

Pixels to Proofs: Probabilistically-Safe Latent World Model Control via Parallel Conformal Robust MPC

Devesh Nath*, Anutam Srinivasan*, Haoran Yin*, Ruitong Jiang, Jeffrey Fang, Glen Chou

Georgia Institute of Technology

{dnath7, asrinivasan350, hyin95, rjiang77, jfang301, chou}@gatech.edu

*Equal contribution (alphabetical ordering)

[Website](#)

[Code](#)

[Video](#)

Abstract: We present SLS², a framework for safe feedback motion planning from pixels using robust model predictive control (MPC) in learned latent world models. Our approach trains an action-conditioned joint-embedding world model with compact Markovian latent states, enabling efficient gradient-based trajectory optimization through learned latent dynamics. To enforce safety for the true system despite imperfect latent predictions, we inform a GPU-accelerated system level synthesis (SLS) robust MPC scheme with conformal prediction to obtain calibrated latent error bounds and robust latent-space constraint sets. We further learn and conformalize a latent constraint checker, allowing the SLS planner to impose probabilistic safety constraints during closed-loop execution. We evaluate our method on vision-based control tasks, where it improves both goal-reaching performance and safety over latent world-model and safe-planning baselines.

Keywords: robust visuomotor control, safety verification, JEPa world models

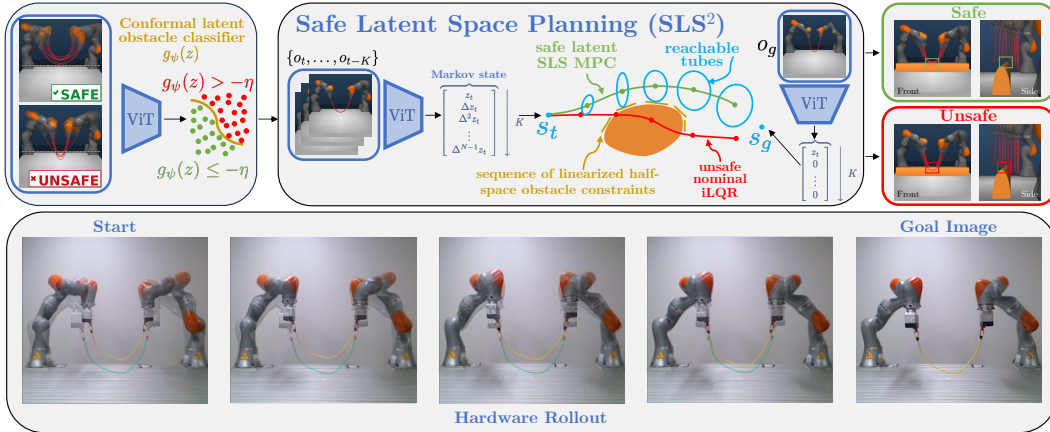


Figure 1: **Method overview.** SLS² enables robustly safe, reachability-constrained latent planning directly from pixels. *Top row:* We summarize the pipeline for our robust planner. *Bottom row:* We show a sample rollout generated by executing control actions computed by the nominal gradient-based planner on which the robust planner is built. We overlay the goal pose, shown with a green rope, on the rollout to highlight the system’s progress toward the goal state.

1 Introduction

Safe control is a central challenge for reliable open-world robot deployment. Most safety-critical control methods [1–3] rely on a well-defined state representation on which planning, control, and verification can be performed. However, in the real world, obtaining such a representation is itself nontrivial. This is especially true for deformable objects and tasks where the state is only implicitly defined through perception. Even when a state representation is known, estimating the state from

raw observations, such as images, can require extensive offline training data with labeled correspondences between visual inputs and ground-truth states – data that is often impractical to obtain [4–7].

Recent advances in representation learning and world modeling [8, 9] offer a promising alternative. Learned world models can infer latent state representations directly from observations and learn dynamics in these latent spaces, enabling prediction and planning without requiring manually specified state variables. These models have shown considerable potential for robotic decision-making [10, 11], particularly in settings where explicit state estimation is difficult. However, ensuring that these latent space control policies yield safe behavior under the true system dynamics is difficult. Latent representations are often high-dimensional, their learned dynamics may be inaccurate, and task constraints expressed in the physical or observation space must be translated into the latent space. These issues complicate the use of classical safety-critical control tools in the latent space, which typically assume low-dimensional state spaces and known, calibrated disturbance bounds. Recent work has explored deep learning for approximating latent safety certificates [12, 13]; however, these methods often rely on heuristic approximations and do not provide sound, calibrated reachability guarantees.

To exploit the advances of world modeling while maintaining rigorous safety assurances, we propose SLS² (Safe Latent Space System Level Synthesis), a scalable robust model predictive control (MPC) framework for safe planning in learned latent spaces based on the SLS framework [14]. Our approach introduces training objectives and optimization strategies that improve gradient-based trajectory optimization through learned latent dynamics. To be robust to prediction error in the latent dynamics, we plan with these learned models using scalable reachability-constrained robust MPC based on SLS. This is made computationally feasible through GPU parallelization, which enables efficient trajectory optimization coupled with local uncertainty propagation around the optimized plan. To enforce safety constraints, we train a constraint checker in the latent space and incorporate it within robust MPC. To enable probabilistic constraint satisfaction, we use conformal prediction (CP) to calibrate bounds on both latent dynamics error and learned constraint-checker error. This ensures that the reachable sets computed by SLS contain the true reachable latent states with high probability, enabling robust safety despite these errors. Specifically, our contributions are:

- We learn compact Markovian latent world models from pixels that are accurate under multi-step rollout and amenable to efficient gradient-based trajectory optimization.
- We conformalize latent dynamics errors and in-domain latent support, producing calibrated uncertainty sets for robust planning. We further learn and conformalize latent safety classifiers, enabling task constraints to be enforced directly during latent space planning.
- We show how to use CP-calibrated uncertainty sets in GPU-accelerated SLS to scalably perform reachability-constrained robust MPC in high-dimensional learned latent spaces.
- We validate nominal trajectory optimization and robust MPC with our learned models on both simulated and hardware vision-based control tasks, spanning navigation, pick-and-place, and deformable object manipulation. Our method improves safety and goal-reaching performance over world-model and safe-planning baselines.

2 Related Work

World Model World models learn predictive representations of dynamics, enabling agents to reason about future outcomes through imagined rollouts [8, 9, 15–18]. Early latent world models paired representation learning with reconstruction-based prediction and policy learning in imagination [9, 15, 16], while TD-MPC and TD-MPC2 combine latent dynamics with online model-predictive control for continuous control [11, 19]. More recently, joint-embedding predictive architectures (JEPAs) have emerged as a promising approach to representation-space world modeling by predicting future embeddings rather than pixels [20–23]. In robotics and offline goal-conditioned planning, PLDM learns reward-free latent dynamics for test-time planning [24], while DINO-WM predicts future DINOv2 visual features from offline action-conditioned trajectories for zero-shot planning without reward supervision [10, 25]. Large-scale generative world models and interactive simulators further

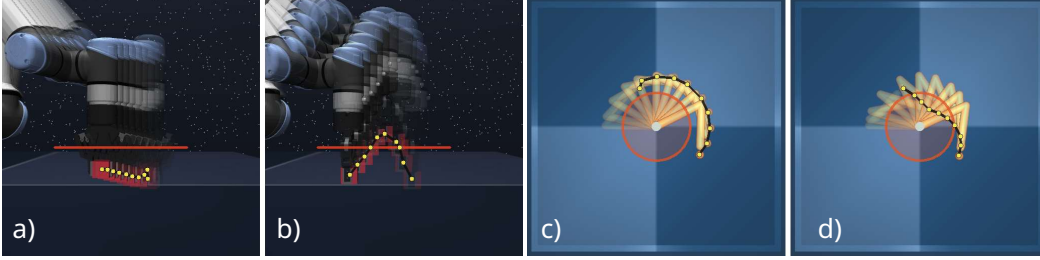


Figure 2: **Robust planning. Cube (a-b) and Reacher (c-d).** SLS² maps pixels directly to robustly-safe actions via latent robust MPC, ensuring robust constraint satisfaction (red) compared to unconstrained baselines. **a, b)** The Cube task with and without SLS² enforcing the height threshold. **c, d)** The Reacher task with (c) and without (d) SLS² enforcing the angle constraint on the second joint.

demonstrate the promise of learned predictive models for controllable environments [26–29]. Our work builds on LeWorldModel (Le-WM), an end-to-end JEPA world model that learns from raw pixels via next-embedding prediction with Gaussian embedding regularization and without explicitly predicting future images, then performs latent-space MPC for goal-reaching control [30]. We introduce training modifications that improve both predictive accuracy and gradient-based planning.

Safe Latent Planning Safety filters enforce constraints by minimally modifying candidate actions from a nominal controller. Classical approaches such as CBF-QPs, HJ reachability, and predictive safety filters offer principled safety guarantees but typically require known state representations, known dynamics, and hand-designed safe or failure sets [1, 2, 31–34]. Recent data-driven methods reduce this burden by learning barrier functions, output-feedback certificates, distributional safety constraints from demonstrations and offline data, or by performing uncertainty-aware model-based control with learned dynamics [35–45]. Closest to our setting, safe latent-space methods extend safety filtering to high-dimensional visuomotor control. In-distribution barrier functions learn CBF-like filters in latent space from safe demonstrations [12], while latent safety filters approximate HJ reachability in learned world-model latents to reason about hard-to-specify failures [13]. Follow-up work improves latent safety filtering with uncertainty-aware OOD detection and conformal calibration [46], smooth latent CBFs [47], and latent barrier certificates learned jointly with world models [48]. Latent Policy Barrier treats the expert demonstration manifold as an implicit safety barrier and corrects actions at inference time using learned latent dynamics [49]. These methods show that latent representations can enable safety reasoning from images. However, most existing methods operate as policy filters without formal guarantees, safety predictors that do not perform control [50], or in-distribution recovery mechanisms around a *given* base policy. In contrast, we directly synthesize safety-constrained feedback policies and motion plans through reachability-informed latent space control, while providing calibrated probabilistic guarantees via SLS and conformal prediction.

3 Preliminaries and Problem Statement

We consider an unknown partially observed dynamical system described by an *unknown* underlying state representation $x_k \in \mathcal{X}$ and control $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, and observations $o_k \in \mathcal{O} \subseteq \mathbb{R}^{n_o}$:

$$x_{k+1} = g(x_k, u_k), \quad o_k = h(x_k), \quad (1)$$

where $o_k \in \mathcal{O}$ are RGB images. In our setting, neither the true state x_k nor its state representation \mathcal{X} is available to the planner. g and h are also unknown; we only observe image measurements $o \in \mathcal{O}$.

Problem Statement. We consider safe feedback motion planning from image observations using a learned latent world model. We are given an offline dataset of image transitions $\mathcal{D} := \{(o_t, u_t, o_{t+1})\}$ collected from task-relevant rollouts. The true system evolves in an unknown state space \mathcal{X} , while the planner only observes images $o \in \mathcal{O}$ from (1). We wish to learn a compact latent representation where prediction, planning, and reachability-based safety verification are tractable.

In this paper, we train a visual encoder $\text{enc}_\theta : \mathcal{O} \rightarrow \mathbb{R}^{n_z}$, with $z_t = \text{enc}_\theta(o_t)$. Using enc_θ , we construct a Markovian latent state $s_t^z = [z_t, \Delta z_t, \dots, \Delta^K z_t] \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$, where $\Delta z_t = z_t - z_{t-1}$,

from a history of encoded images. The learned latent dynamics $f_\phi : \mathcal{S} \times \mathcal{U} \rightarrow \mathcal{S}$ predict the next Markovian state $\hat{s}_{t+1}^z = f_\phi(s_t^z, u_t)$. As f_ϕ can be inaccurate, we model its prediction error as $s_{t+1}^z = f_\phi(s_t^z, u_t) + e_t$, where $\mathcal{E} \subseteq \mathbb{R}^{n_s}$ is calibrated from a held-out subset of \mathcal{D} using CP to contain the true prediction error e_t with a user-specified probability $1 - \delta \in (0, 1)$. Specifically, we solve:

Problem 1. *Learning a planning- and verification-friendly latent world model.* Using \mathcal{D} , learn enc_θ and f_ϕ such that the latent state s_t^z is predictive under actions, the learned dynamics are accurate under multi-step rollout, differentiable, and compatible with gradient-based trajectory optimization. Construct uncertainty sets $\mathcal{E}(s, u) \subseteq \mathbb{R}^{n_s}$ such that the prediction error e at a particular (s, u) satisfies $e \in \mathcal{E}(s, u)$ with probability $1 - \delta$. Finally, learn a latent safety score $c_\psi : \mathbb{R}^{n_s} \rightarrow \mathbb{R}$, where larger values indicate safer states, and define the constraint score $g_\psi(s) := -c_\psi(\Pi_z s)$, where Π_z extracts the zeroth-order latent embedding.

Problem 2. *Reachability-informed robust MPC in latent-space.* Given f_ϕ , \mathcal{E} , and a latent safe set $\mathcal{S}_{\text{safe}} := \{s \in \mathcal{S} \mid g_\psi(s) \leq -\eta\}$ for $\eta \in \mathbb{R}$, optimize a nominal length- T latent trajectory $\zeta := \{\zeta_k\}_{k=0}^T$ and controls $\mathbf{v} := \{v_k\}_{k=0}^{T-1}$ satisfying $\zeta_{k+1} = f_\phi(\zeta_k, v_k)$, together with a causal state-feedback controller $\pi := (\pi_0, \dots, \pi_{T-1})$, $\pi_k : \mathcal{S}^{k+1} \rightarrow \mathcal{U}$, that stabilizes the true latent dynamics $s_{k+1}^z = f_\phi(s_k^z, \pi_k(s_{0:k}^z)) + e_k$ about (ζ, \mathbf{v}) , where e_k is the true prediction error realized by executing on the true system (1). This MPC policy should produce reachable tubes \mathcal{R}_k^s and \mathcal{R}_k^u such that, with probability at least $1 - \delta$, $s_k^z \in \mathcal{R}_k^s \subseteq \mathcal{S}_{\text{safe}}$ for all $k = 0, \dots, T$ and $\pi_k(s_{0:k}^z) \in \mathcal{R}_k^u \subseteq \mathcal{U}$ for all $k = 0, \dots, T - 1$.

4 Methodology

Our method, SLS² (Fig. 1), trains a Markovian latent model and constraint classifier (Sec. 4.1), conformalizes its uncertainty (Sec. 4.2), and incorporates the resulting calibrated uncertainty sets into SLS-based robust MPC for probabilistically-safe feedback motion planning (Sec. 4.3).

4.1 Training World Models Friendly for Gradient-Based Robust MPC

We use the given offline dataset \mathcal{D} to train an action-conditioned latent world model following a Le-WM-style [30] joint-embedding predictive architecture. Given an image observation o_t , an encoder enc_θ maps the image into a compact latent embedding $z_t = \text{enc}_\theta(o_t)$. The encoder is implemented as a ViT backbone trained from scratch, followed by an MLP projector that maps the encoder output to the latent embedding space. Before encoding, image observations are resized to a fixed resolution, normalized, and organized into short temporal windows sampled from offline trajectories. Actions are normalized using the dataset mean and standard deviation. To model dynamics, we construct a Markov latent state by augmenting the current embedding with finite-difference latent features:

$$s_t^z = [z_t, \Delta z_t, \dots, \Delta^K z_t], \quad \Delta z_t = z_t - z_{t-1}, \quad (2)$$

where K denotes the Markov order. The latent dynamics predictor f_ϕ is an MLP that takes the current Markov latent state and action as input and predicts the next Markov latent state:

$$\hat{s}_{t+1}^z = f_\phi(s_t^z, a_t). \quad (3)$$

For multi-step training, the predictor is rolled out autoregressively over a horizon H , where each predicted latent state is fed back into the model for the next prediction step $\hat{s}_{t+h+1}^z = f_\phi(\hat{s}_{t+h}^z, a_{t+h})$ for $h = 0, \dots, H - 1$. The main training loss is a multi-step latent prediction objective. At each prediction step, the predicted Markov latent state is compared against the target Markov latent state computed from the encoded ground-truth future observations:

$$\mathcal{L}_{\text{pred}} = \frac{1}{H} \sum_{h=1}^H \|\hat{s}_{t+h}^z - s_{t+h}^z\|_2^2. \quad (4)$$

We follow Le-WM [30] and prevent representational collapse via the SIGReg loss [51] and additionally encourage consecutive latent velocity vectors to align via temporal straightening [52]:

$$\mathcal{L}_{\text{sig}} = \text{SIGReg}(\{z_t\}_{t=1}^T), \quad (5) \quad \mathcal{L}_{\text{str}} = \frac{1}{T-2} \sum_{t=2}^{T-1} \left(1 - \frac{(z_t - z_{t-1})^\top (z_{t+1} - z_t)}{\|z_t - z_{t-1}\|_2 \|z_{t+1} - z_t\|_2}\right). \quad (6)$$

The full training objective is therefore $\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda_{\text{sig}} \mathcal{L}_{\text{sig}} + \lambda_{\text{str}} \mathcal{L}_{\text{str}}$.

Obstacle Classifier. For constrained tasks, we train a latent obstacle classifier to detect whether a predicted latent state violates task constraints. For each task, we build a balanced dataset of obstacle and non-obstacle observations by sampling task-relevant configurations, labeling them with the known simulator constraint, and rendering them as RGB images. This yields positive and negative constraint examples without relying on rollout failures. While the labeling rule is task-specific, the dataset has a common form across tasks, $\mathcal{D}_{\text{obs}} = \{(o_i, y_i)\}_{i=1}^N$, with $y_i \in \{0, 1\}$, where $y_i = 0$ denotes a constraint-violating image and $y_i = 1$ denotes a non-violating image.

We train the classifier in the latent space rather than in pixel space. Using the frozen encoder, each image is mapped to $z_i = \text{enc}_\theta(o_i)$. A small MLP classifier $c_\psi : \mathbb{R}^{n_z} \rightarrow \mathbb{R}$ maps z_i to a signed score $c_i = c_\psi(z_i)$, where positive scores indicate non-violating states and negative scores indicate constraint-violating states. Thus, binary labels $\{0, 1\}$ are mapped to the signed labels $\{-1, +1\}$, with $y_i = 0$ mapped to -1 and $y_i = 1$ mapped to $+1$, and the classifier is trained with a hinge loss,

$$\mathcal{L}_{\text{obs}} = \frac{1}{N} \sum_{i=1}^N \max(0, m - \tilde{y}_i c_\psi(z_i)), \quad (7)$$

where m is the classification margin. We normalize latent features using the training-split mean and standard deviation, then calibrate the classifier threshold on a held-out split. Denote $\mathcal{I}_{\text{cal}}^- := \{i \in \mathcal{I}_{\text{cal}} : y_i = 0\}$ as the violating calibration indices. For each $i \in \mathcal{I}_{\text{cal}}^-$, we compute the nonconformity score $r_i = \max(0, c_\psi(z_i))$. We then choose a conformal threshold η as the empirical $(1 - \delta)$ quantile of these nonconformity scores. At planning time, a latent Markov state s is treated as safe if

$$c_\psi(\Pi_z s) \geq \eta, \quad \iff \quad g_\psi(s) := -c_\psi(\Pi_z s) \leq -\eta. \quad (8)$$

This yields a conservative obstacle classifier used during constrained planning: the world model predicts latent rollouts, and the classifier detects whether they enter the learned obstacle set.

4.2 Latent Uncertainty Quantification via Conformal Prediction (CP)

To ensure closed-loop safety for planning, we need two additional sources of uncertainty to be bounded: 1) latent dynamics prediction error and 2) in-distribution domain calibration. First, to quantify latent prediction error, we use split CP [53, 54]. Let $\mathcal{D}_{\text{cal}} = \{(s_i, u_i, s_i^+)\}_{i=1}^n$ be a held-out set of latent transitions obtained by encoding a subset of \mathcal{D} , and define residuals $e_i := s_i^+ - f_\phi(s_i, u_i)$. To capture state-action-dependent error geometry, we train an uncertainty model $\Sigma_\psi : \mathcal{S} \times \mathcal{U} \rightarrow \mathbb{S}_{++}^{n_s}$ using the multivariate Gaussian negative log-likelihood loss

$$\mathcal{L}_{\text{MGNNL}} = \frac{1}{2} (e_i^\top \Sigma_\psi(s_i, u_i)^{-1} e_i + \log \det \Sigma_\psi(s_i, u_i)). \quad (9)$$

The covariance model is not assumed to be calibrated; it only provides a local ellipsoid shape for the prediction error. We calibrate its scale using CP with scores $r_i := e_i^\top \Sigma_\psi(s_i, u_i)^{-1} e_i$. Let $r_{(1)} \leq \dots \leq r_{(n)}$ be the sorted scores. For trajectory-level failure probability δ over horizon T , set $\bar{\delta} := \delta/T$ and $q := r_{(\lceil (n+1)(1-\bar{\delta}) \rceil)}$, with $q = +\infty$ if the index exceeds n . For any latent state-action pair (s, u) , the calibrated one-step error set is

$$\mathcal{E}(s, u) := \{e \mid e^\top \Sigma_\psi(s, u)^{-1} e \leq q\}. \quad (10)$$

Writing $\Sigma_\psi(s, u) = L_\psi(s, u)L_\psi(s, u)^\top$ using its Cholesky factor L_ψ , this is equivalently $\mathcal{E}(s, u) = \sqrt{q} L_\psi(s, u) \mathcal{B}^{n_s}$, where $\mathcal{B}^{n_s} := \{s \in \mathbb{R}^{n_s} \mid \|s\|_2 \leq 1\}$.

Second, we conformalize an in-domain latent region to keep the planner within the support of the calibration data. Since the SIGReg loss encourages the latent embeddings to follow an approximately Gaussian distribution [51], we define the in-domain set as a calibrated Gaussian density sublevel set. Concretely, we fit an ellipsoid to the calibration embeddings and choose its conformal threshold so that it contains most calibration latents with the desired coverage. Let μ and Σ_{ID} be the empirical mean and covariance of calibration latents in \mathcal{D}_{cal} , and define $d_i := (s_i - \mu)^\top \Sigma_{\text{ID}}^{-1} (s_i - \mu)$. With q_{ID} the $(1 - \alpha_{\text{ID}})$ split conformal quantile of $\{d_i\}$, the calibrated in-domain set is

$$\mathcal{I} := \{s \mid (s - \mu)^\top \Sigma_{\text{ID}}^{-1} (s - \mu) \leq q_{\text{ID}}\}. \quad (11)$$

During planning, we require $\mathcal{R}_k^s \subseteq \mathcal{I}$ for all k . This controls support mismatch, while the CP guarantee itself assumes that closed-loop residual scores are exchangeable with the calibration scores. In practice, we support this assumption by collecting calibration transitions from representative randomized rollouts that vary initial states and goals.

4.3 Conformalized Robust MPC via System Level Synthesis (SLS)

To synthesize robust latent feedback policies, we use system level synthesis (SLS) [14]. SLS enables efficient robust MPC by optimizing over closed-loop system responses rather than feedback gains directly, and admits fast GPU-parallel implementations for high-dimensional systems [55]. This is particularly important in latent-space planning, where the latent state dimension can be high. We build on [56], using the conformal calibration in Sec. 4.2 to define the disturbance sets to be propagated by SLS. We model the learned latent dynamics with calibrated additive uncertainty,

$$s_{k+1} = f_\phi(s_k, u_k) + E_k w_k, \quad w_k \in \mathcal{B}^{n_s}, \quad (12)$$

where $E_k = \sqrt{q} L_\psi(s_k, u_k)$, recovering $E_k \mathcal{B}^{n_s}$ as the CP-calibrated, ellipsoidal one-step latent prediction-error set defined in (10). Given the current latent state \bar{s}_0 , SLS optimizes a nominal trajectory $\zeta := \{\zeta_k\}_{k=0}^T$, controls $\mathbf{v} := \{v_k\}_{k=0}^{T-1}$, and closed-loop response matrices Φ^s, Φ^u . The responses map disturbances to deviations from the nominal state and control trajectories. Around (ζ_k, v_k) , let $A_k := \nabla_s f_\phi(\zeta_k, v_k)$ and $B_k := \nabla_u f_\phi(\zeta_k, v_k)$. From Sec. 4.2, if $\Sigma_\psi(\zeta_k, v_k) = L_\psi(\zeta_k, v_k) L_\psi(\zeta_k, v_k)^\top$, we set

$$E_k := \sqrt{q} L_\psi(\zeta_k, v_k). \quad (13)$$

Thus SLS propagates the CP-calibrated ellipsoidal model errors through the horizon to construct reachable tubes and constraint tightenings. The resulting CP-calibrated SLS MPC problem is

$$\min_{\zeta, \mathbf{v}, \Phi^s, \Phi^u} J(\zeta, \mathbf{v}) + J_f(\zeta_T) + H(\Phi^s, \Phi^u) \quad (14a)$$

$$\text{s.t. } \zeta_{k+1} = f_\phi(\zeta_k, v_k), \quad \zeta_0 = \bar{s}_0, \quad k = 0, \dots, T-1, \quad (14b)$$

$$\Phi_{k+1, j}^s = A_k \Phi_{k, j}^s + B_k \Phi_{k, j}^u, \quad 0 \leq j < k, \quad k = 0, \dots, T-1, \quad (14c)$$

$$\Phi_{j+1, j}^s = E_j, \quad j = 0, \dots, T-1, \quad (14d)$$

$$g_i(\zeta_k, v_k) + b_i + \tau_{i, k} \leq 0, \quad i = 1, \dots, n_c, \quad k = 0, \dots, T, \quad (14e)$$

$$\mathcal{R}_k^s \subseteq \mathcal{I}, \quad k = 0, \dots, T, \quad (14f)$$

where $\tau_{i, k} := \sum_{j=0}^{k-1} \|\nabla_s g_i(\zeta_k, v_k) \Phi_{k, j}^s + \nabla_u g_i(\zeta_k, v_k) \Phi_{k, j}^u\|_2$, and $\mathcal{R}_k^s := \zeta_k \oplus \bigoplus_{j=0}^{k-1} \Phi_{k, j}^s \mathcal{B}^{n_s}$, $\mathcal{R}_k^u := v_k \oplus \bigoplus_{j=0}^{k-1} \Phi_{k, j}^u \mathcal{B}^{n_s}$ are the state and control reachable tubes, respectively. Here, \bigoplus and \oplus denote Minkowski sums, and b_i is a constraint-index-dependent bias equal to η for obstacle constraints and zero for control constraints. For sets \mathcal{A} and \mathcal{B} , $\mathcal{A} \oplus \mathcal{B} = \{a+b \mid a \in \mathcal{A}, b \in \mathcal{B}\}$. The objective (14a) combines nominal task cost, terminal cost, and a tube-size regularizer $H(\Phi^s, \Phi^u)$ (see App. B for details). The constraint (14b) enforces nominal feasibility under the latent dynamics, and (14c)-(14d) enforce that SLS correctly propagates disturbance, with the CP-calibrated disturbance set injected through (14d). The constraints (14e) are tightened by reachable tube margins $\tau_{i, k}$, enforcing robust satisfaction of the learned latent safety constraints over the full reachable tube, and (14f) keeps the reachable tube inside the conformalized in-domain latent region (11). For instance, with $g_\psi(s) := -c_\psi(\Pi_z s)$, the obstacle constraint $g_\psi(s_k) \leq -\eta$ becomes $g_\psi(\zeta_k) + \eta + \tau_k \leq 0$ at each step k to enforce *robust* constraint satisfaction. This CP-informed SLS scheme provides the following robustness guarantee (proof in App. C):

Theorem 1 (Probabilistic containment of conformal SLS tubes). *Consider the latent closed-loop dynamics $s_{k+1} = f_\phi(s_k, u_k) + e_k$, for $k = 0, \dots, T-1$. Let q be the split conformal quantile of the calibration scores $r_i = e_i^\top \Sigma_\psi(s_i, u_i)^{-1} e_i$ and $e_i := s_i^+ - f_\phi(s_i, u_i)$, computed at per-step misscoverage $\bar{\delta} = \delta/T$. For each closed-loop transition, define the test score $r_k^{\text{test}} := e_k^\top \Sigma_\psi(s_k, u_k)^{-1} e_k$. Assume that, for each k , r_k^{test} is exchangeable with the calibration scores. Let $\Sigma_\psi(s, u) = L_\psi(s, u) L_\psi(s, u)^\top$ and define the local conformal error set $\mathcal{E}(s, u) := \sqrt{q} L_\psi(s, u) \mathcal{B}^{n_s}$ as defined in (10). Suppose the SLS controller is synthesized with disturbance matrices E_k such that the realized conformal error sets are contained in the SLS disturbance sets, $\mathcal{E}(s_k, u_k) \subseteq E_k \mathcal{B}^{n_s}$, for $k = 0, \dots, T-1$. If the SLS controller is robustly correct for all disturbances $w_k \in \mathcal{B}^{n_s}$ in $\Delta s_{k+1} = A_k \Delta s_k + B_k \Delta u_k + E_k w_k$, where $\Delta s_k := s_k - \zeta_k$ and $\Delta u_k := u_k - v_k$, then $\mathbb{P}[(s_k \in \mathcal{R}_k^s, \forall k = 0, \dots, T) \wedge (u_k \in \mathcal{R}_k^u, \forall k = 0, \dots, T-1)] \geq 1 - \delta$.*

In addition to robust MPC, the latent dynamics in (3), obtained via Sec. 4.1, can also be used for gradient-based nominal trajectory optimization by removing the reachability-related terms in (14) and retaining only the nominal latent space dynamics constraints in (14b).

Table 1: Ablation study on Reacher. The reference setting is the full method described in Sec. 4. World-model ablations keep the planner fixed and change only the latent model/training component. The planner ablation keeps the world model fixed and changes only the MPC optimizer.

Ablation	Variant	Success (%) \uparrow	Min. qpos dist. \downarrow	Final qpos dist. \downarrow	Solve time / step (ms) \downarrow
Reference	Full model + iLQR	83.50 \pm 4.06	0.377 \pm 0.119	0.497 \pm 0.152	426.35 \pm 17.52
World model	Non-Markov history-2 + iLQR	1.00 \pm 1.22	1.786 \pm 0.088	2.080 \pm 0.087	67.00 \pm 3.01
	Single-step loss + iLQR	24.50 \pm 4.58	1.283 \pm 0.138	1.746 \pm 0.205	578.33 \pm 16.56
Planner	Full model + CEM	57.00 \pm 1.87	0.380 \pm 0.100	0.511 \pm 0.106	67.00 \pm 0.94

Table 2: Nominal planning performance across benchmark tasks. Success is reported in percentage, and Dist. denotes minimum distance to the goal.

Task	Method	Success (%) \uparrow	Dist. \downarrow	Task	Method	Success (%) \uparrow	Dist. \downarrow
Reacher	Ours	83.5 \pm 4.06	0.377 \pm 0.119	Rope	Ours	93.75 \pm 2.71	0.020 \pm 0.002
	Le-WM	17.0 \pm 7.6	1.305 \pm 1.109		Le-WM	13.0 \pm 4.3	0.093 \pm 0.015
	DINO-WM	24.5 \pm 6.2	0.533 \pm 0.115		DINO-WM	25.0 \pm 7.91	0.058 \pm 0.002
	PLDM	40.5 \pm 9.9	0.821 \pm 0.158		PLDM	7.0 \pm 5.34	0.130 \pm 0.019
Cube	Ours	91.5 \pm 4.2	0.051 \pm 0.055	Push-T	Ours	51.54 \pm 4.40	29.86 \pm 35.55
	Le-WM	28.5 \pm 3.8	0.133 \pm 0.023		Le-WM	1.5 \pm 2.00	113.61 \pm 10.25
	DINO-WM	69.0 \pm 2.95	0.054 \pm 0.048		DINO-WM	2.52 \pm 0.04	112.58 \pm 3.36
	PLDM	28.75 \pm 6.62	0.145 \pm 0.005		PLDM	1.50 \pm 1.37	126.07 \pm 5.26

5 Experiments

To show the utility of our world model learning method (Sec. 4.1) for *nominal gradient-based planning* and the safety assurances provided by our *robust MPC planner* (SLS², Sec. 4.3), we evaluate on four tasks: Reacher [57], OGBench Cube [58], Push-T [10], and a bimanual rope manipulation task.

We compare our gradient-based nominal planner against world-model planning baselines in Sec. 5.1, and compare SLS² against safe latent-space planning baselines in Sec. 5.2. For nominal planning, as discussed in Sec. 4.3, we use GPU-SLS [55] without tubes or constraints (which simplifies to iLQR [59]). For robust planning, we use SLS². We compare against three latent-space world-model planning baselines: Le-WM [30], DINO-WM [10], and PLDM [24]. For safe latent space planning, we compare against two safety-focused baselines: HJ-filtered [13] and LPB [49], as well as two non-robust ablations of SLS² (i.e., constrained iLQR without tubes or constraint (11)).

All results are run on an Intel i9-14900K CPU, 64 GB RAM, with an NVIDIA RTX 4090 GPU. For all tasks, planning performance is evaluated on held-out test trajectories. To benchmark the nominal planner, we evaluate over five random seeds, with 40 evaluation episodes per seed. Unlike prior works that evaluate shorter-horizon planning segments, we require the planner to solve the complete task from the initial state to the corresponding terminal goal, final state, or task-specific completion condition. This makes the benchmark substantially more difficult, since errors accumulate over the full task horizon rather than only over a short local segment. For constrained tasks, we evaluate on 35 randomized start and goal conditions per task. Details on implementation and setup are in App. D.

We report metrics according to the task setting. For unconstrained tasks, we report the task success rate (higher is better) and the minimum distance to the goal across evaluation trials (lower is better). For constrained tasks, we also report the safety rate (adherence to constraints) and the robust success rate (when both constraint satisfaction and goal-reach conditions are met).

5.1 Nominal Planning

Reacher On Reacher, our nominal planner achieves the strongest performance, reaching 83.50% success. Detailed numbers can be found in Table 2. The ablations in Table 1 show that this improvement is not due to a single component: using a non-Markov history-based predictor severely degrades performance, indicating that the learned Markov latent state is important for planning, while removing the multi-step dynamics loss also weakens goal reaching by reducing rollout accuracy. Replacing iLQR with CEM further lowers success, suggesting that although sampling-based planning can find reasonable trajectories, gradient-based planning better exploits the local linearity of our learned latent dynamics.



Figure 3: **Nominal planning; Push-T.** Nominal planning with the T successfully reaching the target. Our planner executes complex maneuvers (including making and breaking contact with the T), to successfully reach the goal state. A denser set of keyframes is given in Fig. 8.

Cube For OGBench Cube, we initialize each planning from an oracle-grasped state, so the experiment isolates the post-grasp transport problem of moving the cube to the target. Under this setup, our nominal planning achieves the strongest nominal planning performance, reaching 91.5% success, while the best baseline, DINO-WM, only reaches 69.0%. Detailed numbers can be found in Table 2. This indicates that the learned Markov latent dynamics together with gradient-based planning more reliably transports the grasped object to the target than prior world-model planners.

Push-T Push-T is the most difficult nominal planning task in our evaluation, as it requires long-horizon contact-rich pushing with accurate control of both block position and orientation. Thus, all methods achieve lower performance compared with the other tasks. Nevertheless, our method maintains the strongest performance, achieving $51.54 \pm 4.40\%$ success and the lowest minimum block-position error of 29.86 ± 35.55 pixels. Figure 3 provides a sample rollout demonstrating the extensive maneuvers taken by our planner to successfully push the T to the goal. In contrast, DINO-WM and PLDM achieve only $2.52 \pm 0.04\%$ and $1.50 \pm 1.37\%$ success, respectively, with much larger minimum distances. These results show that, even when the task difficulty causes overall success rates to decrease, our learned Markov latent dynamics and gradient-based planner still provide the most reliable planning signal for contact-rich manipulation. See App. E for more Push-T baselines.

Rope Lastly, we evaluate on a customized two-manipulator rope task in MuJoCo, where two 7-DoF KUKA iiwa arms are mounted on opposite sides of a table and hold a rope between their end-effectors (Figure 1). Since this task is not a standard benchmark, we define a three-dimensional task state using the rope reach, height, and width, which determines the target rope geometry through symmetric target positions for the two attachment points. In this task, we seek to manipulate the deformable rope from the initial state to the desired rope configuration from the goal image. Under this setting, our nominal planning achieves 93.75% success, while the strongest baseline, DINO-WM, reaches only 25.0%. Table 2 demonstrates that the learned Markov latent dynamics and gradient-based planner are especially effective in the bimanual rope setting, where small errors from either arm can significantly alter the final rope geometry.

Hardware Rope To validate our nominal planner, we replicated the rope task in the real world. Hardware implementation details are in Appendix D.6. In our hardware results, our planner successfully stretches the rope from the start image (MPC step 0) to the goal image (MPC step 40) in Figure 4(a). In another trial (Figure 12, Appendix E), our latent space planner successfully moved the rope forward. Figure 4(b) and (c) plot the task space error between the configuration at each MPC step and the goal configuration and the error between the latent vector and goal latent vector, indicating a clear correlation between latent space goal error and task space error.

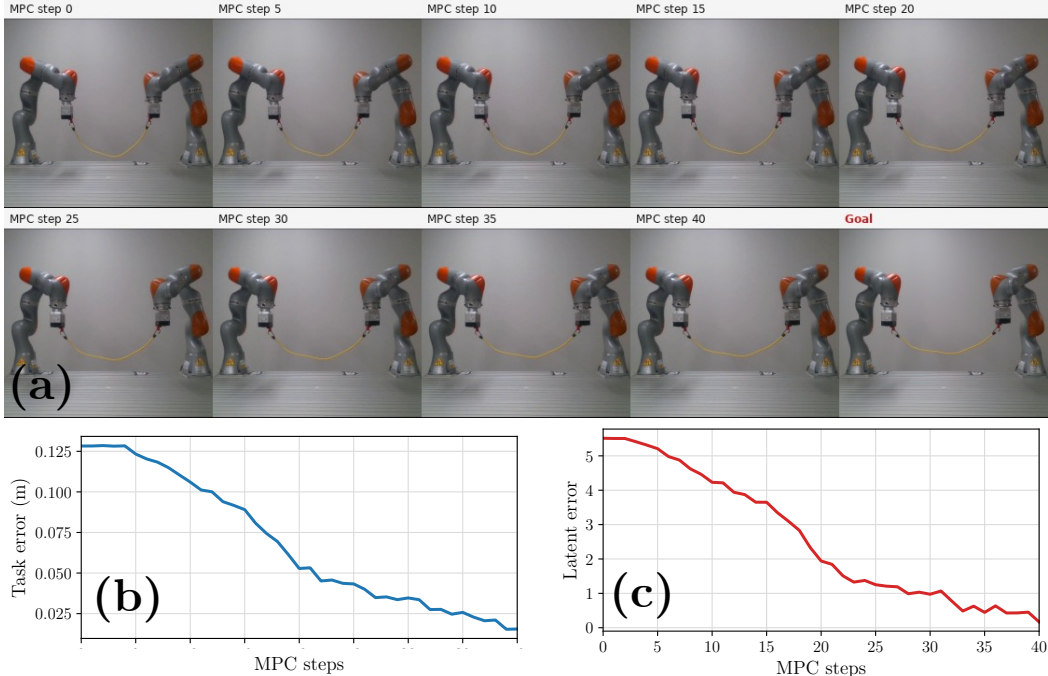


Figure 4: **Hardware Rope.** (a) Time-lapse of iLQR-based long-horizon planning on real hardware directly from pixels without state estimation. In this example, the bimanual setup has to stretch the rope to reach the goal image. (b) Task and (c) latent space error plots demonstrating convergence to the desired goal state on hardware.

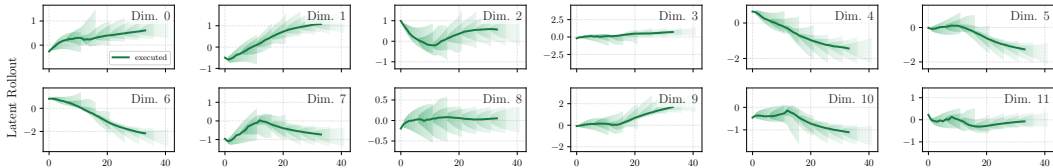


Figure 5: **Rope.** Latent state rollout (dark green), with the projected robust tubes from SLS (shaded).

5.2 Robustly-Constrained Safe Latent Space MPC

Rope For the rope task, we introduce an ellipsoidal obstacle in the center of the task space (details in Appendix D) that the rope must avoid while moving from the start position to a goal position. In Table 3, we find that SLS² is the only method that always avoids the obstacle despite perturbations due to latent model error (Figure 1). Conversely, the value function in the HJ-filtered baseline fails to predict safe and unsafe states, colliding with the ellipsoid. Similarly, the constrained iLQR baselines show that non-robust iLQR incurs safety violations by not accounting for modeling errors. Furthermore, SLS² has the highest rate for robust task completion with a rate of 88.57%. The rollout in the latent space (Figure 5) shows that our trajectories are contained within the projected tubes. Lastly, Figure 6 provides a t-SNE [60] visualization of the rollout against the obstacle region, demonstrating the robust but not overly conservative tubes.

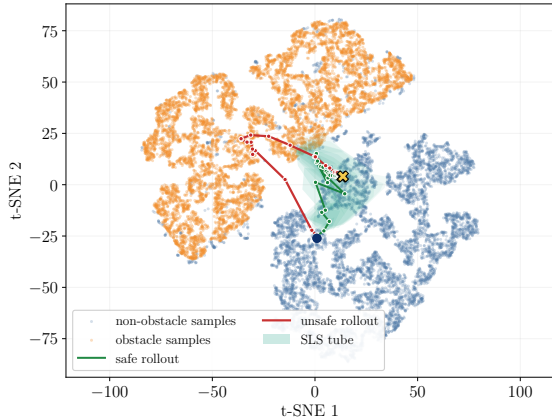


Figure 6: **Robust planning; Rope.** t-SNE visualization of the rollout. Our tubes stay predominantly in the safe region; the small overlap with the obstacle region is due to the lossy 2D t-SNE projection.

Table 3: We evaluate SLS² against baseline approaches for robust task completion. For robust task completion where trajectories are safe and reach the goal, SLS² performs the best compared to all baselines including the constrained iLQR (i.e., SLS² without robust constraints) with or without the ID constraint (14f), nominal iLQR (without any constraints), the HJ-filtered, and LPB baselines.

Task	Method	Safety (%) \uparrow	Success (%) \uparrow	Robust Success (%) \uparrow	Solve time/step (s) \downarrow
Reacher	Ours (SLS ²)	100.00	100.00	100.00	0.360 \pm 0.002
	Constrained iLQR w/ latent ID	100.00	97.22	97.22	0.360 \pm 0.002
	Constrained iLQR w/o latent ID	100.00	97.22	97.22	0.363 \pm 0.003
	HJ-filtered	100.00	100.0	100.0	–
	LPB	0.00	100.00	0.00	–
	Nominal iLQR	0.00	100.0	0.00	–
OGBench (Cube)	Ours (SLS ²)	94.29	97.14	94.29	0.359 \pm 0.016
	Constrained iLQR w/ latent ID	30.56	86.11	27.78	0.345 \pm 0.016
	Constrained iLQR w/o latent ID	25.00	91.67	22.22	0.349 \pm 0.013
	HJ-filtered	0.00	88.57	0.00	–
	LPB	0.00	85.71	0.00	–
	Nominal iLQR	0.00	82.86	0.00	–
Rope	Ours (SLS ²)	100.00	88.57	88.57	0.650 \pm 0.042
	Constrained iLQR w/ latent ID	69.44	91.67	61.11	0.597 \pm 0.026
	Constrained iLQR w/o latent ID	77.78	91.67	69.44	0.593 \pm 0.019
	HJ-filtered	37.14	100.00	37.14	–
	LPB	57.15	100.00	57.15	–
	Nominal iLQR	54.29	100.00	54.29	–

Reacher For Reacher, we introduce joint angle constraints between -2.88 and -2.45 radians on the second joint, which must be robustly satisfied to complete the task (Figure 2). Both our robust method, SLS², and the HJ-filter baseline, using our gradient-based nominal iLQR planner as the nominal policy, robustly complete the task with a 100% success rate. In contrast, the constrained iLQR variants remain safe but fail to complete the task in one instance. Notably, constrained iLQR and SLS² have comparable solve times, demonstrating that the robust constraint tightenings in (14e) introduce limited computational overhead.

Cube For the OGBench Cube task, we enforce that the gripper height must be below 9 cm after the arm has grasped the cube. Due to the model training, the arm has a natural tendency to lift the cube, thus causing the nominal-iLQR baseline to fail (Figure 7(b)). Across the remaining baselines, SLS² has the highest robust task completion rate for OGBench Cube by keeping the cube below the threshold (Figure 7(a)), indicating its ability to robustly complete tasks. Furthermore, by shifting the planning behavior from lifting the cube unnecessarily high, this experiment shows the utility of SLS² as a tool for post-hoc behavior alteration for world-model planning.

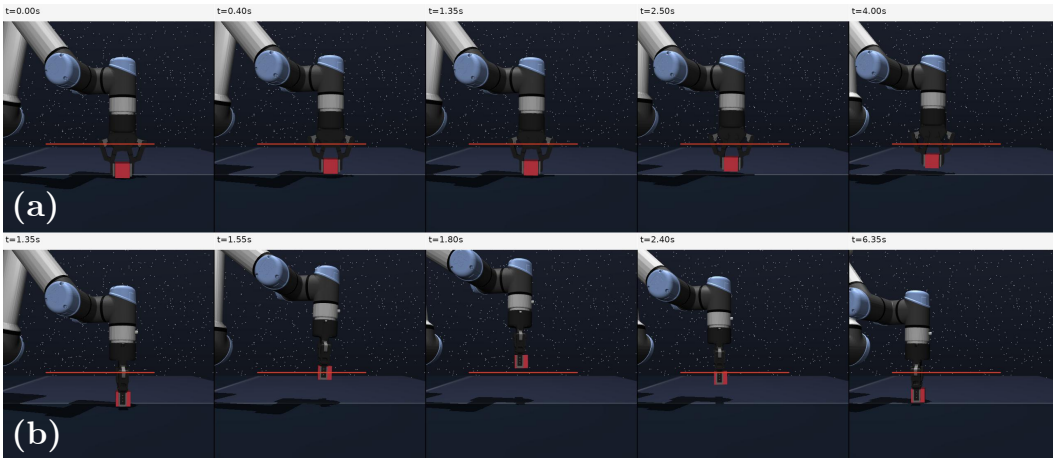


Figure 7: **Robust planning; Cube.** Time-lapse of the rollout using (a) safe planner (SLS²) versus the (b) unsafe planner (nominal-iLQR). SLS² maintains the cube below the height threshold (red line) while reaching the goal, unlike the nominal (unsafe) planner.

6 Discussion, Limitations, and Conclusion

We propose SLS², a method for probabilistically-safe latent space planning using gradient-based planners for robust visuomotor control. Methodologically, we provided an approach rigorously grounded in the uncertainty quantification and propagation tools of conformal prediction and system level synthesis. Empirically, we improve task and robust task success relative to baselines for both nominal planning and robust planning, while ensuring all components of our method, including robust planning, exploit GPU acceleration for scalability.

Limitations: Performance on environments requiring complex, sequential maneuvers, such as Push-T, remains a bottleneck because of our planner’s reliance on Euclidean distance to the latent goal, which assumes an isotropic latent representation. Future work must integrate learned temporal distance metrics designed to preserve geodesic transition distances along the manifold of valid states.

References

- [1] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. Ieee, 2019.
- [2] 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*, 50(7):947–957, 2005.
- [3] I. R. Manchester and J.-J. E. Slotine. Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control*, 62(6):3046–3053, 2017.
- [4] G. Chou, N. Ozay, and D. Berenson. Safe output feedback motion planning from images via learned perception modules and contraction theory. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 349–367. Springer, 2022.
- [5] A. P. Leeman, S. Zhan, M. N. Zeilinger, and G. Chou. Vision-sls: Safe perception-based control from learned visual representations via system level synthesis. *arXiv preprint arXiv:2604.24894*, 2026.
- [6] G. Chou and R. Tedrake. Synthesizing stable reduced-order visuomotor policies for nonlinear systems via sums-of-squares optimization. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 624–631. IEEE, 2023.
- [7] S. Dean, A. Taylor, R. Cosner, B. Recht, and A. Ames. Guaranteeing safety of learned perception modules via measurement-robust control barrier functions. In *Conference on Robot Learning*, pages 654–670. PMLR, 2021.
- [8] D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.
- [9] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=S1l0TC4tDS>.
- [10] G. Zhou, H. Pan, Y. Lecun, and L. Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 79115–79135. PMLR, 13–19 Jul 2025.
- [11] N. Hansen, H. Su, and X. Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pages 47376–47405, 2024.
- [12] F. Castaneda, H. Nishimura, R. T. McAllister, K. Sreenath, and A. 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] K. Nakamura, L. Peters, and A. Bajcsy. Generalizing safety beyond collision-avoidance via latent-space reachability analysis. *arXiv preprint arXiv:2502.00935*, 2025.
- [14] J. Anderson, J. C. Doyle, S. H. Low, and N. Matni. System level synthesis. *Annual Reviews in Control*, 47:364–393, 2019.
- [15] 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*, pages 2555–2565. PMLR, 2019.

- [16] D. Hafner, T. P. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=0oabwyZb0u>.
- [17] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.
- [18] D. Hafner, W. Yan, and T. Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.
- [19] N. A. Hansen, H. Su, and X. Wang. Temporal difference learning for model predictive control. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8387–8406. PMLR, 17–23 Jul 2022.
- [20] Y. LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.
- [21] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15619–15629, 2023.
- [22] A. Bardes, Q. Garrido, J. Ponce, X. Chen, M. Rabbat, Y. LeCun, M. Assran, and N. Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=QaCCuDFBk2>. Featured Certification.
- [23] M. Assran, A. Bardes, D. Fan, Q. Garrido, R. Howes, M. Muckley, A. Rizvi, C. Roberts, K. Sinha, A. Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.
- [24] U. Sobal, W. Zhang, K. Cho, R. Balestriero, T. G. Rudner, and Y. LeCun. Learning from reward-free offline data: A case for planning with latent dynamics models. *Advances in Neural Information Processing Systems*, 38:43905–43941, 2026.
- [25] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.
- [26] V. Micheli, E. Alonso, and F. Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=vhFu1Acb0xb>.
- [27] E. Alonso, A. Jelley, V. Micheli, A. Kanervisto, A. Storkey, T. Pearce, and F. Fleuret. Diffusion for world modeling: Visual details matter in atari. *Advances in Neural Information Processing Systems*, 37:58757–58791, 2024.
- [28] V. Micheli, E. Alonso, and F. Fleuret. Efficient world models with context-aware tokenization. In *Forty-first International Conference on Machine Learning*, 2024. URL <https://openreview.net/forum?id=BiWIERWBFX>.
- [29] J. Bruce, M. D. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024.
- [30] L. Maes, Q. L. Lidec, D. Scieur, Y. LeCun, and R. Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

- [31] 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*, 62(8):3861–3876, 2016.
- [32] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.
- [33] K. P. Wabersich and M. N. Zeilinger. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597, 2021.
- [34] 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*, 43(5):137–177, 2023.
- [35] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724. Ieee, 2020.
- [36] L. Lindemann, A. Robey, L. Jiang, S. Das, S. Tu, and N. Matni. Learning robust output control barrier functions from safe expert demonstrations. *IEEE Open Journal of Control Systems*, 3: 158–172, 2024.
- [37] K. Kang, P. Gradu, J. J. Choi, M. Janner, C. Tomlin, and S. Levine. Lyapunov density models: Constraining distribution shift in learning-based control. In *International Conference on Machine Learning*, pages 10708–10733. PMLR, 2022.
- [38] M. Faroni, C. Odesco, A. M. Zanchettin, and P. Rocco. Uncertainty-aware planning with inaccurate models for robotized liquid handling. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 17725–17731. IEEE, 2025.
- [39] M. E. Cao, M. Bloch, and S. Coogan. An optimistic approach to cost-aware predictive control. *Automatica*, 176:112263, 2025.
- [40] G. Chou, N. Ozay, and D. Berenson. Model error propagation via learned contraction metrics for safe feedback motion planning of unknown systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 3576–3583. IEEE, 2021.
- [41] C. Knuth, G. Chou, N. Ozay, and D. Berenson. Planning with learned dynamics: Probabilistic guarantees on safety and reachability via lipschitz constants. *IEEE Robotics and Automation Letters*, 6(3):5129–5136, 2021.
- [42] C. Knuth, G. Chou, J. Reese, and J. Moore. Statistical safety and robustness guarantees for feedback motion planning of unknown underactuated stochastic systems. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12700–12706. IEEE, 2023.
- [43] H. T. Suh, G. Chou, H. Dai, L. Yang, A. Gupta, and R. Tedrake. Fighting uncertainty with gradients: Offline reinforcement learning via diffusion score matching. In *Conference on Robot Learning*, pages 2878–2904. PMLR, 2023.
- [44] L. Dodeja, K. Schmeckpeper, S. Vats, T. Weng, M. Jia, G. Konidaris, and S. Tellex. Accelerating residual reinforcement learning with uncertainty estimation. *IEEE Robotics and Automation Letters*, 11(1):970–977, 2025.
- [45] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

- [46] J. Seo, K. Nakamura, and A. Bajcsy. Uncertainty-aware latent safety filters for avoiding out-of-distribution failures. *arXiv preprint arXiv:2505.00779*, 2025.
- [47] K. Nakamura, A. L. Bishop, S. Man, A. M. Johnson, Z. Manchester, and A. Bajcsy. How to train your latent control barrier function: Smooth safety filtering under hard-to-model constraints. *arXiv preprint arXiv:2511.18606*, 2025.
- [48] M. Anand and S. Kolathaya. Safety certification in the latent space using control barrier functions and world models. *arXiv preprint arXiv:2507.13871*, 2025.
- [49] Z. Sun and S. Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution. *Advances in Neural Information Processing Systems*, 38:174280–174305, 2026.
- [50] Z. Mao, C. Sobolewski, and I. Ruchkin. How safe am i given what i see? calibrated prediction of safety chances for image-controlled autonomy. In *6th Annual Learning for Dynamics & Control Conference*, pages 1370–1387. PMLR, 2024.
- [51] R. Balestriero and Y. LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.
- [52] Y. Wang, O. Bounou, G. Zhou, R. Balestriero, T. G. Rudner, Y. LeCun, and M. Ren. Temporal straightening for latent planning. *arXiv preprint arXiv:2603.12231*, 2026.
- [53] R. J. Tibshirani, R. Foygel Barber, E. Candes, and A. Ramdas. Conformal prediction under covariate shift. *Advances in neural information processing systems*, 32, 2019.
- [54] J. Lei and L. Wasserman. Distribution-free prediction bands for non-parametric regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 76(1):71–96, 2014.
- [55] J. Fang and G. Chou. Safe large-scale robust nonlinear mpc in milliseconds via reachability-constrained system level synthesis on the gpu. *arXiv preprint arXiv:2604.07644*, 2026.
- [56] A. Srinivasan, A. Leeman, and G. Chou. Safety beyond the training data: Robust out-of-distribution mpc via conformalized system level synthesis. In *8th Annual Learning for Dynamics and Control Conference*, 2026.
- [57] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.
- [58] S. Park, K. Frans, B. Eysenbach, and S. Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, volume 2025, pages 94937–94982, 2025.
- [59] W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *First International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 222–229. SciTePress, 2004.
- [60] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008.
- [61] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.
- [62] R. Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL <https://drake.mit.edu>.

Appendices

In the following, we provide an overview of our appendices. In App. A, we provide training details for the latent world models, including the offline data collection procedure, shared HDF5 dataset format, optimization settings, and task-specific latent dynamics network architectures. In App. B, we review the system level synthesis (SLS) formulation used to compute robust feedback policies and reachable tubes for the learned latent dynamics, and describe how nominal constraints are tightened to account for calibrated model error. In App. C, we prove the probabilistic containment guarantee for the conformal SLS tubes. In App. D, we provide experimental implementation details, including the benchmark setup, nominal planning objectives, world-model baselines, implementation of Problem (14), calibrated model-error bounds, latent constraint classifiers, in-distribution latent support constraints, safety baselines, and the real-world rope manipulation setup. In App. E, we provide additional experimental results, including nominal trajectory time-lapses, safe versus unsafe trajectory comparisons, latent reachable-tube visualizations, and expanded quantitative tables for nominal planning and robust task-completion baselines.

A Training Details

Offline Dataset We train the latent world model from offline trajectories collected independently for each task. Each trajectory consists of a sequence of image observations and actions,

$$\tau = \{(o_t, a_t)\}_{t=0}^{T-1}, \quad (15)$$

where o_t is a rendered RGB observation and a_t is the action applied between consecutive observations. The datasets are collected before world model training and no online interaction is used during model learning. The world model is trained only from the observation-action sequences, while additional simulator states, rewards, and task-specific quantities are stored for analysis, visualization, and evaluation.

Across environments, we use task-specific data-generation policies to produce trajectories with sufficient interaction coverage. For expert-control tasks, such as Reacher, trajectories are collected by rolling out a trained expert policy. For manipulation tasks that require structured behavior, such as OGBench Cube, trajectories are generated using an oracle planner that constructs task-space keyframes and executes the corresponding control sequence. For the two-manipulator rope environment, we use randomized task-space trajectory generators, including spline- and waypoint-based policies, to cover a diverse range of rope configurations and manipulator motions. This produces datasets that are not tied to a single evaluation goal, but instead expose the world model to local dynamics over a broad set of feasible interactions.

All datasets are stored in a common HDF5 format. For each episode, we store the rendered pixel observations, applied actions, episode length, episode offset, episode seed, and per-step episode/step indices. Actions are padded with a final invalid entry so that the action array has the same temporal length as the observation array; this makes it straightforward to sample fixed-length training windows while ignoring the last action of each episode. Episodes shorter than a minimum length are discarded. The collection loop either runs for a fixed number of episodes or until a specified target number of transitions is reached. This common storage format allows the same world-model training pipeline to be reused across all environments.

Training the World Model Across experiments, we train the model using AdamW with learning rate 5×10^{-5} , weight decay 10^{-3} , mixed-precision training, and gradient clipping with maximum norm 1.0. Model checkpoints are saved after each epoch, and the same general training pipeline is used across all tasks, with only task-dependent choices such as latent dimension, Markov order, rollout horizon, and loss weights adjusted for each environment.

MLP Latent Dynamics Models All four tasks use the same vision backbone: a tiny ViT encoder on 224×224 RGB images with patch size 14. The encoder output is projected to a task-specific

latent $z_t \in \mathbb{R}^{n_z}$. The dynamics model then forms a Markov latent state

$$s_t = [z_t, \Delta z_t, \dots, \Delta^K z_t] \in \mathbb{R}^{(K+1)n_z},$$

where K is the desired Markov order. A feed-forward MLP predicts the next Markov state from the current Markov state and action,

$$\hat{s}_{t+1} = f_\phi(s_t, u_t).$$

Here the listed action dimension is the number of control channels concatenated into the predictor input; since all runs use frameskip 1, this is also the effective action dimension. Reacher and Push-T use two-dimensional actions, Rope uses a three-dimensional endpoint action, and OGBench Cube uses the five-dimensional environment action vector. The predictor is trained by autoregressively rolling this one-step MLP for num_preds=5 future steps.

Table 4: Model architecture and state dimensions across tasks.

Task	n_z	K	$\dim(s_t)$	$\dim(v_t)$	MLP Size	Straightening
OGBench-Cube	12	1	24	5	512×2	yes
Reacher	5	1	10	2	512×2	no
Push-T	48	2	144	2	512×3	no
Rope	12	1	24	3	512×2	yes

The MLP column reports hidden width times number of hidden layers, with GELU nonlinearities and a linear output layer. The MLP size column only considers the latent dynamics predictor, excluding the ViT encoder and latent projector. The SIGReg term uses the same sketch size in all runs: 17 quadrature knots and 1024 random projections, weighted by $\lambda_{\text{SIG}} = 0.005$. Temporal straightening is active only for the OGBench Cube and Rope runs.

B System Level Synthesis

System Level Synthesis. We use state-feedback SLS to compute robust feedback policies and reachable tubes for the learned latent dynamics. Let $s_k \in \mathbb{R}^{n_s}$ be the Markov latent state and $u_k \in \mathbb{R}^{n_u}$ the control. Around a nominal trajectory $(\zeta_{0:T}, v_{0:T-1})$ satisfying $\zeta_{k+1} = f_\phi(\zeta_k, v_k)$, we linearize

$$\Delta s_{k+1} = A_k \Delta s_k + B_k \Delta u_k + E_k w_k, \quad w_k \in \mathcal{B}^{n_s},$$

where $A_k = \nabla_s f_\phi(\zeta_k, v_k)$, $B_k = \nabla_u f_\phi(\zeta_k, v_k)$, and $E_k \mathcal{B}^{n_s}$ is a calibrated latent prediction-error set.

SLS parameterizes the closed-loop system by response matrices $\Phi_{k,j}^s$ and $\Phi_{k,j}^u$ such that

$$s_k - \zeta_k = \sum_{j=0}^{k-1} \Phi_{k,j}^s w_j, \quad u_k - v_k = \sum_{j=0}^{k-1} \Phi_{k,j}^u w_j.$$

These responses are valid if they satisfy the finite-horizon SLS constraints

$$\Phi_{k+1,j}^s = A_k \Phi_{k,j}^s + B_k \Phi_{k,j}^u, \quad j < k, \quad \Phi_{k+1,k}^s = E_k.$$

Then defining \oplus and \bigoplus as the Minkowski sum of two sets (i.e., $\mathcal{A} \oplus \mathcal{B} = \{a + b \mid a \in \mathcal{A}, b \in \mathcal{B}\}$ for sets, \mathcal{A} and \mathcal{B}), the corresponding reachable tubes are

$$\mathcal{R}_k^s = \zeta_k \oplus \bigoplus_{j=0}^{k-1} \Phi_{k,j}^s \mathcal{B}^{n_s}, \quad \mathcal{R}_k^u = v_k \oplus \bigoplus_{j=0}^{k-1} \Phi_{k,j}^u \mathcal{B}^{n_s}.$$

For constraints $\mathcal{F} := \{(s, u) \mid g_i(s, u) + b_i \leq 0, i = 1, \dots, n_c\}$, we impose robust satisfaction by tightening the nominal constraints. Linearizing g_i around (ζ_k, v_k) gives the sufficient condition

$$g_i(\zeta_k, v_k) + b_i + \sum_{j=0}^{k-1} \|\nabla_s g_i(\zeta_k, v_k) \Phi_{k,j}^s + \nabla_u g_i(\zeta_k, v_k) \Phi_{k,j}^u\|_2 \leq 0,$$

for all i and k . The SLS MPC problem jointly optimizes $(\zeta, v, \Phi^s, \Phi^u)$ subject to nominal dynamics, the SLS response constraints, and these tightened constraints.

C Proofs

Theorem C.1 (Probabilistic containment of conformal SLS tubes). *Consider the latent closed-loop dynamics $s_{k+1} = f_\phi(s_k, u_k) + e_k$, for $k = 0, \dots, T - 1$. Let q be the split conformal quantile of the calibration scores $r_i = e_i^\top \Sigma_\psi(s_i, u_i)^{-1} e_i$ and $e_i := s_i^+ - f_\phi(s_i, u_i)$, computed at per-step miscoverage $\bar{\delta} = \delta/T$. For each closed-loop transition, define the test score $r_k^{\text{test}} := e_k^\top \Sigma_\psi(s_k, u_k)^{-1} e_k$. Assume that, for each k , r_k^{test} is exchangeable with the calibration scores. Let $\Sigma_\psi(s, u) = L_\psi(s, u)L_\psi(s, u)^\top$ and define the local conformal error set $\mathcal{E}(s, u) := \sqrt{q} L_\psi(s, u)\mathcal{B}^{n_s}$ as defined in (10). Suppose the SLS controller is synthesized with disturbance matrices E_k such that the realized conformal error sets are contained in the SLS disturbance sets, $\mathcal{E}(s_k, u_k) \subseteq E_k\mathcal{B}^{n_s}$, for $k = 0, \dots, T - 1$. If the SLS controller is robustly correct for all disturbances $w_k \in \mathcal{B}^{n_s}$ in $\Delta s_{k+1} = A_k\Delta s_k + B_k\Delta u_k + E_k w_k$, where $\Delta s_k := s_k - \zeta_k$ and $\Delta u_k := u_k - v_k$, then $\mathbb{P}[(s_k \in \mathcal{R}_k^s, \forall k = 0, \dots, T) \wedge (u_k \in \mathcal{R}_k^u, \forall k = 0, \dots, T - 1)] \geq 1 - \delta$.*

Proof. By split CP, exchangeability gives

$$\mathbb{P}[r_k^{\text{test}} \leq q] \geq 1 - \bar{\delta}$$

for each k . Since

$$r_k^{\text{test}} \leq q \iff e_k \in \sqrt{q} L_\psi(s_k, u_k)\mathcal{B}^{n_s} = \mathcal{E}(s_k, u_k),$$

a union bound over the T transitions yields

$$\mathbb{P}[e_k \in \mathcal{E}(s_k, u_k), \forall k = 0, \dots, T - 1] \geq 1 - T\bar{\delta} = 1 - \delta.$$

On this event, the assumed containment $\mathcal{E}(s_k, u_k) \subseteq E_k\mathcal{B}^{n_s}$ implies that each realized prediction error can be written as

$$e_k = E_k w_k, \quad w_k \in \mathcal{B}^{n_s}.$$

Thus the realized closed-loop trajectory is generated by a disturbance sequence inside the uncertainty set used by SLS. Robust correctness of SLS then implies $s_k \in \mathcal{R}_k^s$ for $k = 0, \dots, T$ and $u_k \in \mathcal{R}_k^u$ for $k = 0, \dots, T - 1$. Therefore the tube-containment event holds with probability at least $1 - \delta$. \square

D Experimental Implementation

In the following sections, we discuss the experimental implementations for two of our central contributions: our **nominal planner** and our **robust planner**. Section D.1 discusses the experimental setup for our nominal experiments. Then, Section D.2 discusses the implementation details of our **novel** gradient-based iLQR planner, which leverages the Markovian latent space to enable high task success rates (cf. Table 1 for the ablation study). Next, Section D.3 details the baseline implementations used to evaluate our nominal planner against. For the safety experiments, Section D.4 provides details on how our robust planner, SLS², is implemented, and Section D.5 provides details on the safety baselines we compare our approach to. Lastly, Section D.6 discusses our real-world hardware experiment.

D.1 Experiment Setup

We evaluate on four continuous-control benchmarks spanning low-dimensional reaching, rigid-object manipulation, planar pushing, and deformable-object manipulation. For each task, we train from expert demonstrations and evaluate planning using task-specific goal-reaching criteria.

Reacher. Reacher is the hard variant of the DeepMind Control Suite Reacher task, where a two-joint planar arm must reach a target position. We render 224×224 RGB observations with the target hidden from the image observation. The dataset contains 24,650 expert episodes collected with a Soft Actor-Critic policy. Episodes are collected at 50 Hz for up to 100 control steps and terminate early when the end effector reaches within 0.03 of the target.

Cube. Cube is the OGBench `cube-single-v0` task, where a robot arm must grasp and move a cube to a target 3D position and yaw. We use front-camera pixel observations and continuous 5D actions. The dataset contains 11,227 oracle-planner episodes recorded at 20 Hz for up to 100 steps. Episodes terminate when the cube is within 0.04 m of the target position and within 0.2 rad of the target yaw.

Push-T. Push-T is a 2D planar manipulation task where a circular pusher moves a T-shaped block to a target pose. We use `gym-pusht/PushT-v0` and store RGB observations, pusher positions, and continuous 2D actions. The dataset contains 25,000 expert episodes: 20,000 nominal diffusion-policy rollouts and 5,000 edge-case rollouts. Episodes contain up to 167 recorded steps.

Rope. Rope is a bimanual MuJoCo manipulation environment where two robot arms control the endpoints of a deformable rope. The task exposes the model to nonlinear rope dynamics by moving the rope attachments along randomized task-space trajectories. We collect 15,000 episodes using a cubic-spline policy with six randomized waypoints. Episodes are recorded at 20 Hz, average approximately 80 steps, and are rendered as 224×224 RGB observations.

Planning success criteria. For Reacher, success is measured by the wrapped joint-position distance between the current and goal configurations. The primary threshold is 0.1 in qpos distance, and we also report a stricter full-observation threshold of 0.05 as a diagnostic. For Cube, success requires the cube position to be within 0.04 m of the goal position. For Push-T, success requires the T-block to be within 20 pixels of the goal position with yaw error at most 0.35 rad. For Rope, success requires the task-target distance to reach the dataset goal tolerance of 0.05.

D.2 Nominal Planning Setup

The nominal MPC planners optimize in the learned Markov latent state. For Reacher, OGBench-Cube, and Rope, the nominal planner is receding-horizon iLQR with horizon $T = 15$, but the quadratic cost matrices are task-specific (where weights scale the identity matrix $I_n \in \mathbb{R}^{n \times n}$):

Table 5: **Reacher, OGBench-Cube, Rope:** Nominal iLQR planning horizons and cost weights.

Task	T	Q_T	Q	R
Reacher	15	$5.0I_{10}$	$0.05I_{10}$	$0.1I_2$
OGBench-Cube	15	$15.0I_{24}$	$0.05I_{24}$	$0.5I_5$
Rope	15	$15.0I_{24}$	$0.005I_{24}$	$0.01I_3$

For these iLQR-only planners, the rollout is generated by the learned one-step dynamics $\zeta_{t+1} = f_\phi(\zeta_t, v_t)$, and the open-loop control sequence \mathbf{v} is optimized with the finite-horizon cost

$$J_{\text{iLQR}}(\zeta, \mathbf{v}) = \sum_{t=0}^{T-1} (\|\zeta_t - \zeta_g\|_Q^2 + \|v_t\|_R^2) + \|\zeta_T - \zeta_g\|_{Q_T}^2, \quad (16)$$

where ζ_g is the goal latent state. In Reacher, the same expression is used but the cost is computed only on the zeroth-order latent coordinates (i.e., z), so the cost does not directly penalize the finite-difference components of the Markov state. At each environment step, iLQR linearizes the learned dynamics along the current rollout, performs a Riccati-style backward pass to compute local feedback/feedforward updates, line-searches the updated control sequence, executes the first action, and replans from the newly observed image.

For Push-T, the nominal planner uses MPPI to produce a long-horizon warm-start and then uses a short-horizon iLQR tracker to refine the first segment. The MPPI horizon is $H^{\text{mppi}} = 45$, while the iLQR tracking horizon is $T = 15$. The MPPI cost matrices are $Q_T^{\text{mppi}} = 10.0I_{144}$, $Q^{\text{mppi}} = 0.05I_{144}$, and $R^{\text{mppi}} = 0.01I_2$; the iLQR tracking matrices are $Q_T = 10.0I_{144}$, $Q = 1.0I_{144}$, and $R = 0.01I_2$. MPPI samples noisy control sequences, rolls them out through the same latent

dynamics, and scores each sampled trajectory with

$$J_{\text{MPPI}}(\zeta, \mathbf{v}) = \sum_{t=0}^{H^{\text{mppi}}-1} \left(\|\zeta_t - \zeta_g\|_{Q^{\text{mppi}}}^2 + \|v_t\|_{R^{\text{mppi}}}^2 \right) + \|\zeta_{H^{\text{mppi}}} - \zeta_g\|_{Q_T^{\text{mppi}}}^2. \quad (17)$$

The sampled costs are converted to rewards by negation and weighted by the MPPI reward temperature when updating the nominal control sequence. The iLQR tracker then optimizes over the shorter horizon $T = 15$ against the MPPI reference trajectory $\bar{\zeta}$ rather than a fixed goal:

$$J_{\text{track}}(\zeta, \mathbf{v}) = \sum_{t=0}^{T-1} \left(\|\zeta_t - \bar{\zeta}_t\|_Q^2 + \|v_t\|_R^2 \right) + \|\zeta_T - \bar{\zeta}_T\|_{Q_T}^2. \quad (18)$$

Thus the MPPI stage adds a sampling-based global search over long control sequences, while the iLQR stage locally smooths and tracks the selected latent trajectory before receding-horizon execution.

D.3 World Model Baseline Implementation

Training and evaluation protocol. This section describes the implementation details for the world-model baselines and our planning pipeline. Across all tasks, we follow the training and evaluation workflow of the `stable-worldmodel` codebase whenever possible. We compare against PLDM, DINO-WM, and Le-WM. Unless otherwise specified, each baseline is trained for 10 epochs following the Le-WM training protocol [30], where this training length was reported to provide strong planning performance. When task-compatible pretrained or reusable checkpoints are available, we initialize from those checkpoints and fine-tune on our dataset. In particular, due to the size of the Push-T dataset and the cost of training from scratch, the Le-WM model for Push-T is fine-tuned from a pretrained Push-T object checkpoint. For the customized Rope task, no pretrained checkpoints are available, so all world models are trained from scratch using the same offline dataset.

Shared preprocessing and architecture. All baselines are trained from image observations resized to 224×224 with patch size 14. We use a history length of 3 and a one-step prediction horizon. During planning, the learned one-step latent dynamics are rolled out autoregressively to evaluate multi-step action sequences. The dataset is split into 90% training and 10% validation, and we apply gradient clipping with threshold 1.0. Unless otherwise stated, models are trained with batch size 128. We use frame skip 5 for Reacher, Cube, and Push-T, and frame skip 1 for Rope.

PLDM. For PLDM, we train with AdamW using learning rate 5×10^{-5} and weight decay 10^{-3} . The encoder is a ViT-Tiny model with embedding dimension 192. The predictor has depth 6, 16 attention heads, MLP dimension 2048, head dimension 64, and dropout 0.1. Proprioceptive inputs are disabled. The active loss weights are 18 for the standard-deviation loss, 0.7 for the temporal standard-deviation loss, 12 for the covariance loss, and 0.2 for the temporal alignment loss.

DINO-WM. For DINO-WM, we use a frozen DINOv2-Small visual backbone and train the latent predictor with AdamW using learning rate 5×10^{-4} and no weight decay. The predictor uses the same architecture as PLDM: depth 6, 16 attention heads, MLP dimension 2048, head dimension 64, and dropout 0.1. Actions are encoded with dimension 10. Reacher, Cube, and Rope are trained without proprioceptive encoding, while Push-T uses proprioceptive encoding.

Le-WM. For Le-WM, we train with AdamW using learning rate 5×10^{-5} and weight decay 10^{-3} . The model uses a ViT-Tiny encoder with image size 224, patch size 14, history size 3, prediction horizon 1, and embedding dimension 192. The latent predictor uses the same configuration as the other baselines: depth 6, 16 attention heads, MLP dimension 2048, head dimension 64, and dropout 0.1. For Rope, Le-WM is trained from scratch with batch size 128, frame skip 1, and SigReg regularization with weight 0.09, 17 knots, and 1024 random projections.

For Push-T, we fine-tune the pretrained Le-WM Push-T object checkpoint on the combined Push-T dataset. The Push-T model uses frame skip 5, action dimension 2, and effective action dimension 10. We train for 10 epochs with batch size 110, bfloat16 mixed precision, gradient clipping at 1.0, and the Le-WM latent prediction objective. During fine-tuning, we use SigReg regularization with weight 0.005, 17 knots, and 1024 random projections, together with temporal straightening loss with weight 0.01.

Planning evaluation. For nominal world-model planning baselines, PLDM, DINO-WM, and Le-WM use the standard CEM planner provided by the `stable-worldmodel` implementation. CEM optimizes action sequences in the learned latent space and executes the selected actions in a receding-horizon manner. For our method, we use the same learned latent dynamics as the planning model but replace CEM with our gradient-based latent planner. In the robust setting, this planner is augmented with the SLS-based tube propagation and conformalized latent constraints described in the main text. This setup ensures that baseline performance reflects the default planning pipeline associated with each learned world model, while our method differs only in the planning and robustness layer built on top of the learned latent dynamics.

D.4 Implementing SLS² (Problem (14))

Below we discuss our implementation of Problem (14). For self-containment, we restate the problem:

$$\min_{\zeta, \mathbf{v}, \Phi^s, \Phi^u} J(\zeta, \mathbf{v}) + J_f(\zeta_T) + H(\Phi^s, \Phi^u) \quad (14a)$$

$$\text{s.t. } \zeta_{k+1} = f_\phi(\zeta_k, v_k), \quad \zeta_0 = \bar{s}_0, \quad k = 0, \dots, T-1, \quad (14b)$$

$$\Phi_{k+1,j}^s = A_k \Phi_{k,j}^s + B_k \Phi_{k,j}^u, \quad 0 \leq j < k, \quad k = 0, \dots, T-1, \quad (14c)$$

$$\Phi_{j+1,j}^s = E_j, \quad j = 0, \dots, T-1, \quad (14d)$$

$$g_i(\zeta_k, v_k) + b_i + \tau_{i,k} \leq 0, \quad i = 1, \dots, n_c, \quad k = 0, \dots, T, \quad (14e)$$

$$\mathcal{R}_k^s \subseteq \mathcal{I}, \quad k = 0, \dots, T. \quad (14f)$$

In Section D.4.1 we discuss the cost and reference trajectory generation used for the objective in (14a). Next, in Section D.4.2 we discuss how the model error is calibrated. The calibrated error is used to initialize the system level constraints (SLC) in (14c)-(14d). Details on the obstacles and how we train a latent classifier and conformalize it are available in Section D.4.3. In the optimization problem, the classifier is linearized and tightened in (14e). Lastly, we discuss the in-domain ellipsoid constraint, (14f), and how it is formed in Section D.4.4.

D.4.1 Defining the Objective

To implement Problem (14), we follow [55, 56] and use a real-time-iteration (RTI) scheme which solves a single SQP (instead of solving several successive SQPs until convergence) to enable faster solve times. To avoid ill-conditioned problems and improve numerical stability, we use the MPPI algorithm [61] to warm-start the MPC solve at each time step. Below, we describe the MPPI warm-start objective, the nominal SLS objective, the associated cost weights, and the modifications used for the Constraint-iLQR baselines.

Let $\zeta_t \in \mathbb{R}^{n_s}$ denote the latent state and ζ_g the goal latent state. MPPI first generates a reference trajectory $\bar{\zeta} := \{\bar{\zeta}_k\}_{k=0}^T$ and $\bar{\mathbf{v}} := \{\bar{v}_k\}_{k=0}^{T-1}$ by minimizing the sampling objective

$$J_{\text{MPPI}}(\zeta, \mathbf{v}) = \sum_{t=0}^{H^{\text{mppi}}-1} \left(\|\zeta_t - \zeta_g\|_{Q^{\text{mppi}}}^2 + \|v_t\|_{R^{\text{mppi}}}^2 + J_{\text{soft}}(\zeta_t) \right) + \|\zeta_{H^{\text{mppi}}} - \zeta_g\|_{Q_T^{\text{mppi}}}^2, \quad (20)$$

where J_{soft} denotes task-dependent soft penalties used during MPPI sampling. The MPPI horizon H^{mppi} can be greater than T (the tracking horizon used by SLS) to encourage smaller actions at each step. Furthermore, we define $\|x\|_M^2 := x^\top M x$ for $x \in \mathbb{R}^n$ and $M \in \mathbb{R}^{n \times n}$. The nominal RTI-SQP/SLS objective then tracks the MPPI reference trajectory $\bar{\zeta}$ and $\bar{\mathbf{v}}$ over the MPC horizon T :

$$J_{\text{SLS}}(\zeta, \mathbf{v}) = \sum_{t=0}^{T-1} \left(\|\zeta_t - \bar{\zeta}_t\|_Q^2 + \|v_t - \bar{v}_t\|_R^2 \right) + \|\zeta_T - \bar{\zeta}_T\|_{Q_T}^2. \quad (21)$$

Our cost matrices penalizing state tracking $Q_T, Q, Q_T^{\text{mppi}}, Q^{\text{mppi}} \succeq 0$ and control penalties $R, R^{\text{mppi}} \succ 0$ are summarized in Table 6. While interpretability is minimal in the latent space, we leverage the Markovian structure to apply different penalties to position states (corresponding to the zeroth-order state) and velocity states (corresponding to the first-order state). Compactly, we use $I_n \in \mathbb{R}^{n \times n}$ as the n -dimensional identity matrix, and for $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{m \times n}$ we denote $\text{blkdiag}(A, B)$ as the block diagonal matrix.

In (20), we also add a soft penalty, J_{soft} , defined as

$$J_{\text{soft}}(\zeta) = \lambda_{\text{ID}} \underbrace{[\max\{\rho_{\text{ID}}(\zeta) - 1, 0\}]^2}_{\text{Latent Ellipsoid Penalty}} + \lambda_{\text{obs}} \underbrace{[\text{softplus}(\eta - c_\psi(\Pi_z \zeta))]^2}_{\text{Obstacle Penalty}} \quad (22)$$

where $\rho_{\text{ID}}(\zeta)$ is the conformal latent ellipsoid defined in (11) and $c_\psi(\Pi_z \zeta)$ is the obstacle classifier score evaluated on the zeroth-order latent component. λ_{obs} and λ_{ID} are hyperparameters controlling

Table 6: MPPI warm-start and nominal SLS tracking weights used in our experiments.

Task	Objective	Horizon	State Weights	Control Weight
Rope	MPPI	15	$Q^{\text{mppi}} = \text{blkdiag}(0.005I_{12}, 1.0I_{12})$ $Q_T^{\text{mppi}} = \text{blkdiag}(10I_{12}, 1.0I_{12})$	$R^{\text{mppi}} = 1.0I_3$
	SLS	10	$Q = 0.1I_{24}, Q_T = 5.0I_{24}$	$R = 0.2I_3$
Reacher	MPPI	60	$Q^{\text{mppi}} = \text{blkdiag}(0.005I_5, 1.0I_5)$ $Q_T^{\text{mppi}} = \text{blkdiag}(100I_5, 1.0I_5)$	$R^{\text{mppi}} = 0.01I_2$
	SLS	5	$Q = \text{blkdiag}(0.005I_5, 1.0I_5)$ $Q_T = \text{blkdiag}(5.0I_5, 0.9I_5)$	$R = 0.1I_2$
OGBench-Cube	MPPI	10	$Q^{\text{mppi}} = \text{blkdiag}(0.05I_{12}, 0.10I_{12})$ $Q_T^{\text{mppi}} = \text{blkdiag}(15I_{12}, 1.0I_{12})$	$R^{\text{mppi}} = 1.0I_5$
	SLS	10	$Q = \text{blkdiag}(0.05I_{12}, 0.05I_{12})$ $Q_T = \text{blkdiag}(1.0I_{12}, 1.0I_{12})$	$R = 0.5I_5$

Table 7: MPPI hyperparameters and soft-penalty weights used for warm-start trajectory generation.

Task	Samples	Updates	Reward Weight	Noise Scale	β	λ_{ID}	λ_{obs}
Rope	2048	6	25	0.20	0.65	20	0
Reacher	512	5	20	0.15	0.70	0	1000
OGBench-Cube	512	5	30	0.20	0.60	10	100

the strength of the respective penalties. The soft penalty in (22) encourages the reference trajectory to satisfy the constraints. Notably, these references are not guaranteed to robustly satisfy the constraints, and thus SLS is still required to enforce robust constraint satisfaction. The remaining MPPI hyperparameters alongside the values of λ_{obs} and λ_{ID} used for each task are summarized in Table 7.

D.4.2 Calibrated Error Bounds

To calibrate the error bounds we split our dataset \mathcal{D}_{cal} in half (i.e., $\mathcal{D}_{\text{cal}}^{(1)}$ and $\mathcal{D}_{\text{cal}}^{(2)}$) such that $|\mathcal{D}_{\text{cal}}^{(1)}| = |\mathcal{D}_{\text{cal}}^{(2)}|$. For our experiments, \mathcal{D}_{cal} is constructed using the same expert rollouts used to train the models, as the transitions encountered in \mathcal{D}_{cal} are distributionally similar to the transitions encountered during planning time, preserving the exchangeability assumption. For Reacher, Rope, and OGBench-Cube, $|\mathcal{D}_{\text{cal}}| = 200,000$, $|\mathcal{D}_{\text{cal}}| = 80,000$, and $|\mathcal{D}_{\text{cal}}| = 98,800$ respectively. The first partition, $\mathcal{D}_{\text{cal}}^{(1)}$, is used to learn the error covariance, $\Sigma_{\psi}(s, u)$, structure by minimizing the MGNLL loss (9), while the second half is used to perform the conformal calibration described in Section 4.2. For the Reacher task, we trained $\Sigma_{\psi}(s, u)$ as a neural network with 2 hidden layers (128 neurons per hidden layer) at a learning rate of 0.0003 using a cosine annealing learning rate scheduler to stabilize training. The model directly outputs the Cholesky factor to avoid computing it in real time. For the Rope and OGBench-Cube tasks, we defined $\Sigma_{\psi}(s, u) := \Sigma_{\text{cov}}$ as a fixed parameter. In this case, the loss is minimized by computing $\Sigma_{\text{cov}} = \frac{1}{|\mathcal{D}_{\text{cal}}^{(1)}|} \sum_{(s_i, u_i, s_i^+) \in \mathcal{D}_{\text{cal}}^{(1)}} (s_i^+ - f_{\phi}(s_i, u_i))(s_i^+ - f_{\phi}(s_i, u_i))^{\top}$ as the sample error covariance. For the conformal calibration we set $\alpha = 0.1, 0.4$, and 0.1 for Reacher, Rope, and OGBench-Cube respectively.

D.4.3 Training and Conformalizing the Constraint Classifier

For each constrained task, we train a latent-space classifier from a balanced dataset of rendered constraint-violating and non-violating observations. Let

$$\mathcal{D}_{\text{obs}} = \{(o_i, y_i)\}_{i=1}^N, \quad y_i \in \{0, 1\},$$

where, following the notation in the main text, $y_i = 0$ denotes a constraint-violating observation and $y_i = 1$ denotes a non-violating observation. The task-specific geometric rule used to assign y_i is described below. In the implementation, the stored source label is the complementary indicator

$\tilde{b}_i = 1 - y_i$, so $\tilde{b}_i = 1$ marks the violating class used for conformal calibration. Each image is encoded with the frozen world-model encoder $z_i = \text{enc}_\theta(o_i) \in \mathbb{R}^{n_z}$, and the latent features are standardized using the training-split mean and standard deviation. Let \mathcal{I}_{tr} , \mathcal{I}_{val} , and \mathcal{I}_{cal} denote the index sets for the training, validation, and conformal calibration splits of \mathcal{D}_{obs} , respectively. We then train a small MLP score model $c_\psi : \mathbb{R}^{n_z} \rightarrow \mathbb{R}$ with the convention that larger scores indicate safer states. Equivalently, the signed labels are

$$\tilde{y}_i = \begin{cases} -1, & y_i = 0, \\ +1, & y_i = 1, \end{cases}$$

and the classifier is optimized with the hinge loss

$$\mathcal{L}_{\text{obs}}(\psi) = \frac{1}{N_{\text{tr}}} \sum_{i \in \mathcal{I}_{\text{tr}}} \max(0, m - \tilde{y}_i c_\psi(z_i)).$$

Thus, before conformal calibration, $c_\psi(z) > 0$ is classified as non-violating and $c_\psi(z) \leq 0$ is classified as violating.

We calibrate the classifier using only held-out violating examples. Let $\mathcal{I}_{\text{cal}}^- = \{i \in \mathcal{I}_{\text{cal}} : y_i = 0\}$ and define the nonconformity score

$$r_i = \max\{0, c_\psi(z_i)\}, \quad i \in \mathcal{I}_{\text{cal}}^-.$$

This score is zero for a violating calibration point that is already on the violating side of the nominal classifier and positive only when the point is incorrectly placed on the safe side. For miscoverage level δ , we choose

$$\eta = \text{Quantile}_{\lceil (n_{\text{cal}}^- + 1)(1 - \delta) \rceil}(\{r_i : i \in \mathcal{I}_{\text{cal}}^- \} \cup \{\infty\}),$$

where $n_{\text{cal}}^- = |\mathcal{I}_{\text{cal}}^-|$. The conformalized violating set is

$$\mathcal{O}_{\text{conf}} = \{z : c_\psi(z) < \eta\}.$$

Under the exchangeability assumption for future violating observations and the calibration split, this construction covers a future violating latent with probability at least $1 - \delta$.

To match the SLS notation that is used in the main text, we convert the classifier score into a constraint function of the following form:

$$g_\psi(s) = -c_\psi(z), \quad z = \Pi_z s, \quad \mathcal{S}_{\text{safe}} = \{s : g_\psi(s) \leq -\eta\} = \{s : -c_\psi(\Pi_z s) \leq -\eta\}.$$

Here Π_z extracts the zeroth-order latent embedding from the Markov latent state $s = [z, \Delta z, \dots, \Delta^K z]$. This is the sign conversion used by the SLS planner: the raw network score c_ψ is larger for safe states, while the MPC constraint is written in the paper's standard form $g_\psi(s) \leq 0$. The same threshold also appears in the MPPI warm-start soft penalty as a smooth relaxation of $\eta - c_\psi(z) \leq 0$.

Translating Constraints for SLS: Using the obstacle constraint we derived, it can be plugged into SLS and robustly satisfied by tightening the constraint. For nominal constraint satisfaction at step k , in the trajectory ($0 \leq k \leq T$) we require, $\bar{g}_\psi(\zeta_k, v_k) \leq -\eta$, where we define $\bar{g}_\psi(\zeta_k, v_k) = g(z_k)$ such that z_k is the zeroth-order component of the Markov state ζ_k . We then apply the constraint tightening as a function of the linearized constraint and tube (as discussed in Appendix B) to robustly satisfy the constraint for all Markov states, $s_k \in \mathcal{R}_k^s$ and control inputs $u_k \in \mathcal{R}_k^u$ in the tube. In our implementation we also have box constraints for the control input (i.e., $u_k \leq u_{\text{max}}$ and $-u_k \leq -u_{\text{min}}, \forall 0 \leq k \leq T - 1$). Accounting for both upper and lower bound control constraints, this results in a total of $2n_u + 1 = n_c$ constraints in our problem. For succinctness, we collect all the constraints for each time step as follows:

$$g_i(\zeta_k, v_k) + b_i + \tau_{i,k} \leq 0, \quad i = 1, \dots, n_c, \quad k = 0, \dots, T, \quad (23)$$

where $g_i(\zeta_k, v_k)$ is the constraint function, b_i is the nominal bound (adjusted by η for conformalized obstacle constraint indices), and $\tau_{i,k}$ is the constraint tightening.

Task-specific label rules and classifier sizes

Reacher: For Reacher, the violating set is the joint-space box

$$\mathcal{Q}_{\text{obs}} = \{q = (q_1, q_2) : 0 \leq q_1 \leq 3.1415, -2.88 \leq q_2 \leq -2.45\}.$$

This region corresponds to configurations where the second link folds back very close to the first link. In the simulator this is a pose/joint-space obstacle constraint, while in planning it is enforced as a learned latent-space constraint from rendered images. We sample configurations inside \mathcal{Q}_{obs} as violating examples and configurations outside the box as non-violating examples, then render each configuration from the fixed planner camera to obtain the balanced classifier dataset. The Reacher classifier uses a 5-dimensional latent input and an MLP with one hidden layer of width 6 followed by a scalar output, using GELU nonlinearities.

OGBench: For Cube, the classifier enforces a height constraint on the grasped cube/effector state. We synthesize grasped cube configurations, render front-camera observations with the target hidden, and label an observation as violating when the measured gripper height is above the configured threshold

$$h_{\text{grip}} > h_{\text{max}}, \quad h_{\text{max}} = 0.09.$$

Samples with $h_{\text{grip}} \leq 0.09$ are treated as non-violating. The sampled cube positions use the task-space bounds $x \in [0.30, 0.50]$, $y \in [-0.25, 0.25]$, and $z \in [0.02, 0.30]$. If precomputed dataset splits are unavailable, the script creates stratified source-training and calibration splits before carving out a validation subset. The Cube height classifier uses a 12-dimensional latent input and a linear score model $c_\psi(z) = w_\psi^\top z + b_\psi$, i.e., a single learned separating vector and bias.

Rope: For Rope, each sampled task state is $x = (r, h, w)$, where r is the endpoint reach coordinate, h is the endpoint height, and w is the rope width. The task-space bounds are

$$r \in [-0.10, 0.30], \quad h \in [1.16, 1.35], \quad w \in [0.20, 0.75].$$

The obstacle is a speedbump-shaped barrier in the (r, h) task-space slice, placed on the tabletop over the active reach interval $[0.05, 0.15]$. Formally, it is the upper half of an ellipse with center reach $r_c = 0.10$, half-width $a = 0.05$, base height $z_{\text{base}} = 0.75$, and peak height $z_{\text{peak}} = 0.91$:

$$b(r) = z_{\text{base}} + (z_{\text{peak}} - z_{\text{base}}) \sqrt{\max \left\{ 0, 1 - \left(\frac{r - r_c}{a} \right)^2 \right\}}.$$

Thus the bump rises 0.16 above the table at its midpoint and returns to the table height at $r = 0.05$ and $r = 0.15$. Because the rope sags as a function of width, we estimate a sag profile $s(w)$ from simulated proxy-rope configurations and use

$$\ell(x) = h - s(w)$$

as the approximate lowest rope height. A rope state is labeled violating when

$$r \in [0.05, 0.15] \quad \text{and} \quad \ell(x) \leq b(r),$$

and non-violating otherwise. This gives a formal task-space obstacle that visually behaves like a smooth speedbump the rope must clear; the classifier then learns the corresponding latent-space constraint from fixed-camera renderings. The Rope classifier uses a 12-dimensional latent input and an MLP with one hidden layer of width 12 followed by a scalar output, using GELU nonlinearities.

D.4.4 Calibrating the Latent Support

To further validate the exchangeability assumption at planning time, we estimate and conformalize the support of our calibration dataset. SIGReg loss (5) structures the latent space such that the raw latent vectors follow an isotropic multivariate Gaussian distribution with unit variance. Thus, the Markovian latent vector also inherits the Gaussian structure, but is not guaranteed to be isotropic due to correlation between successive latent states. Hence, we learn both the shape and size of the ellipsoidal geometry characterizing the support of our calibration dataset, and constrain our planner

to remain within it. We refer to this constraint as the in-distribution (ID) latent ellipsoid. To obtain this constraint we use the same calibration dataset, \mathcal{D}_{cal} and partitions, $\mathcal{D}_{\text{cal}}^{(1)}/\mathcal{D}_{\text{cal}}^{(2)}$ from Section D.4.2. We compute the sample covariance of the states, Σ_{ID} , using states, s_i , in $\mathcal{D}_{\text{cal}}^{(1)}$, and fix the mean $\mu = 0$. We then compute the Mahalanobis distance $d_i = s_i^\top \Sigma_{ID}^{-1} s_i$ for each state, $s_i \in \mathcal{D}_{\text{cal}}^{(2)}$. Lastly, we compute the $(1 - \alpha_{ID})$ -quantile q_{ID} of distances d_i to form the boundary of our support, as defined in (11). Formally, this provides a boundary between in-distribution points which we assume to be $(1 - \alpha_{ID}) \cdot 100\%$ and outlier points corresponding to $\alpha_{ID} \cdot 100\%$ of points in the calibration dataset, $\mathcal{D}_{\text{cal}}^{(2)}$. For each of our experiments we set $\alpha_{ID} = 0.1$.

D.5 Safety Baseline Implementation

HJ-filtered safety baseline. We implement the HJ-filtered baseline as a latent safety shield applied on top of a nominal planner. We use the same encoder and latent dynamics notation as in the main method: an observation o_t is encoded as

$$z_t = \text{enc}_\theta(o_t),$$

and the planner state is the Markov latent state

$$s_t^z = [z_t, \Delta z_t, \dots, \Delta^K z_t].$$

Actions are represented in the normalized action coordinates used by the learned world model:

$$\bar{u} = \text{clip}\left(\frac{u - \mu_u}{\sigma_u}, u_{\min}, u_{\max}\right), \quad u_{\min} = -2, \quad u_{\max} = 2.$$

The HJ baseline rolls out the learned one-step latent dynamics

$$\hat{s}_{t+1}^z = f_\phi(s_t^z, \bar{u}_t).$$

The HJ baseline uses its own obstacle classifier convention, inherited from the HJ-filtered baseline implementation, which is distinct from the constraint classifier notation used by our SLS planner in the main body. To avoid overloading g_ψ , we denote the HJ classifier score by c_ω . Its signed safety margin is

$$\ell_{\text{HJ}}(s) = c_\omega(s) - \tau_{\text{HJ}},$$

where τ_{HJ} is the HJ classifier decision threshold, calibrated separately from the SLS constraint margin η . In our experiments, τ_{HJ} is selected using the same held-out calibration split used for conformal safety calibration. Under this baseline convention, positive values indicate states classified as safe and non-positive values indicate states classified as unsafe. Depending on the experiment configuration, the raw margin is optionally transformed by a monotone squashing function,

$$\ell_{\text{HJ}}(s) \leftarrow \tanh(\ell_{\text{HJ}}(s)) \quad \text{or} \quad \ell_{\text{HJ}}(s) \leftarrow \tanh(2\ell_{\text{HJ}}(s)).$$

We train a PyHJ/Avoid-DDPG recovery policy π_{HJ} and critic V_{HJ} in the learned latent dynamics. Episodes are initialized from cached latent states. For a sampled action \bar{u}_t , the imagined next latent state is

$$\tilde{s}_{t+1}^z = f_\phi(s_t^z, \bar{u}_t),$$

and the transition signal is the next-state safety margin:

$$r_t = \ell_{\text{HJ}}(\tilde{s}_{t+1}^z).$$

An episode terminates when $\ell_{\text{HJ}}(\tilde{s}_{t+1}^z) \leq 0$. The learned critic therefore estimates recoverability under the latent recovery policy. At deployment, the HJ barrier score is

$$B_{\text{HJ}}(s) = \min\{\ell_{\text{HJ}}(s), V_{\text{HJ}}(s)\}.$$

Given a nominal raw action u_t^{nom} from the base planner, the filter first normalizes it to \bar{u}_t^{nom} and predicts

$$\hat{s}_{t+1}^{z, \text{nom}} = f_\phi(s_t^z, \bar{u}_t^{\text{nom}}).$$

The nominal action is executed only if the current state is classified as safe and the predicted next state is accepted by the learned barrier. Otherwise, the filter replaces it with the HJ recovery action:

$$\bar{u}_t = \begin{cases} \pi_{\text{HJ}}(s_t^z), & \text{if } \ell_{\text{HJ}}(s_t^z) \leq 0 \text{ or } B_{\text{HJ}}(\hat{s}_{t+1}^{z, \text{nom}}) \leq \epsilon, \\ \bar{u}_t^{\text{nom}}, & \text{otherwise.} \end{cases}$$

We use $\epsilon = 0$ unless otherwise specified. The selected normalized action is then unnormalized before execution:

$$u_t = \bar{u}_t \sigma_u + \mu_u.$$

Latent Policy Barrier baseline. We also evaluate an adapted Latent Policy Barrier (LPB) baseline, which incorporates a nearest-neighbor in-distribution barrier as a regularizer in the nominal iLQR planner. First, we build a bank of safe Markov latent states from the training dataset. Each image is encoded as $z_t = \text{enc}_\theta(o_t)$ and converted into the same Markov latent state s_t^z used by the planner. Unsafe states are removed using the analytical ground-truth obstacle check used to label obstacle data. Thus, the prototype bank contains only latent states whose corresponding environment configurations are collision-free under the geometry-based safety test.

The remaining safe states are whitened,

$$\tilde{s} = \frac{s - \mu_{\text{LPB}}}{\sigma_{\text{LPB}}},$$

and M prototype states $\mathcal{P} = \{p_j\}_{j=1}^M$ are sampled from the whitened safe set. We calibrate a distance threshold ρ as a chosen quantile of nearest-prototype distances on held-out non-prototype safe states. At planning time, each predicted latent state is scored by

$$d_{\text{LPB}}(s) = \min_{p_j \in \mathcal{P}} \left\| \frac{s - \mu_{\text{LPB}}}{\sigma_{\text{LPB}}} - p_j \right\|_2.$$

The LPB penalty is zero inside the calibrated safe tube and grows quadratically outside it:

$$c_{\text{LPB}}(s) = [d_{\text{LPB}}(s) - \alpha_{\text{LPB}}\rho]_+^2,$$

where α_{LPB} is the threshold scale and $[\cdot]_+ = \max(\cdot, 0)$.

Unlike the HJ filter, LPB does not override actions after planning. Instead, it augments the iLQR trajectory objective. For a nominal latent trajectory $\zeta = \{\zeta_k\}_{k=0}^T$ and normalized controls $\mathbf{v} = \{v_k\}_{k=0}^{T-1}$, the LPB-regularized objective is

$$J_{\text{LPB}} = \sum_{k=0}^{T-1} (q_s \|\zeta_k - \zeta_g\|_2^2 + q_u \|v_k\|_2^2) + q_f \|\zeta_T - \zeta_g\|_2^2 + \lambda_{\text{LPB}} \frac{1}{T} \sum_{k=0}^{T-1} c_{\text{LPB}}(\zeta_k),$$

where ζ_g is the goal Markov latent state. In our implementation, the LPB term is differentiable through the latent trajectory, and its gradient and Hessian are included in the iLQR backward pass. Thus, LPB biases the planner toward trajectories that remain close to the empirical distribution of safe latent states while still optimizing the original goal-reaching cost.

Constrained iLQR. For our constrained iLQR implementation we solve Problem (14) without constraints (14c)-(14d), and remove the constraint tightening, $\tau_{i,k}$ in (14e). This results in a constrained safety-aware problem, but does not require a trajectory to robustly satisfy constraints—unlike SLS. Furthermore we have two variants, one with the in-domain latent ellipsoid constraint and one without. Outside of these changes, both of these approaches still use the same parameters and configurations as the SLS implementation.

D.6 Hardware Implementation

Our hardware experiment is the physical equivalent of our simulated bimanual rope task.

Hardware: The bimanual setup consists of two KUKA iiwa arms placed 1.5 meters apart, each with a Schunk end-effector. The rope is attached to the arms via two 3D-printed clips that are screwed into the Schunk end-effector. Figures 12 and 13 provide visualizations of the hardware setup. To capture images, we used an Intel RealSense D415 camera (and use only the RGB images).

Data Collection: We parameterize the rope manipulation task with a three-dimensional task space: reach, the shared forward/backward displacement of the end effectors; height, their shared vertical position; and width, their lateral separation. Each task-space command is converted to bimanual joint targets using Drake [62] inverse kinematics, with the two grippers moved symmetrically about the rope centerline. Data is collected in shards. At the start of each shard, the arms return to a fixed home pose, which defines the nominal task origin. We then sample 25 random task-space knots within the workspace bounds, enforcing a minimum separation between consecutive knots. We linearly interpolate between each pair to produce a dense sequence of intermediate task targets. This waypoint-based trajectory gives broad workspace coverage while keeping each low-level motion small and feasible for hardware execution. We considered smoother spline trajectories, as used in the sibling simulation experiment, but use piecewise-linear waypoint segments on the real system because they make the commanded task deltas explicit, easier to validate for collision and speed limits, and easier to execute with settling between samples. For each interpolated target, the corresponding joint command is executed, the arms are allowed to settle, and an RGB image is captured. We record the planned task-space delta together with the measured end-effector state and executed task-space delta, producing image-action pairs for training the dynamics model.

Model Training: We collect our data with a data structure similar to that of all our data collection scripts for simulation. This enables us to swap in our hardware data directly with our simulation data. All the model parameters are equivalent to the corresponding simulation experiment for nominal planning.

Planning: For nominal planning, we used MPPI to warm-start trajectory generation at each step and then iLQR to track it, similar to the Push-T approach described in Section D.2. Since the low-level controller requires a minimal actuation in order to move, we define

$$J_{\text{act}}(\mathbf{v}) = \lambda_{\text{act}} \sum_{t=0}^{T-1} \left[\max \left(0, \frac{\delta_{\text{min}} - \|v_t\|_{\infty}}{\delta_{\text{min}}} \right) \right]^2, \quad (24)$$

where δ_{min} is the minimum actuation. Since the bimanual rope manipulation task has 3 actions controlling the distance between the two end-effectors (width), the height of the two end-effectors, and the depth of two end-effectors (reach), we only require one of the actions to meet the minimum actuation requirement. Thus, we compute the infinity norm of the nominal action $\|v_t\|_{\infty}$ to get the maximum value among the 3 actions and penalize it if it is below the minimum actuation threshold, δ_{min} . Thus, the cost used for the real-world experiment $J_{\text{mppi}}^{\text{real}}(\zeta, \mathbf{v})$ is,

$$J_{\text{MPPI}}^{\text{real}}(\zeta, \mathbf{v}) = \underbrace{J_{\text{MPPI}}(\zeta, \mathbf{v})}_{\text{Eq. (17)}} + J_{\text{act}}(\mathbf{v}). \quad (25)$$

Similar to the simulation experiment (for Push-T), we use iLQR to track the MPPI-generated reference trajectory, and provide the next control input to apply at each MPC step. The MPPI parameters and iLQR parameters are available in Table 8.

Table 8: MPPI and iLQR planning parameters for the real-world rope manipulation task.

Module	Parameter	Value	Parameter	Value
MPPI Warm-start	Horizon (H^{mppi})	60	Control Weight (R^{mppi})	$0.02I_3$
	Samples	512	Reward Weight	25.0
	Updates	6	Noise Scale	0.2
	State Stage Weight (Q^{mppi})	$1.5I_{24}$	Filter β	0.65
	State Terminal Weight (Q_T^{mppi})	$1000.0I_{24}$		
Actuation Penalty	Minimum Threshold (δ_{\min})	0.006	Penalty Weight (λ_{act})	10.0
iLQR Tracking	Horizon (T)	10	State Terminal Weight (Q_T)	$50.0I_{24}$
	State Stage Weight (Q)	$0.5I_{24}$	Control Weight (R)	$0.02I_3$

E Extended Results

In this section, we provide additional experimental results beyond those reported in the main text. Figures 8 - 11 contain our nominal planning time-lapses where we visualize our goal-reaching performance. Figures 12 - 13 contain our nominal planning time-lapses for our bimanual rope manipulation hardware experiment and figure 14 validates convergence of our planner to the goal configuration in the task space for real hardware. Figures 15 - 17 contain time-lapse comparisons of safe vs. unsafe plans demonstrating SLS²'s ability to enforce safety. Figures 18 - 20 visualize the reachable tubes in the latent space. Lastly, we include the LPB safety baseline on Reacher, the LPB safety baseline on Cube, and the Le-WM nominal planning baseline on Push-T.

E.1 Nominal Trajectory Time-lapses

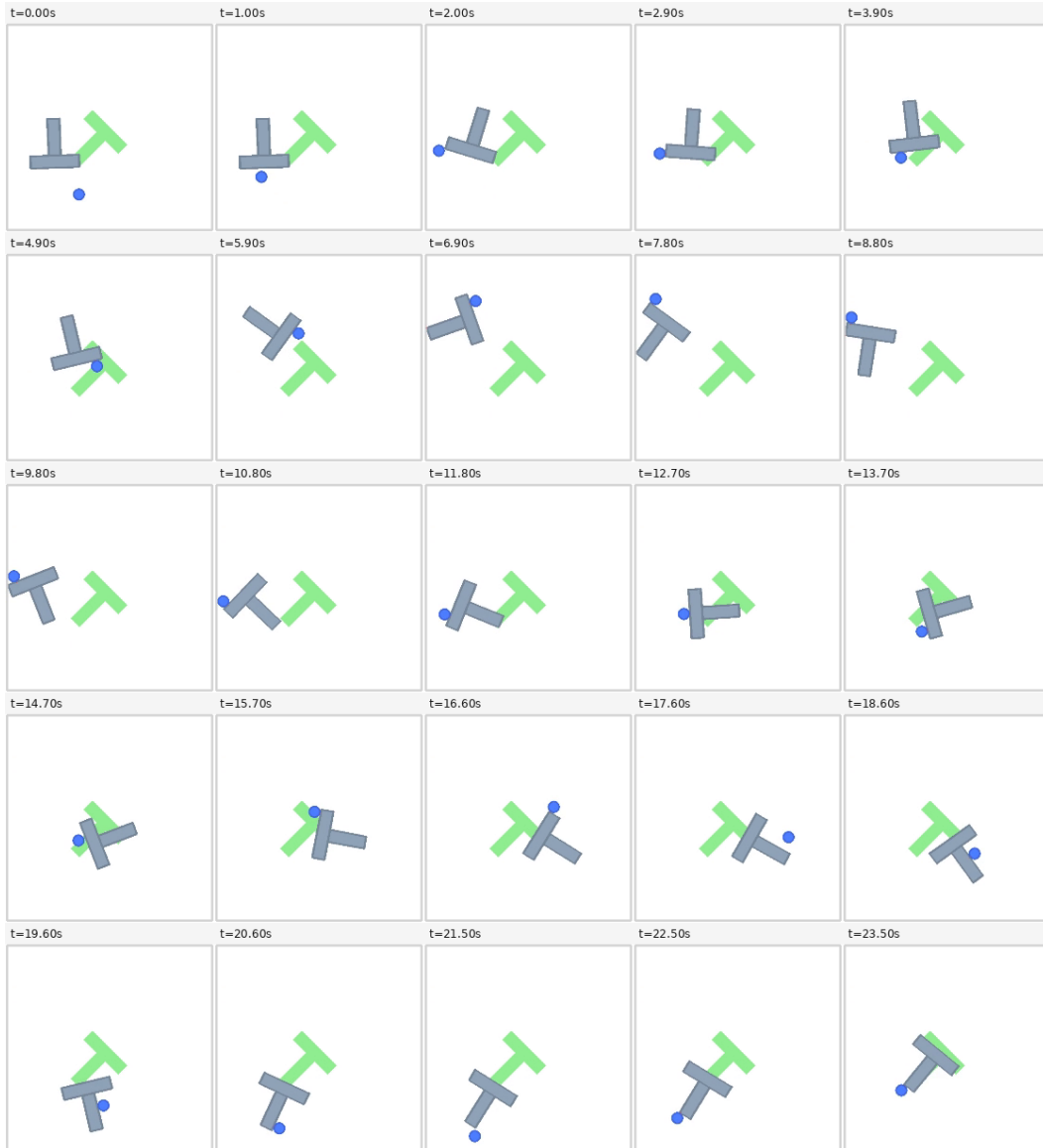


Figure 8: **Push-T**. iLQR-based long-horizon planning, with MPPI warm-start. Our planner is able to successfully make and break contact with the T while executing complex maneuvers in order to push the T to the goal.

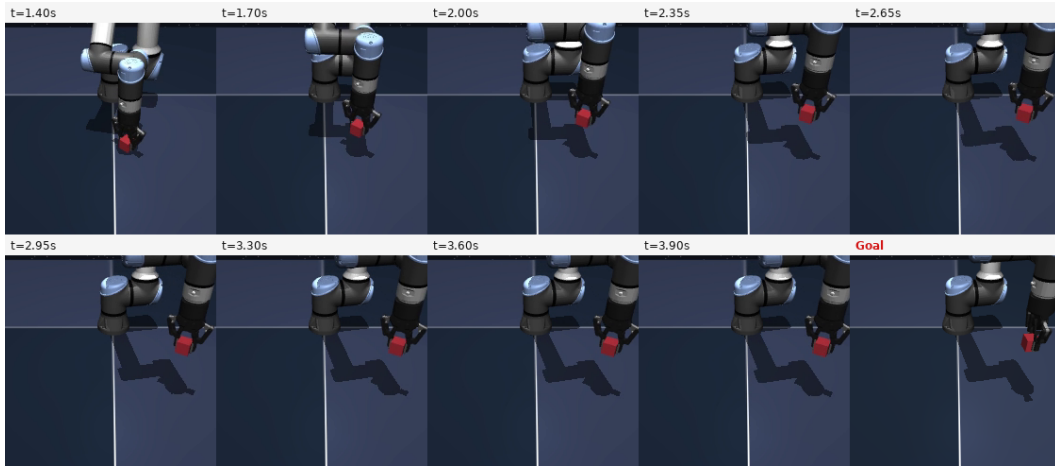


Figure 9: **Cube.** iLQR-based long-horizon planning. The planner successfully moves the cube to the goal pose without dropping it, unlike the baseline approaches.

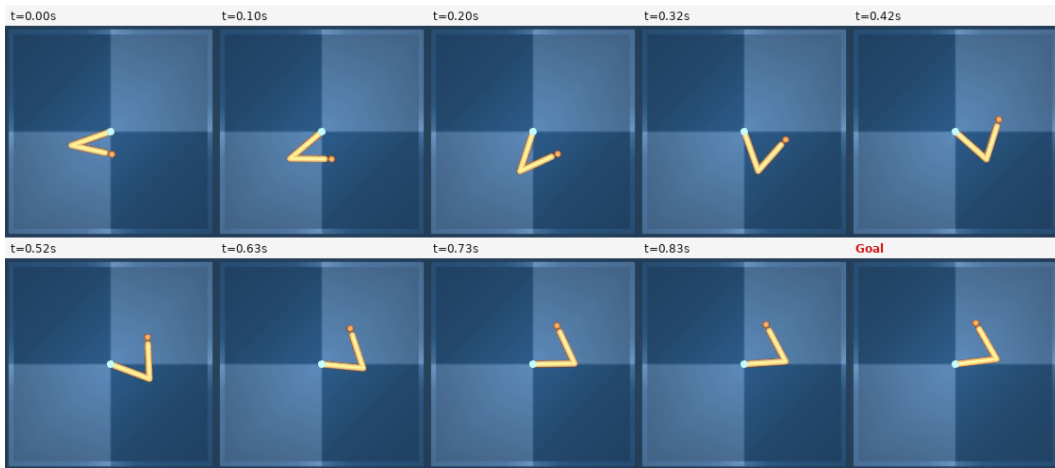


Figure 10: **Reacher.** iLQR-based long-horizon planning. Similar to Figure 9, our gradient-based planner is able to successfully move to the goal in an efficient manner.

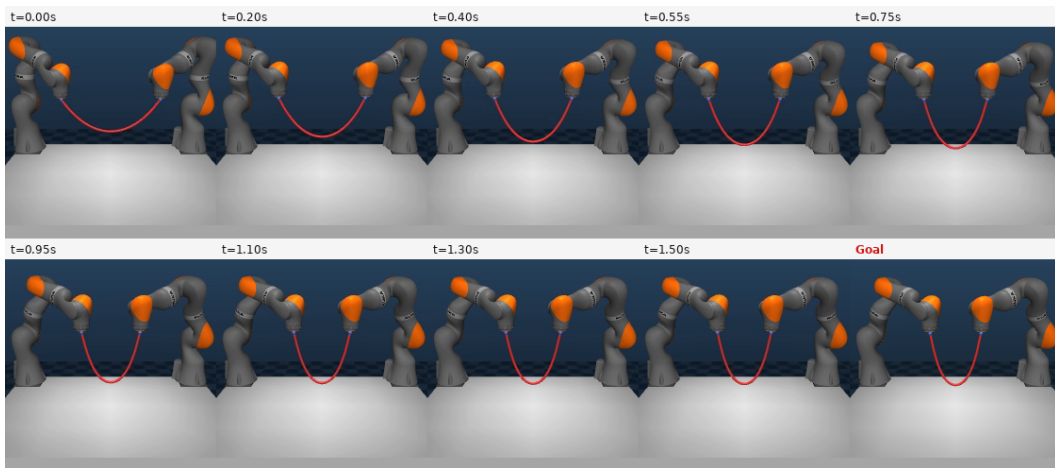


Figure 11: **Rope.** iLQR-based long-horizon planning. The planner is able to correctly manipulate the deformable rope to reach the goal configuration.

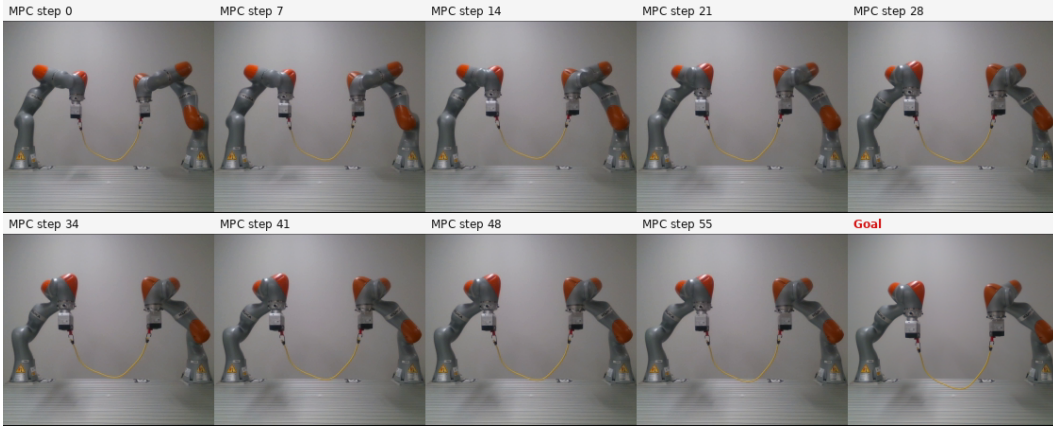


Figure 12: **Real.** iLQR-based long-horizon planning on real hardware directly from pixels without state estimation. In this example, we demonstrate the arms moving forward to reach the goal image.

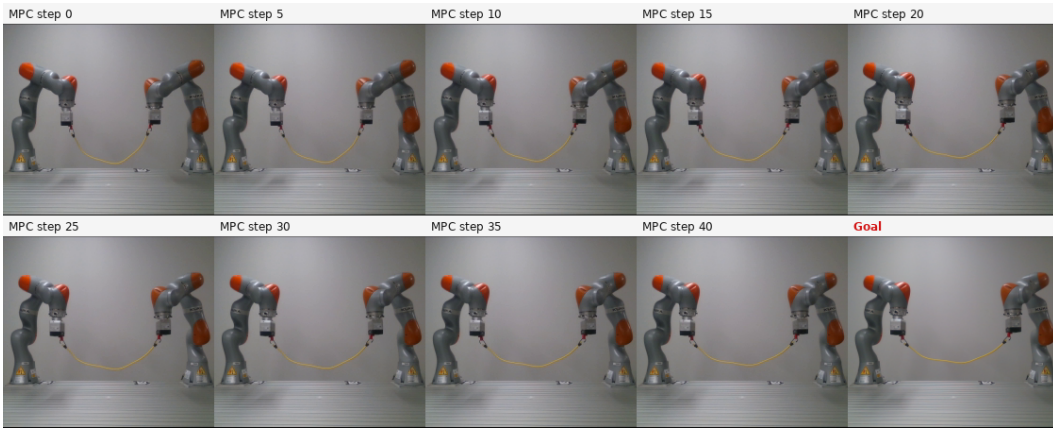


Figure 13: **Real.** Another example of iLQR-based long-horizon planning on real hardware directly from pixels without state estimation. In this example, the bimanual setup has to stretch the rope to reach the goal image.

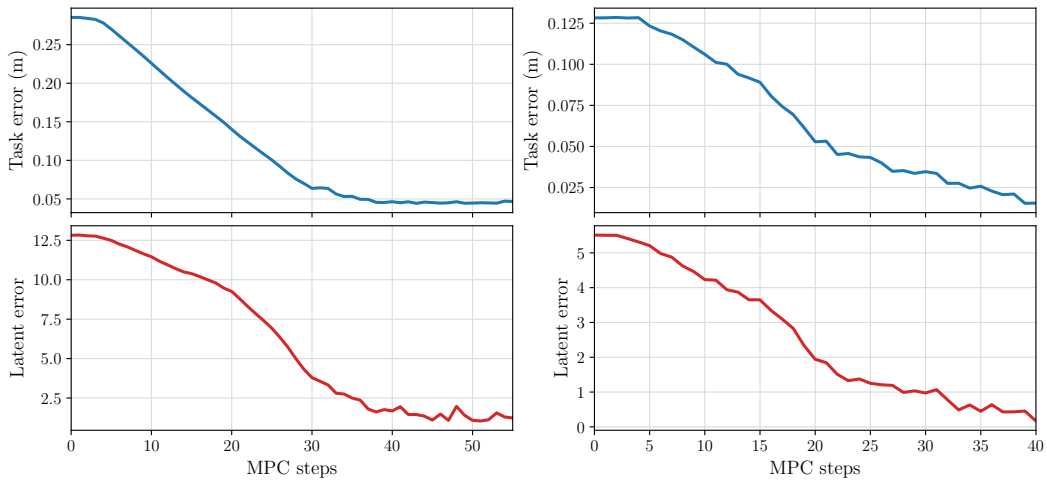


Figure 14: **Real.** Task and latent space error plots demonstrating convergence to the desired goal state on hardware. Plot for Figure 12 on the left and plot for Figure 13 on the right.

E.2 Safe vs. Unsafe Time-lapses

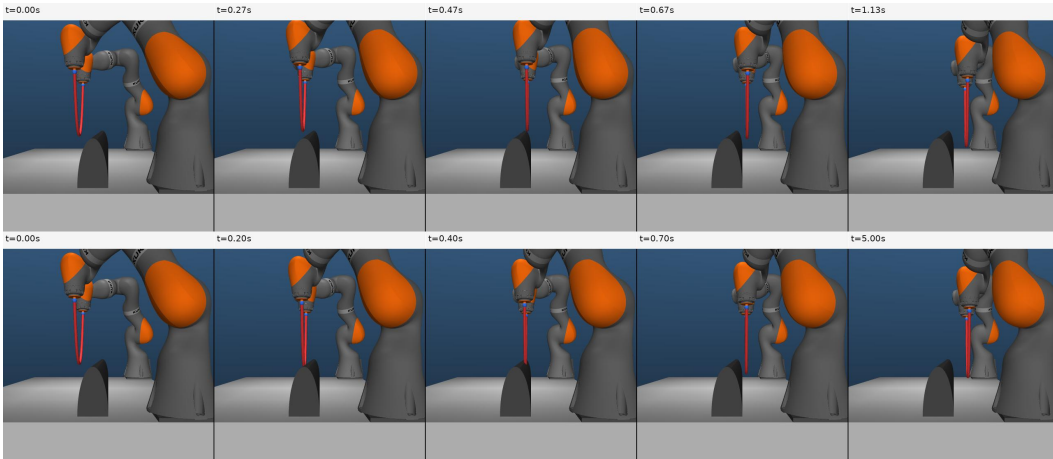


Figure 15: **Rope**. Safe (top) plan vs. unsafe (bottom) plan time-lapse. The SLS² planner successfully maneuvers the rope over the obstacle while reaching the goal image, unlike the nominal planner which collides with the obstacle.

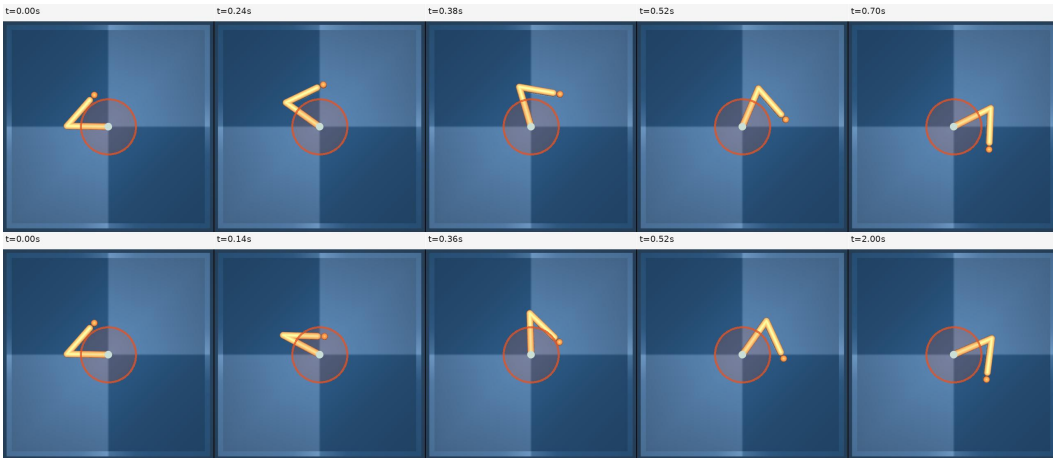


Figure 16: **Reacher**. Safe (top) plan vs. unsafe (bottom) plan time-lapse. Similar to Figure 15, the nominal planner traverses through the “unsafe” region and violates the pose constraint, while SLS² avoids the unsafe region.

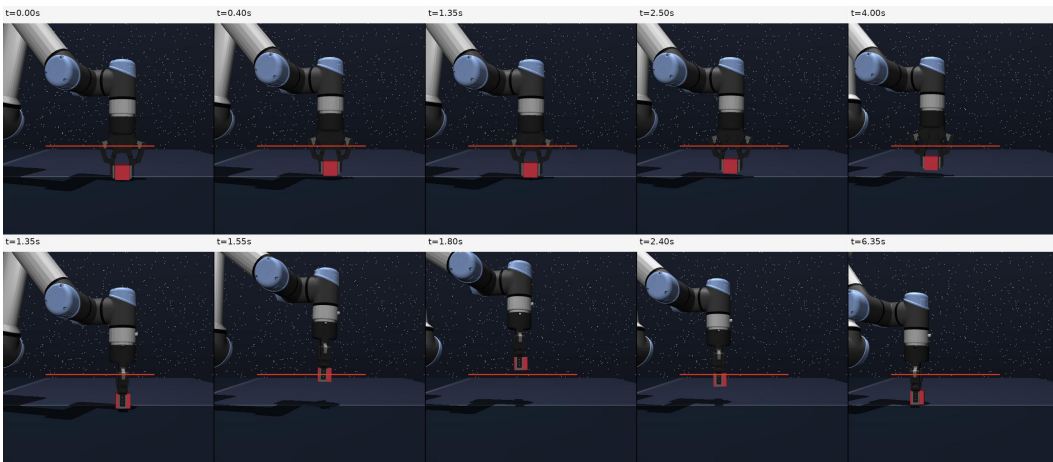


Figure 17: **Cube**. Safe (top) plan vs. unsafe (bottom) plan time-lapse. SLS² maintains the cube below the height threshold while reaching the goal.

E.3 Latent Reachability Tubes

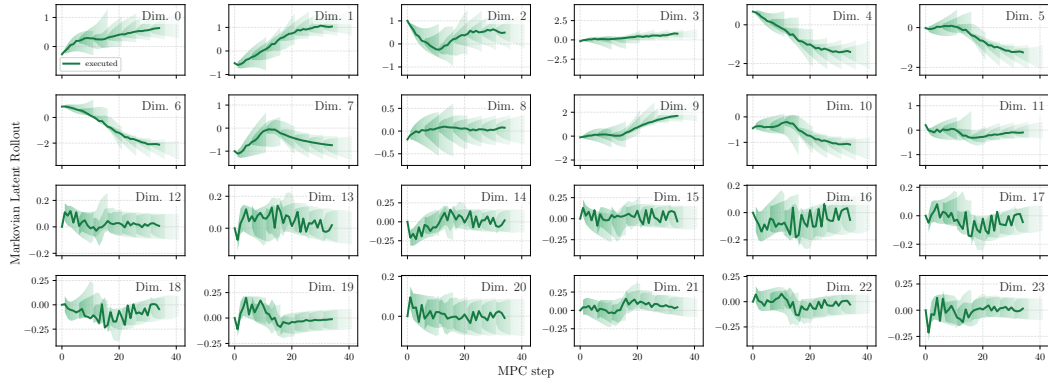


Figure 18: **Rope**. Latent reachability tubes for the full Markov state computed via SLS².

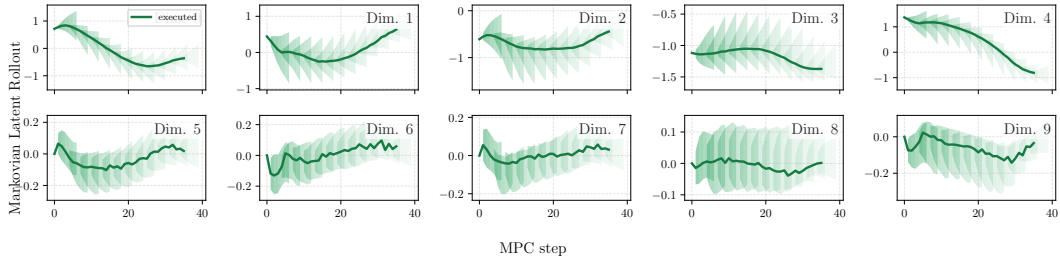


Figure 19: **Reacher**. Latent reachability tubes for the full Markov state computed via SLS².

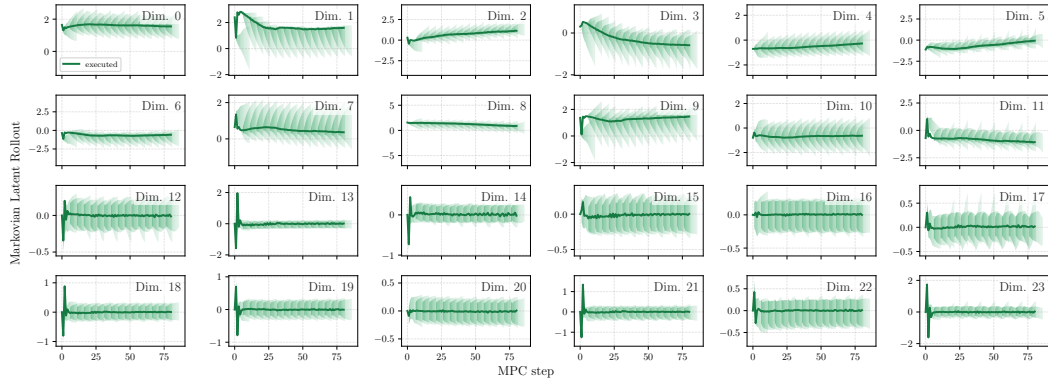


Figure 20: **Cube**. Latent reachability tubes for the full Markov state computed via SLS².