BILINEAR WORLD MODELS: LEARNING REPRESENTATIONS WITH STRUCTURED DYNAMICS FOR EFFICIENT CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

World models jointly learn latent representations and dynamics that predict how high-dimensional observations evolve under actions. In this work, we propose a JEPA-style world model in which, rather than learning arbitrary latent dynamics, we restrict them to follow a bilinear parameterization. This structure enables efficient planning and control while shifting the modeling burden onto the encoder, encouraging richer representations that expose the controllable geometry of the system. In particular, this structured parameterization allows us to structurally enforce action recoverability, thereby preventing representation collapse by construction. Although prescribing a bilinear parametrization may appear restrictive, we show that a broad class of nonlinear dynamical systems admits a transformation under which the dynamics become bilinear. Empirically, we show across standard 2D and 3D control tasks that representations with bilinear-parameterized dynamics can be learned directly from high-dimensional observations, reducing planning time by nearly three orders of magnitude while retaining or even improving control accuracy. We also propose more demanding regimes of longer-horizon planning and real-time control, and demonstrate that our method succeeds in both, moving JEPA-style world models beyond short-horizon offline planning.

1 INTRODUCTION

World models learn a latent representation of an environment together with dynamics that predict how it evolves under actions. This allows planning from high-dimensional observations, such as images, to be performed in a lower-dimensional latent space. Modern approaches typically use neural networks to learn both the encoder that maps observations to latent states and the dynamical model that predicts their evolution. We argue that such flexibility in the dynamical model is often unnecessary and counterproductive for planning and control, motivating structured dynamics.

(C1) We propose to learn world models with structured, bilinear-parameterized latent dynamics.

Rather than learning arbitrary latent dynamics, we restrict their state dependence through a bilinear parameterization followed by a structured normalization. This requires the encoder to discover a representation that satisfies this structure, thus shifting expressivity from the dynamical predictor to the encoder itself. An immediate consequence is that actions can be recovered from latent transitions, thus preventing representation collapse by construction. When choosing the structure to impose, it is desired to have dynamics that enable efficient planning and control while remaining expressive enough to represent a broad class of systems under an appropriate transformation. A bilinear parameterization satisfies both requirements.

(C2) We show that bilinear latent dynamics are expressive for a broad class of nonlinear systems and that the resulting structured parameterization enables efficient control.

To establish expressiveness, we leverage Koopman theory and show that any action-affine nonlinear system admits a transformation under which its dynamics become bilinear. This shows that fixing bilinear dynamics still allows a broad class of nonlinear systems to be represented. The same structure is then exploited for control by replacing computationally expensive zeroth-order methods such as CEM with a gradient-based planner that directly uses the sensitivity of the predicted endpoint to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

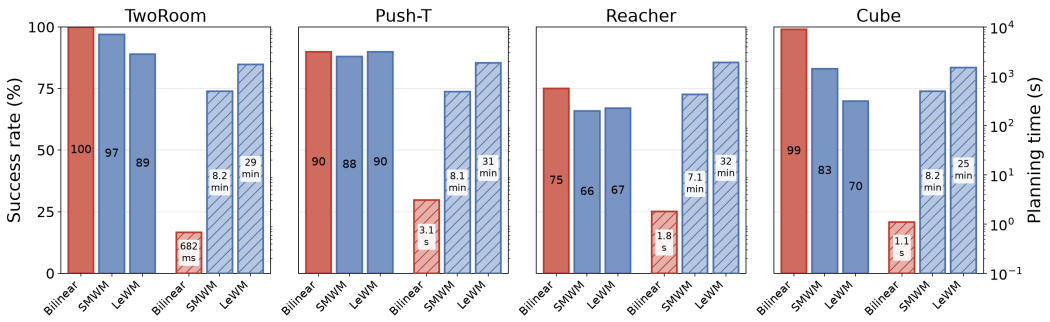


Figure 1: **Planning success and time across four benchmarks.** Solid bars report success rate over 100 evaluation episodes, each defined by a start–goal pair with a goal offset of 25 steps; hatched bars report the average total wall-clock planning time per episode on a log scale. *Our Bilinear World Model matches or improves success across all environments while reducing planning time by two to three orders of magnitude.*

the action sequence. This enables substantially more efficient optimization in the latent space while preserving strong control performance. Both advantages are reflected directly in our experiments.

(C3) Bilinear World Models retain or improve control performance while reducing planning time by three orders of magnitude.

We evaluate Bilinear World Models on four standard 2D and 3D control tasks—TwoRoom, PushT, Reacher, and Cube—against state-of-the-art JEPA-style world models. Despite using substantially more restricted dynamics, our models match or improve success rate across the tasks. At the same time, exploiting the structured, differentiable dynamics reduces planning time by approximately three orders of magnitude, reducing several minutes of planning to seconds. This gain in efficiency enables evaluation on more demanding control regimes.

(C4) We extend world models beyond short-horizon offline planning to longer-horizon and real-time control.

We evaluate progressively longer planning horizons and moving-goal environments that require continuous replanning. Bilinear World Models degrade less as the horizon increases while their planning cost scales substantially better than sampling-based alternatives. Their low control cost also allows the same controller to operate in real time, whereas competing world models require a specialized real-time sampling method. Across the moving-goal tasks, Bilinear World Models achieve substantially higher success, showing that structured latent dynamics enable world models to operate beyond the short-horizon offline regime in which they are typically evaluated.

2 Related Work

Latent World models. Given a dataset of observed transitions generated by an unknown dynamical system, world models aim to jointly learn a latent representation and latent dynamics that capture the behavior of the underlying system (Ha & Schmidhuber, 2018). Initial approaches, such as PlaNet (Hafner et al., 2018) and Dreamer (Hafner et al., 2020), relied on encoder–decoder architectures, where pixel reconstruction losses encouraged representations that retained information about the environment. More recently, Joint-Embedding Predictive Architectures (JEPAs) (Assran et al., 2023; Chen et al., 2026) have provided an alternative that learns latent representations without relying on reconstruction losses. Despite these differences, both paradigms typically model the latent dynamics with flexible neural network predictors.We propose to constrain the latent dynamics through a bilinear parameterization and learn the representation under this structural restriction.

JEPA world models and representation collapse. A challenge in jointly learning flexible latent representations and latent dynamics is the existence of trivial solutions in which both the encoder

and the dynamics predictor collapse to a constant representation (Balestriero & LeCun, 2025). This phenomenon, known as *representation collapse*, is a central limitation of JEPA-based world models. Early approaches sought to prevent collapse by regularizing the distribution of the latent representations (Maes et al., 2026; Assran et al., 2025). More recently, Sensorimotor World Models (SMWM) (Ivashkov et al., 2026) introduced an auxiliary inverse model that predicts the action from latent transitions. Because the action cannot be recovered from a collapsed representation, this auxiliary objective discourages collapse. In our setting, using a bilinear parameterization of the latent dynamics allows action recoverability to be enforced directly through the model structure, thereby preventing representation collapse by construction.

Bilinear dynamical systems. Bilinear systems constitute a classical class of nonlinear control systems in which the dynamics are linear in the state and in the action independently, but contain multiplicative state–action interactions (Mohler, 1974). Their structured form has enabled the development of specialized tools for controllability, stabilization, optimal control, and trajectory optimization, while retaining substantially more expressive dynamics than linear models (Elliott, 2007). More recently, bilinear models have also emerged in data-driven control as structured approximations of nonlinear dynamical systems (Otto et al., 2024). In particular, for control-affine systems, the corresponding Koopman generator inherits an affine dependence on the control, naturally giving rise to bilinear dynamics in lifted coordinates (Bruder et al., 2020).

Koopman embeddings. Koopman operator theory provides a principled framework for representing nonlinear dynamical systems through linear evolution in a lifted, potentially infinite-dimensional space of observables (Koopman, 1931; Mezić, 2005). This perspective has motivated data-driven methods that approximate the Koopman operator on finite-dimensional observables (Williams et al., 2015). Deep Koopman methods extend this idea by learning the observables themselves with neural networks, seeking latent representations in which the dynamics are approximately linear (Lusch et al., 2018; Otto & Rowley, 2021). These methods are often developed from state measurements, while approaches operating on high-dimensional observations commonly employ encoder–decoder objectives (van der Heijden et al., 2021). In contrast, we learn structured latent dynamics directly from high-dimensional observations using a JEPA objective, without requiring input reconstruction.

3 Bilinear World Models

Let $F_s : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ be an *unknown* discrete-time dynamical system, where a current state $s_t \in \mathcal{S} \subseteq \mathbb{R}^n$ and an action $a_t \in \mathcal{A} \subseteq \mathbb{R}^m$ determine the state evolution $s_{t+1} = F_s(s_t, a_t)$. In this work, we seek an encoder $\phi : \mathcal{S} \to \mathcal{Z} \subseteq \mathbb{R}^d$ that maps each state $s \in \mathcal{S}$ to a latent representation $z = \phi(s) \in \mathcal{Z}$ in which the unknown dynamics F_s admit a simpler structured representation. In particular, we begin from latent dynamics with a bilinear parameterization,

$$z_{t+1} = Az_t + \widetilde{M}(z_t)a_t, \quad \widetilde{M}(z_t) := B + Cz_t{}^1. \tag{1}$$

Restricting the structure of the latent dynamics shifts the modeling burden onto the encoder, which must learn a representation in which the prescribed dynamics are sufficiently expressive. In particular, if $\widetilde{M}(z_t)^\top \widetilde{M}(z_t)$ is positive definite, the action that generated a latent transition can be recovered as

$$a_t = \widetilde{M}(z_t)^\dagger (z_{t+1} - Az_t), \quad \widetilde{M}(z)^\dagger := \left(\widetilde{M}(z)^\top \widetilde{M}(z)\right)^{-1} \widetilde{M}(z)^\top. \tag{2}$$

Encouraging recoverability of the action from the state transition has previously been proposed as a mechanism for preventing *representation collapse* (Ivashkov et al., 2026). Unlike SMWM, which learns an unconstrained inverse network, our dynamics admit a closed-form inverse whenever the structured action map is full rank. A convenient way to enforce this condition in practice is to normalize the matrix $\widetilde{M}(z_t)$. We therefore consider the normalized latent dynamics

$$z_{t+1} = Az_t + M(z_t)a_t, \quad M(z) = Q(z)R, \qquad Q(z) = \mathrm{CholQR}\Big(\widetilde{M}(z)\Big), \tag{3}$$

[1] We note that $C \in \mathbb{R}^{d \times m \times d}$ collects d matrices $C_1, \ldots, C_d$, with $Cz := \sum_{i=1}^{d} C_i z_i \in \mathbb{R}^{d \times m}$.

where $Q(z) \in \mathbb{R}^{d\times m}$ is the orthogonal factor obtained through the Cholesky–QR decomposition and $R \in \mathbb{R}^{m\times m}$ is a learned state-independent matrix. We emphasize that the resulting dynamics are no longer strictly bilinear, since the normalization introduces a nonlinear dependence on z. However, their state dependence remains generated by the bilinear parameterization $\widetilde{M}(z) = B + Cz$, which is subsequently normalized before entering the dynamics.

As before, action recoverability requires $M(z_t)^\top M(z_t)$ to be positive definite. Under the normalized parameterization, this condition becomes independent of the latent state and can be controlled directly through the state-independent matrix R,

$$M(z)^\top M(z) = R^\top Q(z)^\top Q(z) R = R^\top R. \tag{4}$$

Two practical considerations must be addressed before formulating the problem. First, optimizing the encoder over the unrestricted space of functions $\phi : \mathcal{S} \to \mathcal{Z}$ is intractable. We therefore restrict it to a finitely parameterized hypothesis class $\phi_\theta \in \mathcal{H}$, with $\theta \in \Theta \subseteq \mathbb{R}^p$. Second, the underlying dynamical system F_s is unknown and is observed only through a finite collection of transitions $\Omega = \{(s_t, a_t, s_{t+1})_i\}_{i=1}^N$. Accordingly, we propose to solve the following optimization problem:

$$\begin{aligned} \underset{\phi_\theta, A, B, C, R}{\text{minimize}} \quad & \frac{1}{N} \sum \|z_{t+1} - Az_t - M(z_t)a_t\|^2 \\ \text{subject to} \quad & R^\top R \succeq \epsilon I, \\ & M(z_t) = \mathrm{CholQR}(B + Cz_t)\, R, \\ & z_t = \phi_\theta(s_t), \qquad z_{t+1} = \phi_\theta(s_{t+1}), \end{aligned} \tag{P}$$

The problem formulation in equation P jointly learns a latent representation of the environment and its dynamics, and can therefore be viewed as a world-model learning problem. Since the model is trained entirely in latent space, without reconstructing the original observations through a decoder, it follows a JEPA-style formulation. However, unlike most latent world-model approaches which parameterize the dynamics with an unconstrained neural network (Ivashkov et al., 2026; Maes et al., 2026), we explicitly restrict the latent dynamics to follow a bilinear parameterization. We remark that the QR decomposition is differentiable, and therefore gradients can be backpropagated through this normalization. Implementation details for solving P, together with an extended discussion of the relationship between the bilinear and normalized dynamics, are provided in Appendix B.1.

4 Existence of Action-Recoverable Bilinear Embeddings

Restricting the latent dynamics to a bilinear form raises the question of whether such representations are sufficiently expressive. In this section, we characterize conditions under which a nonlinear controlled system admits a possibly infinite-dimensional bilinear embedding. Building on Koopman realizations of control-affine systems, we further show that pointwise independent action directions yield a full-rank latent action map, and therefore action recoverability.

We first state the Koopman linear representation theorem, which ensures that for any dynamical system F and a fixed action $a \in \mathcal{A}$, there exists an embedding where the resulting non actuated dynamics are linear in z.

Lemma 1 (Universal Koopman linear representation)**.** *Let $F : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ be a deterministic continuous-time controlled dynamical system. For a fixed action $a \in \mathcal{A}$, there exist a (possibly infinite-dimensional) vector space $\mathcal{Z}$, an injective embedding $E : \mathcal{S} \to \mathcal{Z}$, and a linear operator $K_a : \mathcal{Z} \to \mathcal{Z}$ such that $z = E(s)$ evolves according to*

$$\dot{z} = K_a z. \tag{5}$$

Proof. This is the classical Koopman construction; see for example (Bamieh, 2022, Sec. 1). □

Lemma 1 ensures the existence of a transformation into an embedding space where dynamics are linear in z. However, we note that the linear operator K_a is not necessarily affine on a. To guarantee that we require the original system to be action affine and a mild structural requirement. Moreover, to ensure injectiveness of the control operator we require that the effect actions have in the system for a particular state s are nondegenerate.

Assumption 1 (Affinity in the actions and Lipchitzness)**.** *The dynamical system F_s is* affine in the actions*. That is, there exist functions $f_0, f_1, \ldots, f_m$ such that*

$$F_s(s, a) = f_0(s) + \sum_{j=1}^{m} a_j f_j(s). \tag{6}$$

Moreover, the system is Lipchitz if the functions $f_0, f_1, \ldots, f_m$ are Lipchitz.

Assumption 2 (Independent action directions)**.** *The actions induce linearly independent directions in the system dynamics.*

In particular, for a system affine in the actions (Assumption 1) that requieres,

$$\operatorname{rank}\left[f_1(s) \quad \cdots \quad f_m(s)\right] = m, \qquad \forall s \in \mathcal{S}. \tag{7}$$

Affinity in the actions is a common structure in mechanical systems when the actions correspond to generalized forces or torques, rather than to lower-level actuator signals. Requiring the dynamics to be Lipchitz continuous requires that nearby states can not induce arbitrarily different dynamics. Nontheless, it does not require differentiability and therefore still allows for nonsmooth behaviors, such as those arising from contact interactions. Assumption 2 requires distinct actions to induce distinct direction of motion, at least locally. In particular, no action direction can be reproduced as a linear combination of the effects induced by the remaining actions. Then, whenever the system satisfy this conditions, there exist a transformation were the embeddings follow bilinear dynamics.

Proposition 1 (Action-recoverable bilinear Koopman embedding)**.** *Let Assumptions 1 and 2 hold. Then there exist a possibly infinite-dimensional vector space $\mathcal{Z}$, an injective embedding $E : \mathcal{S} \to \mathcal{Z}$, and linear operators $A : \mathcal{Z} \to \mathcal{Z}$, $B : \mathbb{R}^m \to \mathcal{Z}$ and $C : \mathcal{Z} \to \mathcal{L}(\mathbb{R}^m, \mathcal{Z})$, such that, for $z = E(s)$,*

$$\dot{z} = Az + M(z)a, \qquad M(z) = B + Cz. \tag{8}$$

Moreover, the latent action map $M(z)$ is injective for every $z \in E(\mathcal{S})$. Equivalently,

$$M(z)^T M(z) \succ 0, \qquad \forall z \in E(\mathcal{S}). \tag{9}$$

Proof. See Appendix A. □

Proposition 1 shows that the two structural properties required by our model—bilinear latent dynamics and action recoverability—can hold simultaneously for a broad class of nonlinear controlled systems. Existing Koopman results provide possibly infinite-dimensional bilinear realizations for control-affine systems, while the additional independence assumption on the physical action directions guarantees that the corresponding latent action map is injective, and therefore that actions can be recovered from latent transitions. The result is only an expressiveness argument, as it applies to exact continuous-time, possibly infinite-dimensional representations, whereas Problem P learns a finite-dimensional, discrete-time, normalized approximation. Prior work nevertheless provides empirical evidence that finite-dimensional neural embeddings can approximate useful Koopman representations across a range of dynamical systems (Lusch et al., 2018; Otto et al., 2024; van der Heijden et al., 2021).

5 Control in Normalized Bilinear Systems

A useful world model should not only predict the evolution of a dynamical system, but also enable efficient planning and control between states. In particular, given an initial state z_0 and a goal $z^\star$, we seek a sequence of actions $\mathbf{a} = (a_0, \ldots, a_{H-1})$ that drives the system toward $z^\star$ over a planning horizon H. The structured dynamics introduced in equation 3 make this optimization particularly efficient.

To formalize this objective, let $z_H(\mathbf{a})$ denote the endpoint obtained by recursively applying the learned dynamics under the action sequence $\mathbf{a}$. We optimize

$$\mathcal{L}(\mathbf{a}) = \frac{1}{2}\|z_H(\mathbf{a}) - z^\star\|^2 + \frac{c}{2}\|\mathbf{a}\|^2. \tag{10}$$

The first term measures the error in reaching the goal, while the second regularizes the magnitude of the actions. The objective in equation 10 does not rely on the particular structure of the dynamics and can therefore be optimized with general black-box world models using sampling-based planners such as the Cross-Entropy Method (CEM). CEM repeatedly samples candidate action sequences, evaluates their costs through model rollouts, and updates the sampling distribution toward the best-performing candidates. Because this procedure does not exploit sensitivity information, it can require a large number of model evaluations, making planning computationally expensive, particularly as the planning horizon or action dimension increases.

The advantage of the normalized bilinear parameterization is that the learned dynamics remain explicitly differentiable with respect to both states and actions, allowing the same objective to be optimized using sensitivity information. For the dynamics $z_{t+1} = Az_t + M(z_t)a_t$, define the state Jacobian $F_t := \partial z_{t+1}/\partial z_t = A + \partial(M(z_t)a_t)/\partial z_t$. The Jacobian of the final state with respect to the complete action sequence then takes the form

$$\mathcal{J}(\mathbf{a}) := \frac{\partial z_H(\mathbf{a})}{\partial \mathbf{a}} = \left[\left(\prod_{j=k+1}^{H-1} F_j \right) M(z_k) \right]_{k=0}^{H-1}. \tag{11}$$

The Jacobian in equation 11 characterizes how local perturbations of each action affect the final predicted state. Because $M(z)$ is obtained through differentiable operations, including the Cholesky–QR normalization, these sensitivities can be computed efficiently by automatic differentiation. We can therefore use them to construct a local linear approximation of the endpoint as a function of the action sequence. Substituting this approximation into equation 10 and minimizing the resulting quadratic model gives the update

$$\mathbf{a}^{t+1} = \mathbf{a}^t - \left[\mathcal{J}(\mathbf{a}^t)^\top \mathcal{J}(\mathbf{a}^t) + cI\right]^{-1} \left[\mathcal{J}(\mathbf{a}^t)^\top \left(z_H(\mathbf{a}^t) - z^\star\right) + c\mathbf{a}^t\right]. \tag{12}$$

Applying equation 12 iteratively yields a Gauss–Newton planner Buccini et al. (2023). In practice, several inexpensive modifications improve its robustness, including initializing the action sequence from a small set of candidates, using backtracking line search to avoid overshooting, and adding damping to the linear system for numerical stability. We additionally exploit the structure of the action space and compute the required sensitivities efficiently in batch. Complete implementation details are provided in Appendix C.

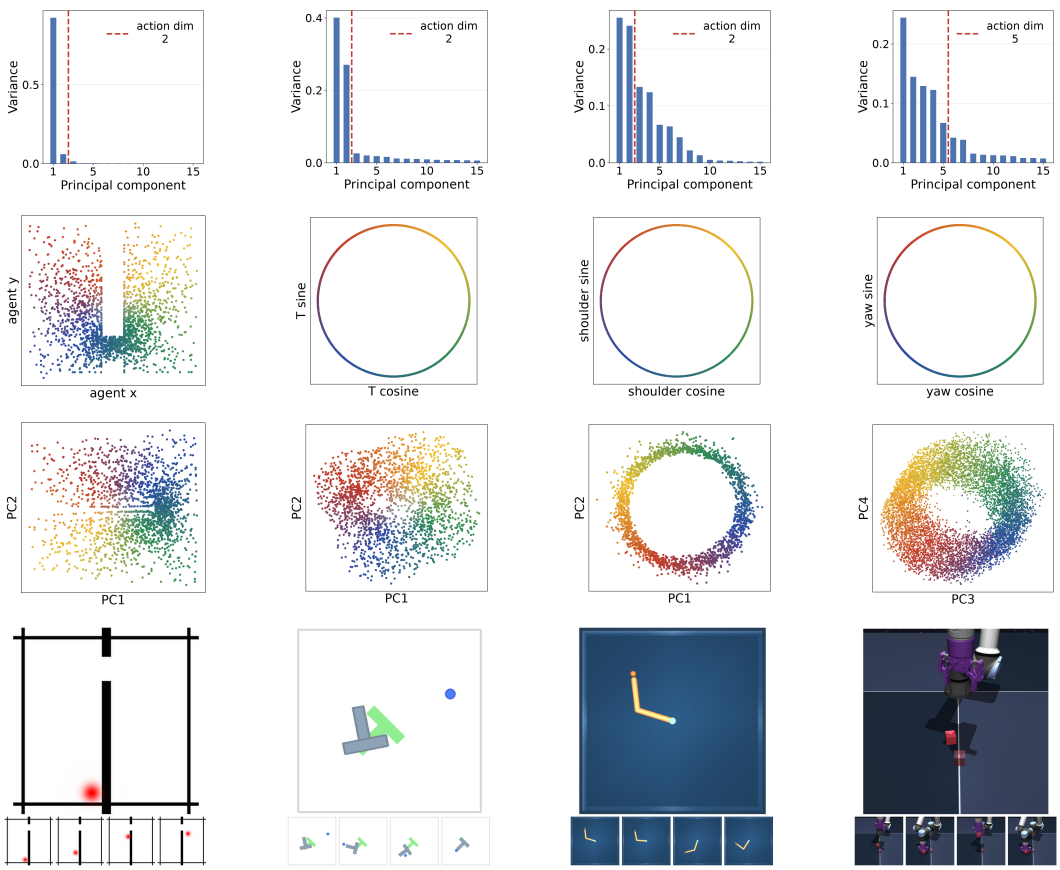
6 Numerical Experiments

In this section,we empirically validate the preceding claims by studying the representations learned under the normalized bilinear parameterization and evaluating their implications for control. We first analyze the geometry of the learned latent space, then compare control performance and planning efficiency against existing JEPA-style world models, and finally evaluate longer-horizon and real-time control regimes.

Tasks. We evaluate our framework on the standard LeWM benchmark suite (Maes et al., 2026), which includes *Two Room* for 2D navigation, *Cube* for 3D manipulation, *Reacher* for continuous control, and *PushT* for contact-rich object interaction. We measure performance using the *Success Rate*, defined as the percentage of 100 start–goal tasks successfully completed within the allowed action budget. We additionally report the planning time t, measured as the computation required by the controller to obtain the action sequence for each task.

Benchmarks. We compare against LeWM (Maes et al., 2026) and Sensorimotor (Ivashkov et al., 2026). We focus on latent world models rather than task-specific controllers because our goal is to evaluate whether structuring the latent dynamics preserves control performance while reducing planning cost within this flexible framework, rather than to develop specialized controllers for the four environments considered here. We select LeWM and Sensorimotor because they are the strongest world-model baselines we identified in the literature for these benchmarks.

Architecture. We adopt the encoder used in LeWM (Maes et al., 2026), a ViT-Tiny with 5M parameters that maps each state to a 192-dimensional latent representation. The predictor follows equation 3 and is therefore parameterized only by the four matrices A, B, C and R.



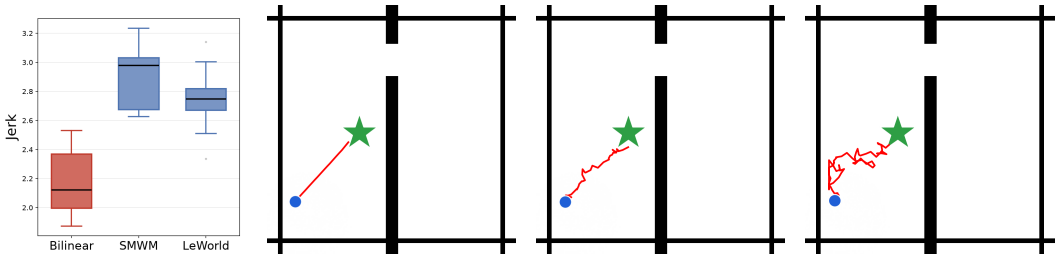
(a) TwoRoom (b) PushT (c) Reacher (d) OG-Cube

Figure 2: **Geometry of the learned latent representations.** (Top) Variance explained by each principal component of the latent embeddings, with the action dimension indicated by the red dashed line. (Middle) Comparison between selected physical state variables and the corresponding projections of the latent embeddings onto their principal components. In TwoRoom, PC1 and PC2 are shown against the agent's horizontal and vertical positions. In PushT, Reacher, and Cube, principal-component projections are shown alongside angular state variables represented through their sine and cosine coordinates. (Bottom) Visualizations of the corresponding environments.

Controllers. For LeWM and Sensorimotor, the main benchmark and long-horizon experiments use CEM (Rubinstein, 1997), matching the planner used in their original evaluations. For our Bilinear World Model, we use the Gauss–Newton planner introduced in Section 5. We additionally evaluate CEM, iCEM (Pinneri et al., 2021), and Gauss–Newton across all three world models in Appendix C.2 to separate the effect of model structure from that of the controller. In the real-time experiments proposed in this section, CEM is too slow to meet the prescribed control budget, and we therefore use iCEM for the neural-dynamics baselines while retaining the same Gauss–Newton controller for our method.

6.1 Learned Representations and Control Performance

Bilinear World Models capture controllable degrees of freedom. Actions induce changes in the physical state of a system. By explicitly structuring how these same actions induce transitions in latent space, our parameterization encourages the encoder to organize representations according to the physical quantities affected by the actions. Figure 2 shows that this correspondence indeed emerges in the learned representations. In TwoRoom (a), the first two principal components closely recover the agent's physical coordinates, with PC1 capturing its horizontal position and PC2 its vertical position. In PushT (b), Reacher (c), and Cube (d), the dominant components recover relevant angular



(a) Bilinear (b) SMWM (c) LeWorld

Figure 3: **Trajectories and action smoothness in TwoRoom.** (Left) Average jerk, measured from changes between consecutive actions, for each world model. (Right) Representative planned trajectories toward the goal. *Bilinear World Models produce lower-jerk action sequences which result in smoother trajectories.*

degrees of freedom, forming circular structures that correspond to sine–cosine representations of the underlying angles. Moreover, across environments, most of the latent variance is concentrated in approximately as many principal components as there are action dimensions, suggesting that the learned manifold is organized around the controllable degrees of freedom of the system.

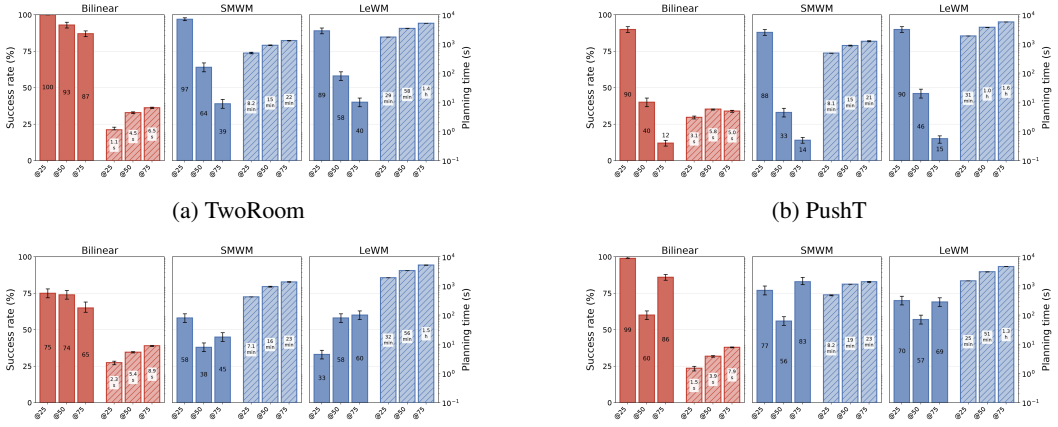
Bilinear World Models achieve high success rates with substantially faster planning. The structured dynamics enable controllers to exploit local sensitivity information, resulting in markedly lower planning cost without sacrificing task performance. As shown in Figure 1, Bilinear World Models match or improve the success rates of the alternative models across all four environments. Our method solves 100% of the TwoRoom tasks and 99% of the Cube tasks, while achieving success rates of 90% on PushT and 75% on Reacher. The largest improvement is observed in Cube, where the success rate exceeds that of the strongest baseline by 17 percentage points. The difference in computational cost is even more pronounced: across the benchmarks, our method reduces planning time by approximately three orders of magnitude, turning planning times of several minutes—up to 7–32 minutes for the baselines—into 0.7–3.1 seconds.

The computational gains are not solely due to the choice of controller. To disentangle the effect of the learned dynamics from that of the planner, we evaluate CEM, iCEM, and Gauss–Newton (GN) on each of the three world models; full results are reported in Appendix C.2. GN is particularly effective when paired with Bilinear World Models, where it can exploit the explicit structure of the latent dynamics, achieving high success at substantially lower planning cost. Applying GN to SMWM or LeWorld does not provide the same combination of performance and efficiency, while iCEM reduces computation relative to CEM at the expense of lower success on the more challenging tasks. Based on this comparison, we use CEM for SMWM and LeWorld in all main benchmark and long-horizon experiments, matching the controller used in their original evaluations, and reserve iCEM only for the real-time setting where CEM cannot satisfy the required planning budget.

Bilinear World Models enable smoother control trajectories. The CEM controllers used in the original baseline implementations optimize primarily for reaching the target and do not explicitly penalize variations in the action sequence. In contrast, the structured dynamics of Bilinear World Models allow such regularization to be incorporated directly into the planning objective, enabling efficiency not only in computation but also in the resulting control trajectories. Figure 3 illustrates this effect in TwoRoom, where our method achieves the lowest average jerk, measured from changes in consecutive actions. This difference is also visible in the planned trajectories: while the baseline methods produce irregular paths with abrupt changes in direction, the Bilinear World Model follows a straight trajectory toward the goal.

6.2 Long-Horizon and Real-Time Control

Bilinear World Models scale robustly to longer planning horizons. We evaluate control performance as the goal offset increases to 25, 50, and 75 steps, denoted @25, @50, and @75, respectively. As shown in Figure 4, increasing the goal offset generally makes the control problem more challenging, although its effect varies across environments. Bilinear World Models remain com-



(a) TwoRoom (b) PushT

(c) Reacher (d) OG-Cube

Figure 4: **Performance across increasing task horizons.** Each panel reports success rate (solid bars) and planning time (hatched bars) for goal offsets of 25, 50, and 75 steps, denoted as @25, @50, and @75. *Bilinear World Models exhibit a smaller degradation as the horizon increases while maintaining substantially lower planning times.*

petitive across all evaluated offsets while preserving a substantial computational advantage over the sampling-based alternatives. In TwoRoom, for example, success remains above 75% at @75, while the alternative models drop to approximately 40%. A similar advantage is observed in Reacher, where increasing the goal offset has a substantially smaller effect on the Bilinear model. In PushT and Cube, performance varies less consistently with the goal offset, but Bilinear World Models continue to achieve comparable success at substantially lower planning cost. This computational advantage becomes increasingly pronounced for longer-horizon tasks: at @75, LeWorld requires nearly 1.4 hours of planning per episode and SMWM approximately 22 minutes, whereas our method requires only 6.5 seconds.

Bilinear World Models enable real-time control. Fast planning is particularly important in dynamic environments, where the controller must repeatedly replan as the goal changes. We therefore introduce moving-goal variants of the four environments, denoted M-TwoRoom, M-PushT, M-Reacher, and M-Cube, in which the target follows a random walk; implementation details are provided in Appendix B.3. Because CEM cannot meet the prescribed real-time budget, we use the more efficient iCEM controller (Pinneri et al., 2021) for the neural-dynamics baselines, while Bilinear World Models retain the same planner used in the remaining experiments. As shown in Figure 5, our method achieves the highest success rate in all four environments, with 68%, 77%, 71%, and 93% success, compared with 46%, 47%, 45%, and 63% for the strongest baseline. Averaged across tasks, this corresponds to an improvement of approximately 27 percentage points.

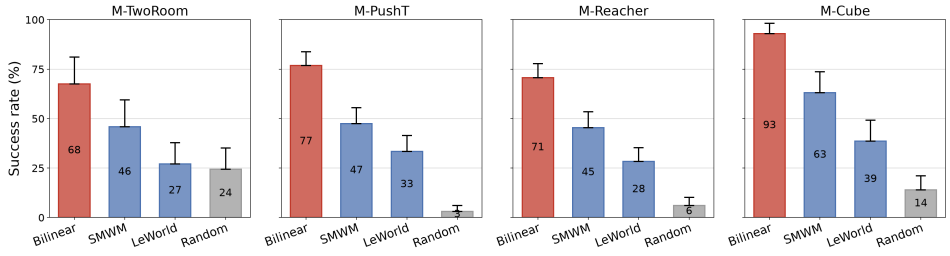


Figure 5: **Control performance on moving-goal tasks.** Success rate on M-TwoRoom, M-PushT, M-Reacher, and M-Cube under a 20 Hz control loop. The target follows a random walk, requiring repeated replanning under controller latency. SMWM and LeWorld use iCEM (Pinneri et al., 2021), while Bilinear World Models use the same Gauss–Newton controller as in the other experiments. The Random baseline applies uniformly sampled actions.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the implementation of experimental methods and the writing and presentation of proofs. We additionally used generative AI tools to suggest experimental parameters, create and edit software code, identify relevant literature, and edit portions of the manuscript to improve clarity and readability. All AI-assisted content was reviewed by the authors. In particular, AI-generated or AI-modified code was manually inspected and tested for correctness, and all AI-assisted mathematical arguments were independently verified by the authors. The authors take full responsibility for the final content of the manuscript, including any text, claims, code, or other artifacts produced with the assistance of generative AI.

We did not use generative AI tools to develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine research hypotheses, design or provide feedback on the research methodology or experiments, generate synthetic datasets, assist with translation, clean or reformat datasets, perform qualitative or thematic data analysis, or interpret experimental results.

ETHICS STATEMENT

Given the nature of the work presented in this manuscript, we do not identify any additional ethical considerations that require discussion.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide a ready-to-run script for reproducing the main experiments reported in the manuscript. The script launches a Docker container to provide a controlled and isolated computational environment. It has been designed to be simple and self-contained, with comments included to improve readability and facilitate its use. A detailed description of the hardware and software environment used to obtain the experimental results is provided in the Experimental section.

For the theoretical results, complete proofs of all formal statements, together with additional explanations where appropriate, are provided in the Appendix.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture, 2023. URL https://arxiv.org/abs/2301.08243.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025. URL https://arxiv.org/abs/2506.09985.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics, 2025. URL https://arxiv.org/abs/2511.08544.

Bassam Bamieh. A short introduction to the koopman representation of dynamical systems, 2022. URL https://arxiv.org/abs/2205.08048.

Daniel Bruder, Xun Fu, and Ram Vasudevan. Advantages of bilinear koopman realizations for the modeling and control of systems with unknown dynamics, 2020. URL https://arxiv.org/abs/2010.09961.

A. Buccini, P. Díaz de Alba, F. Pes, and L. Reichel. An efficient implementation of the gauss–newton method via generalized krylov subspaces. *J. Sci. Comput.*, 97(2), October 2023. ISSN 0885-7474. doi: 10.1007/s10915-023-02360-w. URL https://doi.org/10.1007/s10915-023-02360-w.

Delong Chen, Mustafa Shukor, Theo Moutakanni, Willy Chung, Jade Yu, Tejaswi Kasarla, Yejin Bang, Allen Bolourchi, Yann LeCun, and Pascale Fung. Vl-jepa: Joint embedding predictive architecture for vision-language, 2026. URL https://arxiv.org/abs/2512.10942.

David L. Elliott. *Bilinear Systems*. Wiley Interscience Imprint,, 2007. doi: 10.1002/047134608X.W1003.pub2.

David Ha and Jürgen Schmidhuber. World models. 2018. doi: 10.5281/ZENODO.1207631. URL https://zenodo.org/record/1207631.

Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *CoRR*, abs/1811.04551, 2018. URL http://arxiv.org/abs/1811.04551.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2020. URL https://arxiv.org/abs/1912.01603.

Petr Ivashkov, Randall Balestriero, and Bernhard Schölkopf. Sensorimotor world models: Perception for action via inverse dynamics, 2026. URL https://arxiv.org/abs/2606.20104.

B. O. Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931. doi: 10.1073/pnas.17.5.315. URL https://www.pnas.org/doi/abs/10.1073/pnas.17.5.315.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1), November 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-07210-0. URL http://dx.doi.org/10.1038/s41467-018-07210-0.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworld-model: Stable end-to-end joint-embedding predictive architecture from pixels, 2026. URL https://arxiv.org/abs/2603.19312.

Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41:309–325, 2005. URL https://api.semanticscholar.org/CorpusID:37635186.

Ronald R. Mohler. *Bilinear Control Processes: with Applications to Engineering, Ecology, and Medicine*. Mathematics in Science and Engineering. Academic Press, 1974. ISBN 9780080956152. URL https://books.google.com/books?id=uOV30prut28C.

Samuel Otto, Sebastian Peitz, and Clarence Rowley. Learning bilinear models of actuated koopman generators from partially observed trajectories. *SIAM Journal on Applied Dynamical Systems*, 23 (1):885–923, 2024.

Samuel E. Otto and Clarence W. Rowley. Koopman operators for estimation and control of dynamical systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(Volume 4, 2021):59–87, 2021. ISSN 2573-5144. doi: https://doi.org/10.1146/annurev-control-071020-010108. URL https://www.annualreviews.org/content/journals/10.1146/annurev-control-071020-010108.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In Jens Kober, Fabio Ramos, and Claire Tomlin (eds.), *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1049–1065. PMLR, 16–18 Nov 2021. URL https://proceedings.mlr.press/v155/pinneri21a.html.

Reuven Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(96)00385-2. URL https://www.sciencedirect.com/science/article/pii/S0377221796003852.

Bas van der Heijden, Laura Ferranti, Jens Kober, and Robert Babuška. Deepkoco: Efficient latent planning with a task-relevant koopman representation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 183–189, 2021. doi: 10.1109/IROS51168.2021.9636408.

Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

APPENDIX

A PROOF OF THEOREM 1

Proof. By Assumption 1, the dynamics are control-affine. A standard result for Koopman representations of control-affine systems establishes that there exists a possibly infinite-dimensional collection of state-dependent observables whose evolution is bilinear in the observables and the action (Bruder et al., 2020, Corollary II.1). We choose this collection to include the original state coordinates and denote the resulting embedding by $z = E(s)$. The lifted dynamics can then be written as

$$\dot{z} = Az + (B + Cz)\, a = Az + M(z)a. \tag{13}$$

Since the state coordinates are included among the observables, E is injective and there exists a linear decoding operator D such that

$$DE(s) = s. \tag{14}$$

Applying the recovery operator D to the lifted dynamics and using $Dz = DE(s) = s$ gives

$$\dot{s} = DAz + DM(z)a. \tag{15}$$

Since this realization reproduces the original dynamics, comparison with Assumption 1 yields

$$DM(E(s)) = [f_1(s) \quad \cdots \quad f_m(s)]\,. \tag{16}$$

Now suppose that, for some $v \in \mathbb{R}^m$,

$$M(E(s))v = 0. \tag{17}$$

Applying D to both sides gives

$$[f_1(s) \quad \cdots \quad f_m(s)]\, v = 0. \tag{18}$$

By Assumption 2, the matrix on the left has rank m, and hence $v = 0$. Therefore,

$$\ker M(E(s)) = \{0\}, \tag{19}$$

so $M(E(s))$ has full column rank for every $s \in \mathcal{S}$. Consequently, its Gram operator is positive definite,

$$M(E(s))^\top M(E(s)) \succ 0, \qquad \forall s \in \mathcal{S}, \tag{20}$$

which completes the proof. □

B EXPERIMENTAL DETAILS

This section presents extended experimental evaluations of four planning tasks introduced in Maes et al. (2026). These tasks are intended to be solved by an agent using only observations of the world. In TwoRooms (2D), an agent is placed in one of two spaces divided by a wall and connected through a small opening and required to reach a desired target position, which may involve crossing the opening. This task is intended to model physical constraints of the environment, since the walls cannot be crossed by the agent. The PushT (2D) problem requires an agent to make contact with a T-shaped object and rotate, i.e., push it, to reach a desired position. This task is intended to model

situations that are affected by external movable objects that require interaction. In Reacher (2D), a fixed arm with one joint is allowed to move within a circle to reach a target object, thus modeling rotations and multiple degrees of freedom. In Cube (3D), a gripper arm is required to catch a cube and move it to a target location, modeling complex actuation in spatial conditions.

Hardware. All experiments in the manuscript were performed on a single workstation with one NVIDIA GeForce RTX 3090 (24 GB, driver 550.144.03), an AMD Ryzen Threadripper 3960X (24 cores, 48 threads) and 125 GB of RAM. Every world model was trained on that single GPU in `bf16`. The software stack was PyTorch 2.7.1 (CUDA 12.6, cuDNN 9.5.1), Lightning 2.6.6, MuJoCo 3.13.0, `dm_control` 1.0.46 and Gymnasium 1.3.0.

B.1 Solving the constrained optimization problem P

The problem P is formulated using the normalized action map $M(z) = Q(z)R$, where the state-dependent factor $Q(z)$ is obtained by applying a Cholesky–QR decomposition to the bilinear parameterization $\widetilde{M}(z) = B + Cz$. This normalization changes the exact functional form of the dynamics, which are therefore no longer strictly bilinear. However, it preserves the state-dependent action subspace generated by the original bilinear map. In particular, whenever $\widetilde{M}(z)$ has full column rank and R is nonsingular,

$$\operatorname{col}(M(z)) = \operatorname{col}(Q(z)) = \operatorname{col}\Big(\widetilde{M}(z)\Big). \tag{21}$$

Thus, the bilinear parameterization $\widetilde{M}(z) = B + Cz$ determines the state-dependent action directions, while the normalization provides a well-conditioned basis for the same subspace.

Specifically, the normalized action map is given by

$$M(z) = Q(z)R, \qquad Q(z) = \operatorname{CholQR}\Big(\widetilde{M}(z)\Big) = \operatorname{CholQR}(B + Cz)\,, \tag{22}$$

where $Q(z) \in \mathbb{R}^{d\times m}$ has orthonormal columns and $R \in \mathbb{R}^{m\times m}$ is a learned state-independent matrix. Consequently,

$$\begin{aligned} M(z)^\top M(z) &= R^\top Q(z)^\top Q(z) R \\ &= R^\top R. \end{aligned} \tag{23}$$

Hence, the full-column-rank condition of the normalized action map is independent of the latent state. In particular,

$$M(z)^\top M(z) \succ 0 \quad \Longleftrightarrow \quad \sigma_{\min}(R) > 0. \tag{24}$$

The matrix $Q(z)$ therefore determines the orientation of the state-dependent action directions, while their scaling and conditioning are controlled by the constant matrix R.

Although Problem equation P expresses action recoverability through the hard constraint $R^\top R \succeq \varepsilon I$, enforcing this constraint exactly during stochastic optimization is inconvenient in practice. We therefore solve a regularized surrogate in which action recoverability is encouraged directly from the observed transitions,

$$\begin{aligned} \underset{\phi_\theta, A, B, C, R}{\text{minimize}} \quad & \frac{1}{N}\sum_t \Big\| \phi_\theta(s_{t+1}) - A\phi_\theta(s_t) - \operatorname{CholQR}(B + C\phi_\theta(s_t))\, R\, a_t \Big\|^2 \\ & + \lambda \frac{1}{N}\sum_t \left\| a_t - \Big[\operatorname{CholQR}(B + C\phi_\theta(s_t))\, R\Big]^\dagger \Big(\phi_\theta(s_{t+1}) - A\phi_\theta(s_t)\Big) \right\|^2 . \end{aligned} \tag{$\hat{\text{P}}$}$$

The first term fits the normalized latent dynamics to the observed transitions. The second directly encourages the action that generated each transition to be recoverable from the corresponding latent displacement. This discourages degenerate solutions in which distinct actions produce indistinguishable latent transitions and provides a practical surrogate for the full-rank requirement in Problem equation P. Although this surrogate does not explicitly enforce $R^\top R \succeq \varepsilon I$, we empirically verify in Appendix C.1 that $\sigma_{\min}(R)$ remains bounded away from zero throughout training across all environments, and hence that the learned action map remains full column rank. Since

the Cholesky–QR operation is differentiable, gradients can be propagated through the normalization and the complete model can be trained end-to-end using standard gradient-based optimization. For all Bilinear World Model experiments, we use $\lambda = 30$.

B.2 CONTROLLERS

We provide implementation details for the controllers used in the experiments of the main text. Specifically, we describe the Cross-Entropy Method (CEM) used for the standard planning experiments, the iterative CEM (iCEM) variant used for real-time control, and our implementation of the Gauss–Newton (GN).

B.2.1 CROSS-ENTROPY METHOD

The Cross-Entropy Method (CEM) (Rubinstein, 1997) is the sampling-based planner used in the original evaluations of LeWM and Sensorimotor. Given a planning horizon H, CEM maintains a Gaussian distribution over action sequences

$$\mathbf{a} = (a_0, \dots, a_{H-1}) \sim \mathcal{N}(\mu, \Sigma). \tag{25}$$

At each iteration, N action sequences are sampled from this distribution and rolled out through the learned world model. Each sequence is evaluated according to the planning objective

$$\mathcal{L}(\mathbf{a}) = \frac{1}{2} \left\| z_H(\mathbf{a}) - z^\star \right\|^2. \tag{26}$$

The K sequences with lowest cost are retained as the elite set $\mathcal{E}$. The sampling distribution is then updated according to the empirical mean and variance of these elites,

$$\mu \leftarrow \frac{1}{K} \sum_{\mathbf{a} \in \mathcal{E}} \mathbf{a}, \tag{27}$$

$$\Sigma \leftarrow \frac{1}{K} \sum_{\mathbf{a} \in \mathcal{E}} (\mathbf{a} - \mu)(\mathbf{a} - \mu)^\top. \tag{28}$$

This procedure is repeated for a fixed number of iterations, and the lowest-cost sequence found during the optimization is selected for execution.

CEM treats the learned world model as a black box and does not use the Jacobian of the predicted endpoint with respect to the actions. Its computational cost is therefore dominated by the large number of candidate trajectories that must be rolled out and evaluated during the search.

B.2.2 IMPROVED CROSS-ENTROPY METHOD

We additionally evaluate iCEM, a more sample-efficient variant of CEM. The overall optimization procedure remains unchanged, but the sampling strategy is modified to concentrate computation on more promising and temporally coherent action sequences.

In particular, instead of sampling independent white Gaussian perturbations, our implementation uses temporally correlated colored noise. Starting from white noise ξ, its Fourier coefficients are scaled according to

$$\widehat{\xi}_k^{\mathrm{col}} \propto f_k^{-\beta/2} \widehat{\xi}_k, \tag{29}$$

followed by normalization back to the desired variance. We use $\beta = 2$, which places greater weight on low-frequency components and therefore produces smoother action sequences across the planning horizon. Setting $\beta = 0$ recovers independent white-noise sampling as in standard CEM.

The implementation also reuses elite trajectories between consecutive optimization iterations. Specifically, the best five sequences from the previous iteration are inserted directly into the next candidate population, together with the current distribution mean. The remaining candidates are newly sampled from the updated distribution. This allows promising trajectories to persist across iterations without requiring them to be rediscovered by sampling.

Our implementation does not use population decay, and the number of sampled trajectories therefore remains fixed across optimization iterations. It also does not shift the optimized action sequence between successive replanning steps. Unless an initial action sequence is explicitly provided, each planning problem is initialized from a zero-mean action distribution.

B.2.3 GAUSS–NEWTON PLANNER

We implement the planner of Section 5 in a receding-horizon fashion. At each replan, the current observation and goal are encoded into z_0 and $z^\star$, and an action sequence $\mathbf{a} = (a_0, \dots, a_{H-1})$ is optimized according to equation 10. Each planning step corresponds to the same action block used during training, and the optimized sequence is executed before replanning from the resulting observation.

Reduced action space. In practice, each model action contains several low-level environment actions. Optimizing all of these coordinates independently produces poorly conditioned Gauss–Newton systems, since variations within an action block have little distinguishable effect on the learned dynamics. We therefore restrict the update to a lower-dimensional coherent subspace in which the low-level actions within each block vary together. This reduces the number of optimized variables from 50 to 10 for the 2D tasks with $H = 5$, and from 125 to at most 15 for Cube. The corresponding lifting from the reduced variables to the complete action sequence is denoted by E.

Endpoint sensitivities. Although Section 5 gives an explicit expression for the endpoint Jacobian, in the reported experiments we estimate its restriction to the coherent subspace using directional finite differences. At each iteration, the nominal action sequence and all perturbed sequences are evaluated jointly in a single batched rollout,

$$\widehat{\mathcal{J}E}_{:,j} = \frac{z_H(\mathbf{a} + \varepsilon E_{:,j}) - z_H(\mathbf{a})}{\varepsilon}, \qquad \varepsilon = 10^{-3}. \tag{30}$$

This directly captures the sensitivity of the complete nonlinear predictor, including the state dependence of the normalized action directions and the learned residual dynamics, without explicitly differentiating through the full rollout.

Damped update and line search. Let $C = \widehat{\mathcal{J}E}$ and $e = z_H(\mathbf{a}) - z^\star$. The reduced Gauss–Newton direction is obtained from

$$\left[C^\top C + (c+\mu)I\right] \delta_c = -\left[C^\top e + cE^\top \mathbf{a}\right], \qquad \delta = E\delta_c, \tag{31}$$

with damping $\mu = 10^{-4}$. We then evaluate the step sizes $\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}\}$ and accept the first one that decreases the true nonlinear planning objective. All candidate step sizes are evaluated jointly in a single batched rollout.

Initialization. Since Gauss–Newton is a local method, we optionally initialize the action sequence from a small set of candidate initializations containing the current warm start, several gradient-aligned sequences, and random perturbations. All candidates are evaluated in parallel and the lowest-cost sequence is used to initialize the optimization. The number of candidates, planning horizon, reduced dimension, and iteration budget used for each task are reported in Table **??**.

Planning residual. The structured model is intentionally restricted during representation learning, but this restriction can leave systematic prediction errors in contact-rich environments. We therefore train an auxiliary residual model *a posteriori*, after the encoder and structured dynamics have been learned and fixed. Consequently, the residual does not influence the learned representation or the parameters A, B, C, R, and is used only to improve the transition model employed during planning.

During planning, we augment the structured transition $S(z, a) = Az + M(z)a$ as

$$f_{\text{plan}}(z, a) = S(z, a) + \big(I - P(z)P(z)^\top\big) r_\psi(z, a), \tag{32}$$

where $P(z)$ spans the coherent control directions introduced above. The projection prevents the residual from modifying these directions while allowing it to capture components of the transition that cannot be represented by the structured model alone.

The endpoint sensitivities used by the controller are computed through this complete planning model. Therefore, the directional finite differences described below automatically account for both the structured dynamics and the residual correction.

Task	d_a	H	K	n_b	K_c	iters	n_{cand}	ρ
PushT	2	5	50	5	10	40	16	0.5
Reacher	2	3	30	3	6	20	64	2.0
TwoRoom	2	5	50	3	6	40	0	—
Cube	5	5	125	3	15	20	0	—

Table 1: Gauss–Newton planner settings for the reported cells. H: planning horizon in action blocks; $m = k\,d_a$ with block size $k = 5$; $K = Hm$ raw unknowns, K_c coherent unknowns; n_b: time blocks refreshed per iteration. Shared across all tasks: $\lambda_g = 1$, $c = 0.01$, $\mu = 10^{-4}$, $\varepsilon = 10^{-3}$, line search $\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}\}$, $n_{\text{starts}} = 1$, tolerance 10^{-6}.

B.3 Moving goal implementation

Task construction. We construct moving-goal variants of the four benchmark environments by allowing the target state to follow a random walk throughout the episode. At each environment step, the goal state g_t is displaced by $\sigma v_{\max} u$, where u is a uniformly sampled direction, $v_{\max}$ is the characteristic maximum per-step motion of the corresponding system, and σ controls the relative goal speed. We estimate $v_{\max}$ as the 95th percentile of per-step motion in the expert data, obtaining 22.4 px for PushT, 0.169 rad for Reacher, and 0.028 m for Cube. Invalid displacements that leave the workspace or cross obstacles are resampled. Goal trajectories are seeded per task so that all methods are evaluated on identical moving targets. The resulting environments are denoted M-TwoRoom, M-PushT, M-Reacher, and M-Cube. The goal is re-rendered at every step, and the original environment-specific success criterion is used.

Real-time evaluation protocol. Planning latency is explicitly incorporated into the control loop. For each method and controller, we first measure the wall-clock latency τ of a single planning call on one environment. Given a control period Δ, this corresponds to a delay of $L = \lceil \tau/\Delta \rceil$ environment steps. A plan computed from the observation and goal at step t therefore becomes available at step $t + L$. During this interval, the agent continues executing the remainder of its previous plan, or applies a zero action if no action remains. Methods replan when the current plan is exhausted.

All experiments in Figure 5 use a 20 Hz control loop, corresponding to $\Delta = 50$ ms. The baselines use iCEM (Pinneri et al., 2021), since the execution time of standard CEM is incompatible with the real-time budget. Bilinear World Models use the same Gauss–Newton controller employed in the remaining experiments. For each baseline, we report the best-performing iCEM sample budget under this protocol.

Evaluation settings. Goal speeds are selected independently for each environment and expressed relative to the corresponding $v_{\max}$: 3.0 for TwoRoom, 0.1 for PushT, 0.05 for Reacher, and 0.25 for Cube. To avoid inflating success rates with tasks that require no meaningful control, we evaluate only episodes that are not solved by a policy that never acts.

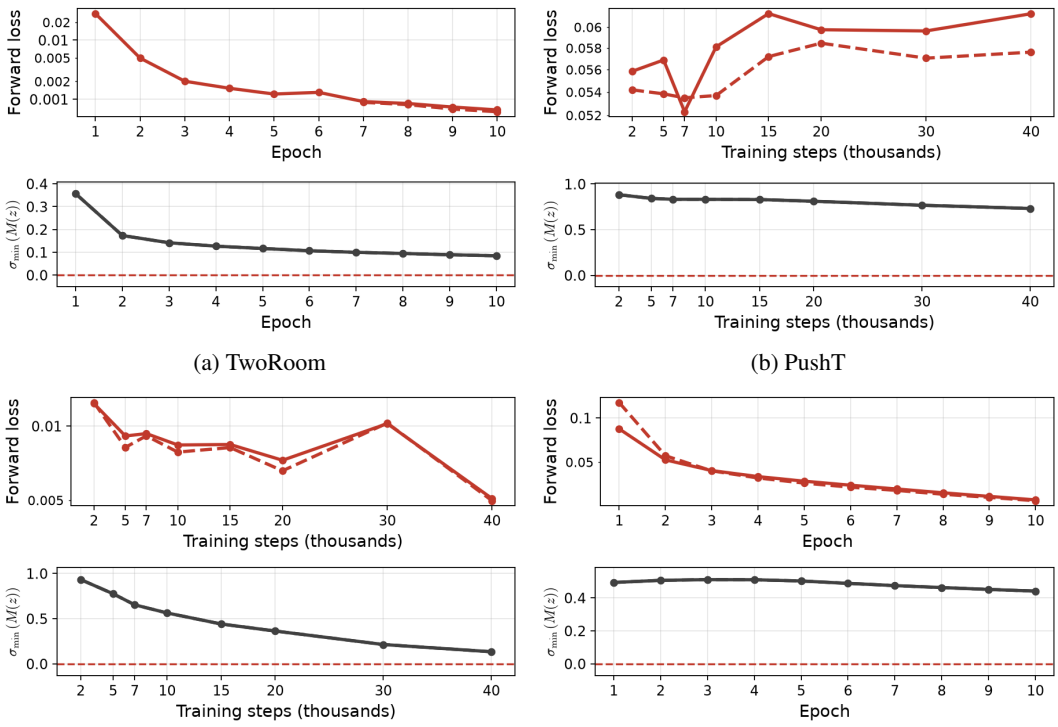
C Additional Results

This section provides additional empirical results complementing the main experiments. We first examine the optimization behavior of Bilinear World Models and verify that the learned action map remains nondegenerate throughout training. We then compare the performance of CEM, iCEM, and Gauss–Newton (GN) across the different world-model architectures. Finally, we provide additional visualizations of the geometry of the learned latent representations.

C.1 Training dynamics and action-map conditioning

Problem P requires the state-dependent action map $M(z)$ to have full column rank. As described in Appendix B.1, our implementation parameterizes the action map as

$$M(z) = Q(z)R, \tag{33}$$



(a) TwoRoom (b) PushT

(c) Reacher (d) Cube

Figure 6: **Training dynamics of Bilinear World Models.** The latent forward-prediction loss decreases throughout training, while the minimum singular value $\sigma_{\min}(R)$ remains bounded away from zero. Since $M(z) = Q(z)R$ with $Q(z)$ orthonormal, $\sigma_{\min}(R) = \sigma_{\min}(M(z))$ for every latent state z. The learned action map therefore remains full column rank throughout training.

where $Q(z)$ has orthonormal columns and R is a learned state-independent matrix. Therefore,

$$M(z)^\top M(z) = R^\top R, \tag{34}$$

and the singular values of $M(z)$ are identical to those of R for every latent state z. In particular,

$$\sigma_{\min}(M(z)) = \sigma_{\min}(R), \qquad \forall z. \tag{35}$$

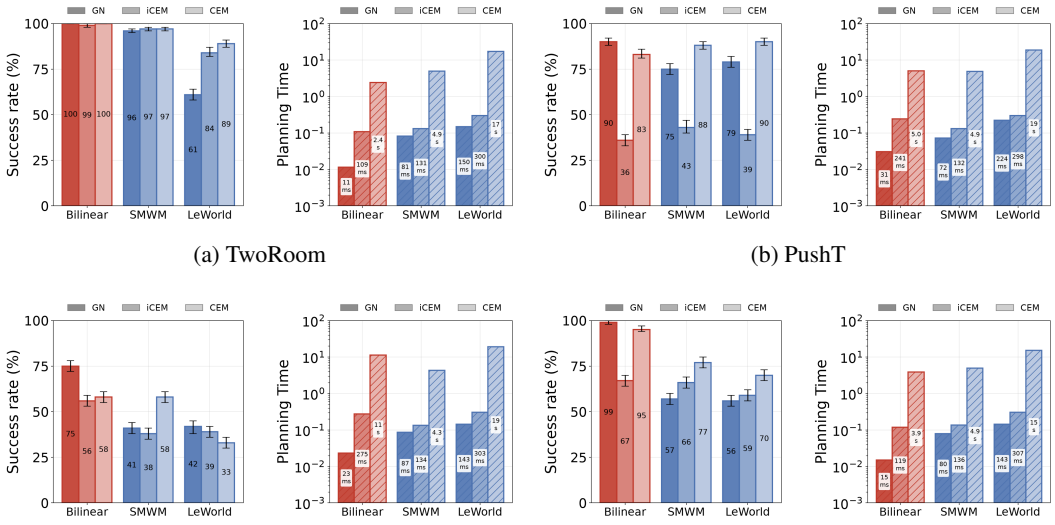
Consequently, no averaging or minimization over latent states is required to evaluate the conditioning of the learned action map.

Figure 6 reports the latent forward-prediction loss together with $\sigma_{\min}(R)$ throughout training. Across all four environments, the prediction loss decreases steadily while the minimum singular value remains bounded away from zero. Thus, the learned action map remains full column rank throughout training while simultaneously fitting the observed latent dynamics.

C.2 Controller comparison across world models

The main advantage of imposing structure on the latent dynamics is that this structure can be exploited directly during control. To separate the effect of the learned world model from that of the controller, we evaluate three planning methods—CEM, iCEM, and Gauss–Newton (GN)—on Bilinear World Models, SMWM (Ivashkov et al., 2026), and LeWorld (Maes et al., 2026). Figure 7 reports both success rate and planning time across the four environments.

GN benefits directly from the structured dynamics of our model. When paired with Bilinear World Models, it achieves strong control performance with substantially lower planning time. In contrast, its performance degrades when applied to SMWM and LeWorld, whose latent dynamics are represented by unconstrained nonlinear models and therefore do not provide the same structure to



(a) TwoRoom (b) PushT

(c) Reacher (d) Cube

Figure 7: **Controller performance across world models.** Success rate and planning time are reported for CEM, iCEM, and Gauss–Newton (GN) applied to Bilinear World Models, SMWM, and LeWorld. GN is particularly effective when paired with Bilinear World Models, where it can exploit the explicit structure of the latent dynamics. CEM remains applicable across all architectures but incurs substantially higher planning cost, while iCEM reduces computation at the expense of degraded performance on the more challenging tasks.

the optimizer. These results indicate that the computational advantage of GN is closely tied to the explicit form of the learned dynamics.

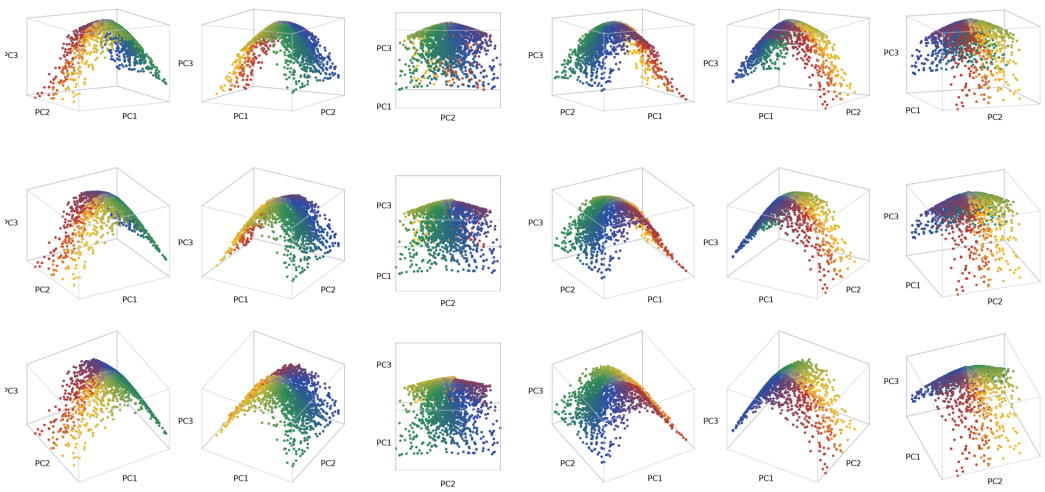
CEM provides a complementary comparison because it is agnostic to the structure of the world model and can therefore be applied uniformly to all three architectures. When used with Bilinear World Models, CEM achieves competitive success rates but incurs substantially greater planning cost than GN, since it does not exploit the available structure. The real-time variant iCEM reduces this computational cost, but its performance deteriorates on the more challenging environments, particularly Cube. For this reason, iCEM is primarily used in the real-time experiments of Section 6, where rapid replanning is required.

C.3 Geometry of the learned latent representations

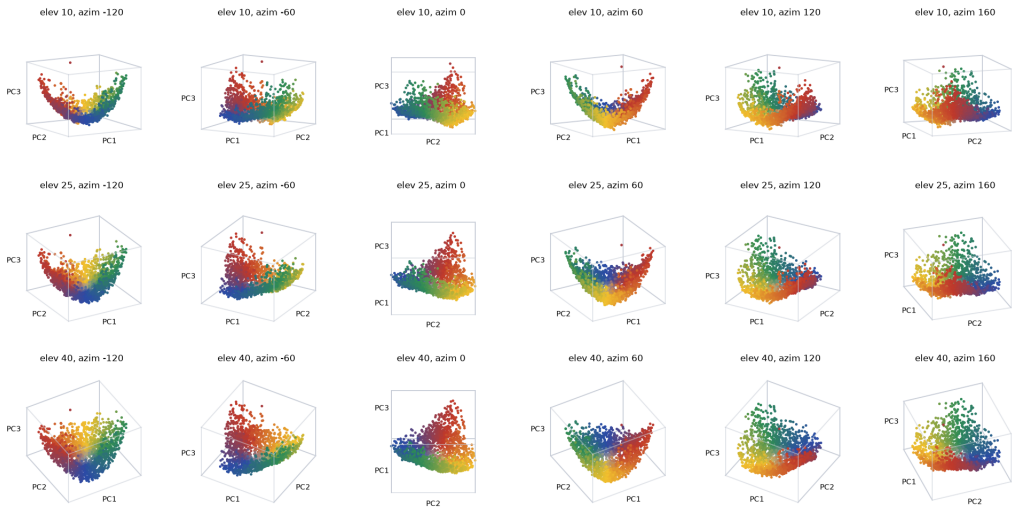
World models jointly learn representations of observations and models of their dynamics. Although the observations are high dimensional, the underlying physical systems evolve on substantially lower-dimensional state spaces. We therefore further examine the geometry of the learned latent representations using principal component analysis (PCA).

Figure 8 shows the latent states projected onto their first three principal components. Across the environments, the learned representations organize into smooth, low-dimensional structures rather than filling the ambient latent space. For TwoRoom, PushT, and Reacher, the dominant variation is largely captured by a two-dimensional surface, whereas Cube exhibits a richer geometry that is not fully represented by a three-dimensional projection.

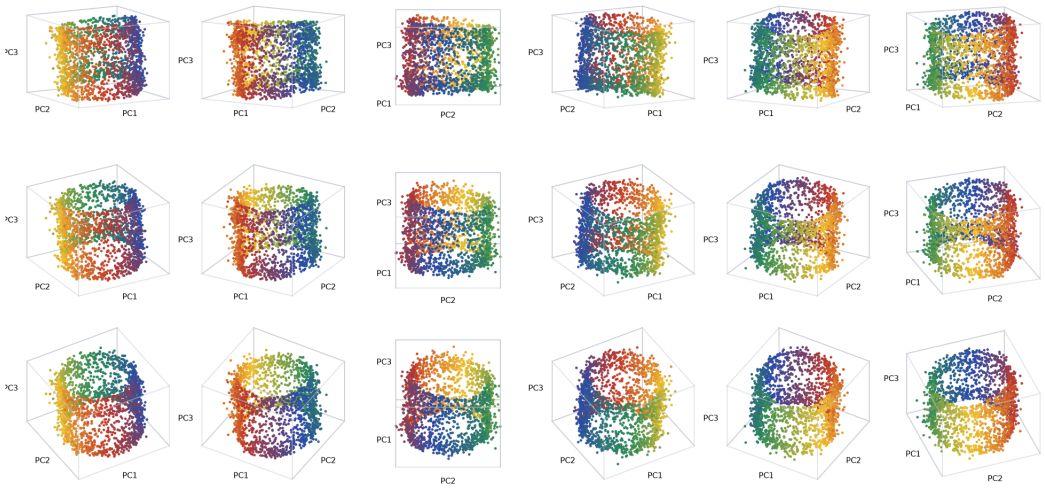
Interestingly, the apparent dimensionality of these structures is closely related to the dimensionality of the corresponding control spaces. The simpler environments have two-dimensional action spaces and exhibit approximately two-dimensional latent geometry, while Cube has a higher-dimensional action space and correspondingly more complex latent structure. While PCA alone does not establish the exact intrinsic dimension of the learned representation, these results suggest that the latent space organizes around a compact set of degrees of freedom that are relevant for control.



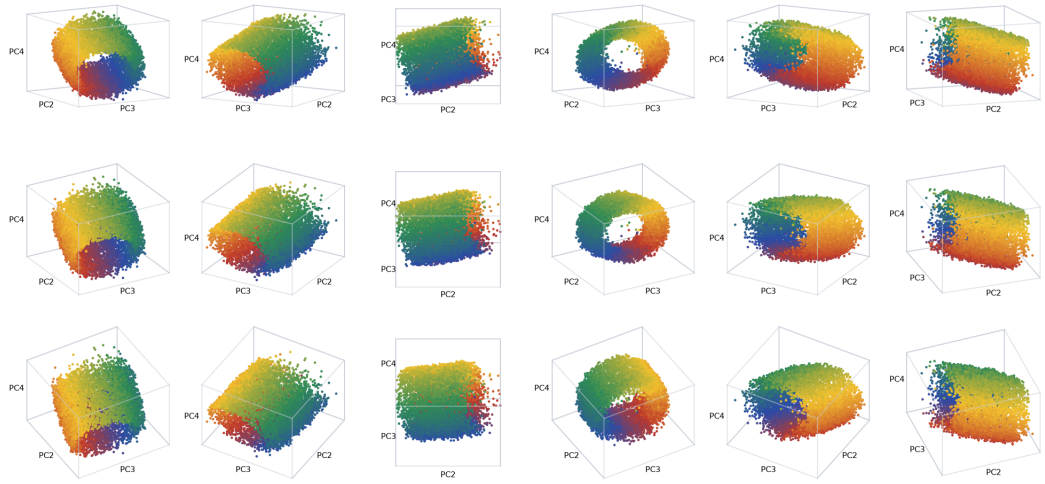
(a) TwoRoom



(b) PushT



(c) Reacher



(d) Cube

Figure 8: **PCA visualization of the learned latent representations.** Latent states are projected onto their first three principal components. TwoRoom, PushT, and Reacher exhibit approximately low-dimensional surface structure, while Cube displays a richer geometry that is not fully captured by a three-dimensional projection.