AnySafe: Adapting Latent Safety Filters at Runtime via Safety Constraint Parameterization in the Latent Space

Sankalp Agrawal*[1], Junwon Seo*[2], Kensuke Nakamura[2], Ran Tian[3,4], Andrea Bajcsy[2]

***Abstract*— Recent works have shown that foundational safe control methods, such as Hamilton–Jacobi (HJ) reachability analysis, can be applied in the latent space of world models. While this enables the synthesis of latent safety filters for hard-to-model vision-based tasks, they assume that the safety constraint is known a priori and remains fixed during deployment, limiting the safety filter's adaptability across scenarios. To address this, we propose *constraint-parameterized latent safety filters* that can adapt to user-specified safety constraints at runtime. Our key idea is to define safety constraints by conditioning on an encoding of an image that represents a constraint, using a latent-space similarity measure. The notion of similarity to failure is aligned in a principled way through conformal calibration, which controls how closely the system may approach the constraint representation. The parameterized safety filter is trained entirely within the world model's imagination, treating any image seen by the model as a potential test-time constraint, thereby enabling runtime adaptation to arbitrary safety constraints. In simulation and hardware experiments on vision-based control tasks with a Franka manipulator, we show that our method adapts at runtime by conditioning on the encoding of user-specified constraint images, without sacrificing performance. Video results can be found on the project website.**

I. Introduction

World models offer a promising paradigm for generalizing robot control to hard-to-simulate physical tasks by learning compact latent state spaces and dynamics directly from high-dimensional observations [1]–[4]. Recent works have demonstrated that foundational safe control methods, such as Hamilton–Jacobi (HJ) reachability analysis [5], [6], can be applied directly in a world model's latent space, enabling safety analysis directly from high-dimensional sensor inputs. By computing robot policies that anticipate and avoid future failures within the world model's imagination, these *latent safety filters* can proactively steer robots away from hard-to-model constraints, such as spilling the contents of deformable bags [7] or toppling complex rigid-body structures [8].

However, most safe control frameworks assume that the state constraints that robots should avoid are determined *a priori* and remain fixed during deployment [6], [9]. In practice, this assumption is overly restrictive: at deployment time, a robot may need to adapt its notion of what is a safety constraint based on changing environments or end-user requirements. For example, consider the robot manipulator in Fig. 1 that must sweep clutter from a table. In one scenario, it needs to avoid sweeping objects in a particular region (top row), but later it may be tasked with intentionally collecting objects into that same region while avoiding a different one (bottom row). This raises the central question of our work:

How can latent safety filters adapt to safety constraints specified at test-time?

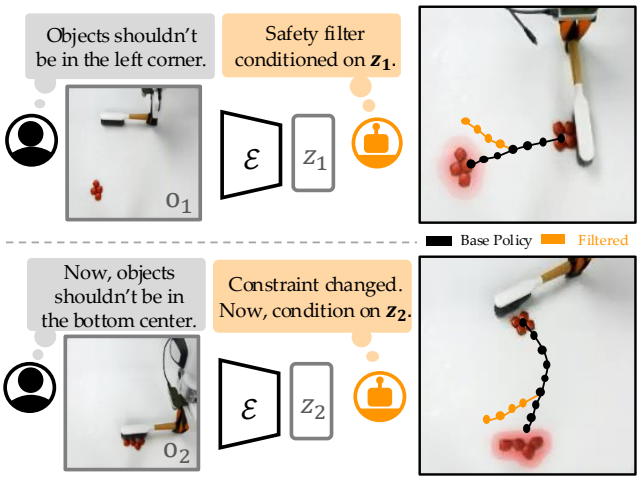

Fig. 1: **Constraint-Parameterized Latent Safety Filter.** The latent safety filter adapts its safety constraint by conditioning on an encoding of an image specified by the user as a failure.

In this work, we design *constraint-parameterized latent safety filters* (called *AnySafe*). The core challenge with parameterizing safety constraints in the latent space is that, unlike in hand-designed state spaces, the structure needed to represent and optimize against a suite of safety constraints does not naturally emerge. In hand-designed state spaces, one can design a low-dimensional parameterization of the constraint set (e.g., a circle by its center and radius) alongside a dense distance measure for guiding policy optimization (e.g., signed distance to the constraint set); this allows for the safety filter to be effectively computed for all possible constraint variations. In latent spaces, by contrast, constraints are typically only implicitly defined by classifiers on the latent states [7], [8] which do not admit a continuous parameterization to represent diverse safety constraints nor yield a notion of proximity from a state to such constraints.

We propose three key ingredients that enable constraint-parameterization in latent safety filters. First, we specify safety constraints via a *similarity measure* between the embedding of a constraint image and the robot's current latent state; this provides a dense signal of how close the policy is to failure. Then, we *calibrate* the resulting constraint set with conformal prediction [10], [11] to align with an end-user's semantic notion of failure. Lastly, we train the safety

*These authors contributed equally to this work. [1]The Ohio State University. agrawal.268@buckeyemail.osu.edu. [2]Carnegie Mellon University. {junwonse, kensuken, abajcsy}@andrew.cmu.edu. [3]UC Berkeley. rantian@berkeley.edu. [4]NVIDIA Research.

filter by treating *any* image in the world model dataset as a possible test-time safety constraint. At runtime, we adapt the latent safety filter by conditioning it on an encoding of a user-specified constraint image, thereby adapting it to the runtime safety specification.

We evaluate our framework on vision-based safe-control tasks, including a simulated vehicle collision-avoidance domain and real-world object sweeping with a Franka manipulator. Our results highlight four key findings: (1) by parameterizing the safety filter with constraint representations, *AnySafe* can adapt to arbitrary constraints provided as images; (2) this adaptability does not come at the cost of performance, as for a given constraint, the parameterized safety filter achieves performance comparable to a specialized filter trained solely on that constraint; (3) *AnySafe* generalizes to constraints beyond those that specialized safety filters can model; and (4) since *AnySafe* learns from continuous latent similarity signals, conformal calibration allows us to control how conservatively the robot avoids specified constraints by adjusting the effective size of the failure set.

II. Related Work

Safety Filtering in Robotics. Safety filtering is a control-theoretic approach for preventing robotic systems from entering unsafe states [6], [9]. Foundational methods enforce safety by correcting the robot's base control policy using Control Barrier Functions [12], Hamilton–Jacobi (HJ) reachability analysis [5], [13], or model predictive shielding [14]. Since the key challenge of these methods is the tractable synthesis of a valid safety value (or barrier) function that encodes a set of safe states, recent methods have leveraged self-supervised learning [15] and reinforcement learning (RL) [16], [17] to scale safety value functions and controllers to high-dimensional nonlinear systems. More recently, latent dynamics models [2], [4] have been used to compute these safety filters in learned latent spaces [7], [8], enabling safe control directly from high-dimensional image observations.

However, most existing approaches assume that the safety constraint is fixed [6], [9], [18], and thus cannot generalize beyond a predefined safety specification. With hand-designed state spaces and dynamics models, safety constraints can be updated online by precomputing families of reachable sets parameterized by environmental or system factors [19], or by incrementally updating safety specifications [20], [21]. Recent works introduce observation-conditioned safety filters, which adapt a safety value function for collision-avoidance with current sensor observations [22]–[24]. In contrast, our method operates in the world model's learned latent state space and adapts the safety filter to runtime user-specified constraints, provided as RGB images of undesirable states.

Learning Conditioned Control Policies. Conditioned (or parameterized) policies are a well-established way to enable control policies to adapt to arbitrary objectives. Within the safe control literature, this conditioning has been applied to constraint thresholds [25], low-dimensional environmental parameters [19], or high-dimensional LiDAR observations [23], [24], all of which adapt the robot's safety specifications at runtime. More broadly, goal-conditioned reinforcement learning (GCRL) trains agents to achieve diverse user-defined goals by conditioning policies and value functions on goal representations, enabling zero-shot deployment with runtime goal images without further training [26], [27]. GCRL relabels past experiences in a self-supervised manner and learn value functions that capture similarities between states [28], [29]. Instead of goal-conditioning, we *constraint*-condition the HJ reachability problem with an RL-based solver [16] in the imagination of a world model, self-labeling past trajectories as potential safety constraints to make a latent safety filter avoid diverse runtime safety constraints.

III. Background: Latent Safety Filter

We briefly introduce latent safety filters [7], which serve as the foundation of our method. A latent safety filter consists of two components: a safety value function $V^{\textcolor{black}{\blacklozenge}} : \mathcal{Z} \to \mathbb{R}$, which measures how close the robot is to inevitable failures, and a safety-preserving policy $\pi^{\textcolor{black}{\blacklozenge}} : \mathcal{Z} \to \mathcal{A}$, which steers the robot away from failure. In this work, we compute these models via Hamilton–Jacobi (HJ) reachability analysis [5]. The key innovation is that the value function and policy are optimized within the learned latent state representation ($z \in \mathcal{Z}$) of a world model learned directly from RGB observations.

Latent States, Dynamics, and Constraints. World models [1], [2], [4] offer a paradigm for learning difficult-to-simulate dynamical systems models directly from raw sensor observations by jointly inferring a lower-dimensional latent state $z \in \mathcal{Z}$ and its associated dynamics f_z. These models are trained with an offline dataset of robot–environment interactions, $\mathcal{D}_{\text{train}} := \{(o_t, a_t, l_t)_{t=1}^{T}\}_{i=1}^{N_{\text{train}}}$. Here, each trajectory consists of high-dimensional observations $o \in \mathcal{O}$ (e.g., proprioception and RGB images), robot actions $a \in \mathcal{A}$, and failure labels $l \in \{-1, 1\}$ that indicate the presence of visible failures from an observation. (e.g., the contents of a bag being spilled [7], or objects toppling into a sensitive region [8]). Note that the failure is modeled as a binary classification, where the notion of failure is assumed to be fixed at training time and remains unchanged at deployment.

The world model consists of an encoder $\mathcal{E}$ that maps an observation and a prior latent state $\hat{z} \in \mathcal{Z}$ into the posterior latent representation $z \in \mathcal{Z}$, and a latent dynamics model f_z that predicts the next latent state conditioned on an action.

$$\begin{aligned} \text{Encoder: } & z_t \sim \mathcal{E}(z_t \mid \hat{z}_t, o_t) \\ \text{Latent Dynamics: } & \hat{z}_t \sim f_z(\hat{z}_t \mid z_{t-1}, a_{t-1}) \\ \text{Failure Classifier: } & l_t = \ell_z(z_t), \end{aligned} \tag{1}$$

where $\ell_z(z_t)$ models the safety constraints as a classifier; it returns whether the z is in failure or not. This formulation describes a wide range of world models [1]–[4], [30].

Latent Safety Filter with a *Fixed* Failure Set. With a binary classifier ℓ_z learned on the latent space, a fixed safety constraint can be represented as a *failure set* $\mathcal{F} := \{z : \ell_z(z) \leq 0\} \subset \mathcal{Z}$ encoded via the zero-sublevel set of the failure margin function. A latent safety filter for

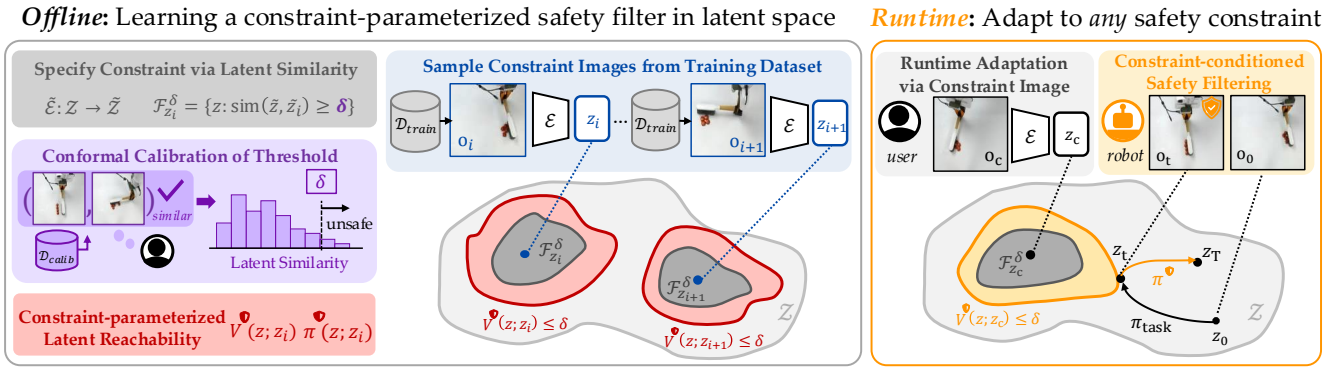

Fig. 2: **Framework: Constraint-Parameterized Latent Safety Filter.** *Left*: The constraint-parameterized latent safety filter is trained by sampling constraint images from the WM training dataset, treating any image as a possible test-time safety constraint. Safety constraints are specified using a latent-space similarity measure, with a calibrated threshold that defines the size of the failure set. *Right*: At runtime, the safety filter adapts to any safety constraint with a user-specified image.

the *fixed* failure set can then be constructed by performing Hamilton–Jacobi (HJ) reachability analysis [5], [9] in the latent space. Using the latent imagination of a pretrained world model as the environment dynamics, the safety filter is learned by solving the fixed-point safety Bellman equation [1]:

$$\begin{aligned} V^{\blacktriangledown}(z_t) &= (1-\gamma)\ell_z(z_t) \\ &\quad + \gamma \min\left\{\ell_z(z_t), \max_{a_t\in\mathcal{A}} \mathbb{E}_{\hat{z}_{t+1}\sim f_z(\cdot|z_t,a_t)} V^{\blacktriangledown}\big(\hat{z}_{t+1}\big)\right\}, \\ \pi^{\blacktriangledown}(z_t) &= \arg\max_{a\in\mathcal{A}} \mathbb{E}_{\hat{z}_{t+1}\sim f_z(\cdot|z_t,a)} V^{\blacktriangledown}\big(\hat{z}_{t+1}\big). \end{aligned} \tag{2}$$

where $\gamma \in [0,1)$ is a time discounting factor ensuring a contraction mapping [16]. We note that, unlike typical RL, which maximizes the cumulative reward, this optimization performs a *min-over-time* to remember if the trajectory ever entered the failure set. Thus, $V^{\blacktriangledown}(z) < 0$ indicates that the robot is doomed to enter the failure set if starting from z, whereas $V^{\blacktriangledown}(z) \geq 0$ implies there exists a safety-preserving action (e.g., provided by $\pi^{\blacktriangledown}$) that can prevent future failure.

At runtime, this learned safety filter can be deployed to safeguard any arbitrary task policy π^{task} with respect to a *fixed* safety constraint. Given the current latent representation z (embedded from the current observations) and the action proposed by the task policy, the safety filter evaluates the sign of $V^{\blacktriangledown}$ of the next state $z' = f_z(z, \pi^{\text{task}})$ as a safety monitor. Based on this evaluation, the filter either allows the action from π^{task} to proceed or overrides it with the fallback policy: $a^{\text{exec}} = \mathbb{1}_{\{V^{\blacktriangledown}(z)>0\}} \cdot \pi^{\text{task}} + \mathbb{1}_{\{V^{\blacktriangledown}(z)\leq 0\}} \cdot \pi^{\blacktriangledown}(z)$.

IV. Constraint-Parameterized Latent Safety Filter

In this section, we generalize the *fixed* latent safety filter by *parameterizing* it on a constraint representation, z_c, obtained by encoding images of constraints that an end-user cares to prevent, yielding $V^{\blacktriangledown}(z; z_c)$ and $\pi^{\blacktriangledown}(z; z_c)$. We describe three key ingredients–a latent similarity measure, calibration, and training–that enable constraint parameterization of latent safety filters, thereby making them adaptable at test time. The overall framework is described in Fig. 2

Learning a Similarity Measure Over Failures. Recall how the fixed latent safety filter requires a classifier for a specific constraint. To generalize beyond a fixed classifier, we propose utilizing a dense similarity measure $\ell_z(z; z_c)$ in the latent space, which represents how close z is to the specified representation z_c, thereby enabling any constraint embedding to be used as a possible safety constraint. For example, a natural choice could be cosine similarity between any world model state, z, and the constraint embedding, z_c. However, as we find in Sec. V-B and Sec. VI-B, using the raw embeddings from the world model to compute this similarity measure is often not sufficiently informative nor aligned with an end-user's understanding of similarity. To address this, we train a projector $\tilde{\mathcal{E}} : \mathcal{Z} \to \tilde{\mathcal{Z}}$ on top of the latent space, which maps the world model's representation $z \in \mathcal{Z}$ into a failure-relevant latent representation $\tilde{z} \in \tilde{\mathcal{Z}}$:

$$\text{Failure Projector: } \tilde{z} = \tilde{\mathcal{E}}(z), \tag{3}$$

$$\text{Latent Failure Margin: } \tilde{\ell}_z(z; z_c) := -\operatorname{sim}(\tilde{z}, \tilde{z}_c). \tag{4}$$

This ensures that the similarity $\operatorname{sim}(\tilde{z}, \tilde{z}_c)$ provides a better aligned measure of meaningful distances in the latent space, enabling safety specifications to be represented by a dense latent failure margin function.

In general, this projector can be trained in various ways to align failure-relevant features, such as supervised metric learning [31], representation learning [28], or alignment [32]. In this work, we adopt a simple supervised learning so that similarity in the projected latent space reflects proximity to failure-relevant features[2].

Ultimately, safety constraints depend on an embedding z_c and are modeled via the δ-sub-level set (left, Fig. 2):

$$\mathcal{F}^{\delta}_{z_c} := \{z : \tilde{\ell}_z(z; z_c) \leq \delta\} = \{z : -\operatorname{sim}(\tilde{z}, \tilde{z}_c) \leq \delta\}, \tag{5}$$

where $\operatorname{sim}(\cdot,\cdot)$ is the cosine similarity between two vectors and δ is a threshold for how similar a latent state must be to the constraint embedding to be also considered a failure.

[1]This includes an expectation over transitions for stochastic dynamics (e.g., RSSM [1]) but can be removed for deterministic ones (e.g., [4]).

[2]A thorough comparison or the design of novel training objectives for metric-space similarity measures is beyond the scope of this work.

Calibrating the Similarity-based Latent Failure Set. Recall that the latent failure set defined in Eqn. (5) depends on the threshold δ, which determines its effective size.

We calibrate the threshold δ that defines the size of the failure set to align with a user's notion of failure set size, thereby controlling how closely the robot is permitted to approach the constraint representation, z_c (middle left, Fig. 2).

Specifically, we employ Conformal Prediction (CP), a distribution-free statistical method [10], [11]. Using a held-out calibration dataset that reflects the user's understanding of failures, we aim to provide a recall guarantee for detecting failures conditioned on a representation [33]. The calibration dataset consists of latent pairs with ground-truth labels, $\mathcal{D}_{\text{calib}} := \big\{\big((z_j, z'_j), y_j\big)\big\}_{j=1}^{N_{\text{calib}}}$, where $y_j \in \{0, 1\}$ indicates whether the pair of representations are similar. We then employ class-conditioned CP [8], [34] to guarantee the recall of failures at a user-specified confidence level $\alpha \in [0, 1]$:

$$\mathbb{P}\left(\tilde{\ell}_z(z_{\text{test}}; z'_{\text{test}}) \leq \delta \mid y_{\text{test}} = 1\right) \geq 1 - \alpha. \tag{6}$$

This is implemented by only using positive latent pairs from the same class (i.e., $y_i = 1$) and defining the conformal nonconformity score as $-\operatorname{sim}(\tilde{z}_j, \tilde{z}'_j)$. The threshold δ is then chosen as the $(1-\alpha)$-quantile of the set $\{-\operatorname{sim}(\tilde{z}_j, \tilde{z}'_j)\}_{i=1}^{N}$, obtained by selecting the $\lceil (1-\alpha)(N+1) \rceil$-th smallest value, where N denotes the number of positive pairs in the calibration dataset. This class-conditioned CP guarantees the recall of failure [34], $\mathbb{P}\Big(z_{\text{test}} \in \mathcal{F}^{\delta}_{z'_{\text{test}}} \mid y_{\text{test}} = 1\Big) \geq 1 - \alpha$.

Training Constraint-Parameterized Latent Safety Filter. Finally, we train a constraint-parameterized latent safety filter entirely within the latent imagination of the world model. During training, we treat *any* observation from the world model dataset as a candidate failure we could see at test-time; we randomly sample observations, encode them into z_i, and solve the constraint-parameterized fixed-point safety Bellman equation conditioned on them (top middle, Fig. 2):

$$\begin{aligned} V^{\boldsymbol{\Phi}}(z_t; z_i) &= (1-\gamma)\,\tilde{\ell}_z(\tilde{z}, \tilde{z}_i) \\ &\quad + \gamma \min\Big\{\tilde{\ell}_z(\tilde{z}, \tilde{z}_i), \max_{a_t \in \mathcal{A}} \mathbb{E}_{\hat{z}_{t+1} \sim f_z(\cdot \mid z_t, a_t)} V^{\boldsymbol{\Phi}}\big(\hat{z}_{t+1}; z_i\big)\Big\}, \\ \pi^{\boldsymbol{\Phi}}(z; z_i) &= \arg\max_{a \in \mathcal{A}} \mathbb{E}_{\hat{z}_{t+1} \sim f_z(\cdot \mid z_t, a)} V^{\boldsymbol{\Phi}}(\hat{z}_{t+1}; z_i), \end{aligned} \tag{7}$$

where $\gamma \in [0, 1)$ is a time discounting factor similar to (2).

Intuitively, the safety value function $V^{\boldsymbol{\Phi}}(z_t; z_i)$ measures how close the robot, starting from $\tilde{z}_t$, comes to the failure representation $\tilde{z}_i$ in $\tilde{\mathcal{Z}}$ despite its best-effort safety policy $\pi^{\boldsymbol{\Phi}}(z_t; z_i)$ to minimize similarity with that representation. As z_i is sampled randomly from the training dataset, the safety filter can treat either one of these or a newly interpolated representation as a potential failure at test time.

Runtime Constraint-Parameterized Safety Filtering. At runtime, a user provides a constraint image o_c, which is encoded into a constraint representation $z_c = \mathcal{E}(o_c)$ and used to adapt the safety filter to the specified constraint (right, Fig. 2). The safety value function evaluates whether the action proposed by the task policy would inevitably enter the latent-space failure set defined by the calibrated threshold, and intervenes with the safety-preserving policy if necessary:

$$a^{\text{exec}}_{z_c} := \begin{cases} \pi^{\text{task}}, & \text{if } V^{\boldsymbol{\Phi}}\big(f_z(z, \pi^{\text{task}}); z_c\big) > \delta, \\ \pi^{\boldsymbol{\Phi}}(z; z_c), & \text{otherwise.} \end{cases} \tag{8}$$

Intuitively, the safety filter ensures that the observations–and the corresponding latent states—are *dissimilar* enough (by a δ margin) to the conditioned runtime safety constraint.

Note that in (8), we apply the calibrated threshold directly to the value function $V^{\boldsymbol{\Phi}}$, while the guarantee in (6) only ensures that the similarity measure $\tilde{\ell}_z$ is calibrated. Approximate value function solvers (e.g., RL [16]) can induce errors, but directly calibrating the value function requires stronger assumptions about access to ground-truth unsafe set labels [33]. We thus apply the similarity-calibrated threshold δ directly to the value function at runtime, assuming that approximation errors are marginal. Importantly, this enables calibration to be performed post hoc, without the need to retrain the safety filter for different calibration results.[3]

V. Simulation Results

We first conduct experiments with a low-dimensional, benchmark collision-avoidance navigation task where privileged information about the state, dynamics, safe set, and safety controller is available. We focus on the following questions: (i) Can *AnySafe* adapt to diverse test-time safety constraints? (ii) Does calibration of *AnySafe* correctly align the safety filter with the user's understanding of failure?

A. Experimental Setup

Privileged Dynamics: 3D Dubins' Car. Let discrete-time dynamics with privileged state be $s = [p^x, p^y, \theta]$ $s_{t+1} = s_t + \Delta t\,[v\cos(\theta_t),\, v\sin(\theta_t),\, a_t]$, where the robot's action is continuous angular velocity $a_t \in \mathcal{A} = [-a_{\max}, a_{\max}]$ with $a_{\max} = 1.25$ rad/s, while the longitudinal velocity is fixed $v = 1\,\text{m/s}$. The time discretization is $\Delta t = 0.05\,\text{s}$.

World Model. We adopt Dreamer [2] with the latent dynamics model of the Recurrent State Space Model (RSSM) [1] with continuous latents. The world model is trained using an offline dataset of $N_{\text{train}} = 4,000$ observation–action trajectories collected without failure labels. Each observation is a $3 \times 128 \times 128$ image of the environment and vehicle (Fig. 3), while actions are randomly sampled during trajectory generation. Each trajectory terminates after $T = 100$ timesteps or earlier if the ground-truth x or y coordinate leaves the environment bounds of $[-1.5\,\text{m}, 1.5\,\text{m}]$.

Failure Projector & Calibration. We train the failure projector, implemented as a 2-layer MLP, sampling from the world model training dataset to construct $\{(z_{i1}, z_{i2}, s_i)\}$. The ground-truth similarity score is defined as $s_i = \max[1 - \frac{1}{\sqrt{2}}\{(p^x_{i1} - p^x_{i2})^2 + (p^y_{i1} - p^y_{i2})^2\}, -1.0]$,

[3] In the Appendix, we prove that with a perfect value function solver, applying the calibrated threshold δ directly to the value function yields the same unsafe set as solving a threshold-dependent value function.

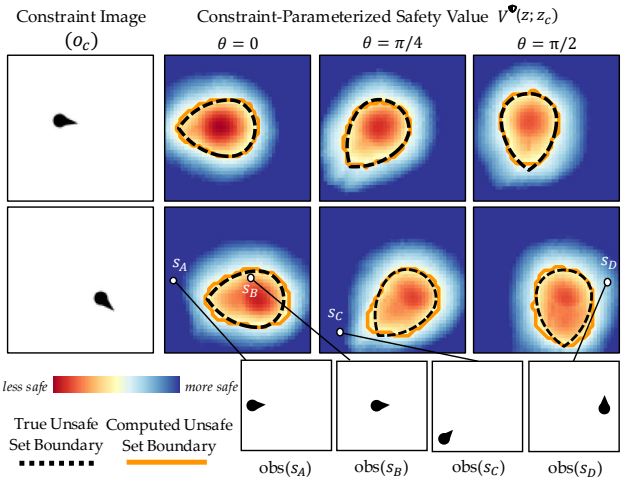

Fig. 3: **Dubins' Car Qualitative Results.** We visualize the safety value function parameterized on two different constraints, shown at heading slices $\theta \in \{0, \pi/4, \pi/2\}$.

Method	f_z	ℓ_z	$\tilde{\ell}_z$	FPR↓	Rec.↑	Pre.↑	F_1 ↑	B.Acc.↑	Safe Rate↑
Privileged-Fix	✗	✗	✗	0.030	0.982	0.991	0.987	0.976	0.960
Privileged-Any	✗	✗	✗	0.050	0.977	0.986	0.982	0.964	0.988
Latent-Fix	✓	✓	✗	0.041	0.971	0.988	0.980	0.965	0.908
Latent-Fix-Cont	✓	✗	✓	0.080	0.972	0.977	0.975	0.946	0.904
Anysafe (w.o. Proj)	✓	✗	✗	0.480	0.991	0.881	0.933	0.755	0.836
AnySafe	✓	✗	✓	0.082	0.966	0.977	0.971	0.942	0.924

TABLE I: **Comparison of Safety Filter in Dubins' Car.** **AnySafe** accurately adapts to different safety constraints while maintaining safety performance comparable to a filter trained for a single safety constraint.

based on the ground-truth robot positions. The failure projector is then trained with the mean-square error (MSE) loss $\tilde{\mathcal{L}}_i = \left(\text{sim}(\tilde{z}_{i1}, \tilde{z}_{i2}) - s_i\right)^2$. The calibration dataset consists of $N_{\text{calib}} = 3,000$ held-out images labeled with robot positions. Pairs within $\epsilon = 0.5\,\text{m}$ are labeled positive, and used to compute the threshold δ with $\alpha = 0.005$.

Latent Safety Filter Setup. We use DDPG [35] as our solver for computing the latent safety filter. We randomly sample initial robot observations and the failure observations from the world model training data and encode both into the latent state. For runtime filtering (8), we add a small margin (0.1) to the calibrated threshold.

Evaluation & Metrics. In this low-dimensional example, we have access to the ground-truth dynamics and can compute a high-quality "ground-truth" safety value function using grid-based methods [36]. This enables us to directly evaluate the safety monitor $V^{\blacklozenge}$'s classification accuracy across all three state dimensions. We evaluate the quality of the safety value function conditioned on 50 different constraint images. We measure the classification accuracy across all three state dimensions. To assess the performance of the fallback policy $\pi^{\blacklozenge}$, we roll out the learned policies from 250 ground-truth safe initial states and measure the safety rate by checking whether the safety policy ensures the robot never enters the ground-truth failure set.

Method	FPR↓	Recall ↑	Pre.↑	F_1 ↑	B.Acc.↑	Safe Rate↑
$\mathcal{Z} \times \mathcal{Z}$	0.082	0.966	0.977	0.971	0.942	0.924
$\mathcal{Z} \times \mathcal{P}$	0.221	0.912	0.946	0.929	0.845	0.904
$\mathcal{Z} \times \tilde{\mathcal{Z}}$	0.064	0.933	0.981	0.957	0.935	0.828
$\tilde{\mathcal{Z}} \times \tilde{\mathcal{Z}}$	0.113	0.910	0.967	0.938	0.899	0.504

TABLE II: **Ablation: Parameterization Strategies.** The constraint-parameterized latent safety filter shows the best performance when parameterized with latent representations, with constraints randomly sampled from the training dataset.

B. Can AnySafe Adapt to Diverse Safety Constraints?

We first study whether *AnySafe* can be parameterized on diverse constraints without sacrificing safety performance.

Baselines. We compare *AnySafe* against the following baselines. *Privileged* baselines train safety filters using RL with privileged ground-truth states [16], with a single circular failure set of radius $\epsilon = 0.5$ centered at $(0, 0)$ (*Privileged-Fix*), or by parameterizing with the position of the constraint in a privileged state-space (*Privileged-Any*). *Latent* baselines use the world model, raw image observations as input, and center the failure set in the middle of the plane. *Latent-Fix* [7] employs a single failure classifier while *Latent-Fix-Cont* uses a continuous similarity signal $\tilde{\ell}_z$ to the representation of an image with the vehicle at the center. Lastly, *Anysafe (w.o. Proj)* solves (7) using raw latent similarities without the failure projector. For baselines restricted to a single safety constraint (the **-Fix* variants), we report performance only against the ground-truth center failure region.

Results. Qualitative results in Fig. 3 show that **AnySafe** adapts to diverse failure sets when conditioned on different constraint representations, constructing accurate constraint-parameterized unsafe sets. Quantitative results in Table I first confirm that latent safety filters achieve performance comparable to safety filters trained with privileged states, consistent with [7]. In both privileged-state (*Privileged-Any*) and latent-state (**AnySafe**) settings, conditioning on failure states during training enables the safety filter to generalize to arbitrary failure sets. Furthermore, **AnySafe** achieves safety monitor performance and safe rates comparable to those of a filter trained for a single fixed failure set (*Latent-Fix*), while maintaining the ability to adapt to arbitrary safety constraints. Moreover, *AnySafe* without the failure projector (*Anysafe (w.o. Proj)*) performs significantly worse with low accuracy and safe rate. This confirms our hypothesis that the raw latent space of the world model is not well aligned for detecting failures and that the failure projector is necessary for effective parameterization of the latent safety filter.

***Ablation*: How should the latent safety filter parameterize the safety constraint?** We hypothesize that sampling constraint representations from the world model training data enables effective parameterization of the latent safety filter, by exposing it to possible test-time constraints and allowing it to learn to interpolate between them. To test this, we ablate the conditioning strategy for both training (7) and runtime filtering (8). We compare four variants: (i) $(z; z_c) \in \mathcal{Z} \times \mathcal{Z}$, which conditions directly on latent states

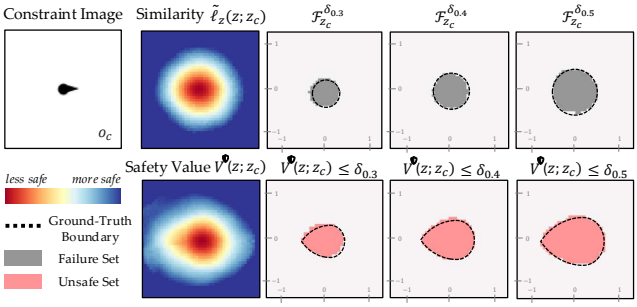

Fig. 4: **Qualitative: Thresholds Calibrated with Different Datasets.** We visualize the failure and unsafe sets constructed with thresholds calibrated from different datasets. *AnySafe* can adjust the size of the failure sets through calibration.

Threshold	FPR↓	Recall ↑	Pre.↑	F_1 ↑	B.Acc.↑	Safe Rate↑	Min. Dist
$\delta = \delta_{0.3}$	0.145	0.986	0.986	0.982	0.917	0.908	0.377
$\delta = \delta_{0.4}$	0.111	0.974	0.981	0.977	0.932	0.880	0.477
$\delta = \delta_{0.5}$	0.082	0.966	0.977	0.971	0.942	0.924	0.569

TABLE III: **Quantitative: Thresholds Calibrated with Different Datasets.** The same safety filter is used with varying thresholds calibrated from different datasets. Minimum distances to the failure set are measured using ground-truth states and averaged over trajectories.

and constraint representations; (ii) $(z;\, p) \in \mathcal{Z} \times \mathcal{P}$, which samples only from a small set of "prototypes "of the latent representations $\mathcal{P} \subset \mathcal{Z}$ during training and uses the nearest prototype to the embedding of a user-specified constraint image at runtime. Prototypes are computed via K-means clustering with 9 cluster centers. (iii) $(z;\, \tilde{z}_c) \in \mathcal{Z} \times \tilde{\mathcal{Z}}$, which conditions the safety filter on projected latent representations; and (iv) $(\tilde{z};\, \tilde{z}_c) \in \tilde{\mathcal{Z}} \times \tilde{\mathcal{Z}}$, which conditions both the current state and the projected latent failure representation.

Table II shows that the latent safety filter performs best when conditioning on the original latent states. Conditioning on diverse failure representations yields high-quality value functions without sacrificing performance, whereas conditioning on a restricted subset $\mathcal{P} \subset \mathcal{Z}$ fails to generalize across diverse failure sets. Notably, conditioning on projected latent representations underperforms, as the projection omits critical state information (e.g., angles) required to accurately recover unsafe sets, preserving only features needed for similarity measures (e.g., positions).

C. Does Calibration Adaptively Control the Failure Set Size?

Since *AnySafe* specifies safety constraints based on continuous latent similarities, the conformal calibration defines the effective size of the failure set and, thus, the unsafe set. To evaluate the conformal calibration process, we test whether the calibrated threshold leads to failure and unsafe sets that align with the user's understanding of the safety constraints.

Setup. We construct three calibration datasets of equal size, but with labels defined under different criteria for detecting failures, $y_i = \mathbb{1}\left\{(p_{i1}^x - p_{i2}^x)^2 + (p_{i1}^y - p_{i2}^y)^2 < \epsilon\right\}$. Specifically, we build $D_{\text{calib}}^{0.3}, D_{\text{calib}}^{0.4}, D_{\text{calib}}^{0.5}$ using thresholds $\epsilon \in \{0.3, 0.4, 0.5\}$, and compute respective $\{\delta_{0.3}, \delta_{0.4}, \delta_{0.5}\}$.

Results. Fig. 4 shows that unsafe sets of different sizes can be accurately estimated using different calibration datasets. The value function learned from continuous similarity signals produces a smooth estimate of the worst-case similarity to the safety constraint. By applying calibrated thresholds to this value function, the safety filter can construct different-sized unsafe sets for runtime filtering. Table III further confirms that calibration enables accurate construction of unsafe sets of varying sizes, effectively constraining how closely the system can approach a safety constraint. The calibrated thresholds yield a high-quality value function with high accuracy. It also maintains a high safety rate during rollouts and ensures that minimum distances to the failure position remain above the desired value, which is the radius ϵ plus a small margin (0.1) that we use for runtime filtering.

VI. Hardware Results: Vision-based Sweeping with a Robotic Manipulator

We scale *AnySafe* to a real-world visual manipulation task using a Franka Research 3 arm equipped with a third-person camera. The robot is tasked with sweeping small objects on a table using a brush, while avoiding a constraint specified at runtime with an image showing objects in the failure region.

A. Experimental Setup

World Model. Following [7], we adopt DINO-WM [4] as the latent dynamics model, and DINOv2 [37] as the encoder. We record $3\times244\times244$ RGB images at 15 Hz, along with the end-effector poses. Actions control only the end-effector's $x-y$ positions and yaw angle. For training the world model, we collect 1,300 trajectories: 300 from teleoperation and 1,000 from random Gaussian-sampled actions.

Failure Projector & Calibration. Since the safety specification for this task depends on the positions of objects on the table, labels are generated from similarity scores computed using the pixel coordinates of the objects' centers of mass. We train a failure projector with 300 labeled trajectories, sampling latent pairs of images and optimizing an MSE loss as in Sec. V. For calibration, we construct a dataset of $N_{\text{calib}} = 4,000$ labeled images, from which pairs are sampled, and the threshold is then calibrated with $\alpha = 0.1$.

Latent Safety Filter Setup. We use DDPG [35] as the solver for the latent safety filter, sampling initial states and constraints from the world model training data.

B. Can AnySafe Safeguard a Runtime-Specified Constraint?

Setup. The task policy π^{task} is human teleoperation, where the teleoperator controls the end-effector's pose while being filtered by *AnySafe*. At runtime, the failure representation z_c is specified by an image depicting an undesirable outcome, such as placing objects within the failure region.

Results: Qualitative. As illustrated in Fig. 5, the teleoperator can freely sweep the objects when they are sufficiently far from the safety constraint specified at runtime. When the objects approach the region indicated by the constraint image,

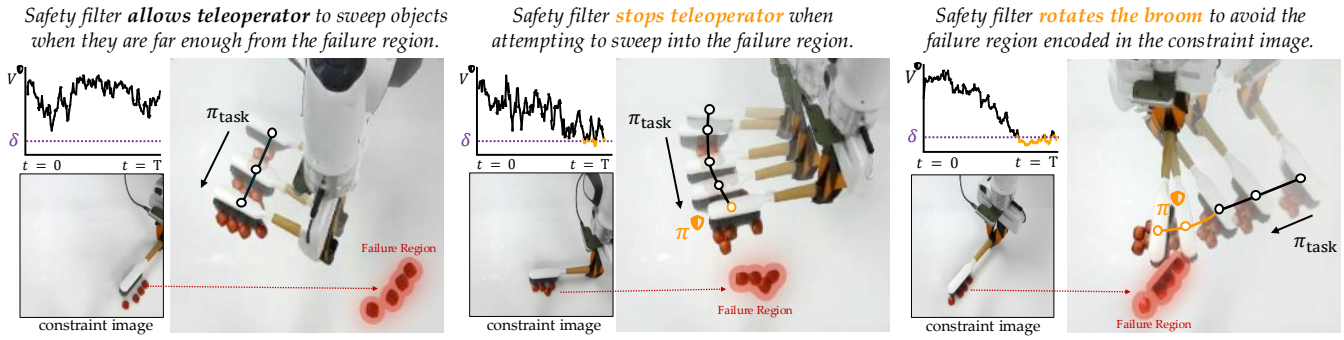

Fig. 5: **Qualitative Results: Filtering π^{task} on Hardware.** *AnySafe* adapts the safety filter based on the different constraint images, intervening only when the chocolates come close to the runtime failure regions indicated in the constraint images.

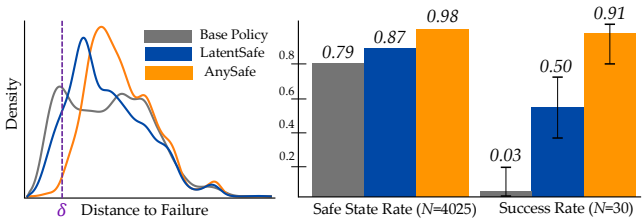

Fig. 6: **Quantitative Results.** *Left*: Empirical distribution of distances between the current object positions and those in the failure images. *Right*: Portion of states outside the failure set and success rate. **AnySafe** keeps distances from the runtime failure region above the threshold.

however, the safety filter detects the proximity and prevents entry into that region. This safety behavior is conditioned on the constraint representation, showing adaptive behavior for different safety constraints (middle and right of Fig. 5).

Results: Quantitative. We record 30 trajectories with different safety constraints, each sweeping the objects close to the failure region specified by the constraint image. We then replay these action sequences as π^{task} from the same initial states, with the safety filter adapted to each safety constraint. We compare against **LatentSafe** [7], which trains three separate safety filters for different safety constraints using binary classifiers, each corresponding to one of three distinct regions. At test time, the constraint image is assigned to one of these regions, and the corresponding safety filter is applied to the base policy. Fig. 6 shows that **AnySafe** achieves higher success rates and consistently keeps the distances to the failure regions below the thresholds ($< 2\%$). In contrast, **LatentSafe** fails to satisfy arbitrary safety constraints, highlighting its limited adaptability.

C. AnySafe's Generalization Beyond Fixed Safety Filters

We next ask whether a single **AnySafe** can generalize across diverse constraints better than multiple fixed filters. We compare **AnySafe** to three fixed filters (**LatentSafe**), each specialized on a different failure region with its own failure classifier. We measure the safety values of each method during a replayed open-loop trajectory that sweeps objects through the three failure regions.

Results. Fig. 7 shows that when **AnySafe** is adapted with a constraint image corresponding to one of these failure

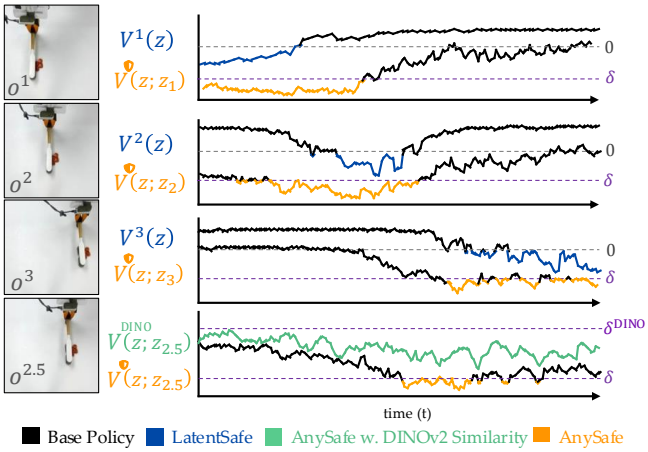

Fig. 7: **Parameterized (AnySafe) vs Fixed (LatentSafe) Safety Filter.** *Left*: Images of three constraints (o^1, o^2, o^3) corresponding to each fixed filter. *AnySafe* adapts to a new constraint, $o^{2.5}$, in between 2 and 3 at test time. *Right*: Safety value functions of the fixed and the parameterized filter.

regions (o^1, o^2, o^3), it produces a safety value function similar to that of **LatentSafe** (top three graphs of Fig. 7). By contrast, when **AnySafe** is adapted with a constraint image located between the failure regions $(o^{2.5})$, it identifies the new unsafe set (bottom of Fig. 7), whereas none of the safety value functions **LatentSafe** detect it as unsafe. Moreover, the *AnySafe* computed directly from raw DINOv2 representations without the failure projector (3) yields ineffective value function estimates (bottom graph of Fig. 7), as raw DINO similarities are not aligned with safety similarity semantics.

D. Can AnySafe Adapt its Level of Conservativeness?

We show that the calibration process allows control over the level of conservativeness by selecting different values of α in (6), enabling adaptive conservativeness of filtering.

Setup. Using the same calibration dataset, we perform calibration with $\alpha \in \{0.01, 0.1\}$ to get $\delta_{0.1}$ and $\delta_{0.01}$. For each calibrated threshold, we replay 15 action sequences as π^{task} that attempt to sweep the objects in the same direction from different starting points, while **AnySafe** filters the π^{task}.

Results. Fig. 8 shows the filtered trajectories with different thresholds. It shows that **AnySafe** can enforce different levels of conservativeness depending on α during the calibration.

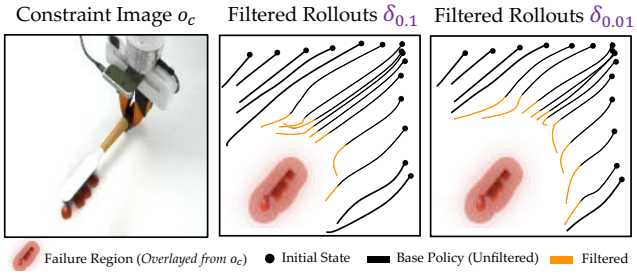

Fig. 8: **Safety Filtering with Different Conservativeness.** Smaller α in calibration leads to more conservative filtering.

While consistently preventing the system from entering the failure region specified by the constraint images, a smaller α yields larger margins from the failure region by activating the safety filter more proactively with a calibrated threshold.

VII. Conclusion & Limitations

In this work, we propose *AnySafe*, a constraint-parameterized latent safety filter that adapts to runtime safety constraints. To enable constraint parameterization in latent space, we condition safety filters on constraint representations encoded from images, using a latent-space similarity measure and conformal calibration to align similarities with the user's notion of failure. Through experiments, we show that *AnySafe* adapts to arbitrary safety constraints specified by a failure image, while also allowing control over the conservativeness of the filtering through conformal calibration.

Limitations. Our work assumes adaptation only to images seen during world model training, leaving open the challenge of generalizing to unseen constraints. It also relies on dense labels for learning failure-related similarity measures, motivating future work on data-efficient metric learning to better align similarity with a user's notion of failure.

References

[1] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *International Conference on machine learning (ICML)*, 2019.

[2] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse control tasks through world models," *Nature*, vol. 640, no. 8059, pp. 647–653, Apr. 2025.

[3] N. Hansen, H. Su, and X. Wang, "Td-mpc2: Scalable, robust world models for continuous control," in *International Conference on Learning Representations (ICLR)*, 2024.

[4] G. Zhou, H. Pan, Y. LeCun, and L. Pinto, "Dino-wm: World models on pre-trained visual features enable zero-shot planning," *International Conference on machine learning (ICML)*, 2025.

[5] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games," *IEEE Transactions on Automatic Control*, 2005.

[6] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger, "Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems," *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 137–177, 2023.

[7] K. Nakamura, L. Peters, and A. Bajcsy, "Generalizing safety beyond collision-avoidance via latent-space reachability analysis," *Robotics: Science and Systems (RSS)*, 2025.

[8] J. Seo, K. Nakamura, and A. Bajcsy, "Uncertainty-aware latent safety filters for avoiding out-of-distribution failures," *Conference on Robot Learning (CoRL)*, 2025.

[9] 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*, vol. 7, 2023.

[10] G. Shafer and V. Vovk, "A tutorial on conformal prediction." *Journal of Machine Learning Research*, vol. 9, no. 3, 2008.

[11] A. N. Angelopoulos, S. Bates *et al.*, "Conformal prediction: A gentle introduction," *Foundations and Trends® in Machine Learning*, 2023.

[12] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, 2016.

[13] K. Margellos and J. Lygeros, "Hamilton–jacobi formulation for reach–avoid differential games," *IEEE Transactions on Automatic Control*, vol. 56, no. 8, pp. 1849–1861, 2011.

[14] O. Bastani, "Safe reinforcement learning with nonlinear dynamics via model predictive shielding," in *American Control Conference (ACC)*. IEEE, 2021, pp. 3488–3494.

[15] S. Bansal and C. J. Tomlin, "Deepreach: A deep learning approach to high-dimensional reachability," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1817–1824.

[16] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging hamilton-jacobi safety analysis and reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8550–8556.

[17] K.-C. Hsu, V. Rubies-Royo, C. J. Tomlin, and J. F. Fisac, "Safety and liveness guarantees through reach-avoid reinforcement learning," in *Robotics: Science and Systems (RSS)*, 2021.

[18] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *IEEE Conference on Decision and Control (CDC)*, 2017, pp. 2242–2253.

[19] J. Borquez, K. Nakamura, and S. Bansal, "Parameter-conditioned reachable sets for updating safety assurances online," in *International Conference on Robotics and Automation (ICRA)*, 2023.

[20] 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 *IEEE Conference on Decision and Control (CDC)*, 2019, pp. 1758–1765.

[21] L. Santos, Z. Li, L. Peters, S. Bansal, and A. Bajcsy, "Updating robot safety representations online from natural language feedback," *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[22] K.-C. Hsu, A. Z. Ren, D. P. Nguyen, A. Majumdar, and J. F. Fisac, "Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees," *Artificial Intelligence*, 2023.

[23] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile but safe: Learning collision-free high-speed legged locomotion," in *Robotics: Science and Systems (RSS)*, 2024.

[24] A. Lin, S. Peng, and S. Bansal, "One filter to deploy them all: Robust safety for quadrupedal navigation in unknown environments," *arXiv preprint arXiv:2412.09989*, 2024.

[25] Y. Yao, Z. Liu, Z. Cen, J. Zhu, W. Yu, T. Zhang, and D. Zhao, "Constraint-conditioned policy optimization for versatile safe reinforcement learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 12 555–12 568, 2023.

[26] L. P. Kaelbling, "Learning to achieve goals," in *IJCAI*, 1993.

[27] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *International Conference on Machine Learning (ICML)*, 2015, pp. 1312–1320.

[28] B. Eysenbach, T. Zhang, S. Levine, and R. R. Salakhutdinov, "Contrastive learning as goal-conditioned reinforcement learning," *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[29] R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, and D. Pathak, "Discovering and achieving goals via world models," *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[30] M. Assran *et al.*, "V-jepa 2: Self-supervised video models enable understanding, prediction and planning," *arXiv preprint arXiv:2506.09985*, 2025.

[31] S. Kim, D. Kim, M. Cho, and S. Kwak, "Proxy anchor loss for deep metric learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3238–3247.

[32] T. Tian, C. Xu, M. Tomizuka, J. Malik, and A. Bajcsy, "What matters to you? towards visual representation alignment for robot learning," in *International Conference on Learning Representations (ICLR)*, 2024.

[33] A. Lin and S. Bansal, "Generating formal safety assurances for high-dimensional reachability," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 10 525–10 531.

[34] K. Chakraborty, A. Gupta, and S. Bansal, "Enhancing safety and robustness of vision-based controllers via reachability analysis," *arXiv preprint arXiv:2410.21736*, 2024.

[35] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[36] I. M. Mitchell *et al.*, "A toolbox of level set methods," *UBC Department of Computer Science Technical Report TR-2007-11*, 2007.
[37] M. Oquab *et al.*, "Dinov2: Learning robust visual features without supervision," 2023.
[38] J. Li, D. Lee, J. Lee, K. S. Dong, S. Sojoudi, and C. Tomlin, "Certifiable reachability learning using a new lipschitz continuous value function," *IEEE Robotics and Automation Letters*, 2025.

VIII. Appendix

A. Implementation Details

Dubins Car. For the Dubins Car experiments, we use a Recurrent State-Space Model (RSSM) [1] as the world model. RSSM decomposes the latent state into deterministic and stochastic components, $z_t := [h_t | x_t]$, where the stochastic latent is modeled as a distribution and optimized via the KL divergence between its prior and posterior. We build on the open-source implementation of DreamerV3[4]. We use a continuous stochastic latent space modeled as a 32-dimensional Gaussian. The action space is normalized to [-1, 1]. The hyperparameters for the Dubins Car experiments are provided in Table IV.

Hyperparameter	Value
Image Dimension	[128, 128, 3]
Action Dimension	1
Stochastic Latent	Gaussian
Latent Dim (Deterministic)	512
Latent Dim (Stochastic)	32
Latent Dim (Failure Projector)	512
Activation Function	SiLU
Encoder CNN Depth	32
Encoder MLP Layers	5
Failure Projector Layers	2
Batch Size	16
Batch Length	64
Optimizer	Adam
Learning Rate	1e-4
Iterations	10000

TABLE IV: Dreamer Hyperparameters

Vision-Based Sweeping. For the real-world hardware task, we use DINO-WM [4], a transformer-based world model that represents latent states using the patch tokens of DINOv2 [37]. The DINOv2 encoder is kept frozen, and only the parameters of a vision transformer are trained. The latent $\hat{z}_{t+1}$ consisting of dense patch tokens is deterministically predicted by conditioning on a sequence of past normalized actions $a_{t-H:t}$ and latent tokens $z_{t-H:t}$ from the previous H timesteps. The model is trained with teacher forcing to ensure temporal consistency by regressing the latent patches, using the mean squared error (MSE) loss: $\mathcal{L}_{\text{DINO}} = \|\mathcal{E}(o_{t+1}) - f_z(z_{t-H:t}, a_{t-H:t})\|^2$. The hyperparameters for DINO-WM are provided in Table V. To measure the similarity of DINOv2 features in Sec. VI-C, we compute the norm of the patch tokens to obtain a global feature $z \in \mathbb{R}^{384}$ from the dense patches $z \in \mathbb{R}^{N_{\text{patches}} \times 384}$.

[4]https://github.com/NM512/dreamerv3-torch

Hyperparameter	Value
Image Dimension	[224, 224, 3]
Action Dimension	3
DINOv2 Patch Size	(16 × 16, 384)
ViT depth	6
ViT attention heads	16
ViT MLP dim	2048
Latent Dim (Failure Projector)	512
Activation Function	SiLU
Failure Projector Layers	2
Batch Size	16
Batch Length	4
Optimizer	Adam
Learning Rate	5e-5
Iterations	100000

TABLE V: DINO-WM Hyperparameters

HJ Reachability Analysis. To solve the latent fixed-point safety Bellman equation, we adopt DDPG [35] within an off-policy, model-based reinforcement learning framework, using the implementation from [38][5]. We model the safety value function as a latent-action value function conditioned on the constraint representation $Q(z, a; z_c)$. The safety policy is parameterized by an actor-network $a = \pi^{\blacktriangledown}(\cdot \mid z, z_c) \in [-1, 1]^{d_{\text{action}}}$, and the safety value is evaluated by $V^{\blacktriangledown}(z) = \max_a Q(z, a; z_c) = Q(z, \pi^{\blacktriangledown}(z, z_c))$.

Each trajectory starts from random initial states with randomly sampled constraint representations. The replay buffer $\mathcal{B}$ then stores transitions of the form $(z, z_c, a, \tilde{l}, z')$. The safety filter is optimized using the following objectives:

$$\mathcal{L}_{\text{critic}} := \mathbb{E}_{\mathcal{B}}\left[\left(Q(z, a; z_c) - y\right)^2\right] \tag{9}$$

$$y = (1-\gamma)\,\tilde{l} + \gamma \min\{\tilde{l}, \max_{a'} Q(z', a'; z_c)\}. \tag{10}$$

$$\mathcal{L}_{\text{actor}} := \mathbb{E}_{z\sim\mathcal{B}}\left[-Q(z, a; z_c)\right], \quad a = \pi^{\blacktriangledown}(\cdot \mid z, z_c), \tag{11}$$

where γ is scheduled from 0.85 to 0.9999. The hyperparameters for training DDPG are summarized in Table. VI.

Hyperparameter	Value
Actor Architecture	[512, 512, 512, 512]
Critic Architecture	[512, 512, 512, 512]
Normalization	LayerNorm
Activation	ReLU
Discount Factor γ	0.9999
Learning Rate (Critic)	1e-3
Learning Rate (Actor)	1e-4
Optimizer	AdamW
Number of Iterations	640000
Replay Buffer Size	1000000
Batch Size	512
Max Imagination Steps	30

TABLE VI: DDPG hyperparameters.

B. Failure Projector

In Sec. IV and Eq. (3), we define a failure projector that maps the raw latent representation of the world model into a metric space where similarities are better aligned with the

[5]https://github.com/jamesjingqili/Lipschitz_Continuous_Reachability_Learning

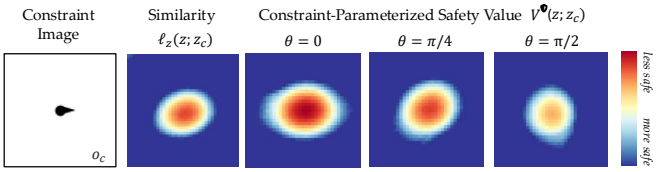

Fig. 9: **Dubins' Car Qualitative Result with Raw Unprojected Features.** We visualize the latent similarity and safety value function at heading slices $\theta \in \{0, \pi/4, \pi/2\}$ using the raw Dreamer features without applying the failure projector.

user's notion of failure. Since *AnySafe* constructs the failure set based on a latent-space similarity metric, the projected latent space retains only failure-relevant features. In Sec. VI-C, we qualitatively demonstrate that the raw DINOv2 latent space is insufficient to define a failure set, yielding a noisy and uninformative similarity metric that cannot capture fine-grained position-based differences, which in turn degrades the quality of the value function.

Architecture. We implement the failure projector as a 2-layer MLP, with the architecture summarized in Table VII. For RSSM, the input is the latent vector formed by concatenating the deterministic and stochastic components, $z_t = [h_t \,|\, x_t]$, resulting in an input dimension of $d_{\text{in}} = 512 + 32 = 544$. For DINO-WM, we compute the norm of the patch tokens from the dense latent features $z \in \mathbb{R}^{N_{\text{patches}} \times 384}$ and concatenate the proprioceptive state, yielding an input dimension of $d_{\text{in}} = 384 + 3 = 387$.

LAYER	INPUT DIM	OUTPUT DIM	NORMALIZATION
`Linear`	d_{in}	d_{in}	`LayerNorm`
`Linear`	d_{in}	32	`LayerNorm`

TABLE VII: Failure Projector Architecture

Ablation. We provide a qualitative evaluation of the raw world model latent space of Dreamer in the Dubins Car. Fig. 9 shows both the feature similarities computed using unprojected world model latents and the corresponding safety value function learned with these latents. Compared to the results learned from projected features shown in Fig. 3 and Fig. 4, the unprojected similarities fail to represent position-based, failure-relevant distinctions, resulting in poorly learned value functions.

C. Calibration of Latent Similarity

In Sec. IV, we compute a threshold for defining a latent failure set and then apply this threshold at runtime for safety filtering in (8), operating on the safety value function. Formally, this calibration procedure guarantees only that the similarity measure—and the corresponding failure set—are calibrated. Approximate value function solvers (e.g, RL) can still induce errors in the downstream safety filter, and calibrating the value function directly requires stronger assumptions about access to the ground-truth unsafe set labels [33]. Nevertheless, we prove in Theorem 1 that under a perfect value function solver, the calibrated threshold δ can be applied directly to the value function learned from the similarity measure, yielding an unsafe set defined as the sub-threshold level set of the value function.

Specifically, we show that learning a threshold-dependent safety value function $V_\delta^{\blacklozenge}$ with a safety margin $\ell_z^\delta := \ell_z - \delta$, is equivalent to learning the value function $V^{\blacklozenge}$ from the raw latent similarity ℓ_z and applying the threshold δ post hoc. This equivalence implies that calibration can be performed after computing the value function using the latent similarities without adjusting them with thresholds, whereas threshold-dependent training would require recomputing the value function whenever the threshold changes.

Theorem 1. *Let $\ell_z : \mathcal{Z} \to \mathbb{R}$ be a safety margin function. Consider the discrete-time safety Bellman backup operator:*

$$(TV)(z_t) = (1-\gamma)\,\ell(z_t) + \gamma \min\Big\{\ell(z_t),\ \max_{a_t \in \mathcal{A}} \mathbb{E}_{\hat{z}_{t+1} \sim f_z(\cdot|z_t, a_t)}\big[V(\hat{z}_{t+1})\big]\Big\}. \tag{12}$$

Let $V^{\blacklozenge}$ denote its unique fixed point. For a constant $\delta \in \mathbb{R}$, define $\ell_z^\delta := \ell_z - \delta$ and let T_δ be the safety Bellman backup operator T where the margin function ℓ_z is replaced by ℓ_z^δ:

$$(T_\delta V)(z_t) = (1-\gamma)\,(\ell(z_t) - \delta) + \gamma \min\Big\{\ell(z_t) - \delta,\ \max_{a_t \in \mathcal{A}} \mathbb{E}_{\hat{z}_{t+1} \sim f_z(\cdot|z_t, a_t)}\big[V(\hat{z}_{t+1})\big]\Big\}. \tag{13}$$

Let the fixed point of this Bellman backup be $V_\delta^{\blacklozenge}$. Then:

$$V_\delta^{\blacklozenge} = V^{\blacklozenge} - \delta \quad \textit{and} \tag{14}$$

$$\{\, z :\ V_\delta^{\blacklozenge}(z) < 0 \,\} = \{\, z :\ V^{\blacklozenge}(z) < \delta \,\}. \tag{15}$$

Proof. Let $V : \mathcal{Z} \to \mathbb{R}$ be any value function. Recall the linearity of expectation and the shift-invariance identities:

$$\min\{a - \delta, b - \delta\} = \min\{a, b\} - \delta,$$
$$\max\{a - \delta, b - \delta\} = \max\{a, b\} - \delta.$$

Using $\ell_z^\delta = \ell_z - \delta$, we compute

$$\begin{aligned}
(T_\delta(V - \delta))(z_t) &= (1-\gamma)\,(\ell_z(z_t) - \delta) \\
&\quad + \gamma \min\Big\{\ell_z(z_t) - \delta,\ \max_{a_t \in \mathcal{A}} \mathbb{E}\big[\,V(\hat{z}_{t+1}) - \delta\,\big]\Big\} \\
&= (1-\gamma)\,\ell_z(z_t) - (1-\gamma)\delta \\
&\quad + \gamma\Big(\min\Big\{\ell_z(z_t),\ \max_{a_t \in \mathcal{A}} \mathbb{E}\big[V(\hat{z}_{t+1})\big]\Big\} - \delta\Big) \\
&= \Big((1-\gamma)\,\ell_z(z_t) \\
&\quad + \gamma \min\Big\{\ell_z(z_t),\ \max_{a_t \in \mathcal{A}} \mathbb{E}\big[V(\hat{z}_{t+1})\big]\Big\}\Big) - \delta \\
&= (TV)(z_t) - \delta.
\end{aligned}$$

In particular, if $V := V^{\blacklozenge}$ is the fixed point of T, then,

$$T_\delta(V^{\blacklozenge} - \delta) = TV^{\blacklozenge} - \delta = V^{\blacklozenge} - \delta, \tag{16}$$

where $V^{\blacklozenge} - \delta$ is a fixed point of T_δ. Since the fixed point is unique [16], $V_\delta^{\blacklozenge} = V^{\blacklozenge} - \delta$.

Thus, the corresponding unsafe sets are equivalent:

$$\begin{aligned}
\{\, z :\ V_\delta^{\blacklozenge}(z) < 0 \,\} &= \{\, z :\ V^{\blacklozenge}(z) - \delta < 0 \,\} \\
&= \{\, z :\ V^{\blacklozenge}(z) < \delta \,\}.
\end{aligned} \tag{17}$$

□