Robotics: Science and Systems 2025
Los Angeles, CA, USA, June 21–25, 2025

Generalizing Safety Beyond Collision-Avoidance via Latent-Space Reachability Analysis

Kensuke Nakamura
Carnegie Mellon University
kensuken@andrew.cmu.edu

Lasse Peters
Delft University of Technology
l.peters@tudelft.nl

Andrea Bajcsy
Carnegie Mellon University
abajcsy@andrew.cmu.edu

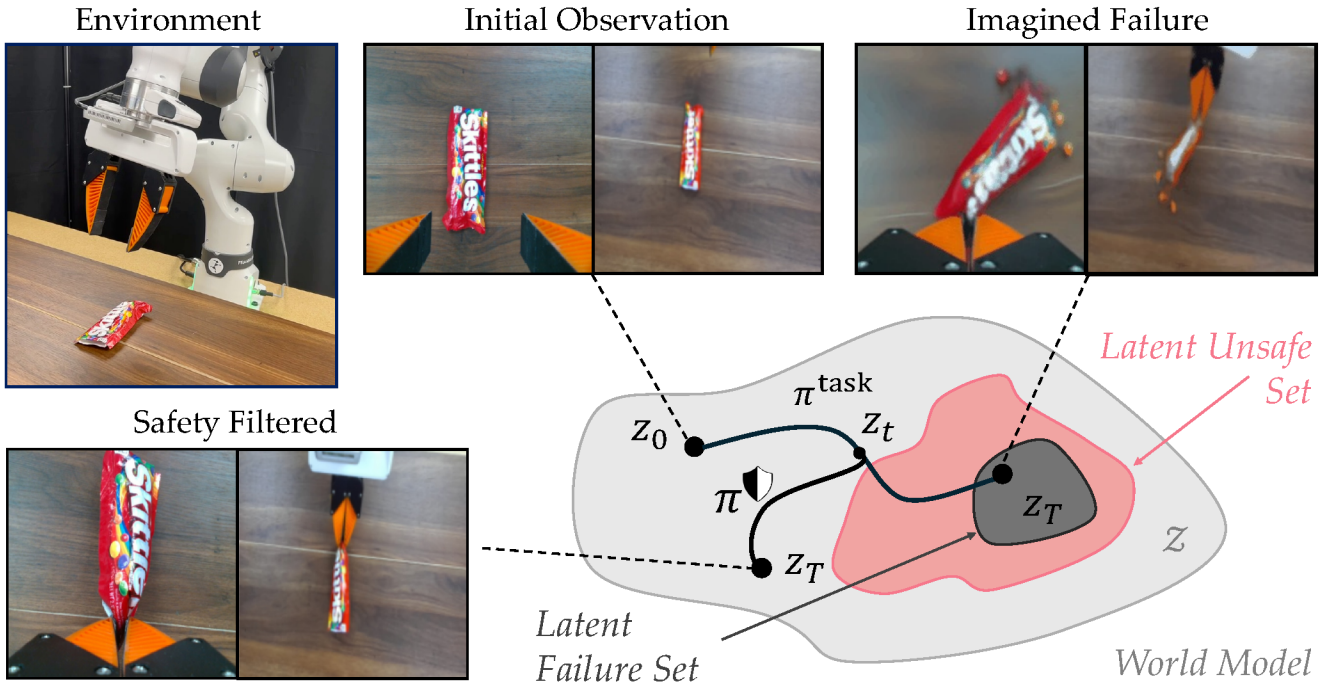

Fig. 1: Our *Latent Safety Filter* can detect, predict, and mitigate failures that are hard to model (e.g., spilling the contents of a bag), such as those encountered in vision-based manipulation. Our idea is to perform approximate reachability analysis in the latent space of a world model (light grey region). The latent failure set is shown as a black region, with an example of an imagined failure observation shown in the upper right. Our method identifies latent states from which the robot is doomed to enter visually-observable failures no matter what actions it takes (larger red set shown above), and automatically overrides the base policy π^{task} with safety-preserving actions from our safety policy $\pi^{\shield}$ to prevent spilling the content of the bag.
Video results can be found on the project website: https://kensukenk.github.io/latent-safety/ .

***Abstract*—Hamilton-Jacobi (HJ) reachability is a rigorous mathematical framework that enables robots to simultaneously detect unsafe states and generate actions that prevent future failures. While in theory, HJ reachability can synthesize safe controllers for nonlinear systems and nonconvex constraints, in practice, it has been limited to hand-engineered collision-avoidance constraints modeled via low-dimensional state-space representations and first-principles dynamics. In this work, our goal is to *generalize* safe robot controllers to prevent failures that are hard—if not impossible—to write down by hand, but can be intuitively identified from high-dimensional observations: for example, spilling the contents of a bag. We propose *Latent Safety Filters*, a latent-space generalization of HJ reachability that tractably operates directly on raw observation data (e.g., RGB images) to automatically computes safety-preserving actions without explicit recovery demonstrations by performing safety analysis in the latent embedding space of a generative world model. Our method leverages diverse robot observation-action data of varying quality (including successes, random exploration, and unsafe demonstrations) to learn a world model. Constraint specification is then transformed into a classification problem in latent space of the learned world model. In simulation and hardware experiments, we compute an approximation of Latent Safety Filters to safeguard arbitrary policies (from imitation-learned policies to direct teleoperation) from complex safety hazards, like preventing a Franka Research 3 manipulator from spilling the contents of a bag or toppling cluttered objects.**

I. Introduction

Imagine that a robot manipulator is deployed in your home, like shown in Figure 1. What safety constraints should the robot reason about? It is common to equate robot safety with "collision avoidance", but in unstructured open world environments, a robot's representation of safety should be much more nuanced. For example, the household manipulator should understand that pouring coffee too fast will cause the

liquid to overflow; pulling a mug too quickly from a cupboard will cause other dishes to fall and break; or, in Figure 1, aggressively pulling up from the bottom of an open bag will cause the contents to spill.

While these failure states—like liquid overflowing, objects breaking, items spilling—are possible to directly identify from high-dimensional observations, understanding how the robot could enter those states is extremely hard. Consider the set of failure states visualized in the black region in Figure 1. These failures correspond to states where the Skittles have *already* spilled onto the table. But even before the spill is visible, there are states from which the robot manipulator is doomed to end up spilling no matter what it does (visualized as a red set in Figure 1): for example, after yanking the bottom of the bag up too quickly, no matter if the robot slows down or reorients the bag, the Skittles are doomed to spill all the way out.

This is where safe control theory can provide some insight. Frameworks like Hamilton-Jacobi (HJ) reachability analysis [44, 48] mathematically model safety constraints very generally—as arbitrary (non-convex) sets in a state space—and automatically identify states from which the robot is doomed to fail in the future by solving an optimal control problem. However, the question remains how to practically instantiate this theoretical framework to safeguard against more nuanced failures—*beyond collision-avoidance*—in robotics.

Our key insight is that the latent representations learned by generative world models [23, 64] enable safe control for constraints not traditionally expressable in handcrafted state-space representations. While world models require diverse coverage (success, play, and/or failure data) to accurately predict the dynamical consequences of robot actions, this formulation makes constraint specification as easy as learning a classifier in the latent space [61], and HJ reachability can safely evaluate possible outcomes of different actions (to determine if the robot will inevitably fail) within the "imagination" of the world model without additional unsafe environment interactions.

We evaluate our approach in three vision-based safe-control tasks in both simulation and hardware:

1) a classic collision-avoidance navigation problem where we can compare our latent approximation with a privileged state solution,
2) a high-fidelity simulation of manipulation in clutter where the robot can touch, push, and tilt objects as long as they do not topple over, and
3) hardware experiments with a Franka Research 3 arm picking up an open bag of Skittles without spilling.

Our quantitative results show that, without assuming access to ground-truth dynamics or hand-designed failure specifications, *Latent Safety Filters* can learn a high-quality safety monitor (F_1 score : 0.982) and the safety controller provides a 63.6% safety failure violation decrease over a base policy trained via imitation [13]. In qualitative experiments, we also find that *Latent Safety Filters* allow teleoperators to freely grasp, move, and pull up on an opened bag of Skittles, while automatically correcting any motions that would lead to spilling.

Statement of Contributions. To summarize, we make four key contributions in this work.

- We formulate a HJ reachability problem in the latent space of a world model. This enables the specification of hard-to-model constraints that go beyond collision-avoidance (e.g., spilling) as a classification problem on the embedding space. The solution to our latent reachability problem automatically yields a safety-preserving policy and unsafe set that operate on high-dimensional RGB observations, without the need for explicit expert recovery demonstrations.
- We tractably compute *Latent Safety Filters* via a reinforcement learning approximation [20] in the world model's latent "imagination", minimizing the need for additional unsafe online interaction data beyond what is needed to learn an accurate world model.
- In a benchmark safe navigation task (with a privileged safety controller and unsafe set) and a contact-rich manipulation task in simulation, we find that *Latent Safety Filters* steer a base policy away from failures while minimizing incompletion rates more effectively than soft constraints or constrained MDP formulations [53, 54].
- We deploy Latent Safety Filters in hardware where the constraint "beyond collision-avoidance" is not to spill Skittles from an opened bag during interaction with a Franka Panda 7DOF manipulator. Our experimental results demonstrate that the **same** *Latent Safety Filter* minimally corrects a human teleoperator from spilling, does not impede a performant imitation-learned policy, makes a suboptimal imitation-learned policy safer, and generalizes to out-of-distribution Skittles bag colors and background changes.

II. A Brief Background on HJ Reachability

Hamilton-Jacobi (HJ) reachability [45, 48] is a control-theoretic safety framework for identifying when present actions will cause future failures, and for computing best-effort policies that minimize failures. Traditionally, reachability assumes access to a privileged state space $s \in \mathcal{S}$ and a corresponding bounded, discrete-time nonlinear dynamics model $s_{t+1} = f(s_t, a_t)$[1] which evolves via the robot's control action, $a \in \mathcal{A}$ where $\mathcal{A}$ is a compact set.

A domain expert will first specify what safety means by imposing a constraint on the state space, referred to as the *failure set*, $\mathcal{F} \subset \mathcal{S}$. Given the failure set, HJ reachability will automatically compute two entities: (i) a safety monitor, $V : \mathcal{S} \to \mathbb{R}$, which quantifies if the robot is doomed to enter $\mathcal{F}$ from its current state s despite the robot's best efforts, and (ii) a best-effort safety-preserving policy, $\pi^{\mathbf{0}} : \mathcal{S} \to \mathcal{A}$. These

[1]Readers familiar with the *continuous-time* HJ reachability formulation [48] will note that a common assumption is Lipschitz-continuous dynamics: $\exists L \geq 0,\ ||f(s_1, a) - f(s_2, a)|| \leq L||s_1 - s_2||$. This ensures the existence and uniqueness of the state trajectory when the dynamics are an ordinary differential equation [44]. In *discrete-time*, assuming Lipschitz-continuous dynamics ensures convergence of traditional numerical methods for the non-discounted optimal control problem [17]. In this work, we only assume the dynamics are bounded, $\exists C \in \mathbb{R},\ ||f(s, a)|| \leq C$, as explained in Section III

two entities are co-optimized via the solution to an optimal control problem that satisfies the fixed-point safety Bellman equation [20]:

$$V(s) = \min\Big\{\ell(s),\ \max_{a\in\mathcal{A}} V(f(s,a))\Big\}, \tag{1}$$

where $\ell : \mathcal{S} \to \mathbb{R}$ is a bounded margin function that encodes the safety constraint $\mathcal{F}$ via its zero-sublevel set $\mathcal{F} = \{s \mid \ell(s) < 0\}$, typically modeled as a signed-distance function. The maximally safety-preserving policy can be obtained via

$$\pi^{\mathbf{0}}(s) := \arg\max_{a\in\mathcal{A}} V(f(s,a)). \tag{2}$$

Finally, the *unsafe set*, $\mathcal{U} \subset \mathcal{S}$, which models the set of states from which the robot is doomed to enter $\mathcal{F}$, can be recovered from the zero-sublevel set of the value function: $\mathcal{U} := \{s : V(s) < 0\}$.

At deployment time, the safety monitor and safety policy can be utilized together to perform *safety filtering*: detecting an unsafe action generated by any base policy, π^{task}, and minimally modifying it to ensure safety. While there are a myriad of safety filtering schemes (see surveys [31, 60] for details), a common minimally-invasive approach switches between the nominal and the safety policy when the robot is on the verge of being doomed to fail: $a^{\text{exec}} = \mathbb{1}_{\{V(s)>0\}}\cdot\pi^{\text{task}} + \mathbb{1}_{\{V(s)\le 0\}}\cdot\pi^{\mathbf{0}}$.

III. Latent Safety Filters

To tackle both detecting and mitigating hard-to-model failures, we present a latent-space generalization of HJ reachability (from Section II) that tractably operates on raw observation data (e.g., RGB images) by performing safety analysis in the latent embedding space of a generative world model. This also transforms nuanced constraint specification into a classification problem in latent space and enables reasoning about dynamical consequences that are hard to simulate.

Setup: Environment and Latent World Models. We model the robot as operating in an environment $\mathbf{E} \in \mathbb{E}$, which broadly characterizes the deployment context—e.g., in a manipulation setting, this includes the geometry and material properties of the table, objects, and gripper. The robot has a sensor $\sigma : \mathcal{S} \times \mathbf{E} \to \mathcal{O}$ that generates observations $o \in \mathcal{O}$ depending on the true state of the world. While we are in a partially-observable setting and never have access to the true state, we will leverage a world model to jointly infer a lower-dimensional latent state and its associated dynamics that correspond to the high-dimensional observations.

A world model consists of an encoder that maps observations o_t (e.g., images, proprioception, etc.) and latent state $\hat{z}_t$ into a posterior latent z_t, and a transition function that predicts the future latent state conditioned on an action. This can be mathematically described as:

$$\text{Encoder: } z_t \sim \mathcal{E}_\psi(z_t \mid \hat{z}_t, o_t)$$
$$\text{Transition Model: } \hat{z}_{t+1} \sim p_\phi(\hat{z}_{t+1} \mid z_t, a_t).$$

This formulation describes a wide range of world models [21, 22, 23, 24, 64], and our latent safety filter is not tied to a particular world model architecture. We focus on world models that are trained via self-supervised learning (observation reconstruction, teacher forcing, etc.) and do not require access to a privileged state. Specifically, in Section IV we use a Recurrent State Space Model (RSSM) [22] trained with an observation reconstruction objective and in Section V we use DINO-WM [64] which is trained with via teacher-forcing.

Safety Specification: Failure Classifier on Latent State. A common approach for representing $\mathcal{F}$ is to encode it as the zero-sublevel set of a function $\ell(s)$ (as in Eq. 1). Domain experts typically design this "margin function" to be a signed distance function to the failure set, which easily expresses constraints like collision-avoidance (e.g., distance between positional states of the robot and environment entities being less than some threshold). However, other types of constraints, such as liquid spills, are much more difficult to directly express with this class of functions and traditional state spaces. To address this, we chose to learn $\ell_\mu(z)$ from data by modeling it as a classifier over latent states $z \in \mathcal{Z}$, with with learnable parameters μ.

We train our classifier on labelled datasets of observations corresponding to safe and unsafe states, $o^+ \in \mathcal{D}_{\text{safe}}$ and $o^- \in \mathcal{D}_{\text{unsafe}}$, and optimize the following loss function inspired by [63]:

$$\begin{aligned}\mathcal{L}(\mu) = &\frac{1}{N_{\text{safe}}}\sum_{o^+\in\mathcal{D}_{\text{safe}}} \text{ReLU}\big(\delta - \ell_\mu(\mathcal{E}_\psi(o^+))\big) \\ &+\frac{1}{N_{\text{fail}}}\sum_{o^-\in\mathcal{D}_{\text{fail}}} \text{ReLU}\big(\delta + \ell_\mu(\mathcal{E}_\psi(o^-))\big),\end{aligned} \tag{3}$$

where loss function is parameterized by $\delta \in \mathbb{R}^+$ to prevent degenerate solutions where all latent states are labeled as zero by the classifier. Intuitively, this loss penalizes latent states corresponding to observations in the failure set from being labeled positive and vice versa. The learned classifier represents the failure set $\mathcal{F}_{\text{latent}}$ in the latent space of the world model via: $\mathcal{F}_{\text{latent}} = \{z \mid \ell_\mu(z) < 0\}$. Our failure classifier can be co-trained (Section IV) or trained after (Section V) world model learning.

Latent-Space Reachability in Imagination. Traditionally, reachability analysis requires either an analytic model of the robot and environment dynamics [4, 47] or a high-fidelity simulator [20, 32] to solve the fixed-point Bellman equation, both of which are currently inadequate for complex system dynamics underlying nuanced safety problems (e.g., liquid interaction). Instead, we propose using the latent imagination of a pretrained world model as our environment model, capturing hard-to-design and hard-to-simulate interaction dynamics. We introduce the latent fixed-point Bellman equation:

$$V_{\text{latent}}(z) = \min\Big\{\ell_\mu(z),\ \max_{a\in\mathcal{A}} \mathbb{E}_{\hat{z}'\sim p_\phi(\cdot\mid z,a)}\Big[V_{\text{latent}}(\hat{z}')\Big]\Big\}. \tag{4}$$

Note that in contrast to Equation 1, this backup operates on the

latent state z and, for full generality, includes an expectation[2] over transitions to account for world models with stochastic transitions (e.g., RSSMs). For world models with deterministic transitions (e.g., DINO-WM), the expectation can be removed.

While the world model allows us to compress high-dimensional observations into a compact informative latent state, computing an exact solution to the latent reachability problem is still intractable due to the dimensionality of the latent embedding (e.g., our latent is a 544 dimensional vector in Section IV). This motivates the use of a learning-based approximation to the value function in Equation 4. We follow [20] and induce a contraction mapping for the Bellman backup by adding a time discounting factor $\gamma \in [0, 1)$:

$$V_{\text{latent}}(z) = (1-\gamma)\ell_\mu(z) + \gamma \min\left\{\ell_\mu(z), \max_{a\in\mathcal{A}} \mathbb{E}_{\hat{z}'\sim p_\phi(\cdot \mid z,a)}[V_{\text{latent}}(\hat{z}')]\right\} \quad (5)$$

We note that the contraction induced by this time-discounted Bellman backup converges to a unique value function under the mild assumptions on the boundedness of the margin function and dynamics [7, 8].

In theory, if solved to optimality, this latent value function would offer a safety assurance only with respect to the data used to train the world model and the failure classifier. Intuitively, this implies that that the robot can only provide an assurance that it will try its hardest to avoid failure *in its representation of the world*. In the following section, we study our overall latent safety framework and the effect of world model dataset coverage on a benchmark safe control task for which we have exact solutions. We then scale to high-dimensional manipulation examples in both simulation and hardware.

IV. Simulation Results

We conduct simulation experiments across two different vision-based tasks to assess the performance of our latent safety framework. These experiments are designed to support the claim that latent safety filters recover performant safety-preserving policies from partial observations alone (i.e., without assuming access to ground-truth dynamics, states, or constraints), for progressively more complex safety specifications and dynamical systems.

A. How Close Does Latent Safety Get to Privileged Safety?

We start by studying a canonical safe-control benchmark: collision-avoidance of a static obstacle with a vehicle. Although this particular setting does not "require" latent-space generalizations of safety, its low-dimensionality and well-studied unsafe set allow us to rigorously compare the quality of the safety filter to a traditional numerical grid-based solution [11] which we take as ground-truth and a privileged-state RL-based safety filter.

[2]This expectation could also be taken to be a risk metric (e.g., CVaR) or a worst-over-N samples of the transition function to induce additional conservativeness.

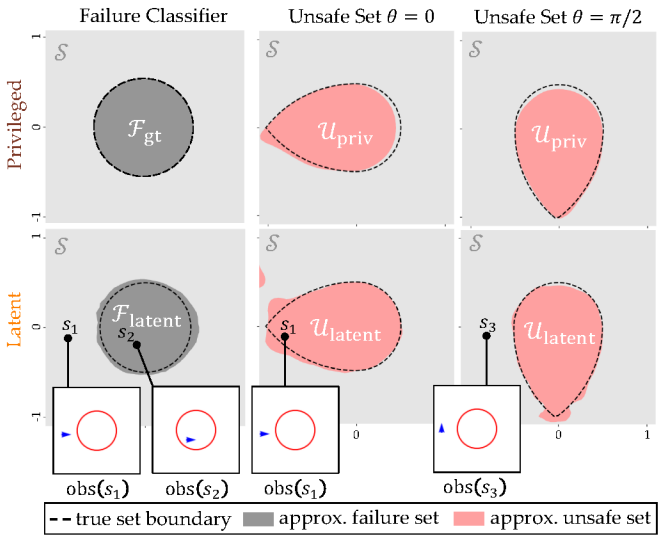

Fig. 2: **Latent Safety vs. Privileged Safety.** Dubins' car collision-avoidance qualitative results. Dashed lines indicate the ground-truth set boundary. We visualize each method's failure specification and corresponding unsafe set, shown at heading slices $\theta \in \{0, \pi/2\}$. While **PrivilegedSafe** uses the ground-truth s and $\mathcal{F}_{\text{gt}}$, **LatentSafe** uses the latent state from encoding the observation, $z = \mathcal{E}_\psi(o)$, and the inferred failure set $\mathcal{F}_{\text{latent}}$. Insets on the bottom row show the observations corresponding to select privileged states s_1, s_2, s_3.

Dynamical System & Safety Specification. In this experiment, the ground-truth dynamics are that of a discrete-time 3D Dubin's car with state $s = [p^x, p^y, \theta]$ evolving as

$$s_{t+1} = f(s_t, a_t) = s_t + \Delta t[v\cos(\theta_t), v\sin(\theta_t), a_t],$$

where the robot's action a controls angular velocity while the longitudinal velocity is fixed at $v = 1\,\text{m}\,\text{s}^{-1}$, and the time-discretization is $\Delta t = 0.05\,\text{s}$. The action space for the robot is discrete and consists of $\mathcal{A} = \{-a_{\max}, 0, a_{\max}\}$ where $a_{\max} = 1.25$ rad/s. To remain safe, the robot must avoid an obstacle of radius $r = 0.5\,\text{m}$ centered at the origin (see red circle in the bottom row of Figure 2). Hence, the ground-truth failure set is a cylinder in state space, $\mathcal{F}_{\text{gt}} := \{s : \forall\theta,\ |p^x|^2 + |p^y|^2 < r^2\}$. The ground-truth margin function $\ell_{\text{gt}}(s) = |p^x|^2 + |p^y|^2 - 0.5^2$ captures the failure condition via its zero-sublevel set.

Baseline: Privileged Safety. The **PrivilegedSafe** baseline computes the HJ value function using the *ground-truth* state, dynamics, and margin function, using a reinforcement learning (RL)-based solver. Specifically, we approximate solutions to the discounted Bellman equation (5) in the framework of [29] via DDQN [57] with Q-functions parameterized by a 3-layer MLP with 100 hidden units. We evaluate $\pi^{\blacklozenge}_{\text{priv}}(s) = \arg\max_{a\in\mathcal{A}} Q^{\blacklozenge}_{\text{priv}}(s, a)$ via action enumeration and a simple lookup procedure due to the discrete action space.

Latent Safety Filter Setup. The **LatentSafe** method does not get privileged access to ground-truth information but

instead learns all model components from data. Specifically, we train a world model from an offline dataset $\mathcal{D}_{\text{WM}} = \{\{(o_t, a_t)\}_{t=0}^{T}\}_{i=1}^{N}$ of $N = 2{,}000$ observation-action trajectories. The observations $o_t := (\mathcal{I}_t, \theta_t)$ consist of (128x128) RGB images $\mathcal{I}$ and the robot heading $\theta_t \in [-\pi, \pi]$. During trajectory generation, we uniformly sampled random actions. Each trajectory terminates after $T = 100$ timesteps or if the ground-truth x or y coordinate left the environment bounds of $[-1\,\text{m}, 1\,\text{m}]$. We trained the world model on this offline dataset using the default hyperparameters of [49]. We used the privileged state to automatically label each observation $o_t \in \mathcal{D}_{\text{WM}}$ as violating a constraint or not, constructing the datasets $\mathcal{D}_{\text{safe}}$ and $\mathcal{D}_{\text{unsafe}}$ for classifier training in Eq. 3. For **LatentSafe**, we parameterize the safety classifier $\ell_\mu(z)$ by a 2-layer MLP with 16 hidden units and co-train it with the world model. The zero-sublevel set of $\ell_\mu(z)$ captures the learned failure set, $\mathcal{F}_{\text{latent}} := \{z \mid \ell_\mu(z) < 0\}$.

One important implementation detail when solving Equation (5) via reinforcement learning in latent space is dealing with the reset mechanism in the world model's imagination. Naively initializing the latent state may result in a latent state that does not correspond to any observation seen by the world model. In practice, we reset the world model by encoding a random observation from our offline dataset and only collect rollouts in the latent imagination for $T = 25$ steps to prevent drifting too far out-of-distribution. For approximating the latent HJ value function and safety controller, we use the same toolbox and hyperparameters as the privileged baseline. We also obtain the safe control via the same procedure as for the priviledged state baseline, via enumerating over the discrete action space to solve for $\pi^{\blacklozenge}_{\text{latent}}(z) = \arg\max_{a \in \mathcal{A}} Q^{\blacklozenge}_{\text{latent}}(z, a)$

Results: On the Quality of the Runtime Monitor, V_{latent}. Since the ground-truth dynamics are known and its state-space is low-dimensional, we can solve for the safety value function *exactly* using traditional grid-based methods [47]. This allows us to report the accuracy of the safe/unsafe classification of the safety filter's monitor (V) in Table I for both the privileged state value function V_{priv} and latent state value function V_{latent} based on their alignment (in terms of sign) with the ground-truth value function V_{gt}. Since both safety filters use the same toolbox for learning the value function, any degradation in the latent safety filter can be attributed to the quality of the learned world model. We also qualitatively visualize the zero-sublevel set of the value function for both methods at different fixed values of θ in Figure 2. In summary, we find that the *image-based* runtime monitor learned by our method (**LatentSafe**) closely matches the accuracy of the *true-state-based* of the privileged baseline (**PrivilegedSafe**).

Results: On the Quality of the Safety Policy, $\pi^{\blacklozenge}_{\text{latent}}$. Each runtime monitor V induces a corresponding safety policy: $\pi^{\blacklozenge}_{\text{latent}}$ for **LatentSafe** and $\pi^{\blacklozenge}_{\text{priv}}$ for **PrivilegedSafe**. To evaluate the quality of these safety policies, we check if they are capable of steering the robot away from failure. To determine initial conditions from which steering away from failure is feasible, we compute a state-based ground-truth value function, V_{gt}, whose zero-sublevel set gives us a dense grid of 250 initial states for which the exact safe controller $\pi^{\blacklozenge}_{\text{gt}}$ can guarantee safety. For all of these states, we simulate each policy executing its best effort to keep the robot outside the privileged failure set, $\mathcal{F}_s$. We find that our *image-based* safety policy closely matches the performance of the *privileged* baseline: **LatentSafe** maintains safety for 240/250 (96%) states and **PrivilegedSafe** maintains safety for 246/250 (98.4%) states.

Method	True Safe	False Safe	False Unsafe	True Unsafe	F_1-score
PrivilegedSafe	0.771	0.011	0.008	0.209	0.987
LatentSafe	0.759	0.003	0.021	0.217	0.984

TABLE I: **Quality of the Runtime Monitor.** Performance of latent (**LatentSafe**) and privileged (**PrivilegedSafe**) safety value functions. Note that this is computed over all three dimensions of the Dubins' car state.

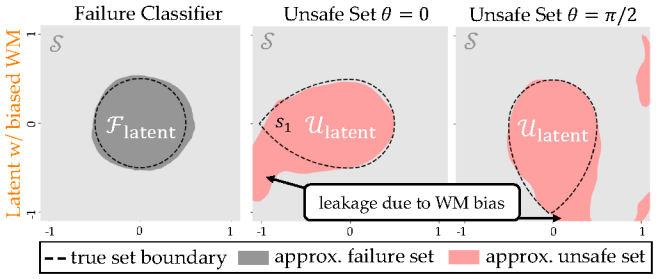

Fig. 3: **Ablation: Latent Safety with Incomplete WM.** Unsafe set approximated by **LatentSafe** using the latent space of a *biased* world-model built from incomplete action coverage $\tilde{\mathcal{A}} = \{0, a_{\max}\} \subset \mathcal{A}$.

Ablation: Effect of Incomplete Knowledge of the World. Thus far, we have had strong coverage of all observation-action pairs when training the world model; however, complete knowledge of the world may not be achievable in reality. To study this, we train our latent safety filter on top of a world model that has seen a biased dataset, wherein the robot's action space is limited to only moving straight or turning left: $\tilde{\mathcal{A}} = \{0, a_{\max}\} \subset \mathcal{A}$. Figure 3 shows that the bias of the world model affects the robot's understanding of safety: since the world model did not learn about the possibility of turning right, **LatentSafe** pessimistically classifies states as unsafe if they require a right turn to avoid collision.

B. Can Latent Safety Scale to Visual Manipulation?

In our simulated manipulation setting, we adapt a contact-rich manipulation task from [54] where a robot is tasked with grasping and lifting a green block that is placed closely between two red blocks (see Figure 4). In this setting, we generalize the safety representation to more nuanced failures, such as the red blocks falling down from aggressive interaction. We train a task policy for this setting that accounts for safety only via a soft constraint and compare the unfiltered behavior

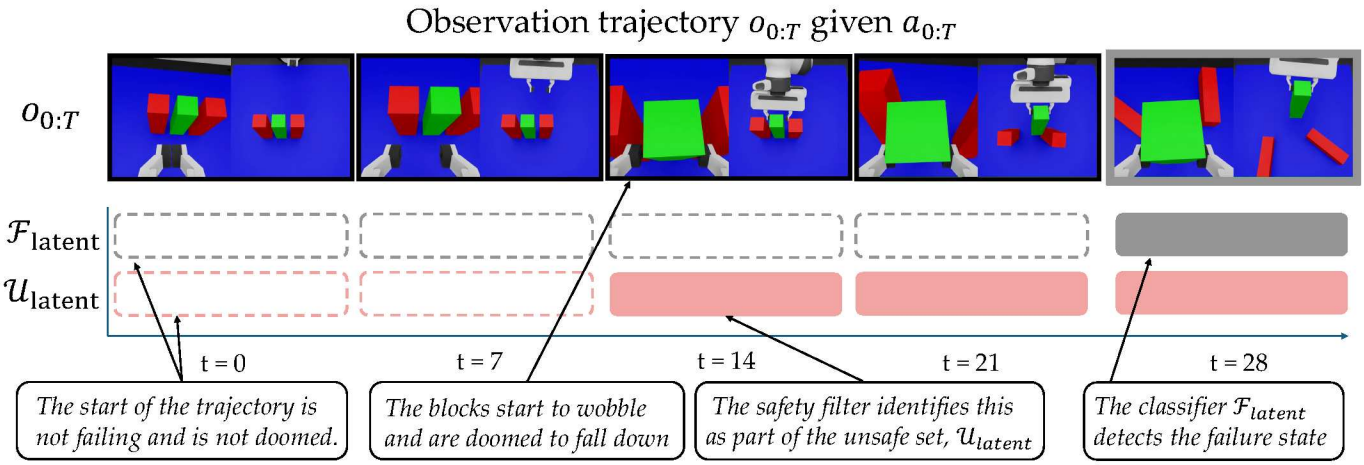

Fig. 4: **Visual Manipulation: Simulation.** *Top row*: Robot's observations corresponding to a known *unsafe* action sequence. *Middle row*: Our learned failure classifier correctly identifies only the final observations at $t = 28$ as being in the failure state since the red blocks have fallen all the way over. *Bottom row*: Our unsafe set (obtained via the latent-space HJ value function) correctly identifies that the robot is doomed to fail the moment that the two red blocks begin to tip over at time $t = 14$.

against two methods for safety filtering: a constrained MDP (CMDP) baseline and our latent safety filter.

Safety Specification. We treat a state as a failure if either of the two red blocks is knocked down. We categorize a block as having fallen if it is angled within 1 rad (measured using privileged simulator information not seen by any of the methods) of the ground plane. Crucially, this safety specification is *not* a collision avoidance specification: the robot is allowed to touch, push, and tilt the red blocks in order to grasp the green block, as long as the red blocks do not topple over.

Experimental Setup. Our nominal task policy π^{task} is obtained via DreamerV3 [24] trained using a dense reward for lifting the block and a sparse cost for violating constraints (**Dreamer**). The observation space $\mathcal{O}$ of the robot is given by two $3 \times 128 \times 128$ RGB camera views (table view and wrist-mounted) along with 8-dimensional proprioception (7-dimensional joint angle and gripper state) information. We co-train the **Dreamer** world model with our failure classifier $\ell_\mu(z)$ for 100k iterations and reuse the world model with frozen weights for our method and all other baselines. The failure classifier is implemented as a 3-layer MLP with ReLU activations. During training, we sampled batches consisting of both rollouts collected by the Dreamer policy (90% of each batch) and a dataset of 200 teleoperated demonstrations (10% of each batch) comprising both safe and unsafe behavior.

We also compare **LatentSafe** to a constrained MDP safety policy, **SQRL** [53, 54]. This method leverages the same world model and failure classifier as **LatentSafe**, but optimizes a different loss function to obtain the safety critic, Q^{risk} [53]:

$$\mathcal{L}(\theta) = \mathbb{E}_{(\hat{z}_t, a_t, \hat{z}_{t+1}, a_{t+1}) \sim \rho_{\pi^{\text{risk}}_{\text{latent}}}} \Big[\frac{1}{2} \big(Q^{\text{risk}}(\hat{z}_t, a_t) - \big(c_t + (1 - c_t)\gamma Q^{\text{risk}}(\hat{z}_{t+1}, a_{t+1})\big)\big)^2 \Big] \tag{6}$$

where $\rho_{\pi^{\text{risk}}_{\text{latent}}}$ is the state-action distribution induced by policy $\pi^{\text{risk}}_{\text{latent}}$ and $c_t \in \{0, 1\}$ takes value 1 if a constraint is violated at timestep t. The resulting Q-value can be interpreted as the empirical risk of violating a constraint by taking action a_t from z_t and following policy $\pi^{\text{risk}}_{\text{latent}}$ thereafter. To mimic the role of our safety filter that is agnostic to any task-driven base policy, we train the **SQRL** critic with trajectories sampled from $\pi^{\text{risk}}_{\text{latent}}(z) := \arg\min_{a \in \mathcal{A}} Q^{\text{risk}}(z, a)$ and no additional task-relevant information. We can then use this policy to filter any actions whose risk is higher than some threshold, ϵ^{risk}. The actor and critic for both **SQRL** and **LatentSafe** are trained in the latent dynamics of the RSSM using DDPG [37, 38]. Since we are in the continuous action space, DDPG trains both a critic network $V_{\text{latent}}(z)$ that approximates the safety value function, and an actor network $\pi_{\text{latent}}(z)$ which is trained to optimize $V_{\text{latent}}(p_\phi(\hat{z}' \mid z, a))$ since a table-lookup is not possible in continuous action spaces. We again reset the latent state of the world model by encoding an observation of previously collected data. All training hyperparameters are included in the Appendix.

During deployment, we use the actor head of **Dreamer** to be the nominal policy, $\pi^{\text{task}}(z)$. To get a performant nominal policy, we tuned the **Dreamer** reward function by sweeping over the reward weights for the components consisting of reaching the green block, lifting the green block, action regularization, and a penalty for the red blocks falling.

Safety Filtering. We instantiate our two safety filters, **LatentSafe** (comprised of $\pi^{\blacklozenge}_{\text{latent}}(z)$ and $V_{\text{latent}}(z)$) and **SQRL** (comprised of $\pi^{\text{risk}}_{\text{latent}}(z)$ and $V^{\text{risk}}(z)$), to shield the base **Dreamer** policy. During each timestep t, we query a candidate action a_t from **Dreamer** that we seek to filter. We instantiate a modified version of the minimally-invasive safety filtering scheme described at the end of Section II. We take the action

in the world model to obtain latent state $\hat{z}_{t+1}$ and evaluate this latent state to obtain $V(\hat{z}_{t+1})$. This value $V(\hat{z}_{t+1})$ will serve as our monitoring signal for whether we are safe or if we should start applying our safety policy. For **LatentSafe**, the filtered (and thus executed) action a_t^{exec} follows the filtering law:

$$a_t^{\text{exec}} = \begin{cases} \pi^{\text{task}}(z_t), & \text{if } V(\hat{z}_{t+1}) > \epsilon, \\ \pi^{\blacksquare}_{\text{latent}}(z_t), & \text{otherwise.} \end{cases} \tag{7}$$

This is a least-restrictive filter[3] that executes the safe control policy $\pi^{\blacksquare}_{\text{latent}}$ whenever $V(\hat{z}_{t+1}) \leq \epsilon = 0.4$. The **SQRL** baseline follows a similar filtering control law defined by:

$$a_t^{\text{exec}} = \begin{cases} \pi^{\text{task}}(z_t), & \text{if } V^{\text{risk}}(\hat{z}_{t+1}) < \epsilon_{\text{risk}}, \\ \pi^{\text{risk}}_{\text{latent}}(z_t), & \text{otherwise.} \end{cases} \tag{8}$$

where ϵ_{risk} is a manually tuned risk threshold that we ablate in our experiments to be $\epsilon_{\text{risk}} \in \{0.1, 0.05\}$.

Results: Qualitative. First, we qualitatively studied if there was a difference between the failure set, $\mathcal{F}_{\text{latent}}$, learned by our classifier and the unsafe set, $\mathcal{U}_{\text{latent}}$, recovered by learning the HJ value function in this visual manipulation task. If $\mathcal{U}_{\text{latent}} \supset \mathcal{F}_{\text{latent}}$, then we have identified a non-trivial unsafe set for this high-dimensional problem. In Figure 4, we show the observations $o_{0:T}$ corresponding to a known *unsafe* action sequence, $a_{0:T}$, where $T = 28$ steps. When the robot executes this action sequence, half-way through the robot touches the red blocks with high enough force that they end up falling over. We pass the observation trajectory into world model encoder to obtain a corresponding posterior latent state trajectory $z_{0:T}$. We evaluate $\text{sign}[\ell_\mu(z_t)]$ and $\text{sign}[V_{\text{latent}}(z_t)], \forall t \in \{0, \ldots, T\}$ to check which latent states are in the failure set and unsafe set respectively. The two rows in Figure 4 correspond to each model's classification. We see that $\mathcal{F}_{\text{latent}}$ correctly identifies that only the final observation at $t = 28$ is in failure, since this is the only observation where the red blocks have fully fallen down. However, $\mathcal{U}_{\text{latent}}$ detects that at timestep $t = 14$, the robot has perturbed the blocks in such a way, that they are doomed to fail.

Results: Quantitative. We rollout the un-shielded base **Dreamer** policy and the policy shielded via **LatentSafe** and **SQRL** for 50 initial conditions of the blocks randomly initialized in front of the robot within $\pm 0.05\,\text{m}$ in the x and y directions. We report the success, constraint violation, and incompletion rates in Table II. We define a constraint violation as any rollout where at least one of the red blocks fall, success as any rollout where the robot successfully picked up the green block without toppling a red block, and incompletion as any rollout that does not violate constraints but failing to picking up the green block.

Despite the penalty for knocking over obstacles, we found that **Dreamer** learned to lift the block even when doing so would incur a safety violation. Although further tuning the reward function could potentially improve the behavior of the robot, reward engineering is notoriously tedious for engineers. This motivates using a safety filter to improve the safety of an unsafe task policy. We report the performance of **SQRL** for two different values of ϵ^{risk} and found that **SQRL** is extremely sensitive to choice of while ϵ^{risk}, growing the task incompletion rate from 4% when $\epsilon^{\text{risk}} = 0.1$ to 70% when $\epsilon^{\text{risk}} = 0.05$ while only marginally improving safety. In contrast, **LatentSafe** overrides our nominal task policy only when needed, significantly reducing the number of constraint violations while still succeeding at the task.

[3]While in theory $\epsilon = 0$ is an appropriate threshold for safety filtering, practitioners often select $\epsilon > 0$ to account for potential numerical errors or latency in the system.

Method	Safe Success % (↑)	Constraint Violation % (↓)	Incompletion % (↓)
Dreamer	64	36	**0**
SQRL ($\epsilon_{\text{risk}} = 0.1$)	68	28	4
SQRL ($\epsilon_{\text{risk}} = 0.05$)	8	22	70
LatentSafe	**80**	**20**	**0**

TABLE II: **Visual Manipulation: Simulation.** Success at the task without any safety violations, constraint violations, and incompletion rates across 50 rollouts corresponding to 50 random initial conditions of the blocks. Task success is picking up the green block; constraint violation is where either of the red blocks fall down on the table.

V. Hardware Results: Preventing Hard-to-Model Robot Failures

Finally, we design a set of experiments in hardware to see if our Latent Safety Filter can be applied in the real world (shown in Figure 1). We use a fixed-base Franka Research 3 manipulator equipped with a 3D printed gripper from [14]. The robot is tasked with interacting with an opened bag of Skittles on the table. The safety constraint is not to spill any Skittles. We test the efficacy of our approach by deploying the *same* Latent Safety Filter to safeguard a human teleoperator (Section V-A) and a strong and weak Diffusion Policy [13] from spilling (Section V-B), as well as stress-testing our safety filter to out-of-distribution candy bags and environment backgrounds (Section V-C).

Safety Specification. Our safety specification is to prevent the contents of the Skittles bag from falling out of the bag. Given only image observations and proprioception, this problem is clearly partially observed since the robot cannot directly recover the position of the Skittles in the bag. Even if privileged state information was available, designing a function to characterize the set of failure states and a dynamics model for interactions between all relevant objects (e.g., the manipulator, the soft bag, the Skittles, and the table) would be extremely difficult.

Latent Safety Filter Setup. We use DINO-WM [64], a Vision Transformer-based world model that uses DinoV2 as an encoder [50]. The manipulator uses a 3rd person camera and a wrist-mounted camera and records $3 \times 256 \times 256$ RGB

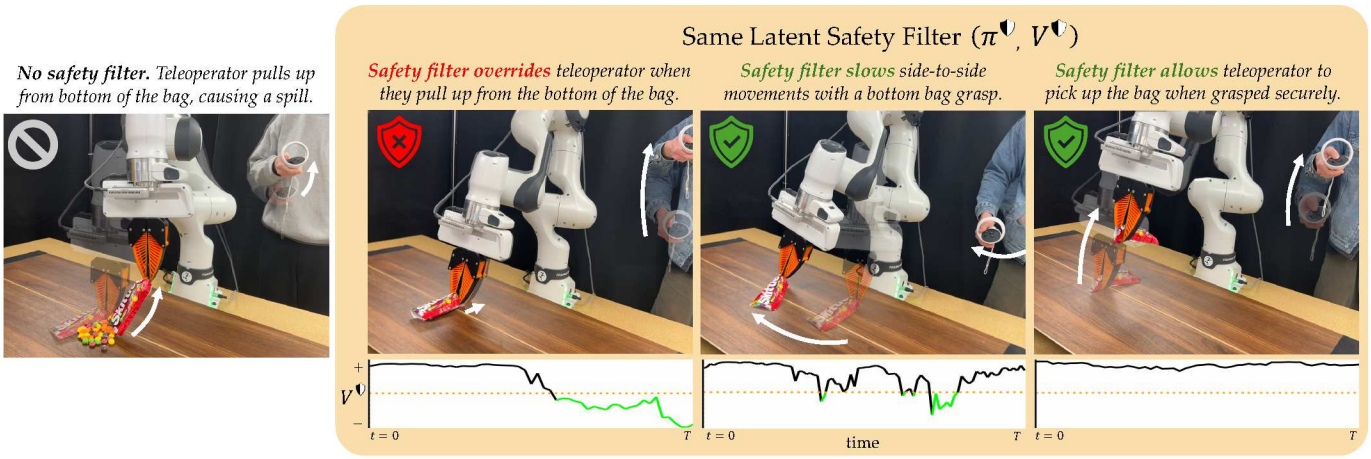

Fig. 5: *Far Left*: Without a safety filter, a teleoperator lifts the closed-end of the bag too quickly and spills the Skittles. *Middle Left*: By using **LatentSafe**, the same action of lifting the closed-end leads to the value function $V^{\blacktriangledown}_{\text{latent}}$ dipping below the safe threshold (orange) and prompting the safety policy to override the teleoperator (green); the robot does not allow the human pull the bag up sharply. *Middle Right*: At the same time, **LatentSafe** slows down the human's attempt to move the bag side-to-side while grasping the closed end, indicating that the safety filter has a nuanced understanding of which actions will and won't violate safety. *Right*: Grasping the bag from the open end and lifting is deemed safe and is allowed by **LatentSafe**.

images at 15 Hz. For world model training, we collected a dataset $\mathcal{D}_{\text{WM}}$ of 1,300 offline trajectories: 1,000 of the trajectories are generated sampling random actions drawn from a Gaussian distribution at each time step, 150 trajectories are demonstrations where the bag is grasped without spilling any Skittles, and 150 demonstrations pick up the bag while spilling candy on the table. We manually labeled the observations in the trajectory dataset for apparent failures. However, in principle, we believe that this data annotation process could also be automated using alternative methods, like state-of-the-art foundation models (e.g., vision-language models).

Our world model is trained by first preprocessing and encoding the two camera view using DINOv2 to obtain a set of dense patch tokens for each image. We use the DINOv2 ViT-S, the smallest DINOv2 model with 14M parameters, resulting in latent states z of size 256×384 corresponding to 256 image patches each with embedding dimension 384. The transition function is implemented as a vision transformer, which takes as input the past $H = 3$ patch tokens, proprioception, and actions to predict the latent. The transformer employs frame-level causal attention to ensure that predictions can only depend on previous observations. The model is trained via teacher-forcing minimizing mean-squared error between the ground-truth DINO embeddings of observations and proprioception information from $\mathcal{D}_{\text{WM}}$ and the embeddings and proprioception predicted by the model. Additional details on the model and hyperparameters are included in the appendix and in [64]. After world model training, we separately learn the failure classifier (implemented as a 2-layer MLP with hidden dimension 788 and ReLU activations) on the DINO patch tokens corresponding to the manually labeled constraint-violating observations. For approximating the HJ value function, we use DDPG [37, 38].

A. Shielding Human Teleoperators

To emphasize the policy-agnostic nature of our latent safety filters, we demonstrate filtering the actions of a teleoperator.

Setup. The teleoperator controls the end-effector position and gripper state via a Meta Quest pro similar to [35], and can freely move the robot around. They are tasked with interacting with the Skittles bag however they like. The safety filter operates according to the following control law:

$$a_t^{\text{exec}} = \begin{cases} \pi^{\text{task}}(z_t), & \text{if } V(\hat{z}_{t+1}) > \epsilon \\ \pi^{\blacktriangledown}_{\text{latent}}(z_t), & \text{otherwise} \end{cases} \tag{9}$$

where $\hat{z}_{t+1} \sim p_\phi(\hat{z}_{t+1} \mid z_t, \pi^{\text{task}}(z_t))$ is a one-step rollout of the world model using the action proposed by the unshielded teleoperator. In hardware experiments, we set $\epsilon = 0.3$. Both the teleoperation and safety filtering were executed at 15 Hz.

Results: Shielding Unsafe Grasps and Dynamic Motions. We visualize our qualitative results in Figure 5. Un-shielded by our safety filter, the teleoperator can grab the opened bag of Skittles by the base and pull up sharply, spilling its contents on the table (left, Figure 5). By using **LatentSafe**, the same behavior gets automatically overridden by the safety filter, preventing the teleoperators "pull up" motion from being executed and keeping the Skittles inside (center, Figure 5). At the same time, the latent safety filter is not overly pessimistic (right-most images in Figure 5). When the teleoperator moves the Skittles bag side-to-side while grasping the *bottom* of the opened bag, the safety filter accurately accounts for these dynamics and minimally modifies the teleoperator to slow them down, preventing any Skittles from falling out while still allowing the general motion to be executed. When the teleoperator chooses a safe grasp—grabbing the bag by the

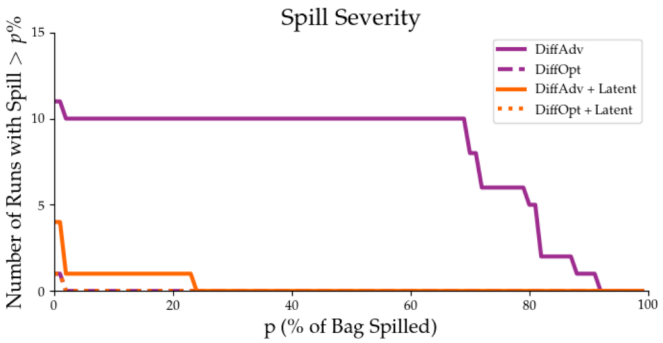

Fig. 6: **Shielding IL Policies: Hardware Results.** Percent of bag spilled (p) vs number of runs that spilled at least $p\%$ of the bag. While **DiffusionAdv** frequently spills a large percentage of the bag ($\sim 85\%$), **DiffusionAdv** + **LatentSafe** spills less than 5% of the bag in all but one of the constraint-violating rollouts. **DiffusionOpt** with and without **LatentSafe** spills only 1 skittle in across all 15 rollouts, showing that latent safety is not overly conservative.

top, open side—the safety filter does not activate and allows the person to complete the task safely and autonomously.

B. Shielding Autonomous Imitation-Learned Policies

Next we study how well the same Latent Safety Filter from Section V-A can shield autonomous imitation-learned (IL) policies. Specifically, we test whether the latent safety filter does *not* impede a strong IL policy (i.e., our filter is not overly conservative) and *improves* the safety of a suboptimal IL policy (i.e., our filter shields effectively), while removing teleoperator bias that may be present in our prior experiments.

Methods. For our base task policy, $\pi^{\text{task}}(o)$, we use a generative imitation-learned (IL) policy trained with a diffusion objective [13] and which takes as input RGB images and end effector pose as observations $o \in \mathcal{O}$ (implementation details can be found in the Appendix). We train two diffusion policies—**DiffusionAdv** and **DiffusionOpt**—which represent relevant extremes of a base policy's capabilities. **DiffusionOpt** represents the "upper bound" of a strong base policy that uses carefully curated demos of the task. We use this baseline to study whether our safety filter is not overly conservative when shielding a strong base policy. We train it with 100 teleoperated demonstrations wherein the expert grasps the Skittles bag from the middle and lifts it off the table without spilling. We also train **DiffusionAdv**, which represents a "lower bound" of a base policy trained with demonstrations that could lead to unsafe outcomes. This policy is trained with 100 potentially unsafe teleoperated demonstrations: the expert grasps and lifts the Skittles bag from its closed end, but the opening is internally sealed during data collection time so that no Skittles could fall out. This results in a base policy that has an incomplete understanding of how to interact safely with the Skittles bag, allowing us to test our safety filter's ability to prevent failures in a controlled and repeatable manner.

Metrics. We compare the performance of the $\pi^{\text{task}} \in \{\textbf{DiffusionAdv}, \textbf{DiffusionOpt}\}$ with and without using **LatentSafe** (yielding four methods in total). We use exactly the same *Latent Safety Filter* as we used to shield the human teleoperator in Section V-A. For each method, we record 15 rollouts where the policy successfully grasped the bag (ignoring missed grasps) 15 times in hardware. We measure the frequency of constraint violations (if even one Skittle falls out during an episode) and spill severity (percentage of the Skittles spilled) in each of these trajectories.

Results: Shielding a Weak IL Policy. When **LatentSafe** shields the weak **DiffusionAdv**, we see a 63.6% decrease in constraint violations compared to **DiffusionAdv** acting alone. While **LatentSafe** does still fail 26.4% of the time, we note that 3 out of the 4 failures only spilled a single skittle. In contrast, **DiffusionAdv**'s autonomous spill rate was 73.4% with many instances where a large percentage of the bag was spilled. To better understand the *severity* of the constraint violations, we report in Figure 6 how often each method spilled more than $p\%$ of the bag. While **DiffusionAdv** frequently spills a large percentage of the bag ($\sim 85\%$), the safety filter spills less than 5% of the bag in all but one of the constraint-violating rollouts, where it spilled only 24%. Overall, **LatentSafe** minimizes both the failure rate and severity when the base IL policy is erroneous and can cause difficult-to-model failures.

Results: Shielding a Strong IL Policy. We report the severity of constraint violations in Figure 6. We note that both when **DiffusionOpt** acts alone and when it is shielded by **LatentSafe**, the policies safely grasp and lift the bag in the first 15 trials. Both methods exhibited only one safety violation, where a single Skittle was spilled. Overall, we see that our Latent Safety Filter is not overly-conservative, allowing a strong base policy to operate without unnecessary overrides. We also note that practically, since the same Latent Safety Filter was used for both the weak and the strong base IL policy, this provides a promising avenue for safely improving a base task-driven policy without the need to also change the safety representation and fallback controller.

C. Testing Out-of-Distribution Generalization of Latent Safety

Finally, we stress-test the performance of the same Latent Safety Filter on out-of-distribution (OOD) bag colors, candy dynamics, and background changes.

Setup. Recall that our Latent Safety Filter was trained only on classic red Skittles bag and with a wooden table-top background. In these tests, we first vary the color of the Skittles bag, where OOD Skittles are green (Sour) and purple (Wild Berry). We also test OOD candy bags: M&Ms Classic, M&Ms Peanut Butter, M&Ms Peanut. Note that these are both visually and dynamically OOD: we qualitatively observe that M&Ms bags are stiffer and have a papery texture compared to the Skittles bags. Furthermore, the M&Ms are have differing weights (e.g., with peanuts). Finally, we also deploy our method to interact with a red Skittles bag, but cover the table with an OOD black cloth.

Methods. To isolate the influence of OOD conditions on our Latent Safety Filter, we replay a known unsafe demonstration from our dataset in open-loop as our task "policy". We reset the bag to the same initial condition and shield this replayed demonstration with **LatentSafe** filter for all OOD conditions.

Results. We plot the safety value function and the final observation of the system for the OOD Skittles and background in Figure 7 and OOD M&Ms in Figure 8. When shielding OOD Skittles bags and the novel background, the filtering override profile is similar for all of the bag colors and results in the same final observation where none of the candy is spilled. In contrast, we observe a noticeably different safety value profile for the OOD M&Ms in Figure 8. For both the Classic and Peanut M&Ms, our method fails to prevent spills. Note that for the Classic M&Ms (brown), the safety filter started to activate prior to the candy spilling (grey line) but was unable to prevent a spill. We hypothesize that the powerful, pre-trained DINOv2 embeddings are able to identify semantic equivalences between the differing bags[4], which would be sufficient for safe control across dynamically equivalent bags. However, despite the encoder's ability to map visually similar observations to similar embeddings, there is no reason to suggest that the transition model can generalize across different bag dynamics in our extremely low data regime, which would explain the gap in closed-loop performance between M&Ms and Skittles.

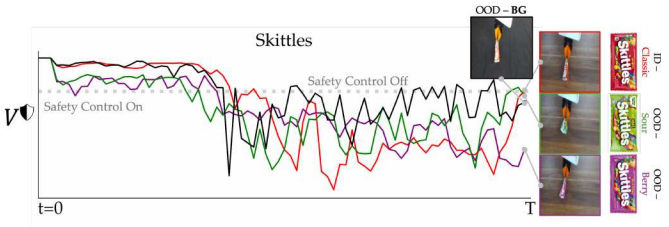

Fig. 7: **OOD Generalization: Skittles.** Our latent safety filter is trained only on a red Skittles bag. It is deployed to shield an open-loop known unsafe trajectory for two OOD skittles bag colors and an OOD background. **LatentSafe** generalizes—maintaining the same performance of preventing spills—to OOD Skittles bag colors and OOD background change.

VI. Related Work

Safety Filtering for Robotics. Safety filtering is a control-theoretic approach for ensuring the safety of a robotic system in a way that is agnostic to a task-driven base policy [31]. A safety filter monitors a base task policy and overrides it with a safe control action if the system is on the verge of becoming unsafe. Control barrier functions [2], HJ reachability [19, 45, 48], and model-predictive shielding [59] are all common ways of instantiating safety filters (see [60] for a survey). The most relevant recent developments include computing safety filters from simulated rollouts of first-principles dynamics models or high-fidelity simulators via reinforcement learning or self-supervised learning [4, 20, 29, 32, 51], safety filters that operate on a belief-state instead of a perfect state [1, 3, 33, 56], and safety filters that keep a system in-distribution of a learned embedding space [12]. Our work contributes to the relatively small body of work that attempts to bridge the gap between high-dimensional observations, such as LiDAR [27, 39] or RGB images [30, 55], and safety filters. However, unlike previous work, our method does not infer hand-crafted intermediary representations of state from the high-dimensional observations nor does it represent safety as collision-avoidance. Instead, our safety filter operates in the latent space of a world model, reasoning directly about the embedded RGB observations and shielding against hard-to-model failure specifications.

[4]In the Appendix, we provide qualitative evidence by visualizing the top three PCA components of the DINOv2 embeddings of all OOD scenarios.

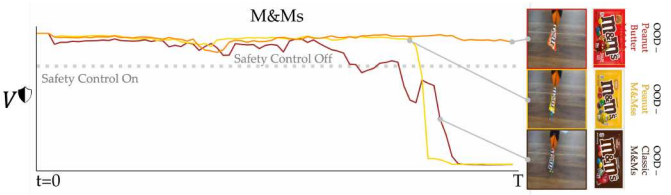

Fig. 8: **OOD Generalization: M&Ms.** Our latent safety filter is trained only on a red Skittles bag. It is deployed to shield an open-loop known unsafe trajectory. **LatentSafe** is deployed with 3 M&M bags with different colors and dynamics. Our filter does *not* prevent the manipulator from lifting these OOD bags. For the brown bag, even though the filter begins to override the recorded trajectory, it does not manage to prevent the spill, potentially due to differing dynamics.

Latent Space Control. The control and model-based reinforcement learning community has recently demonstrated the potential for using generative world models for real-world robot control [46, 62]. One of the advantages of world models is that they transform a control problem with partial observability into a Markov decision process in the learned latent state space. Many methods for shaping this latent state representation exist, from observation reconstruction [22, 23, 24], teacher-forcing [64], reward predictions [25, 26], or explicit metric learning in the latent space [28]. While prior works traditionally use the world model to learn a policy for a specific task, we use the world model to learn a policy-agnostic safety filter that reasons about unsafe consequences it can "imagine" (but are hard to model) in the latent space via reachability analysis. We note that our paper is not the first to apply control-theoretic principles in the latent space of a learned world model. Latent models have been used to estimate the regions of attraction for data-driven policies [58], collision-free motion planning [34], and forward reachability methods have been used for structured exploration toward states that are far from a system's initial state [9]. Instead, we take a *backward* reachability approach [5] that determines the set of

states that will *inevitably* lead to failure despite the robot's best effort to prevent failure. This allows us to simultaneously compute a runtime monitor as well as a safety recovery policy that can preemptively prevent failures.

Learning about Safety from Expert Demonstrations. Given an offline dataset of expert data and a privileged state space, there are two predominant methods for learning about safety. The first uses expert demonstrations to learn a safe policy directly. For example, control barrier functions can be learned from expert demonstrations if the ground-truth dynamics and their Lipschitz constants are known [40] and [51]. Inverse constraint learning methods [15, 36, 52] infer constraints given expert trajectories that are optimal with respect to a known reward and unknown constraint; high-reward trajectories that the expert demonstrator did not take must have a safety violation. The assumption of a known reward has been relaxed in [16, 41] but still requires trajectories that are (locally) optimal with respect to some task. Our work does not assume expert demonstration trajectories; it only assumes observations labeled with a binary indication of failure (which we use to learn our failure state classifier). We also do not assume a known state or dynamical system model, can use diverse robot deployment data regardless of its optimality (for world model building and failure classifier training), and we compute the safety policy as an optimization rather than requiring demonstrations of safety-preserving behavior.

Computing Safety Behaviors from Offline Data. Our method aligns with the other set of approaches focusing on programmatically computing the safety filter via synthesis techniques (e.g., optimal control) once the *safety specification* (i.e., the failure set) is encoded or learned. Constraints more nuanced than collision avoidance can be specified with signal and linear temporal logic using demonstrations [6] or language [18], but have still been restricted to constraints that can be expressed using hand-designed predicates of privileged state variables. Prior works have learned about failure states by using intervention data in a learned latent space [42, 43] or binary indicators of constraint violation (provided by an oracle) [53, 54, 61]. Our method leverages labels on robot observations (provided by an annotator) to learn a margin function on the embeddings of high-dimensional observations, which represents hard-to-model constraints via its zero-sublevel set. We then utilize reinforcement-learning-based HJ reachability to synthesize safe behavior automatically, rather than relying on a demonstrator to provide recovery behavior as is done in [40, 43, 51]. While similar to our work in its use of a latent space, [61] solves for a policy which co-optimizes safety and task performance. This renders the resulting safe policy *task-specific*, preventing its use as a general-purpose, policy-agnostic safety filter like we formulate. Furthermore, while prior work uses additional interactions with the ground truth environment to update its safety policy and world model [43, 54, 61], our work learns a safety filter entirely within a frozen world model, minimizing the need for any additional (potentially unsafe) environment interactions.

VII. Limitations

Our latent HJ reachability formulation generalizes the space of failures robots can safeguard against. However, it is not without its limitations. One limitation is that the safety filter can only protect against outcomes that the world model can predict. This means that the world model needs to be trained on some amount of unsafe data in order to effectively predict these unsafe outcomes and compute control policies that steer clear of them. Additionally, the least-restrictive safety filter mechanism we implemented returns a single control action that attempts to steer the robot away from danger maximally and at the last moment. More sophisticated approaches to search through the set of safe actions and align them with the base policy's task performance should be explored [31, 60]. We also inherit some limitations from value-based safe control techniques, such as the need to recompute a safety value function for any new additional constraints. However, we view this as an exciting opportunity for future work where the safety value function is parameterized by the failure specification [10] for additional flexibility without the need for recomputation. Finally, since we are concerned with robot safety, it is important to acknowledge what types of assurances we can expect from this framework. Although our method is grounded in rigorous theory, our practical implementation currently lacks formal assurances due to the combination of possible errors when learning the world model, failure classifier, and resulting HJ value function. Characterizing how the errors in one learned component propagate to the downstream safety assurances is exciting future work.

VIII. Conclusion

In this work, our goal was to generalize robot safety beyond collision-avoidance, accounting for hard-to-model failures like spills, items breaking, or items toppling. We introduce *Latent Safety Filters*, a generalization of the safety filtering paradigm that operates in the learned representation of a generative world model. We instantiated our method on a suite of simulation and hardware experiments, demonstrating that our latent reachability formulation is comparable to privileged state formulations, outperforms other safe control paradigms while being less sensitive to hyperparameter selection, and protects against extremely hard-to-specify failures like spills in the real world for both generative IL policies and human teleoperation. Future work should thoroughly investigate the uncertainties within each component of our latent space safety generalization and investigate theoretical or statistical assurances on this new safety paradigm.

IX. Acknowledgements

The authors would like to thank Gaoyue Zhou for the guidance on DINO-WM and Chenfeng Xu for assistance with ViTs. We also thank Gokul Swamy, Yilin Wu, and Junwon Seo for helpful discussions. This material is supported in part by the National Science Foundation Graduate Research Fellowship Program under Grant NoDGE2140739 and NSF CAREER Award No. 2441014. Any opinions, findings, and conclusions

or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Mohamadreza Ahmadi, Andrew Singletary, Joel W Burdick, and Aaron D Ames. Safe policy synthesis in multi-agent pomdps via discrete-time barrier functions. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 4797–4803. IEEE, 2019.

[2] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

[3] Andrea Bajcsy, Anand Siththaranjan, Claire J Tomlin, and Anca D Dragan. Analyzing human models that adapt online. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2754–2760. IEEE, 2021.

[4] Somil Bansal and Claire J Tomlin. Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824. IEEE, 2021.

[5] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[6] Ezio Bartocci, Cristinel Mateis, Eleonora Nesterini, and Dejan Nickovic. Survey on mining signal temporal logic specifications. *Information and Computation*, 289: 104957, 2022.

[7] Richard Bellman. On the theory of dynamic programming. *Proceedings of the national Academy of Sciences*, 38(8):716–719, 1952.

[8] Dimitri Bertsekas. *Dynamic Programming and Optimal Control*, volume 2, page 14–22. Athena Scientific, 4 edition, 2018.

[9] Homanga Bharadhwaj, Animesh Garg, and Florian Shkurti. Leaf: Latent exploration along the frontier. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 677–684. IEEE, 2021.

[10] Javier Borquez, Kensuke Nakamura, and Somil Bansal. Parameter-Conditioned Reachable Sets for Updating Safety Assurances Online. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10553–10559, May 2023. doi: 10.1109/ICRA48891.2023.10160554. URL https://ieeexplore.ieee.org/document/10160554/?arnumber=10160554.

[11] Minh Bui, George Giovanis, Mo Chen, and Arrvindh Shriraman. Optimizeddp: An efficient, user-friendly library for optimal control and dynamic programming. *arXiv preprint arXiv:2204.05520*, 2022.

[12] Fernando Castaneda, Haruki Nishimura, Rowan Thomas McAllister, Koushil Sreenath, and Adrien Gaidon. In-distribution barrier functions: Self-supervised policy filters that avoid out-of-distribution states. In *Learning for Dynamics and Control Conference*, pages 286–299. PMLR, 2023.

[13] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[14] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[15] Glen Chou, Dmitry Berenson, and Necmiye Ozay. Learning constraints from demonstrations. In *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*, pages 228–245. Springer, 2020.

[16] Glen Chou, Necmiye Ozay, and Dmitry Berenson. Learning constraints from locally-optimal demonstrations under cost function uncertainty. *IEEE Robotics and Automation Letters*, 5(2):3682–3690, 2020.

[17] Marizio Falcone and Roberto Ferretti. Discrete time high-order schemes for viscosity solutions of hamilton-jacobi-bellman equations. *Numerische Mathematik*, 67 (3):315–344, 1994.

[18] Cameron Finucane, Gangyuan Jing, and Hadas Kress-Gazit. Ltlmop: Experimenting with language, temporal logic and robot control. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1988–1993. IEEE, 2010.

[19] J. Fisac, M. Chen, C. J. Tomlin, and S. Sastry. Reach-avoid problems with time-varying dynamics, targets and constraints. In *HSCC*, 2015.

[20] Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019.

[21] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

[22] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[23] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *International Conference on Learning Representations*, 2021.

[24] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL https://arxiv.org/abs/2301.04104.

[25] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *ICML*, 2022.

[26] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control, 2024.

[27] Tairan He, Chong Zhang, Wenli Xiao, Guanqi He, Changliu Liu, and Guanya Shi. Agile but safe: Learning collision-free high-speed legged locomotion. In *Robotics: Science and Systems (RSS)*, 2024.

[28] Mineui Hong, Kyungjae Lee, Minjae Kang, Wonsuhk Jung, and Songhwai Oh. Dynamics-aware metric embedding: Metric learning in a latent space for visual planning. *IEEE Robotics and Automation Letters*, 7(2): 3388–3395, 2022.

[29] Kai-Chieh Hsu, Vicenç Rubies-Royo, Claire J Tomlin, and Jaime F Fisac. Safety and liveness guarantees through reach-avoid reinforcement learning. *Robotics: Science and Systems*, 2021.

[30] Kai-Chieh Hsu, Allen Z. Ren, Duy P. Nguyen, Anirudha Majumdar, and Jaime F. Fisac. Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees. *Artificial Intelligence*, page 103811, 2022. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2022.103811. URL https://www.sciencedirect.com/science/article/pii/S0004370222001515.

[31] Kai-Chieh Hsu, Haimin Hu, and Jaime 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.

[32] Kai-Chieh Hsu, Duy Phuong Nguyen, and Jaime Fernàndez Fisac. Isaacs: Iterative soft adversarial actor-critic for safety. In *Learning for Dynamics and Control Conference*, pages 90–103. PMLR, 2023.

[33] Haimin Hu, Zixu Zhang, Kensuke Nakamura, Andrea Bajcsy, and Jaime Fernández Fisac. Deception game: Closing the safety-learning loop in interactive robot autonomy. In *7th Annual Conference on Robot Learning*, 2023.

[34] Brian Ichter and Marco Pavone. Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters*, 4(3):2407–2414, 2019.

[35] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[36] Konwoo Kim, Gokul Swamy, Zuxin Liu, Ding Zhao, Sanjiban Choudhury, and Steven Z Wu. Learning shared safety constraints from multi-task demonstrations. *Advances in Neural Information Processing Systems*, 36, 2024.

[37] Jingqi Li, Donggun Lee, Jaewon Lee, Kris Shengjun Dong, Somayeh Sojoudi, and Claire Tomlin. Certifiable deep learning for reachability using a new lipschitz continuous value function, 2025. URL https://arxiv.org/abs/2408.07866.

[38] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. URL https://arxiv.org/abs/1509.02971.

[39] Albert Lin, Shuang Peng, and Somil Bansal. One filter to deploy them all: Robust safety for quadrupedal navigation in unknown environments, 2024. URL https://arxiv.org/abs/2412.09989.

[40] Lars Lindemann, Alexander Robey, Lejun Jiang, Satyajeet Das, Stephen Tu, and Nikolai Matni. Learning robust output control barrier functions from safe expert demonstrations. *IEEE Open Journal of Control Systems*, 2024.

[41] David Lindner, Xin Chen, Sebastian Tschiatschek, Katja Hofmann, and Andreas Krause. Learning safety constraints from demonstrations with unknown rewards. In *International Conference on Artificial Intelligence and Statistics*, pages 2386–2394. PMLR, 2024.

[42] Huihan Liu, Yu Zhang, Vaarij Betala, Evan Zhang, James Liu, Crystal Ding, and Yuke Zhu. Multi-task interactive robot fleet learning with visual world models. In *8th Annual Conference on Robot Learning*.

[43] Huihan Liu, Shivin Dass, Roberto Martín-Martín, and Yuke Zhu. Model-based runtime monitoring with interactive imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[44] John Lygeros. On reachability and minimum cost optimal control. *Automatica*, 40(6):917–927, 2004.

[45] Kostas Margellos and John Lygeros. Hamilton–jacobi formulation for reach–avoid differential games. *IEEE Transactions on automatic control*, 56(8):1849–1861, 2011.

[46] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. *arXiv preprint arXiv:2308.10901*, 2023.

[47] I. Mitchell. A toolbox of level set methods. *http://www. cs. ubc. ca/mitchell/ToolboxLS/toolboxLS. pdf, Tech. Rep. TR-2004-09*, 2004.

[48] Ian Mitchell, Alex Bayen, and Claire J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control (TAC)*, 50(7):947–957, 2005.

[49] Naoki Morihira. dreamerv3-torch: Implementation of dreamer v3 in pytorch. https://github.com/NM512/dreamerv3-torch, 2025. Accessed: 2025-01-25.

[50] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[51] Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. IEEE, 2020.

[52] Dexter RR Scobee and S Shankar Sastry. Maximum likelihood constraint inference for inverse reinforcement learning. *International Conference on Learning Representations*, 2019.

[53] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. Learning to be safe: Deep rl with a safety critic. *arXiv preprint arXiv:2010.14603*, 2020.

[54] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E. Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6(3):4915–4922, 2021. doi: 10.1109/LRA.2021.3070252.

[55] Mukun Tong, Charles Dawson, and Chuchu Fan. Enforcing safety for vision-based controllers via control barrier functions and neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10511–10517. IEEE, 2023.

[56] Matti Vahs, Christian Pek, and Jana Tumova. Belief control barrier functions for risk-aware control. *IEEE Robotics and Automation Letters*, 2023.

[57] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[58] Ewerton R Vieira, Aravind Sivaramakrishnan, Sumanth Tangirala, Edgar Granados, Konstantin Mischaikow, and Kostas E Bekris. Morals: Analysis of high-dimensional robot controllers via topological tools in a latent space. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 27–33. IEEE, 2024.

[59] Kim P Wabersich and Melanie N Zeilinger. Predictive control barrier functions: Enhanced safety mechanisms for learning-based control. *IEEE Transactions on Automatic Control*, 68(5):2638–2651, 2022.

[60] Kim P Wabersich, Andrew J Taylor, Jason J Choi, Koushil Sreenath, Claire J Tomlin, Aaron D Ames, and Melanie N Zeilinger. Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems. *IEEE Control Systems Magazine*, 43(5):137–177, 2023.

[61] Albert Wilcox, Ashwin Balakrishna, Brijen Thananjeyan, Joseph E Gonzalez, and Ken Goldberg. Ls3: Latent space safe sets for long-horizon visuomotor control of sparse reward iterative tasks. In *Conference on Robot Learning*, pages 959–969. PMLR, 2022.

[62] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. Daydreamer: World models for physical robot learning. *Conference on Robot Learning*, 2022.

[63] H Yu, C Hirayama, C Yu, S Herbert, and S Gao. Sequential neural barriers for scalable dynamic obstacle avoidance. in 2023 ieee. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11241–11248.

[64] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

APPENDIX

A. Hyperparameters

In Section IV, we showcase results for a Recurrent State Space Model (RSSM) [22] that uses both a deterministic recurrent state h_t and stochastic component x_t.

$$\begin{aligned} h_{t+1} &= f_\phi(h_t, x_t, a_t) & \hat{x}_t &= p_\phi(\hat{x}_t \mid h_t) \\ x_t &= \mathcal{E}_\phi(x_t \mid h_t, o_t) & \hat{o}_t &= \text{dec}_\phi(\hat{o}_t \mid h_t, x_t) \end{aligned} \tag{10}$$

For the RSSM we define $z := \{h_t, x_t\}$. The RSSM shares weights between each module and is trained to minimize the KL divergence between the encoding of the current latent x and the latent $\hat{x}_t$ predicted by the transition dynamics. The RSSM additionally utilizes an observation reconstruction objective to maintain the informativeness of its latent state. We refer the readers to [22, 23, 24] for a more comprehensive overview of RSSMs. We use the implementation from [49] and list relevant hyperparameters in Table III. For Dubin's car experiments, we use a continuous stochastic latent parameterized as a 32-dimensional Gaussian. We use 32 categorical random variables with 32 classes for the high-dimensional simulation experiments.

Hyperparameter	Values
Image size	128
Optimizer	Adam
Learning rate (lr)	$1e-4$
Hidden dim	512
Dyn deterministic	512
Activation fn	SiLU
CNN depth	32
Batch size	16
Batch Length	64
Recon loss scale	1
Dyn loss scale	0.5
Representation loss scale	0.1

TABLE III: Hyperparameters for Dreamer

In Section V we train DINO-WM [64], a transformer-based world model that uses the patch-tokens of DINOv2 [50] as a representation for the latent state. Here, the DINOv2 encoder is kept frozen, and we train only the parameters of a vision transformer, which is used to predict future patch tokens $\hat{z}_{t+1}$ conditioned on a sequence of actions $a_{t-H:t}$ and tokens $z_{t-H:t}$ from the previous H timesteps.

$$z_t = \mathcal{E}_\psi(o_t) \qquad \hat{z}_{t+1} = p_\phi(z_{t-H:t}, a_{t-H:t}) \tag{11}$$

This model is trained via teacher-forcing with the following consistency loss $\mathcal{L}(\phi) = ||\mathcal{E}_\psi(o_{t+1}) - p_\phi(z_{t-H:t}, a_{t-H:t})||$. We tokenize both the wrist and front camera views, leading to two sets of image patches with embedding dimension 384. We additionally encode the action and proprioception into a 10-dimensional latent vector. We concatenate the image patches for both cameras, action embedding, and proprioception embedding for $H = 3$ frames. We use learnable positional and temporal embeddings. We pass the output from the transformer into a wrist camera, front camera, and proprioception head, which predicts the corresponding input for the next time step. These are implemented as MLP heads with three layers, a hidden dimension of 788, and a learning rate of $5e-5$. The remaining hyperparameters are the same as [64] and are showing in Table IV

Hyperparameter	Values
Image size	224
DINOv2 patch size	$(14 \times 14, 384)$
Optimizer	AdamW
Predictor lr	5e-5
Decoder lr	3e-4
Action Encoder lr	5e-4
Action emb dim	10
Proprioception emb dim	10
Batch size	16
Training iterations	100000
ViT depth	6
ViT attention heads	16
ViT MLP dim	2048

TABLE IV: Hyperparameters for DINO-WM

The Dubin's car experiments use a discrete action space, so we train the value function using DDQN [57] using the toolbox from [29]. The Q-function is implemented as a 3-layer MLP with 100-d hidden dimension. The remaining hyperparameters are shown in Table V.

Hyperparameter	Values
Optimizer	AdamW
Learning rate	1e-3
Learning rate decay	0.8
Hidden dims	$[100, 100]$
Time discount γ	0.9999
Activations	Tanh
Batch size	64
Training iterations	400000

TABLE V: Hyperparameters for DDQN HJ Reachability

For both the simulation and hardware manipulation experiments, we use DDPG [38] using the implementation from [37] using the standard discounted Bellman equation from [20].

Hyperparameter	Values
Optimizer	AdamW
Actor lr	1e-4
Critic lr	1e-3
Actor + Critic hidden dims	$[512, 512, 512, 512]$
Time discount γ	0.9999
Activations	ReLU
Batch size	512
Epochs	50

TABLE VI: Hyperparameters for DDPG HJ Reachability

To smooth the loss landscape for reachability learning, we apply $\tanh$ to the output of ℓ_μ. This is because the loss function for the failure classifier may output very different values to states that are similarly constraint violating / far from constraint violation, so long as it achieves low loss by (3). This

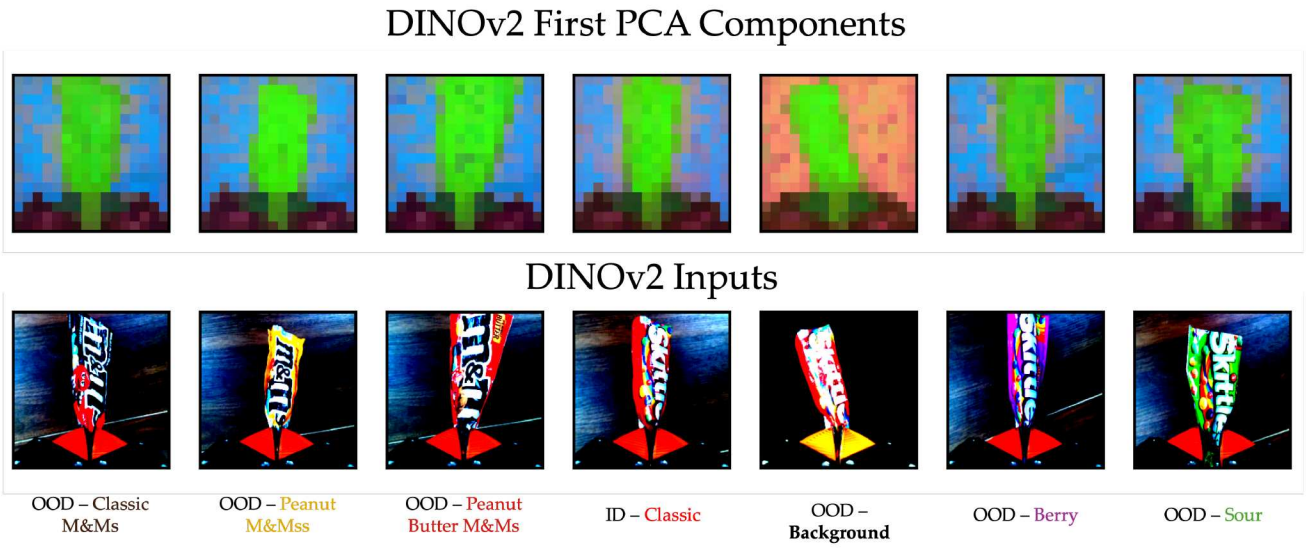

Fig. 9: Bottom: Input to DINOv2 after applying image transformations. Top: Top 3 PCA components visualized in the RGB channels. Despite the difference in bag or background color, the PCA visualization shows that all bags are similarly represented in the DINOv2 embedding space.

thresholding ensures that sufficiently safe or unsafe states are evaluated similarly by $\tanh(\ell_\mu(z))$.

For the Diffusion Policy experiments, we match the CNN-based implementation from [13] using the hyperparameters shown in Table VII. While we train for 500000 iterations, we in practice choose the best-performing checkpoint for each model.

Hyperparameter	Values
State Normalization	Yes
Action Normalization	Yes
Action Space	End Effector Delta Position
Rotation Representation	Axis Angle
Action Chunk	16
Image Chunk	2
Image Size	256
Batch size	100
Training Iterations	500000
Learning Rate	1e-4
Learning Rate Schedule	Cosine
Optimizer	AdamW

TABLE VII: Hyperparameters for Diffusion Policy

B. Common Questions

Q1: *How do we get labels for training $\mathcal{F}_{latent}$?*

When training in simulation, we used privileged data provided by the simulator. For the real-world hardware tasks, we labeled the observations from the world model training trajectories by hand. In hardware, for our 1,300 collected trajectories, this simply meant identifying a single frame in the trajectory where all subsequent timesteps were in violation of the safety constraint and all previous timesteps were safe. This took the lead author about 2.5 hours. While this process is manually intensive right now, it is (1) easier for non-experts to label observations that appear in failure instead of designing a functional representation of the failure and (2) in principle, foundation models (e.g., VLMs) could be used to annotate visually-apparent failures automatically. Furthermore, when training the failure classifier separately from the world model, one can select a smaller subset of data to label and train the failure classifier.

Q2: *How does the policy generalize?*

While an in-depth study on the generalization capabilities of world models and resulting policies is out of the scope of this work, we provide qualitative evidence to our claim in the main body that the DINOv2 [50] encoder used in DINO-WM [64] may help generalize across semantically equivalent instances (e.g., bags of different colors). In Figure 9 follow the visualization procedure from [50] and compute a principal component analysis (PCA) between all of the image patches, and show their first three components as a channel in an RGB image. Despite the difference in bag and/or background color, the PCA visualization of the DINOv2 features shows that the bags of candy are consistently represented with the color green. Although this does not reveal information beyond the first three PCA components, it suggests that the DINOv2 encoder maps these images to similar embeddings.