# FROM OBSERVATION PREDICTION TO CONTROL COORDINATES: LEARNING DIFFERENTIALLY FLAT LATENT REPRESENTATIONS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Latent dynamics models enable control of systems from high-dimensional observations. Existing methods primarily emphasize predictive accuracy, while control depends on the relationships among the learned representation, the underlying physical state, and the control input. Accurate observation-level prediction alone may be insufficient for effective control, particularly when observations do not uniquely determine the physical state. We address this gap by introducing differential flatness as a structural bias. For differentially flat systems, we establish a sufficient observation condition: observations need only distinguish states with distinct flat outputs, whose finite-order derivatives enable state and input recovery. The model combines a flat-jet encoder and structured latent dynamics, trained jointly without supervision on the true flat coordinates. This representation supports direct trajectory planning in learned flat coordinates and translation into control inputs. Experiments on an analytically tractable flat system and image-based differentially flat systems under incomplete observations show that our model learns coordinates with differential flatness properties, improves prediction and control accuracy over predictive latent baselines, particularly with limited data, and reduces computation for closed-loop control.

## 1 INTRODUCTION

In robotic and physical control problems, high-dimensional observations are often compressed into latent states intended to capture the system's controlled dynamics (Watter et al., 2015; Hafner et al., 2019). Most latent dynamics learning methods evaluate such representations through observation reconstruction and controlled forward prediction: given a current latent state and an applied control input, the model should predict the subsequent latent state or observation (Hafner et al., 2025a). These objectives are sufficient for learning a predictive model, but control raises a complementary inverse question: given a desired local evolution in latent space, can the corresponding control input be determined, and if so, what finite information about that evolution is sufficient to determine it?

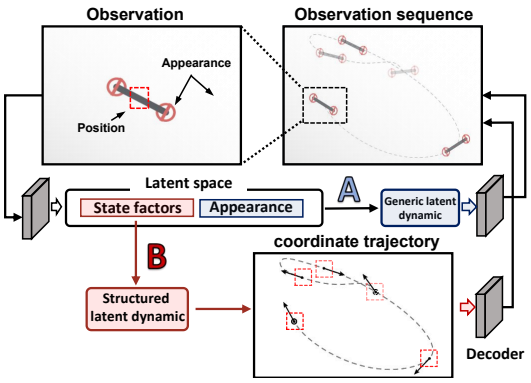



Figure 1: The latent space conceptually contains state-related factors and appearance variations. Predictive latent models (A) capture observation evolution but may leave the underlying state dynamics implicit, while our approach (B) introduces structures latent dynamics to explicitly relate latent evolution to the underlying physical state and control input.

Existing latent control methods typically address this inverse problem by combining learned forward dynamics with model predictive control (MPC). Locally linear (Banijamali et al., 2018; Watter et al., 2015) and Koopman-based approaches (Lusch et al., 2018; Naiman et al., 2024), for example, impose controller-compatible latent dynamics to facilitate control optimization.



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.

Their representations are shaped primarily by prediction accuracy (Hafner et al., 2019), transition consistency (Watter et al., 2015), and controller-compatible geometry such as linearity or low curvature (Levine et al., 2020). These criteria, however, remain primarily tied to observation-level prediction, which may not fully capture the physical state dynamics when observations are incomplete. For example, a single image may reveal an object's position but not its velocity (Efroni et al., 2022). A latent representation may therefore capture observation evolution without preserving the underlying state dynamics needed for control. MPC based on such a representation evaluates inputs through predicted observation-level outcomes, so inputs that appear effective in the learned observation prediction model may fail to produce the intended physical motion. Although standard predictive sequence models, such as Hafner et al. (2025b); Xie et al. (2023), use temporal context to reduce observation ambiguity, temporal conditioning and predictive training alone do not guarantee that a finite latent evolution determines the underlying physical state and control input.

Differential flatness (FLIESS et al., 1995) provides a geometric means of making these relationships explicit. Differential flatness holds for standard models of fully actuated manipulators and several underactuated or nonholonomic systems, including quadrotors Mellinger & Kumar (2011), cranes, and car-like robots Zhu et al. (2015). For these systems, it provides a finite differential parameterization of system behavior: the state and control input at each time can be obtained algebraically from flat outputs and finitely many of their derivatives. Thus, flat outputs and their finite-order derivatives can fully characterize the physical state and control input for these systems Beaver & Malikopoulos (2024). This motivates us to learn flat coordinates from data and use their derivative structure to construct a latent space. In this way, observation sequences are transformed into a structured latent representation that supports reconstruction and controlled prediction, while providing explicit relations to the underlying Our contributions are threefold.

(1) We formalize a representation gap under incomplete observations: accurate observation-level latent dynamics do not ensure that a finite latent evolution determines the underlying state and control input. By introducing the differential structure of flat outputs into latent dynamics learning, we analytically establish that this determination is guaranteed under a relaxed observation condition: observations need only distinguish states with distinct flat outputs.

(2) We develop a model based on a variational autoencoder (VAE) that learns flat coordinates and structured latent dynamics from observations, enabling prediction and control without supervision on the true flat coordinates.

(3) We empirically show that, under incomplete observations, the learned representation exhibits a flatness-equivalent structure, outperforms predictive latent baselines in prediction and control inference, and supports trajectory planning and closed-loop control.

## 2 RELATED WORK

We review two main approaches to connecting latent dynamics learning with control: **Structured Latent Dynamics** and **Task-oriented Latent Model**. Additional related work on **Differential Flatness** and **Variational Autoencoders** is discussed in the appendix.

Table 1: Structural biases and control interfaces of latent dynamics models.

| Method | Latent structural bias | Control mechanism | Control computation | Latent interpretation |
|---|---|---|---|---|
| E2C / PCC | Local linearity | iLQG / iLQR | Local optimization | Predictive state |
| Deep Koopman | Linear evolution | Linear MPC / LQR | Linear optimization | Lifted state |
| Deep Lagrangian | Lagrangian mechanics | Energy control | Energy shaping | Mechanical state |
| Deep FBL | Linearizing maps | Linear feedback | Input transformation | Linearizing state |
| **Ours** | **Flat-output jets** | **Flat-space planning** | **Jet-based recovery** | **Trajectory coordinate** |

**Structured Latent Dynamics for Prediction and Control.** Learning controllable embedding (LCE) methods map high-dimensional observations into low-dimensional representations whose dynamics are structured for locally linear control. Representative methods include E2C (Watter et al., 2015), RCE (Banijamali et al., 2018), PCC (Levine et al., 2020), and PC3 (Shu et al., 2020), which progressively improve probabilistic modeling, predictive consistency, and latent regularization for locally linear control. Koopman-based approaches, including Deep Koopman models (Cheng et al., 2026; Miao et al., 2026), DKRCI (Laferrière et al., 2021), KoVAE (Naiman et al., 2024),and DeepKoCo (van der Heijden et al., 2021), learn lifted representations with linear or bilinear controlled

dynamics to support prediction and model-based control. Physics-constrained models embed known dynamics in latent representations (Yang et al., 2022; Stölzle & Della Santina, 2024), while Lagrangian models learn generalized coordinates and energy structure from observations to enable energy-based control (Zhong & Leonard, 2020). Data-driven feedback-linearization (FBL) methods further seek control-equivalent representations of unknown nonlinear systems (Goswami et al., 2023; Gadginmath et al., 2024). Collectively, these developments progressively transform latent variables from generic predictive representations into structured coordinates that carry physical and control meaning and directly support model-based control.

**Task-Oriented Latent Model Learning.** Another line of work shapes latent dynamics through task objectives Wang et al. (2024). TD-MPC Hansen et al. (2022) and TD-MPC2 Hansen et al. (2024) jointly learn latent dynamics, reward predictions, and value functions for model predictive control without reconstructing observations. These methods emphasize representations useful for task-level decision-making.

Both lines of work make latent models useful for control, but leave the relationship between latent evolution, the physical state, and the control input largely implicit. Our work primarily extends the first line of research by introducing differential flatness to construct a structured latent space that improves prediction and control while providing theoretical guarantees on its relationship to the underlying physical state.

## 3 Problem Formulation

This section formalizes latent dynamics learning from high-dimensional, state-incomplete observations. We contrast flatness-structured representation with generic latent representations and propose a structured latent-dynamics learning objective motivated by its theoretical advantages.

Consider a partially observed controlled system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \boldsymbol{o}(t) = \boldsymbol{h}(\boldsymbol{x}(t)), \tag{1}$$

where $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ is state, $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^m$ is control input, and $\boldsymbol{o} \in \mathcal{O} \subseteq \mathbb{R}^p$ is observation, typically with $p \gg n$. Given observation–control trajectories sampled from this system at a fixed interval $\Delta t$, with $\boldsymbol{o}_k = \boldsymbol{o}(k\Delta t)$ and $\boldsymbol{u}_k = \boldsymbol{u}(k\Delta t)$, the resulting dataset is $\mathcal{D} = \{(\boldsymbol{o}_{0:K}, \boldsymbol{u}_{0:K-1})\}_{i=1}^{N}$, the learning problem is to construct an encoder $\boldsymbol{E}_\theta$ that maps a finite observation history to a latent state $\boldsymbol{z}$, together with a controlled latent dynamics model $\boldsymbol{F}_\theta$. The learned latent states and dynamics are then used for prediction, planning, and control of the underlying physical system:

$$\boldsymbol{z}_k = \boldsymbol{E}_\theta(\boldsymbol{o}_{k-H:k}), \quad \boldsymbol{z}_{k+1} = \boldsymbol{F}_\theta(\boldsymbol{z}_k, \boldsymbol{u}_k), \tag{2}$$

Here, $H = 0$ corresponds to a single observation, while $H > 0$ allows a finite observation history. We focus on the case where the observation map $\boldsymbol{h}$ is non-injective: there exist two distinct states $\boldsymbol{x}_a \neq \boldsymbol{x}_b$ such that $\boldsymbol{h}(\boldsymbol{x}_a) = \boldsymbol{h}(\boldsymbol{x}_b)$. Thus, despite its high dimensionality, a single observation does not uniquely determine the underlying state. Under this observation setting, we next contrast generic latent dynamics learning with our flatness-structured formulation.

**Generic Latent Dynamics.** Given only $\boldsymbol{o}_k = \boldsymbol{h}(\boldsymbol{x}_k)$, generic latent dynamics methods learn an encoder–decoder pair $\boldsymbol{z}_k = \boldsymbol{E}_\theta(\boldsymbol{o}), \widehat{\boldsymbol{o}}_k = \boldsymbol{D}_\theta(\boldsymbol{z}_k)$. Even with exact reconstruction, $\boldsymbol{D}_\theta(\boldsymbol{E}_\theta(\boldsymbol{o}_k)) = \boldsymbol{o}_k$, the learned correspondence holds only between the observation and the latent representation. When $\boldsymbol{h}$ is non-injective, the resulting information chain is

$$\boldsymbol{x}(t) \xrightarrow{\boldsymbol{h}} \boldsymbol{o}(t) \xleftrightarrow{\boldsymbol{E}_\theta, \boldsymbol{D}_\theta} \boldsymbol{z}(t), \tag{3}$$

rather than $\boldsymbol{x} \leftrightarrow \boldsymbol{z}$. Consequently, a state-recovery mapping $\boldsymbol{x} = R_x(\boldsymbol{z})$ does not generally exist. The learned dynamics therefore reproduce observation-space evolution, but their rollouts need not correspond to unique state–input trajectories. If the missing state information affects future evolution or control, observation-based MPC may produce inaccurate control despite accurate predictions. Using observation histories may reduce this ambiguity, but prediction alone does not guarantee a correspondence between latent and physical states. We next show how differential flatness provides such a guarantee.

**Flatness-Structured Latent Dynamics.** In contrast, differential flatness allows the latent model to relax the requirement of complete state information in each observation. We consider systems admitting a state-dependent flat output $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x})$ that parameterizes the state and input at consecutive

differential orders. Specifically, for the known differential order $r \geq 1$, there exist smooth maps $\boldsymbol{\Psi}_x$ and $\boldsymbol{\Psi}_u$ such that

$$\boldsymbol{x} = \boldsymbol{\Psi}_x(J^{r-1}\boldsymbol{y}), \quad \boldsymbol{u} = \boldsymbol{\Psi}_u(J^r\boldsymbol{y}) \tag{4}$$

Here, $J^q\boldsymbol{y} := \mathrm{col}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ldots, \boldsymbol{y}^{(q)}\right)$ denotes the $q$-th order jet of $\boldsymbol{y}$. Thus, the state is determined by the flat jet $\boldsymbol{\xi} = J^{r-1}\boldsymbol{y}$, while input recovery additionally requires the highest-order derivative $\boldsymbol{\nu} = \boldsymbol{y}^{(r)}$ and $\boldsymbol{u} = \boldsymbol{\Psi}_u(\boldsymbol{\xi}, \boldsymbol{\nu})$. This consecutive-order structure is common in a broad class of differentially flat underactuated mechanical systems. Systems with larger state–input order gaps can be brought into this form through dynamic extension, as discussed in Appendix A.2. We next establish a sufficient observation condition under which this structure can be embedded into latent dynamics learning and derive the resulting guarantees for state–input recovery and control.

**Definition 1** (**Observation Sufficiency for Flat Outputs**). *An observation $\boldsymbol{o} = \boldsymbol{h}(\boldsymbol{x})$ is flat-output sufficient if the flat output can be uniquely identified from the observation; namely, there exists an observation map $\boldsymbol{E}_y$ such that $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{E}_y(\boldsymbol{o}) = \boldsymbol{E}_y(\boldsymbol{h}(\boldsymbol{x}))$.*

Thus, the observation only needs to determine the flat output, rather than the complete physical state, which is practically relevant because flat outputs in many mechanical systems depend on configurations that can often be identified from observations, such as images or direct measurements.

**Proposition 1** (**Recoverability of the Flat-Jet Representation**). *Given a sufficiently smooth local observation trajectory satisfying Definition 1, Suppose the learned map $\boldsymbol{E}_y$ realizes a state-dependent flat output $\widehat{\boldsymbol{y}}(t) = \boldsymbol{E}_y(\boldsymbol{o}(t)) = \boldsymbol{\phi}(\boldsymbol{x}(t))$ on a regular domain, then the resulting latent state $\boldsymbol{\xi}(t) := J^{r-1}\widehat{\boldsymbol{y}}(t)$ and its corresponding latent dynamics input $\boldsymbol{\nu}(t) := \widehat{\boldsymbol{y}}^{(r)}(t)$ determine the physical state and input.*

Therefore, when observations are incomplete but satisfy Definition 1, using the flat output and its derivatives as the latent representation $\boldsymbol{z}_f(t) := (\boldsymbol{\xi}(t), \boldsymbol{\nu}(t))$ guarantees the following correspondence with the physical state and input. The two chains Eq. (3) and (5) differ in where the learned correspondence terminates.

$$\boldsymbol{x}(t) \xrightarrow{\boldsymbol{h}} \boldsymbol{o}(t) \xrightarrow{\boldsymbol{E}_y} \widehat{\boldsymbol{y}}(t) \longrightarrow \boldsymbol{z}_f(t) \xrightarrow{\boldsymbol{\Psi}_x, \boldsymbol{\Psi}_u} (\boldsymbol{x}(t), \boldsymbol{u}(t)). \tag{5}$$

**Corollary 1** (**Latent Control Reparameterization**). *Suppose the observation condition in Definition 1 holds and the learned coordinates exactly realize the flatness parameterization of Proposition 1 in a domain, the original control optimization problem admits a locally **equivalent** formulation in the learned latent space.*

Overall, the above theory establishes that, once the flat output is identifiable from observations, differential flatness enables the construction of structured latent dynamics, providing a theoretical basis for solving control problems in latent space and applying the resulting solutions to the original system. Detailed proofs of Proposition 1 and Corollary 1 are provided in Appendix A.3 and A.4. The above theoretical results establish structural sufficiency rather than learnability, thereby motivating a learning objective that biases the representation toward such a realization without flat-coordinate supervision while its identification is evaluated empirically.

**Flatness-Structured Latent Learning.** To explicitly realizes Eq. (5), we briefly formulate the corresponding learning objective. For a known derivative-chain order $r$, the encoder learns the flat output $\boldsymbol{y} = \boldsymbol{E}_y(\boldsymbol{o})$, and its finite-order jet is used as the latent state $\boldsymbol{\xi} := J^{r-1}\boldsymbol{y}$. Rather than learning an unconstrained latent transition, we impose the derivative-chain structure

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}_r\boldsymbol{\xi} + \boldsymbol{B}_r\boldsymbol{\nu}, \quad \boldsymbol{\nu} := \boldsymbol{y}^{(r)} = \boldsymbol{G}(\boldsymbol{\xi}, \boldsymbol{u}) \tag{6}$$

where $\boldsymbol{A}_r$ and $\boldsymbol{B}_r$ are the fixed block companion matrices defining an $r$-th order integrator chain as Eq.(17) , while $\boldsymbol{G}$ model the unknown highest-order dynamics. Given observation–control trajectories $\mathcal{D}$, we jointly learn $\boldsymbol{E}_y$, the structured latent dynamics (6), and an observation decoder $\widehat{o} = \boldsymbol{D}_o(\boldsymbol{\xi})$, without supervision on the analytical flat output or the underlying state. Furthermore, by explicitly exploiting the derivative-chain structure of differentially flat systems, this learning objective can improve data efficiency.

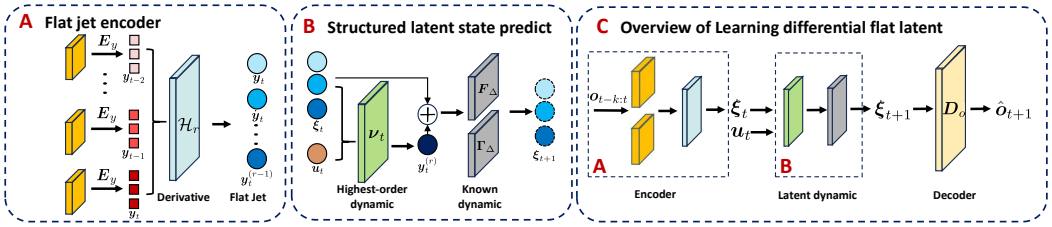



Figure 2: **Flatness-structured latent model.** (A) Flat-jet encoding from observation histories. (B) Learned highest-order dynamics with a prescribed derivative chain. (C) Overall architecture for observation prediction.

# 4 METHOD

We implement the flatness structure established in Section 3 by modifying two components of a sequential variational autoencoder: the encoder constructs a flat-output jet from observation histories, and the latent transition enforces its derivative-chain dynamics. Figure 2 summarizes the model.

## 4.1 FLAT-JET ENCODER

Given a observation history $\mathcal{O}_t = \{\boldsymbol{o}_{t-K}, \cdots, \boldsymbol{o}_t\}$, history window length is mathematically required to satisfy $K \geq r-1$ for jet, a shared map $\boldsymbol{E}_y$ is applied to each time to extract flat outputs $\boldsymbol{y}_{t-j} = \boldsymbol{E}_y(\boldsymbol{o}_{t-j})$ and standard deviations $\boldsymbol{\sigma}_{y,t-j}$. The resulting sequence is passed through a fixed causal finite-difference operator $\mathcal{H}_{r,\Delta t}$, such that the current flat output and all its derivatives are estimated from the same learned mapping $\boldsymbol{o}_{t-j} \xrightarrow{\boldsymbol{E}_y} \boldsymbol{y}_{t-j}, \boldsymbol{y}_{t-K:t} \xrightarrow{\mathcal{H}_{r,\Delta t}} \boldsymbol{\xi}_t$ and get the inference distribution as

$$q_\phi(\boldsymbol{\xi}_t \mid \boldsymbol{o}_{t-K:t}) = \mathcal{N}(\boldsymbol{\mu}_{\xi,t}, \boldsymbol{\Sigma}_{\xi,t}), \quad \boldsymbol{\mu}_{\xi,t} = \mathcal{H}_{r,\Delta t}\boldsymbol{\mu}_{Y,t}, \quad \boldsymbol{\Sigma}_{\xi,t} = \mathcal{H}_{r,\Delta t}\,\mathrm{diag}\big(\boldsymbol{\sigma}^2_{Y,t}\big)\,\mathcal{H}^\top_{r,\Delta t} \tag{7}$$

Here, $\boldsymbol{\mu}_{Y,t} = \mathrm{vec}(\boldsymbol{E}_{y,t-K}, \ldots, \boldsymbol{E}_{y,t})$, $\boldsymbol{\sigma}_{Y,t} = \mathrm{vec}(\boldsymbol{\sigma}_{y,t-K}, \ldots, \boldsymbol{\sigma}_{y,t})$ stack the means and standard deviations produced by applying $\boldsymbol{E}_y$ to each observation in the history $\boldsymbol{o}_{t-K:t}$. The uncertainty in the learned flat-output map $\boldsymbol{E}_y$ is propagated through the finite-difference operator to the latent state $\boldsymbol{\xi}$, while preserving the discrete derivative relations required by its flat-jet structure. The differential operator and the design of the flat-jet encoder are discussed in detail in Appendix A.5.

## 4.2 STRUCTURED LATENT DYNAMICS

The flat-jet representation separates the latent evolution into a prescribed derivative chain and an unknown highest-order controlled dynamics. Specifically, the highest-order derivative is predicted from the current latent state and input, while all jet coordinates are propagated through the fixed derivative chain in Eq. (6): $(\boldsymbol{\xi}_t, \boldsymbol{u}_t) \xrightarrow{\boldsymbol{a}_\theta, \boldsymbol{G}_\theta} \boldsymbol{\nu}_t \xrightarrow{\boldsymbol{A}_r, \boldsymbol{B}_r} \boldsymbol{\xi}_{t+1}$. The network predicts the mean highest-order dynamics $\boldsymbol{\mu}_{\boldsymbol{\nu},t} = \boldsymbol{G}_\theta(\boldsymbol{\xi}_t, \boldsymbol{u}_t)$ together with $\boldsymbol{\Sigma}_{\boldsymbol{\nu},t}$ that represents modeling uncertainty in the highest-order derivative. Over each sampling interval, this uncertainty acts through the highest-order channel and is propagated through the prescribed derivative chain, yielding

$$p_\theta(\boldsymbol{\xi}_{t+1} \mid \boldsymbol{\xi}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{\mu}_{\xi,t+1}, \boldsymbol{Q}_{\Delta,t}), \tag{8}$$

where $\boldsymbol{\mu}_{\xi,t+1} = \boldsymbol{F}_\Delta \boldsymbol{\xi}_t + \boldsymbol{\Gamma}_\Delta \boldsymbol{\mu}_{\nu,t}$ and $\boldsymbol{Q}_{\Delta,t} = \boldsymbol{\Gamma}_\Delta \boldsymbol{\Sigma}_{\nu,t} \boldsymbol{\Gamma}_\Delta^\top + \epsilon \boldsymbol{I}$ are the propagated mean and covariance. The term $\epsilon \boldsymbol{I}$ ensures a positive-definite covariance for the Gaussian transition and the subsequent KL divergence. The discrete-time derivative-chain matrices are $\boldsymbol{F}_\Delta = e^{\boldsymbol{A}_r \Delta t}$ and $\boldsymbol{\Gamma}_\Delta = \int_0^{\Delta t} e^{\boldsymbol{A}_r(\Delta t - \tau)} \boldsymbol{B}_r \, \mathrm{d}\tau$. The derivation of the probabilistic transition is provided in Appendix A.6. By constraining the latent state and its evolution to the form, our model embeds differential flatness into latent dynamics, replacing an unconstrained latent transition with a flatness-structured one.

## 4.3 LEARNING OBJECTIVE

Based on the flatness-structured probabilistic model above, we derive the training objective using variational inference. Since the flat jet parameterizes the physical state, the observation can also be

reconstructed from $\boldsymbol{\xi}_t$. We use a decoder $\boldsymbol{D}_\psi$ to parameterize $p_\psi(\boldsymbol{o}_t \mid \boldsymbol{\xi}_t)$ and jointly train the flat-jet encoder, structured latent dynamics, and observation decoder by minimizing

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{rec}} + \lambda_{\text{dyn}}\mathcal{L}_{\text{dyn}}, \quad \mathcal{L}_{\text{rec}} = -\sum_{t=0}^{T} \mathbb{E}_{q_\phi(\boldsymbol{\xi}_t|\mathcal{O}_t)} \left[\log p_\psi(\boldsymbol{o}_t \mid \boldsymbol{\xi}_t)\right] \\ \mathcal{L}_{\text{dyn}} &= \sum_{t=0}^{T-1} \mathbb{E}_{q_\phi(\boldsymbol{\xi}_t|\mathcal{O}_t)} \left[D_{\text{KL}}(q_\phi(\boldsymbol{\xi}_{t+1} \mid \mathcal{O}_{t+1}) \,\|\, p_\theta(\boldsymbol{\xi}_{t+1} \mid \boldsymbol{\xi}_t, \boldsymbol{u}_t))\right]. \end{aligned} \tag{9}$$

The reconstruction term preserves observation information, while the dynamics term aligns the encoded next-state distribution with the flatness-structured latent transition. The outer expectation accounts for uncertainty in the current encoded flat jet evaluated using the VAE style reparameterization $\boldsymbol{Y}_t = \boldsymbol{\mu}_{Y,t} + \boldsymbol{\sigma}_{Y,t} \odot \boldsymbol{\epsilon}_t, \boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \boldsymbol{\xi}_t = \mathcal{H}_{r,\Delta t}\boldsymbol{Y}_t$. Together, these components learn a flatness-structured latent model from observation–control trajectories, enabling planning and control based on differential flatness.

### 4.4 Downsteam Planning and Control

We consider a finite-horizon transfer task in which the initial and desired terminal conditions are specified by short observation histories $\mathcal{O}_{\text{ini}} = o_{-K:0}$ and $\mathcal{O}_{\text{goal}} = o^d_{T-K:T}$. Applying the learned flat-jet encoder to these histories provides the initial and desired terminal latent states $\boldsymbol{\xi}_0, \boldsymbol{\xi}^d_T$. By Corollary 1, the transfer task can be formulated as boundary-value planning in the learned flat-output space. As commonly done for differentially flat systems Beaver & Malikopoulos (2024), we plan an $r$-order smooth trajectory $\boldsymbol{y}_d(t)$ in the learned flat-output coordinates that satisfies these boundary conditions. Differentiating this trajectory gives the desired latent state $\boldsymbol{\xi}_d(t)$ and latent dynamic input $\boldsymbol{\nu}_d(t)$. We then tracked $\boldsymbol{\xi}_d(t)$ in the latent space with latent dynamic feedforward $\boldsymbol{\nu}_d(t)$.

During execution, the current latent state $\widehat{\boldsymbol{\xi}}_t$ can be estimated from a rolling observation history ending at the latest observation, The desired latent-dynamics input is corrected using feedback in the flat-jet latent space, and the control input $\boldsymbol{u}_t$ is then computed by an auxiliary recovery network $R_u$.

$$\boldsymbol{\nu}^{\text{cmd}}_t = \boldsymbol{\nu}_d(t) + \boldsymbol{K}_\xi \left(\boldsymbol{\xi}_d(t) - \widehat{\boldsymbol{\xi}}_t\right) + \boldsymbol{K}_I \int_0^t \left(\boldsymbol{\xi}_d(\tau) - \widehat{\boldsymbol{\xi}}_\tau\right) d\tau, \quad \boldsymbol{u}_t = R_u\left(\widehat{\boldsymbol{\xi}}_t, \boldsymbol{\nu}^{\text{cmd}}_t\right). \tag{10}$$

where $\boldsymbol{K}_\xi$ and $\boldsymbol{K}_I$ are the latent-space tracking and integral gains, respectively. The trajectory planning, latent state estimation, and auxiliary input recovery network are detailed in Appendix A.7 -A.9.

Together, these components turn differential flatness into a learnable representation that provides a direct interface from observed trajectories to planning and closed-loop control.

## 5 Experiments

We evaluate whether the model learns flat coordinates without ground-truth coordinate supervision and supports prediction, planning, and control under incomplete observations. On a three-dimensional nonlinear system and an image-based control system, we compare our method with prior structured latent dynamics methods (history-based PCC and Deep Koopman) in prediction accuracy, data efficiency, control performance, and computational cost. Note that flat outputs are nonunique and admit equivalent representations under invertible affine transformations. For ease of comparison, all learned flat outputs are aligned with their analytical counterparts using an affine map fitted on training data only, detailed in Appendix A.10. Experiment details are provided in the Appendix A.11.

### 5.1 Experiments: Three-Dimensional Nonlinear System

We consider a three-dimensional nonlinear system with two observation maps:

$$\begin{aligned} \dot{x}_1 &= x_3 - x_2 u \\ \dot{x}_2 &= -x_2 + u \\ \dot{x}_3 &= x_2 - x_1 + 2x_2(u - x_2), \end{aligned} \qquad \begin{aligned} \boldsymbol{o}^{(1)} &= [x_1, x_2, x_3]^\top \\ \boldsymbol{o}^{(2)} &= [x_1, x_2, x_3^2]^\top \end{aligned}$$

The system admits the flat output $y = x_1 + \frac{1}{2}x_2^2$ for $x_2 \neq -1$. While $o^{(1)}$ retains full-state information, $o^{(2)}$ is incomplete but still determines $y$. Comparing the two tests whether the learned flatness-structured latent space offers an advantage over generic latent models under incomplete observations. The results show comparable performance under full-state observations, but under incomplete observations our method better distinguishes the underlying states and improves prediction and closed-loop control, particularly with limited training data.

Figure 3 shows that our method learns the system's flat output. After fitting output scale and offset on the training data, the learned $y$, $\dot{y}$, and $\ddot{y}$ closely match their analytical counterparts on both training and test data; the same scale is applied to the derivatives.

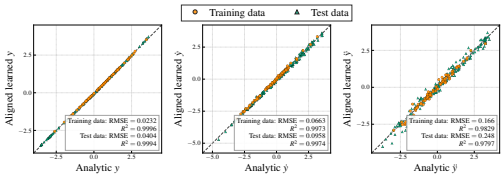
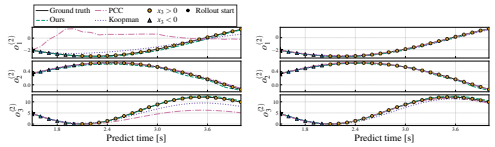



Figure 3: **Learning of analytical flat coordinates.** Dashed lines indicate exact agreement.

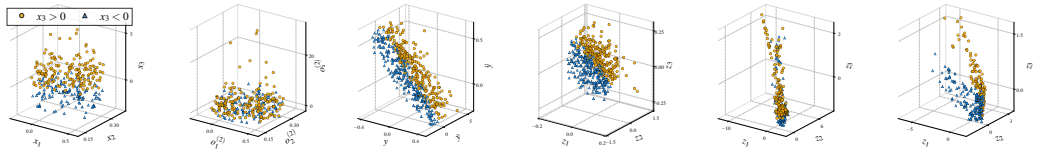
Figure 4 compares open-loop predictions on an unseen test trajectory with limited training data. By imposing a flatness-based prior on latent dynamics, our model achieves accurate predictions with fewer samples than history-based PCC and Koopman. The baselines improve substantially as more training data become available

Figure 5 visualizes how representations learned using only $o^{(2)}$ observations organize states with opposite signs of $x_3$, providing visual evidence that the learned flat-output jet, used as the latent state, retains a correspondence with the physical state. These states overlap in observation space, while the analytical differentially flat coordinates reveal distinct sign-dependent regions. Using only $o^{(2)}$ observations, our method learns differentially flat coordinates in which states with opposite signs of $x_3$ form visibly distinct regions. With the same limited training data, PCC exhibits greater mixing between the two groups; increasing its training data makes their separation more apparent.




(a) 15% training data (b) 60% training data

Figure 4: **Open-loop prediction under ambiguous observations $o^{(2)}$.** (a) less and (b) more training data.



(a) State (b) Observation (c) Analytical flat coordinates (d) Ours (e) PCC(small data) (f) PCC(large data)

Figure 5: **Latent representations.** The same samples are shown in (a) state space, (b) observation space, (c) analytical flat coordinates, (d) learned flat coordinates, and PCC latent spaces with (e) less and (f) more training data. All learned representations use only $o^{(2)}$ observations; (d) and (e) use the same training data.

For control, we implement two representative structured latent dynamics methods, PCC and Deep Koopman(DKP), as baselines. PCC uses iLQR-based MPC (Levine et al., 2020), while Deep Koopman uses linear MPC in its learned latent space (Li et al., 2025). Both jointly optimize $H$ input variables over a prediction horizon of $H$ steps, execute the first optimized input, and replan using the latest observations. Our method instead plans a parameterized trajectory in the learned flat-output space, whose number of coefficients is independent of the temporal discretization for a fixed parameterization (fixed-degree polynomial with undetermined coefficients). The resulting trajectory and its derivatives then support feedback control and direct input recovery through Eq. 10, avoiding repeated optimization over the input sequence.

Table 2 highlights the benefit of the flatness structure under incomplete observations. Table 2 reports mean results over fixed-duration transfers between different initial-target observation pairs, without state access. Our feedback frequency matches the baselines' MPC replanning frequency. Time denotes average computation per update, including amortized trajectory-planning cost. While

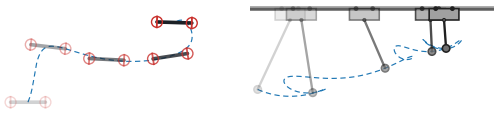
Table 2: **Prediction and control performance.** Percentages indicate training data usage. Pred.: model predict RMSE in unseen trajectories; Term.: terminal control error; Time: average computation time per control step.

| Observation | Method | 15% | | 30% | | 60% | | Time (ms) |
|---|---|---|---|---|---|---|---|---|
| | | Pred.↓ | Term.↓ | Pred.↓ | Term.↓ | Pred.↓ | Term.↓ | |
| $o^{(1)}$ | PCC | 0.035 | 0.132 | 0.012 | 0.075 | 0.005 | 0.021 | 51.71 |
| | DKP | 0.016 | 0.098 | 0.011 | 0.057 | 0.004 | 0.017 | 86.31 |
| | Ours | 0.022 | 0.109 | 0.009 | 0.062 | 0.004 | 0.016 | **14.64** |
| $o^{(2)}$ | PCC | 1.147 | 0.884 | 0.864 | 0.414 | 0.458 | 0.177 | 60.237 |
| | DKP | 0.988 | 0.797 | 0.672 | 0.402 | 0.344 | 0.135 | 95.288 |
| | Ours | **0.281** | **0.239** | **0.130** | **0.131** | **0.101** | **0.097** | **15.249** |

the methods perform comparably under $o^{(1)}$, our model achieves lower prediction and control errors under $o^{(2)}$, especially with limited data, supporting the data-efficiency benefit of its flatness structure. This structure also reduces control computation by enabling flat-output trajectory planning and direct input recovery, avoiding repeated input-sequence optimization in MPC.

### 5.2 Experiments: Image-Based Control of Differentially Flat Systems

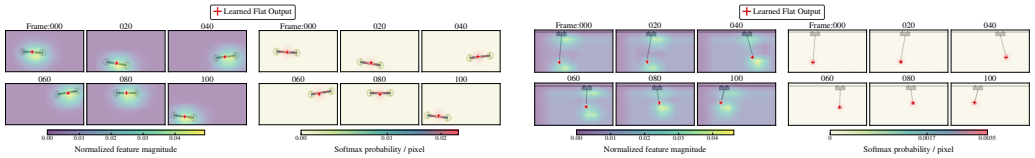
We construct two visual control environments by rendering a planar bicopter and a planar overhead crane with variable rope length as Figure 6. Detailed dynamic model is shown in Appendix A.11.1. The images capture their configurations but do not directly reveal velocities or internal actuator states. For both systems, the configurations visible in the images determine the flat outputs, satisfying our observation condition. Models are trained on image sequences and control inputs, without supervision on physical states or analytical flat coordinates.



(a) Bicopter (b) Crane

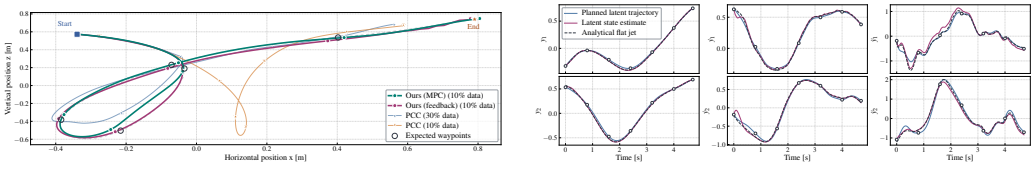
Figure 6: **Visual control environments.** Overlaid configurations illustrate motion over time.

For visual observations, we evaluate two implementations of the flat-jet encoder in our method: CNN+softmax, which incorporates a coordinate-extraction bias, and a standard CNN+MLP. Figure 7 shows that both learn flat outputs from images and generalize to unseen trajectories, indicating that the learned flatness structure does not depend on a specialized coordinate-extraction architecture. These results indicate that the flat-jet encoder and structured latent dynamics help our method capture control-relevant variations in images, as highlighted in the figure.




(a) Bicopter (b) Crane

Figure 7: **Learned flat-output along unseen image trajectories for the planar bicopter and crane.** Left: normalized feature magnitudes from CNN+MLP. Right: per-pixel softmax probabilities from CNN+softmax. Red mark: the learned flat-output coordinates




(a) Waypoint Transfers (b) Latent Trajectory Tracking

Figure 8: **Bicopter Control from Images**

Figure 8 compares bicopter control on fixed-time transfers between multiple waypoints. Using only 10% of the training data, our learned model achieves accurate waypoint transfers with both image-based latent feedback and the same latent-space MPC scheme used for PCC. PCC requires 30%

of the data to achieve comparable performance, supporting the data-efficiency benefit of the flatness structure beyond the choice of controller. The right panels show tracking of the planned latent trajectories using image-based feedback, with the latent estimates closely matching the analytical flat jets. Together with the successful waypoint transfers in physical space, these results provide evidence that the learned flat-jet encoder and structured latent dynamics capture the system's differential-flatness structure. Quantitative comparisons with additional methods are provided in the Table 4.

### 5.3 ABLATION STUDIES OF FLAT-JET ENCODER AND STRUCTURED LATENT DYNAMICS

We examine how the flat-jet encoder and structured latent dynamics contribute to learning flat coordinates. Together, they impose the derivative relationships and evolution structure of a flat-output jet, encouraging $\boldsymbol{E}y$ to extract flat outputs from observations. We ablate each component separately and jointly. We replace the fixed finite-difference operator $\mathcal{H}_{r,\Delta t}$ with an MLP applied to the concatenated encoded observation sequence, preserving the input window and output dimension while removing the explicit derivative constraint and replace the prescribed integrator chain and learned highest-order dynamics with a general latent transition model. Table 3 reports $R^2$ for the learned flat output and its derivatives against their analytical counterparts on unseen trajectories. Alignment is fitted only to the flat output on training trajectories and applied consistently to the entire test jet: the scale applies to all derivative orders, while any offset applies only to the output.

Table 3: **Ablation of three-dimensional nonlinear system.** Percentages means training data usage.

| | Flat-jet encoder | Structured dynamics | 15% | | | 30% | | | 60% | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Variant | | | $y$ | $\dot{y}$ | $\ddot{y}$ | $y$ | $\dot{y}$ | $\ddot{y}$ | $y$ | $\dot{y}$ | $\ddot{y}$ |
| Full model | ✓ | ✓ | 0.981 | 0.980 | 0.980 | 0.992 | 0.990 | 0.990 | 0.997 | 0.997 | 0.997 |
| w/o encoder structure | × | ✓ | 0.892 | 0.695 | 0.606 | 0.925 | 0.867 | 0.853 | 0.967 | 0.958 | 0.955 |
| w/o dynamics structure | ✓ | × | 0.807 | 0.763 | 0.719 | 0.896 | 0.887 | 0.844 | 0.943 | 0.940 | 0.940 |
| w/o both | × | × | <0.5 | <0.5 | <0.5 | <0.5 | <0.5 | <0.5 | 0.787 | <0.5 | <0.5 |

The results show that either component alone can recover flat coordinates in the evaluated systems, although more training data are needed. The dynamics-consistency loss couples the encoded latent states with their predicted evolution, allowing the structure imposed by either the flat-jet encoder or the structured dynamics to guide the other component. The missing structural constraint must therefore be learned from data. Combining both components explicitly constrains the latent coordinates and their evolution, reducing the hypothesis space and improving data efficiency.

These variants also connect our method to existing latent dynamics approaches. Retaining structured dynamics with a general encoder is closely related to learning feedback-linearizing coordinates (Goswami et al., 2023), although that work assumes measurable states rather than addressing recovery from incomplete observations. Removing both components yields a simplified encoder–transition–decoder baseline sharing the backbone of E2C and PCC, without their method-specific structural constraints (Watter et al., 2015; Levine et al., 2020).

## 6 CONCLUSION

For latent dynamics learning of differentially flat systems, we introduced flatness structure to explicitly relate learned representations to the underlying physical states, rather than merely predict observations. We established that, when observations determine the flat output and the learned representation exactly realizes its flatness structure, latent trajectories support state–input recovery and a locally equivalent formulation of the original control problem. These results motivated such structured latent dynamic model based on a VAE. Experiments demonstrate that the model learns a flatness-equivalent representation from incomplete observations, improving prediction and control with limited data while enabling efficient trajectory planning and closed-loop execution. An open question is whether an approximate flatness structure learned from data can support effective control beyond differentially flat systems. Addressing this question could guide the learning of representations with control priors and deepen our understanding of the underlying controlled dynamics from high-dimensional observations.

### AI USE STATEMENT

We used generative AI tools to improve the clarity and readability of the manuscript, assist with literature searches, and support the development of experimental code. We also used these tools to help document experimental settings and implementation details from our code to improve reproducibility. We reviewed the AI-assisted text, references, and code, and checked the experimental descriptions against the implementation. We take responsibility for the final content of this work, including all text, claims, code, and reported results.

### ETHICS STATEMENT

Source code can be found at `https://github.com/LEEDJ7/Differentially-Flat-4-VAE`

### REPRODUCIBILITY STATEMENT

Detailed theoretical analyses and proofs are provided in the Appendix A.3-A.9, together with experimental settings and implementation details Appendix A.10 and A.11, including system dynamics, data generation, network architectures, and control configurations. Source code can be found at `https://github.com/LEEDJ7/Differentially-Flat-4-VAE`

## REFERENCES

Ershad Banijamali, Rui Shu, mohammad Ghavamzadeh, Hung Bui, and Ali Ghodsi. Robust locally-linear controllable embedding. In Amos Storkey and Fernando Perez-Cruz (eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pp. 1751–1759. PMLR, 09–11 Apr 2018. URL `https://proceedings.mlr.press/v84/banijamali18a.html`.

Logan E. Beaver and Andreas A. Malikopoulos. Optimal control of differentially flat systems is surprisingly easy. *Automatica*, 159:111404, 2024. ISSN 0005-1098. doi: 10.1016/j.automatica.2023.111404.

Xiaoyuan Cheng, Wenxuan Yuan, Yiming Yang, Yuanzhao Zhang, Sibo Cheng, Yi He, and Zhuo Sun. Information shapes koopman representation. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 151380–151429, 2026.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2067–2075, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/chung15.html`.

Yonathan Efroni, Chi Jin, Akshay Krishnamurthy, and Sobhan Miryoosefi. Provable reinforcement learning with a short-term memory. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5832–5850. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/efroni22a.html`.

MICHEL FLIESS, JEAN LÉVINE, PHILIPPE MARTIN, and PIERRE ROUCHON. Flatness and defect of non-linear systems: Introductory theory and examples. *International Journal of Control*, 61(6):1327–1361, 1995. doi: 10.1080/00207179508921959.

Darshan Gadginmath, Vishaal Krishnan, and Fabio Pasqualetti. Data-driven feedback linearization using the koopman generator. *IEEE Transactions on Automatic Control*, 69(12):8844–8851, 2024. doi: 10.1109/TAC.2024.3417188.

Raktim Gautam Goswami, Prashanth Krishnamurthy, and Farshad Khorrami. Data-driven deep learning based feedback linearization of systems with unknown dynamics. In *2023 American Control Conference (ACC)*, pp. 66–71, 2023. doi: 10.23919/ACC55779.2023.10156140.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. 640(8059):647–653. ISSN 1476-4687. doi: 10.1038/s41586-025-08744-2. URL https://doi.org/10.1038/s41586-025-08744-2.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, April 2025a. ISSN 1476-4687. doi: 10.1038/s41586-025-08744-2.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, April 2025b. ISSN 1476-4687. doi: 10.1038/s41586-025-08744-2.

Adam W. Hall, Melissa Greeff, and Angela P. Schoellig. Differentially flat learning-based model predictive control using a stability, state, and input constraining safety filter. *IEEE Control Systems Letters*, 7:2191–2196, 2023. doi: 10.1109/LCSYS.2023.3285616.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8387–8406. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/hansen22a.html.

Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Second international conference on learning representations, ICLR*, volume 19, pp. 121, 2014.

Philippe Laferrière, Samuel Laferrière, Steven Dahdah, James Richard Forbes, and Liam Paull. Deep koopman representation for control over images (dkrci). In *2021 18th Conference on Robots and Vision (CRV)*, pp. 158–164, 2021. doi: 10.1109/CRV52889.2021.00029.

Nir Levine, Yinlam Chow, Rui Shu, Ang Li, Mohammad Ghavamzadeh, and Hung Bui. Prediction, consistency, curvature: Representation learning for locally-linear control. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxG_0EtDS.

Zhaoyang Li, Minghao Han, and Xunyuan Yin. MamKO: Mamba-based Koopman operator for modeling and predictive control. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 93458–93487, 2025.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950, November 2018. ISSN 2041-1723. doi: 10.1038/s41467-018-07210-0.

Shangjie Frank Ma, Anni Zhao, and Jian-Qiao Sun. Identification of linear flat outputs using neural networks—Examples of two-degree-of-freedom underactuated mechanical systems. *Mechanical Systems and Signal Processing*, 216:111471, 2024. ISSN 0888-3270. doi: 10.1016/j.ymssp.2024.111471.

Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pp. 2520–2525, 2011. doi: 10.1109/ICRA.2011.5980409.

Keyan Miao, Han Wang, Xuda Ding, Konstantinos Gatsis, Andreas Krause, and Antonis Papachristodoulou. Learning koopman representations with controllability guarantees. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jITPFROpWN.

Ilan Naiman, N. Benjamin Erichson, Pu Ren, Michael W Mahoney, and Omri Azencot. Generative modeling of regular and irregular time series data via koopman vaes. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 50547–50573. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/dbeb7e621d4a554069a6a775da0f7273-Paper-Conference.pdf.

Ilan Naiman, N. Benjamin Erichson, Pu Ren, Michael W. Mahoney, and Omri Azencot. Generative modeling of regular and irregular time series data via koopman VAEs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=eY7sLb0dVF.

Rui Shu, Tung Nguyen, Yinlam Chow, Tuan Pham, Khoat Than, Mohammad Ghavamzadeh, Stefano Ermon, and Hung Bui. Predictive coding for locally-linear control. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8862–8871. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/shu20a.html.

Maximilian Stölzle and Cosimo Della Santina. Input-to-state stable coupled oscillator networks for closed-form model-based control in latent space. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 82010–82059. Curran Associates, Inc., 2024. doi: 10.52202/079017-2607.

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.

Shengjie Wang, Shaohuai Liu, Weirui Ye, Jiacheng You, and Yang Gao. EfficientZero v2: Mastering discrete and continuous control with limited data. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51041–51062. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/wang24at.html.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Shaorong Xie, Zhenyu Zhang, Hang Yu, and Xiangfeng Luo. Recurrent prediction model for partially observable MDPs. *Information Sciences*, 620:125–141, 2023. ISSN 0020-0255. doi: 10.1016/j.ins.2022.11.065.

Fengjun Yang, Jake Welde, and Nikolai Matni. Learning flatness-preserving residuals for pure-feedback systems. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, pp. 3035–3042, 2025. doi: 10.1109/CDC57313.2025.11313004.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. Learning physics constrained dynamics using autoencoders. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 17157–17172. Curran Associates, Inc., 2022. doi: 10.52202/068431-1248.

Yaofeng Desmond Zhong and Naomi Leonard. Unsupervised learning of lagrangian dynamics from images for prediction and control. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 10741–10752. Curran Associates, Inc., 2020.

Zhijie Zhu, Edward Schmerling, and Marco Pavone. A convex optimization approach to smooth trajectories for motion planning with car-like robots. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 835–842, 2015. doi: 10.1109/CDC.2015.7402333.

# A Appendix

## A.1 Additional related work

**Differential Flatness.** Differential flatness provides a finite differential parameterization of system trajectories, expressing states and inputs through flat outputs and their derivatives (FLIESS et al., 1995). This property supports trajectory planning directly in flat-output space. Recent work exploits this parameterization to simplify nonlinear optimal control (Beaver & Malikopoulos, 2024) and combines flatness-based MPC with learned input transformations and safety filters (Hall et al., 2023). Data-driven approaches also seek to identify or preserve flat structure. (Ma et al., 2024) identifies linear flat outputs for linearized underactuated mechanical systems using neural networks. (Yang et al., 2025) learns structured residual dynamics that preserve the flatness and flat outputs of a nominal pure-feedback model. These developments connect differential flatness with dynamics learning while relying on specified model structures or physical measurements. Our work studies flatness as a structural bias for representation learning from high-dimensional observations, linking the observation conditions for such representations to their use in prediction, planning, and control.

**Variational Autoencoders for Latent Dynamics Learning.** Variational autoencoders provide a probabilistic framework for jointly learning latent representations and generative models through variational inference (Kingma & Welling, 2014). Sequential extensions, such as variational recurrent neural networks (Chung et al., 2015), incorporate temporal dependencies into latent-variable modeling. This framework also underlies recurrent state-space world models used for visual prediction and control, including DreamerV3 (Hafner et al.). Recent studies introduce explicit dynamical or geometric structure into variational models. KoVAE uses a Koopman-inspired linear conditional prior to model regular and irregular time series (Naiman et al.). These works illustrate how variational learning can accommodate structured representations and temporal evolution. Our model adopts this probabilistic framework and organizes the latent representation around learned flat outputs and their derivative chain, connecting observation-based learning with physical-state information and control.

## A.2 Differential Flatness and Dynamic Extension

Consider the nonlinear control system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \qquad \boldsymbol{x} \in \mathbb{R}^n, \quad \boldsymbol{u} \in \mathbb{R}^m. \tag{11}$$

A system is differentially flat if there exists a flat output $\boldsymbol{y} \in \mathbb{R}^m$ whose finite-order derivatives parameterize the state and input. We consider the state-dependent flat output

$$\boldsymbol{y} = \phi(\boldsymbol{x}) \tag{12}$$

and the consecutive-order parameterization used in the main text:

$$\boldsymbol{x} = \boldsymbol{\Psi}_x(J^{r-1}\boldsymbol{y}), \qquad \boldsymbol{u} = \boldsymbol{\Psi}_u(J^r\boldsymbol{y}), \tag{13}$$

where $r$ is the known flatness order and

$$J^q\boldsymbol{y} := \operatorname{col}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ldots, \boldsymbol{y}^{(q)}\right). \tag{14}$$

Define the flat-jet state and its highest-order derivative as

$$\boldsymbol{\xi} := J^{r-1}\boldsymbol{y} = \operatorname{col}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ldots, \boldsymbol{y}^{(r-1)}\right), \qquad \boldsymbol{\nu} := \boldsymbol{y}^{(r)}. \tag{15}$$

The dynamics of $\boldsymbol{\xi}$ follow the prescribed derivative chain

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}_r\boldsymbol{\xi} + \boldsymbol{B}_r\boldsymbol{\nu}, \tag{16}$$

where

$$\boldsymbol{A}_r = \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_m & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \vdots & \vdots & \ddots & \boldsymbol{I}_m \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}, \qquad \boldsymbol{B}_r = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \boldsymbol{I}_m \end{bmatrix}. \tag{17}$$

Thus, the flat-space dynamics are fixed except for the highest-order derivative $\boldsymbol{\nu}$. On a regular flatness domain, $\boldsymbol{\nu}$ is determined by the current flat jet and physical input,

$$\boldsymbol{\nu} = \boldsymbol{F}(\boldsymbol{\xi}, \boldsymbol{u}), \tag{18}$$

while the corresponding physical state and input are recovered as

$$\boldsymbol{x} = \boldsymbol{\Psi}_x(\boldsymbol{\xi}), \qquad \boldsymbol{u} = \boldsymbol{\Psi}_u(\boldsymbol{\xi}, \boldsymbol{\nu}). \tag{19}$$

Consequently, the original controlled dynamics can be represented in the flat space as

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}_r \boldsymbol{\xi} + \boldsymbol{B}_r \boldsymbol{G}(\boldsymbol{\xi}, \boldsymbol{u}). \tag{20}$$

Dynamic extension augments the original input as part of the state and uses its derivative as the new control input. The construction therefore applies to sufficiently smooth input trajectories and may change the form of the corresponding input and rate constraints. Suppose that a flat system admits the general parameterization

$$\boldsymbol{x} = \boldsymbol{\Psi}_x(J^{r_x}\boldsymbol{y}), \qquad \boldsymbol{u} = \boldsymbol{\Psi}_u(J^{r_u}\boldsymbol{y}), \qquad r_u > r_x + 1. \tag{21}$$

Introduce the extended state and input

$$\widetilde{\boldsymbol{x}} := \mathrm{col}(\boldsymbol{x}, \boldsymbol{u}), \qquad \widetilde{\boldsymbol{u}} := \dot{\boldsymbol{u}}. \tag{22}$$

The extended dynamics are

$$\dot{\widetilde{\boldsymbol{x}}} = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ \widetilde{\boldsymbol{u}} \end{bmatrix}. \tag{23}$$

Because $r_u > r_x$, the extended state satisfies

$$\widetilde{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{\Psi}_x(J^{r_x}\boldsymbol{y}) \\ \boldsymbol{\Psi}_u(J^{r_u}\boldsymbol{y}) \end{bmatrix} =: \widetilde{\boldsymbol{\Psi}}_x(J^{r_u}\boldsymbol{y}). \tag{24}$$

Differentiating the original input parameterization gives

$$\begin{aligned} \widetilde{\boldsymbol{u}} &= \frac{d}{dt}\boldsymbol{\Psi}_u(J^{r_u}\boldsymbol{y}) \\ &= \sum_{k=0}^{r_u} \frac{\partial \boldsymbol{\Psi}_u}{\partial \boldsymbol{y}^{(k)}} \boldsymbol{y}^{(k+1)} =: \widetilde{\boldsymbol{\Psi}}_u(J^{r_u+1}\boldsymbol{y}). \end{aligned} \tag{25}$$

Letting $\widetilde{r} = r_u + 1$, we obtain

$$\widetilde{\boldsymbol{x}} = \widetilde{\boldsymbol{\Psi}}_x(J^{\widetilde{r}-1}\boldsymbol{y}), \qquad \widetilde{\boldsymbol{u}} = \widetilde{\boldsymbol{\Psi}}_u(J^{\widetilde{r}}\boldsymbol{y}). \tag{26}$$

Hence, the dynamically extended system has the consecutive-order form considered in the main text while retaining $\boldsymbol{y}$ as its flat output.

### A.3 Proof of Proposition 1

*Proof.* Consider an observation trajectory $\boldsymbol{o}(\tau) = \boldsymbol{h}(\boldsymbol{x}(\tau))$ over a finite history interval ending at time $t$. By Definition 1, applying $E_y$ pointwise to this trajectory gives

$$\boldsymbol{E}_y(\boldsymbol{o}(\tau)) = \boldsymbol{E}_y(h(\boldsymbol{x}(\tau))) = \boldsymbol{\phi}(\boldsymbol{x}(\tau)) = \boldsymbol{y}(\tau) \tag{27}$$

for every $\tau$ in the interval. The encoded observation history therefore recovers the corresponding flat-output trajectory.

Taking its time derivatives at $t$ gives

$$J^r[\boldsymbol{E}_y(\boldsymbol{o})](t) = J^r\boldsymbol{y}(t). \tag{28}$$

Thus, the latent representation constructed from the observation history satisfies

$$\boldsymbol{z}_f(t) = J^r\boldsymbol{y}(t). \tag{29}$$

Since this jet contains both $J^{r-1}\boldsymbol{y}(t)$ and $\boldsymbol{y}^{(r)}(t)$, the flatness reconstruction maps yield

$$\boldsymbol{x}(t) = \boldsymbol{\Psi}_x(J^{r-1}\boldsymbol{y}(t)), \qquad \boldsymbol{u}(t) = \boldsymbol{\Psi}_u(J^r\boldsymbol{y}(t)). \tag{30}$$

Hence, the flat jet constructed from the encoded observation history contains sufficient information to recover the physical state and input at time $t$. In practice, the continuous-time jet is approximated from the sampled observation history by the finite-difference operator. □

### A.4 Proof of Corollary 1

*Proof.* We first specify what it means for the learned model to realize the flatness parameterization. On a regular domain, assume that the learned observation encoder $\boldsymbol{E}_y$, together with the derivative construction from the encoded observation history, recovers a valid state-dependent flat output and its jet up to order $r$.

$$\widehat{\boldsymbol{y}}(t) = \boldsymbol{E}_y(\boldsymbol{o}(t)) = \phi(\boldsymbol{x}(t)) = \boldsymbol{y}(t), \qquad \boldsymbol{z}_f(t) := J^r\widehat{\boldsymbol{y}}(t) = J^r\boldsymbol{y}(t) \tag{31}$$

where $\boldsymbol{\phi}$ is a flat output of the original system. Consider a dynamically consistent latent trajectory $\boldsymbol{z}_f(t) = J^r\boldsymbol{y}(t) \in \mathcal{F}_f$ that remains in the flatness domain. Its corresponding physical trajectory is given by

$$\boldsymbol{\Gamma}(\boldsymbol{z}_f(t)) := \big(\boldsymbol{\Psi}_x(J^{r-1}\boldsymbol{y}(t)), \boldsymbol{\Psi}_u(J^r\boldsymbol{y}(t))\big) = (\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{32}$$

$\boldsymbol{\Gamma}$ is the map from the flat-jet latent space to the physical state–input space. Conversely, for every feasible trajectory $(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)) \in \mathcal{F}_x$ that remains in the flatness domain, $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{z}_f = J^r\boldsymbol{y}$ define a dynamically consistent latent trajectory satisfying

$$\boldsymbol{\Lambda}[\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)](t) := J^r[\boldsymbol{\phi}(\boldsymbol{x}(\cdot))](t) = \boldsymbol{z}_f(t) \tag{33}$$

Hence, $\boldsymbol{\Lambda}$ maps $\mathcal{F}_x$ into $\mathcal{F}_f$, while $\boldsymbol{\Gamma}$ maps $\mathcal{F}_f$ back to $\mathcal{F}_x$. The two feasible trajectory sets therefore correspond locally.

Let the original control optimization problem be

$$\mathcal{P}_x : \min_{(\boldsymbol{x},\boldsymbol{u})\in\mathcal{F}_x} \mathcal{J}_x(\boldsymbol{x}, \boldsymbol{u}), \tag{34}$$

where $\mathcal{F}_x$ denotes the set of trajectories satisfying the original dynamics, boundary conditions, and state–input constraints. The original boundary and state–input constraints are expressed in the latent space through the same reconstruction map $\boldsymbol{\Gamma}$. The resulting latent-space problem is

$$\mathcal{P}_f : \min_{\boldsymbol{z}_f(\cdot)\in\mathcal{F}_f} \mathcal{J}_f(\boldsymbol{z}_f(\cdot)), \tag{35}$$

where $\mathcal{F}_f$ is the set of dynamically consistent latent trajectories whose reconstructed state–input trajectories satisfy the original boundary and path constraints. For every $\boldsymbol{z}_f(\cdot) \in \mathcal{F}_f$ and its reconstructed trajectory $(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot))$, we have

$$\mathcal{J}_f(\boldsymbol{z}_f(\cdot)) = \mathcal{J}_x(\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)). \tag{36}$$

Therefore, under the exact flatness realization, the reconstruction map $\boldsymbol{\Gamma}$ and the induced flat-jet map $\boldsymbol{\Lambda}$ preserves the constraints and objective values. Consequently, the two control optimization problems have the same local optimal value, and a solution of either problem is mapped to a solution of the other. This proves the locally equivalent formulation stated in Corollary 1.

□

### A.5 Choice of the Finite-Difference Operator and Flat Jet Encoder

The operator $\boldsymbol{H}_{r,\Delta t}$ is a fixed linear map from a flat-output window to the current flat jet. For example, we use backward finite differences when the flat-jet state contains derivatives up to second order:

$$\widehat{\dot{\boldsymbol{y}}}_t = \frac{\boldsymbol{y}_t - \boldsymbol{y}_{t-1}}{\Delta t}, \qquad \widehat{\ddot{\boldsymbol{y}}}_t = \frac{\boldsymbol{y}_t - 2\boldsymbol{y}_{t-1} + \boldsymbol{y}_{t-2}}{\Delta t^2}. \tag{37}$$

The resulting flat jet can be written as the linear transformation

$$\underbrace{\begin{bmatrix} \boldsymbol{y}_t \\ \widehat{\dot{\boldsymbol{y}}}_t \\ \widehat{\ddot{\boldsymbol{y}}}_t \end{bmatrix}}_{\boldsymbol{\xi}_t} = \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & -1/\Delta t & 1/\Delta t \\ 1/\Delta t^2 & -2/\Delta t^2 & 1/\Delta t^2 \end{bmatrix} \otimes \boldsymbol{I}_{d_y}}_{\boldsymbol{H}_{3,\Delta t}} \underbrace{\begin{bmatrix} \boldsymbol{y}_{t-2} \\ \boldsymbol{y}_{t-1} \\ \boldsymbol{y}_t \end{bmatrix}}_{\boldsymbol{Y}_t}. \tag{38}$$

Higher-order flat jets are constructed analogously by adding the corresponding backward-difference rows to $\boldsymbol{H}_{r,\Delta t}$.

Our framework is not restricted to backward differences. When samples on both sides of the estimation time are available, such as in offline trajectory encoding, centered finite differences can be used to improve derivative accuracy. These alternatives can be incorporated into the same probabilistic formulation and training objective by replacing $\boldsymbol{H}_{r,\Delta t}$ with the corresponding linear operator. If future samples are used for more accurate jet construction during offline training but are unavailable during online control, the flat-jet state can instead be estimated causally by propagating the previous estimate through the learned structured dynamics and correcting it using the current encoded flat output which is detailed in Appendix A.8.

The explicit combination of $\boldsymbol{E}_y$ and $\boldsymbol{H}_{r,\Delta t}$ is not the only possible realization of the flat-jet encoder. In principle, it can be replaced by a generic sequence encoder that maps an observation history directly to the $r$ coordinate blocks of $\boldsymbol{\xi}_t$. Because the latent transition is restricted to the prescribed derivative-chain form, the dynamics-consistency loss $\mathcal{L}_{\mathrm{dyn}}$ encourages the encoder outputs to evolve as successive derivatives of a common latent coordinate. However, this alternative imposes the jet structure only through the training objective. In contrast, our flat-jet encoder constructs all derivative coordinates from a single learned output $\boldsymbol{E}_y(\boldsymbol{o})$ and therefore enforces their discrete derivative relations by the construction. It reduces the degrees of freedom of the inference model and provides a stronger structural bias toward the desired flat-jet representation, facilitating its identification from limited data.

### A.6 Probabilistic Transition of the Flatness-Structured Latent State

The learned highest-order dynamics are represented as

$$\boldsymbol{\nu}_t = \boldsymbol{\mu}_{\nu,t} + \boldsymbol{\varepsilon}_{\nu,t}, \qquad \boldsymbol{\varepsilon}_{\nu,t} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\nu,t}). \tag{39}$$

We assume that $\boldsymbol{\nu}_t$ is constant over the sampling interval $[t, t+\Delta t]$. Integrating the continuous-time derivative chain

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}_r \boldsymbol{\xi} + \boldsymbol{B}_r \boldsymbol{\nu}_t \tag{40}$$

over this interval gives

$$\boldsymbol{\xi}_{t+1} = e^{\boldsymbol{A}_r \Delta t} \boldsymbol{\xi}_t + \left( \int_0^{\Delta t} e^{\boldsymbol{A}_r (\Delta t - \tau)} \boldsymbol{B}_r \, d\tau \right) \boldsymbol{\nu}_t. \tag{41}$$

Defining

$$\boldsymbol{F}_\Delta := e^{\boldsymbol{A}_r \Delta t}, \qquad \boldsymbol{\Gamma}_\Delta := \int_0^{\Delta t} e^{\boldsymbol{A}_r (\Delta t - \tau)} \boldsymbol{B}_r \, d\tau, \tag{42}$$

the discrete transition becomes

$$\boldsymbol{\xi}_{t+1} = \boldsymbol{F}_\Delta \boldsymbol{\xi}_t + \boldsymbol{\Gamma}_\Delta \boldsymbol{\mu}_{\nu,t} + \boldsymbol{\Gamma}_\Delta \boldsymbol{\varepsilon}_{\nu,t}. \tag{43}$$

Since the highest-order uncertainty enters linearly, the induced latent-state transition is Gaussian:

$$p_\theta(\boldsymbol{\xi}_{t+1} \mid \boldsymbol{\xi}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{\mu}_{\xi,t+1}, \boldsymbol{Q}_{\Delta,t}), \tag{44}$$

with

$$\boldsymbol{\mu}_{\xi,t+1} = \boldsymbol{F}_\Delta \boldsymbol{\xi}_t + \boldsymbol{\Gamma}_\Delta \boldsymbol{\mu}_{\nu,t}, \qquad \boldsymbol{Q}_{\Delta,t} = \boldsymbol{\Gamma}_\Delta \boldsymbol{\Sigma}_{\nu,t} \boldsymbol{\Gamma}_\Delta^\top + \epsilon \boldsymbol{I}. \tag{45}$$

For the $r$-th order integrator chain,

$$\boldsymbol{\Gamma}_\Delta = \begin{bmatrix} \frac{\Delta t^r}{r!} \boldsymbol{I}_{d_y} \\ \frac{\Delta t^{r-1}}{(r-1)!} \boldsymbol{I}_{d_y} \\ \vdots \\ \frac{\Delta t^2}{2!} \boldsymbol{I}_{d_y} \\ \Delta t \, \boldsymbol{I}_{d_y} \end{bmatrix} = \begin{bmatrix} \frac{\Delta t^r}{r!} \\ \frac{\Delta t^{r-1}}{(r-1)!} \\ \vdots \\ \frac{\Delta t^2}{2!} \\ \Delta t \end{bmatrix} \otimes \boldsymbol{I}_{d_y}. \tag{46}$$

Thus, uncertainty in the highest-order derivative is propagated to all flat-jet coordinates with the corresponding integration factors. The term $\epsilon \boldsymbol{I}$ ensures a positive-definite covariance for the Gaussian transition and the subsequent KL divergence.

### A.7 Trajectory Planning in Learned Flat Coordinates

Given the initial and target observation histories, we obtain the boundary latent states $\boldsymbol{\xi}_0$ and $\boldsymbol{\xi}_T^d$ using the flat-jet encoder. Let $\boldsymbol{\xi}_0^{[k]}$ and $\boldsymbol{\xi}_T^{d,[k]}$ denote their $k$-th derivative blocks, respectively, for $k = 0, \dots, r-1$. For a prescribed transfer duration $T > 0$, we construct a polynomial trajectory in the learned flat-output space:

$$\boldsymbol{y}_d(t) = \sum_{j=0}^{2r-1} \boldsymbol{c}_j \left(\frac{t}{T}\right)^j, \qquad t \in [0, T], \tag{47}$$

where $\boldsymbol{c}_j \in \mathbb{R}^{d_y}$ are determined by the boundary conditions

$$\boldsymbol{y}_d^{(k)}(0) = \boldsymbol{\xi}_0^{[k]}, \qquad \boldsymbol{y}_d^{(k)}(T) = \boldsymbol{\xi}_T^{d,[k]}, \qquad k = 0, \dots, r-1. \tag{48}$$

Substituting Eq. 47 into these conditions gives

$$\frac{k!}{T^k}\boldsymbol{c}_k = \boldsymbol{\xi}_0^{[k]}, \qquad \frac{1}{T^k}\sum_{j=k}^{2r-1} \frac{j!}{(j-k)!}\boldsymbol{c}_j = \boldsymbol{\xi}_T^{d,[k]}. \tag{49}$$

These linear equations uniquely determine the polynomial coefficients for each flat-output component.

The reference latent state and feedforward latent input are obtained analytically:

$$\boldsymbol{\xi}_d(t) = J^{r-1}\boldsymbol{y}_d(t), \qquad \boldsymbol{\nu}_d(t) = \boldsymbol{y}_d^{(r)}(t). \tag{50}$$

Thus, the reference satisfies the prescribed derivative chain by construction. During execution, these quantities are evaluated at each sampling instant and used in Eq. 10.

Planning requires solving for $2rd_y$ coefficients once per transfer, independently of the number of execution steps. Subsequent control updates evaluate the reference and apply latent feedback and input recovery, without repeated input-sequence optimization.

**Planning with objectives and constraints.** When additional objectives or path constraints are considered, the same parameterization reduces trajectory planning to optimization over polynomial coefficients. We use $\boldsymbol{y}_{\boldsymbol{c}}(t) = \sum_{j=0}^{p} \boldsymbol{c}_j (t/T)^j$ with $p > 2r-1$ to retain free coefficients after imposing the boundary conditions. Writing $\boldsymbol{\xi}_{\boldsymbol{c}} = J^{r-1}\boldsymbol{y}_{\boldsymbol{c}}$ and $\boldsymbol{\nu}_{\boldsymbol{c}} = \boldsymbol{y}_{\boldsymbol{c}}^{(r)}$, the planning problem becomes

$$\begin{aligned} \min_{\boldsymbol{c}} \quad & \mathcal{J}_f\big(J^r \boldsymbol{y}_{\boldsymbol{c}}\big) \\ \text{s.t.} \quad & \boldsymbol{\xi}_{\boldsymbol{c}}(0) = \boldsymbol{\xi}_0, \qquad \boldsymbol{\xi}_{\boldsymbol{c}}(T) = \boldsymbol{\xi}_T^d, \\ & \boldsymbol{g}_f\big(\boldsymbol{\xi}_{\boldsymbol{c}}(t), \boldsymbol{\nu}_{\boldsymbol{c}}(t)\big) \le \boldsymbol{0}, \qquad t \in [0, T], \end{aligned} \tag{51}$$

where $\mathcal{J}_f$ may encode trajectory smoothness, input effort, or reference-tracking objectives. By Corollary 1, under an exact flatness realization, objectives and constraints defined in the original state–input space can be equivalently expressed in the learned flat coordinates through the local reconstruction maps. Thus, polynomial coefficients can be optimized directly in the learned flat-output space while retaining the original objectives and constraints within the considered flatness domain.

### A.8 Latent-State Estimation for Feedback Control

Feedback control requires an estimate of the current flat-jet state $\boldsymbol{\xi}_t$. We consider two estimators: direct estimation from a rolling observation history and recursive estimation using the learned structured latent dynamics.

**Direct estimation from observation histories.** The most direct approach applies the flat-jet encoder used during training to the latest observation window $\boldsymbol{o}_{t-K:t}$. The shared encoder first extracts the flat output from each observation,

$$\widehat{\boldsymbol{y}}_{t-j} = \boldsymbol{\mu}_{y,t-j} = E_y^{\mu}(\boldsymbol{o}_{t-j}), \qquad j = 0, \dots, K, \tag{52}$$

and the current latent state is then obtained by applying the finite-difference operator:

$$\widehat{\boldsymbol{\xi}}_t^{\mathrm{dir}} = \boldsymbol{H}_{r,\Delta t}\,\mathrm{col}\left(\widehat{\boldsymbol{y}}_{t-K},\ldots,\widehat{\boldsymbol{y}}_t\right). \tag{53}$$

This estimator is simple and requires no additional model beyond the flat-jet encoder. However, finite differences can amplify observation and encoding noise, particularly in the higher-order derivative coordinates. Such noise may be further amplified when $\widehat{\boldsymbol{\xi}}_t^{\mathrm{dir}}$ is directly used for feedback control.

**Model-based latent-state observer.** A smoother causal estimate can be obtained by combining the learned latent dynamics with the current encoded flat output. Given the previous estimate $\widehat{\boldsymbol{\xi}}_{t-1}$ and the applied input $\boldsymbol{u}_{t-1}$, we first predict

$$\widehat{\boldsymbol{\nu}}_{t-1} = \boldsymbol{\mu}_\nu\left(\widehat{\boldsymbol{\xi}}_{t-1}, \boldsymbol{u}_{t-1}\right), \tag{54}$$

$$\widehat{\boldsymbol{\xi}}_{t|t-1} = \boldsymbol{F}_\Delta\widehat{\boldsymbol{\xi}}_{t-1} + \boldsymbol{\Gamma}_\Delta\widehat{\boldsymbol{\nu}}_{t-1}. \tag{55}$$

The current observation provides a direct measurement of the first block of the flat jet,

$$\widehat{\boldsymbol{y}}_t = E_y^\mu(\boldsymbol{o}_t), \qquad \boldsymbol{C}\boldsymbol{\xi}_t = \boldsymbol{y}_t, \qquad \boldsymbol{C} = \begin{bmatrix}\boldsymbol{I}_{d_y} & \boldsymbol{0} & \cdots & \boldsymbol{0}\end{bmatrix}. \tag{56}$$

A fixed-gain observer then corrects the predicted flat jet according to

$$\widehat{\boldsymbol{\xi}}_t = \widehat{\boldsymbol{\xi}}_{t|t-1} + \boldsymbol{L}\left(\widehat{\boldsymbol{y}}_t - \boldsymbol{C}\widehat{\boldsymbol{\xi}}_{t|t-1}\right), \tag{57}$$

where $\boldsymbol{L}$ is a fixed observer gain. Eq. 57 corrects not only the flat-output coordinate but also its derivative coordinates. Since the prescribed derivative chain is observable from its first coordinate, $\boldsymbol{L}$ can be selected by standard observer design or tuned for the sampling interval and observation noise. The observer may be initialized using the direct estimate in Eq. 53.

The probabilistic model also supports covariance-based estimators. In particular, the fixed gain $\boldsymbol{L}$ can be replaced by a Kalman-style gain computed from the propagated latent-dynamics uncertainty and the uncertainty produced by $\boldsymbol{E}_y$. We use the fixed-gain form above as a simple realization of model-based latent state estimation.

### A.9 Auxiliary Input-Recovery Network

We use an auxiliary network $R_u(\boldsymbol{\xi}, \boldsymbol{\nu})$ to recover the physical input. After training the main model, we freeze all its components and construct supervised training tuples

$$\mathcal{D}_R = \{(\boldsymbol{\mu}_{\boldsymbol{\xi},t}, \boldsymbol{\mu}_{\boldsymbol{\nu},t}, \boldsymbol{u}_t)\}, \qquad \boldsymbol{\mu}_{\boldsymbol{\nu},t} = \boldsymbol{G}_\theta\left(\boldsymbol{\mu}_{\boldsymbol{\xi},t}, \boldsymbol{u}_t\right), \tag{58}$$

where $\boldsymbol{\mu}_{\boldsymbol{\xi},t}$ is obtained from Eq. 7. The recovery network is trained by supervised regression:

$$\mathcal{L}_R(\omega) = \frac{1}{|\mathcal{D}_R|}\sum_t \left\|R_\omega\left(\boldsymbol{\mu}_{\boldsymbol{\xi},t}, \boldsymbol{\mu}_{\boldsymbol{\nu},t}\right) - \boldsymbol{u}_t\right\|_2^2. \tag{59}$$

Training it after freezing the main model prevents the auxiliary objective from altering the learned flat representation.

By Proposition 1, differential flatness guarantees the existence of a local input-recovery map $\boldsymbol{u} = \boldsymbol{\Psi}_u(\boldsymbol{\xi}, \boldsymbol{\nu})$ on the considered domain. The auxiliary network $R_\omega$ provides a data-driven approximation of this map in the learned flat coordinates.

### A.10 Flat-Coordinate Alignment and Evaluation

Flat outputs are nonunique: an invertible affine transformation preserves their flatness. To compare learned and analytical flat coordinates, we fit an affine alignment using only the flat-output values on training trajectories:

$$(\boldsymbol{A}^\star, \boldsymbol{b}^\star) \in \arg\min_{\boldsymbol{A},\boldsymbol{b}} \sum_{t\in\mathcal{I}_{\mathrm{train}}} \left\|\boldsymbol{A}\widehat{\boldsymbol{y}}_t + \boldsymbol{b} - \boldsymbol{y}_t\right\|_2^2, \tag{60}$$

where $\widehat{\boldsymbol{y}}_t = E_y^\mu(\boldsymbol{o}_t)$ is the learned flat output, $\boldsymbol{y}_t$ is its analytical counterpart, and $\mathcal{I}_{\mathrm{train}}$ indexes samples from training trajectories. For a scalar flat output, $\boldsymbol{A}$ and $\boldsymbol{b}$ reduce to a scale and an offset.

The fitted transformation is fixed for evaluation and applied consistently to all derivative orders:

$$\widetilde{\boldsymbol{y}}_t = \boldsymbol{A}^\star \widehat{\boldsymbol{y}}_t + \boldsymbol{b}^\star, \qquad \widetilde{\boldsymbol{y}}_t^{(k)} = \boldsymbol{A}^\star \widehat{\boldsymbol{y}}_t^{(k)}, \quad k = 1, \dots, r-1. \tag{61}$$

Here, $\widehat{\boldsymbol{y}}_t^{(k)}$ denotes the corresponding learned jet component. The offset affects only the flat output, while the same linear transformation applies to every derivative order. No derivative-specific alignment or test-set fitting is performed. For the ablations, the same transformation is applied to the corresponding latent blocks.

We evaluate agreement on unseen test trajectories using $R^2$. For derivative order $k$ and coordinate $i$,

$$R^2_{k,i} = 1 - \frac{\sum_{t \in \mathcal{I}_{\text{test}}} \left( \widetilde{y}^{(k)}_{t,i} - y^{(k)}_{t,i} \right)^2}{\sum_{t \in \mathcal{I}_{\text{test}}} \left( y^{(k)}_{t,i} - \bar{y}^{(k)}_i \right)^2}, \qquad R^2_k = \frac{1}{d_y} \sum_{i=1}^{d_y} R^2_{k,i}, \tag{62}$$

where $\mathcal{I}_{\text{test}}$ indexes test samples, $\bar{y}^{(k)}_i$ is their analytical mean, and $k = 0$ denotes the flat output itself. Scores are reported separately for each derivative order.

Alignment is used only for visualization and evaluation; model training, planning, and control use the original learned coordinates. Consequently, agreement across derivative orders measures consistency of the learned jet under a single output-level alignment.

## A.11 Experimental Setup

### A.11.1 System Dynamics and Analytical Flat Coordinates

**Three-dimensional nonlinear system.** We consider the nonlinear system

$$\begin{aligned} \dot{x}_1 &= x_3 - x_2 u, \\ \dot{x}_2 &= -x_2 + u, \\ \dot{x}_3 &= x_2 - x_1 + 2x_2(u - x_2), \end{aligned} \tag{63}$$

with state $\boldsymbol{x} = (x_1, x_2, x_3)^\top$ and scalar input $u$. The system admits the flat output $y = x_1 + \frac{1}{2}x_2^2$. Differentiating along the dynamics gives

$$\dot{y} = x_3 - x_2^2, \qquad \ddot{y} = x_2 - x_1, \qquad y^{(3)} = (1 + x_2)u - x_2 - x_3. \tag{64}$$

The Jacobian of the state-to-jet map satisfies

$$\det \left( \frac{\partial(y, \dot{y}, \ddot{y})}{\partial(x_1, x_2, x_3)} \right) = -(1 + x_2), \tag{65}$$

so the map is locally invertible for $x_2 \neq -1$.

To recover the state, observe that $1 + 2(y + \ddot{y}) = (1 + x_2)^2$. On a fixed branch $\sigma = \operatorname{sign}(1 + x_2) \in \{-1, 1\}$, the reconstruction is

$$x_2 = -1 + \sigma\sqrt{1 + 2(y + \ddot{y})}, \qquad x_1 = y - \frac{1}{2}x_2^2, \qquad x_3 = \dot{y} + x_2^2. \tag{66}$$

The input is then recovered as

$$u = \frac{y^{(3)} + x_2 + x_3}{1 + x_2}. \tag{67}$$

Thus, this system has the consecutive-order parameterization with $r = 3$: $\boldsymbol{\xi} = (y, \dot{y}, \ddot{y})^\top$ and $\nu = y^{(3)}$.

We use two observation maps,

$$\boldsymbol{o}^{(1)} = (x_1, x_2, x_3)^\top, \qquad \boldsymbol{o}^{(2)} = (x_1, x_2, x_3^2)^\top. \tag{68}$$

Both determine the flat output through $y = o_1 + \frac{1}{2}o_2^2$. Although $\boldsymbol{o}^{(2)}$ does not distinguish the sign of $x_3$ in a single observation, the flat-output jet recovers $x_3$ through Eq. 66. The analytical coordinates are used for evaluation, not as supervision for representation learning.

**Planar bicopter.** The bicopter has state $\boldsymbol{s} = (x, z, \dot{x}, \dot{z}, T, \theta)^\top$ and input $\boldsymbol{u} = (T_{\mathrm{cmd}}, \omega)^\top$, where $T$ is the total thrust, $\theta$ is the pitch angle, and $\omega$ is the commanded angular velocity. Its dynamics are

$$\ddot{x} = -\frac{T}{m}\sin\theta, \qquad \ddot{z} = \frac{T}{m}\cos\theta - g, \qquad \dot{T} = \frac{T_{\mathrm{cmd}} - T}{\tau}, \qquad \dot{\theta} = \omega, \tag{69}$$

where $m$ is the mass, $g$ is gravitational acceleration, and $\tau > 0$ is the thrust-response time constant.

We choose the planar position as the flat output, $\boldsymbol{y} = (x, z)^\top$. Define

$$\boldsymbol{a} = \begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 + g \end{bmatrix}, \qquad \dot{\boldsymbol{a}} = \begin{bmatrix} y_1^{(3)} \\ y_2^{(3)} \end{bmatrix}, \qquad \rho = \|\boldsymbol{a}\|_2. \tag{70}$$

On the domain $T > 0$, position and velocity are given directly by $\boldsymbol{y}$ and $\dot{\boldsymbol{y}}$, while thrust and pitch are recovered from

$$T = m\rho, \qquad \theta = \mathrm{atan2}(-a_1, a_2), \tag{71}$$

using a locally continuous angle branch. Differentiating these expressions gives

$$\dot{T} = m\frac{\boldsymbol{a}^\top \dot{\boldsymbol{a}}}{\rho}, \qquad \dot{\theta} = \frac{a_1\dot{a}_2 - a_2\dot{a}_1}{\rho^2}. \tag{72}$$

The inputs are therefore recovered as

$$T_{\mathrm{cmd}} = m\rho + \tau m\frac{\boldsymbol{a}^\top \dot{\boldsymbol{a}}}{\rho}, \qquad \omega = \frac{a_1\dot{a}_2 - a_2\dot{a}_1}{\rho^2}. \tag{73}$$

Thus, the full state is determined by $J^2\boldsymbol{y}$ and the input by $J^3\boldsymbol{y}$, giving $r = 3$, $\boldsymbol{\xi} = J^2\boldsymbol{y}$, and $\boldsymbol{\nu} = \boldsymbol{y}^{(3)}$.

The rendered image determines the position $\boldsymbol{y} = (x, z)^\top$, satisfying the flat-output observation condition. Velocities and the internal thrust state need not be directly observed.

**Planar overhead crane.** We consider a planar overhead crane with trolley mass $M$, payload mass $m$, and a massless taut rope of variable length $L$. The trolley position is $x_c$, and the swing angle $\theta$ is measured from the downward vertical. We augment the mechanical state with the rope tension $H$ and its rate $V_H$, governed by a second-order actuator model. The resulting state and input are

$$\boldsymbol{s} = (x_c, L, \theta, \dot{x}_c, \dot{L}, \dot{\theta}, H, V_H)^\top, \qquad \boldsymbol{u} = (F, H_{\mathrm{cmd}})^\top. \tag{74}$$

Neglecting friction, the dynamics are

$$\begin{aligned} \ddot{x}_c &= \frac{F + H\sin\theta}{M}, \\ \ddot{L} &= L\dot{\theta}^2 + g\cos\theta - \frac{H}{m} - \ddot{x}_c\sin\theta, \\ \ddot{\theta} &= -\frac{2\dot{L}\dot{\theta} + g\sin\theta + \ddot{x}_c\cos\theta}{L}, \\ \dot{H} &= V_H, \qquad \dot{V}_H = \omega_H^2(H_{\mathrm{cmd}} - H) - 2\zeta_H\omega_H V_H, \end{aligned} \tag{75}$$

where $\omega_H > 0$ and $\zeta_H > 0$ are the natural frequency and damping ratio of the tension actuator.

The flat output is the payload position, with its vertical coordinate measured downward:

$$\boldsymbol{y} = \begin{bmatrix} x_c + L\sin\theta \\ L\cos\theta \end{bmatrix}. \tag{76}$$

Define the rope direction and its angular derivative as $\boldsymbol{e} = (\sin\theta, \cos\theta)^\top$ and $\boldsymbol{e}_\perp = (\cos\theta, -\sin\theta)^\top$. The payload dynamics give

$$\ddot{\boldsymbol{y}} = \begin{bmatrix} 0 \\ g \end{bmatrix} - \frac{H}{m}\boldsymbol{e}, \qquad \boldsymbol{y}^{(3)} = -\frac{V_H}{m}\boldsymbol{e} - \frac{H\dot{\theta}}{m}\boldsymbol{e}_\perp. \tag{77}$$

To reconstruct the state, let $\boldsymbol{a} = (0, g)^\top - \ddot{\boldsymbol{y}}$ and $\rho = \|\boldsymbol{a}\|_2$. On the domain $L > 0$, $H > 0$, and $\cos\theta > 0$,

$$H = m\rho, \qquad \theta = \operatorname{atan2}(a_1, a_2), \qquad L = \frac{y_2}{\cos\theta}, \qquad x_c = y_1 - L\sin\theta. \tag{78}$$

The remaining state components follow from

$$\begin{aligned} V_H &= -m\boldsymbol{e}^\top \boldsymbol{y}^{(3)}, & \dot{\theta} &= -\frac{m}{H}\boldsymbol{e}_\perp^\top \boldsymbol{y}^{(3)}, \\ \dot{L} &= \frac{\dot{y}_2 + L\sin\theta\,\dot{\theta}}{\cos\theta}, & \dot{x}_c &= \dot{y}_1 - \dot{L}\sin\theta - L\cos\theta\,\dot{\theta}. \end{aligned} \tag{79}$$

Hence, all eight state components are determined by $J^3\boldsymbol{y}$.

Differentiating once more yields

$$\boldsymbol{y}^{(4)} = \frac{H\dot{\theta}^2 - \ddot{H}}{m}\boldsymbol{e} - \frac{2V_H\dot{\theta} + H\ddot{\theta}}{m}\boldsymbol{e}_\perp. \tag{80}$$

For $\boldsymbol{\nu} = \boldsymbol{y}^{(4)}$, the required tension and angular accelerations are

$$\ddot{H} = H\dot{\theta}^2 - m\boldsymbol{e}^\top\boldsymbol{\nu}, \qquad \ddot{\theta} = -\frac{m\boldsymbol{e}_\perp^\top\boldsymbol{\nu} + 2V_H\dot{\theta}}{H}. \tag{81}$$

Substitution into Eq. 75 then gives the input reconstruction:

$$\begin{aligned} F &= -\frac{M}{\cos\theta}\left(L\ddot{\theta} + 2\dot{L}\dot{\theta} + g\sin\theta\right) - H\sin\theta, \\ H_{\mathrm{cmd}} &= H + \frac{\ddot{H} + 2\zeta_H\omega_H V_H}{\omega_H^2}. \end{aligned} \tag{82}$$

The augmented system therefore has $r = 4$, with $\boldsymbol{\xi} = J^3\boldsymbol{y}$ and $\boldsymbol{\nu} = \boldsymbol{y}^{(4)}$. The rendered payload position determines the flat output, while velocities and the internal actuator states $(H, V_H)$ need not be directly observed.

### A.11.2 Data Generation

**Three-dimensional nonlinear system.** We generate trajectories with initial states sampled uniformly from $[-2, 2]^3$ using two input-generation schemes. The first linearly interpolates six evenly spaced knots with values sampled uniformly from $[-1.2, 1.2]$. The second uses ten piecewise-constant segments with values sampled uniformly from $[-2.5, 2.5]$, with a randomly located three-step interval replaced by a pulse value sampled uniformly from $[-3.75, 3.75]$. The system is simulated using an adaptive RK4 solver, and each trajectory contains 100 samples at an interval of $\Delta t = 0.05\,\mathrm{s}$. Observations are constructed using the two observation maps defined in the main text, without added noise. We use 15%, 30%, or 60% of the generated 100 trajectories for training and reserve the remaining trajectories for evaluation, with splitting performed at the trajectory level.

**Planar bicopter.** We generate 100 trajectories using a feedback controller that tracks randomized sinusoidal position references, with amplitudes sampled from $[0.5, 0.8]\,\mathrm{m}$ and frequencies from $[0.08, 0.24]\,\mathrm{Hz}$. An additional sinusoidal pitch reference has amplitude 12°–20° and frequency 0.16–0.22 Hz. To enrich the excitation, each input is perturbed by a sum of three sinusoids with randomized amplitudes and phases and frequencies in $[0.3, 0.9]\,\mathrm{Hz}$.

Initial positions and velocities are sampled uniformly from $[-0.6, 0.6]\,\mathrm{m}$ and $[-0.8, 0.8]\,\mathrm{m/s}$ per coordinate, respectively. The initial pitch lies in $[-12°, 12°]$, and the initial thrust lies in $[0.75mg, 1.25mg]$. Inputs are clipped to $T_{\mathrm{cmd}} \in [0.6mg, 1.6mg]$ and $\omega \in [-4.5, 4.5]\,\mathrm{rad/s}$. We use $m = 1\,\mathrm{kg}$, $g = 9.8\,\mathrm{m/s^2}$, and $\tau = 0.1\,\mathrm{s}$. Each trajectory is simulated for $5\,\mathrm{s}$ using RK4 and sampled at $\Delta t = 0.05\,\mathrm{s}$, yielding 101 observations and 100 sampling intervals. Image sequences are rendered from the simulated position and pitch against a plain background, without added noise.

Images are rendered at $960 \times 960$ pixels on a white background, without axes or grid lines. The bicopter is depicted as a dark rigid body with two red rotor outlines, with body half-span $0.45\,\mathrm{m}$ and rotor radius $0.14\,\mathrm{m}$. Its position and orientation follow the simulated $(x, z, \theta)$, while rotor blades remain fixed relative to the body. Each trajectory uses a fixed square viewing window, automatically centered and scaled to its position range, with equal horizontal and vertical scales. Frames are rendered at 20 Hz, matching the simulation sampling rate, without added image noise.

**Planar overhead crane.** We generate 100 trajectories using feedback tracking with independent multisine excitation on the trolley force and tension command. The trolley reference combines two cosine components at $0.2$ and $0.4\,\mathrm{Hz}$, with randomized phases and amplitude magnitudes in $[0.72, 0.88]\,\mathrm{m}$ and $[0.12, 0.20]\,\mathrm{m}$, respectively. The rope-length reference follows a smooth quintic transfer from $[1.68, 1.84]\,\mathrm{m}$ to $[0.70, 0.86]\,\mathrm{m}$. Three sinusoidal components are added to each input, with frequencies spanning $0.32$–$1.28\,\mathrm{Hz}$ for force and $0.40$–$1.42\,\mathrm{Hz}$ for tension. Initial trolley and rope positions and velocities match their references, while the initial swing angle and angular velocity are randomized. The tension actuator is initialized at $H = mg$ and $V_H = 0$.

We use $M = 1\,\mathrm{kg}$, $m = 0.25\,\mathrm{kg}$, $g = 9.8\,\mathrm{m/s^2}$, $\omega_H = 14\,\mathrm{rad/s}$, and $\zeta_H = 0.9$. Inputs are clipped to $F \in [-5, 5]\,\mathrm{N}$ and $H_{\mathrm{cmd}} \in [0.45, 5.8]\,\mathrm{N}$. Each trajectory lasts $5\,\mathrm{s}$ and is integrated using RK4 with a step size of $0.005\,\mathrm{s}$. Inputs are updated and held constant over each $0.05\,\mathrm{s}$ sampling interval, yielding 101 observations and 100 transitions per trajectory. Candidate trajectories are resampled if they violate state or tension bounds, exhibit insufficient motion or input excitation, or exceed the saturation threshold.

Images are rendered at $960 \times 960$ pixels and $20\,\mathrm{Hz}$ using a fixed viewing window shared across trajectories, spanning $[-3, 3]\,\mathrm{m}$ horizontally and $[-2.9, 2.9]\,\mathrm{m}$ vertically. The scene contains a horizontal rail, a gray trolley, a dark rope, and a circular payload on a white background, without text, axes, or added image noise. The trolley measures $0.68 \times 0.26\,\mathrm{m}$, and the payload radius is $0.085\,\mathrm{m}$.

### A.11.3 Network Components

**Three-dimensional nonlinear system.** The flat-output mapping, highest-order dynamics, and input-recovery map are implemented as MLPs, each with two hidden layers of 64 units, Tanh activations, and a linear output layer. The flat-output encoder maps the three-dimensional observation to a scalar output. With $r = 3$, the latent state $\boldsymbol{\xi} = (y, \dot{y}, \ddot{y})^\top$ has dimension three. The flat-jet encoder uses a five-observation history $\boldsymbol{o}_{t-4:t}$. The shared MLP encodes each observation into a scalar flat output, and the backward finite-difference operator $\mathcal{H}_{3,\Delta t}$ constructs $\boldsymbol{\xi}_t = (y_t, \dot{y}_t, \ddot{y}_t)^\top$ from the resulting output sequence, with $\Delta t = 0.05\,\mathrm{s}$. The highest-order dynamics network takes $(\boldsymbol{\xi}, u)$ as input and predicts the scalar $\nu = y^{(3)}$, while the recovery network maps $(\boldsymbol{\xi}, \nu)$ to the physical input $u$. Their architectures are summarized below:

$$\begin{aligned} E_y &: \quad 3 \to 64 \to 64 \to 1, \\ G_\theta &: \quad 4 \to 64 \to 64 \to 1, \\ R_u &: \quad 4 \to 64 \to 64 \to 1. \end{aligned} \tag{83}$$

The input-recovery network is trained separately after freezing the learned encoder and dynamics model.

**Planar bicopter.** The image-to-flat-output mapping uses a CNN followed by an MLP to produce a two-dimensional flat output. The CNN contains three convolutional layers with 32, 64, and 128 channels, respectively, each using a $3 \times 3$ kernel, stride 2, padding 1, and ReLU activation. Adaptive average pooling produces a $4 \times 4$ feature map per channel, which is flattened and passed through an MLP with architecture $2048 \to 128 \to 2$, with ReLU activation in the hidden layer and a linear output layer.

The shared encoder is applied independently to each image in the seven-frame history $\boldsymbol{o}_{t-6:t}$. For each output coordinate, a cubic polynomial is fitted by least squares to the seven encoded values at their sampling times. Its first and second derivatives are evaluated at the current time to estimate $\dot{\boldsymbol{y}}_t$ and $\ddot{\boldsymbol{y}}_t$, while $\boldsymbol{y}_t$ is taken directly from the current encoded image. This defines a causal linear jet-construction operator with $\Delta t = 0.05\,\mathrm{s}$.

With $r = 3$, the resulting latent state has dimension six. The highest-order dynamics and input-recovery networks are MLPs with architectures

$$\begin{aligned} G_\theta &: \quad 8 \to 128 \to 128 \to 2, \\ R_u &: \quad 8 \to 128 \to 128 \to 2. \end{aligned} \tag{84}$$

Both use Tanh activations in the hidden layers and linear output layers. Their inputs are $(\boldsymbol{\xi}, \boldsymbol{u})$ and $(\boldsymbol{\xi}, \boldsymbol{\nu})$, respectively. The recovery network is trained after freezing the encoder and dynamics model.

**Planar overhead crane.** The crane uses the same CNN+MLP image encoder as the bicopter, producing a two-dimensional flat output. The shared encoder processes a seven-frame history $\boldsymbol{o}_{t-6:t}$ with $\Delta t = 0.05\,\mathrm{s}$. For each output coordinate, a quartic polynomial is fitted by least squares to the encoded sequence. Its first three derivatives are evaluated at the current time, while the zeroth-order coordinate is taken directly from the current encoded image. This constructs the eight-dimensional latent state $\boldsymbol{\xi} = J^3\boldsymbol{y}$ for $r = 4$.

The highest-order dynamics and input-recovery networks use the architectures

$$\begin{aligned} G_\theta :&\quad 10 \to 128 \to 128 \to 2, \\ R_u :&\quad 10 \to 128 \to 128 \to 2. \end{aligned} \tag{85}$$

Both use Tanh activations in the hidden layers and linear output layers. The dynamics network maps $(\boldsymbol{\xi}, \boldsymbol{u})$ to $\boldsymbol{\nu} = \boldsymbol{y}^{(4)}$, and the recovery network maps $(\boldsymbol{\xi}, \boldsymbol{\nu})$ to $\boldsymbol{u} = (F, H_{\mathrm{cmd}})^\top$. The recovery network is trained after freezing the encoder and dynamics model.

### A.11.4 MPC AND FEEDBACK FREQUENCIES

All methods receive observations and update their inputs at $20\,\mathrm{Hz}$, corresponding to $\Delta t = 0.05\,\mathrm{s}$. For MPC-based methods, including Ours (MPC), we use a prediction horizon of $H = 20$ steps ($1\,\mathrm{s}$). At each update, the controller optimizes the input sequence, applies its first input, and replans using the latest observation history. The optimization is warm-started by shifting the previous solution.

For our flatness-based feedback controller, the reference trajectory is planned once at the beginning of each transfer. The latent-state estimate, reference derivatives, feedback correction, and recovered input are updated at $20\,\mathrm{Hz}$. The integral term is accumulated using the same sampling interval. Applied inputs are held constant between updates. Thus, MPC replanning and latent feedback use the same observation and input-update frequencies.Since some control computations exceed the sampling interval, we pause the simulator until each computation finishes, then apply the input and advance the simulation.

## A.12 ADDITIONAL QUANTITATIVE COMPARISONS ON IMAGE-BASED CONTROL

For both systems, prediction is evaluated over 16 future frames given an initial observation history and the corresponding input sequence, with a sampling interval of 0.05s. Control tasks specify fixed-duration transfers to a desired terminal image window, defining the target load position and velocity for the crane and analogous terminal conditions for the bicopter. Terminal errors are evaluated against the corresponding physical states, which are used only for evaluation. Prediction scores for PC3 are marked as N/A because its original formulation does not include an image decoder.

Table 4: **Prediction and control performance from images.** Percentages indicate training data usage. Pred.: image prediction SSIM on unseen trajectories; Term.: mean terminal error over waypoint transfers; Time: average computation time per control step.

| System | Method | 10% | | 30% | | 60% | | Time (ms) |
|---|---|---|---|---|---|---|---|---|
| | | Pred.↑ | Term.↓ | Pred.↑ | Term.↓ | Pred.↑ | Term.↓ | |
| Bicopter | E2C | 0.968 | 0.434 | 0.978 | 0.273 | 0.983 | 0.212 | $> 200$ |
| | PCC | 0.969 | 0.397 | 0.979 | 0.229 | 0.984 | 0.176 | $> 200$ |
| | PC3 | N/A | 0.429 | N/A | 0.171 | N/A | 0.103 | $> 200$ |
| | 16-DKP | 0.967 | 0.477 | 0.980 | 0.191 | 0.984 | 0.132 | $> 200$ |
| | Ours (MPC) | 0.985 | 0.089 | 0.988 | 0.074 | 0.988 | 0.068 | 20.26 |
| | Ours (feedback) | 0.985 | 0.073 | 0.988 | 0.070 | 0.988 | 0.069 | 130.54 |
| Crane | E2C | 0.934 | 0.621 | 0.956 | 0.358 | 0.981 | 0.157 | $> 200$ |
| | PCC | 0.941 | 0.588 | 0.957 | 0.342 | 0.983 | 0.149 | $> 200$ |
| | PC3 | N/A | 0.574 | N/A | 0.370 | N/A | 0.153 | $> 200$ |
| | 16-DKP | 0.940 | 0.530 | 0.952 | 0.331 | 0.984 | 0.166 | $> 200$ |
| | Ours (MPC) | 0.975 | 0.120 | 0.978 | 0.095 | 0.984 | 0.091 | 23.61 |
| | Ours (feedback) | 0.975 | 0.101 | 0.978 | 0.094 | 0.984 | 0.092 | 147.45 |