Real-Time Out-of-Distribution Failure Prevention via Multi-Modal Reasoning

Milan Ganai[1], **Rohan Sinha**[1,†], **Christopher Agia**[1,†], **Daniel Morton**[1], **Marco Pavone**[1,2]
[1]Stanford University, [2]NVIDIA Research
{mganai,rhnsinha,cagia,dmorton,pavone}@stanford.edu

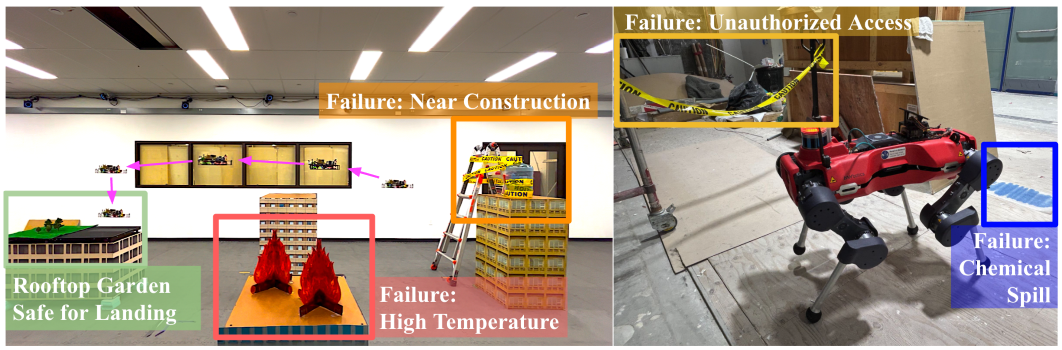


Figure 1: We introduce `FORTRESS`, a framework for robots in open-world environments to avoid out-of-distribution (OOD) failures. When nominal plans become unreliable, our approach uses multi-modal reasoning to rapidly generate semantically safe fallback paths that prevent OOD failures. For instance, `FORTRESS` enables our quadrotor drone (on left) to identify safe building roofs for landing, avoiding hazards like burning buildings, and allows the ANYmal robot (on right) to intuit semantically unsafe areas like chemical spills.

Abstract: Foundation models can provide robust high-level reasoning on appropriate safety interventions in hazardous scenarios beyond a robot's training data, i.e. out-of-distribution (OOD) failures. However, due to the high inference latency of Large Vision and Language Models, current methods rely on manually defined intervention policies to enact fallbacks, thereby lacking the ability to plan generalizable, semantically safe motions. To overcome these challenges we present `FORTRESS`, a framework that generates and reasons about semantically safe fallback strategies in real time to prevent OOD failures. At a low frequency in nominal operations, `FORTRESS` uses multi-modal reasoners to identify goals and anticipate failure modes. When a runtime monitor triggers a fallback response, `FORTRESS` rapidly synthesizes plans to fallback goals while inferring and avoiding semantically unsafe regions in real time. By bridging open-world, multi-modal reasoning with dynamics-aware planning, we eliminate the need for hard-coded fallbacks and human safety interventions. `FORTRESS` outperforms on-the-fly prompting of slow reasoning models in safety classification accuracy on synthetic benchmarks and real-world ANYmal robot data, and further improves system safety and planning success in simulation and on quadrotor hardware for urban navigation. Website can be found at https://milanganai.github.io/fortress/.

Keywords: Multi-modal Reasoning, Semantic Safety, OOD Reliability

1 Introduction

Across various industries, autonomous robots [1, 2, 3, 4, 5, 6, 7] are expanding their deployment regions from structured, localized settings to unstructured, open-world environments [8]. In the process, they increasingly encounter out-of-distribution (OOD) failure scenarios—situations that differ significantly from the system's design assumptions and training data, leading to degraded performance, unsafe behavior, or even complete system malfunction [9, 10]. To ensure the reliability

†Equal contribution.

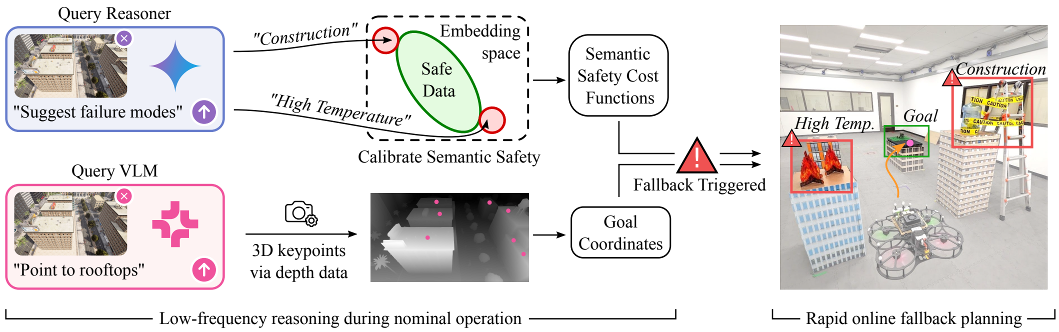


Figure 2: **Overview of FORTRESS:** Our proposed framework prevents OOD failures by reasoning about concrete goal locations based on semantic descriptions of fallback strategies, anticipating failure modes, and constructing cost functions that identify semantically unsafe regions at a low frequency during the nominal trajectory. When a safety response is triggered, the algorithm rapidly produces semantically safe fallback plans.

and safety of autonomous robots in expanding operations, it is crucial to: adaptively identify goals for fallback strategies, accurately discern unsafe regions where entering could lead to OOD failures, and quickly generate executable fallback plans that are semantically safe.

Pre-trained foundation models, like Large Language Models (LLMs) and Vision-Language Models (VLMs) [11], can understand OOD scenarios [12, 10] and reason in a zero-shot manner about high-level semantically appropriate responses [13]. However, these high-level descriptions of safety interventions can be misleading: while a model might suggest a drone to "land in a parking lot" to avoid a large firetruck, this could lead to semantically unsafe plans that crash into people or land on a building on fire. These models are inherently dynamics-agnostic, and attempts to integrate them [14] into end-to-end control pipelines can compromise robustness [15, 16]. Moreover, the models' high inference latency [17] renders them unsuitable for real-time querying for granular planning in safety-critical moments. Approaches resort to rigid, hard-coded fallback regions [13] and human interventions for safety [18], which are impractical in open-world environments where potential failure scenarios are vast. The challenge is to use foundation model reasoning for the real-time generation of fallback plans that are semantically safe and dynamics-feasible.

We demonstrate how to leverage foundation models to identify fallback goals, anticipate failures, and infer semantically dangerous regions. By preparing slow reasoning in advance, we expedite the generation of semantically safe fallback plans. Particularly, we introduce FORTRESS, a framework for OOD Failure Prevention in Real Time by Generating and Reasoning about Fallback Strategies (overview in Fig. 2). FORTRESS is a multitiered, slow-fast hierarchical approach: At a low frequency or offline, the algorithm (i) uses VLMs to translate abstract semantic strategies into physical fallback goal suggestions and (ii) constructs cost functions to quickly identify semantically unsafe state spaces, entering which can result in OOD failures, by calibrating embedding models with failure modes from foundation model reasoners. Once a runtime monitor triggers a response, FORTRESS (iii) enables existing motion planners to rapidly infuse semantic safety into fallback trajectories.

2 Related Works

Safe control for autonomous systems has been studied through various formal methods [19, 20, 21]. Hamilton–Jacobi (HJ) reachability can provide worst-case reach-avoid guarantees [22, 23, 24], and Control Barrier Functions (CBFs) certify control invariance [25]. They can be used as online safety filters to monitor the system behavior and invoke corrective controls when violations are imminent [26, 27, 28]. However, these tools require precise models of the environment and explicitly defined failure regions which are not always readily available in open-world settings.

Foundation models [11, 14], including LLMs and VLMs, have enabled semantic planning in robotics [29, 30], with applications in quadrotors [31, 32], manipulation [33, 34], and vehicles [35]. However, the models' high latency hinders real-world deployment [17]. One approach [18] mitigates this by feeding language constraints to a fast VLM object detector [36] and updating reachability–based online safety filters [37, 38], but the constraints are obtained from human intervention.

Real-time recovery planning helps ensure safety in autonomous systems once a response is triggered. Current methods include game-theoretic autonomous vehicle trajectory repair [39], LLM-based semantic trajectory fixes [40], satisfiable modulo theory and reachability analysis for traffic-rule-compliant repairs [41], and perception-failure detection with a trained safety monitor that triggers learned recovery plans [42]. However, these approaches only offer localized trajectory repairs and lack dynamic real-time generation of new fallback goals and paths, a crucial capability when the nominal path and goal suddenly become unsafe in open-world environments.

Furthermore, real-world systems face OOD scenarios that significantly deviate from training distributions. Some methods to improve system robustness include training for distributionally robust optimization [43, 44] and model adaptation [45, 46]. Recent research works investigate OOD detection [47, 48, 49, 50] paired with fallback controllers [51, 52, 53]. Foundation models employed in a zero-/few-shot manner [54] can understand semantic anomalies [12] and enable closed-loop failure detection [12, 55] along with fallback set selection [13]. However, these rely on handcrafted policies or static fallback sets, which are inadequate in preventing OOD failures.

Despite significant advances in robotics planning and control, existing methods fall short in dynamic, open-world conditions. By relying on precise environment models, human interventions, and rigid fallback plans and/or goals, they leave a critical gap in unstructured settings when the nominal path's deployment becomes unsafe. We need systems that adaptively generate real-time fallback strategies, ensuring recoveries are semantically safe in uncertain real-world environments.

3 Problem Formulation

We investigate robots with known discrete-time system dynamics given by $x_{t+1} = f(x_t, u_t)$, for state $x \in \mathcal{X} \subset \mathbb{R}^n$ and control $u \in \mathcal{U} \subset \mathbb{R}^m$. While we can control the robot's motion in its physical space, we aim to imbue understanding of semantic safety in the responses to OOD failures not captured in the robot's state – like avoiding a roof on fire. To build semantic awareness, we assume access to safe semantic descriptions $\Omega_s := \{\omega_i\}_{i=1}^N$—for example, state descriptions with "buildings in city" for drones or "trees on median strip" for autonomous vehicles. This is practical since systems have vast descriptions of logs from successful deployments and training data. We also assume we have a set of semantic descriptions of high-level fallback strategies $\Sigma := \{\sigma_1, \sigma_2, \dots\}$, like "empty flat roofs" for drone landing or "vacant road shoulders" for vehicle parking. While engineers or language models can easily generate these general strategy descriptions from prior data, we aim to realize the physical execution of such strategies. The robot has function $\texttt{nearby}(x, l)$ that returns a state description of all concepts within distance l of coordinate x, which onboard RGBD cameras and pre-trained object detectors (like OWL-ViT [56], OWLv2 [36], and YOLOv8 [57]) can rapidly provide. We also have cost functions θ_c to capture collision hazards, where $\theta_c(x) > 0$ if and only if x is within distance l_c of a physical obstacle, which can be obtained from depth sensors. Following [13, 27, 28], we assume that the robot is equipped with a runtime monitor, which is a system that raises an alarm when it detects anomalies or potential hazards in the robot's environment. The monitor's alarm prompts us to reevaluate the robot's plan and identify an appropriate response.

We address the challenge of generating semantically safe fallback plan responses. In open-world environments, novel semantically unsafe concepts, with descriptions referred to as Ω_d, can appear anywhere. For example, while "person" and "ladder" are individually safe, "person on a ladder" poses a "Worker Injury" hazard for the ANYmal robot, as illustrated in Fig. 4. These unsafe concepts may be related or even unrelated to what triggered the fallback. Robots generally have limited access to data on experiencing and responding to failure. So if a robot encounters state descriptions in Ω_d, these would be considered OOD failures relative to our training data, Ω_s.

Objective: We aim to design an algorithm to identify, plan, and execute semantically safe fallback behaviors when anomalous conditions render the robot's original task unsafe. The robot must:

M1: Be prepared with relevant concrete fallback strategy goal locations from semantic strategy idea set Σ especially when the original nominal path is infeasible,

M2: Bootstrap its understanding of semantically unsafe states x where $\texttt{nearby}(x, l) \in \Omega_d$, which would result in OOD failures, using our plentiful safe state descriptions Ω_s,

M3: Rapidly generate safe plans that implement a fallback strategy while satisfying the semantic safety constraints and physical constraints.

4 Proposed Approach

We describe our framework FORTRESS, which generates and reasons about semantically safe fallback strategies to prevent OOD failures (Fig. 2). Our algorithm has three key components: (i) constructing physical fallback goal locations from abstract semantic strategy ideas (Section 4.1), (ii) rapidly inferring semantically unsafe regions (Section 4.2), and (iii) employing foundation model reasoning with worst-case analysis to rapidly generate semantically safe fallback plans (Section 4.3).

4.1 Generating potential fallback strategy sets (M1)

Figure 3: For a drone agent in the CARLA simulator, FORTRESS identifies goal locations for semantic fallback strategy description "empty, horizontal building roofs" using VLMs like Molmo. VLMs produce safe goals, but sometimes they also generate unsafe ones (e.g. near people, cars, etc) and are not dynamics-aware (e.g. tight landing spots require complex maneuvering).

A crucial aspect in executing fallback strategies is generating clear goals for contingency plans when the original plan's goal is unattainable. We leverage the abstract semantic fallback characteristics Σ defined in Section 3. These fallback goals serve various purposes, such as waiting for external dangers to subside, recalibrating perception systems, enabling controlled hardware/software maintenance, providing extra time to reassess the environment, creating more robust mitigation strategies, or receiving human intervention. To transform semantic descriptions into physical fallback goal locations, we utilize the general-purpose reasoning of VLMs. In particular, we employ the VLM Molmo trained on the PixMo dataset [58] to analyze a robot's scene image with a query based on a strategy description $\sigma_i \in \Sigma$. The VLM outputs pixel coordinates for potential fallback points (shown in Fig. 3), which are then converted into 3D global coordinates using depth information and camera intrinsics, to form a set of goal locations $\mathcal{G}_i$. VLM queries can incur several seconds of latency, depending on model size and token limits. Therefore we propose mitigating delays during critical moments by preemptively querying the VLM at a low frequency during normal operations (see Table 2 for times) to identify potential fallback locations, caching their corresponding 3D global coordinates, and loading these locations immediately when a fallback response is needed.

4.2 Reasoning about Semantic Safety Constraints (M2)

Another key contribution is an approach for rapid and efficient reasoning to identify semantically unsafe state spaces. While the robot has access to many descriptions of safe, nominal data Ω_s, it faces the challenge of identifying semantically unsafe regions that are naturally OOD compared with Ω_s. We propose to anticipatively enumerate a set of high-level failure modes Φ (e.g. "Near Human," "Turbulence," "High Temperature"). Then, we construct functions that determine the safety of a description semantically by measuring if a new state description is anomalously close to a high-level failure mode $\phi \in \Phi$, relative to the safe observations in Ω_s. Our approach is based on the insight that despite the potentially infinite variations in details of dangerous scenarios, most semantically unsafe scenarios are associated with a limited number of abstract, anticipated modes.

One approach to anticipate the abstract semantic failure modes that the environment may produce is manually identifying them, but this does not scale and adapt well to novel OOD settings. We propose using pre-trained foundation models to reason about failure modes capturing broad classes of potential semantically unsafe scenarios a robot may encounter in an environment. We create a structured prompt which has a general description of the environment, an image (if the reasoning model can process them), and a query for a list of relevant semantic failure modes. We extract the phrases of failure modes set Φ from the response as shown in the left part of Fig. 4. This step of forecasting failure modes morally corresponds to the *Failure Mode and Effects Analysis* stage of describing potential failures in a system for robust risk assessment [59].

To quantify the relationship of state description ω to some failure mode $\phi \in \Phi$, we leverage semantic understanding capabilities of a text embedding model $\texttt{Embed}$. During the nominal trajectory or offline, we construct a set of safe embedding vectors $\mathcal{E}_s := \{\texttt{Embed}(\omega_s), \forall \omega_s \in \Omega_s\}$ from safe state observation set Ω_s and obtain each failure mode's embedding $\mathbf{e}_\phi := \texttt{Embed}(\phi)$. We use cosine similarity-based function

$$\texttt{sim}(\mathbf{e}_i, \mathbf{e}_\phi) := 1 - \frac{\mathbf{e}_i \cdot \mathbf{e}_\phi}{||\mathbf{e}_i||\, ||\mathbf{e}_\phi||}$$

to measure semantic affinity of state description embedding vector $\mathbf{e}_i$ to ϕ. We identify if a state description is significantly closer to failure mode ϕ compared with safe data Ω_s and therefore is unsafe, by calibrating a threshold Δ_ϕ using conformal prediction on the $\alpha \in (0, 1)$ quantile of safe data:

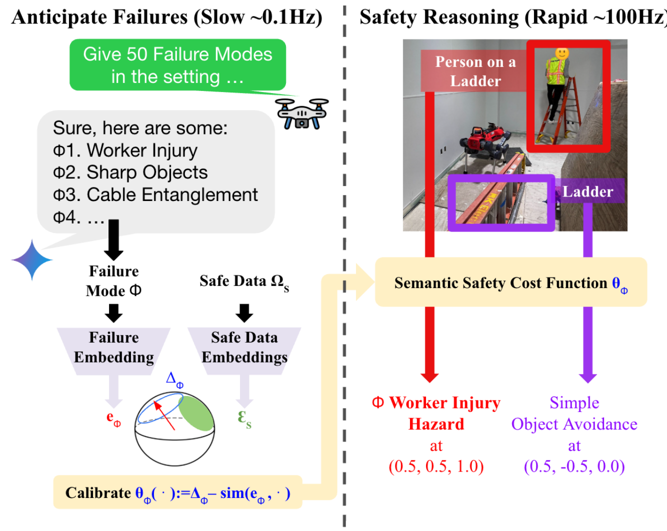


Figure 4: FORTRESS employs foundation model reasoners to anticipate failure modes. It then calibrates thresholds in the embedding model space to determine if new state descriptions more similar to failure modes than safe data Ω_s. During safety-critical moments, the semantic safety cost functions rapidly identify physical unsafe state regions during an ANYmal robot's deployment. FORTRESS differentiates the safety of a ladder from a person standing on one, anticipating worker injuries without encountering failures in Ω_s.

$$\Delta_\phi \;:=\; \sup\big\{\delta \in \mathbb{R} \;:\; \big|\{e_s \in \mathcal{E}_s \;:\; \texttt{sim}(e_s,\, e_\phi) \geq \delta\}\big| \;\geq\; (1-\alpha)N\big\}, \tag{1}$$

which is the maximum real scalar that lower bounds at least $\lceil (1-\alpha)N \rceil$ safe data similarity scores.

To determine which regions are semantically safe or hazardous for the robot, we construct cost functions with physical awareness of the state space. With function $\texttt{nearby}$ (Section 3) and failure mode ϕ, we propose semantic safety cost functions that analyze concepts within radius l_ϕ of state x:

$$\theta_\phi(x) := \Delta_\phi - \texttt{sim}(\texttt{Embed}(\texttt{nearby}(x,\, l_\phi)),\, e_\phi), \tag{2}$$

where the robot at x is hazardous if $\theta_\phi(x) > 0$. Intuitively, any state description's embedding within Δ_ϕ semantic similarity of failure mode embedding e_ϕ is classified as semantically unsafe (right part of Fig 4). We use the notation $h \in \Phi \cup \{c\}$ to indicate all failure modes and the collision hazard.

4.3 Reasoning about Safe Plans (M3)

FORTRESS's third component is generating a semantically safe path to a goal. We propose a control theoretic optimization framework that reasons about worst-case failure modes with the tools we developed in previous sections to generate semantically safe fallback plans to prevent OOD failures.

We construct a Reach-Avoid problem of entering a region $\mathcal{B}_\rho(g) := \{x : ||x - g|| \leq \rho\}$ (a ball of radius ρ centered around some goal g from Section 4.1) while avoiding semantically unsafe regions in the state space. We leverage semantic safety cost functions from (2) for all failure modes in Φ. We obtain trajectory plan $\tau = x_{\{1:T\}}$ with horizon T that starts at the robot's current location b and reaches goal location g by solving the following where $\mathbb{N}^{\leq T}$ indicates the set $\{1, 2, ...T\}$:

$$\tau^* = \arg\min_{\tau} \max_{h \in \Phi \cup \{c\}} \max_{x \in \tau} \theta_h(x) \qquad \begin{aligned} \text{s.t.}\quad & x_1 = b \\ & \exists k \in \mathbb{N}^{\leq T}, \quad \{x_i | k \leq i \leq T\} \subseteq \mathcal{B}_\rho(g) \\ & \exists u \in \mathcal{U}, \forall j \in \mathbb{N}^{\leq T-1} \quad x_{j+1} = f(x_j, u), \end{aligned} \tag{3}$$

where τ^* is the optimal trajectory. We define $\Theta^* := \max_{h \in \Phi \cup \{c\}} \max_{x \in \tau^*} \theta_h(x)$ as the minimaximax objective value. This optimization ensures the trajectory remains semantically and physically

Method	(Best) Embed. Model	Boat (Synthetic) TPR	TNR	Bal. Acc.	Vehicle (Synthetic) TPR	TNR	Bal. Acc.	ANYmal **(HARDWARE)** TPR	TNR	Bal. Acc.
10 modes (Ours)	Voyage AI	0.97	0.97	**0.97**	0.81	0.98	0.89	0.59	0.87	0.73
	Qwen2	0.78	0.66	0.72	0.65	0.90	0.78	0.82	0.98	**0.90**
	OpenAI	0.72	0.90	0.81	0.83	0.98	**0.90**	0.51	0.82	0.66
1 mode "Safe"	Voyage AI	0.34	0.78	0.56	0.05	0.96	0.50	0.02	0.98	0.50
	Qwen2	0.92	0.10	0.51	0.05	1.00	0.53	0.65	0.60	0.63
	OpenAI	0.61	0.47	0.54	0.25	0.86	0.55	0.15	0.97	0.56
GPT-4o	N/A - prompting	0.79	0.73	0.76	1.00	0.23	0.61	0.89	0.40	0.64
o3-mini	N/A - prompting	0.79	0.87	0.83	0.73	0.93	0.83	0.95	0.80	0.87
o4-mini	N/A - prompting	0.72	0.79	0.76	0.88	0.85	0.86	1.00	0.70	0.85

Table 1: Accuracy of our approach for calibrating embedding model-based cost functions to detect semantically unsafe descriptions on synthetic datasets and data collected from ANYmal hardware exploring a room under construction. Performance metrics are measured with True Negative Rate (TNR), True Positive Rate (TPR), and Balanced Accuracy. Results are shown for `FORTRESS` with 10 failure modes, ablation baselines that compute similarity distance to 1 mode called "Safe," and on-the-fly prompting of slow reasoning models.

safe (i.e. $\theta_h(x) \leq 0$) if possible by minimizing worst-case influences of failures along the trajectory. The first constraint forces the trajectory to start at the robot's current location, the second requires the trajectory will reach and remain indefinitely near the goal, and the third guarantees the trajectory is dynamically feasible. This optimization structure is related to HJ reachability's [60] game-theoretic optimization by scoping out the "adversary's" strategies and considering worst-case bounded "disturbances" that the environment may produce (i.e. potential hazards $\Phi \cup \{c\}$).

We employ a combination of motion planning and path tracking to find an approximate solution to optimization of (3) and compute the executable controls. We use Rapidly exploring Random Trees (RRT) [61] to plan a trajectory to the goal region using the cost functions $\theta_h(x)$ to guide and invalidate unsafe states during the search procedure. Then, we employ Model Predictive Control (MPC) or Linear Quadratic Regulator (LQR) to obtain controls for tracking the RRT plan. By inflating the distance thresholds $l_h, \ h \in \Phi \cup \{c\}$ for the cost functions outlined in Section 4.2 and ensuring the incremental distance of the planner is small enough to account for the error of the path tracking algorithm, we can leverage results similar to [62, 63] and prove the executed trajectory will align with the safety and reachability requirements (formal statement and proof in appendix).

Once entering the goal region $\mathcal{B}_\rho(g)$, the robot executes controls to ensure control invariance of the fallback set (e.g. initiate landing, hovering in place, etc). If the planner cannot identify a safe trajectory (i.e. $\Theta^* > 0$), we iterate through strategies $\sigma_i \in \Sigma$ from Section 4.1, selecting a new goal from $\mathcal{G}_i$ and recomputing the optimization of (3), until a safe trajectory is produced (i.e., $\Theta^* \leq 0$). During planning, we can log which hazards in $\Phi \cup \{c\}$ prevented the implementation of any fallback strategy, providing interpretable explanations for why certain strategies in Σ could not be executed.

5 Experiments

We conduct various experiments to test three hypotheses of our approach `FORTRESS`:

H1: By reasoning about the safety of a scene using text embedding models and a limited set of failure modes, we can identify OOD failures (i.e. semantically unsafe descriptions) more accurately than on-the-fly prompting of slow foundation models reasoning about safety of a description.

H2: `FORTRESS` can leverage foundation model reasoners to automatically anticipate failure modes that semi-monotonically improve classification accuracy using embedding-based cost functions.

H3: By performing rapid constraints and planning computation when requested for a fallback plan, we can generate fallback plans in *real time* that have improved planning success and safety compared to baseline approaches that hard-code fallbacks and/or perform naïve object avoidance.

5.1 Embedding classification

We test **H1** by measuring the accuracy of our approach in extracting semantic safety constraints. In particular, we create synthetic datasets in several domains such as aerial vehicle drone taxis, self-navigating maritime vessels, and autonomous vehicles. We compile synthetic training data for

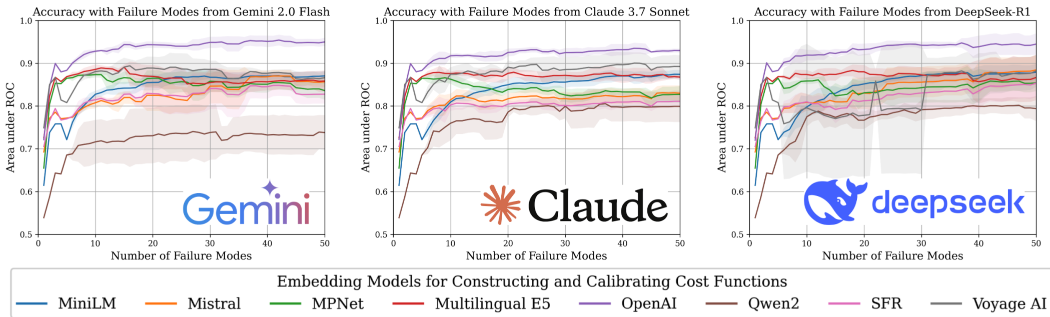


Figure 5: **Increasing Failure Coverage from Prompting Reasoners:** We prompt Gemini 2.0 Flash, Claude 3.7 Sonnet, and DeepSeek-R1 reasoners to anticipate 1 to 50 failure modes and calibrate cost functions based on eight embedding models to classify if semantically unsafe drone descriptions are closer to failure modes than the distance from the safe drone data to these modes. We use AUROC as a metric for classifiers' performance. Results for more models like OpenAI's reasoners are in the appendix.

semantic state descriptions Ω_s^{train} that have safe collections of environments and concepts for each domain that robots are known to handle in their nominal operations safely. We also construct safe description test data Ω_s^{test} different from those in training data, as well as dangerous description test data Ω_d^{test} that contain unsafe collections. We evaluate and calibrate eight text embedding models for classification: MiniLM (22M) [64], Mistral (7B) [65, 66], MPNet (110M) [67, 68], Multilingual-E5 (560M) [69], OpenAI Text Embedding Large [70], Qwen2 (7B) [71], Salesforce (SFR) Embedding Mistral (7B) [72], and Voyage AI's voyage-3-large [73].

In the first set of experiments, we define around 10 high-level descriptions of failure modes for each domain. These descriptions are kept concise to cover a wide range of potential unsafe scenarios. We calibrate thresholds for each model and domain using the safe dataset Ω_s^{train}. We present accuracy results on the total testing dataset $\Omega_s^{\text{test}} \cup \Omega_d^{\text{test}}$ for the Autonomous Boat and Vehicle synthetic datasets in Table 1. We additionally compare with an ablation that considers only a single mode "Safe" and prompting (slow) reasoning models. The results demonstrate that for each domain, FORTRESS's approach can achieve a balanced accuracy performance higher than 0.90, effectively distinguishing between safe and unsafe scenes. The other approaches perform poorly since they reason about semantic safety too abstractly (i.e. with 1 mode "Safe") or too fine-grained (i.e. reasoning models can over-extrapolate from scene descriptions).

5.2 Accuracy of Proactively Anticipating Failure Modes

We verify **H2** by exploring automating failure mode generation by querying LLM with a prompt that specifies the robot's setting and asks for potential failure modes to anticipate. Effectively, we employ reasoning models to scope out the "adversarial" strategies. Note that this querying process can be performed during the execution of the nominal trajectory of the robot when it enters a new deployment region or offline if the environment is already known. Fig. 5 shows results of querying various reasoning models for up to 50 failure modes and measuring the performance of the classification performance of the calibrated embedding model cost functions. By querying models for more failure modes, we observe the Area Under the Receiver Operating Characteristic (AUROC) curves, which capture semantic safety classification performance, generally improve and stabilize. OpenAI and Voyage AI embedding models achieve increases in accuracy, with AUROC values above 0.9, outperforming those generated from manual failure modes. Overall, FORTRESS can effectively identify failure modes to improve coverage of semantically unsafe OOD failure descriptions and avoid suggestions of spurious modes that exacerbate false positive rates.

We also test the classification accuracy of our approach on data we collect from deploying an ANYmal robot (Fig. 4) in a room under construction[1]. We feed a prompt and an image of the environment to Claude 3.7 Sonnet and extract around 10 failure modes. We calibrate thresholds on a deployment

[1]ANYmal video demo link at https://www.youtube.com/watch?v=xU-egPQjkFo

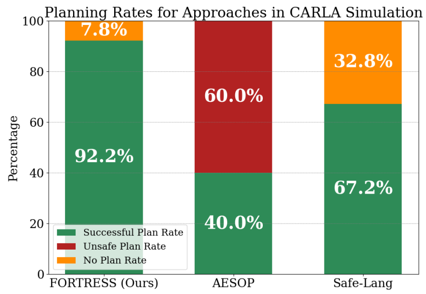


Figure 6: Planning rates of FORTRESS versus AESOP [13] and Safe-Lang [18] for drone robot in CARLA simulation. We augment baselines with our VLM goal identification for fair comparison.

Computation Times for FORTRESS on Drone Hardware

Component	Mean Time (s)	STD Dev. (s)
Query Molmo for Goal points	5.82	0.13
Query 2.0 Flash for Failure Modes	3.68	0.33
Query 3.7 Sonnet for Failure Modes	15.65	0.51
Query R1 for Failure Modes	12.76	1.22
Calibrate Voyage AI Embeddings	3.91	0.08
Calibrate OpenAI Embeddings	5.43	0.19
Safety Reasoning Inference	0.011	0.006
Reach-Avoid Planner	1.28	0.55

Table 2: Component times of FORTRESS algorithm during quadrotor drone hardware deployment with Jetson Nano. Px4 manages the communication interface, and motion capture sensors enable drone localization. The first grouping of rows is computations done at a low frequency during nominal trajectory or offline; the second grouping is executed during the safety-critical moment when the response is needed.

with relatively safe and manageable concepts detected with an object detector (e.g. ladders, paint cans, and boxes), and measure semantic safety detection rate when deploying the ANYmal near both safe concepts and unsafe ones (e.g. person on a ladder, caution tape, and cables). We present results in the rightmost column group of Table 1 and in the appendix.

5.3 Safe Fallback Planning in Real Time

We measure the success and safety of the generated plans for a single fallback strategy of landing on buildings in the CARLA simulation and hardware experiments on a drone to validate **H3**. In CARLA (Fig. 3), we create an agent with the dynamics of a drone to navigate over an urban landscape with firetrucks, people, cars, traffic cones, etc. We compare our approach with recovery planning based on AESOP [13], which does not avoid semantically unsafe regions in its fallback stage, and adapting Safe-Lang [18] to have a VLM identify unsafe regions for its naïve avoidance approach. In Fig. 6, FORTRESS improves the success rate and safety of generated fallback plans due to its nuanced reasoning that can accurately demarcate semantically safe and unsafe regions.

We also deploy our framework on drone hardware[2] to test whether FORTRESS can operate in real-world settings: FORTRESS determines fallback goals for strategy of landing on building roofs, extract semantic safety constraints like High Temperature and Construction, and rapidly generate semantically safe plans seen in Fig. 1. We measure each component of our approach on Jetson Nano hardware shown in Table 2. The safe embedding reasoner and the reach-avoid planner operate rapidly and therefore are deployed instantaneously once the runtime monitor has triggered a fallback response. While our fallback strategy identification and safety calibration modules that query the multi-modal reasoners take longer, we perform these operations at a low frequency during nominal operations (see Sections 4.1 & 4.2), caching potential locations and semantic safety functions and loading when needed. Additional details on hardware experiments are in the appendix.

6 Discussion and Conclusion

We present FORTRESS, a framework that prevents OOD failures by bridging open-world reasoning with dynamics-aware planning and control to generate fallback strategies. At a low frequency in nominal operations or offline, FORTRESS uses foundation models to identify fallback goals, anticipate failure modes, and calibrate semantic safety reasoners, caching the goals and semantic cost functions to mitigate latency in safety-critical moments. At runtime, it adaptively synthesizes fallback plans via reach-avoid analysis guided by these goals and semantic constraints. We validate our semantic safety reasoners' detection of potential OOD failures on synthetic maritime and vehicle datasets as well as real-world data from an ANYmal robot. We deploy FORTRESS on drone hardware and in simulation, with improved planning success and safety.

[2] Quadrotor hardware video demo link at https://www.youtube.com/watch?v=a0XZgwoNLos

7 Limitations

Our proposed framework currently has limitations that open several promising avenues for future work. Although we currently write semantic fallback strategies for robots and environments, extensions could develop methods that automatically extract semantic descriptions, infer the relevant fallback goals, and adapt them across diverse robotic platforms and operation settings. Non-static fallback strategy goals can also be investigated depending on the robot and environment, such as merging into a different lane with moving traffic in the autonomous vehicle setting. Furthermore, our mapping of semantically unsafe regions to static, fixed-radius avoidance regions could be extended to handle dynamic or context-dependent constraints such as adaptive avoid boundaries, dynamic obstacles, terrain-specific factors including varying surface friction, or environmental disturbances like wind to enable more nuanced reasoning about when and how failures arise. These semantic fallback goals and failures can be retrieved from rule books with high-level specifications on handling the presence of semantically unsafe situations (e.g. a handbook on aviation, naval, and traffic regulations). Finally, while in this paper we focus on preventing OOD failures, future work can expand on our framework for diagnosis and fallback strategies that manage the safety and recovery of robots actively experiencing failures (e.g. recovery with minimal damage during a collision or from a fire).

Acknowledgments

We thank the Stanford Robotics Center for their assistance with experiments using the ANYmal robot. This work is supported by the NASA University Leadership Initiative, Torc Robotics, and Toyota Research Institute.

References

[1] S. Council. Waymo expands to four more bay area cities. *SFGATE*, March 2025. URL `https://www.sfgate.com/tech/article/waymo-map-expands-bay-area-20215700.php`.

[2] BMW Group. Bmw group tests humanoid robots in car production. Press Release, March 2024. URL `https://www.bmwgroup.com/en/news/general/2024/humanoid-robots.html`.

[3] Oil Review Middle East. Saudi aramco launches auv for underwater surveying and inspection. *Oil Review Middle East*, March 2024. URL `https://oilreviewmiddleeast.com/industry/saudi-aramco-launches-auv-for-underwater-surveying-and-inspection`.

[4] Greek City Times. Greece expands use of drones for fire response. *Greek City Times*, June 2024. URL `https://greekcitytimes.com/2024/06/24/greece-expands-drones-for-fire-response/`.

[5] UK Civil Aviation Authority. Infrastructure inspections with drones made easier under new rules. *UK Civil Aviation Authority Newsroom*, Oct 2024. URL `https://www.caa.co.uk/newsroom/news/infrastructure-inspections-with-drones-made-easier-under-new-rules/`.

[6] EV Magazine. How autonomous vehicles are transforming agriculture. *EV Magazine*, June 2024. URL `https://evmagazine.com/articles/autonomous-vehicles-transform-agriculture`.

[7] Sixth Tone. Drones take flight to deliver rural china. *Sixth Tone*, June 2024. URL `https://www.sixthtone.com/news/1016069`.

[8] N. Drummond and R. Shearer. The open world assumption. In *eSI workshop: the closed world of databases meets the open world of the semantic web*, volume 15, page 1, 2006.

[9] S. Lu, Y. Wang, L. Sheng, A. Zheng, L. He, and J. Liang. Recent advances in ood detection: Problems and approaches. *arXiv preprint arXiv:2409.11884*, 2024.

[10] R. Sinha, A. Sharma, S. Banerjee, T. Lew, R. Luo, S. M. Richards, Y. Sun, E. Schmerling, and M. Pavone. A system-level view on out-of-distribution data in robotics. *arXiv preprint arXiv:2212.14020*, 2022.

[11] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[12] A. Elhafsi, R. Sinha, C. Agia, E. Schmerling, I. A. Nesnas, and M. Pavone. Semantic anomaly detection with large language models. *Autonomous Robots*, 47(8):1035–1055, 2023.

[13] R. Sinha, A. Elhafsi, C. Agia, M. Foutter, E. Schmerling, and M. Pavone. Real-time anomaly detection and reactive planning with large language models. In *Robotics: Science and Systems*, 2024.

[14] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, page 02783649241281508, 2023.

[15] H. Pham, Z. Dai, G. Ghiasi, K. Kawaguchi, H. Liu, A. W. Yu, J. Yu, Y.-T. Chen, M.-T. Luong, Y. Wu, et al. Combined scaling for zero-shot transfer learning. *Neurocomputing*, 555:126658, 2023.

[16] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[17] J. Jabbour and V. J. Reddi. Generative ai agents in autonomous machines: A safety perspective. *arXiv preprint arXiv:2410.15489*, 2024.

[18] L. Santos, Z. Li, L. Peters, S. Bansal, and A. Bajcsy. Updating robot safety representations online from natural language feedback. *arXiv preprint arXiv:2409.14580*, 2024.

[19] L. Lindemann, Y. Zhao, X. Yu, G. J. Pappas, and J. V. Deshmukh. Formal verification and control with conformal prediction. *arXiv preprint arXiv:2409.00536*, 2024.

[20] K. Garg, S. Zhang, O. So, C. Dawson, and C. Fan. Learning safe control for multi-robot systems: Methods, verification, and open challenges. *Annual Reviews in Control*, 57:100948, 2024.

[21] C. Dawson, S. Gao, and C. Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 2023.

[22] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-Jacobi reachability: A brief overview and recent advances. In *Conf. on Decision and Control*, 2017.

[23] M. Chen and C. J. Tomlin. Hamilton–jacobi reachability: Some recent theoretical advances and applications in unmanned airspace management. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:333–358, 2018.

[24] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *Hybrid Systems: Computation and Control*. ACM, 2015.

[25] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *European Control Conf.*, 2019.

[26] K.-C. Hsu, H. Hu, and J. F. Fisac. The safety filter: A unified view of safety-critical control in autonomous systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 7, 2023.

[27] Y. Chen, M. Jankovic, M. Santillo, and A. D. Ames. Backup control barrier functions: Formulation and comparative study. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6835–6841. IEEE, 2021.

[28] O. So, Z. Serlin, M. Mann, J. Gonzales, K. Rutledge, N. Roy, and C. Fan. How to train your neural control barrier function: Learning safety filters for complex input-constrained systems. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11532–11539. IEEE, 2024.

[29] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[30] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[31] A. Saviolo, P. Rao, V. Radhakrishnan, J. Xiao, and G. Loianno. Unifying foundation models with quadrotor control for visual tracking beyond object categories. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7389–7396. IEEE, 2024.

[32] G. Chen, X. Yu, N. Ling, and L. Zhong. Typefly: Flying drones with large language model. *arXiv preprint arXiv:2312.14950*, 2023.

[33] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[34] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[35] H. Gao, Z. Wang, Y. Li, K. Long, M. Yang, and Y. Shen. A survey for foundation models in autonomous driving. *arXiv preprint arXiv:2402.01105*, 2024.

[36] M. Minderer, A. Gritsenko, and N. Houlsby. Scaling open-vocabulary object detection. *Advances in Neural Information Processing Systems*, 36:72983–73007, 2023.

[37] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin. An efficient reachability-based framework for provably safe autonomous navigation in unknown environments. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1758–1765. IEEE, 2019.

[38] S. L. Herbert, S. Bansal, S. Ghosh, and C. J. Tomlin. Reachability-based safety guarantees using efficient initializations. In *Conf. on Decision and Control*, 2019.

[39] Y. Wang, Y. Lin, and M. Althoff. Interaction-aware trajectory repair in compliance with formalized traffic rules. In *The 27th IEEE International Conference on Intelligent Transportation Systems (IEEE ITSC 2024)*, 2024.

[40] Y. Lin, C. Li, M. Ding, M. Tomizuka, W. Zhan, and M. Althoff. Drplanner: Diagnosis and repair of motion planners for automated vehicles using large language models. *IEEE Robotics and Automation Letters*, 2024.

[41] Y. Lin, Z. Xing, X. Han, and M. Althoff. Traffic-rule-compliant trajectory repair via satisfiability modulo theories and reachability analysis. *arXiv preprint arXiv:2412.15837*, 2024.

[42] K. Chakraborty, Z. Feng, S. Veer, A. Sharma, B. Ivanovic, M. Pavone, and S. Bansal. System-level safety monitoring and recovery for perception failures in autonomous vehicles. *arXiv preprint arXiv:2409.17630*, 2024.

[43] F. Lin, X. Fang, and Z. Gao. Distributionally robust optimization: A review on theory and applications. *Numerical Algebra, Control and Optimization*, 12(1):159–212, 2022.

[44] J. Hejna, C. Bhateja, Y. Jiang, K. Pertsch, and D. Sadigh. Re-mix: Optimizing data mixtures for large scale imitation learning. *arXiv preprint arXiv:2408.14037*, 2024.

[45] A. S. Chen, G. Chada, L. Smith, A. Sharma, Z. Fu, S. Levine, and C. Finn. Adapt on-the-go: Behavior modulation for single-life robot deployment. *arXiv preprint arXiv:2311.01059*, 2023.

[46] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr, 2018.

[47] M. Salehi, H. Mirzaei, D. Hendrycks, Y. Li, M. H. Rohban, and M. Sabokrou. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *arXiv preprint arXiv:2110.14051*, 2021.

[48] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.

[49] W. Liu, X. Wang, J. Owens, and Y. Li. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475, 2020.

[50] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[51] A. Gupta, K. Chakraborty, and S. 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.

[52] R. Sinha, E. Schmerling, and M. Pavone. Closing the loop on runtime monitors with fallback-safe mpc. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 6533–6540. IEEE, 2023.

[53] C. Richter and N. Roy. Safe visual navigation via deep learning and novelty detection. 2017.

[54] M. Wortsman, G. Ilharco, J. W. Kim, M. Li, S. Kornblith, R. Roelofs, R. G. Lopes, H. Hajishirzi, A. Farhadi, H. Namkoong, and L. Schmidt. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7959–7971, 2022.

[55] C. Agia, R. Sinha, J. Yang, Z. Cao, R. Antonova, M. Pavone, and J. Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 689–723. PMLR, 2025.

[56] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, et al. Simple open-vocabulary object detection. In *European conference on computer vision*, pages 728–755. Springer, 2022.

[57] M. Sohan, T. Sai Ram, R. Reddy, and C. Venkata. A review on yolov8 and its advancements. In *International Conference on Data Intelligence and Cognitive Informatics*, pages 529–545. Springer, 2024.

[58] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv preprint arXiv:2409.17146*, 2024.

[59] M. Rausand and A. Hoyland. *System reliability theory: models, statistical methods, and applications*, volume 396. John Wiley & Sons, 2003.

[60] A. Bajcsy and J. F. Fisac. Human-ai safety: A descendant of generative ai and control systems safety. *arXiv preprint arXiv:2405.09794*, 2024.

[61] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[62] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin. Fastrack: A modular framework for fast and guaranteed safe motion planning. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1517–1522, 2017. doi:10.1109/CDC.2017.8263867.

[63] B. D. Luders, S. Karaman, E. Frazzoli, and J. P. How. Bounds on tracking error using closed-loop rapidly-exploring random trees. In *Proceedings of the 2010 american control conference*, pages 5406–5412. IEEE, 2010.

[64] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788, 2020.

[65] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*, 2023.

[66] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[67] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867, 2020.

[68] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[69] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*, 2024.

[70] OpenAI. New embedding models and api updates. Blog Post, Jan 2024. URL `https://openai.com/index/new-embedding-models-and-api-updates/`.

[71] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

[72] R. Meng, Y. Liu, S. R. Joty, C. Xiong, Y. Zhou, and S. Yavuz. Sfr-embedding-mistral:enhance text retrieval with transfer learning. Salesforce AI Research Blog, 2024. URL `https://www.salesforce.com/blog/sfr-embedding/`.

[73] V. AI. Text embeddings documentation, 2025. URL `https://docs.voyageai.com/docs/embeddings`. Accessed: 2025-04-09.

Contents

A Notation

Notation	Description
x	Robot State
$\mathcal{X}$	State Set
u	Control action
$\mathcal{U}$	Control Set
f	Dynamics
Ω_s	Set of safe, nominal semantic state descriptions
N	number of descriptions in of Ω_s
Ω_d	Set of semantically unsafe semantic state descriptions
ω	a semantically unsafe semantic state description
Σ	Set of semantic description of fallback strategies
σ_i	i^{th} semantic description of in Σ
$\mathcal{G}_i$	set of 3D goal coordinates from querying for strategy σ_i from VLM
g	a 3D goal coordinate
Φ	Set of semantic failure modes
ϕ	a failure mode
l_ϕ	physical distances to keep from failure mode ϕ and physical objects
l_c	physical distances to keep from failure mode ϕ and physical objects
$\texttt{nearby}(x, l)$	function that returns state description of concepts in radius l around x
$\theta_c(x)$	collision hazard cost function returning positive iff x is within l_c of physical obstacle
$\texttt{Embed}(\omega)$	function Text embedding model producing a vector from description ω
$\mathcal{E}_s$	Set of embeddings vectors of descriptions in Ω_s
e_ϕ	embedding vector of failure ϕ
$\texttt{sim}(e_i, e_\phi)$	similarity cost function based on cosine similarity of vectors e_i and e_ϕ
Δ_ϕ	failure embedding similarity threshold calibrated for ϕ on safe data Ω_s
α	quantile for threshold
$\theta_\phi(x)$	semantic safety cost function detecting if state x is in region that could experience ϕ
$\Phi \cup \{c\}$	all failure modes and the collision hazard (used when representing cost functions and distances)
ρ	radius around goal coordinate determining reach region
$\mathcal{B}_\rho(g)$	ball of radius ρ around goal g
b	beginning point for plan/location where fallback response was triggered
τ	trajectory plan
T	horizon/steps in trajectory
$x_{\{1:T\}}$	sequence of states $\{x_1, x_2, ...x_T\}$
$\mathbb{N}^{\leq T}$	set of natural numbers from 1 to T
Ω_s^{train}	training/calibrating set of safe, nominal semantic state descriptions
Ω_s^{test}	testing set of safe, nominal semantic state descriptions
Ω_d^{test}	testing/validating set of unsafe/failure semantic state descriptions
Λ	inverse covariance matrix of safe embedding vector set $\mathcal{E}_s$

B Evaluation of Semantic Safety Cost Functions

B.1 Additional Experiments

ROC Curves of Calibrated Embedding Models using 10 Failure Modes on Synthetic and Real World Datasets

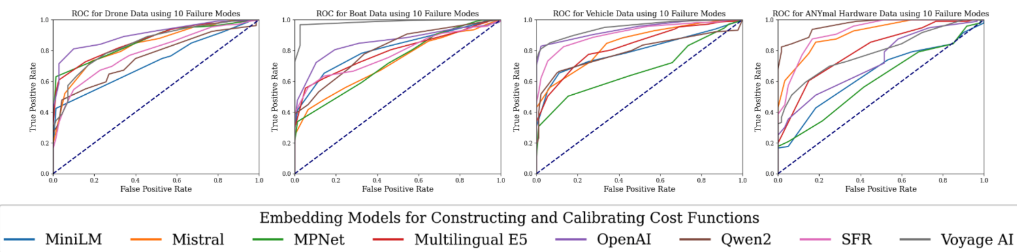


Figure 7: ROC curves using around 10 failure modes with varying percentile α thresholds on autonomous drones, boats, and vehicle environments using cosine similarity on 8 embedding models.

ROC Curves of Calibrated Embedding Models using Mahalanobis Distance on Synthetic and Real World Datasets

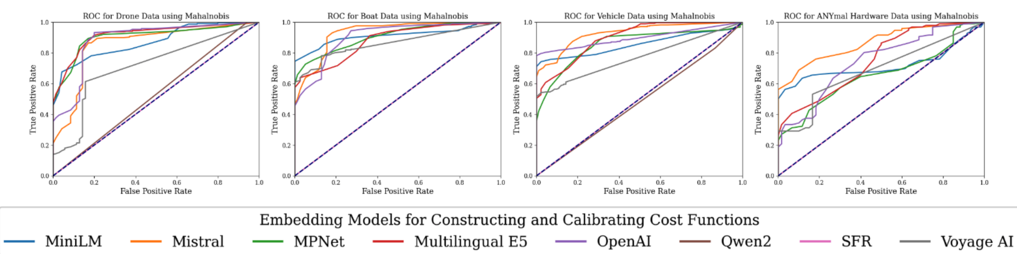


Figure 8: ROC curves using around 10 failure modes with varying percentile α thresholds on autonomous drones, boats, and vehicle environments using Mahalanobis distance calibrated on cosine similarity on 8 embedding models.

ROC Curves of Calibrated Embedding Models using 1 Mode “Safe” on Synthetic and Real World Datasets

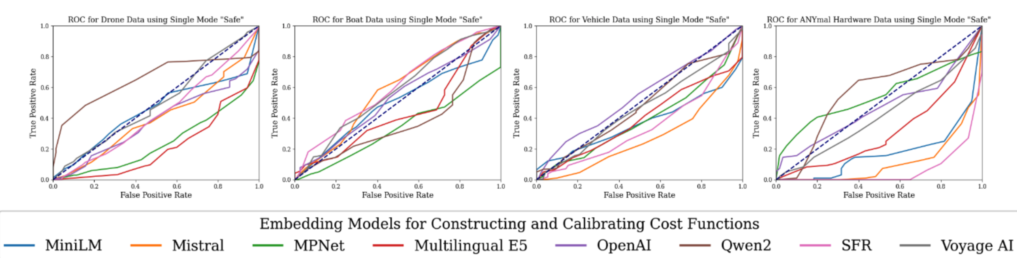


Figure 9: ROC curves using only the “Safe” Mode with varying percentile α thresholds on autonomous drones, boats, and vehicle environments using cosine similarity on 8 embedding models.

We perform a more comprehensive analysis on our semantically safety cost functions to extract OOD failures. Specifically, we present ROC curves for multiple datasets and calibrate various embedding models. Different classification accuracies are given by varying the quantile threshold alpha in calibration defined in Section 4.2. Our benchmarks include synthetic datasets for autonomous drones, boats, and vehicles. We also present results on real-world data we collected from ANYmal robot deployments.

Our approach with cosine similarity: For the main `FORTRESS` classifier, we construct failure modes per domain and use cosine similarity in the embedding space to measure proximity to each mode (see Fig. 7). For the ANYmal dataset, we query Claude Sonnet 3.7 and extract 10 failure modes.

Our approach with Mahalanobis: We also explore alternative similarity measures and implement a variant inspired by the Mahalanobis distance. We compute the inverse covariance matrix Λ of the safe dataset embeddings $\mathcal{E}_s$. Given a new scene embedding e_t, the similarity distance function to embedding failure e_ϕ is given by:

$$\texttt{sim}_{\texttt{mahalanobis}}(e_t, e_\phi) := \sqrt{(e_t - e_\phi)^\top \Lambda (e_t - e_\phi)}. \tag{4}$$

Using this similarity metric, we calibrate the corresponding threshold $\Delta_\phi^{\texttt{mahalanobis}}$ for our cost function. The results of the classification performance for the models on the various datasets are shown in Fig. 8. The curves for Mahalanobis distance do not show a significant improvement in accuracy compared with just cosine similarity.

Ablation with single mode: As an ablation baseline, we employ a single mode called "Safe." We compute the cosine similarity between each scene embedding and the "Safe" embedding, and classify scenes with closer similarity (i.e. lower `sim` values) as safe (opposite of our failure mode decision rule in (1)). Results in Fig. 9.

B.2 Prompts for Embedding Models

For autonomous drones:

```
I am the runtime monitor for a vision-based autonomous vertical takeoff and
    landing (VTOL) drone operating as an urban air taxi. My job is to analyze
     the observations of the VTOL and identify whether certain points along
    the VTOL's trajectory are prone to failure rendering the VTOL unable to
    handle them as part of its nominal operations. The VTOL's observations
    take the form of object detections generated by an onboard perception
    pipeline.

Scene Description: The surroundings of the VTOL include:
${CONCEPT_LIST}
```

For autonomous boats:

```
I am the runtime monitor for a vision-based autonomous boat transporting
    cargo. My job is to analyze the observations of the boat and identify
    whether certain points along the boat's trajectory are prone to failure
    rendering the boat unable to handle them as part of its nominal
    operations. The autonomous boat's observations take the form of object
    detections generated by an onboard perception pipeline.

Scene Description: The surroundings of the boat include:
${CONCEPTS_LIST}
```

For autonomous vehicles:

```
I am the runtime monitor for a vision-based autonomous vehicle operating as
    an urban ground taxi cab. My job is to analyze the observations of the
    vehicle and identify whether certain points along the vehicle's
    trajectory are prone to failure rendering the autonomous vehicle unable
    to handle them as part of its nominal operations. The vehicle's
    observations take the form of object detections generated by an onboard
    perception pipeline.

Scene Description: The surroundings of the autonomous vehicle include:
${CONCEPTS_LIST}
```

For ANYmal robot:

```
I am the runtime monitor for a vision-based autonomous quadruped ANYmal robot
     operating in a construction area. My job is to analyze the observations
    of the quadruped and identify whether certain points along the quadruped'
    s trajectory are prone to failure for the robot or danger caused to the
    environment. The quadruped ANYmals's observations
take the form of object detections generated by an onboard perception
    pipeline.

Scene Description: The surroundings of the quadruped include:
${CONCEPTS_LIST}
```

B.3 Baseline implementation of on-the-fly prompting of slow reasoning models

As shown in Table 1 of the main paper, we also compare our approach with on-the-fly prompting of the reasoning models. In particular, we directly query the foundation model reasoners with a structured prompt to analyze whether or not a scene is safe. This would theoretically be queried during the safety critical moment to understand the semantically unsafe regions, so we term this as on-the-fly prompting. Table 1 in the main paper and Table 3 below show that simply querying the reasoning models results in poor accuracy and is much slower ($> 50\times$) than `FORTRESS` in analyzing semantic safety. An example prompt is:

```
I am the runtime monitor for a vision-based autonomous vertical takeoff and
    landing (VTOL) drone operating as an urban air taxi. My job is to analyze
     the observations of the VTOL and identify whether certain points along
    the VTOL's trajectory are prone to failure rendering the VTOL unable to
    handle them as part of its nominal operations. The VTOL's observations
    take the form of object detections generated by an onboard perception
    pipeline.

Scene Description: The surroundings of the VTOL include:
${CONCEPT_LIST}
Is this situation safe for the agent? Respond in the following format:
Answer: {Yes/No}
```

Reasoning Model	Mean Time (s)	STD Dev. (s)
GPT-4o	0.49	0.03
o3-mini	3.23	0.86
o4-mini	2.68	0.72

Table 3: Inference times of on-the-fly prompting of reasoning models.

C Reasoning Models for Failure Mode Generation

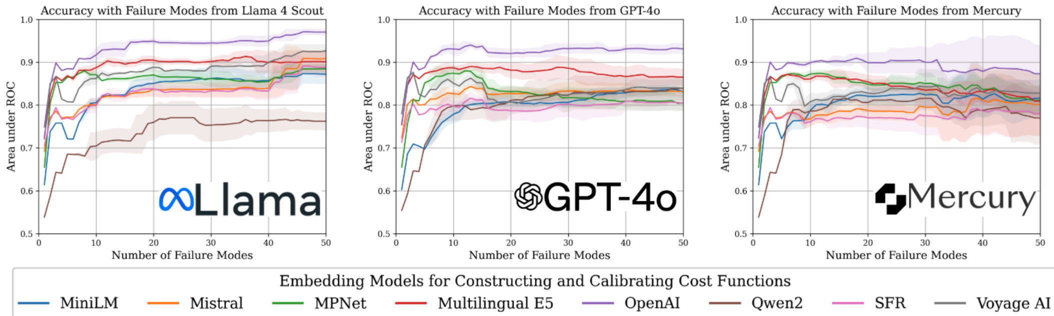


Figure 10: Increasing number of failure modes (1 to 50) taken from prompting Llama 4 Scout, OpenAI GPT-4o, and Mercury reasoners with Area under ROC curve as the measure of classifier model performance queried with eight different embedding models for cost functions over five seeds on the autonomous drone synthetic dataset.

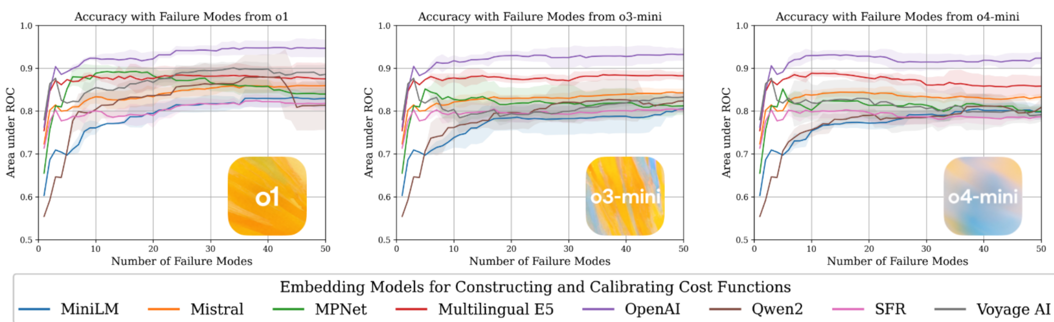


Figure 11: Increasing number of failure modes (1 to 50) taken from prompting o1, o3-mini, and o4-mini which are part of OpenAI's reasoning model series with Area under ROC curve as the measure of classifier model performance queried with eight different embedding models for cost functions over five seeds on the autonomous drone synthetic dataset.

We present results for querying foundation model reasoners for failure modes given scene descriptions and calibrating all the embedding model cost functions with cosine similarity to the failure modes. We present the accuracy of the embedding model cost functions incrementally increasing from 1 to 50 failure modes from each reasoner. We use Area under the ROC curve as the classification performance metric. We present the results on Llama 4 Scout, OpenAI GPT-4o, and Mercury in Fig. 10 and OpenAI reasoning model series in Fig. 11. For reference, each plot constitutes 8 embedding models $\times$ 50 evaluations on varying numbers of failure modes $\times$ 5 queries for failure mode lists from the reasoning models $=$ 2000 calibration experiments for each reasoning model.

Overall, `FORTRESS` demonstrates strong performance in leveraging foundation models to anticipate OOD failure modes and accurately classify the failure, with the OpenAI text embedding large model consistently achieving over 0.95 AUROC and Multilinguale-E5 and Voyage AI embedding models sometimes performing over 0.9. It is worth noting that the AUROC largely holds steady as we include more failure modes. A significant drop would be anticipated if spurious suggestions were causing safe descriptions to be misclassified as unsafe. However, the embedding model cost functions become gradually more accurate with more failure modes generated by the reasoners, suggesting that `FORTRESS` can employ the reasoners to effectively understand and generate more failure modes that accurately improve failure coverage.

A query and response from the Llama 4 Scout model look like:

```
Query: Given an urban view from a drone agent, provide concise descriptions
    no more than four words of at least 50 general potential external
    failures that the drone may encounter in this environment. Avoid internal
     problems/failures of the drone or its equipment. Examples include Low
    Visibility, Near Human, Near Vehicle in motion, High Temperature,
    Unauthorized Access.
Provide in the format:
n. failure

Response: Here are 50 potential external failures that the drone may
    encounter in an urban environment:

1. Low Visibility
2. Near Human
3. Near Vehicle
4. High Temperature
5. Unauthorized Access
6. Power Line Nearby
7. Tall Building Nearby
8. Strong Wind Gust
9. Rainy Weather
10. Snowfall
11. Foggy Conditions
12. Bird Strike
13. Insect Swarm
14. Air Traffic Nearby
15. Emergency Vehicle
16. Pedestrian Crossing
17. Road Construction
18. Bridge Nearby
19. Tunnel Entrance
20. Overhanging Tree
21. Loose Debris
22. Nearby Crane
23. High Voltage Lines
24. Radio Frequency Interference
25. Water Body Nearby
26. Steep Slope
27. Unstable Ground
28. Nearby Fire
29. Toxic Gas Leak
30. Nearby Accident
31. Protests or Riots
32. Large Animal Nearby
33. Narrow Alleyway
34. Billboard or Sign
35. Overhead Wires
36. Construction Site
37. Parked Vehicle
38. Moving Train
39. Boat or Ship
40. Nearby Helicopter
41. Telephone Lines
42. Severe Weather
43. Sandstorm
44. Nearby Explosion
45. Unmarked Obstacle
46. Cycling Pedestrian
47. Dense Vegetation
48. Steep Cliff
49. Water Spray
50. Dense Traffic

Let me know if you'd like me to expand on this list!
```

D CARLA Simulation Experiments

We also deploy and evaluate FORTRESS in the CARLA simulator. We model the ego spectator view as a drone agent observing the city from above the buildings. Some of the buildings have people, firetrucks, traffic cones, and cars on their roofs. Given that a runtime monitor has triggered the need for a fallback response in this situation, we are tasked with generating and executing a semantically safe fallback plan.

D.1 Implementation Details of FORTRESS in CARLA:

We first query the VLM Molmo [58] for 2D coordinates. For the fallback strategy of landing on a building roof, we might query something like:

```
Query: point to empty, horizontal building roofs.

Response: <points x1="32.4" y1="51.2" x2="38.5" y2="9.5" x3="42.1" y3="24.0"
    x4="74.6" y4="8.1" x5="89.5" y5="20.6" x6="92.4" y6="53.5" x7="97.5" y7
    ="69.3" alt="empty, horizontal building roofs">empty, horizontal building
     roofs</points>
```

The response has 2D xy coordinates (that have been normalized to be in the range 0-100) from which we extract the proposed empty building roof points. They correspond to the following locations:

From this we can use CARLA's built-in pose and position estimates, depth view, and camera intrinsics to construct 3D global coordinates from these points, which form our goal points.

For the semantic safety cost function, we employ the OpenAI text embedding model [70] and calibrated with cosine similarity (the accuracy of which is displayed on the leftmost image in Fig. 7).

During runtime, we identify the semantically unsafe regions in the state space and perform reach-avoid planning to enter into a 1-meter radius around one of the goals identified by Molmo while avoiding the unsafe regions. Specifically, we use open-vocabulary object detectors YOLOv8 [57] and OWLViT [56] to identify the concepts on the building roofs and depth maps for their locations. We make l_c as 2 meters and l_ϕ for all ϕ as 4 meters. We employ an RRT planner with an incremental step size of 0.5 meters and LQR to track the planned path. Because the object detectors are not always accurate from far distances, we perform a cycle of path tracking at most 15 steps of the RRT plan while querying the object detectors and replanning a trajectory to the goal from the anticipated location. In the replanning, stage if we find the original goal is infeasible (because there is no semantically safe plan to reach it), we try to plan paths to new nearby goals identified by the VLM.

D.2 Implementation Details of Baselines in CARLA

While there are no works we are aware of directly generate semantically safe fallback plans in real time to prevent OOD failures, we compare our approach with two baselines by adapting from adjacently related works AESOP [13] and Safe-Lang [18].

While AESOP focuses on deciding whether a situation is OOD and what semantic fallback response is needed, its fallback planner produces a trajectory to a manually predefined fallback goal without considering the safety of the plan. We use this baseline with a few modifications such as augmenting it with our approach of querying Molmo for fallback goal identification and employing the same planning. We also include naïve collision hazard avoidance of keeping 2 meters from physical objects to highlight the comparative improvements provided by our semantic safety cost function.

Safe-Lang, on the other hand, uses human language input to identify the semantically unsafe regions for the safe fallback policy/plans and does a simple object avoidance for this. Since it is impractical for humans to perform descriptive safety interventions especially in large scale deployments, we model this using the open-vocabulary object detectors to identify objects automatically and perform a blanket avoidance distance of 4 meters. We augment Safe-Lang with our VLM goal identification since their approach also does not identify new fallback strategies and goals when the nominal goal is infeasible.

To ensure a fair comparison of `FORTRESS` and the baselines, we used the same planning and path tracking methods (i.e. RRT+LQR) for generating controls for the fallback plans.

D.3 Discussion on Results

The results of all the approaches are shown in Fig. 6 of the main paper. Since AESOP does not consider semantic safety, it generates plans that enter into semantically unsafe regions such as near a firetruck with traffic cones or a rooftop party with people. Furthermore, while Safe-Lang's fallback approach avoids objects identified by the object detector, it is unable to distinguish between semantically safe and unsafe concepts and therefore incurs a high no-plan rate (i.e. no safe plan could be found to reach any fallback strategy goal). Our approach can reliably generate safe plans successfully with more than a 90% success rate for the strategy of "landing on building roofs." The situations when `FORTRESS` cannot find a safe plan to implement are because all building roofs are occupied by unsafe concepts.

E Replanning with Dynamic Concepts

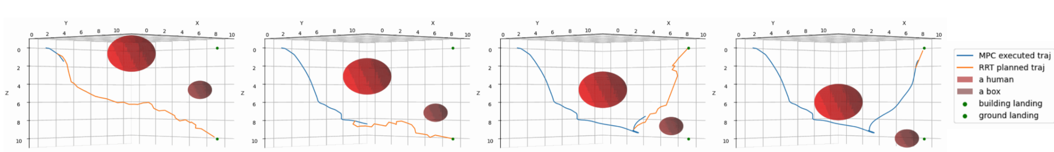


Figure 12: Example demonstrating (re)planning of safe fallback plans with moving objects. `FORTRESS` originally produced a plan to the first goal point that implemented the strategy of landing on the ground by avoiding the human (skydiver) and box. However as the human and box descended, the original fallback strategy was physically and semantically infeasible so it replanned a path to implement the next strategy of landing on a building.

We demonstrate an example of how `FORTRESS` changes plans and implements a new strategy when circumstances evolve to ensure both semantic and physical safety of the fallback. Specifically, we employ a robot with the dynamics of a double integrator along 3 dimensions. The environment consists of a human skydiving and a falling box, and we have two strategies: land on the ground or land on a building. In this simple example, FORTRESS understands the semantically unsafe regions in the state space (i.e. skydiving human is unsafe to be near while the box is relatively safe) as it is changing and in response adjusts its fallback plans or even implements a wholly new strategy. In this setting, we use RRT to replan every 10 steps and MPC for path tracking.

F Details on Hardware

F.1 ANYmal Robot Hardware Experiments

We discuss some details on the setup for the experiments conducted on the ANYmal hardware (Generation D) for testing the accuracy performance of FORTRESS's semantic safety cost function in identifying potential OOD failures. The setting is a room that is actively under construction. We first collect some safe, nominal data with represent concepts that are within the operational capabilities of the ANYmal. Specifically, we navigate the robot around the room with objects like boxes, paper scraps, ladders, paint cans, and people, which we detect using the OWLv2 [36] open-vocabulary object detector. We extract around failure modes from querying Claude 3.7 with an image of the environment, such as Sharp Objects, Unauthorized Access, Chemical Spill, Unstable Region, Worker Injury, Entanglement, and Slippery region. Then, after calibrating our semantic safety cost functions using the Qwen2 text embedding model, we deploy the robot again but this time we place unsafe objects in the room. As seen in Fig. 13, this includes a person standing on a ladder (which is unsafe since the ANYmal may cause the person to fall down), cables (which can cause the robot to get entangled and trip), caution tape, and a blue toxic spill. FORTRESS can detect these semantically unsafe concepts even though no similar failures existed in the training dataset. Notice how "person" and "ladder" are individually detected as semantically safe but a "person on a ladder" triggers "Worker Injury" failure – entering into this unsafe region can result in destabilizing the person on the ladder and potentially injure them. Full video at https://www.youtube.com/watch?v=xU-egPQjkFo.

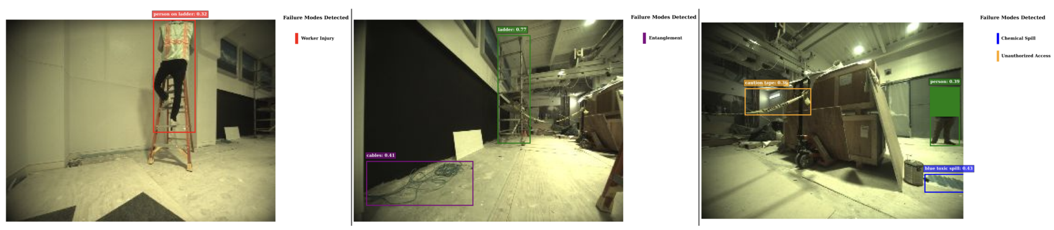


Figure 13: Examples of OOD failures detected by FORTRESS for deployment of ANYmal hardware in a room under construction. The green boxes indicate semantically safe concepts for the robot such as a ladder or a person. The other colors show potential hazards: in the image, the boxes are labeled with what objects are detected and on the legend we list their corresponding failure modes that have been identified by the semantic safety cost functions.

F.2 Quadrotor Drone Hardware Experiments

We explain our experiments for implementing FORTRESS on quadrotor drone hardware. We equip our quadrotor with a Jetson nano for computation and an Intel Realsense D435 camera to stream RGBD data. We use the Optitrack motion capture system for localization. We deploy the drone in an environment with buildings and task FORTRESS with producing a response of implementing the fallback strategy of landing on building rooftops. As seen in Fig. 1 of the main paper, the rooftops consist of various safe and unsafe concepts. Specifically, for unsafe regions, we have a building with a ladder and caution tape (depicting a construction), one on fire, and a rooftop parking lot with two vehicles. Additionally, there are two buildings that are relatively safe by themselves such as one with nothing on it and one with a garden rooftop with grass and trees. As seen in the demo video at https://www.youtube.com/watch?v=a0XZgwoNLos, the drone lands on the garden rooftop. FORTRESS successfully identifies that the garden rooftop provides a safe spot to implement the strategy of landing on a building while the other buildings are either centers of OOD failures or are very close to them.

To obtain goal points, we query Molmo for rooftop landing locations as follows:

```
Query: point to middle of building rooftop.

Response: <point x="67.1" y="71.1" alt="middle of building rooftop">middle of
     building rooftop</point>
```

This point corresponds to

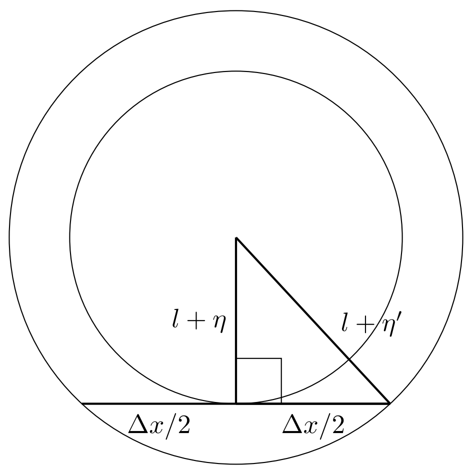
Similar to the CARLA setting, we identify the semantically unsafe regions in the state space by employing the OpenAI text embedding model calibrated with cosine similarity. We perform reach avoid planning to enter into a 0.5-meter radius around the goals identified by Molmo while avoiding the unsafe regions. We use the OWLv2 [36] open-vocabulary object detector to identify the concepts on the building roofs and their locations. We make l_c as 0.1 meters and l_ϕ for all ϕ as 1.5 meters. We also validate the safety of our fallback goals by filtering out goal points g when $\exists\phi \in \Phi$ where $\hat{\theta}_\phi(g) > 0$. $\hat{\theta}_\phi$ is the same as θ_ϕ except its distance parameter l_ϕ is inflated by 0.5 meters. We employ an RRT planner with an incremental step size of 0.1 meters. We track the plan with interpolation and publish the interpolated waypoints on ROS2 to the PX4 controller at 100Hz.

G Theorem on Safely and Successfully Solving Optimization (3)

Theorem. *Let $\eta > 0$ be the worst case error bound of the path tracking control algorithm, and let $\hat{\theta}_h, h \in \Phi \cup \{c\}$ be the same formulation as θ_h except the corresponding distance thresholds are inflated to $\hat{l}_h = l_h + \eta'$ where $\eta' > \eta$, and let $\hat{\mathcal{X}} \subseteq \mathcal{X}$ be the set of states where $\max_{h\in\Phi\cup\{c\}} \hat{\theta}_h(x) \leq 0$. Furthermore, suppose a motion planner is employed in state space $\hat{\mathcal{X}}$ to start from point b and reach control-invariant region $\mathcal{B}_\rho(g)$, where $\rho > \eta$, using incremental step planner size of $\Delta x < \min(\rho - \eta, \min_{h\in\Phi\cup\{c\}} 2\sqrt{(\eta' - \eta)^2 + 2(l_h + \eta)(\eta' - \eta)})$ produces a trajectory $\hat{\tau} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k]$. Then when the path tracking control algorithm is used to follow $\hat{\tau}$, it produces a trajectory $\tau^* = [x_1^*, x_2^*, \ldots, x_k^*, \ldots]$ that safely and successfully executes a fallback strategy (i.e. solves optimization of (3) with $\Theta^* \leq 0$).*

Proof. The first and third constraints of (3) are satisfied trivially with the planner and path tracking control algorithm. The core proof of the theorem therefore is about guaranteeing that planning and path tracking will satisfy the optimization and second constraint of (3).

For the optimization, the worst case to consider is when two consecutive waypoints in the safe/valid space $\hat{\mathcal{X}}$ are on the border of the sphere created by `nearby` for cost/failure $h \in \Phi \cup \{c\}$ using the inflated radii – this is the closest that the waypoints can be to the epicenter of the unsafe region. In this case, the two waypoints are a distance $l_h + \eta'$ from the point/concept/obstacle. The length of the line segment between the two waypoints is Δx which is the planning algorithm's step size parameter. The robot, and therefore the line segment, should be at most $l_h + \eta$ close to the failure point/concept/obstacle since path tracking of the line segment has worst case error of η. The line segment in the worst case is tangent to the sphere of radius $l_h + \eta$ and its midpoint is on a sphere with the same center but with radius $l_h + \eta$. This creates a right angle triangle from which we obtain the upper bound of $(\Delta x)/2$ which is the distance from a waypoint to the midpoint of the line segment: $(\Delta x/2)^2 + (l_h + \eta')^2 < (l_h + \eta)^2$. Intuitively larger Δx would mean less granular planning and therefore more error. This inequality can be rearranged to get $\Delta x < 2\sqrt{(\eta' - \eta)^2 + 2(l_h + \eta)(\eta' - \eta)}$. See the below diagram for a visualization of the geometry. This constraint ensures that if a planner plans a path avoiding any state with positive values from the inflated length cost functions $\hat{\theta}$, then the path tracking algorithm will generate a trajectory that avoids states with positive values from the original length cost functions θ.



Furthermore, we need another constraint to ensure that the path tracking trajectory of the plan reaches and remains in the goal. Since we assume region $\mathcal{B}_\rho(g)$ is control invariant, meaning that once we enter it, there are controls that ensure the robot remains within the region, we only need to guarantee the plan definitively enters the region. Once again, since the path tracking error is η, we simply ensure that the planner's incremental distance does not miss the sphere region ρ around goal g. The planner will reach a point within Δx of the goal g, and the path tracker will be at most η from that point. We can ensure the path tracker enters the goal region using the constraint that $\eta + \Delta x < \rho$. Bringing together all the constraints and considering the worst case failure, we get the upper bound of $\Delta x < \min(\rho - \eta, \min_{h \in \Phi \cup \{c\}} 2\sqrt{(\eta' - \eta)^2 + 2(l_h + \eta)(\eta' - \eta)})$. Ultimately, this bound of the planning step size ensures that if the planning algorithm is employed in modified state space $\hat{\mathcal{X}}$ and reaches the goal, then the path tracking algorithm will be able to produce a safe trajectory (i.e. avoids the semantically unsafe regions with OOD failures) and implements the fallback strategy by reaching the fallback goal region. $\square$