearance Errors
Keywords: pillar, attenuator, underpass, overpass, vertical clearance, sign structure
Description: The ego car clips or crashes into fixed road or bridge infrastructure, or a vehicle’s height exceeds overhead clearance.

9. Name: Other Rare / Long-Tail Cases
Keywords: wrong-way, camera tilt/misalignment, construction equipment, forklift, oversized load, extreme/edge-case
Description: Extremely infrequent scenarios such as head-on wrong-way collisions, sensor/camera failures, or rare large obstacles.

Generated clusters with keywords and their descriptions for indoor robot navigation.

```
1. Name: Thin-Protruding Objects
Keywords: folding chair, foldable chair, thin metal legs, chair frame, chair legs,
chair base, chair seat, office chair, casters, wheels, central post, desk leg,
table leg, desk support
Description: Robot fails to detect or underestimates thin, low-contrast legs and
bases, colliding with folding or office chairs, table/desk legs, etc.

2. Name: Uniform/Featureless Surfaces
Keywords: white cabinet, filing cabinet, locker/lockers, cabinet, uniform surface,
featureless wall, light-colored surface, wall base, panel
Description: Robot treats large flat, light-colored walls or cabinets as free space
due to poor depth cues and lack of texture or edges.

3. Name: Narrow-Gap/Clearance Misjudgment
Keywords: narrow space, tight passage, narrow passage, misjudged gap, insufficient
clearance, turning radius, misjudged space
Description: Robot squeezes through tight passages|between obstacles or
underestimates turning radius|leading to collisions in narrow spaces.

4. Name: Low-Height Clutter  Small Floor Obstacles
Keywords: backpack, cables, wires, power brick, small debris, equipment debris,
low-lying object, floor clutter, soft object
Description: Robot runs into low-profile items on the floor (backpacks, cables,
small debris) that blend into the background.

5. Name: Box-Like Equipment  Carts
Keywords: computer tower, server, server cabinet, grey box, box-like object,
equipment, pedestal, cart, machinery
Description: Robot fails to detect or underestimates clearance around bulky,
rectangular objects such as computer towers, servers, carts or pedestal-type
furniture.

6. Name: Structural Edges: Door Frames, Jambs  Wall Corners
Keywords: door frame, door jamb, wooden frame/panel, frame edge, threshold, open
door edge, corner, wall corner, external corner, protruding corner
Description: Robot collides with rigid vertical edges like door frames, jambs,
wooden panels or cuts corners too sharply on wall intersections.

7. Name: Bins  Waste Receptacles
Keywords: trash bin, recycling bin, bin lid, blue-lidded bin, green bin, waste
receptacle
Description: Robot collides with trash bins, recycling bins or their lids in tight
spaces.

8. Name: Transparent  Reflective Surfaces
Keywords: glass door, glass panel, mirror, reflective surface, transparent panel,
reflection, deceptive surface
Description: Robot mistakes glass doors, panels or mirrors for free space or is
deceived by reflections.

9. Name: Overhead  Ceiling Fixtures
Keywords: ceiling, low ceiling, fixture, overhead, overhead obstacle, piping,
ceiling fixture
Description: Robot drives into low-hanging fixtures or ceiling obstructions due to
upward blind spots in its sensors.
```

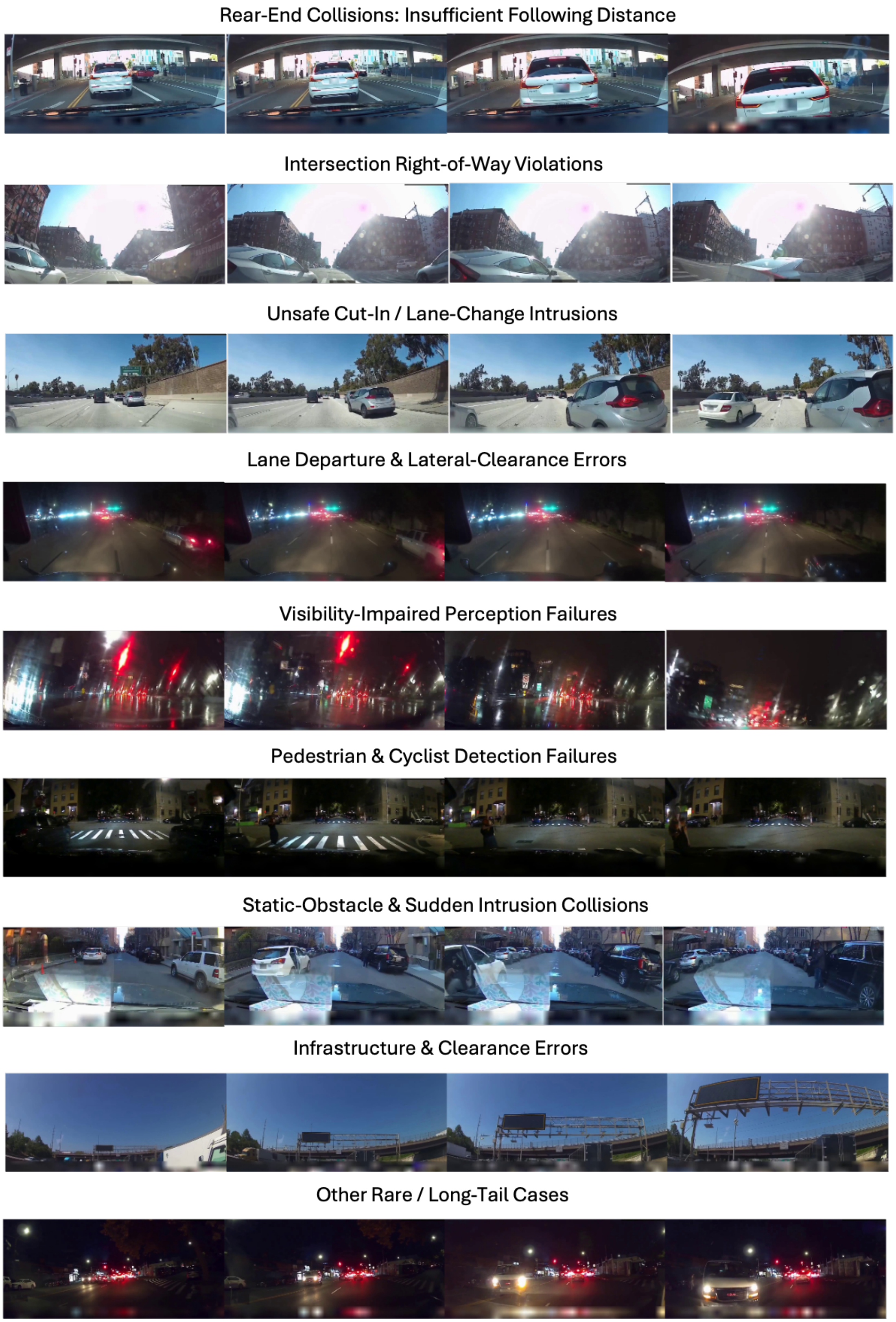
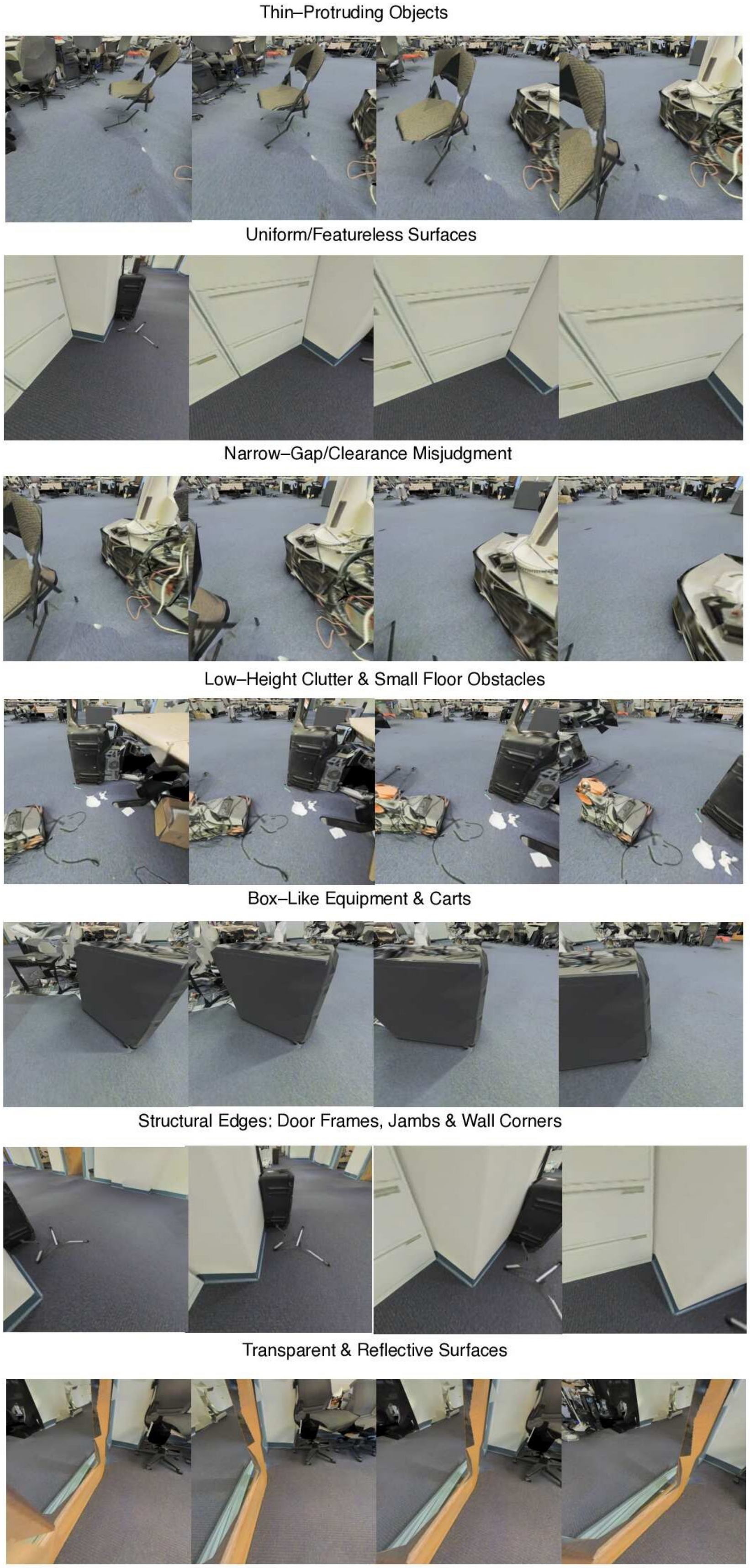

Rear-End Collisions: Insufficient Following Distance
Intersection Right-of-Way Violations
Unsafe Cut-In / Lane-Change Intrusions
Lane Departure & Lateral-Clearance Errors
Visibility-Impaired Perception Failures
Pedestrian & Cyclist Detection Failures
Static-Obstacle & Sudden Intrusion Collisions
Infrastructure & Clearance Errors
Other Rare / Long-Tail Cases

Fig. 10. Example recording for each cluster in the driving dashcam dataset.



Thin–Protruding Objects

Uniform/Featureless Surfaces

Narrow–Gap/Clearance Misjudgment

Low–Height Clutter & Small Floor Obstacles

Box–Like Equipment & Carts

Structural Edges: Door Frames, Jambs & Wall Corners

Transparent & Reflective Surfaces

Fig. 11. Example recording for each cluster in indoor robot navigation.